Multivariate Statistical Methods - Assignment 1

Milda Poceviciute, Henrik Karlsson, Ugurcan Lacin and Ramon Laborda

4 December 2017

Question 1

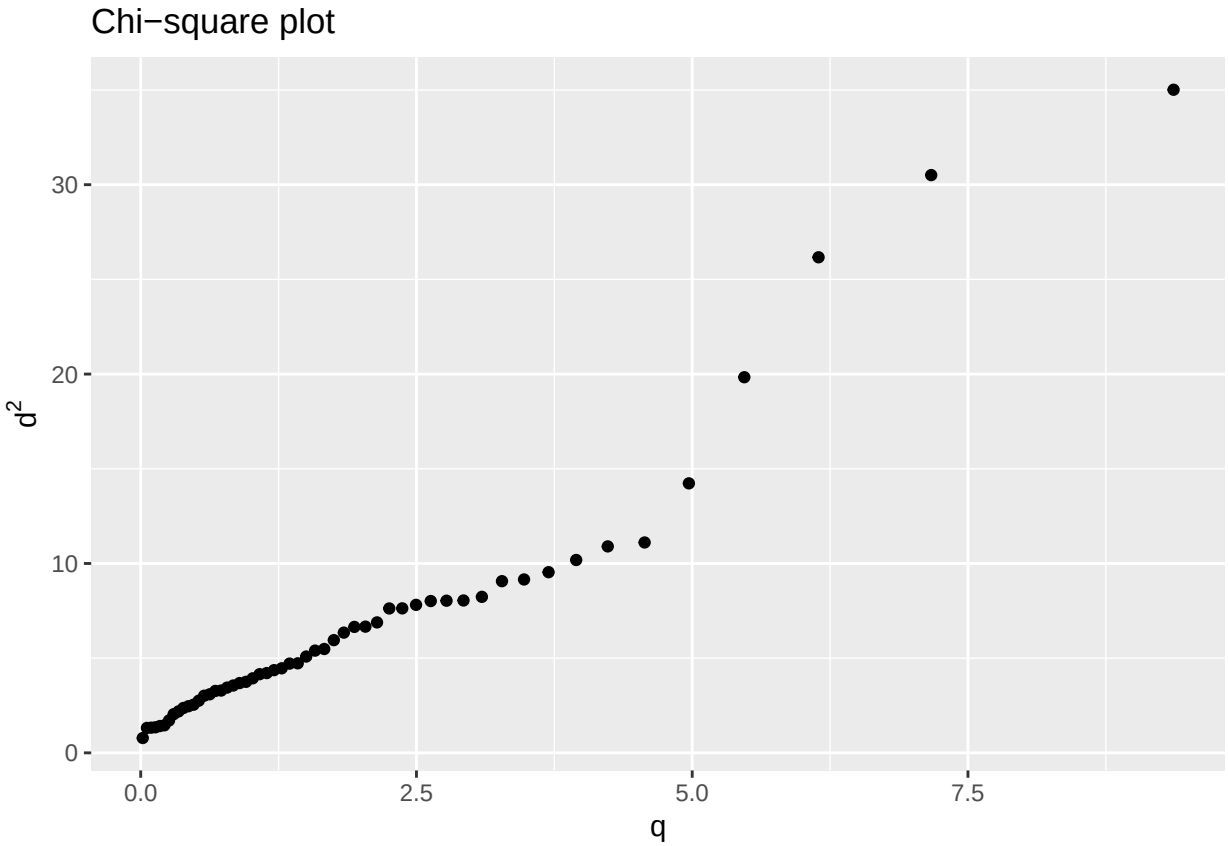
a)

Import data and calculate the Mahalanobis distance:

```
df <- read.csv("./T1-9.dat", sep = "\t", header = FALSE)
names(df) <- c("country", "m100", "m200", "m400", "m800", "m1500", "m3000", "marathon")

X <- as.matrix(df[,2:8])
X_avg <- colMeans(X)
resX_avg <- t(t(X)-X_avg)

# Compute Mahalanobis distance per row so there is a number per country
D_ma <- resX_avg %*% solve(cov(X)) %*% t(resX_avg)
dist <- diag(D_ma)

ind <- which(dist %in% sort(dist, decreasing = TRUE)[1:5])
countries3 <- df[ind,1]
```

We start off by constructing a chi-square plot of generalize distances to test bivariate normality. The data is bivariate normally distributed if the point form a straight line. If there is points with unusually large distances, these could be outliers and would be furthest away from the origo in the plot, but more formal test needs to be conducted in order to draw that conclusion.

```
orgi_order <- order(dist)
order_dist <- sort(dist)
len <- seq_along(order_dist)
chi_plot <- qchisq(p = ((len - .5)/length(order_dist)), df = 2)

df_plot <- data.frame(orgi_order, order_dist, chi_plot)

library(ggplot2)
ggplot(df_plot) +
    geom_point(aes(x = chi_plot, y = order_dist)) +
    #geom_abline(intercept = 0, slope = 1) +
    labs(title = "Chi-square plot", x = "q", y = expression(paste(d^{2})))
```



In order to test for each observation without multiple testing correction procedure (we take $\alpha = 0.001$ for each test). We compare the Mahalanobis distance to the chi square distribution value (where $\alpha = 0.001$). Outliers will have distance greater than the chi square value indicating that they are unlikely to come form this distribution:

```
# Chi square value with
c2 <- qchisq(.999, df=ncol(X))
result_df <- data.frame(countries = df[,1],test_diff = dist-c2)
outliers <- subset(result_df, result_df[,2]>0)
```

So after testing, we get that outliers are:

```
##    countries test_diff
## 31      KORN  1.845255
## 40       PNG  6.185361
## 46       SAM 10.692177
```

Now, one way of doing multiple testing correction is to use α divided by the number of tests. The reasoning is that as we sum up 54 tests, the total confidence is not $(1 - \alpha)$, but actually $(1 - \alpha)^{54}$. We can asses this problem by taking $\alpha/54$:

```
c3 <-qchisq(1 - (0.001/nrow(X)), df=ncol(X))
result2_df <- data.frame(countries = df[,1],test_diff = dist-c3)
outliers2 <- subset(result2_df, result2_df[,2]>0)
```

