

# Übung 3

Aufgaben 14 bis 18

24.10.2022

## Aufgabe 14:

a.

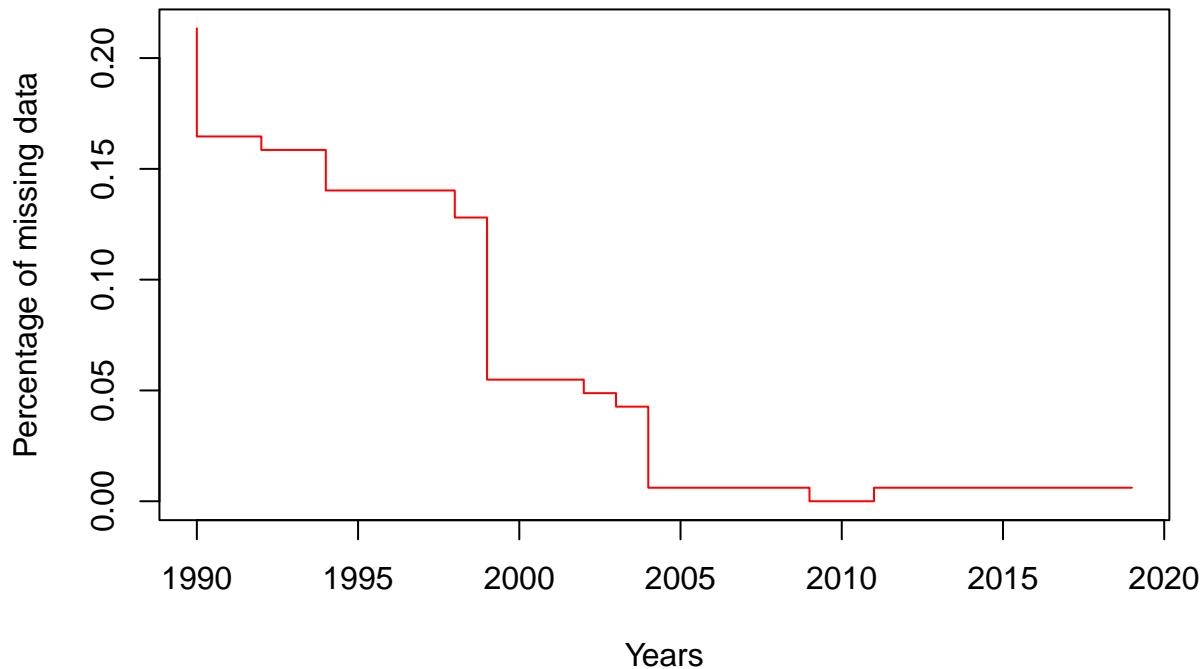
```
sdi <- read.csv("sdi.csv")
#sdi
```

b.

```
missing_data <- sapply(X = sdi[, -1], FUN = function (x) sum(is.na(x)))
percentages <- missing_data / length(sdi$country)
#percentages

plot(1990:2019, percentages, type = "S",
      xlab ="Years", ylab ="Percentage of missing data", col ="red",
      main = "Percentages of missing data through years")
```

## Percentages of missing data through years



c.

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

```
## [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.

```

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

```

f.

```

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

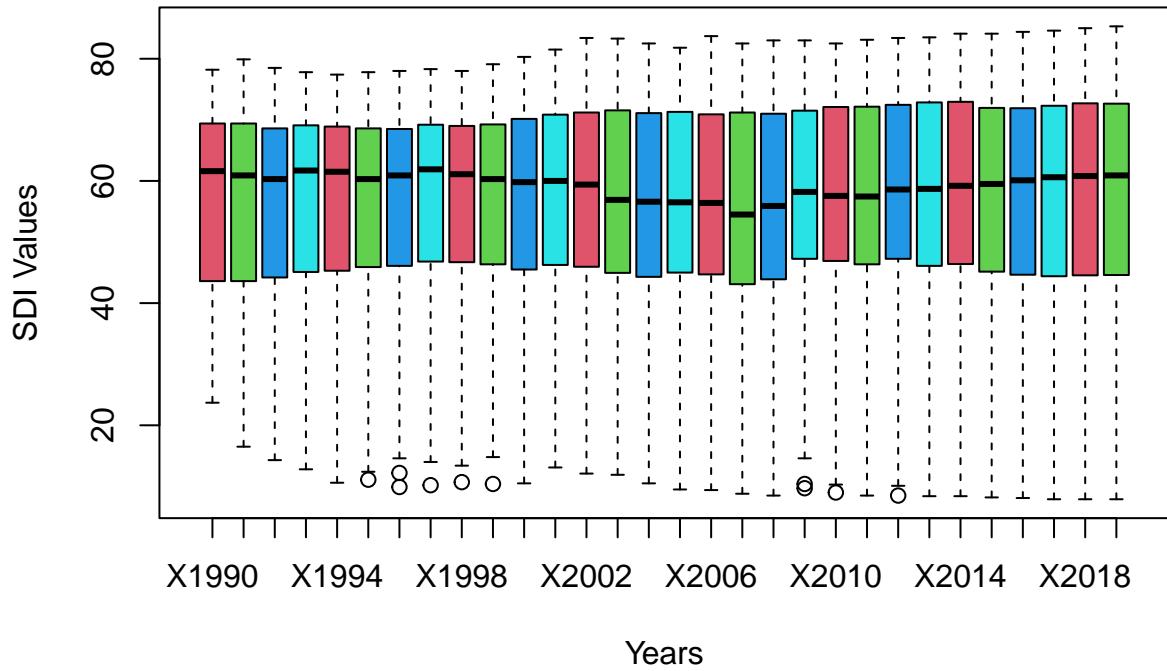
```

g.

```

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

```

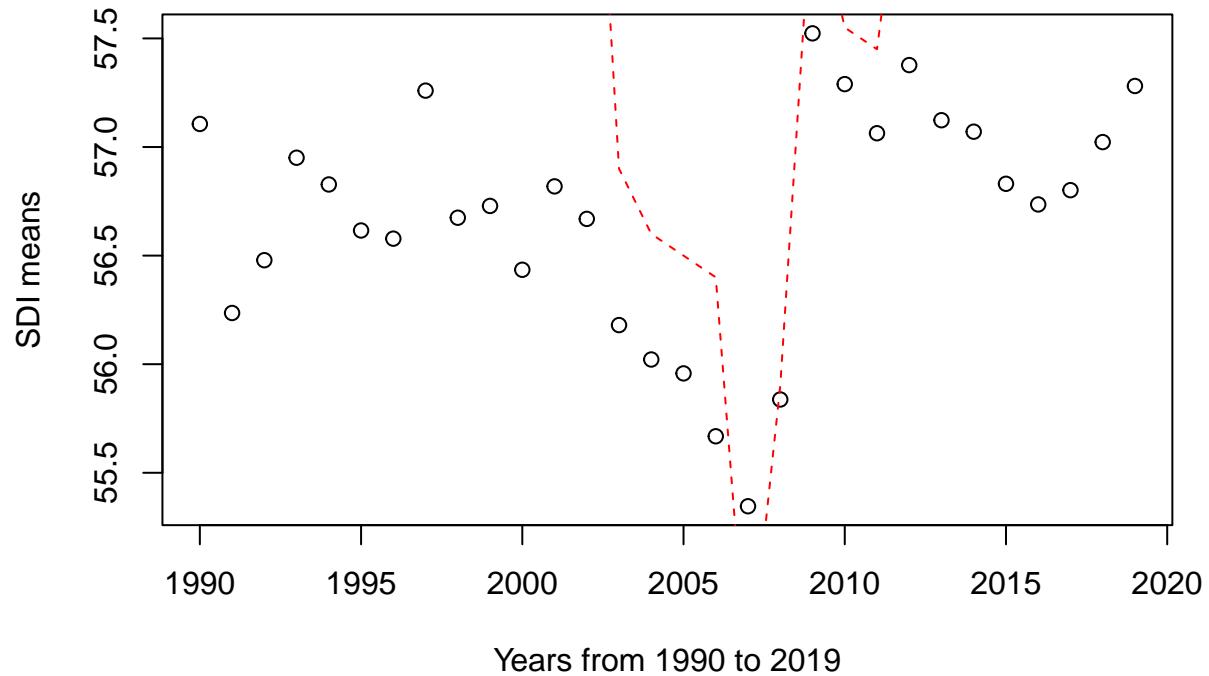


Anhand des obigen Boxplots kann man auf Verteilung von SDI-Werte für das jeweilige Jahr schließen. Das arithmetische Mittel zeigt sich durch die schwarze Linie des jeweiligen Boxplots. Ist diese Linie ungefähr in der Mitte, sind die SDI-Werte für dieses Jahr ungefähr normalverteilt. Eine Linie die nach oben gerutscht ist, weist auf eine linksschiefe Verteilung hin (d.h. die Schiefe g1 ist  $< 0$ ). Eine schwarze Linie, die sich im unteren Teil des Boxplots befindet, deutet auf rechtsschiefe Verteilung (die Schiefe g1 ist  $> 0$ ) hin. Kleine Punkte außerhalb der Wiskers sind die Ausreißer. D.h. diese Datenpunkte wirken sich auf das aritmetische Mittel aus. Die sind entweder viel kleiner oder größer als die anderen Daten in der Stichprobe und passen nicht in den interquartilen Abstand  $\times 1.5$  (IQR\*1.5 - maximale Länge von Wiskers).

h.

