Exercise 3

Tasks 14 to 18

21.10.2023

Contents

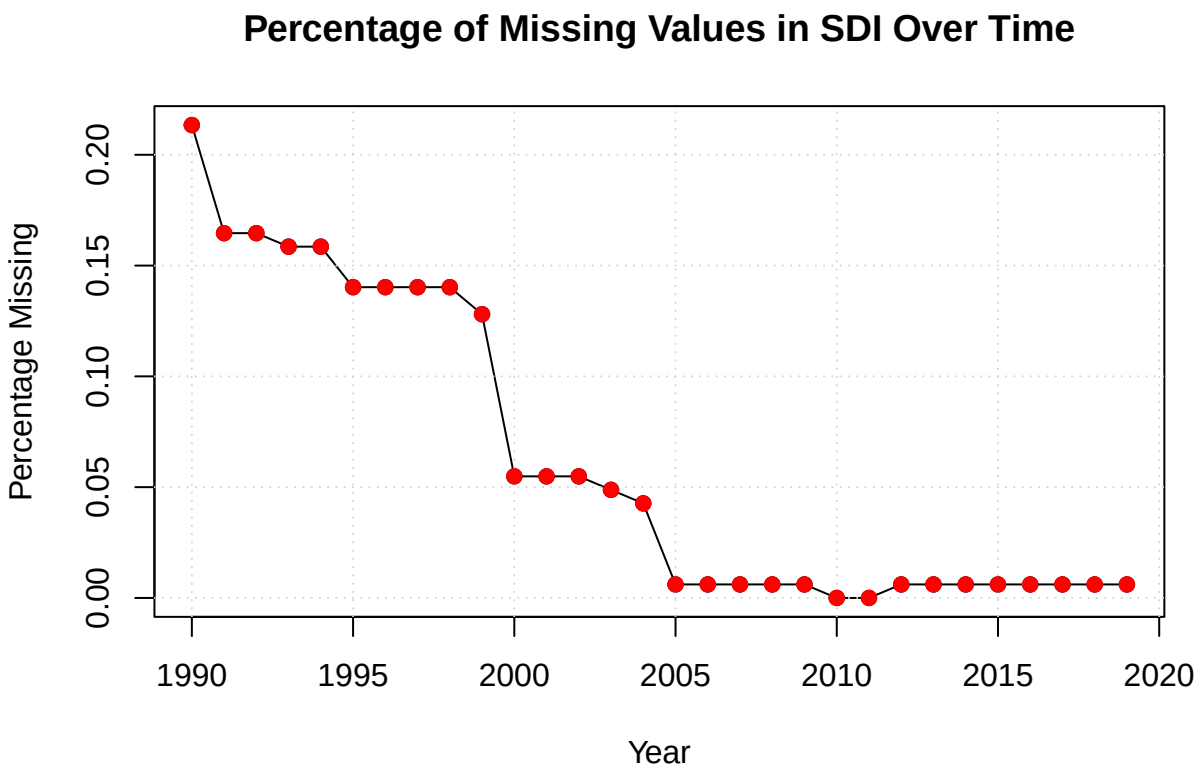
Task 14:

14a: download and installation

```
sdi <- read.csv("./Exercise_3/sdi.csv")
#sdi
```

14b: missing values:

```
# Calculate the percentage of missing values for each year
missing_data <- sapply(X = sdi[, -1], FUN = function (x) sum(is.na(x)))
percentage_missing <- missing_data / length(sdi$country)


# Create a line plot
plot(1990:2019, percentage_missing, type = "o", xlab = "Year", ylab = "Percentage Missing",
     main = "Percentage of Missing Values in SDI Over Time")

# Add gridlines
grid()

# Add points to the plot for better visualization
points(1990:2019, percentage_missing, pch = 19, col = "red")
```



14c: percentage with complete observations:

```
# Count the number of complete observations (non-missing values) for each country
complete_observations <- apply(sdi[, -1], 1, function(row) sum(!is.na(row)) == length(row))

# Calculate the percentage of countries with complete observations
percentage_complete <- (sum(complete_observations) / length(complete_observations)) * 100

# Print the result
cat("Percentage of countries with complete observations in the SDI:", percentage_complete, "%\n")
```

```
## Percentage of countries with complete observations in the SDI: 78.65854 %
```

To calculate the percentage of countries with complete observations in the SDI, you can count the number of years for which there are no missing values in the SDI for each country and then calculate the percentage of countries with complete observations

14d: missing values per country

```
# Calculate the number of missing values per country using rowSums
missings_per_country <- rowSums(is.na(sdi))

# Assign country names as names for the vector
names(missings_per_country) <- sdi$country

# Filter countries with at least one missing value & Sort the vector in decreasing order
missings_per_country <- sort(missings_per_country[missings_per_country > 0], decreasing = TRUE)
missings_per_country
```

```
##                Eritrea           Turkmenistan    Antigua and Barbuda
##                     23                     20                     15
##                 Bhutan                Lebanon                 Serbia
##                     15                     15                     15
##                Vanuatu               Suriname                Nigeria
##                     15                     14                     13
##           Burkina Faso               Bahamas Bosnia and Herzegovina
##                     10                     10                     10
##             Cape Verde               Ethiopia                Georgia
##                     10                     10                     10
##             Madagascar       North Macedonia                   Oman
##                     10                     10                     10
##             Seychelles                   Chad             Uzbekistan
##                     10                     10                     10
##                 Angola                Liberia             Azerbaijan
##                      9                      9                      5
##               Djibouti               Maldives         Czech Republic
##                      5                      5                      3
##                Armenia             Kazakhstan        Kyrgyz Republic
##                      1                      1                      1
##                Moldova                 Russia               Slovenia
##                      1                      1                      1
##             Tajikistan                Ukraine
##                      1                      1
```

14e: convert to long

```
sdi_long <- reshape(sdi,
  varying = c(2:31),
  v.names = "sdi",
  timevar ="year",
  direction ="long",
  times = c(1990:2019))

# Print only the first few rows
head(sdi_long)
```

```
##                       country year  sdi id
## 1.1990          Afghanistan 1990 32.5  1
## 2.1990               Angola 1990   NA  2
## 3.1990              Albania 1990 69.9  3
## 4.1990 United Arab Emirates 1990 46.9  4
## 5.1990            Argentina 1990 77.1  5
## 6.1990              Armenia 1990   NA  6
```

```
sdi_long$year <- factor(sdi_long$year, ordered = TRUE)
str(sdi_long)
```

```
## 'data.frame':    4920 obs. of  4 variables:
##  $ country: chr  "Afghanistan" "Angola" "Albania" "United Arab Emirates" ...
##  $ year   : Ord.factor w/ 30 levels "1990"<"1991"<..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ sdi    : num  32.5 NA 69.9 46.9 77.1 NA NA 33.6 68 NA ...
##  $ id     : int  1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, "reshapeLong")=List of 4
##   ..$ varying:List of 1
##   .. ..$ sdi: chr [1:30] "X1990" "X1991" "X1992" "X1993" ...
##   .. ..- attr(*, "v.names")= chr "sdi"
##   .. ..- attr(*, "times")= int [1:30] 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 ...
##   ..$ v.names: chr "sdi"
##   ..$ idvar  : chr "id"
##   ..$ timevar: chr "year"
```

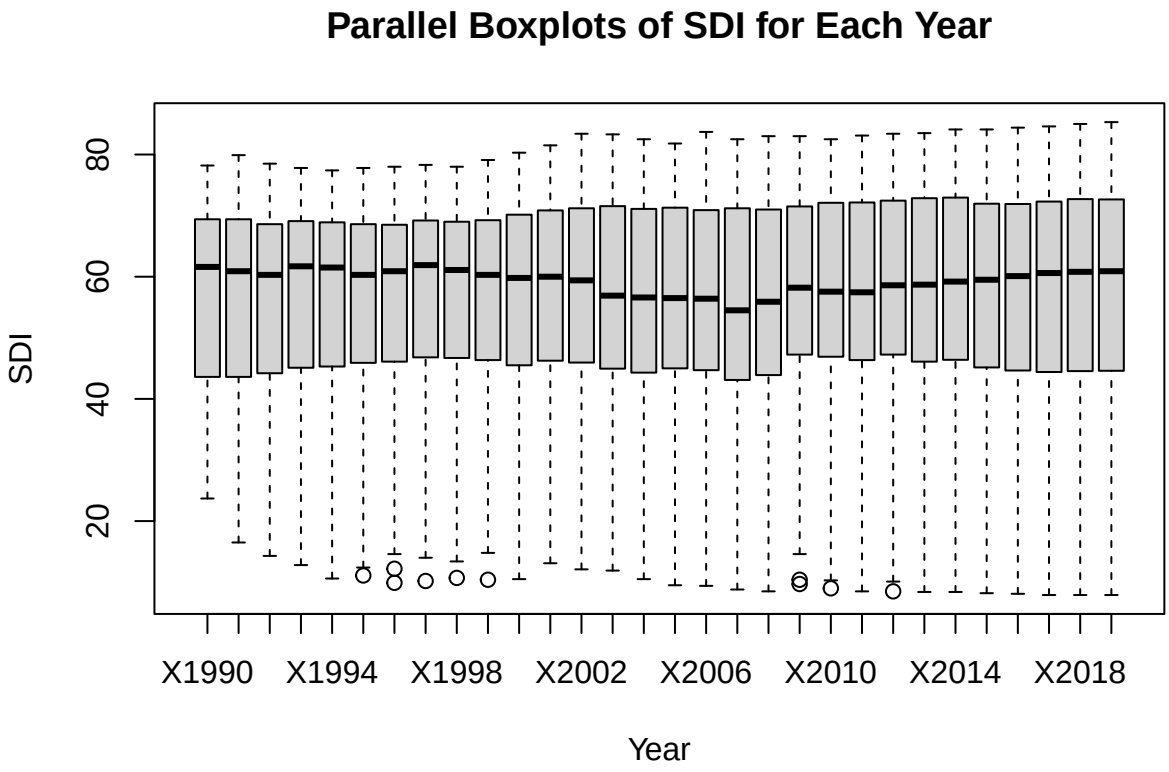
14f: no missing data

```
# Eliminate rows with missing values
sdi_long_no_missing <- na.omit(sdi_long)

# Print the first few rows of the resulting dataset
head(sdi_long_no_missing)
```

```
##                       country year  sdi id
## 1.1990          Afghanistan 1990 32.5  1
## 3.1990              Albania 1990 69.9  3
## 4.1990 United Arab Emirates 1990 46.9  4
## 5.1990            Argentina 1990 77.1  5
## 8.1990            Australia 1990 33.6  8
## 9.1990              Austria 1990 68.0  9
```

