Exercise 1

Mathematics for Big Data

Sergi Vilardell

Contents

Introduction

In this exercise we will work out the basics of linear regression data simulation and graphics. These are the libraries that we are going to use:

```
library(mnormt)
library(tidyverse)
library(rgl)
```

`tidyverse` is a library that contains a bunch of libraries for data manipulation and visualization. Here will be mostly used for the package `ggplot2` .

Part 1: Data Generation

We start by initialising the variables μ_1 and Σ_1 as follows:

```
mu_1 <- c(3, 1, 13)
sigma_1 <- matrix(c(5, 1.574, -3.352, 1.574, 2, 1.241,-3.352, 1.241, 6),nrow = 3,ncol = 3)
```

These are used to generate a multivariate Gaussian random sample that we will use as a dataframe.

```
#Data set 1
data_1 <- rmnorm(n = 10^3, mu_1, sigma_1)
data_1 <- as.data.frame(data_1)
group<- rep(1, 10^3)
data_1$group <- group
colnames(data_1) <- c("X1", "X2", "X3", "G")
```

The new data sample is in `data_1` and it contains 4 columns, with the 4th being a column of 1s to indicate that it belongs to `data_1`. Now repeat the process for another μ_2 and Σ_2:

```
mu_2 <- c(7, -2, 10)
sigma_2 <- matrix(c(3, -1.385, 2.939, -1.385, 1, -1.979, 2.939, -1.979, 8),nrow = 3,ncol = 3)

#Data set 2
data_2 <- rmnorm(n=10^3, mu_2, sigma_2)
data_2 <- as.data.frame(data_2)
group<- rep(2,10^3)
data_2$group <- group
colnames(data_2) <- c("X1", "X2", "X3", "G")
```

Finally we merge the two datasets into one named `data`:

```
#Merge the datasets
data <- rbind(data_1, data_2)
colnames(data) <- c("X1", "X2", "X3", "G")
```

Part 2: Linear Regression

We will carry out a simple linear regression model using the function `lm()` with the dataset `data_1`:

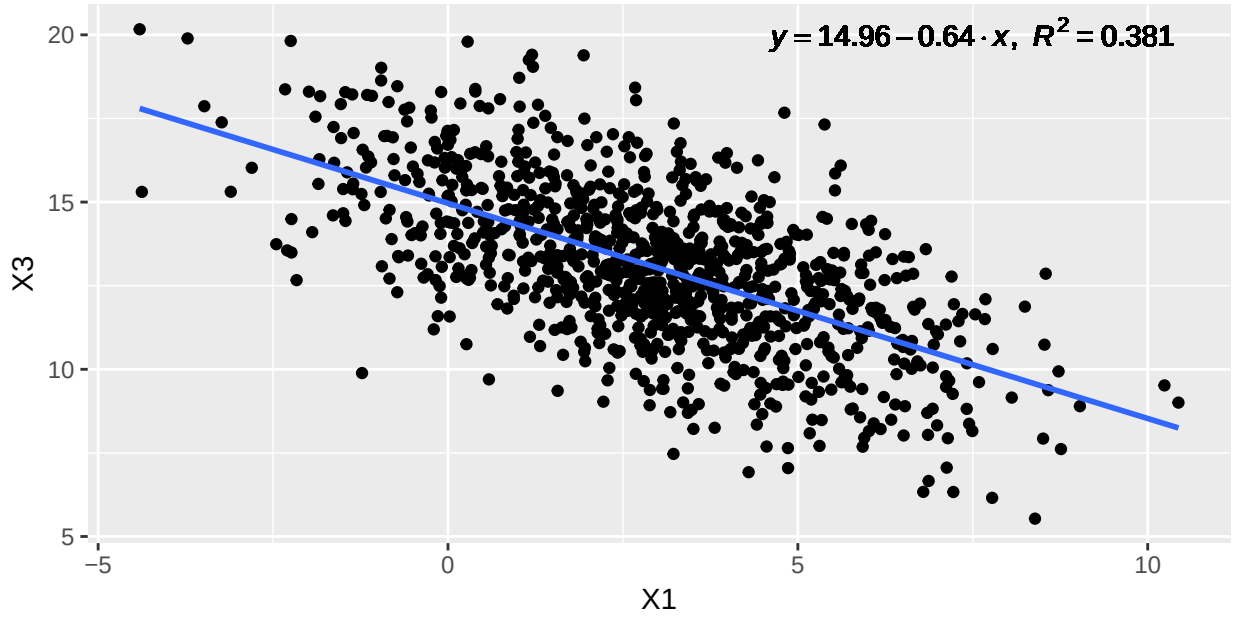
```
#Linear regression data1
mod0 <- lm(X3~X1, data = data_1)
```