```
sdi_means <- with(sdi_long,
                    tapply(sdi_long$sdi, sdi_long$year, mean))
sdi_medians <- with(sdi_long,
                     tapply(sdi_long$sdi, sdi_long$year, median))
plot(1990:2019, sdi_means,
     xlab = "Years from 1990 to 2019",
     ylab = "SDI means",
     main = "Plot of mean SDI through years")
lines(1990:2019, sdi_medians,
      lty = "dashed", col = "red")
```

### Plot of mean SDI through years



## Aufgabe 15:

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

a.

```
#left join, da ich hiermit ein Data Frame erstellen will, in dem
#alle Filme drinnen sind, die von dem jeweilen Nutzer eine Bewertung bekommen haben
user_ratings <- merge(movies1, ratings, by = c("movieId"), all.x = TRUE)

#sum(!complete.cases(user_ratings)) wollte nur sehen ob movies1
#keine bewertung bekommen haben
users <- unique(ratings$userId) #alle user, die es gibt
#length(users)
sum(!(users %in% user_ratings$userId))

## [1] 15
```

Da ich es aus der Angabe nicht klar lesen konnte, wie diese Aufgabe zu machen war, hier folgt meine Motivation: Erstens habe ich ein Data Frame erstellt, indem ich auf *movie1* und *ratings* INNER JOIN angewendet habe. Das heißt, in meinem Data Frame erscheinen die Zeilen, wo keine Na Werte erlaubt sind (anders ausgedrückt, nur jene Zeilen, wo ein User einen Film aus *movies1* bewertet hat). Aus dem ursprünglichen Data Frame kann ich mit der *unique* Funktion rauslesen, wie viele **verschiedene** Nutzer\_innen es insgesamt gibt. Dann kann ich diesen neuen Vektor, bestehend aus *userID*, verwenden, um zu überprüfen, ob sich alle diese Nutzer\_innen in Data Frame *user\_ratings* befinden. Falls sich diese Nutzer\_innen im Data Frame *user\_ratings* nicht befinden, heißt das, dass sie keinen Film aus *movies1* bewertet haben, da in dem Data Frame *user\_rating* nur jene Nutzer\_inne erscheinen, die mindestens eine Bewertung abgelegt haben. So kann ich dann alle Nutzer\_innen zählen, die keinen Film aus *movies1* bewertet haben.

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

dim(user_ratings_all)
```

```
## [1] 100836      6
```

Falls wir doch alle Nutzer\_innen in unserem Data Set haben wollen, obwohl sie auch Filme bewertet haben, die nicht im Data Frame *movies1* sind, brauchen wir RIGHT JOIN. D.h. im neuen Data Frame werden alle *movieIDs* erscheinen, obwohl dazu passende Informationen aus *movies1* nicht gegeben sind. Wo diese Informationen fehlen, wird es mit NA-Werten ausgefüllt.

```
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, "movie1_ratings.txt",
#             sep = ";", na = "999999", col.names = FALSE)
```

## Aufgabe 16:

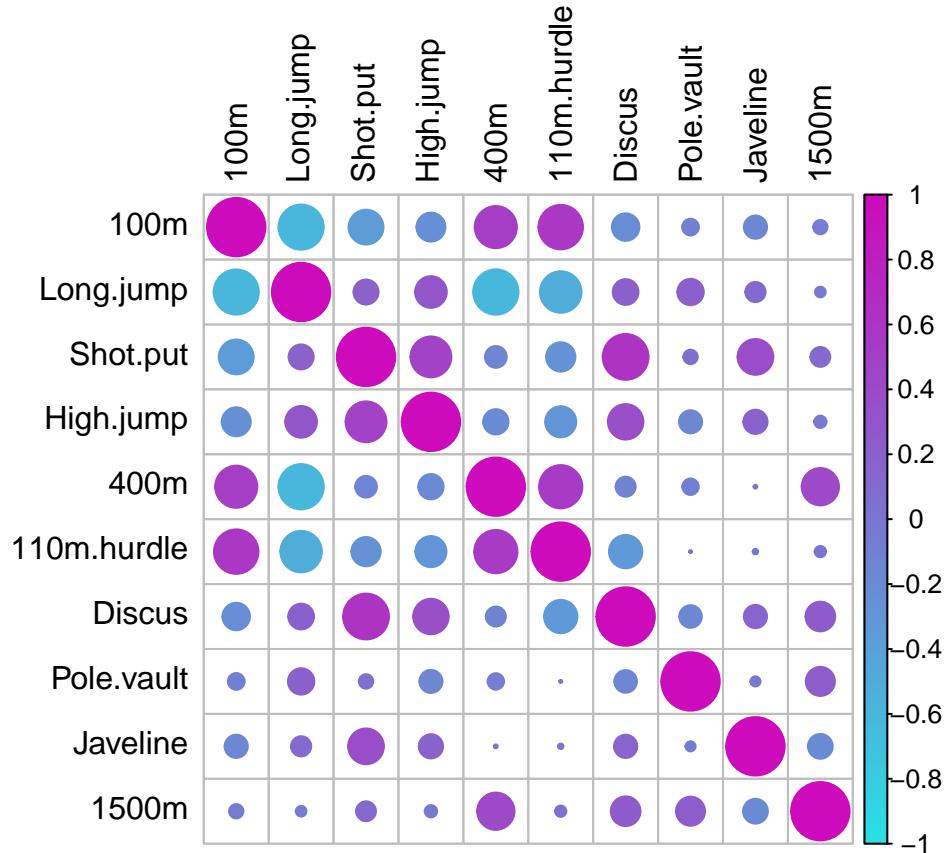
```
#instal.packages("FactoMineR")
library(FactoMineR)
data("decathlon")
```

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])
```

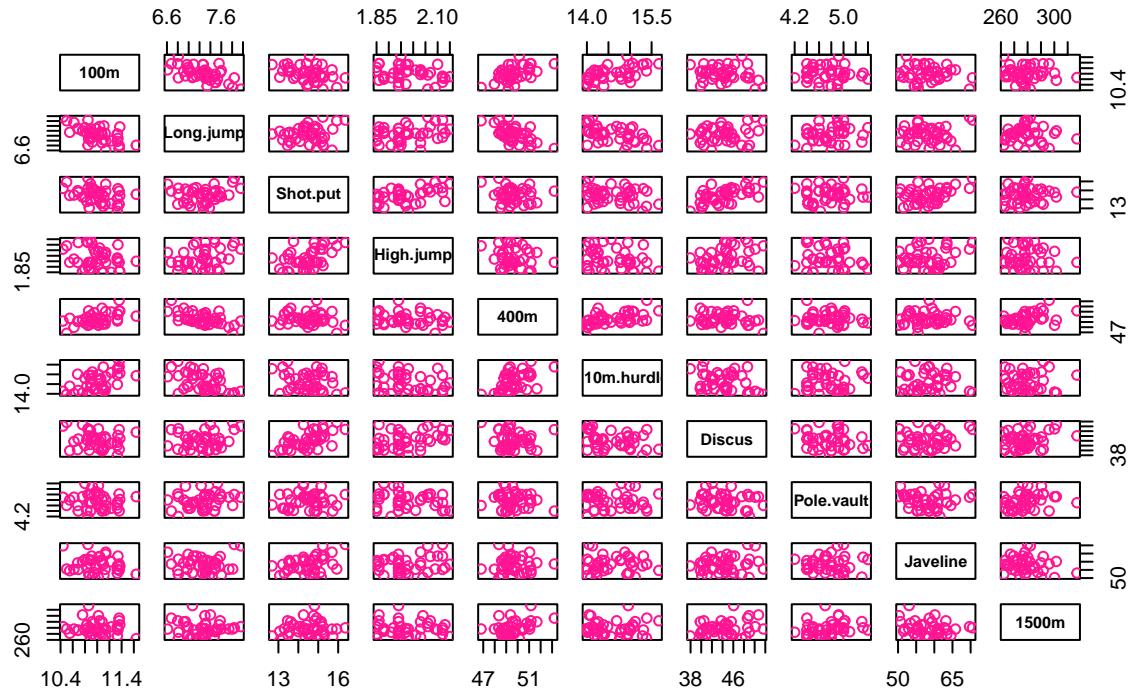
Interpretation:

- 1 weist auf “perfekte positive” Korrelation hin
- -1 weist auf “perfekte negative” Korrelation
- 0 -> es gibt keine Korrelation zwischen den zwei Variablen
- Werte zwischen -1 und 1 geben an, wie stark die Korrelation ist

Andere Möglichkeit:

```
pairs(decathlon[, 1:10],
      bg= "deeppink",
      col= "deeppink",
      rowlattop = TRUE,
      gap = 1,
      cex.labels = NULL,
      font.labels = 2,
      main = "Correlation between variables")
```

## Correlation between variables



b.

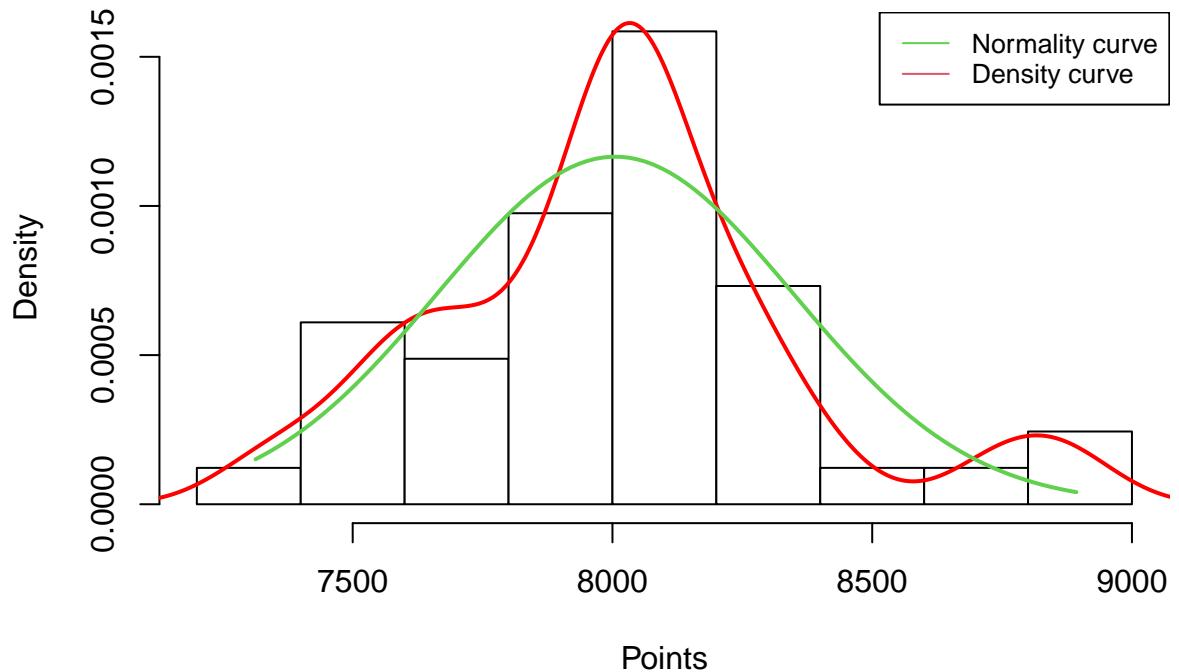
```
x2 <- seq(min(decathlon$Points), max(decathlon$Points), length = 100)
fun <- dnorm(x2, mean = mean(decathlon$Points), sd = sd(decathlon$Points))

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)

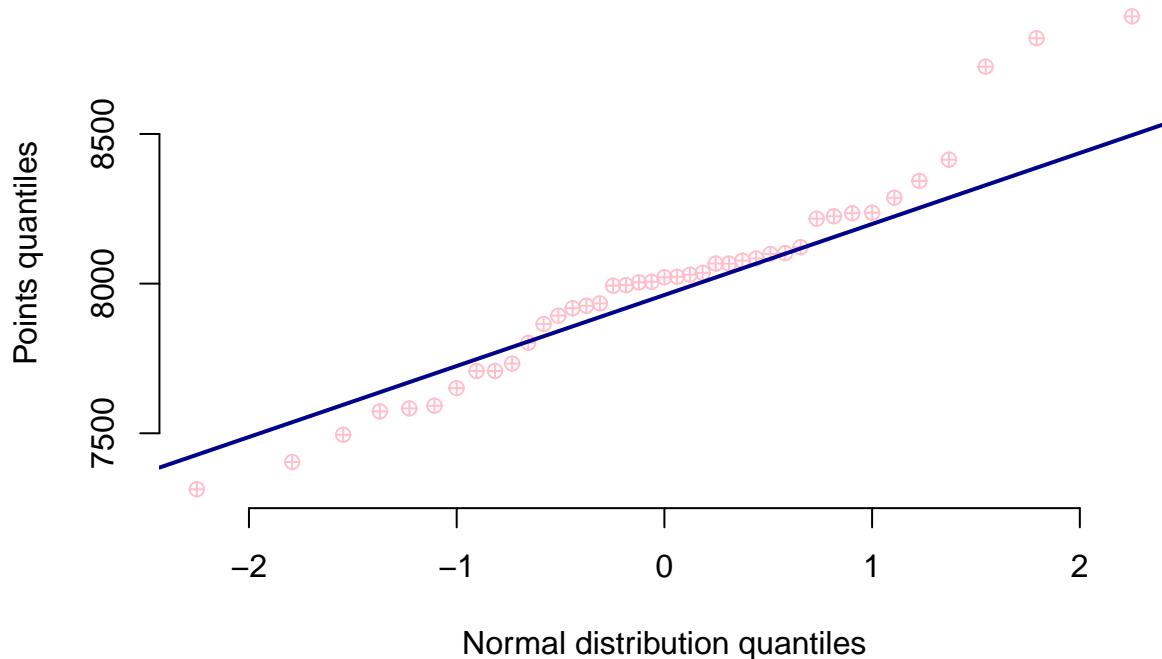
legend("topright", legend = c("Normality curve", "Density curve"),
       col = c(3,2), lty = 1, cex = 0.8)
```

## Points histogram



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

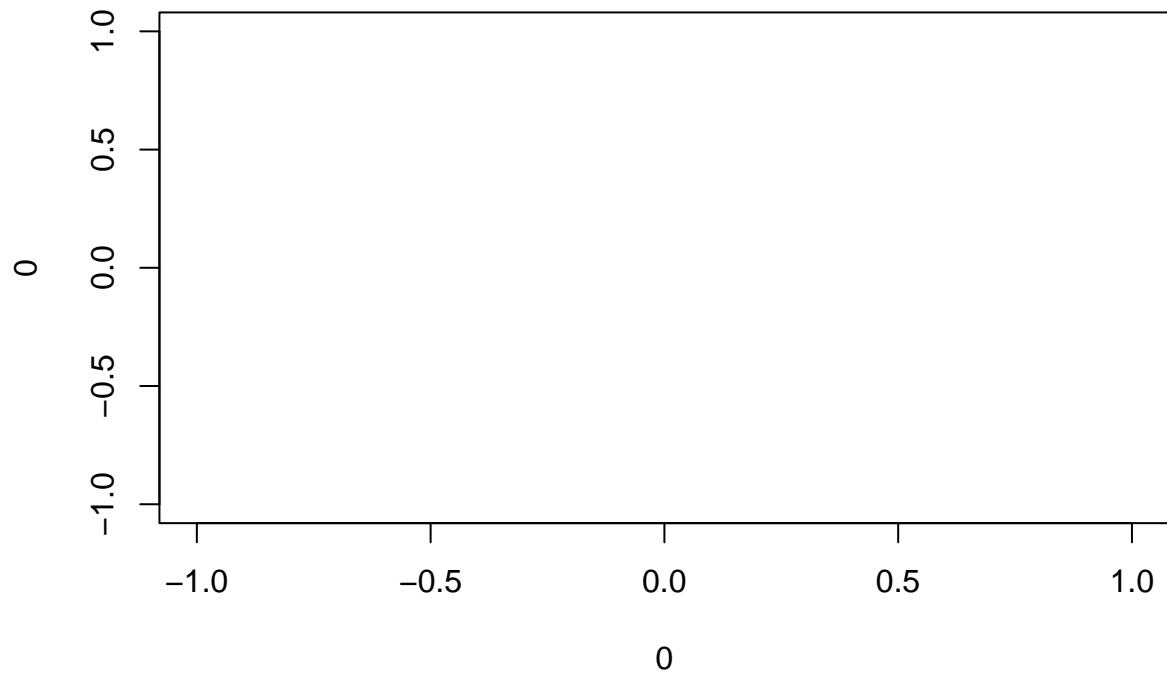
#andrews curves function

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 <- f_t(tseq, decathlon[, i])  
#  #lines(tseq, y)  
#}  
#}
```

## Aufgabe 17:

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

a.

Jede Zeile im Dataset gibt Informationen zu einer mit Ebola infizierten Person an. Wichtige Variablen sind jedenfalls ID der erkrankten Person (*case\_id*), Daten wann sie hospitalisiert wurde (*date\_onset*), wann sie entlassen wurde (*date\_outcome*) und wann sich die Person infiziert hat (*date\_infection*). Andere Variablen geben Auskunft darüber, ob die Person genesen wurde, oder ob sie an der Infektion verstorben ist. Noch sind alle Nebenwirkungen, die erkrankte Person hatte, aufgezählt in Form von univariaten Variablen (diese Variablen nehmen entweder **TRUE** an, falls diese Nebenwirkung gegeben war oder **FALSE**, wenn nicht). Im Datenset sind auch Missing Values enthalten. Wenn zur jeweiligen Person eine der Variablen nicht bekannt war, wurde sie nicht ausgefüllt. Wenn bei einer Beobachtung beispielsweise *outcome* Tot war, kann das Datum der Entlassung aus Spital nicht angegeben werden und wird mit NA ausgefüllt.

b.