14g: parallel boxplots

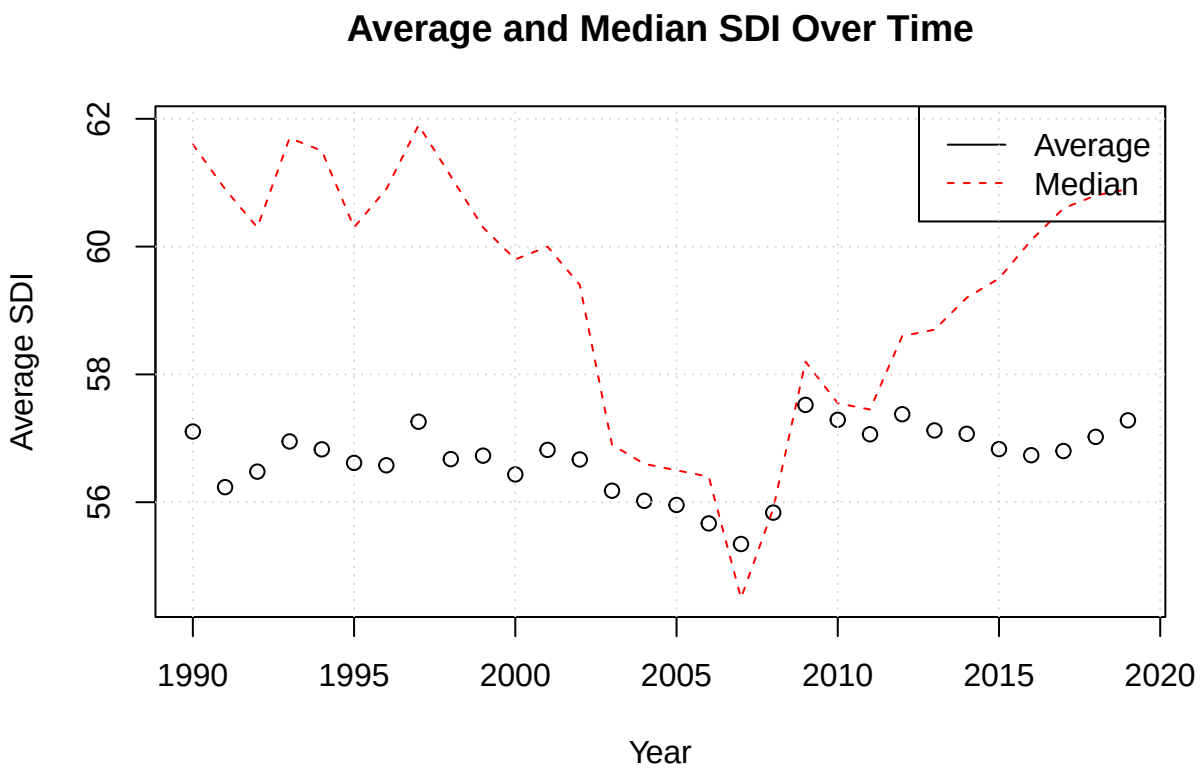
```
# Create parallel boxplots
boxplot(sdi[,-1], xlab = "Year", ylab = "SDI",
        main = "Parallel Boxplots of SDI for Each Year")
```



Based on the above boxplot, one can infer the distribution of SDI values for each year. The arithmetic mean is indicated by the black line within each boxplot. If this line is approximately in the middle, the SDI values for that year are roughly normally distributed. A line that has shifted upwards suggests a left-skewed distribution (i.e., skewness g1 is < 0). A black line located in the lower part of the boxplot indicates a right-skewed distribution (skewness g1 is > 0). Small points outside the whiskers are outliers, meaning that these data points significantly affect the arithmetic mean. They are either much smaller or larger than the other data in the sample and do not fit within the interquartile range times 1.5 (IQR*1.5 - maximum whisker length).

14h: avarage and median from sdi_long

```r
# Calculate average and median SDI for each year
average_sdi <- tapply(sdi_long$sdi, sdi_long$year, mean, na.rm = TRUE)
median_sdi <- tapply(sdi_long$sdi, sdi_long$year, median, na.rm = TRUE)



# Create a line plot
plot(1990:2019, average_sdi, xlab = "Year", ylab = "Average SDI", main = "Average and Median SDI Over T
     ylim = c(min(average_sdi, median_sdi), max(average_sdi, median_sdi)))

# Add dashed lines for medians
lines(1990:2019, median_sdi, lty = "dashed", col = "red")

# Add a legend for the line types
legend("topright", legend = c("Average", "Median"), lty = c(1, 2), col = c("black", "red"))

# Add gridlines
grid()
```



Task 15:

```
# Read movies1.csv and ratings.csv
movies1 <- read.csv("Exercise_3/movies1.csv", stringsAsFactors = FALSE)
movies2 <- read.table("Exercise_3/movies2.csv", sep=";", header = TRUE)
ratings <- read.csv("Exercise_3/ratings.csv")
```

15a: combining 2 datasets

```
# Combine the datasets based on the "movieId" column
user_ratings <- merge(ratings, movies1, by.x = "movieId", all.y = TRUE)
#combined_data

# Find users who didn't rate any movies in movies1.csv:
# first we get all users. Then we get all users that rated.
# Then we get the length of the difference between those two.
all_users <- unique(ratings$userId)
users_with_ratings <- unique(user_ratings$userId)
cat("Users who didn't rate any movies in movies1.csv:",
    length(setdiff(all_users, users_with_ratings)), "\n")
```

```
## Users who didn't rate any movies in movies1.csv: 15
```

```
# Get the dimensions of the combined datasets
user_ratings_all <- merge(movies1, ratings, by.x = "movieId", all.y = TRUE)
user_ratings_user <- merge(ratings, movies1, by.x = "movieId", all.y = TRUE)


cat("Dimension of the combined dataset with all user ratings:",
    dim(user_ratings_all)[1], "rows and",
    dim(user_ratings_all)[2], "columns \n")
```

```
## Dimension of the combined dataset with all user ratings: 100836 rows and 6 columns
```

```
cat("Dimension of the combined dataset with all user ratings:",
    dim(user_ratings_user)[1], "rows and",
    dim(user_ratings_user)[2], "columns \n")
```

```
## Dimension of the combined dataset with all user ratings: 16420 rows and 6 columns
```

15b: combined movies data set:

```
movies <- merge(movies1, movies2, all = TRUE )
#movies
```

15c: merge movies with ratings

```
# Combine the datasets using an outer join
combined_data <- merge(movies, ratings, by.x = "movieId", all.x = TRUE)

# Find movies that did not receive any reviews
movies_with_no_reviews <- combined_data[is.na(combined_data$userId), ]
cat("Number of movies with no reviews:", nrow(movies_with_no_reviews), "\n")
```

```
## Number of movies with no reviews: 18
```

15d: write table into file

```
user_ratings[is.na(user_ratings)] <- 999999 # Replace missing values with 999999
#write.table(user_ratings, file = "combined_data.txt", quote = FALSE, row.names = FALSE, sep = ";")
```

Task 16:

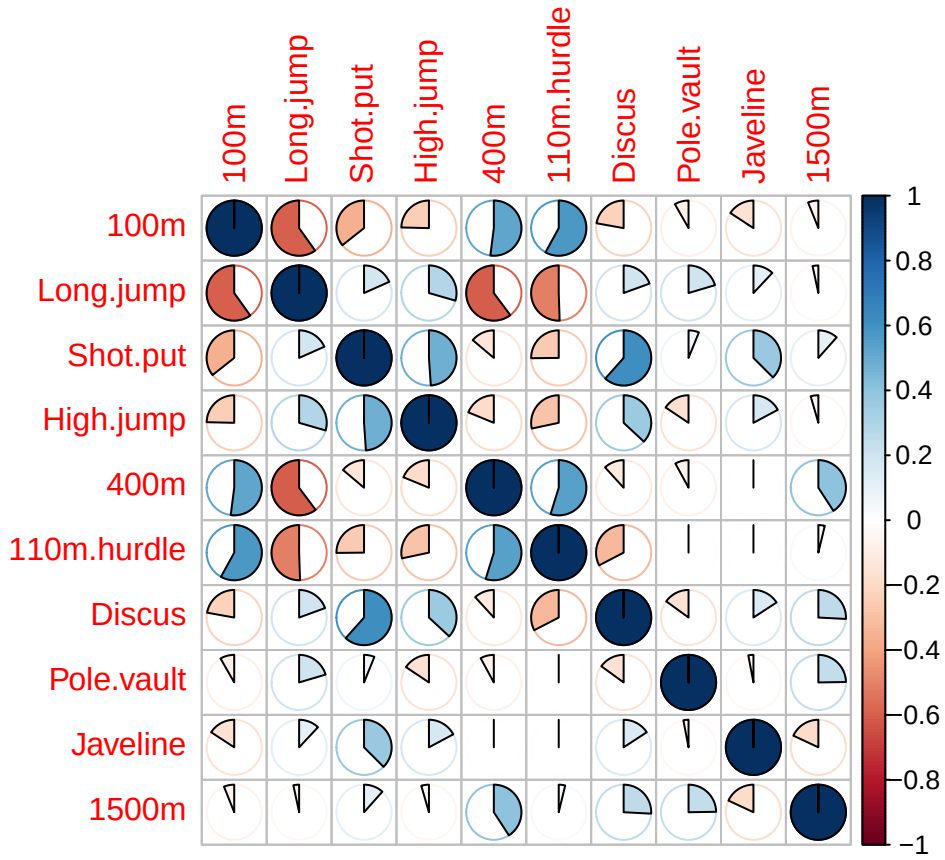
```
#instal.packages("FactoMineR")
library(FactoMineR)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
data("decathlon")
#?decathlon
```

16a: corralation plot:

```
numeric_data <- decathlon[, 1:10]     #Select the first 10 numeric variables
cor_matrix <- cor(numeric_data)       #Calculate the correlation matrix
corrplot(cor_matrix, method = "pie") #and then create the correlation plot
```