We find that `ggplot` is a better tool to make plots, therefore we use it instead of `plot()` to make a graph of the variables involved in the linear model including R^2. Before plotting we create a function that writes the equation predicted by the model and R^2. It is a function that takes the parameters created by `lm()` and displays them in text:

```
#Function to display lm() variables
lm_eqn = function(m) {

  l <- list(a = format(coef(m)[1], digits = 4),
            b = format(abs(coef(m)[2]), digits = 2),
            r2 = format(summary(m)$r.squared, digits = 3));

  if (coef(m)[2] >= 0)  {
    eq <- substitute(italic(y) == a + b %.% italic(x)*","~~italic(R)^2~"="~r2,l)
  } else {
    eq <- substitute(italic(y) == a - b %.% italic(x)*","~~italic(R)^2~"="~r2,l)
  }

  as.character(as.expression(eq));
}
```

Now we can make the plot:

```
#Plot
ggplot(data = data_1, aes(x = X1, y = X3)) +
  geom_point() +
  geom_smooth(method = lm, se = FALSE)+
  geom_text(aes(x = 7.5, y = 20, label = lm_eqn(mod0)), parse = TRUE)
```



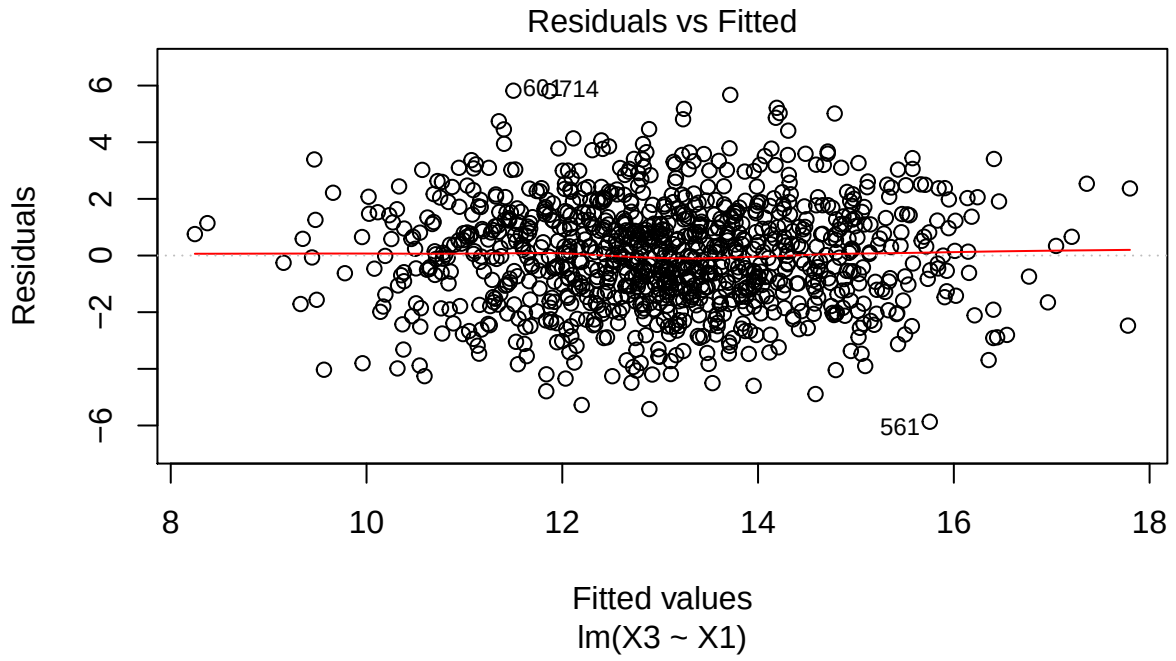
Read the summary of `mod0`:

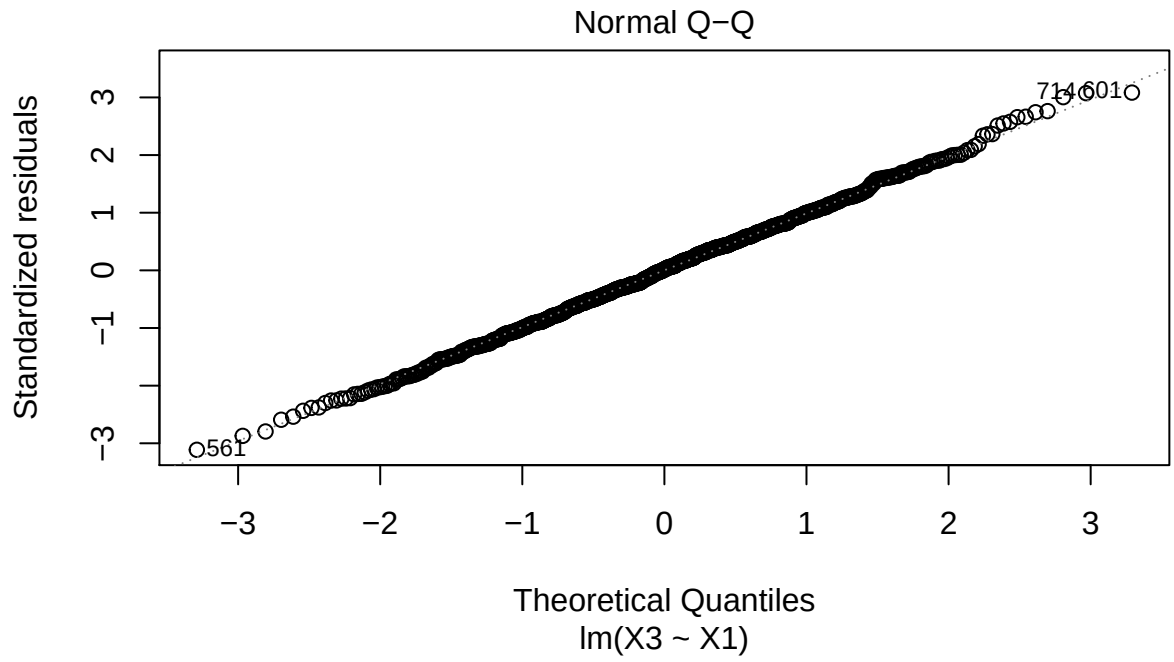
```
sum.mod <- summary(mod0)
sum.mod
```