And therefore, we get that outliers are:

```
##    countries test_diff
## 46       SAM  1.182225
```

In the second test, our significance level was 54 times smaller than in the first test, so it's intuitive that

smaller amount of countries were considered outliers. In fact, only one country is selected as an outlier in the latest test. In general, accounting for uncertainty coming from 54 tests is useful as the most extreme values are classified as outliers. However, in this case testing at 0.001 significance level is very extreme because 0.001/54 goes close to zero. It is not reasonable to test at confidence level so close to 100% as then it is implied that the outcome is certain. At least in this case, the outcome is definiately not certain, so we would suggest to choose a larger α value.

b)

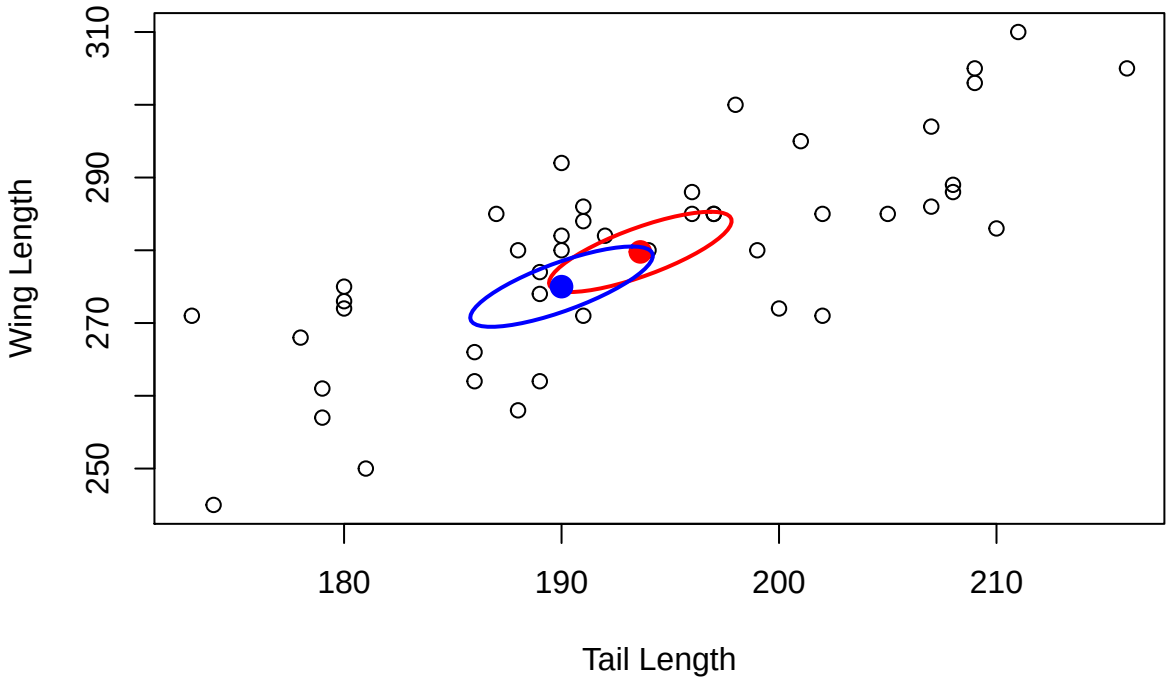
Euclidean distance does not take into account the covariances of the variables and only base the distance measure on the distance from the mean ignoring the covariances (and variances). Therefore, if covariances across the countries are very different, Euclidean distance does not capture it. In contrast, Mahalanobis distance takes into account the covariances of the variables which means that extremely high or low covariance does not distort the results. This could explain why North Korea is not included into outliers based on Euclidean distance, but it is included in the outliers based on the Mahalanobis distance.

Question 2

a)

Find and sketch the 95% confidence ellipse for the population means μ_1 and μ_2. Suppose it is known that μ_1 = 190mm and μ_2 = 275mm for male hook-billed kites. Are these plausible values for the mean tail length and mean wing length for female birds? Explain

```
data <- read.delim("T5-12.DAT", header=FALSE,sep="")

n <- nrow(data)

# Compute colMeans
dataMean <- (1/n) * t(data) %*% rep(1, n)
# cov matrix (2-27)
covm <- (1/(n-1)) * t(data) %*%
  (diag(n) - (1/n * matrix(rep(1, n), ncol = 1) %*% t(rep(1,n)))) %*%
  as.matrix(data)

#plot the data
plot(data, xlab = "Tail Length", ylab = "Wing Length")

#plot the ellipse for female (first) and male
radius <- sqrt(2 * (45-1) * qf(0.95, 2, 45-2)/(45*(45-2)))

library(car)
car::ellipse(center = as.vector(dataMean), shape = covm, radius = radius, col = 'red')
car::ellipse(center = c(190, 275), shape = covm, radius = radius, col = 'blue')
```



The red ellipse is the 95% confidence interval ellipse for the sample mean for female birds. The mean values for the male birds are represented by the blue dot and ellipse. Since the blue dot, the given population mean for males, lies within the red ellipse, it's plausible that the population means do not differ for male and females birds.

b)

Construct the simultaneous 95% T^2-intervals for μ_1 and μ_2 and the 95% Bonferroni intervals for μ_1 and μ_2. Compare the two sets of intervals. What advantage, if any, do the T^2-intervals have over the Bonferroni intervals?

```
#Confidence intervals using T^2 Hotellings
a <- matrix(c(1,0))
c1 <- t(a) %*% cov(data) %*% a
a2 <- matrix(c(0,1))
c2 <- t(a2) %*% cov(data) %*% a2

ci1A <- dataMean[1] - sqrt(((2*44) / (45-2) * 3.21 / 45 ) * c1)
ci1B <- dataMean[1] + sqrt(((2*44) / (45-2) * 3.21 / 45 ) * c1)
ci2A <- dataMean[2] - sqrt(((2*44) / (45-2) * 3.21 / 45 ) * c2)
ci2B <- dataMean[2] + sqrt(((2*44) / (45-2) * 3.21 / 45 ) * c2)
Hotellings <- list(TailLength =c(ci1A, ci1B) , WingLength=c(ci2A, ci2B))

# Confidence intervals using Bonferroni
Bf1A <- dataMean[1] - 2.32 * sqrt(cov(data)[1,1] / 45)
Bf1B <- dataMean[1] + 2.32 * sqrt(cov(data)[1,1] / 45)
Bf2A <- dataMean[2] - 2.32 * sqrt(cov(data)[2,2] / 45)
Bf2B <- dataMean[2] +  2.32 * sqrt(cov(data)[2,2] / 45)
Bonferroni <- list(TailLength =c(Bf1A, Bf1B) , WingLength=c(Bf2A, Bf2B))

l <- list(Hotellings = Hotellings,
          Bonferroni = Bonferroni)
```

```r
print(l)
```

```
## $Hotellings
## $Hotellings$TailLength
## [1] 189.4247 197.8198
##
## $Hotellings$WingLength
## [1] 274.2602 285.2954
##
##
## $Bonferroni
## $Bonferroni$TailLength
## [1] 189.8227 197.4217
##
## $Bonferroni$WingLength
## [1] 274.7835 284.7721
```