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

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

#class(linelist_messy_dates$date_onset)

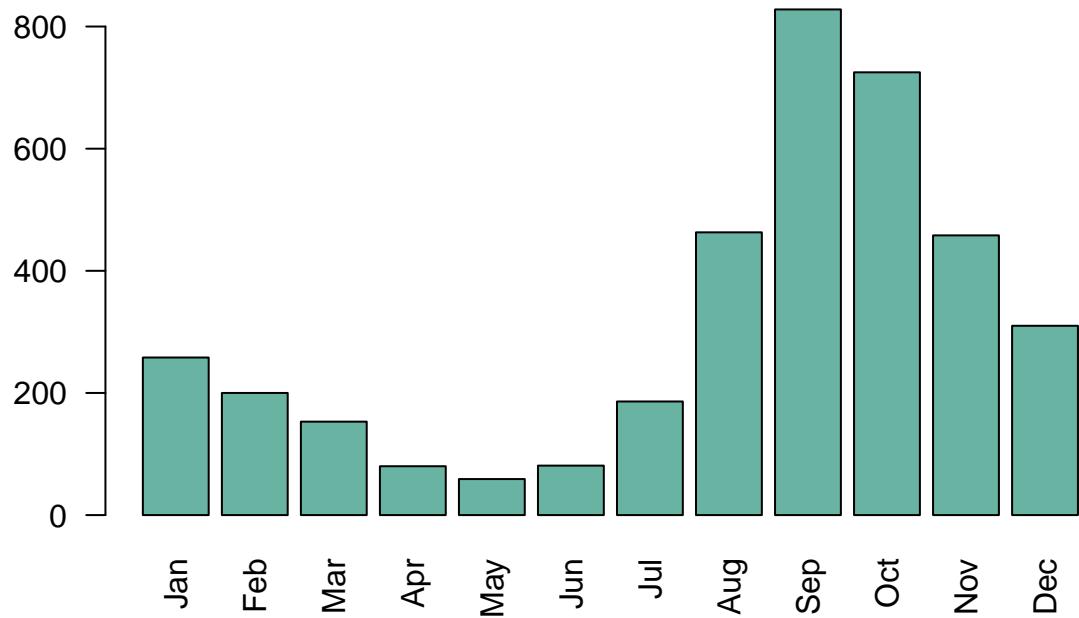
linelist_messy_dates$date_onset <-
  as.Date(linelist_messy_dates$date_onset,
          "%y/%B/%d")
class(linelist_messy_dates$date_outcome) #muss nicht geändert werden

## [1] "Date"
```

c.

```
months <- format(linelist_messy_dates$date_infection, format ="%b")
months <- factor(months, levels = month.abb)
months<- months[!is.na(months)]
months_to_plot <- table(months)
#months_to_plot
barplot(months_to_plot, las = 2,
       main = "Number of infections in each month",
       col ="#69b3a2")
```

## Number of infections in each month



Ich habe einen Faktor mit Monaten im Jahr erzeugt, damit ich dann einfach mittels `table()` Funktion ablesen kann, wie oft eine Erkrankung in jedem Monat festgestellt wurde. Diese Tabelle kann dann ganz einfach mit Hilfe von Funktion `barplot()` gezeichnet werden. Man kann sehen, dass September und Oktober, zwei Monate mit der größten Anzahl an Infektionen sind. Mai ist Monat mit geringster Anzahl an registrierten Infektionen.

d.

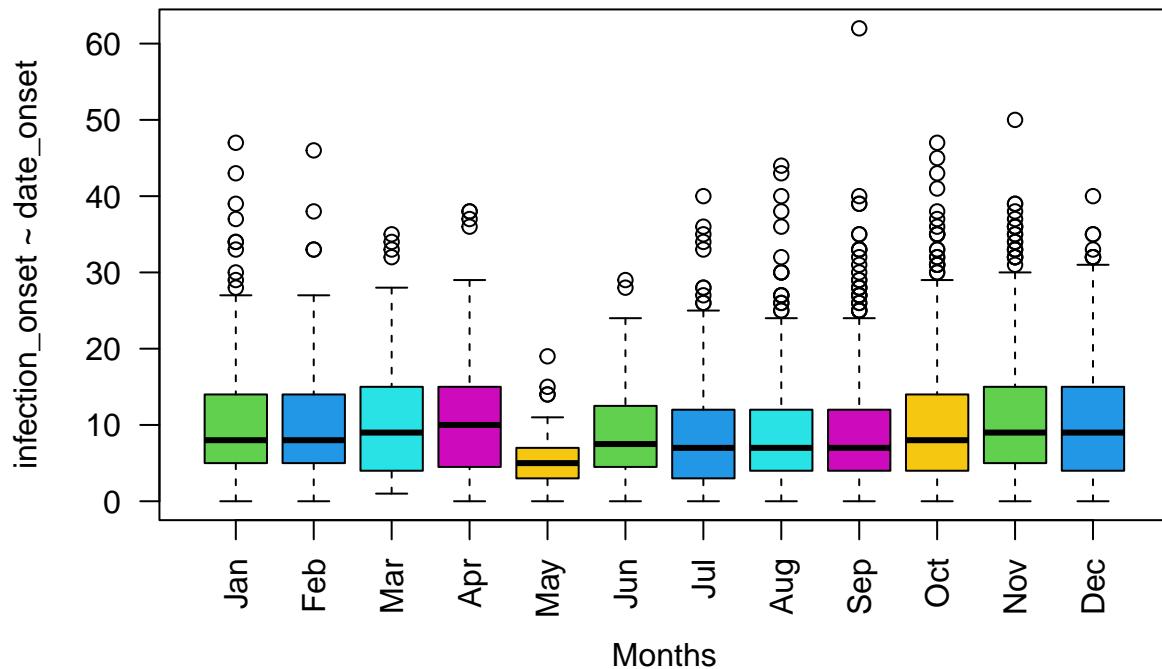
```
linelist_messy_dates$date_infection_onset <-
  linelist_messy_dates$date_onset - linelist_messy_dates$date_infection

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 = "infection_onset ~ date_onset", col = 3:7, las = 2)
```

## Infection – Onset Boxplots for each month



In der Grafik sind *date\_infection\_onset* Boxplots für jeden Monat zu sehen. Gruppierungsvariable ist in diesem Fall jeweils der Monat in dem *date\_onset* statt gefunden hat. Es werden so zu sagen Inkubationszeiten nach Monaten zusammengesetzt und es wird deren Boxplot angezeigt. Man kann beispielsweise aus Grafik für August ablesen, dass durchschnittliche Inkubationszeit weniger als 10 Tage gedauert hat. Es gab in jedem Monat aber Inkubationszeiten die über "erwarteten" Werte hinausgehen. Diese sind als Ausreißer zu sehen und die befinden sich außerhalb Wiskers (das wurde bereits in der Aufgabe 14 erklärt).

e.

```
#table(linelist_messy_dates$date_hospitalisation)
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$week_days <- factor(
  ifelse(wday(daily_hospitalizations$Group.1) <= 5,
```

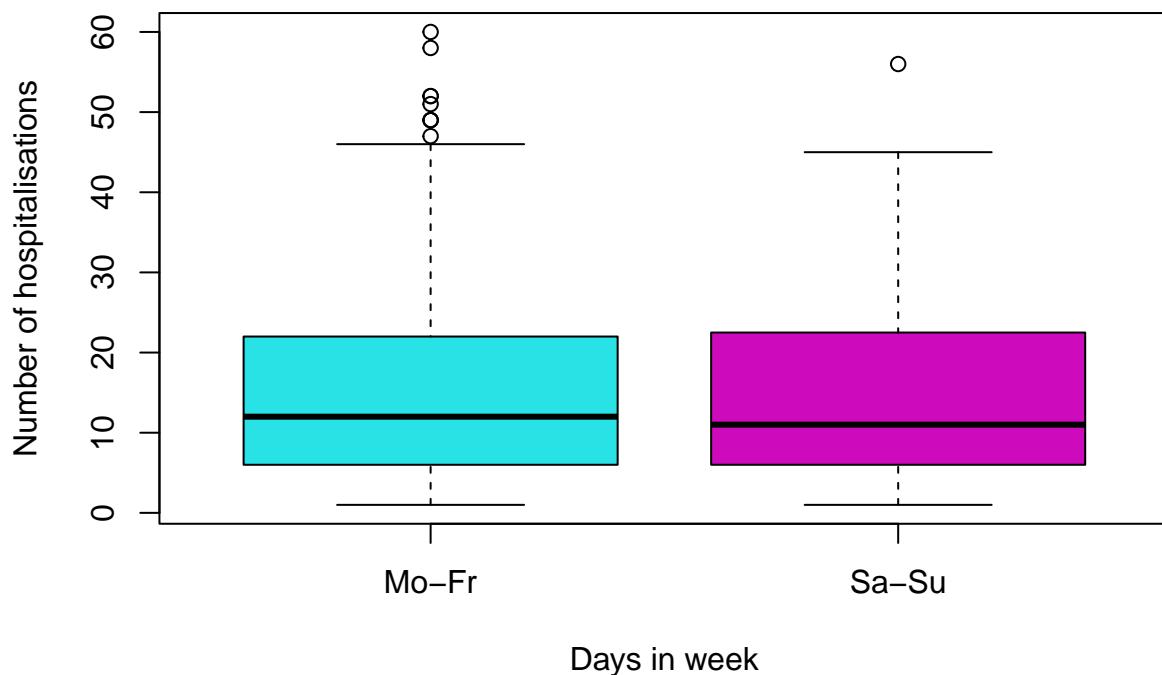
```