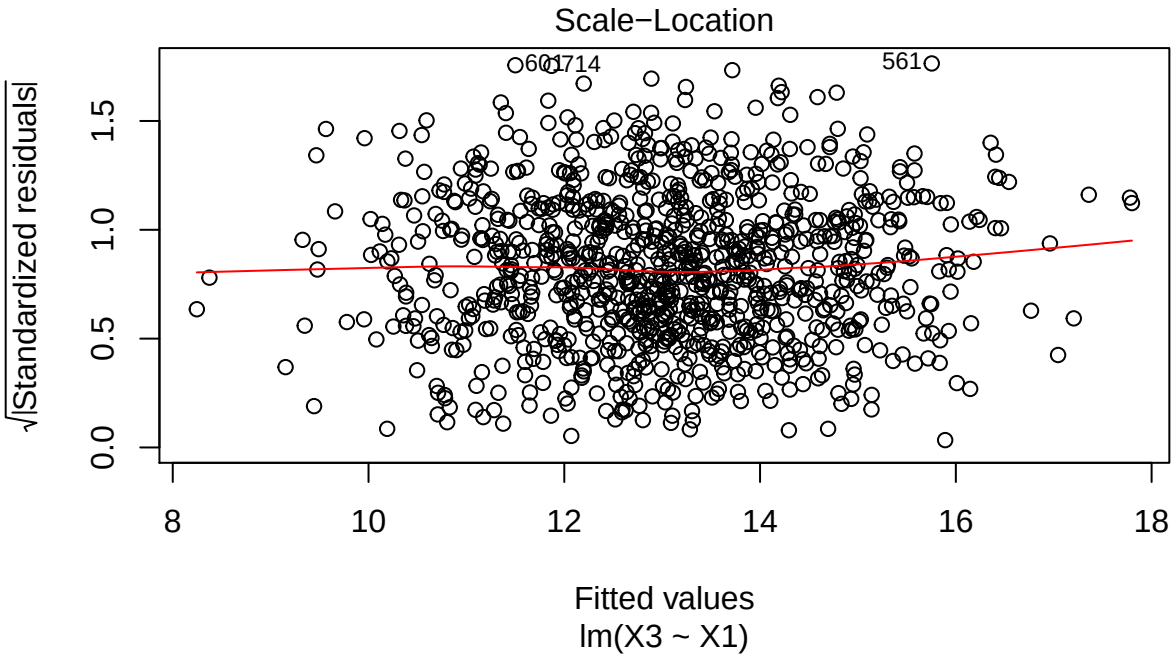
```
##
## Call:
## lm(formula = X3 ~ X1, data = data_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8671 -1.2537  0.0306  1.2583  5.8192
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.96377    0.09718  153.97   <2e-16 ***
## X1          -0.64352    0.02599  -24.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.889 on 998 degrees of freedom
## Multiple R-squared:  0.3805, Adjusted R-squared:  0.3799
## F-statistic: 613.1 on 1 and 998 DF,  p-value: < 2.2e-16
```

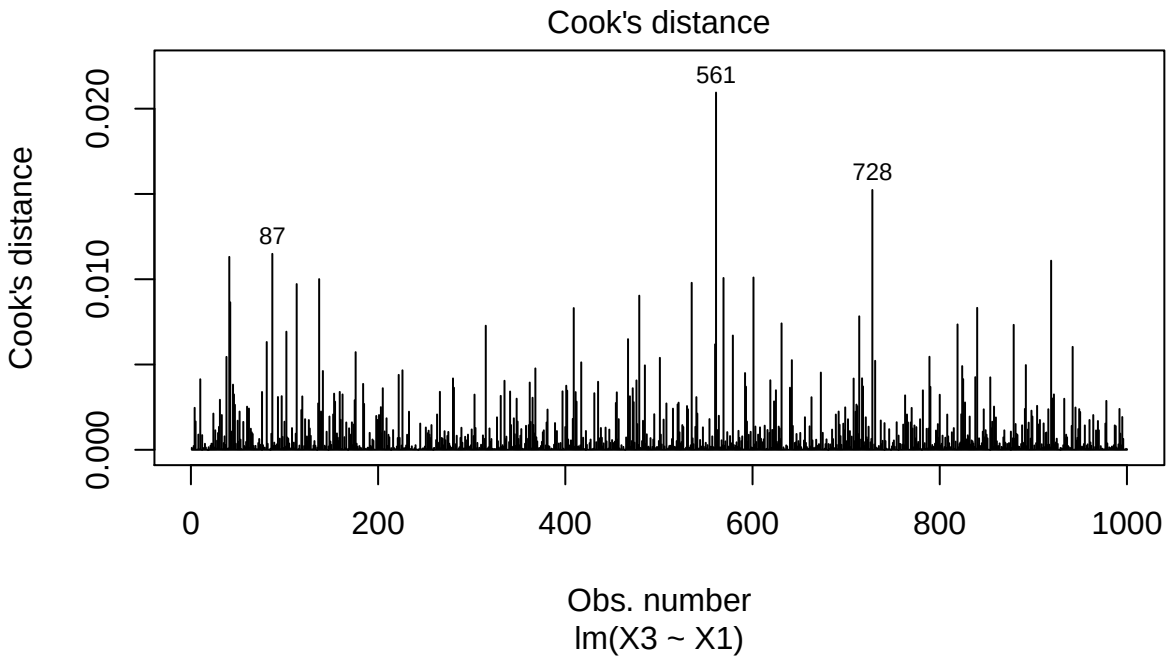
We show also the residual plots for `mod0`:

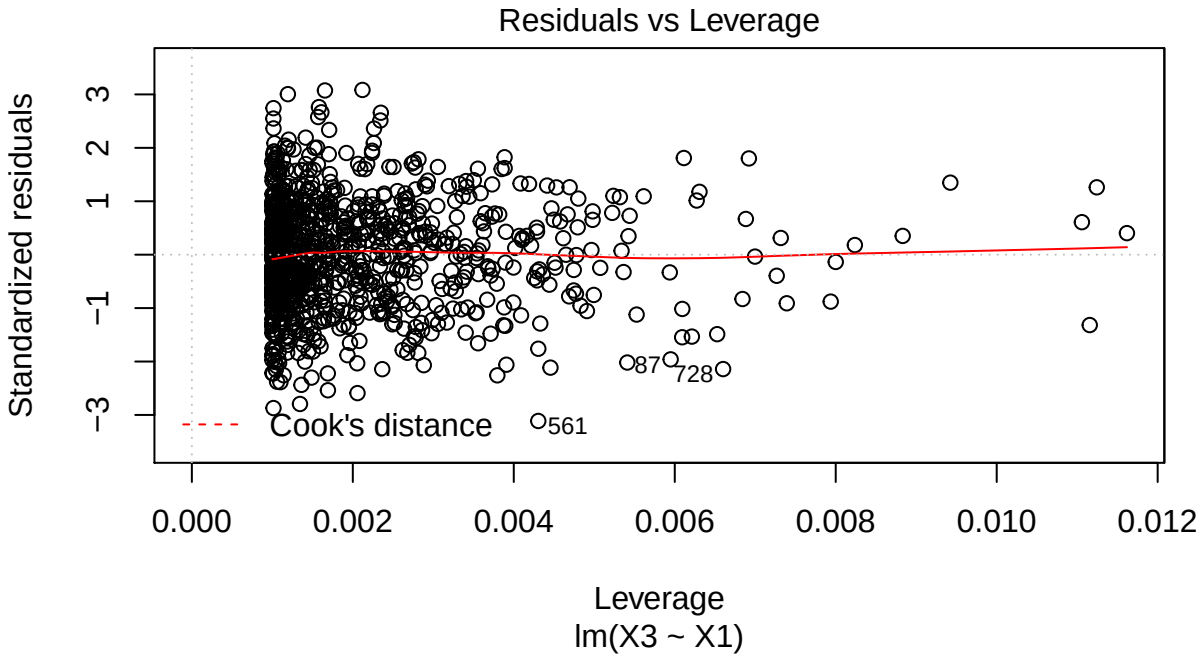
```
#Residual plots
plot(mod0,1:6)
```

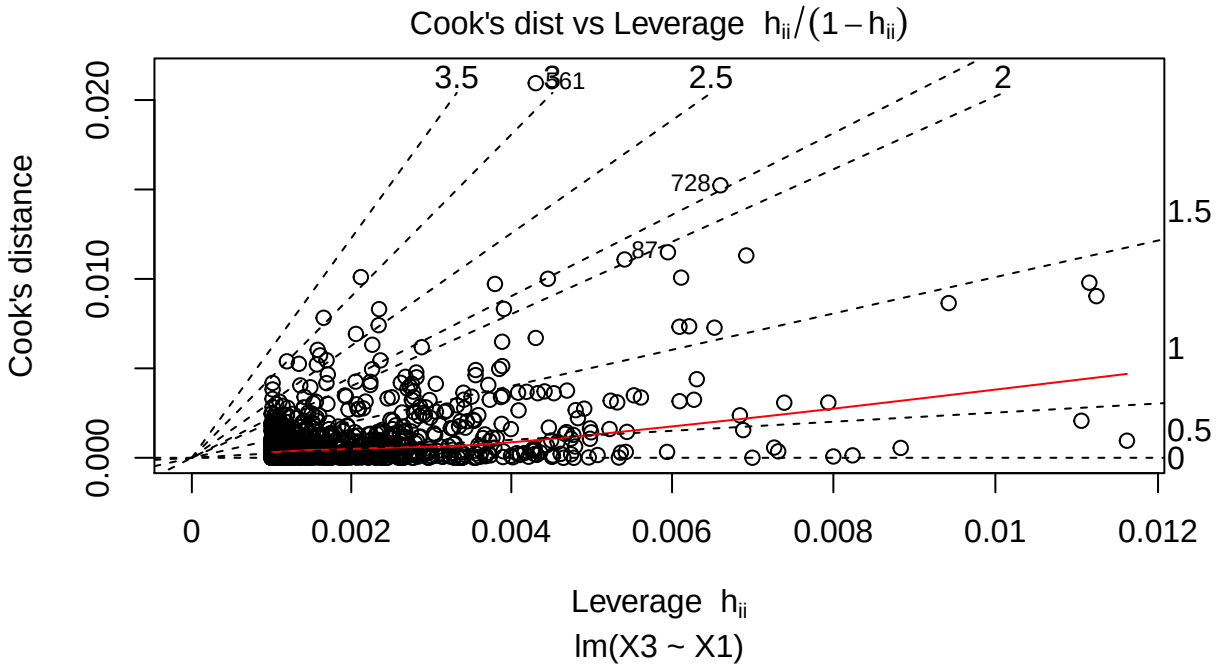



Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(X3 ~ X1)
561
714
601


Scale−Location
√|Standardized residuals|
Fitted values
lm(X3 ~ X1)
601
714
561


Cook's distance
561
728
87
Cook's distance
0.000
0.010
0.020
0
200
400
600
800
1000
Obs. number
lm(X3 ~ X1)


Residuals vs Leverage
Standardized residuals
−3
−1
1
2
3
87
728
Cook's distance
561
0.000
0.002
0.004
0.006
0.008
0.010
0.012
Leverage
lm(X3 ~ X1)



This plots are very useful to get information about the model. For instance the Residuals vs Fitted values; the plot is generated with the Residuals($e_i = y_i - \hat{y}_i$), where y_i is the response and $\hat{y}_i$ are the predicted values for the response. In order to see the trend of the Residuals a fit is generated. In this case the it is a straight line, suggesting that the response and the input values have a linear dependency. This can also be seen in the Normal Q-Q; this is used to determine if the relationship between the input and the response is linear. As the plot generates a straight line we can conlcude that it is a linear dependency, although there are values that do no fit very well into this argument. In both plots there are some points marked with their index number in the dataframe, these are outliers. Outliers can be better identified computing the Cook's distance. One of the plots uses this to help us visualize the points that have negative influence when generating our model, i.e. the points that are worth eliminating to improve the model. In our case, instances 293, 593 and 767 are marked as outliers as they have the biggest Cook's distance.