The 95% confidence interval for mean of tail and wing length using Hotellings T^2:

Tail Length: CI(μ)=(189.4247, 197.8198)

Wing Length: CI(μ)=(274.2602, 285.2954)

The 95% confidence interval for mean of tail and wing length using Bonferroni:

Tail Length: CI(μ)=(189.8227, 197.4217)

Wing Length: CI(μ)=(274.7835, 284.7721)

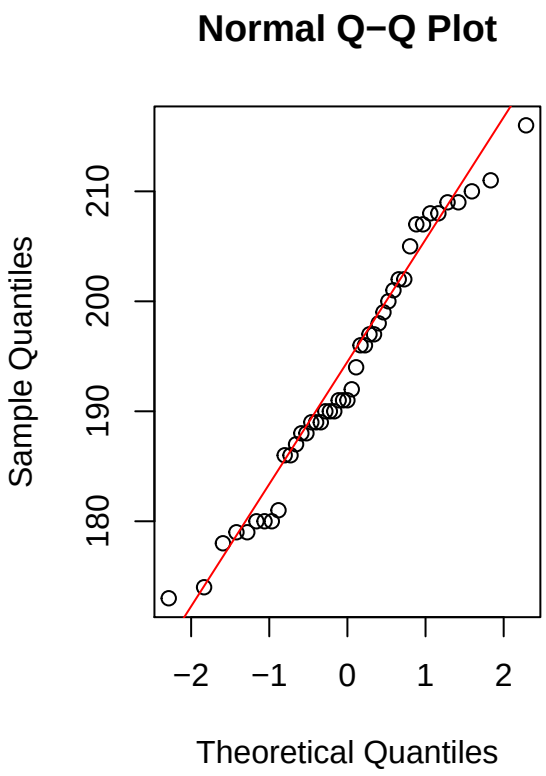
As expected, the Bonferroni intervals are tighter than T^2, but the difference between the intevals is small. When the mean components is large, it is prefered to use a T^2-intervals.

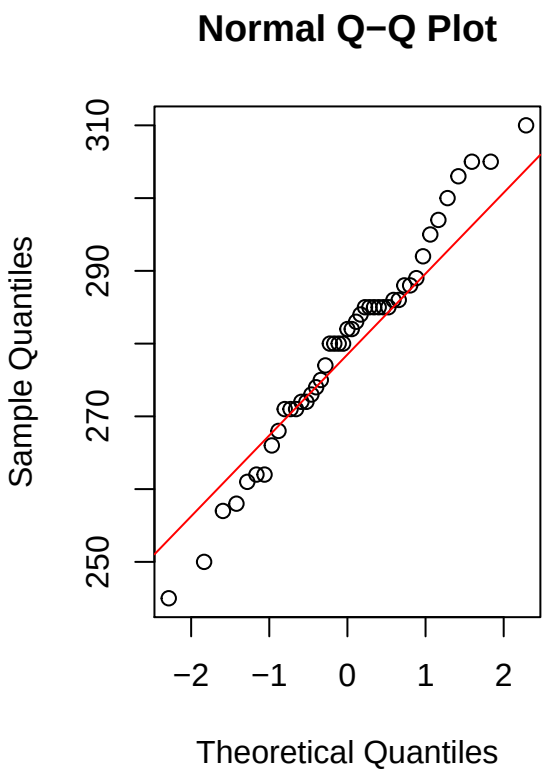
c)

Is the bivariate normal distribution a viable population model? Explain with reference to Q-Q plots and a scatter diagram.