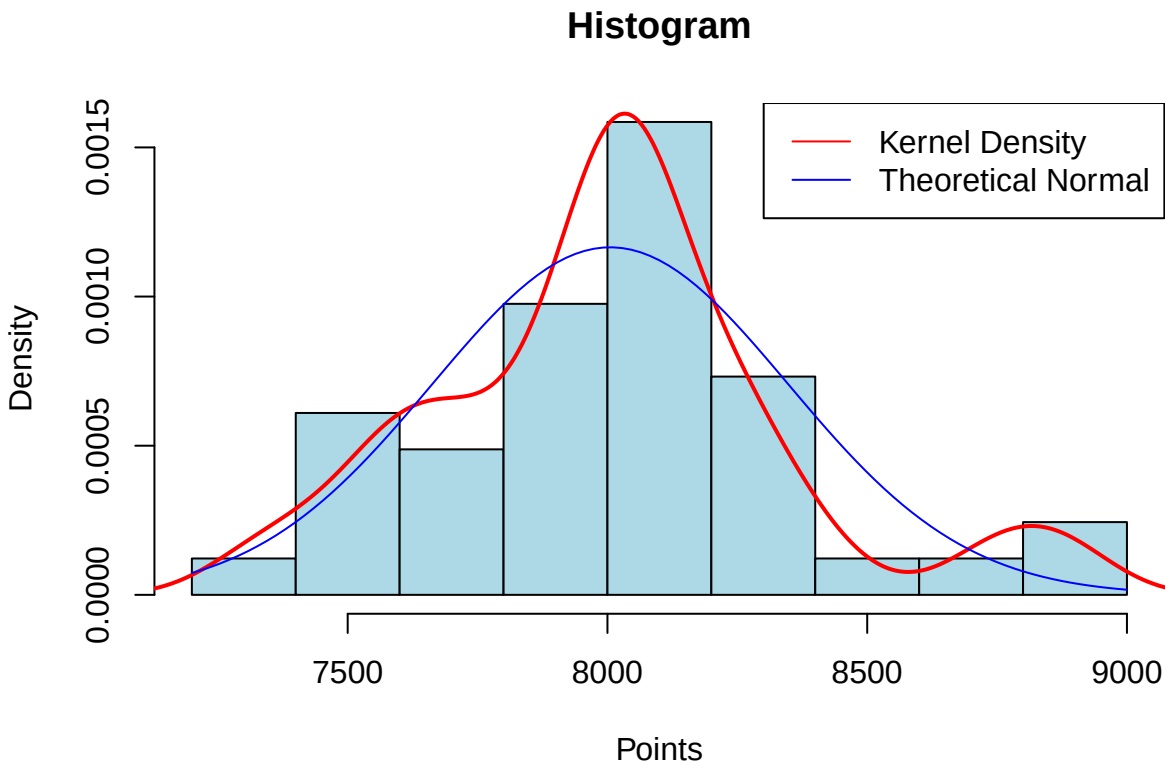


Interpretation:

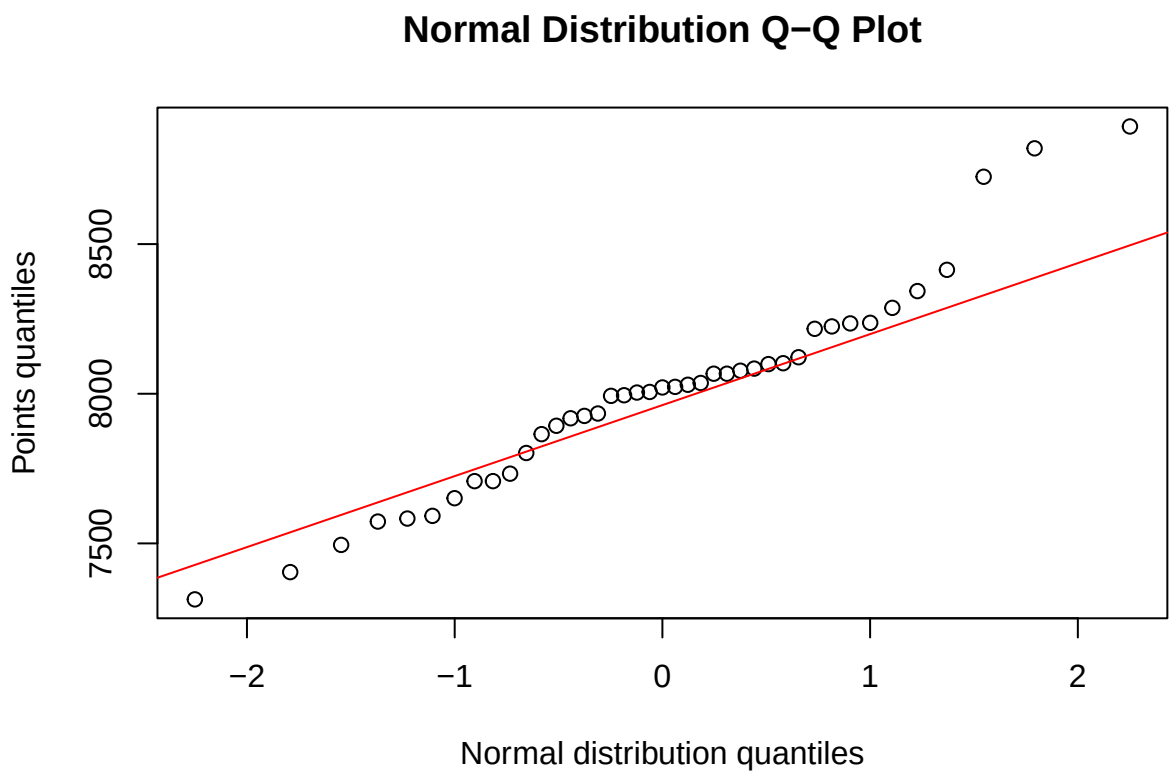
- 1 indicates “perfect positive” correlation.
- -1 indicates “perfect negative” correlation.
- 0 -> there is no correlation between the two variables.
- Values between -1 and 1 indicate the strength of the correlation.

16b: histogram & Q-Q Plot

```
x2 <- seq(min(decathlon$Points), max(decathlon$Points), length = 100)
fun <- dnorm(x2, mean = mean(decathlon$Points), sd = sd(decathlon$Points))

hist(decathlon$Points, col ="lightblue",prob = TRUE,main = "Histogram", xlab = "Points")

lines(density(decathlon$Points), lwd = 2, col ="red")
curve(dnorm(x, mean = mean(decathlon$Points), sd = sd(decathlon$Points)), col = "blue",add=TRUE)

legend("topright",legend=c("Kernel Density", "Theoretical Normal"),col = c("red", "blue"),lty=1)
```



```r
qqnorm(decathlon$Points, main = "Normal Distribution Q-Q Plot",
       xlab ="Normal distribution quantiles", ylab ="Points quantiles")
qqline(decathlon$Points, col = "red")
```

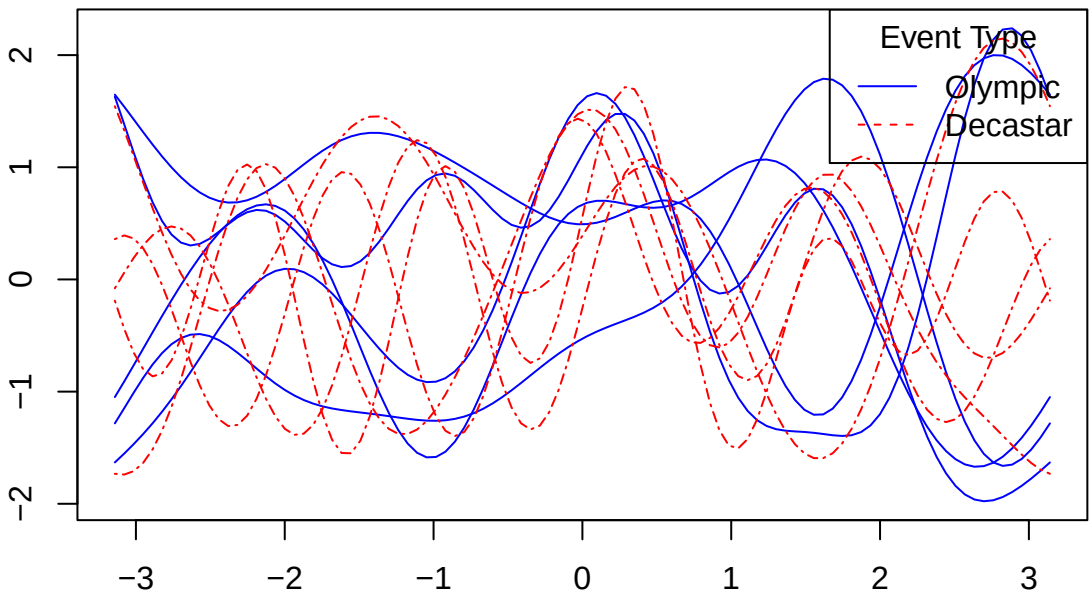


16c: Andrews curve: TODO: THIS DOESN'T WORK

This doesn't work...

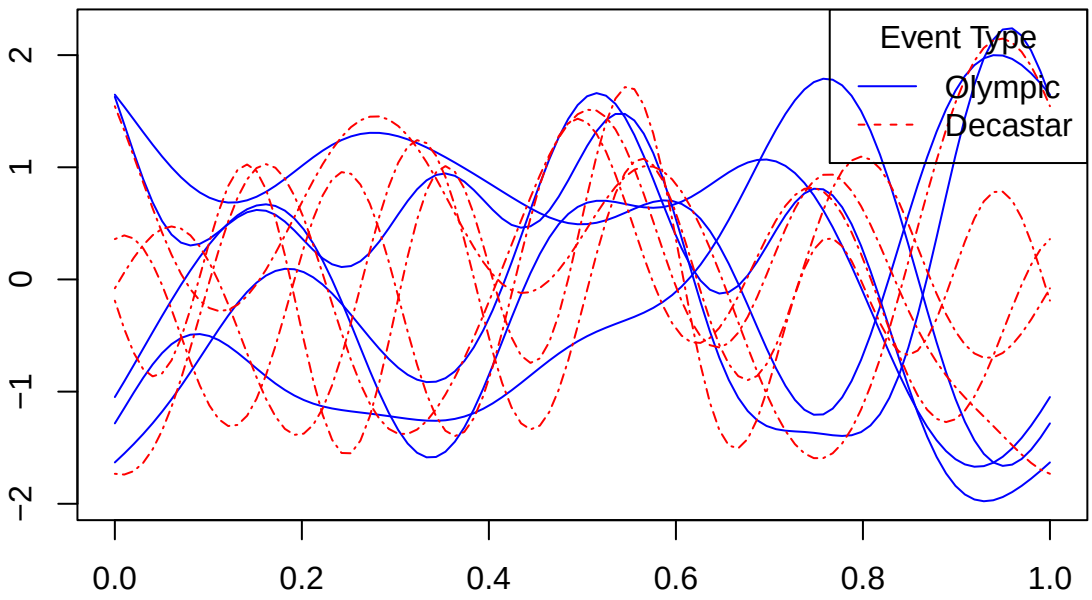
```r
data <- cor_matrix

f_t <- function(t, x) {
  p <- length(x)
  x[1] / sqrt(2) +
    sum(x[2 * (1:floor(p/2))] * sin((1:floor(p/2)) * t)) +
    sum(x[2 * (1:(floor(p/2)-1)) + 1] * cos((1:(floor(p/2) - 1)) * t))
}

# Sequence of t values
tseq <- seq(-pi, pi, length = 100)

# Number of observations
n <- nrow(data)

# Initialize a matrix to store the Andrews curves
andrews_curves <- matrix(0, nrow = n, ncol = length(tseq))
```

```r
# Compute Andrews curves for each observation
for (i in 1:n) {
  andrews_curves[i, ] <- sapply(tseq, function(t) f_t(t, data[i, ]))
}

# Example event_type vector (replace with your actual data)
event_type <- rep(c("Olympic", "Decastar"), each = n / 2)

# Define line types and colors for each event
line_types <- c("solid", "dashed")
line_colors <- c("blue", "red")

# Initialize a plot
plot(0, type = "n", xlim = c(-pi,pi), ylim = range(andrews_curves), xlab = "", ylab = "")

# Iterate through the data and plot Andrews curves
for (i in 1:n) {
  lines(tseq,
        andrews_curves[i, ],
        type = "l",
        pch = 1,
        lty = ifelse(event_type[i] == "Olympic", "solid", "twodash"),
        col = ifelse(event_type[i] == "Olympic", "blue", "red"),
        ylab="",
        xlim=c(0,1))
}

# Add a legend
legend("topright", legend = unique(event_type), col = line_colors, lty = 1:length(unique(event_type)),
```