    "Mo-Fr", "Sa-Su"))
#week_days

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(5,6))

```

### Boxplot for hospitalisations on weekdays vs weekend



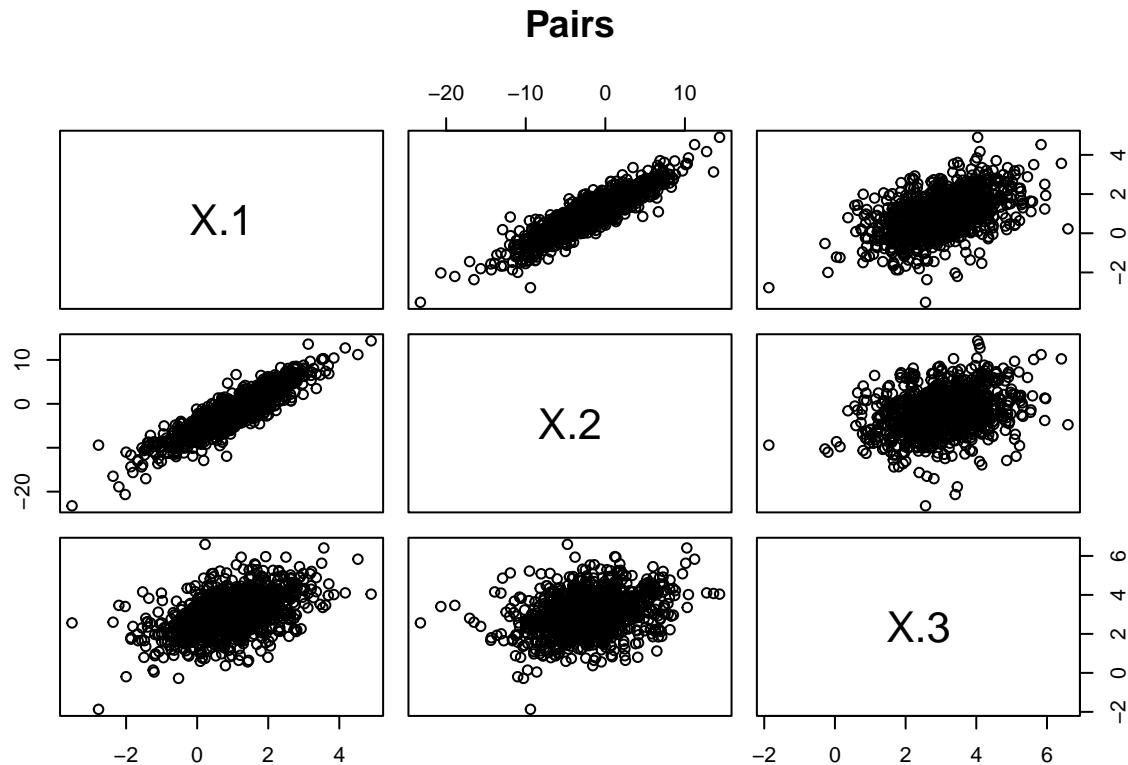
Wenn man zwei Boxplots so nebeneinander aufgestellt hat, kann man aus beiden ganz einfach das arithmetische Mittel der beiden ablesen. Im Durchschnitt infizieren sich mit Ebola an einem Arbeitstag etwas mehr als 10 Personen. Im Vergleich dazu liegt die Anzahl der neu infizierten am Wochenende etwas unter 10. Es gibt ja auch Ausreißer, die sich an Verteilung der Daten auswirken. Bei den Arbeitstagen sind diese zahlreicher, wobei am Wochenende in den gemessenen Daten nur ein Ausreißer außerhalb Wiskers liegt.

### Aufgabe 18:

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

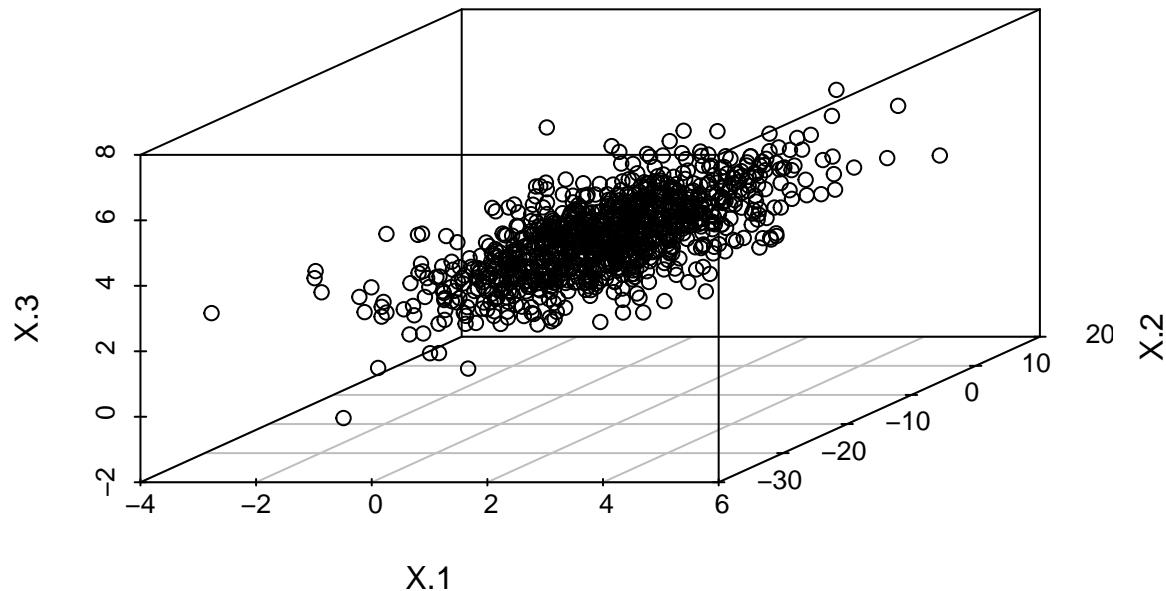
a.

```
library(scatterplot3d)
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))

## [1] 13

sum(detect_outlier(mvar_data$X.2))

## [1] 13

sum(detect_outlier(mvar_data$X.3))

## [1] 11

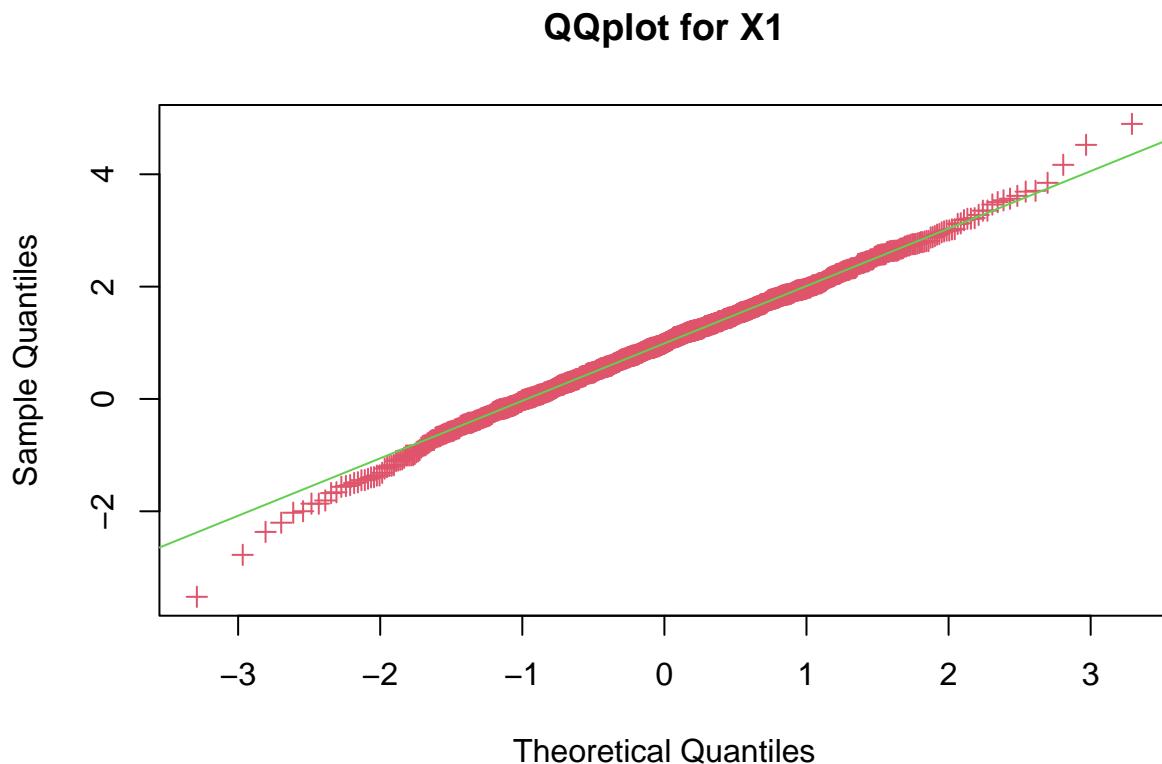
```

c.

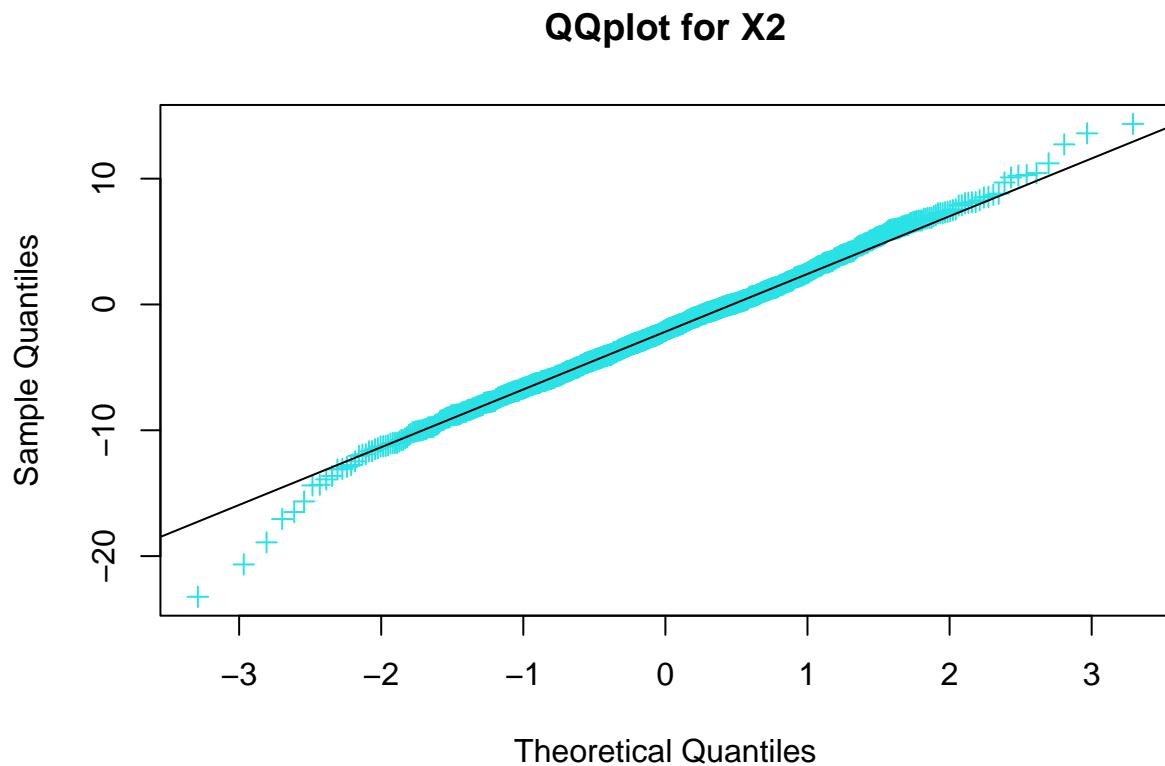
```

