

Homework assignment 3

Vseslav Levchenko

2023-10-30

Contents

Task 14	2
a	2
b	2
c	2
d	3
e	3
f	3
g	3
h	4
Task 15	5
a	5
b	6
c	6
d	6
Task 16	6
a	6
b	7
c	9
Task 17	10
a	10
b	10
c	11
d	11
e	12
Task 18	13
a	13
b	15
c	16
d	17
e	17
f	18
g	23

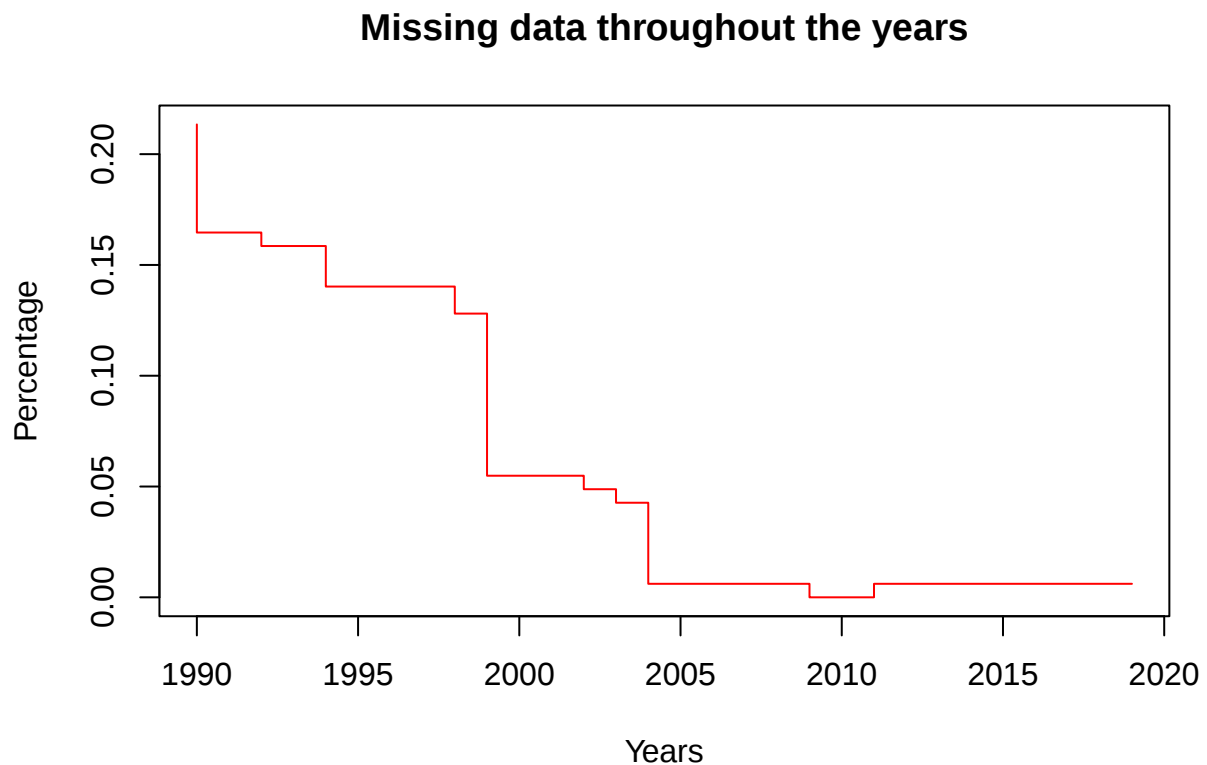
Task 14

a

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

b

```
missing <- sapply(X = sdi[, -1], FUN = function (x) sum(is.na(x)))  
percentages <- missing/length(sdi$country)  
#percentages  
  
plot(1990:2019, percentages, type = "S",  
     xlab = "Years", ylab = "Percentage", col="red",  
     main = "Missing data throughout the years")
```



c

```
complete <- sum(complete.cases(sdi))/length(sdi$country)  
complete
```

```
## [1] 0.7865854
```

d

```
missings_per_country <- rowSums(is.na(sdi))
names(missings_per_country) <- sdi$country

sort(missings_per_country[missings_per_country>0], decreasing = TRUE)
```

##	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	

e

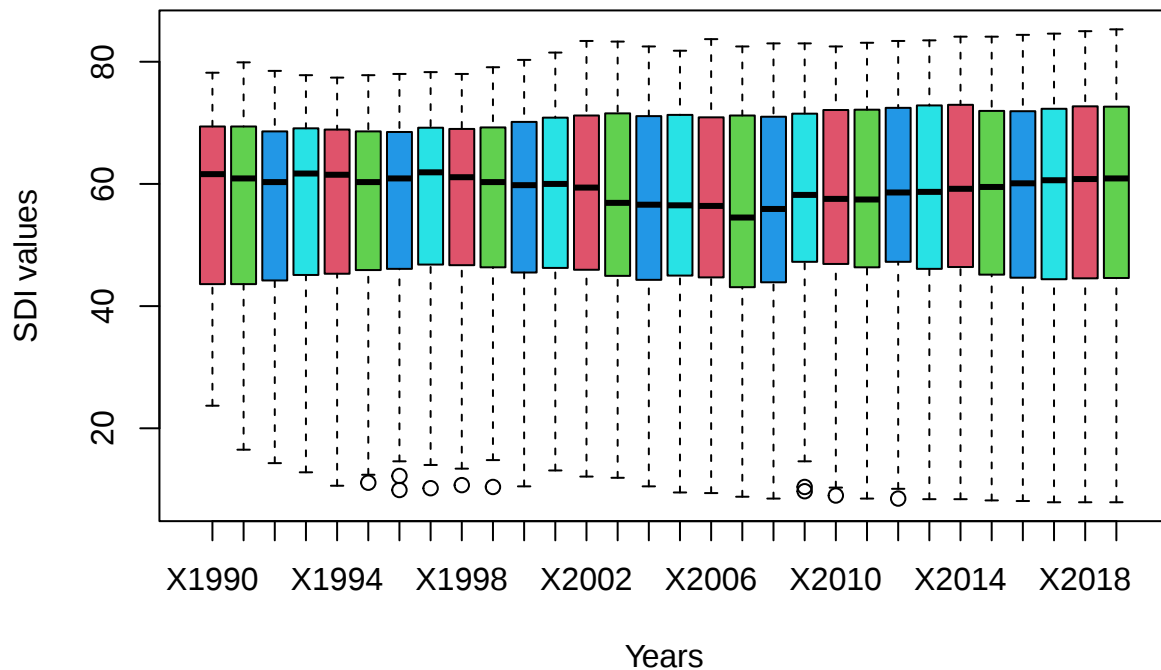
```
library(tidyr)
sdi_long <- gather(sdi, key="year", value = "Value", -country)
#sdi_long
```

f

```
sdi_long <- sdi_long[complete.cases(sdi_long),]
#sdi_long
```

g

```
boxplot(sdi[, -1], use.cols=TRUE, col=2:5,
        xlab="Years", ylab="SDI values")
```



The box in the boxplot represents the interquartile range (IQR) of the SDI values for each year. The height of the box indicates the spread of the middle 50% of the data.

The black line represents the median of the SDI values for each year.

The whiskers extend from the edges of the box to the minimum and maximum values within a certain range.

Outliers are individual data points that fall beyond the whiskers. They are shown as individual points on the plot.