To better visualize it, I am creating 2 plots next to each other:

```
# Normalise: Is this correct?
tseq <- (tseq - min(tseq)) / (max(tseq) - min(tseq))
plot(0, type = "n", xlim = c(0,1), ylim = range(andrews_curves), xlab = "", ylab = "")

for (i in 1:n) {
  lines(tseq,
        andrews_curves[i, ],
        type = "l",
        pch = 1,
        lty = ifelse(event_type[i] == "Olympic", "solid", "twodash"),
        col = ifelse(event_type[i] == "Olympic", "blue", "red"),
        ylab="",
        xlim=c(0,1))
}

# Add a legend
legend("topright", legend = unique(event_type), col = line_colors, lty = 1:length(unique(event_type)),
```


Event Type
Olympic
Decastar
2
1
0
−1
−2
0.0
0.2
0.4
0.6
0.8
1.0

Task 17:

```
linelist_messy_dates <- readRDS("Exercise_3/linelist_messy_dates.rds")
#linelist_messy_dates
```

17a: import

The dataset contains information about individuals infected with Ebola. Key variables include the ID of the affected person (case_id), the date of hospitalization (date_onset), the date of discharge (date_outcome), and the date of infection (date_infection). Other variables indicate whether the person recovered or died. Additionally, various side effects experienced by the infected individuals are listed as univariate variables, taking the value TRUE if the side effect was present or FALSE if not. The dataset also contains missing values. If information about a specific variable was not known for an individual, it was left blank. For example, if the outcome for an observation is "Dead," the discharge date from the hospital may not be available and will be filled with "NA."

```
#missing values:
cat("Amount of missing data (=Dead patients):", sum(is.na(data)))
```

```
## Amount of missing data (=Dead patients): 0
```

17b: convert into class "Date"

```
#class(linelist_messy_dates$date_infection)
linelist_messy_dates <- readRDS("Exercise_3/linelist_messy_dates.rds")

#since the month names are in English, we need to set the language temporarily to English:
current_locale <- Sys.getlocale("LC_TIME")
Sys.setlocale("LC_TIME", "English")
```

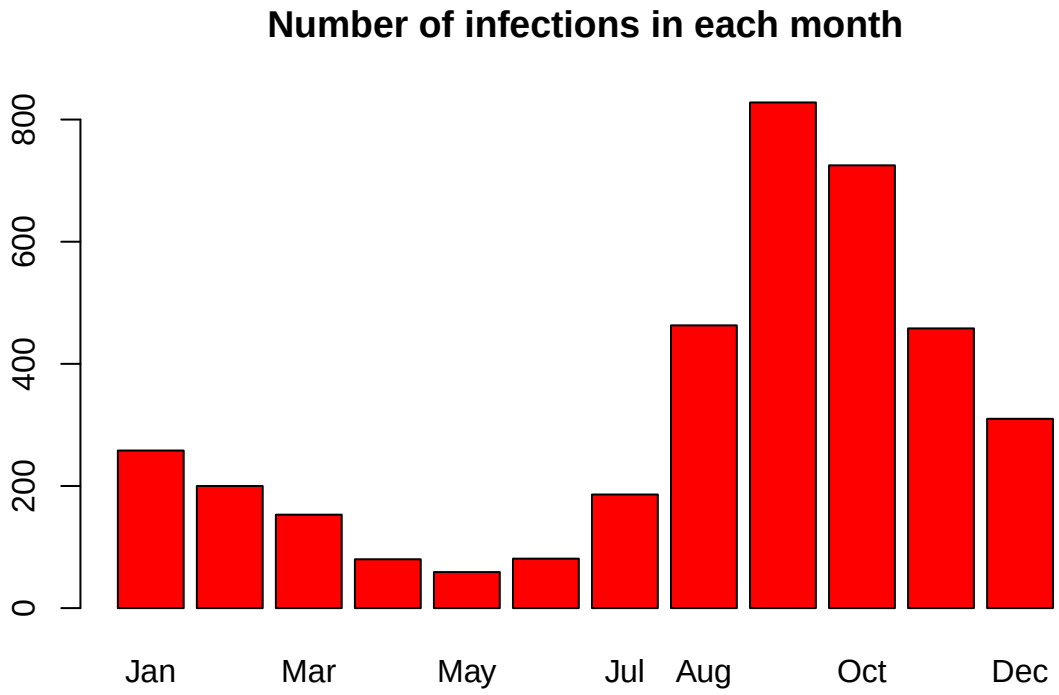
```
## [1] "English_United States.1252"
```

```
linelist_messy_dates$date_infection <- as.Date(
      as.character(linelist_messy_dates$date_infection),
      format="%Y%m%d")

linelist_messy_dates$date_onset <-as.Date(
      linelist_messy_dates$date_onset,
      "%y/%B/%d")


linelist_messy_dates$date_hospitalisation <-as.Date(
      linelist_messy_dates$date_hospitalisation,
      origin = "1970-01-01")


Sys.setlocale("LC_TIME", current_locale)
```

```
## [1] "German_Austria.utf8"
```

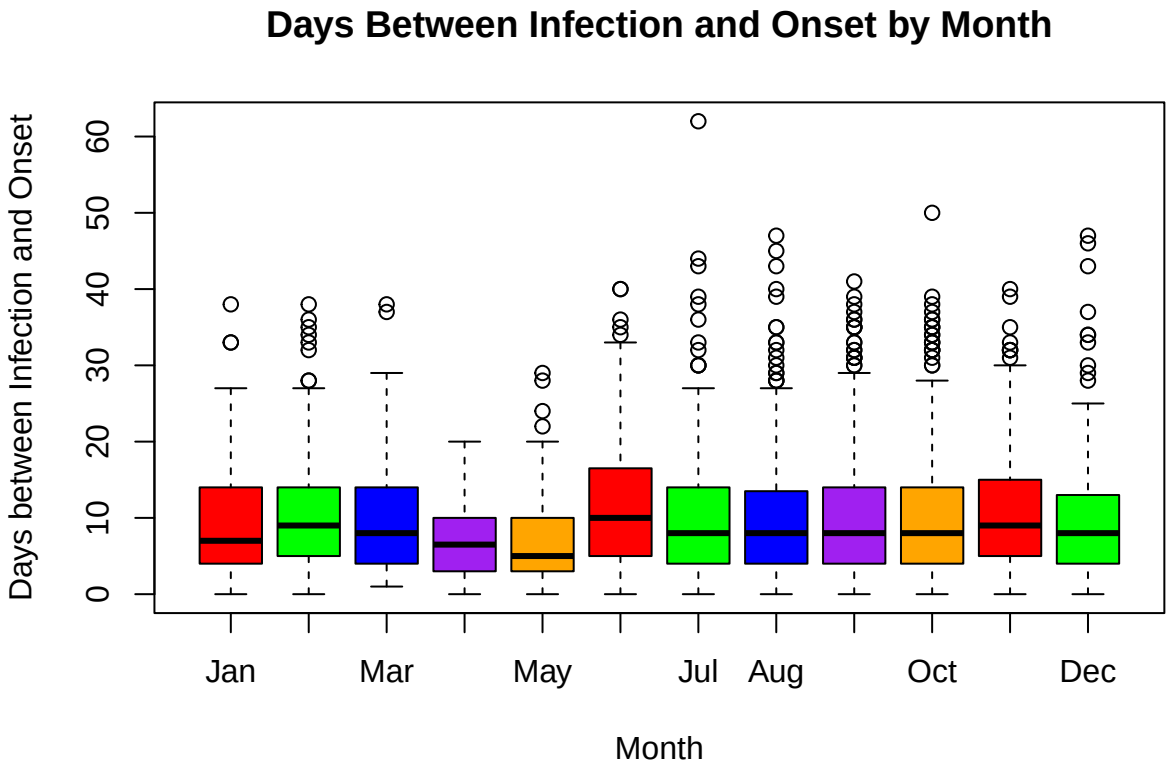
17c: bar plot:

```
months <- format(linelist_messy_dates$date_infection, format ="%m")
months <- factor(months)[!is.na(months)]


barplot(table(months),
        main = "Number of infections in each month",
        names = month.abb,
        col ="red")
```



17d: parallel boxplt date_infection_onsets

```
linelist_messy_dates$date_infection_onset <-
  linelist_messy_dates$date_onset - linelist_messy_dates$date_infection

linelist_messy_dates$date_infection_onset <-
  as.double(linelist_messy_dates$date_infection_onset)

boxplot(linelist_messy_dates$date_infection_onset ~ factor(
                    format(linelist_messy_dates$date_infection,
                    "%m")),
        xlab = "Month",
        ylab = "Days between Infection and Onset",
        col = c("red","green", "blue", "purple", "orange"),
        border = "black",
        main = "Days Between Infection and Onset by Month",
        names = month.abb)
```



The graphic displays boxplots of date_infection_onset for each month. In this case, the grouping variable is the month in which date_onset occurred. It combines incubation periods by month and shows their boxplots. For example, from the graph for July, we can observe that the average incubation period was less than 10 days. However, for every month, there were incubation periods that exceeded the "expected" values. These are seen as outliers and are located outside the whiskers