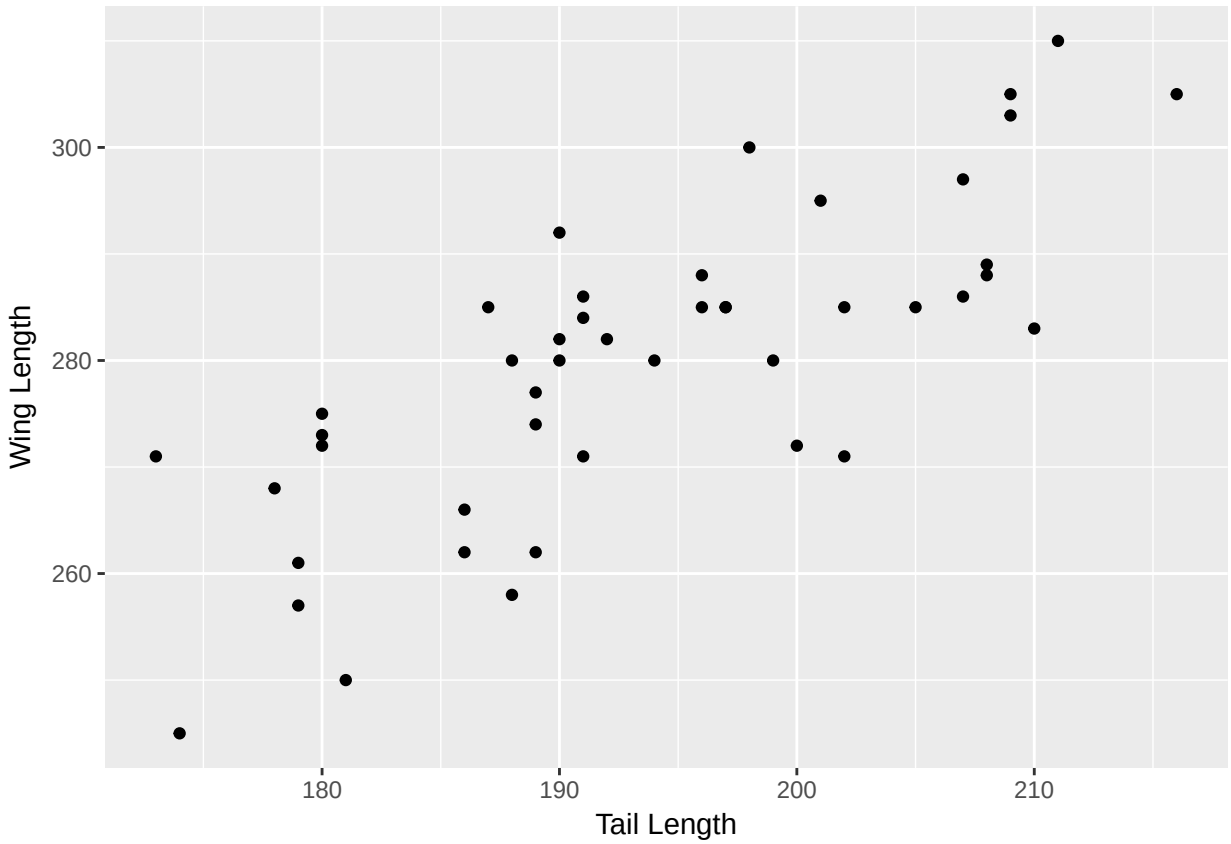
In the Q-Q plot below to the left the values lies well alongside the red line. The one to the right is more irregular but is still good. This would suggest that the bivariate normal distribution is a viable population model.

```r
#Q-Q
par(mfrow=c(1,2))
qqnorm(data[,1]); qqline(data[,1], col = 2)
qqnorm(data[,2]); qqline(data[,2], col = 2)
```






```r
par(mfrow=(c(1,1)))

library(ggplot2)
ggplot(data, aes(x=V1, y=V2)) +
    geom_point() +
    labs(x = "Tail Length", y = "Wing Length")
```



If we observe the scatter diagram above, we notice how de dots would match into an ellipse. That supports the assumption for viability of the bivariate normal distribution as a population model.

Question 3

In this step, Egyptian skull measurements will be studied. This data set, published in 1905 and now taken from the heplots R-package.

Researchers have suggested that a change in skull size over time is evidence of the interbreeding of a resident population with immigrant populations. Four measurements were made of male Egyptian skulls for three different time periods: period 1 is 4000 B.C., period 2 is 3300 B.C., and period 3 is 1850 B.C. The measured variables are

X_1 = maximum breadth of skull (mm)

X_2 = basibregmatic height of skull (mm)

X_3 = basialveolar length of skull (mm)

X_4 = nasal height ofskull (mm)

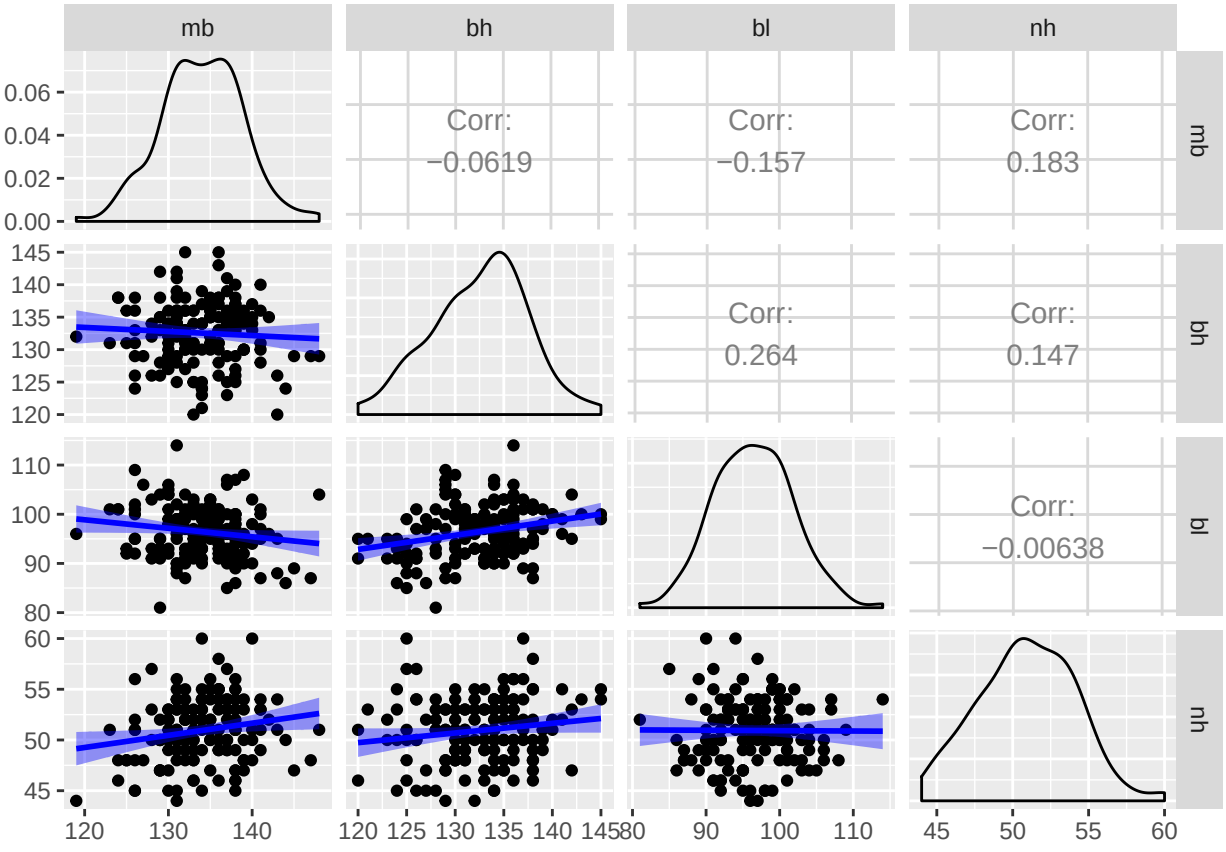
a)