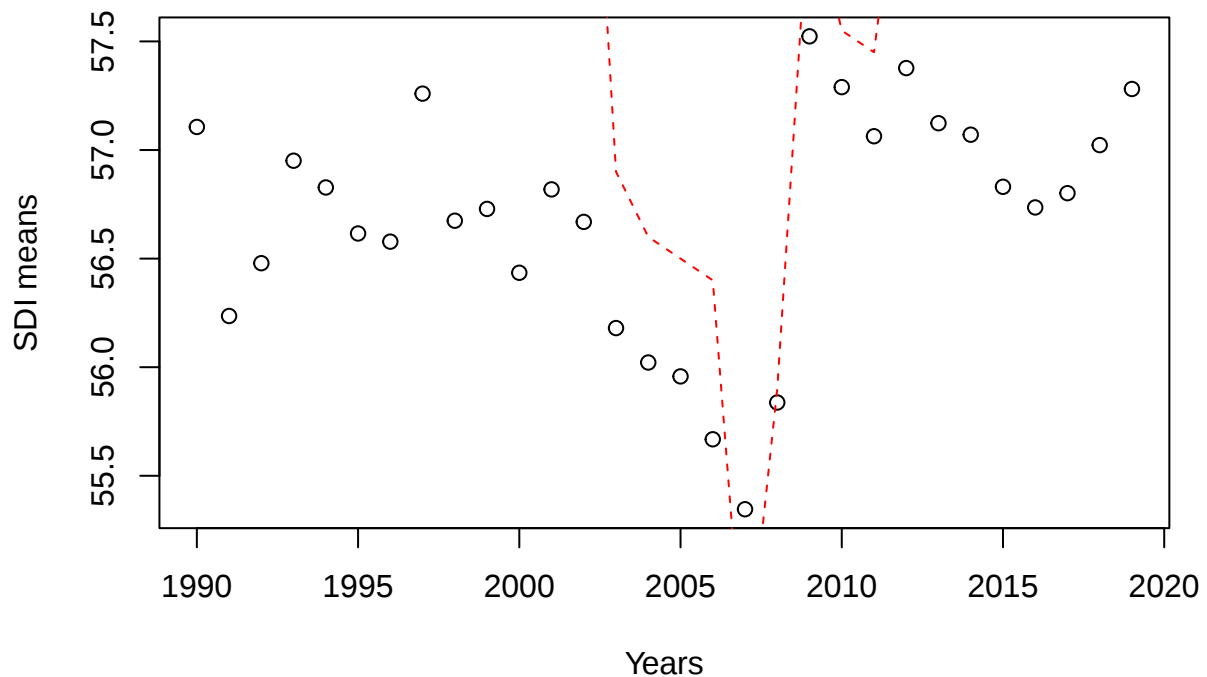
h

```
sdi_means <- with(sdi_long,
                  tapply(sdi_long$Value, sdi_long$year, mean))

sdi_medians <- with(sdi_long,
                   tapply(sdi_long$Value, sdi_long$year, median))

plot(1990:2019, sdi_means,
     xlab="Years", ylab="SDI means",
     main="SDI means throughout the years")
lines(1990:2019, sdi_medians,
      lty="dashed", col="red")
```

SDI means throughout the years



Task 15

```
movies1 <- read.csv("movies1.csv")
movies2 <- read.table("movies2.csv", sep = ";", header = TRUE)
ratings <- read.csv("ratings.csv")

#movies1; movies2; ratings
```

a

```
user_ratings <- merge(movies1, ratings, by = c("movieId"), all.x=TRUE)
users <- unique(ratings$userId)
sum(!(users %in% user_ratings$userId))

## [1] 15

user_ratings_all <- merge(movies1, ratings, by = c("movieId"), all.y=TRUE)
dim(user_ratings_all)

## [1] 100836      6

dim(user_ratings)

## [1] 16420      6
```

b

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

c

```
m_r <- merge(movies, ratings, by = c("movieId"), all.x=TRUE)
newset <- m_r[!complete.cases(m_r),]
length(unique(newset$movieId))
```

```
## [1] 18
```

d

```
write.table(user_ratings, "movies1.txt",
            sep = ";", na = "999999", col.names = FALSE)
```

Task 16

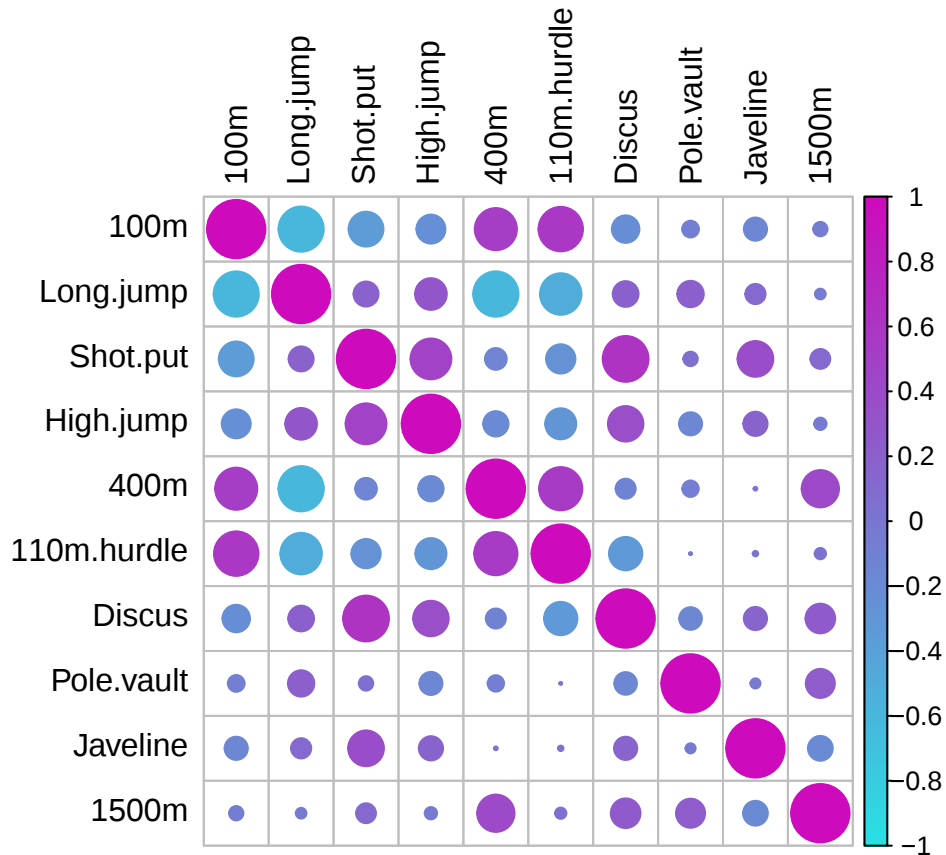
```
library(FactoMineR)
data("decathlon")
```

a

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
corrplot::corrplot(cor(decathlon[, 1:10]),
                    method = "circle", tl.col = "black",
                    col = colorRampPalette(c(5:6))(200))
```