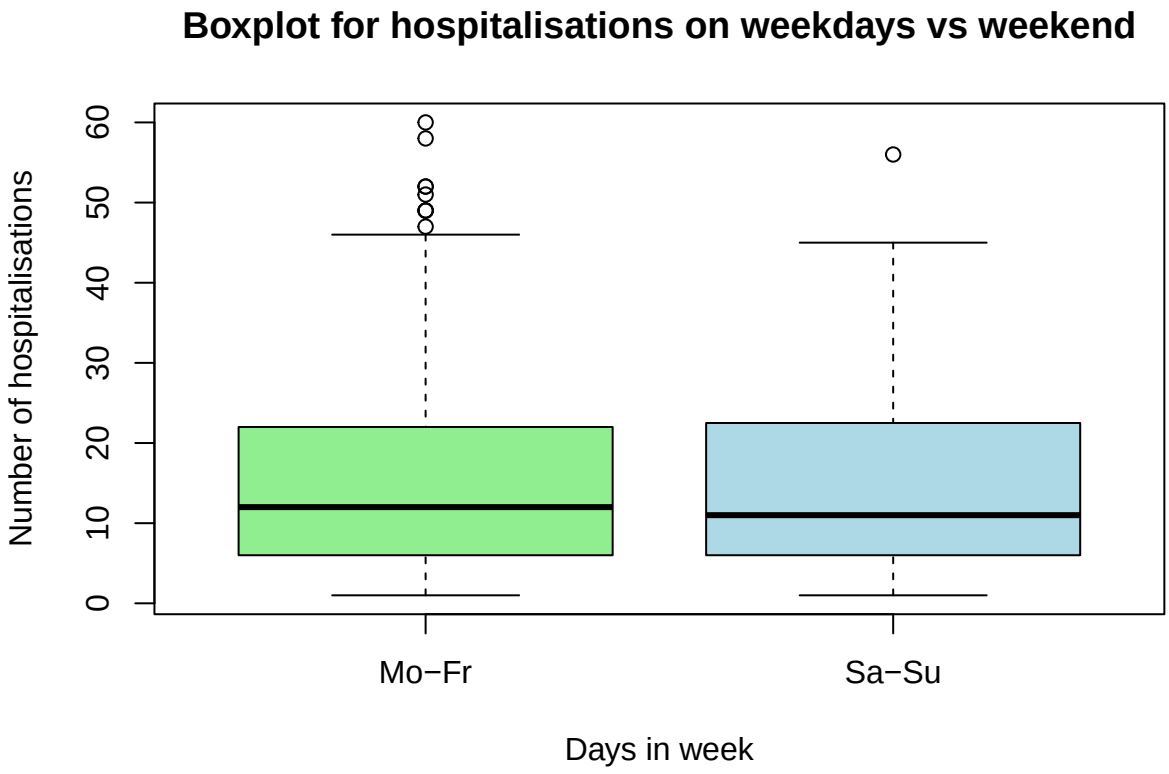
17e: daily number of hospitailsations:

```
#install.packages('lubridate')
library(lubridate)
```

```
##
## Attache Paket: 'lubridate'
```

```
## Die folgenden Objekte sind maskiert von 'package:base':
##
##     date, intersect, setdiff, union
```

```
daily_hospitalizations <-
  aggregate(linelist_messy_dates$case_id,list(linelist_messy_dates$date_hospitalisation),FUN = length)

daily_hospitalizations$week_days <-
  factor(ifelse(wday(daily_hospitalizations$Group.1) <= 5,"Mo-Fr", "Sa-Su"))

boxplot(daily_hospitalizations$x ~ daily_hospitalizations$week_days,
        main = "Boxplot for hospitalisations on weekdays vs weekend",
        xlab = "Days in week",
        ylab ="Number of hospitalisations",
        col = c("lightgreen", "lightblue"))
```



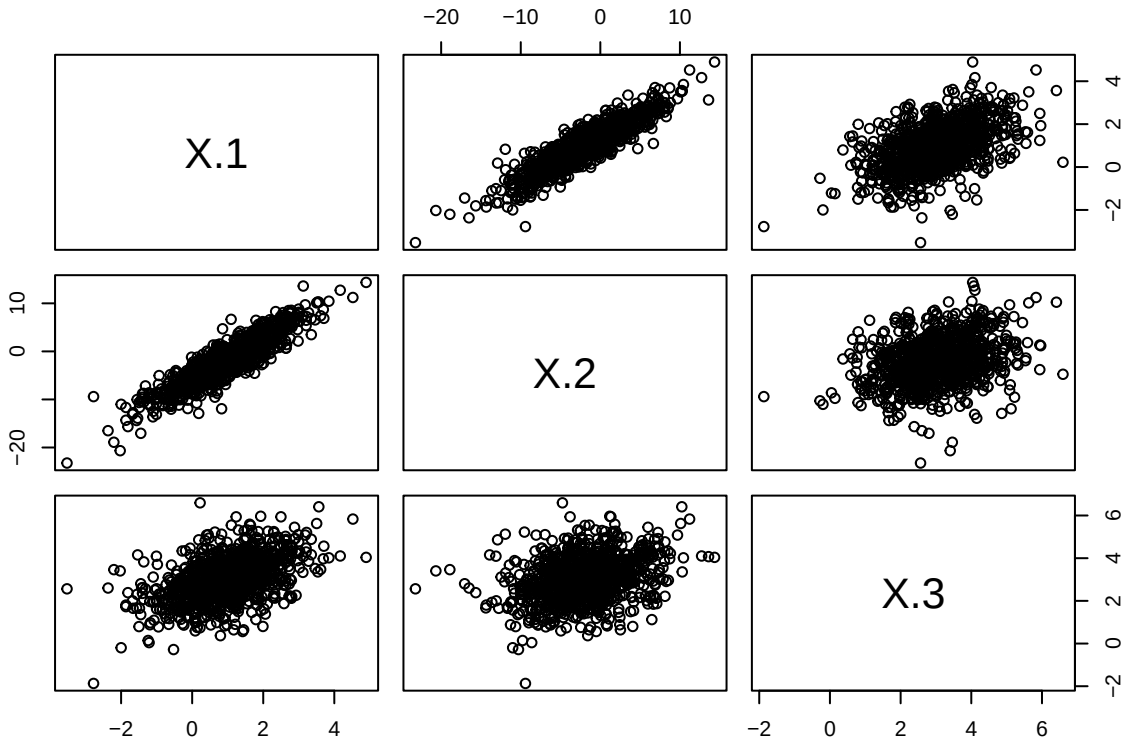
When you have two side-by-side boxplots, you can easily read the arithmetic mean from both. On average, on a workday, slightly more than 10 people get infected with Ebola. In comparison, the number of new infections on the weekend is slightly below 10. There are outliers that affect the data distribution. On workdays, these outliers are more numerous, while on the weekend, there is only one outlier beyond the whiskers.

Task 18:

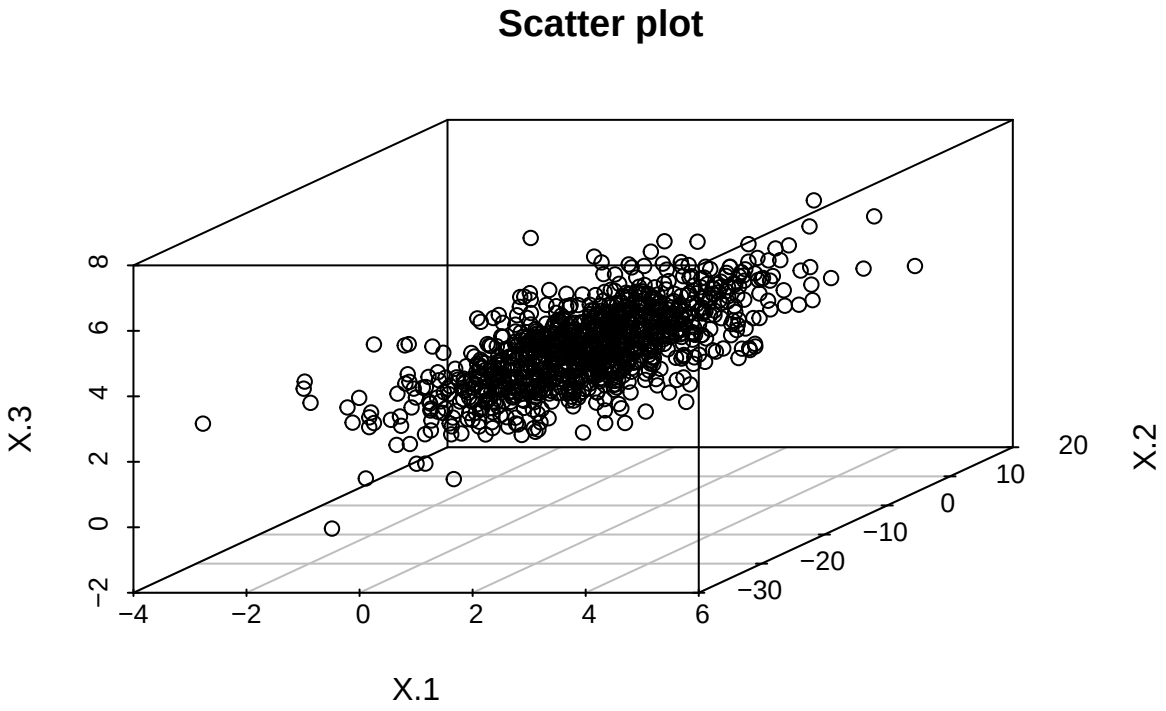
```
mvdo_data <- readRDS("Exercise_3/multivariate_data_outliers.rds")
```

18a: pairwise scatterplot:

```
library(scatterplot3d)
pairs(mvdo_data)
```



```r
scatterplot3d(mvdo_data, main ="Scatter plot")
```



18b: univariante outliers:

```
detect_outlier <- function(x) {
  Quantile1 <- quantile(x, probs=.25)
  Quantile3 <- quantile(x, probs=.75)
  IQR = Quantile3-Quantile1
  return (x > Quantile3 + (IQR*1.5) | x < Quantile1 - (IQR*1.5))
}
mvdo_data$univ_out <- 0
mvdo_data$univ_out <- ifelse(detect_outlier(mvdo_data[[1]]), 1, 0)
mvdo_data$univ_out[mvdo_data$univ_out == 0] <- ifelse(
detect_outlier(mvdo_data[[2]]), 2, 0)
```

```
## Warning in mvdo_data$univ_out[mvdo_data$univ_out == 0] <-
## ifelse(detect_outlier(mvdo_data[[2]]), : Anzahl der zu ersetzenden Elemente ist
## kein Vielfaches der Ersetzungslänge
```

```
mvdo_data$univ_out[mvdo_data$univ_out == 0] <- ifelse(
detect_outlier(mvdo_data[[3]]), 3, 0)
```

```
## Warning in mvdo_data$univ_out[mvdo_data$univ_out == 0] <-
## ifelse(detect_outlier(mvdo_data[[3]]), : Anzahl der zu ersetzenden Elemente ist
## kein Vielfaches der Ersetzungslänge
```

```
table(mvdo_data$univ_out)
```

```
##
##   0   1   2   3
## 963  13  13  11
```

```
sum(detect_outlier(mvdo_data$X.1))
```

```
## [1] 13
```

```
sum(detect_outlier(mvdo_data$X.2))
```