The libraries to be used during the assignment are imported.

```
library(heplots)
library(ggplot2)
library(GGally)
library(gridExtra)
```

Before working with the data we should investigate it. The data will be plotted with a combination of a scatter plot and correlation graph. In this graph will provide information about the correlation between variables as well as their distribution.

```
data <- Skulls
my_fn <- function(data, mapping, ...){
  p <- ggplot(data = data, mapping = mapping) +
  geom_point() +
  geom_smooth(method=lm, fill="blue", color="blue", ...)
  p
}
ggpairs(data[,2:5],columns = 1:4, lower = list(continuous = my_fn))
```



According to graph, the scatter plot don't contain any visual linear pattern and the density plot indicate that each variable is normal distributed. The correlation is close to zero between the different variables which indicates that the variables are linearly independent.

b)

In this part, manova() function will be used in the analysis and $\alpha = 0.05$ will be used for evaluation.

```
a <- manova(cbind(mb,bh,bl,nh) ~ epoch, data)
summary(a)
```

```
##           Df  Pillai approx F num Df den Df    Pr(>F)
## epoch      4 0.35331    3.512     16    580 4.675e-06 ***
## Residuals 145
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

According to p value analysis, we can say that null hypothesis is rejected. The means differ.

Additionally, we can analyze the results by applying different test types, such as, Hotelling-Lawley and Wilks'.

```
summary(a,test="Hotelling-Lawley")
```

```
##           Df Hotelling-Lawley approx F num Df den Df    Pr(>F)
## epoch      4          0.48182    4.231     16    562 8.278e-08 ***
## Residuals 145
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(a,test="Wilks")
```

```
##           Df   Wilks approx F num Df den Df   Pr(>F)
## epoch      4 0.66359   3.9009     16 434.45 7.01e-07 ***
## Residuals 145
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary.aov(a)
```

```
##  Response mb :
##              Df  Sum Sq Mean Sq F value    Pr(>F)
## epoch         4  502.83 125.707  5.9546 0.0001826 ***
## Residuals   145 3061.07  21.111
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response bh :
##              Df Sum Sq Mean Sq F value  Pr(>F)
## epoch         4  229.9  57.477  2.4474 0.04897 *
## Residuals   145 3405.3  23.485
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response bl :
##              Df Sum Sq Mean Sq F value    Pr(>F)
## epoch         4  803.3 200.823  8.3057 4.636e-06 ***
```

```
## Residuals   145 3506.0  24.179
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response nh :
##              Df Sum Sq Mean Sq F value Pr(>F)
## epoch         4   61.2  15.300   1.507 0.2032
## Residuals   145 1472.1  10.153
```

We will interpret the p-value for each variable with $\alpha = 0.05$ as confidence level. When we interpret the following data accordingly:

- MB is significant
- BH is significant, but it is close to the cut off.
- BL highly significant
- NH is not significant

If the variable is significant, it means that the variance between the epochs is bigger than within the epoch which implies that there is a significant difference between the epochs.

c)

We inspect the residuals whether they have mean 0. According to result they are close to zero, so we can accept them as zero.

```
colMeans(a$residuals)
```

```
##            mb            bh            bl            nh
## -4.107825e-17  2.177887e-16 -8.881784e-17 -2.572017e-17
```

Simultaneous Confidence Interval Calculation

```
# Simultaneous confidence interval

w <- c(3061.067,3405.267,3505.967,1472.133) # taken from printed a variable
epoches <- unique(data$epoch)
n <- nrow(data)
g <- length(epoches)
p <- ncol(data[2:5])
tResult <- 0.5002

# Column means for  each epoch
df <- data.frame(epoch = character(),mb = numeric(), bh = numeric(),bl = numeric(),nh = numeric())
for(i in 1:g){
  epoch <- epoches[[i]]
  onlyChosenEpoch <- data[ data$epoch == epoch,]
  columnMeans <- colMeans(onlyChosenEpoch[2:5])
  row <- data.frame(epoch = epoch, mb = columnMeans["mb"],
                    bh = columnMeans["bh"],bl = columnMeans["bl"],nh = columnMeans["nh"])
  df <- rbind(df,row)
}
row.names(df) <- seq(nrow(df))
```

```
results <- data.frame(upper = numeric(), under = numeric(), xki = character(),xki_value = numeric(), xli

convertValuesToRow <- function(ith_Row,targetRow,variable,i,second_in_formula,third_in_formula){
  subs <- ith_Row[variable] -  targetRow[variable]
  under <- subs[1,1] + second_in_formula + third_in_formula
  upper <- subs[1,1] - second_in_formula + third_in_formula
  row <- data.frame(upper=upper, under=under, xki = paste(variable,"-",as.character(df[i,1])), xki_value
  return(row)
}

for(i in 1:4){
  ith_Row <- df[i,]
  indexes <- seq(nrow(df))
  indexes <- indexes[which(!indexes==i)]
  ith_w <- w[i]

  third_in_formula <- sqrt(ith_w/(n-g)*(1/4+1/5))
  second_in_formula <- tResult

  for(z in 1:4){
    row <- convertValuesToRow(ith_Row,df[indexes[z],],"mb",i,second_in_formula,third_in_formula)
    results <- rbind(results, row)
  }
  for(z in 1:4){
    row <- convertValuesToRow(ith_Row,df[indexes[z],],"bh",i,second_in_formula,third_in_formula)
    results <- rbind(results, row)
  }
  for(z in 1:4){
    row <- convertValuesToRow(ith_Row,df[indexes[z],],"bl",i,second_in_formula,third_in_formula)
    results <- rbind(results, row)
  }
  for(z in 1:4){
    row <- convertValuesToRow(ith_Row,df[indexes[z],],"nh",i,second_in_formula,third_in_formula)
    results <- rbind(results, row)
  }