```
correlations <- cor(decathlon[, 1:10])
```

1 is a perfect positive correlation, -1 is a perfect negative correlation 0 is no correlation Color indicates the strength and direction of correlation

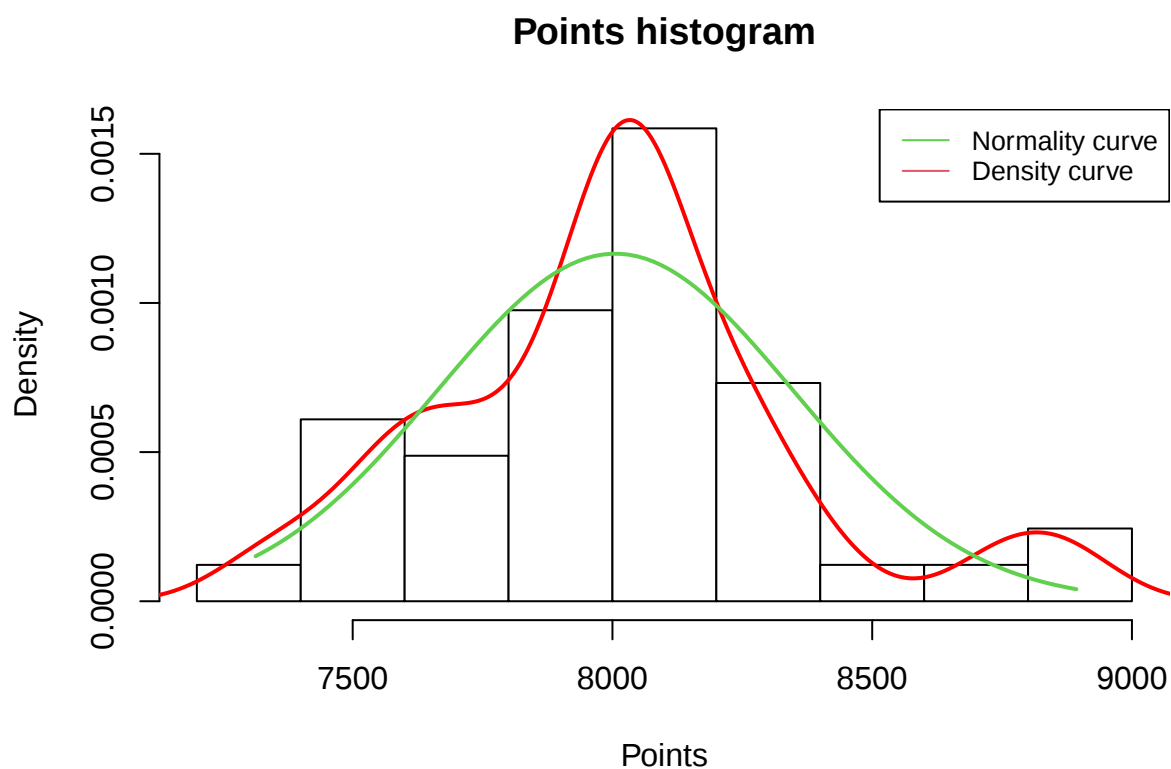
b

```
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="white", prob=TRUE,
     main = "Points histogram", xlab = "Points")

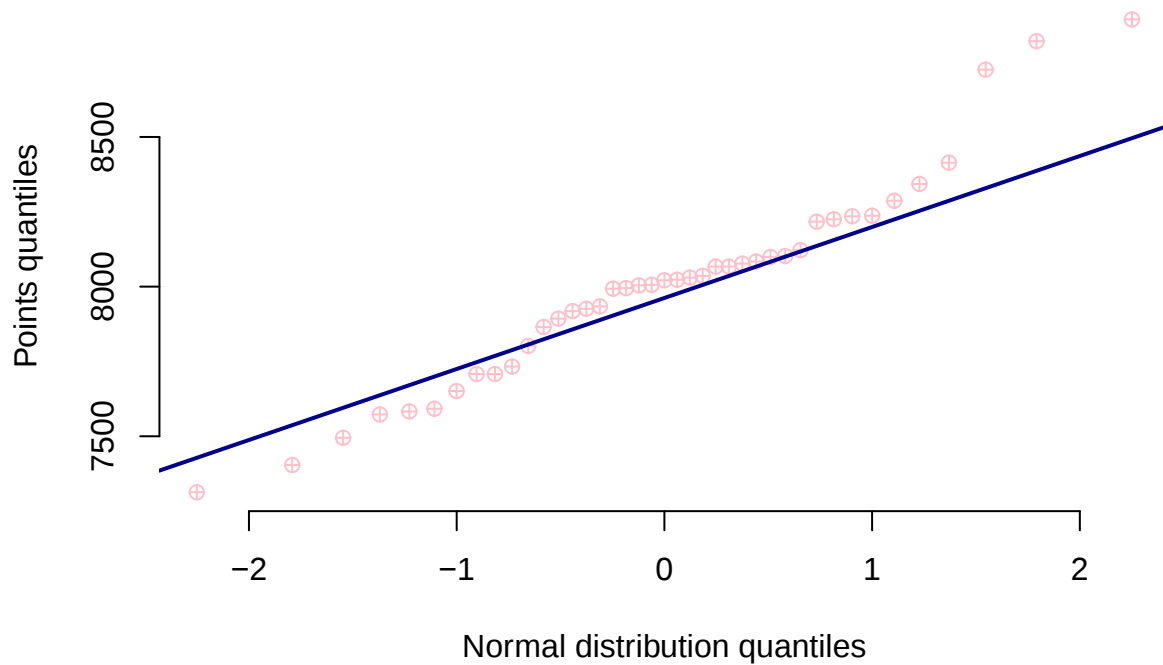
lines(density(decathlon$Points), lwd=2, col="red")
lines(x2, fun, col=3, lwd=2)

legend("topright", legend = c("Normality curve", "Density curve"),
     col = c(3,2), lty = 1, cex = 0.8)
```



```
qqnorm(decathlon$Points, pch=10, col="pink", frame=FALSE,  
        xlab = "Normal distribution quantiles",  
        ylab = "Points quantiles")  
qqline(decathlon$Points, col="darkblue", lwd=2)
```

Normal Q-Q Plot



c

```
#normalizing function
normalize <- function(x, na.rm=TRUE) {
  return((x-min(x)) / (max(x)-min(x)))
}

#Andrew's curves function
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))
}

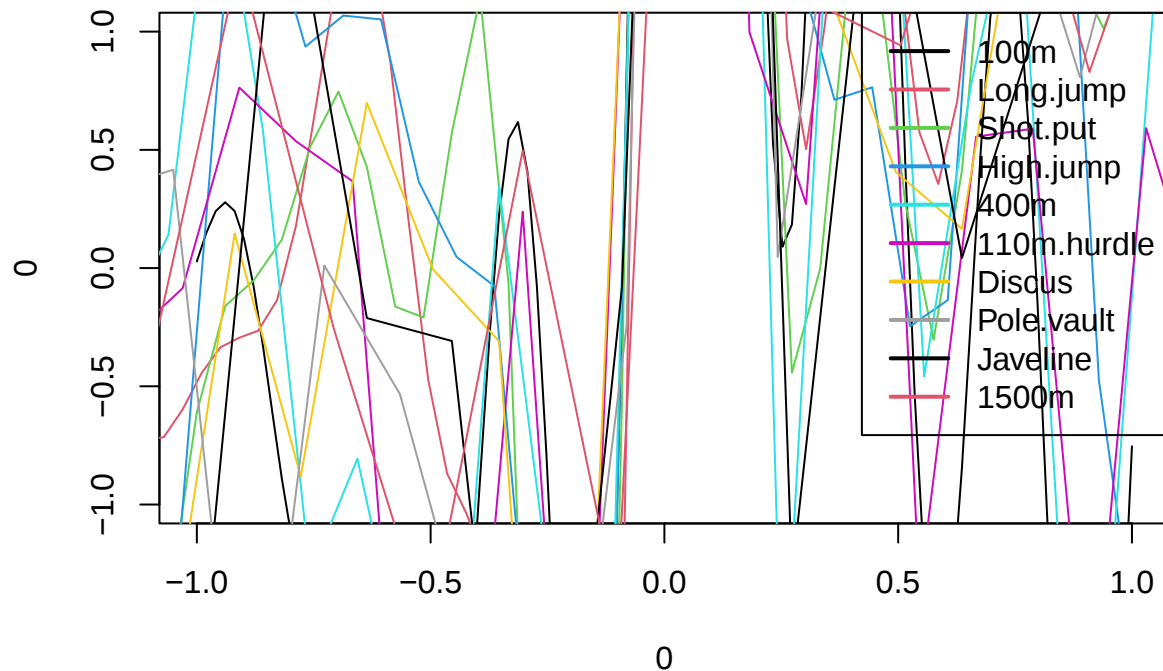
decathlon[, 1:10] <- lapply(decathlon[, 1:10], normalize)
decathlon$Competition <- factor(decathlon$Competition)
plot(0, 0, type="n")

for (i in 1:10) {
  tseq <- seq(-i, i, length=100)
  #dim(tseq) <- length(tseq)
  y <- sapply(tseq, function(t) f_t(t, decathlon[, i]))
}
```