qqnorm(mvar_data$X.1, pch = 3, col = 2,
       main = "QQplot for X1")
qqline(mvar_data$X.1, pch = 3, col = 3)

```

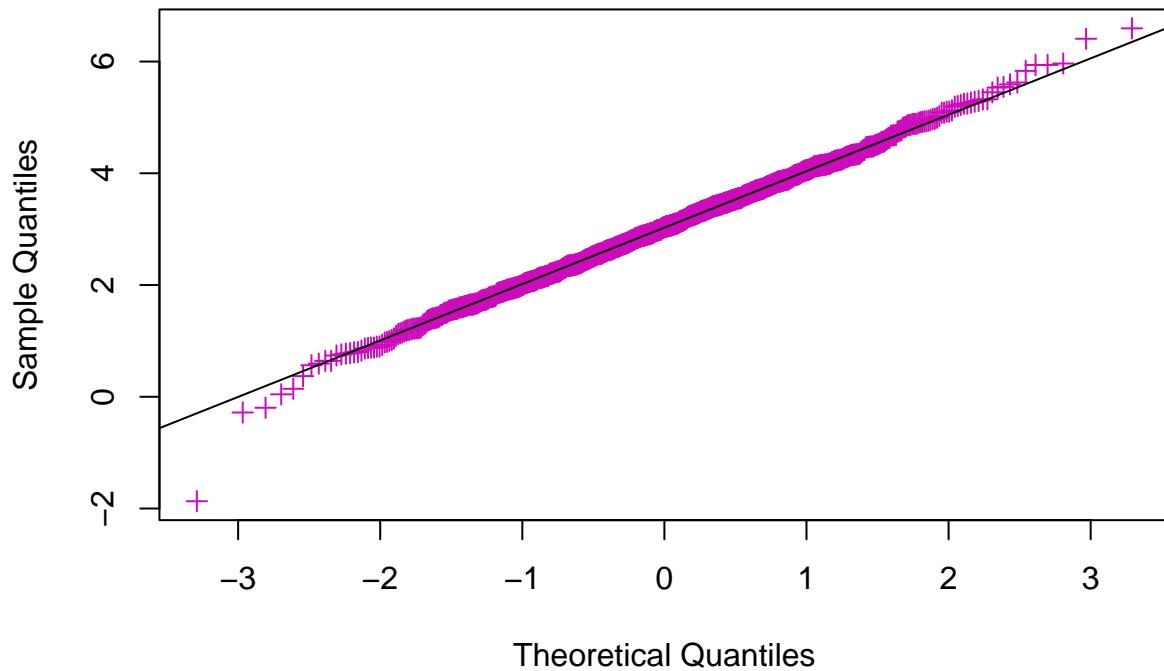


```
qqnorm(mvar_data$X.2, pch = 3, col = 5,
       main = "QQplot for X2")
qqline(mvar_data$X.2, pch = 3, col = 1)
```



```
qqnorm(mvar_data$X.3, pch = 3, col = 6,
       main = "QQplot for X3")
qqline(mvar_data$X.3, pch = 3, col = 1)
```

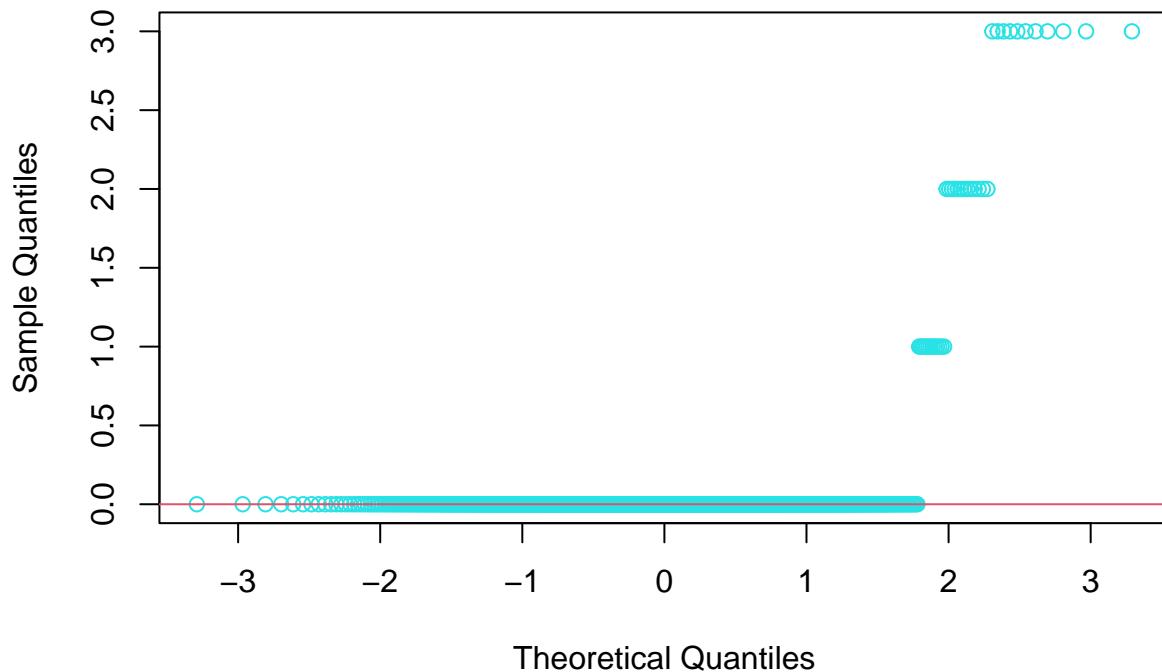
### QQplot for X3



Ich bin mir ziemlich sicher, dass QQplots nur für die ersten drei variablen anzuseigen waren, aber zur Sicherheit, hier auch QQplot für die *univar\_out* Variable:

```
qqnorm(mvar_data$univ_out, pch = 1, col = 5,
       main = "Our variable univ_out")
qqline(mvar_data$univ_out, pch = 3, col = 2)
```

## Our variable univ\_out



Eine Möglichkeit Normalverteilung der Daten zu testen ist mithilfe der Funktionen *qqnorm* und *qqline*. Mit *qqnorm* werden zuerst auf der y-Achse Quantile aus unserer Stichprobe markiert (in unserem Fall sind das jeweils die entsprechenden Spalten des Dataframes) und auf der x-Achse sind theoretische Quantile zu finden (aus einer theoretischen Normalverteilung). Dann kann mit *qqline* eine Linie gezeichnet werden. Sind ursprüngliche Daten normalverteilt, so liegen Quantile entlang dieser Linie. In unserem Fall trifft das zu. Es gibt jedoch Ausreißer, die wir in der vorigen Unteraufgabe entdeckt haben, aber die wirken sich nicht signifikant auf die Verteilung der Daten.

Es gibt auch die Möglichkeit Library *car* zu laden und die Funktion *qqPlot* zu verwenden. Diese Funktion markiert Bereiche, in den Quantile liegen würden, falls die Daten aus Stichprobe normalverteilt sind.