}
print(results)
```

```
##         upper       under          xki xki_value          xli xli_value
## 1   1.5819848  2.58238479 mb - c4000BC 131.36667 mb - c3300BC 132.36667
## 2  -0.5180152  0.48238479 mb - c4000BC 131.36667 mb - c1850BC 134.46667
## 3  -1.5513485 -0.55094854 mb - c4000BC 131.36667  mb - c200BC 135.50000
## 4  -2.2180152 -1.21761521 mb - c4000BC 131.36667  mb - cAD150 136.16667
## 5   3.4819848  4.48238479 bh - c4000BC 133.60000 bh - c3300BC 132.70000
## 6   2.3819848  3.38238479 bh - c4000BC 133.60000 bh - c1850BC 133.80000
## 7   3.8819848  4.88238479 bh - c4000BC 133.60000  bh - c200BC 132.30000
## 8   5.8486515  6.84905146 bh - c4000BC 133.60000  bh - cAD150 130.33333
## 9   2.6819848  3.68238479 bl - c4000BC  99.16667 bl - c3300BC  99.06667
## 10  5.7153181  6.71571813 bl - c4000BC  99.16667 bl - c1850BC  96.03333
## 11  7.2153181  8.21571813 bl - c4000BC  99.16667  bl - c200BC  94.53333
## 12  8.2486515  9.24905146 bl - c4000BC  99.16667  bl - cAD150  93.50000
## 13  2.8819848  3.88238479 nh - c4000BC  50.53333 nh - c3300BC  50.23333
```

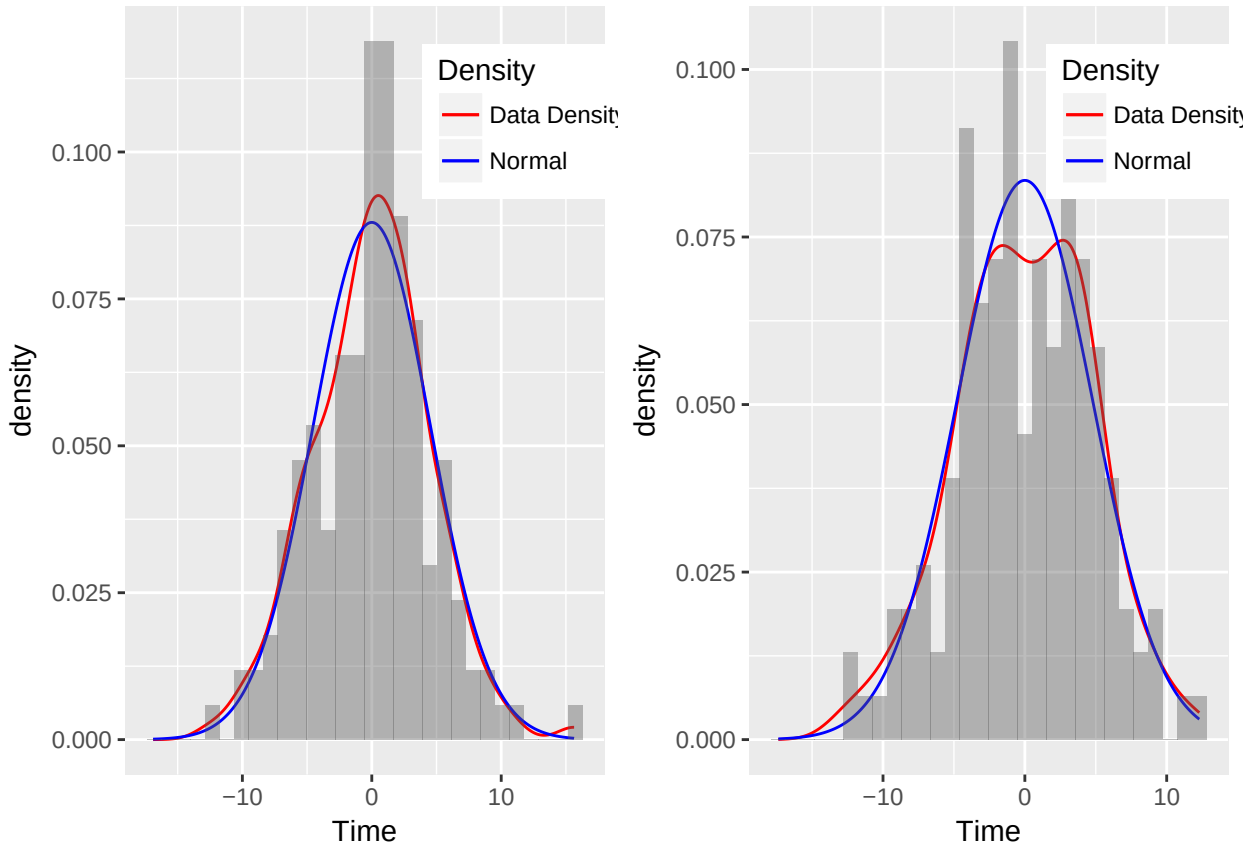
```
## 14  2.5486515  3.54905146 nh - c4000BC  50.53333 nh - c1850BC  50.56667
## 15  1.1486515  2.14905146 nh - c4000BC  50.53333  nh - c200BC  51.96667
## 16  1.7486515  2.74905146 nh - c4000BC  50.53333  nh - cAD150  51.36667
## 17  3.7506568  4.75105681 mb - c3300BC 132.36667 mb - c4000BC 131.36667
## 18  0.6506568  1.65105681 mb - c3300BC 132.36667 mb - c1850BC 134.46667
## 19 -0.3826765  0.61772348 mb - c3300BC 132.36667  mb - c200BC 135.50000
## 20 -1.0493432 -0.04894319 mb - c3300BC 132.36667  mb - cAD150 136.16667
## 21  1.8506568  2.85105681 bh - c3300BC 132.70000 bh - c4000BC 133.60000
## 22  1.6506568  2.65105681 bh - c3300BC 132.70000 bh - c1850BC 133.80000
## 23  3.1506568  4.15105681 bh - c3300BC 132.70000  bh - c200BC 132.30000
## 24  5.1173235  6.11772348 bh - c3300BC 132.70000  bh - cAD150 130.33333
## 25  2.6506568  3.65105681 bl - c3300BC  99.06667 bl - c4000BC  99.16667
## 26  5.7839901  6.78439014 bl - c3300BC  99.06667 bl - c1850BC  96.03333
## 27  7.2839901  8.28439014 bl - c3300BC  99.06667  bl - c200BC  94.53333
## 28  8.3173235  9.31772348 bl - c3300BC  99.06667  bl - cAD150  93.50000
## 29  2.4506568  3.45105681 nh - c3300BC  50.23333 nh - c4000BC  50.53333
## 30  2.4173235  3.41772348 nh - c3300BC  50.23333 nh - c1850BC  50.56667
## 31  1.0173235  2.01772348 nh - c3300BC  50.23333  nh - c200BC  51.96667
## 32  1.6173235  2.61772348 nh - c3300BC  50.23333  nh - cAD150  51.36667
## 33  5.8983735  6.89877352 mb - c1850BC 134.46667 mb - c4000BC 131.36667
## 34  4.8983735  5.89877352 mb - c1850BC 134.46667 mb - c3300BC 132.36667
## 35  1.7650402  2.76544018 mb - c1850BC 134.46667  mb - c200BC 135.50000
## 36  1.0983735  2.09877352 mb - c1850BC 134.46667  mb - cAD150 136.16667
## 37  2.9983735  3.99877352 bh - c1850BC 133.80000 bh - c4000BC 133.60000
## 38  3.8983735  4.89877352 bh - c1850BC 133.80000 bh - c3300BC 132.70000
## 39  4.2983735  5.29877352 bh - c1850BC 133.80000  bh - c200BC 132.30000
## 40  6.2650402  7.26544018 bh - c1850BC 133.80000  bh - cAD150 130.33333
## 41 -0.3349598  0.66544018 bl - c1850BC  96.03333 bl - c4000BC  99.16667
## 42 -0.2349598  0.76544018 bl - c1850BC  96.03333 bl - c3300BC  99.06667
## 43  4.2983735  5.29877352 bl - c1850BC  96.03333  bl - c200BC  94.53333
## 44  5.3317068  6.33210685 bl - c1850BC  96.03333  bl - cAD150  93.50000
## 45  2.8317068  3.83210685 nh - c1850BC  50.56667 nh - c4000BC  50.53333
## 46  3.1317068  4.13210685 nh - c1850BC  50.56667 nh - c3300BC  50.23333
## 47  1.3983735  2.39877352 nh - c1850BC  50.56667  nh - c200BC  51.96667
## 48  1.9983735  2.99877352 nh - c1850BC  50.56667  nh - cAD150  51.36667
## 49  5.7705824  6.77098243  mb - c200BC 135.50000 mb - c4000BC 131.36667
## 50  4.7705824  5.77098243  mb - c200BC 135.50000 mb - c3300BC 132.36667
## 51  2.6705824  3.67098243  mb - c200BC 135.50000 mb - c1850BC 134.46667
## 52  0.9705824  1.97098243  mb - c200BC 135.50000  mb - cAD150 136.16667
## 53  0.3372491  1.33764909  bh - c200BC 132.30000 bh - c4000BC 133.60000
## 54  1.2372491  2.23764909  bh - c200BC 132.30000 bh - c3300BC 132.70000
## 55  0.1372491  1.13764909  bh - c200BC 132.30000 bh - c1850BC 133.80000
## 56  3.6039158  4.60431576  bh - c200BC 132.30000  bh - cAD150 130.33333
## 57 -2.9960842 -1.99568424  bl - c200BC  94.53333 bl - c4000BC  99.16667
## 58 -2.8960842 -1.89568424  bl - c200BC  94.53333 bl - c3300BC  99.06667
## 59  0.1372491  1.13764909  bl - c200BC  94.53333 bl - c1850BC  96.03333
## 60  2.6705824  3.67098243  bl - c200BC  94.53333  bl - cAD150  93.50000
## 61  3.0705824  4.07098243  nh - c200BC  51.96667 nh - c4000BC  50.53333
## 62  3.3705824  4.37098243  nh - c200BC  51.96667 nh - c3300BC  50.23333
## 63  3.0372491  4.03764909  nh - c200BC  51.96667 nh - c1850BC  50.56667
## 64  2.2372491  3.23764909  nh - c200BC  51.96667  nh - cAD150  51.36667
##    t_value      wii value_after_wii
## 1   0.5002 3061.067        3.082185
## 2   0.5002 3061.067        3.082185
```