```

    lines(tseq, y, col=i)
  }
  legend("topright", legend = colnames(decathlon)[1:10], col = 1:10, lwd = 2)

```



Task 17

a

```

linelist_messy_dates <- readRDS("linelist_messy_dates.rds")
#linelist_messy_dates

```

The dataset appears to have cases of an infection - their ID, generation and dates of infection, onset and hospitalization, all in different formats, hence “messy dates”.

b

```

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, format = "%y/%B/%d")
linelist_messy_dates$date_hospitalisation <-
  as.Date(as.numeric(linelist_messy_dates$date_hospitalisation), origin = "1970-01-01")
class(linelist_messy_dates$date_outcome)

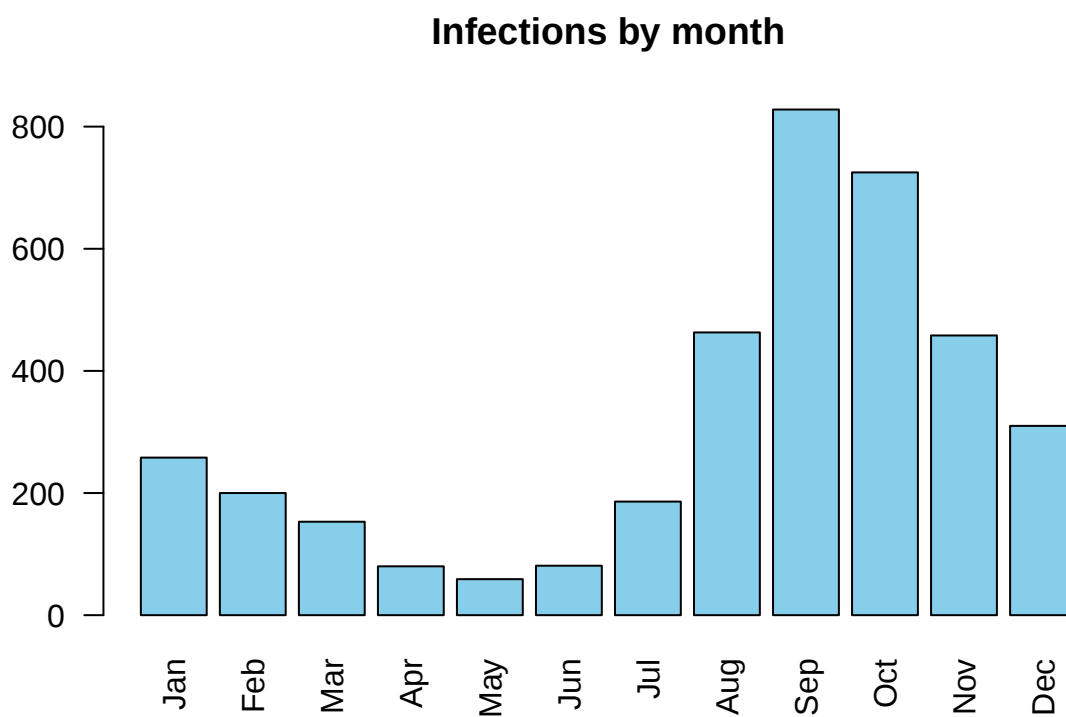
```

```
## [1] "Date"
```

c

```
months <- format(linelist_messy_dates$date_infection, format="%b")
months <- factor(months, levels=month.abb)
months <- months[!is.na(months)]
month_infections <- table(months)

barplot(month_infections, las=2,
        main = "Infections by month",
        col = "skyblue")
```



d

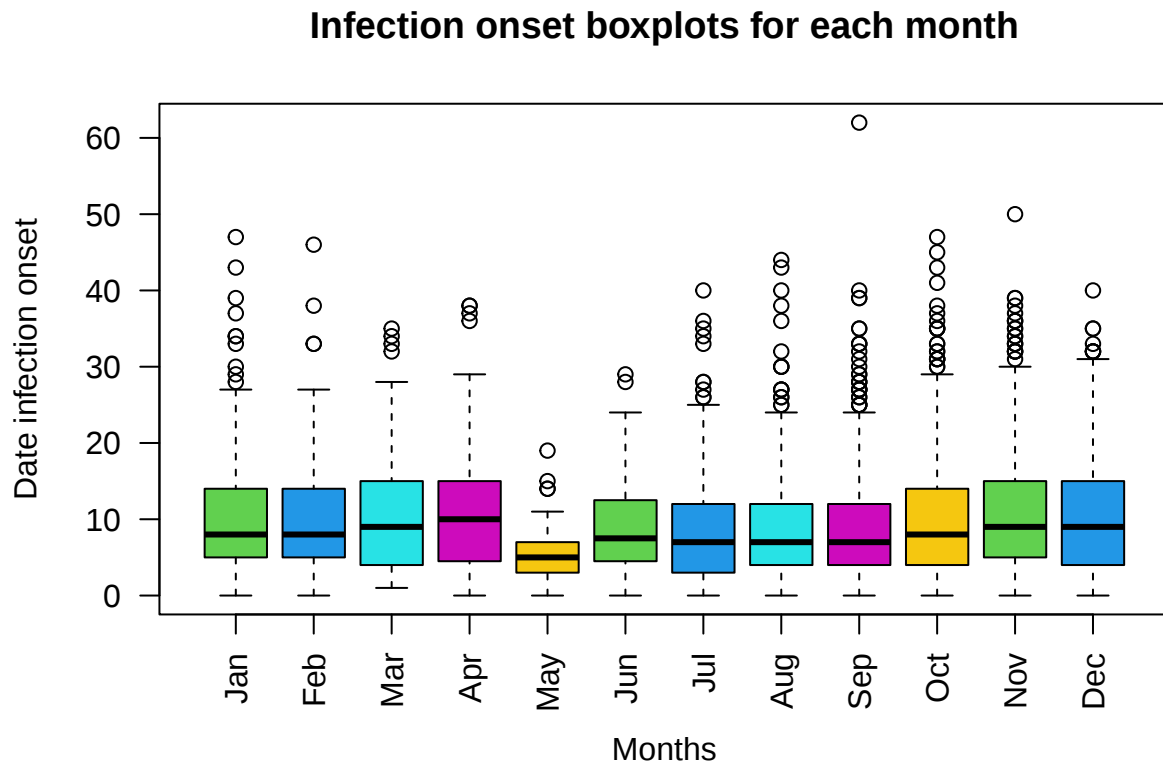
```
linelist_messy_dates$date_infection_onset <-
  linelist_messy_dates$date_onset - linelist_messy_dates$date_infection

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

month_name <- month.abb

boxplot(linelist_messy_dates$date_infection_onset
        ~ as.POSIXlt(linelist_messy_dates$date_onset)$mon,
```

```
names = month_name,
main = "Infection onset boxplots for each month",
xlab = "Months",
ylab = "Date infection onset",
col = 3:7, las = 2)
```



The boxplots show incubation time of the infection, although with outliers. The rest was already explained in task 14.

e

```
library(lubridate)