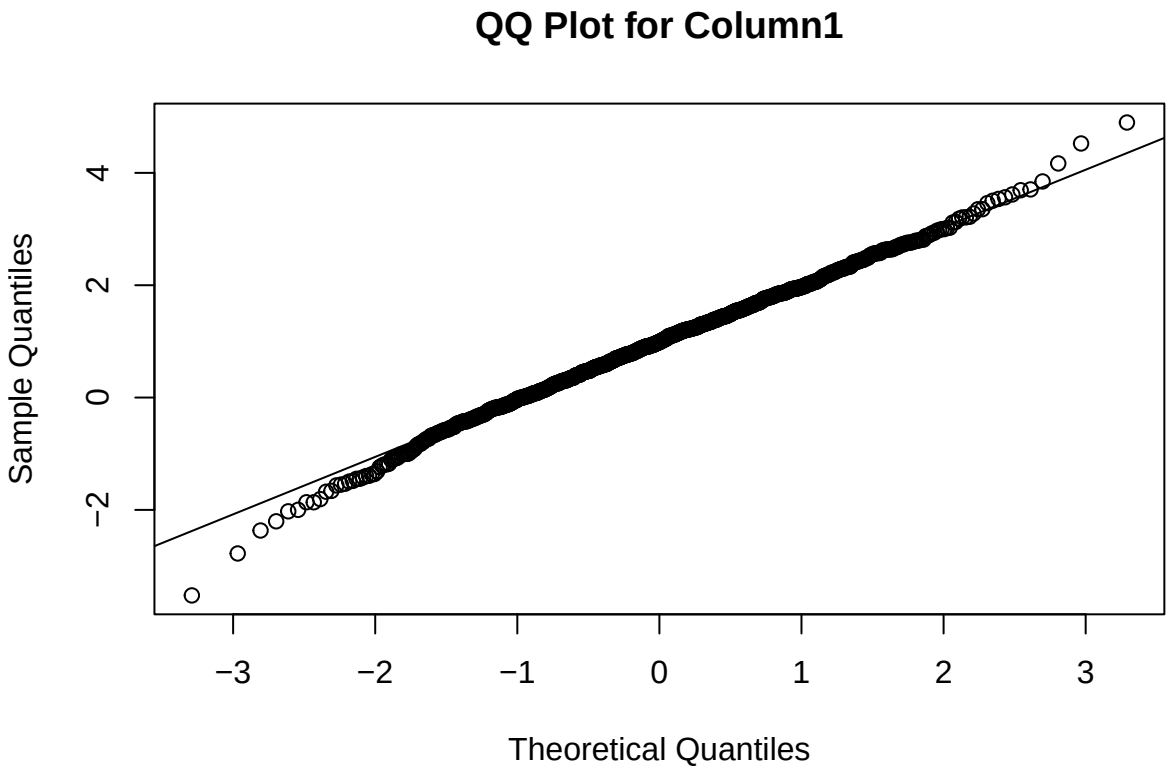
```
## [1] 13
```

```
sum(detect_outlier(mvdo_data$X.3))
```

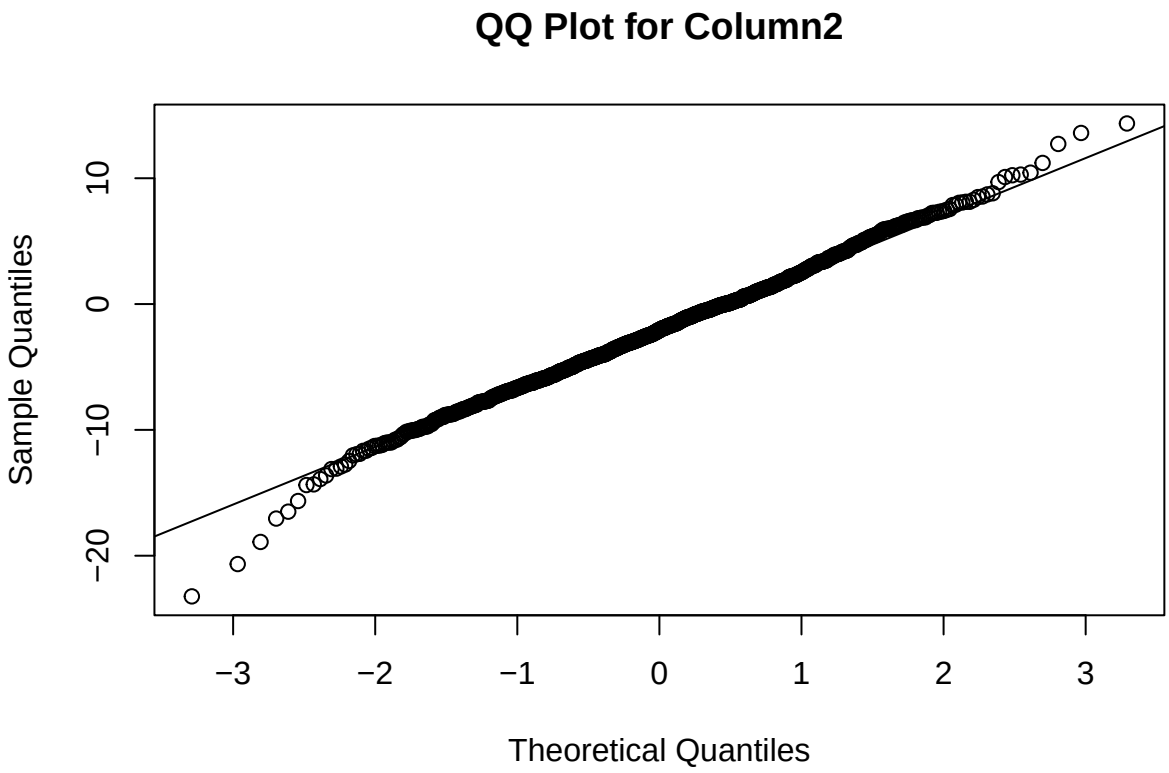
```
## [1] 11
```

18c: check normality

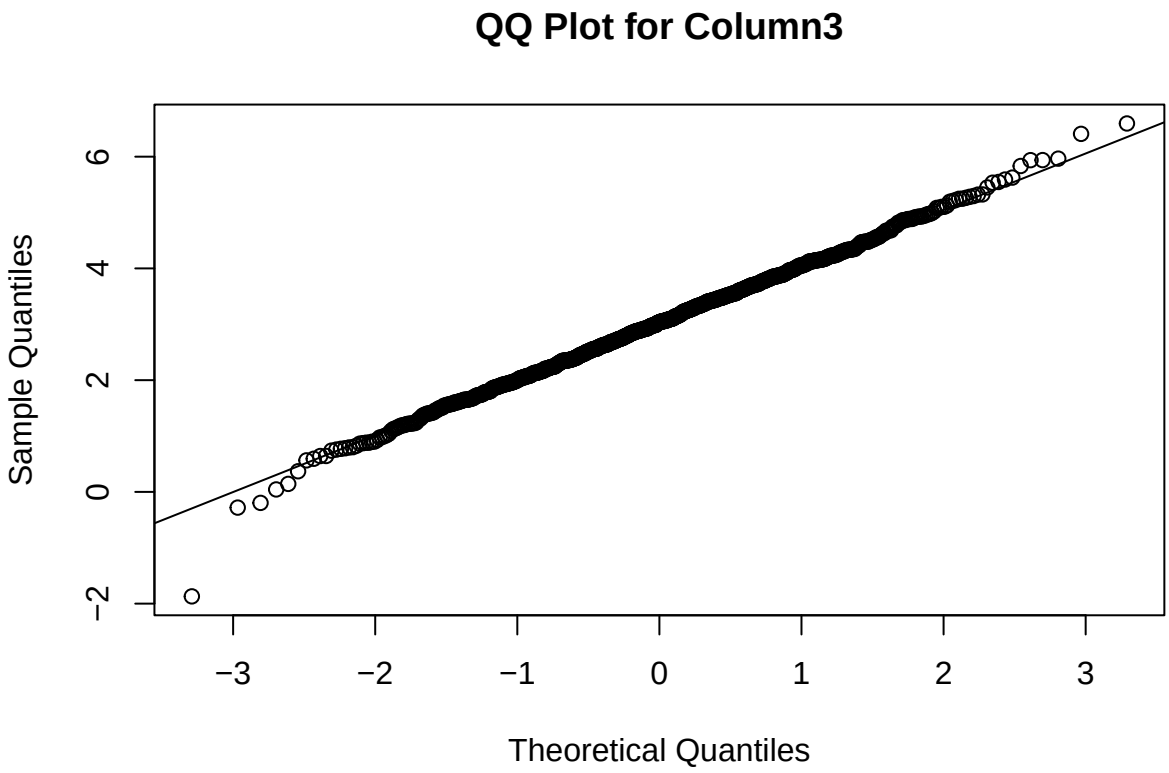
```
# Check normality for Column1 (replace with the actual column name)
qqnorm(mvdo_data$X.1, main = "QQ Plot for Column1")
qqline(mvdo_data$X.1)
```



```r
# Check normality for Column2 (replace with the actual column name)
qqnorm(mvdo_data$X.2, main = "QQ Plot for Column2")
qqline(mvdo_data$X.2)
```



```r
# Check normality for Column3 (replace with the actual column name)
qqnorm(mvdo_data$X.3, main = "QQ Plot for Column3")
qqline(mvdo_data$X.3)
```



18d: Mahalanobis distance:

```
# MD^2:
mvdo_data$MD2 <- mahalanobis(
  mvdo_data[, 1:3],
  colMeans(mvdo_data[, 1:3]),
  cov(mvdo_data[, 1:3]))

# chi squared distribution
chi_squared_distribution <- qchisq(p = 0.975, df = 3)
cat("chi squared distribution:", chi_squared_distribution, "\n")
```

```
## chi squared distribution: 9.348404
```

```
# MD_out:
mvdo_data$MD_out <- ifelse(mvdo_data$MD2 > chi_squared_distribution, 1, 0)
cat("MD^2:", sum(mvdo_data$MD_out[mvdo_data$MD_out == 1]))
```

```
## MD^2: 34
```

> Quote: The mahalanobis function in R calculates the Mahalanobis distance between a set of data points and a reference point (typically the mean vector) using a given covariance matrix. The Mahalanobis distance is a measure of the distance between a point in multivariate space and a distribution of points.
>
> Quote: The qchisq function in R is used to calculate the quantiles of the chi-squared distribution.