```
## 3   0.5002 3061.067        3.082185
## 4   0.5002 3061.067        3.082185
## 5   0.5002 3061.067        3.082185
## 6   0.5002 3061.067        3.082185
## 7   0.5002 3061.067        3.082185
## 8   0.5002 3061.067        3.082185
## 9   0.5002 3061.067        3.082185
## 10  0.5002 3061.067        3.082185
## 11  0.5002 3061.067        3.082185
## 12  0.5002 3061.067        3.082185
## 13  0.5002 3061.067        3.082185
## 14  0.5002 3061.067        3.082185
## 15  0.5002 3061.067        3.082185
## 16  0.5002 3061.067        3.082185
## 17  0.5002 3405.267        3.250857
## 18  0.5002 3405.267        3.250857
## 19  0.5002 3405.267        3.250857
## 20  0.5002 3405.267        3.250857
## 21  0.5002 3405.267        3.250857
## 22  0.5002 3405.267        3.250857
## 23  0.5002 3405.267        3.250857
## 24  0.5002 3405.267        3.250857
## 25  0.5002 3405.267        3.250857
## 26  0.5002 3405.267        3.250857
## 27  0.5002 3405.267        3.250857
## 28  0.5002 3405.267        3.250857
## 29  0.5002 3405.267        3.250857
## 30  0.5002 3405.267        3.250857
## 31  0.5002 3405.267        3.250857
## 32  0.5002 3405.267        3.250857
## 33  0.5002 3505.967        3.298574
## 34  0.5002 3505.967        3.298574
## 35  0.5002 3505.967        3.298574
## 36  0.5002 3505.967        3.298574
## 37  0.5002 3505.967        3.298574
## 38  0.5002 3505.967        3.298574
## 39  0.5002 3505.967        3.298574
## 40  0.5002 3505.967        3.298574
## 41  0.5002 3505.967        3.298574
## 42  0.5002 3505.967        3.298574
## 43  0.5002 3505.967        3.298574
## 44  0.5002 3505.967        3.298574
## 45  0.5002 3505.967        3.298574
## 46  0.5002 3505.967        3.298574
## 47  0.5002 3505.967        3.298574
## 48  0.5002 3505.967        3.298574
## 49  0.5002 1472.133        2.137449
## 50  0.5002 1472.133        2.137449
## 51  0.5002 1472.133        2.137449
## 52  0.5002 1472.133        2.137449
## 53  0.5002 1472.133        2.137449
## 54  0.5002 1472.133        2.137449
## 55  0.5002 1472.133        2.137449
## 56  0.5002 1472.133        2.137449
```

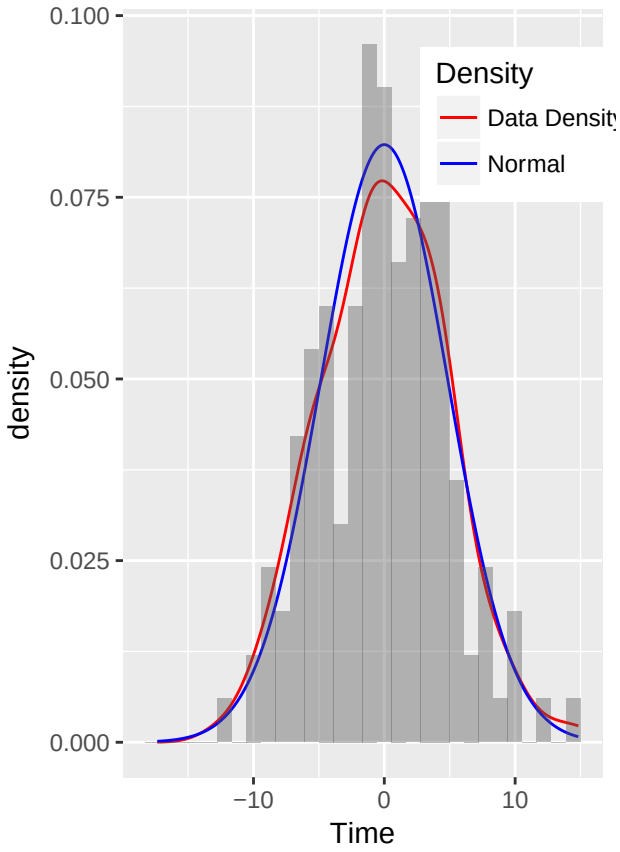
```
## 57  0.5002 1472.133        2.137449
## 58  0.5002 1472.133        2.137449
## 59  0.5002 1472.133        2.137449
## 60  0.5002 1472.133        2.137449
## 61  0.5002 1472.133        2.137449
## 62  0.5002 1472.133        2.137449
## 63  0.5002 1472.133        2.137449
## 64  0.5002 1472.133        2.137449
```

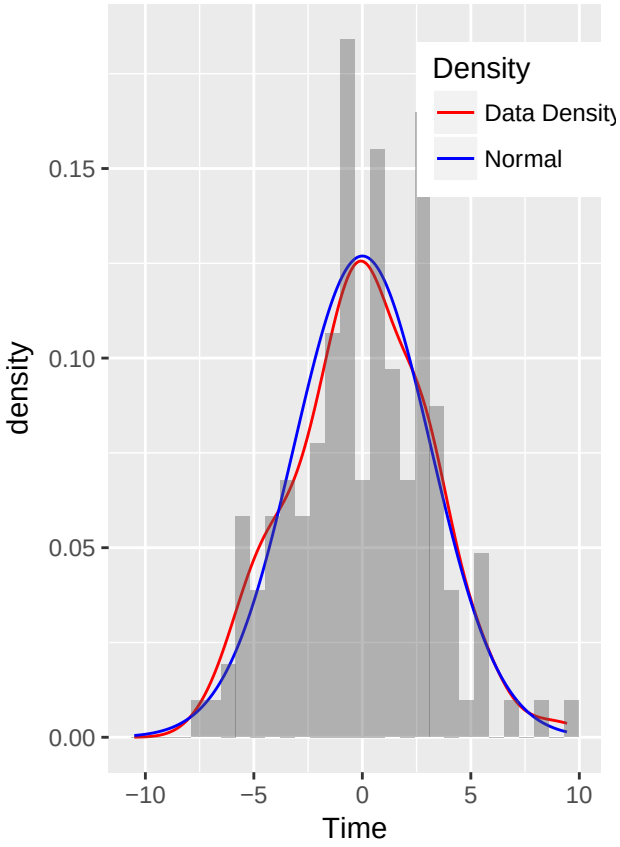
Now, each residual variable is plotted with normal distribution line and its density line. So, it is possible to observe them if they deviate from normality as graphically.