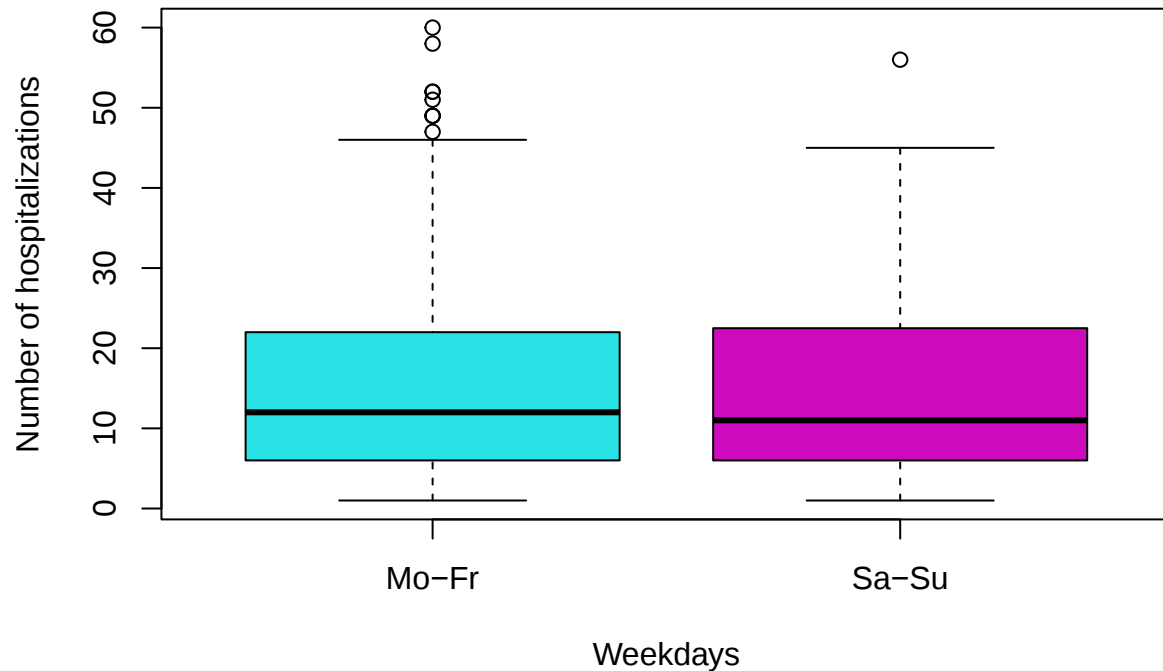
##
## Attaching package: 'lubridate'
## The following objects are masked from 'package:base':
##
##   date, intersect, setdiff, union

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

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

```
boxplot(daily_hospitalizations$x ~ daily_hospitalizations$weekdays,
        main = "Boxplot for hospitalizations on weekdays vs weekend",
        xlab = "Weekdays",
        ylab = "Number of hospitalizations",
        col = c(5,6))
```

Boxplot for hospitalizations on weekdays vs weekend

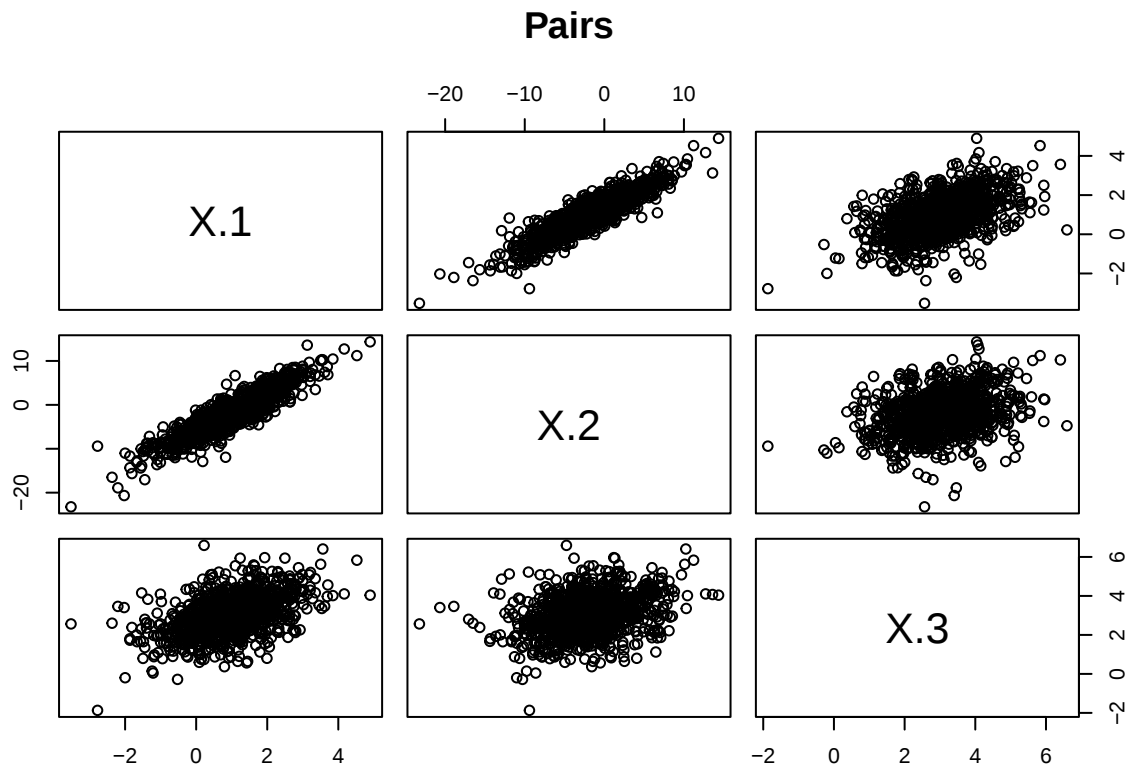


Task 18

```
library(scatterplot3d)
mvar_data <- readRDS("multivariate_data_outliers.rds")
```

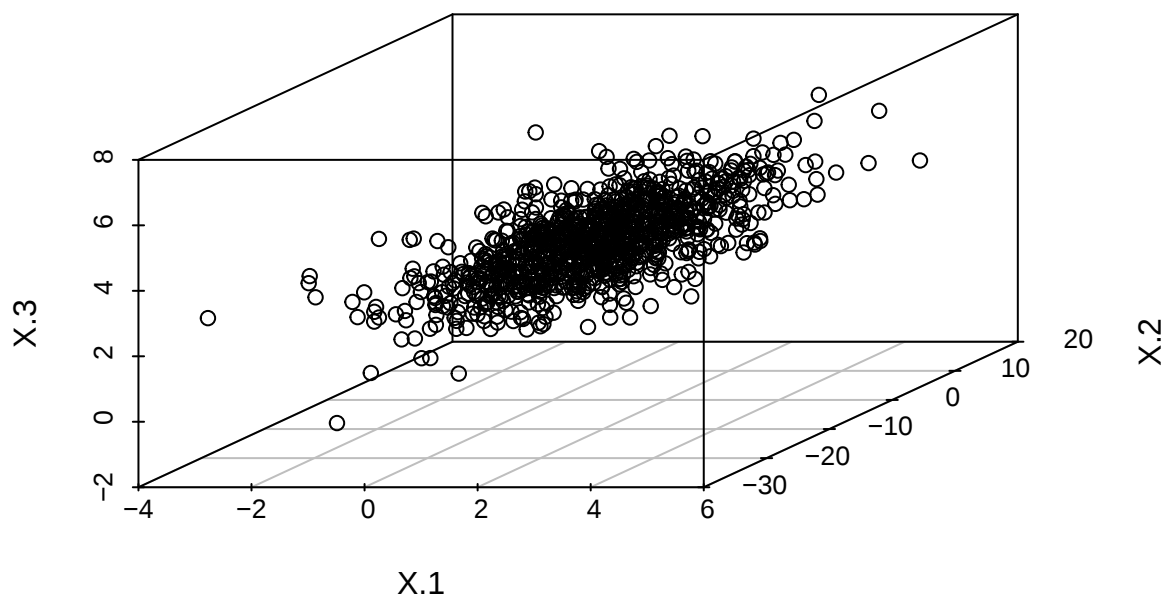
a

```
pairs(mvar_data, main="Pairs")
```



```
scatterplot3d(mvar_data, main="Scatter plot")
```