```
library(car)

## Loading required package: carData

par(mfrow = c(2,2))
qqPlot(mvar_data$X.1, xlab = "Quantile aus Normalverteilung",
       ylab = "X.1",
       main =" QQplot x1")

## [1] 245 113

qqPlot(mvar_data$X.2, xlab = "Quantile aus Normalverteilung",
       ylab = "X.2",
       main =" QQplot x2")
```

```

## [1] 245 899

qqPlot(mvar_data$X.3, xlab = "Quantile aus Normalverteilung",
       ylab = "X.3",
       main =" QQplot x3")

```

```

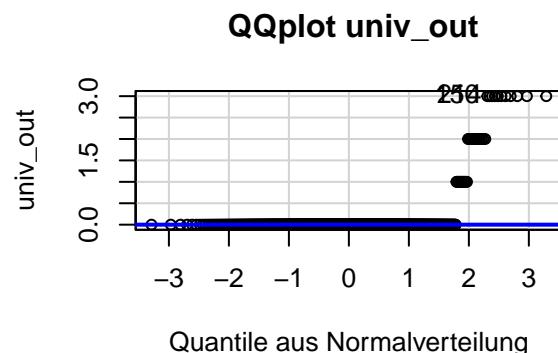
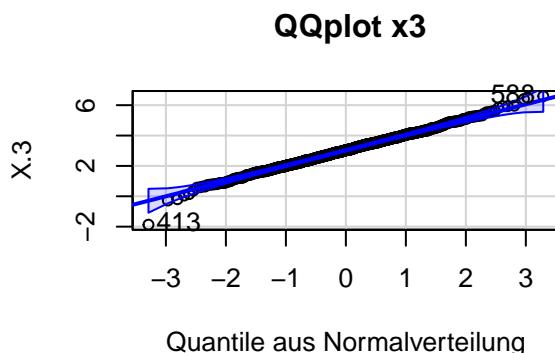
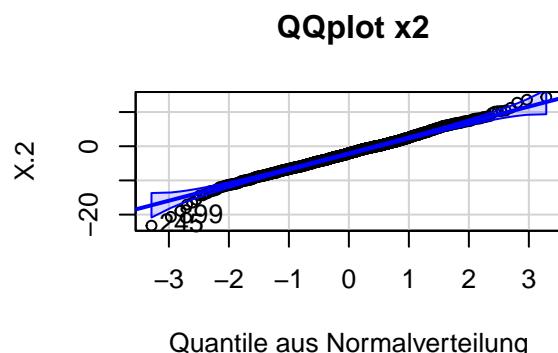
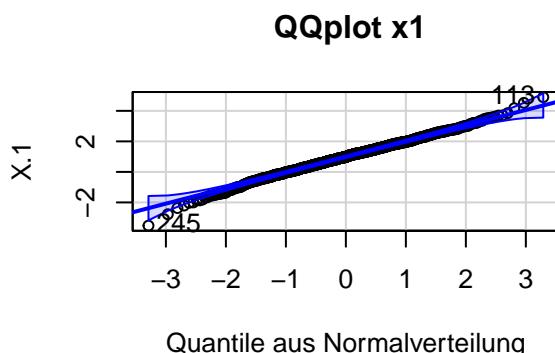
## [1] 413 588

```

```

qqPlot(mvar_data$univ_out, xlab = "Quantile aus Normalverteilung",
       ylab = "univ_out",
       main =" QQplot univ_out")

```



```

## [1] 150 214

```

d.

```

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

```

```

#zweiter Unterpunkt

```

```

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

Bei dem zweiten Unterpunkt kann man sich den kritischen Wert von MD2 mittels Funktion *qchisq()* ausrechnen. Unser Datensatz besteht aus 3 Spalten, die wir in Betracht miteinbeziehen wollen: X1, X2, X3. So ergeben sich insgesamt 3 Freiheitsgrade (Anzahl der Spalten die einbezogen werden sollen). Falls MD2 größer als der kritische Wert ist, ist MD2 dann ein Ausreißer.

e.

```

#install.packages("MASS")
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

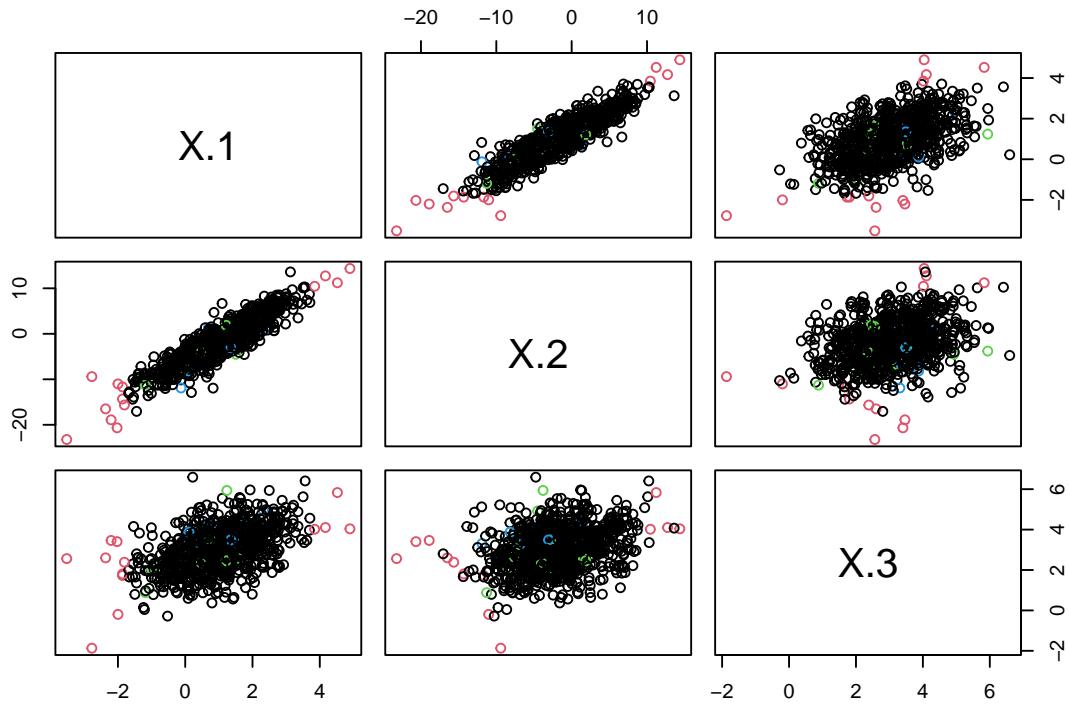
f.

- one pairwise and one 3D scatterplot for *univ\_out*

```

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)

```

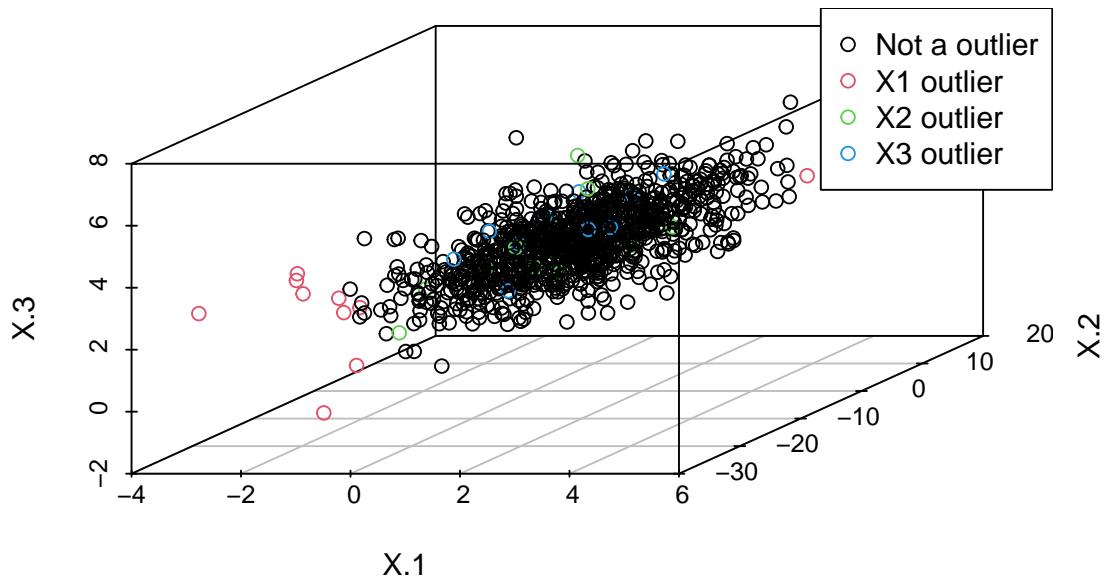


```

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")

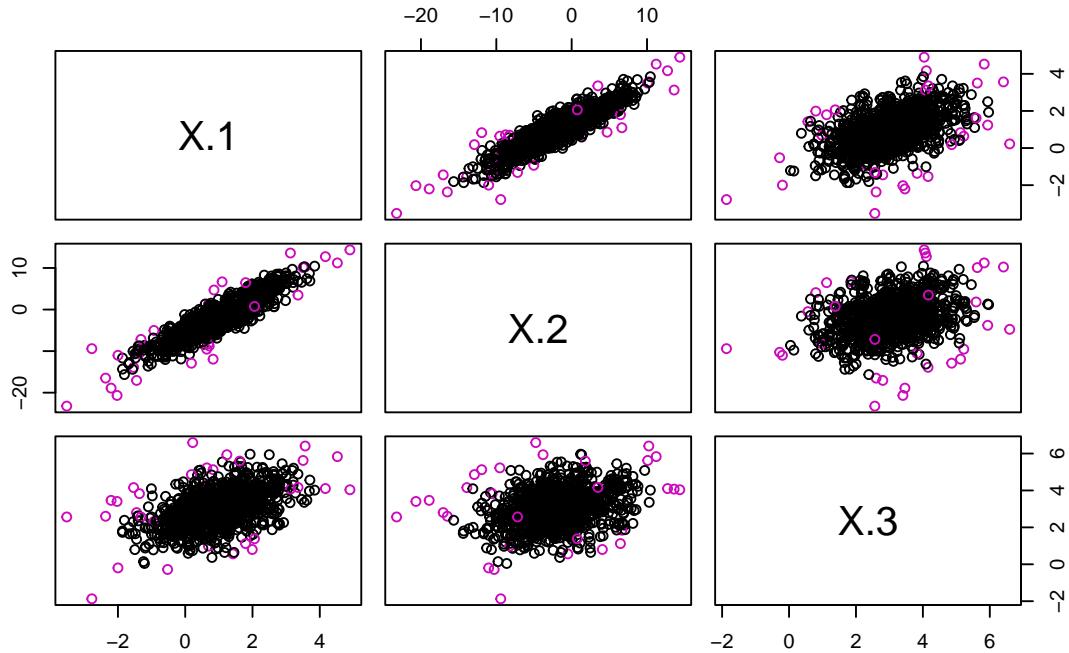
```



- one pairwise and one 3D scatterplot where you color the multivariate outliers based on MD