18e: robust estimations:

```
# Load the required library
library("MASS")

# Your data matrix (replace 'x' with your actual data)
x <- mvdo_data

# Number of variables
p <- 3

# Calculate the robust covariance matrix using the MCD estimator
robust_cov <- cov.mcd(x[, 1:p])$cov

# Calculate the robust mean using the median
robust_mean <- apply(x[, 1:p], 2, median)

# Save into MD 2 robust
mvdo_data$MD2rob <- mahalanobis(mvdo_data[, 1:3], robust_mean, robust_cov)

# Calculate the chi squared distribution
chi_squared_distribution <- qchisq(p =0.975, df = 3)
cat("chi squared distribution:", chi_squared_distribution, "\n")
```

```
## chi squared distribution: 9.348404
```

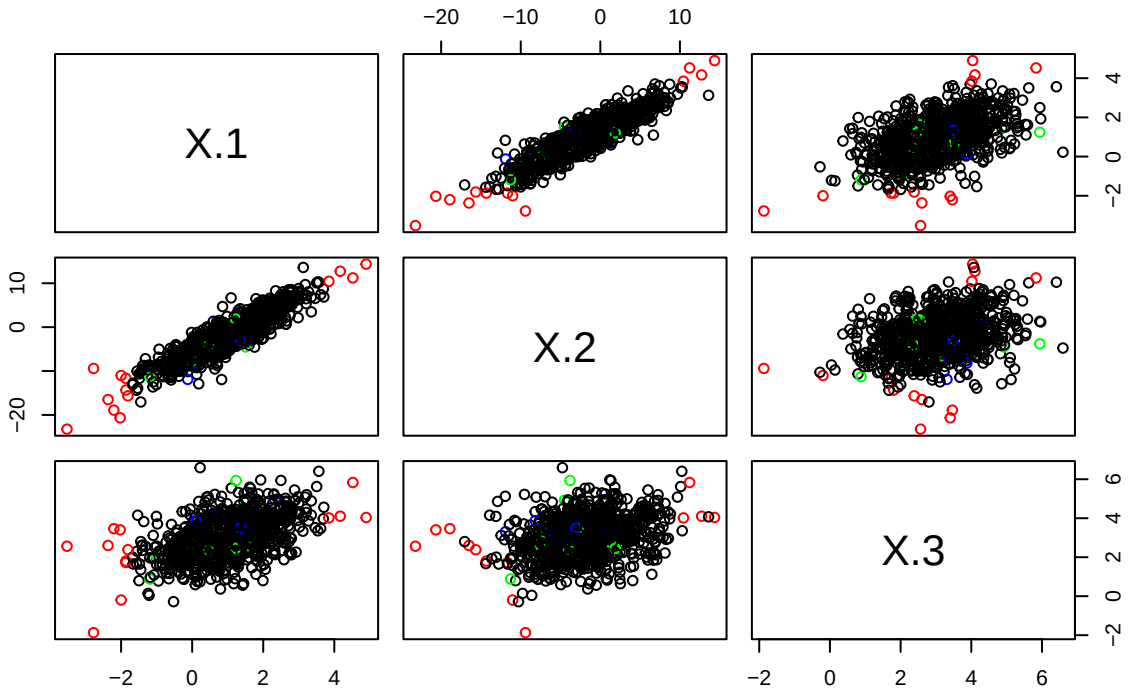
```
mvdo_data$MDrob_out <- ifelse(mvdo_data$MD2rob > chi_squared_distribution, 1, 0)
cat("MD^2:",sum(mvdo_data$MDrob_out[mvdo_data$MDrob_out == 1]))
```

```
## MD^2: 53
```

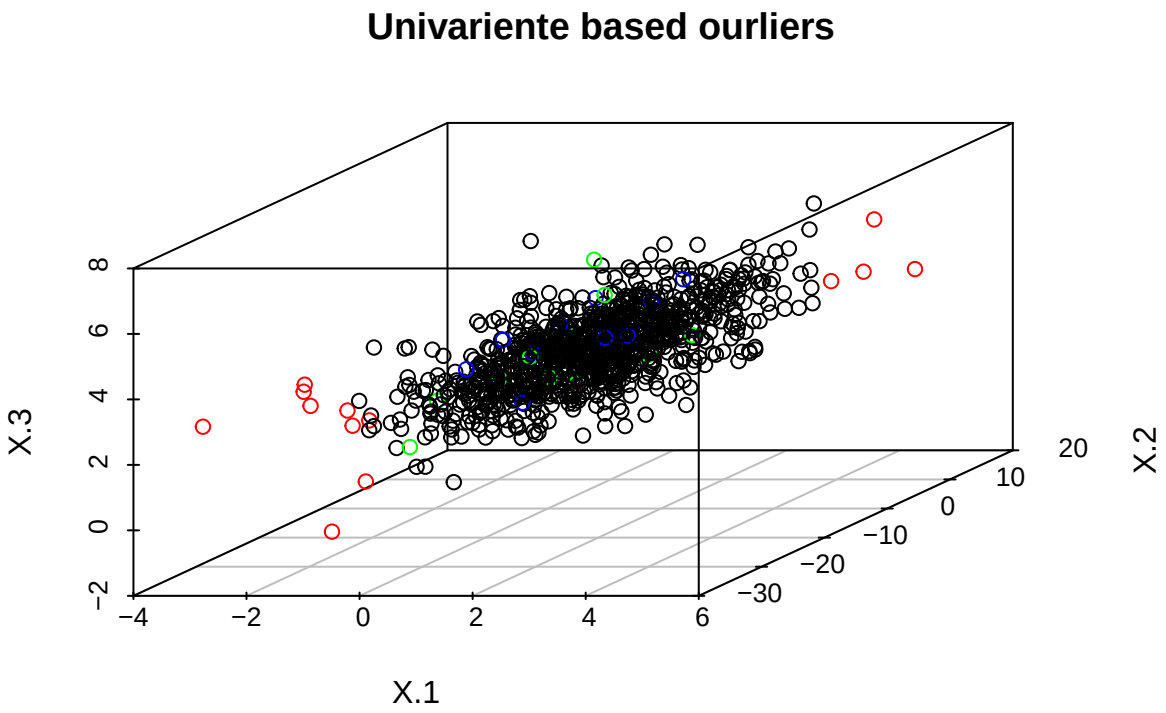
18f: Visualize different Outliers:

18f.1: univariant outliers

```
# one pairwise and one 3D scatterplot where you color the univariate outliers with
#3 different colors (for 1, 2, 3)
color <- 1
color[mvdo_data$univ_out == 0] <- "black"
color[mvdo_data$univ_out == 1] <- "red"
color[mvdo_data$univ_out == 2] <- "green"
color[mvdo_data$univ_out == 3] <- "blue"
pairs(mvdo_data[1:3],
      col = color,
      main = "Univariente based ourliers")
```

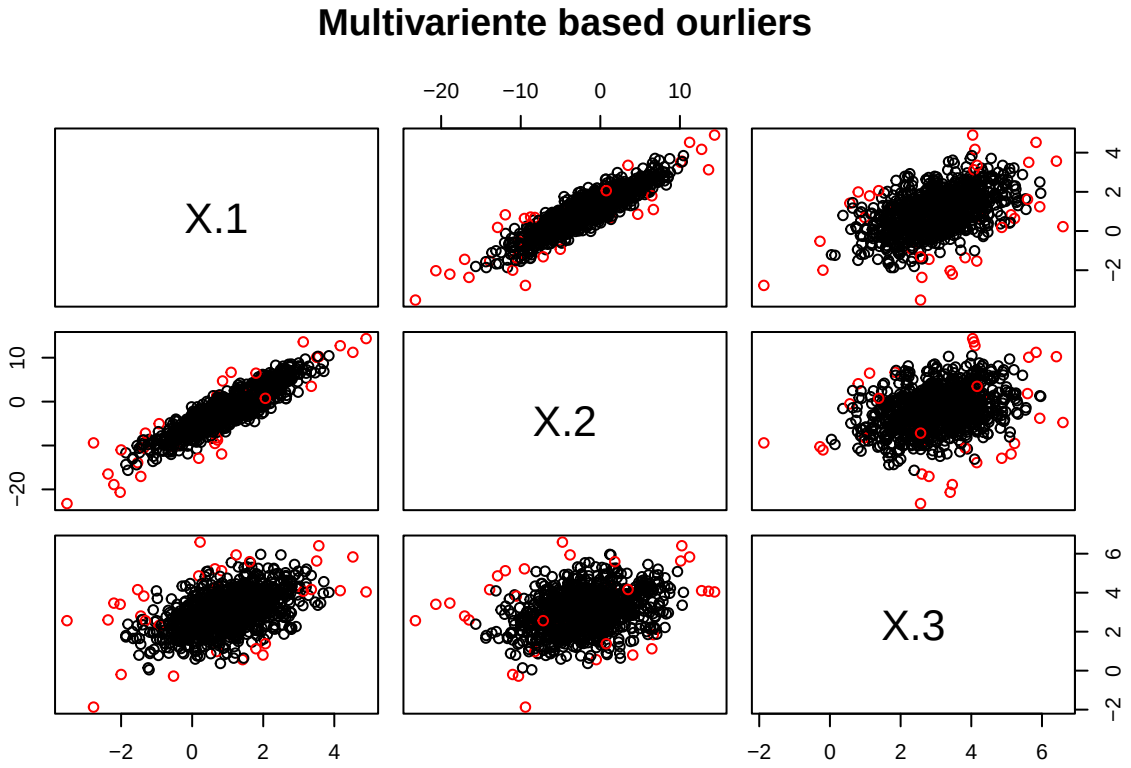


```r
scatterplot3d(mvdo_data[1:3],
              color = color,
              main = "Univariente based ourliers")
```