Scatter plot



b

```
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))
}

mvar_data$univ_out <- 0
mvar_data$univ_out <- ifelse(detect_outlier(mvar_data[[1]]), 1, 0)

mvar_data$univ_out[mvar_data$univ_out == 0] <- ifelse(
  detect_outlier(mvar_data[[2]]), 2, 0)

## Warning in mvar_data$univ_out[mvar_data$univ_out == 0] <-
## ifelse(detect_outlier(mvar_data[[2]]), : number of items to replace is not a
## multiple of replacement length
mvar_data$univ_out[mvar_data$univ_out == 0] <- ifelse(
  detect_outlier(mvar_data[[3]]), 3, 0)

## Warning in mvar_data$univ_out[mvar_data$univ_out == 0] <-
## ifelse(detect_outlier(mvar_data[[3]]), : number of items to replace is not a
## multiple of replacement length
```

```
table(mvar_data$univ_out)
```

```
##
```

```
##    0    1    2    3
```

```
## 963  13  13  11
```

```
sum(detect_outlier(mvar_data$X.1)); sum(detect_outlier(mvar_data$X.2)); sum(detect_outlier(mvar_data$X.3))
```

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

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

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

c

```
library("car")
```

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

```
par(mfrow = c(2,2))
```

```
qqPlot(mvar_data$X.1, xlab = "Quantiles from normal distribution",  
       ylab = "X.1",  
       main = "QQplot x1")
```

```
## [1] 245 113
```

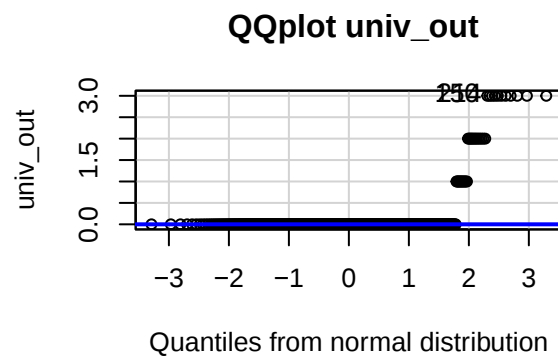
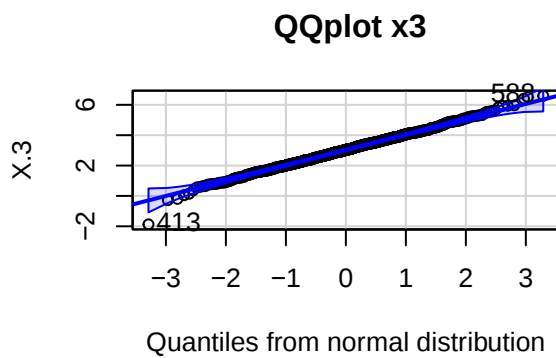
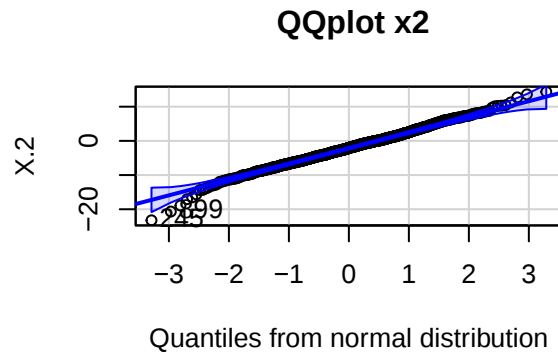
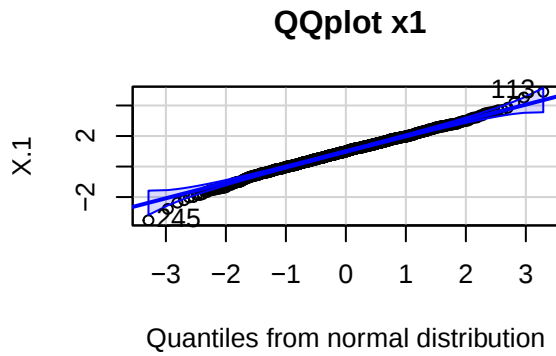
```
qqPlot(mvar_data$X.2, xlab = "Quantiles from normal distribution",  
       ylab = "X.2",  
       main = "QQplot x2")
```

```
## [1] 245 899
```

```
qqPlot(mvar_data$X.3, xlab = "Quantiles from normal distribution",  
       ylab = "X.3",  
       main = "QQplot x3")
```

```
## [1] 413 588
```

```
qqPlot(mvar_data$univ_out, xlab = "Quantiles from normal distribution",  
       ylab = "univ_out",  
       main = "QQplot univ_out")
```



```
## [1] 150 214
```

d

```

sx <- cov(mvar_data[, 1:3])
mvar_data$MD2 <- mahalanobis(mvar_data[, 1:3], colMeans(mvar_data[, 1:3]), sx)

critical <- qchisq(p = 0.975, df = 3)
critical

```

```
## [1] 9.348404
```

```

mvar_data$MD_out <- ifelse(mvar_data$MD2 > critical, 1, 0)
sum(mvar_data$MD_out[mvar_data$MD_out == 1])

```

```
## [1] 34
```

e

```

library("MASS")
robust_cov <- cov.mcd(mvar_data[,1:3])$cov
robust_mean <- apply(mvar_data[, 1:3], 2, median)
mvar_data$MD2rob <- mahalanobis(mvar_data[, 1:3], robust_mean, robust_cov)

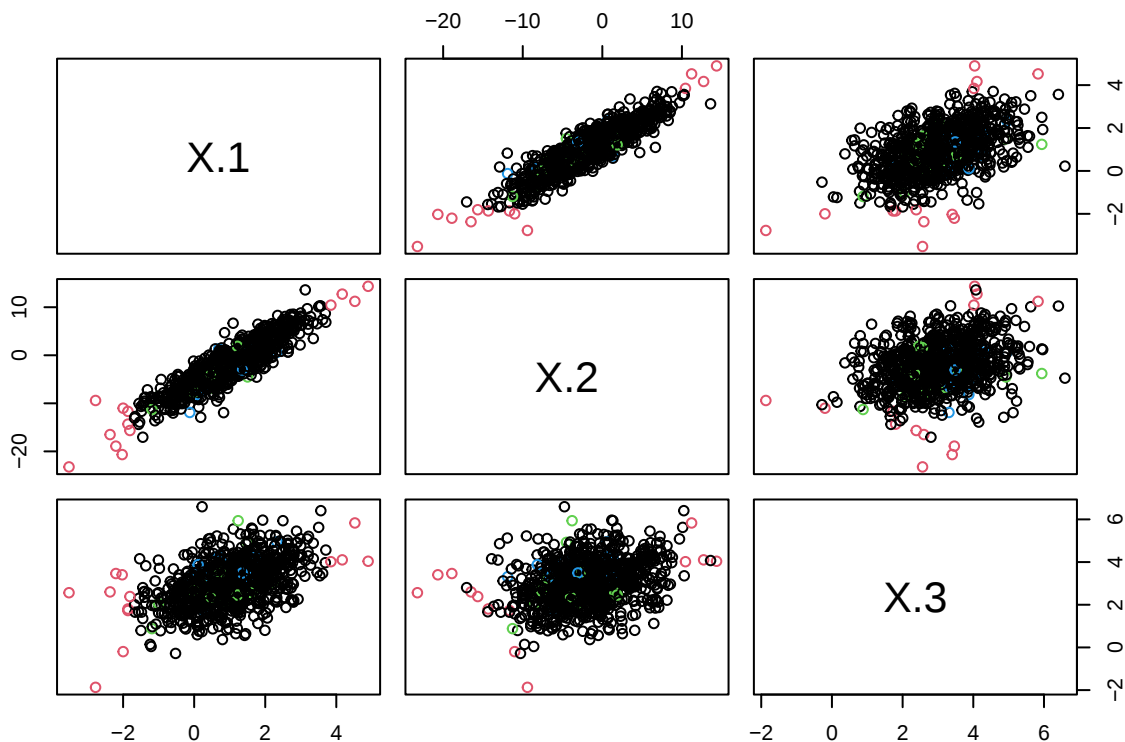
critical_values <- qchisq(p = 0.975, df = 3)
mvar_data$MDrob_out <- ifelse(mvar_data$MD2rob > critical_values, 1, 0)
sum(mvar_data$MDrob_out[mvar_data$MDrob_out == 1])