```
graphs <- list()
for(i in 1:ncol(a$residuals)){
    selectedColumn <- a$residuals[,i]
    minColumn <- min(selectedColumn)
    maxColumn <- max(selectedColumn)
    meanColumn <- mean(selectedColumn)
    sdColumn <- sd(selectedColumn)
    range <- 1:150
    range <- eval(substitute(seq(as.numeric(minColumn) - as.numeric(sdColumn),
                                 as.numeric(maxColumn), by = 0.01),
                             list(i=i,minColumn=minColumn,
                                  sdColumn=sdColumn,maxColumn=maxColumn)))
    ynorm <- eval(substitute(dnorm(x = range, mean = meanColumn, sd = sdColumn),
                             list(i=i,sdColumn=sdColumn,meanColumn=meanColumn)))
    graphs[[i]] <- eval(substitute(qplot(a$residuals[,i], geom = "blank") +
        geom_line(aes(y = ..density.., colour = "Data Density"), stat = 'density') +
        geom_line(aes(x = range, y = ynorm, color = "Normal")) +
        geom_histogram(aes(y = ..density..), alpha = 0.4) +
        scale_colour_manual(name = 'Density', values = c('red', 'blue')) +
        theme(legend.position = c(0.85, 0.85)) +
        xlab("Time") +
        ggtitle(names(a$residuals[i])),list(i=i,range=range,ynorm=ynorm)))
}

grid.arrange(graphs[[1]], graphs[[2]], ncol=2)
```




```
grid.arrange(graphs[[3]],graphs[[4]], ncol=2)
```


0.100
0.075
0.050
0.025
0.000
density
−10
0
10
Time
Density
Data Density
Normal


0.15
0.10
0.05
0.00
density
−10
−5
0
5
10
Time
Density
Data Density
Normal