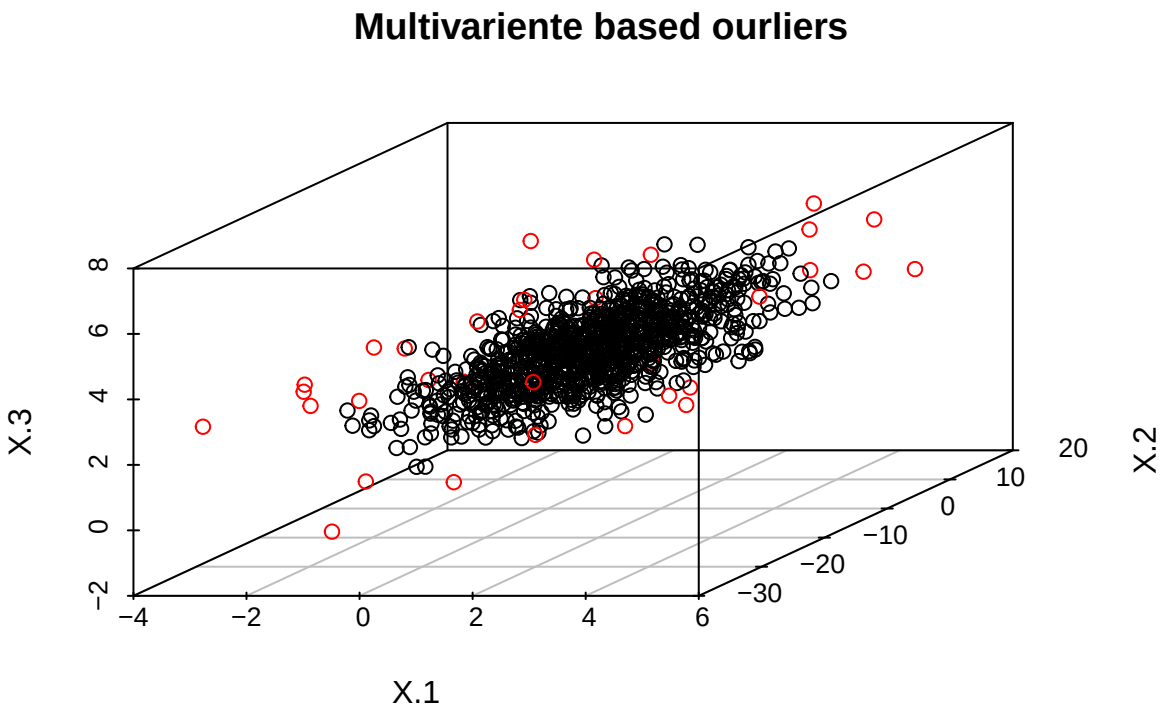


18f.2: multivariant outliers

```
pairs(mvdo_data[1:3],col = ifelse(mvdo_data$MD_out == 1, "red","black"),
              main = "Multivariente based ourliers")
```

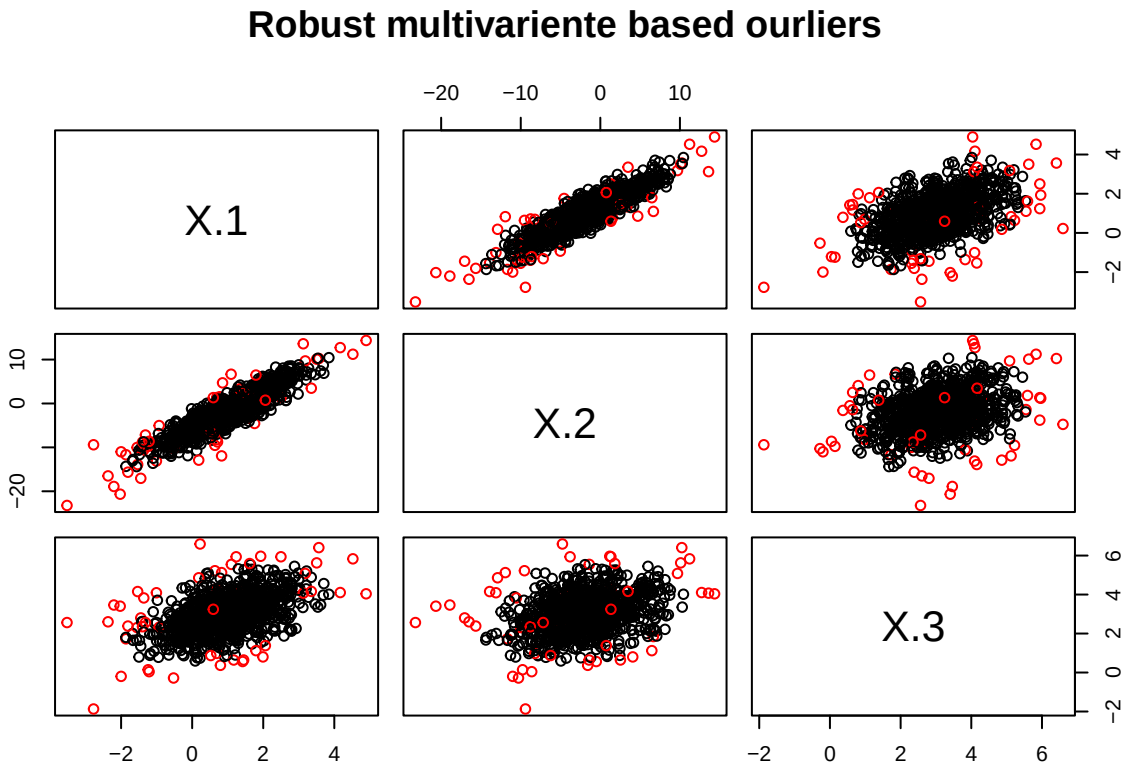


```r
scatterplot3d(mvdo_data[1:3],
              color = ifelse(mvdo_data$MD_out == 1, "red","black"),
              main = "Multivariente based ourliers")
```

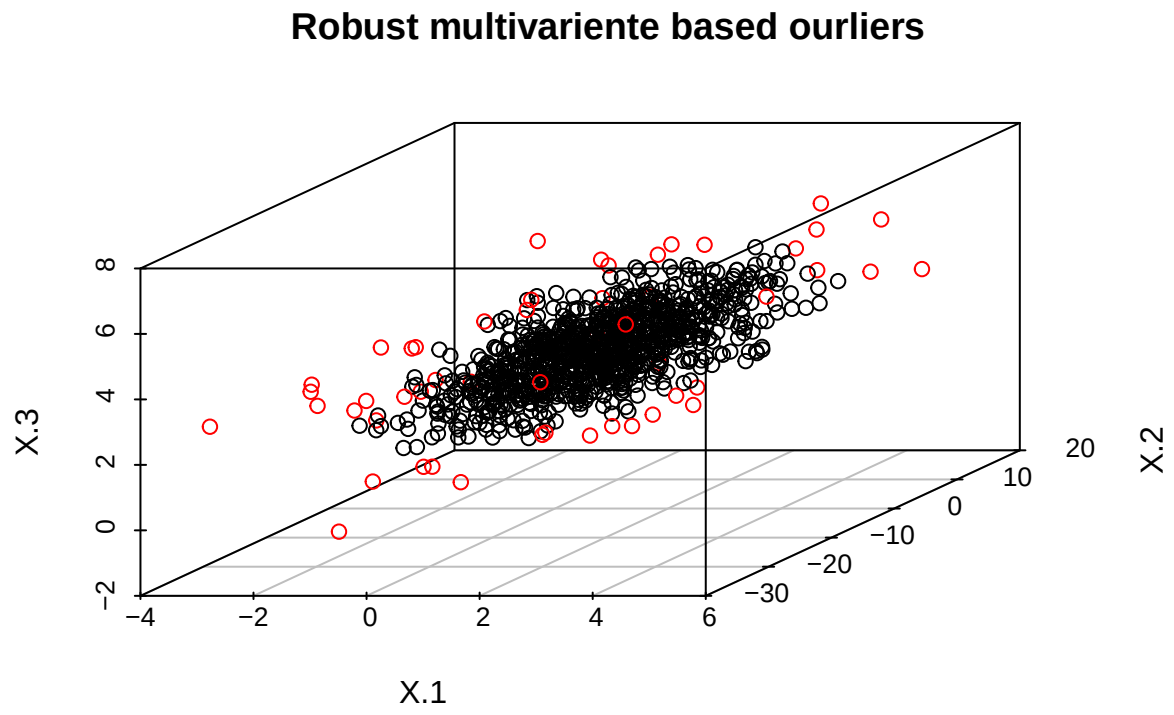


18f.3: robust multivariant outliers

```
pairs(mvdo_data[1:3],
      col = ifelse(mvdo_data$MDrob_out == 1, "red","black"),
      main = "Robust multivariente based ourliers")
```



```
scatterplot3d(mvdo_data[1:3],
              color = ifelse(mvdo_data$MDrob_out == 1, "red","black"),
              main = "Robust multivariente based ourliers")
```



18f.4: Interpretation of results:

We have to use length and sum to get the total number, since a few points are being covered by other dots in the Scatter plot.

```
cat("Number of Univariante based outliers:",
    length(mvdo_data$univ_out[mvdo_data$univ_out != 0]),
    "\n")
```

```
## Number of Univariante based outliers: 37
```

```
cat("Number of Multiivariante based outliers:"
    ,sum(mvdo_data$MD_out[mvdo_data$MD_out == 1]),
    "\n")
```

```
## Number of Multiivariante based outliers: 34
```

```
cat("Number of Robust multivariante based outliers:",
    sum(mvdo_data$MDrob_out[mvdo_data$MDrob_out == 1]),
    "\n")
```

```
## Number of Robust multivariante based outliers: 53
```

Number of outliers	Description
37	These are the outliers that have been detected using univariate methods, meaning each variable in the dataset is considered individually without considering their relationships with other variables. It indicates that there are 37 data points that are considered outliers in at least one variable.
32	These are the outliers detected using multivariate methods, which take into account the relationships and interactions between multiple variables simultaneously. It suggests that there are 34 data points that are considered outliers when considering all variables together.
51	These are the outliers detected using robust multivariate methods, which are designed to handle outliers more effectively than traditional methods. It suggests that there are 51 data points that are considered outliers when using a robust multivariate approach.

Note that the results may vary because the matrix is randomly generated. If you want to compare the results, set a seed!

18g: difference Euclidiean & Mahalanobis distance

The difference is, that Euclidean distance treats all variables as equally important and doesn't account for correlations or variable variances, making it sensitive to data with different scales and variances. On the other hand, Mahalanobis distance is normally used for multivariate normally distributed data because it considers the covariance structure, adjusting for variable correlations and variances. It provides a more accurate measure of distance, especially when dealing with data that follows an elliptical shape.

I would say, for robust outlier detection in such cases (If you know that beforehand), Mahalanobis distance is the go-to choice.

Feedback:

Task 14h: The mean values should be displayed by a line plot. -1

Task 16c: Your Andrew curves does not look correct. -4

Task 18b: Interesting note

Great work!