```

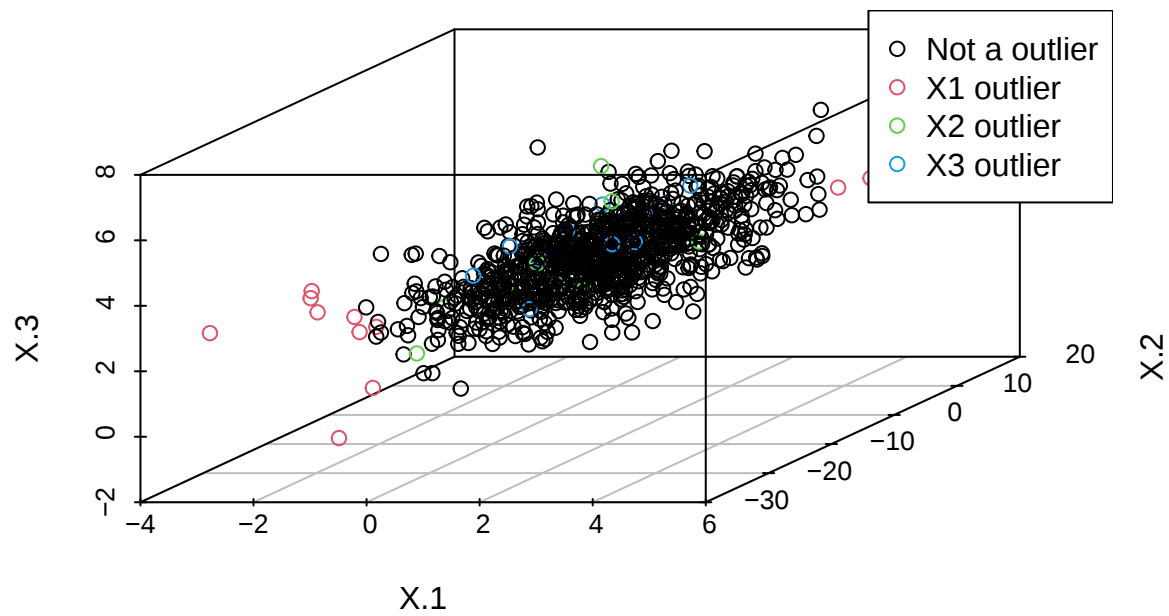
```
## [1] 51
```

f

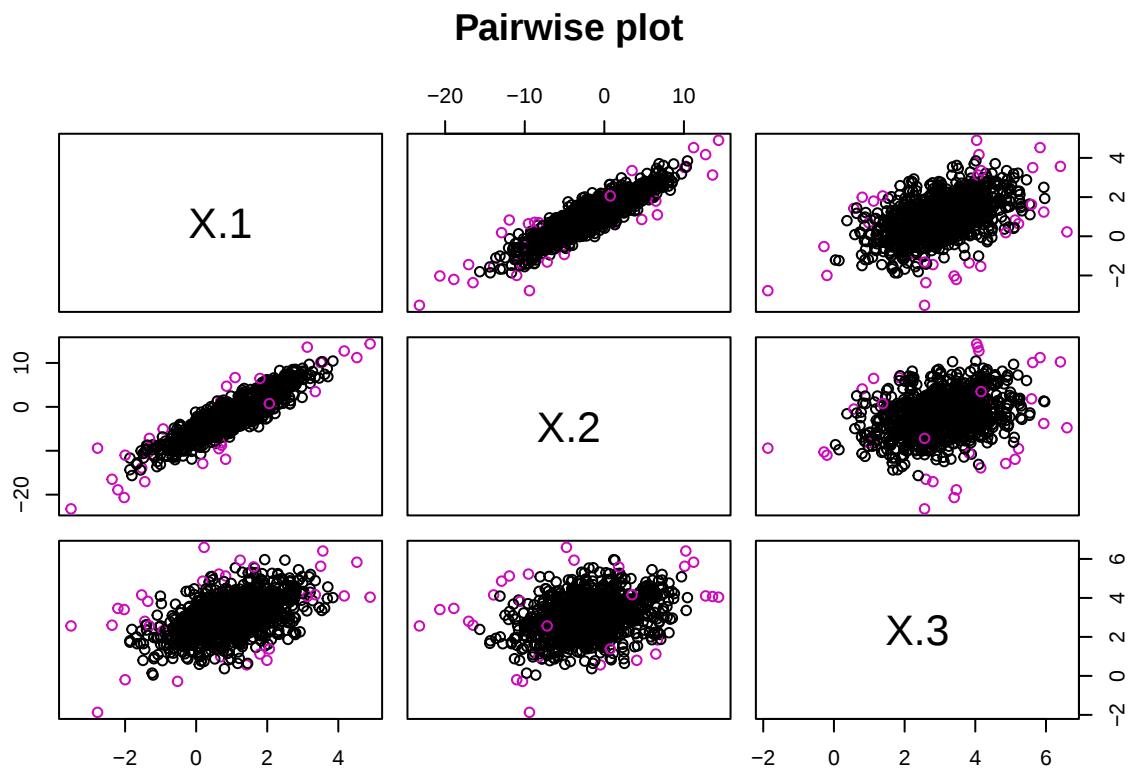
```
my_color <- 1
my_color[mvar_data$univ_out == 0] <- 1
my_color[mvar_data$univ_out == 1] <- 2
my_color[mvar_data$univ_out == 2] <- 3
my_color[mvar_data$univ_out == 3] <- 4
pairs(mvar_data[1:3], col = my_color)
```



```
scatterplot3d(mvar_data[1:3], color = my_color)
legend("topright",
      col = c(1, 2, 3, 4),
      legend = c("Not a outlier", "X1 outlier",
                  "X2 outlier", "X3 outlier"),
      pch = 1,
      bg = "white")
```