```
#outliers will be drawn with color 6
my_color <- ifelse(mvar_data$MD_out == 1, 6, 1)
pairs(mvar_data[1:3], col = my_color,
      main = "Pairwise plot")
```

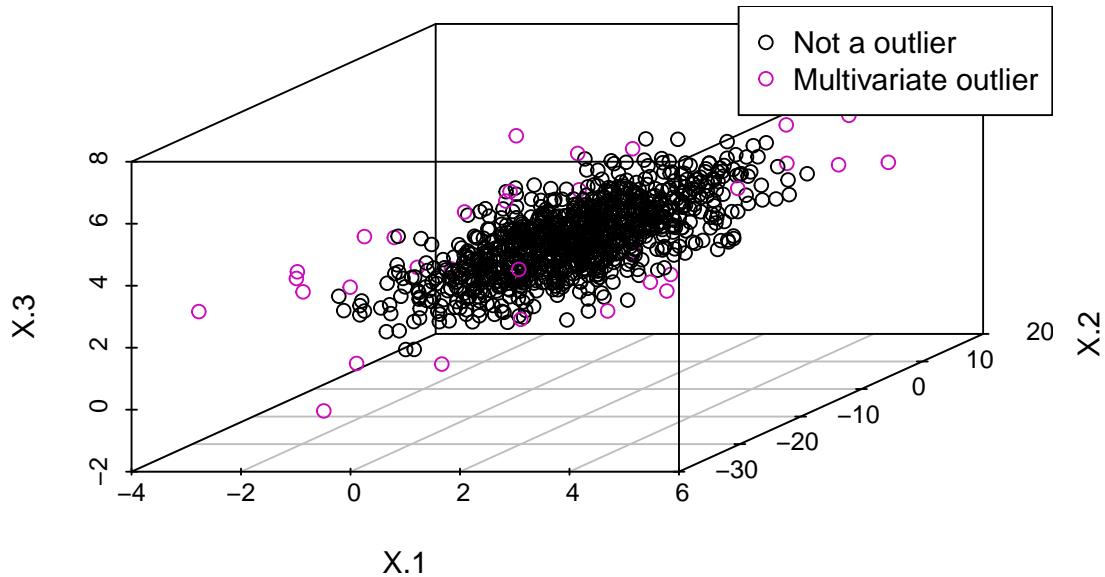
## Pairwise plot



```
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")
```

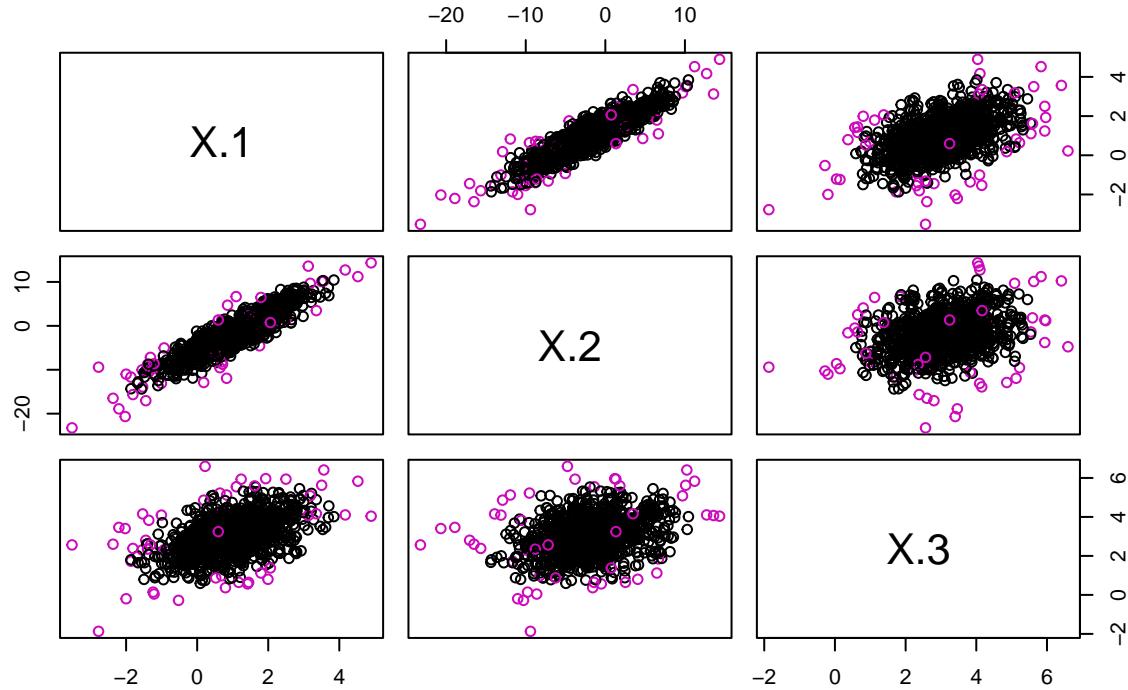
### 3d Scatter plot



- one pairwise and one 3D scatterplot where you color the multivariate outliers based on robust MD

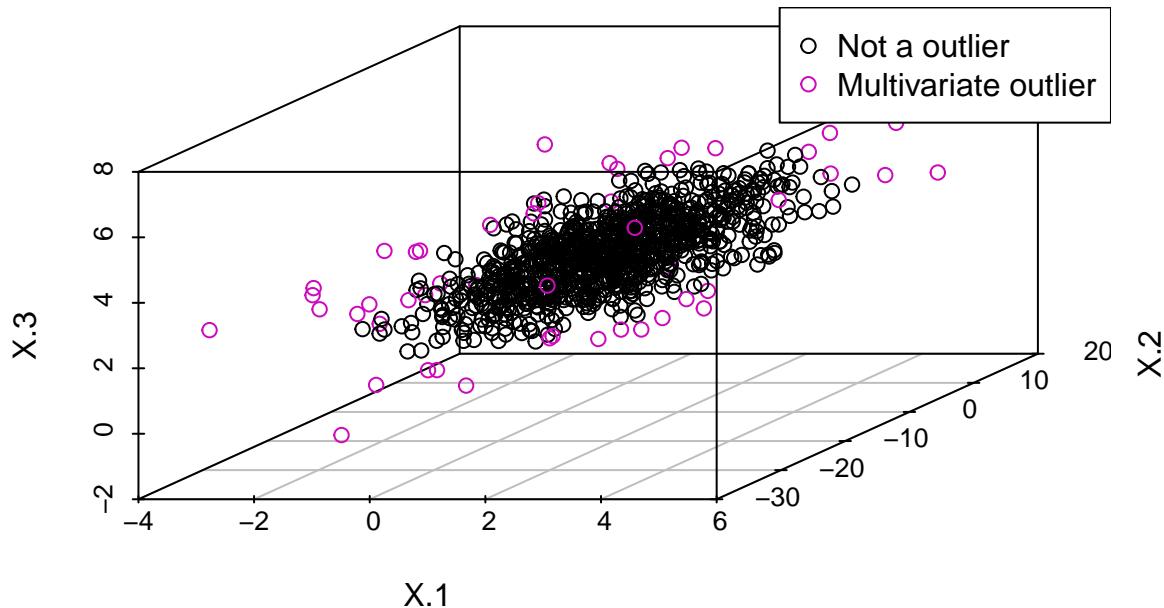
```
my_color <- ifelse(mvar_data$MDrob_out == 1, 6, 1)
pairs(mvar_data[1:3], col = my_color,
      main = "Pairwise plot for robust outliers")
```

## 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



Da ganz viele Daten geplottet werden, sind nicht alle Ausreißer in den Grafiken zu sehen, weil die von anderen Punkten überdeckt werden können. Wenn man sich anschauen will, wieviele Ausreißer jede der Methoden entdeckt hat, kann man dies mit der `length()` für die univariaten Variante Funktion machen (da in der Spalte `univar_out` auch Zahlen gegeben sind die ungleich 0 und 1 sind). So ergibt sich für das erste Modell:

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

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

Für das zweite Modell verwenden wir `sum()` Funktion, da wir alle Vorkommen der 1er aufzählen wollen:

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

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

Und für das robuste Modell:

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

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

g.

Beide Ansätze sind sinnvoll. Die Euklid'sche Distanz geht davon aus, dass die Daten isotop gaußförmig sind, d.h. jede Variable wird als gleich betrachtet. Andererseits misst die Mahalanobis-Distanz Korrelation zwischen Variablen und lockert die Annahme der Euklid'schen Distanz, indem sie eine anisotopische Gauß'sche Verteilung annimmt. Wenn man die Mahalanobis Distanz verwendet, muss man die Daten davor nicht standardisieren, wie bei Euklid'schen Distanz. Das passiert bereits durch Berechnung der Kovarianzmatrix. Wenn man im Vorhinein weiß, dass es eine Art Korrelation zwischen Variablen gibt, würde ich Mahalanobis Distanz bevorzugen.