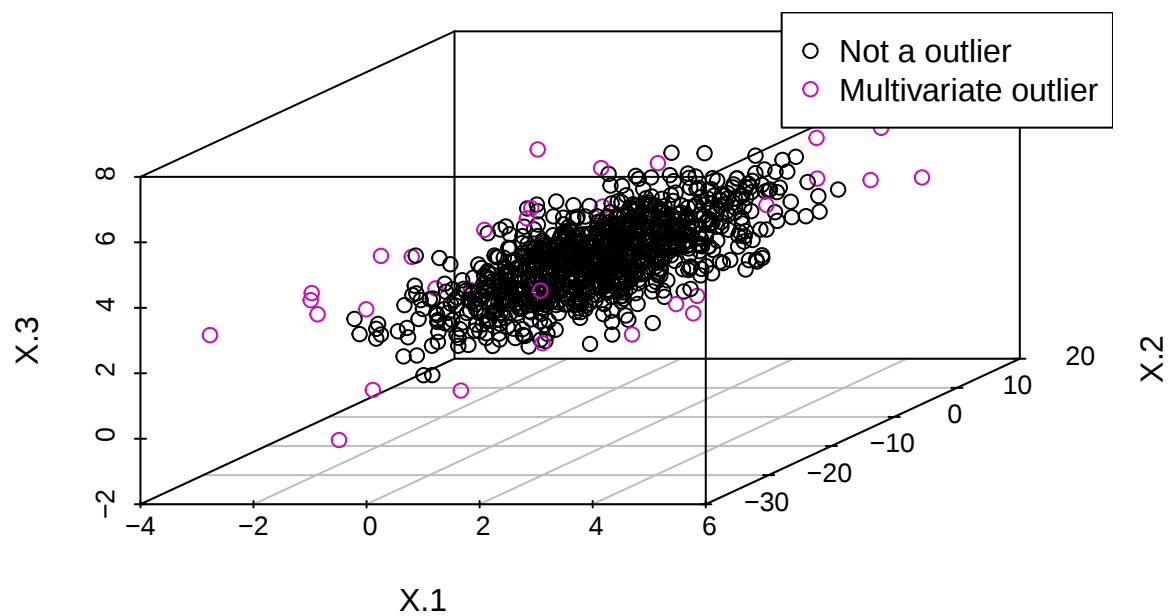


```
my_color <- ifelse(mvar_data$MD_out == 1, 6, 1)
pairs(mvar_data[1:3], col = my_color,
      main = "Pairwise plot")
```



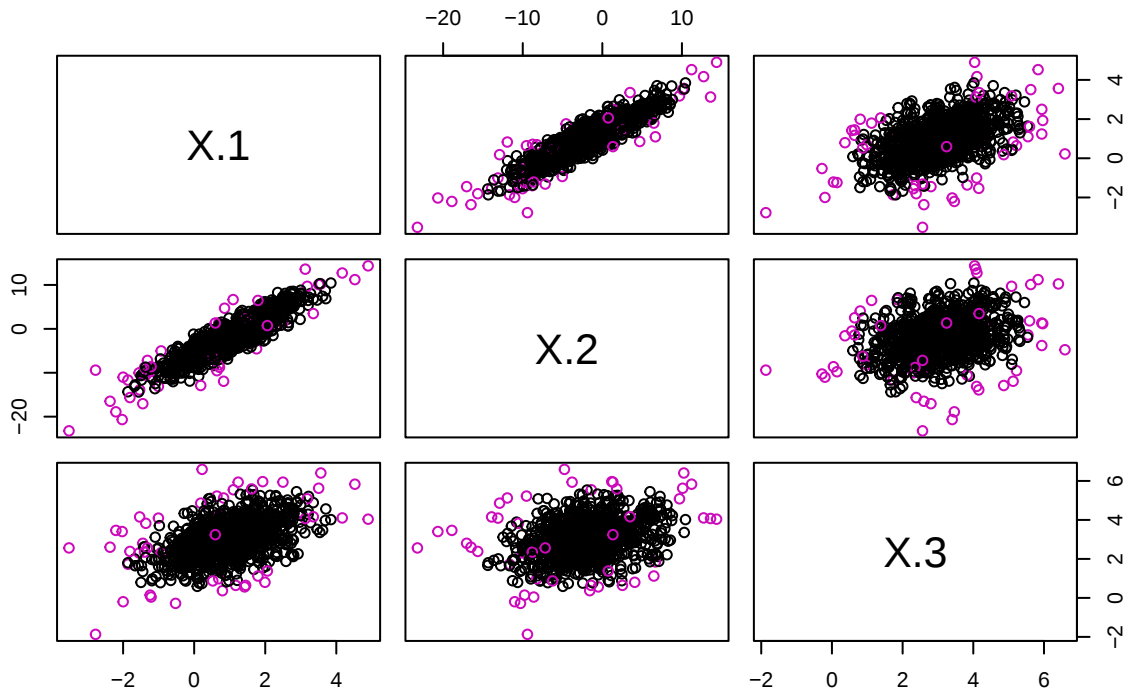
```
scatterplot3d(mvar_data[1:3], color = my_color,
              main="3d Scatter plot")
legend("topright", col = c(1, 6),
      legend = c("Not a outlier", "Multivariate outlier"),
      pch = 1, bg = "white")
```

3d Scatter plot



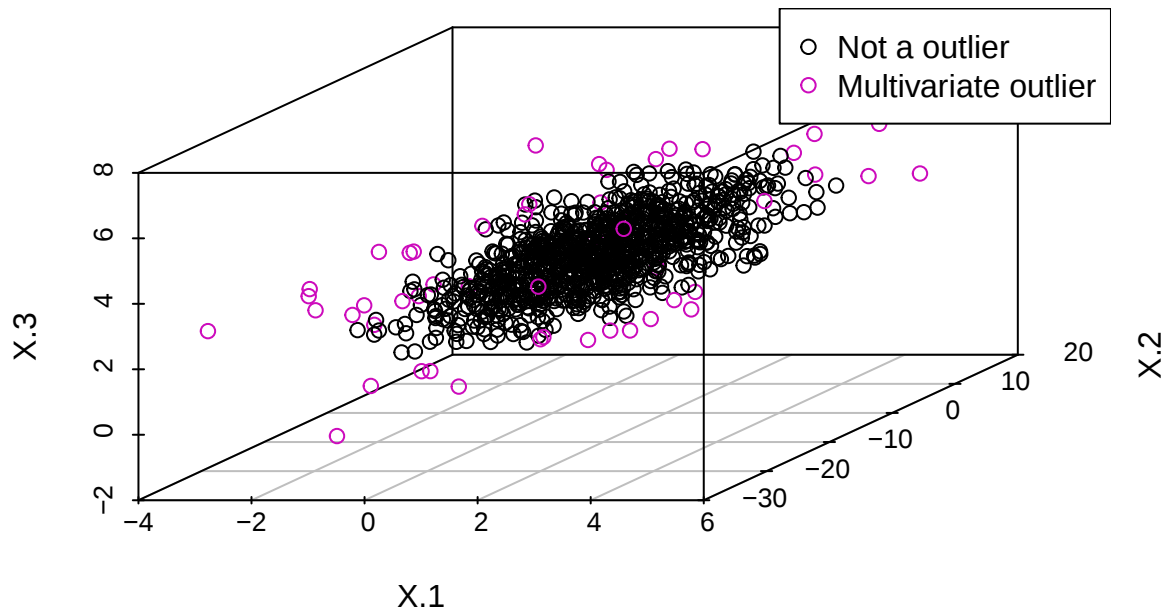
```
my_color <- ifelse(mvar_data$MDrob_out == 1, 6, 1)
pairs(mvar_data[1:3], col = my_color,
      main = "Pairwise plot for robust outliers")
```

Pairwise plot for robust outliers



```
scatterplot3d(mvar_data[1:3], color = my_color,
              main = "3d Scatter plot for robust outliers")
legend("topright", col = c(1, 6),
      legend = c("Not a outlier", "Multivariate outlier"),
      bg = "white", pch = 1)
```

3d Scatter plot for robust outliers



```
length(mvar_data$univ_out[mvar_data$univ_out != 0])
```

```
## [1] 37
```

```
sum(mvar_data$MD_out[mvar_data$MD_out == 1])
```

```
## [1] 34
```

```
sum(mvar_data$MDrob_out[mvar_data$MDrob_out == 1])
```

```
## [1] 51
```

g

The Euclidean distance assumes that the data is isotopically Gaussian, i.e. each variable is considered the same. Mahalanobis distance measures correlation between variables and relaxes the Euclidean distance assumption by assuming an anisotropic Gaussian distribution.

If you use the Mahalanobis distance, you don't have to standardize the data beforehand, as with Euclidian distance. This already happens by calculating the covariance matrix. Knowing in advance that there is some sort of correlation between variables, I would prefer Mahalanobis distance.