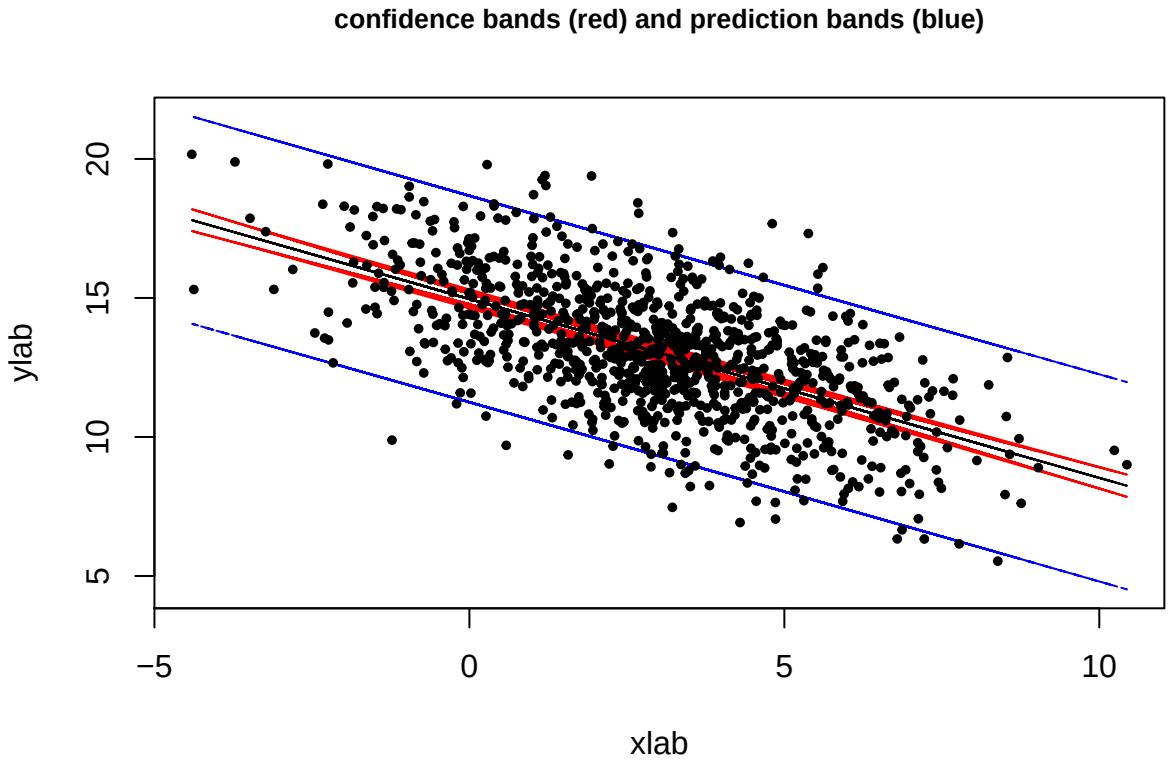
Now let us display the confidence and prediction bands in the simple linear regression model. We use the code provided by the teacher adapted to our code:

```
# confidence band
conf.band<-predict(mod0 ,interval="confidence")
head(conf.band)
```

```
##        fit      lwr      upr
## 1 14.11242 13.96882 14.25603
## 2 13.11509 12.99781 13.23237
## 3 11.97091 11.82507 12.11674
## 4 13.10932 12.99205 13.22659
## 5 14.12534 13.98114 14.26954
## 6 13.10982 12.99255 13.22709
```

```
# prediction, lower and upper bounds
# prediction band
pred.band<-predict(mod0,interval="prediction")
yban<-data.frame(conf.band,pred.band[,-1])
```

```r
# the first column was repeated

matplot(data_1$X1, yban, ylim = c(min(yban), max(yban)),
        lty = c(1,1,1,2,2), col=c(1,2,2,4,4), type = "l",
        ylab = "ylab",xlab="xlab",
        main ="confidence bands (red) and prediction bands (blue)",
        cex.main = .8)
points(data_1$X1, data_1$X3, pch = 20, cex = .8)
```



Part 3: Multiple Linear Model

It is also really simple to perform a multiple linear model with the function `lm()`:

```
#Multiple linear model
mod1 <- lm(X3~X1+X2, data = data_1)
```

The summary for the linear model:

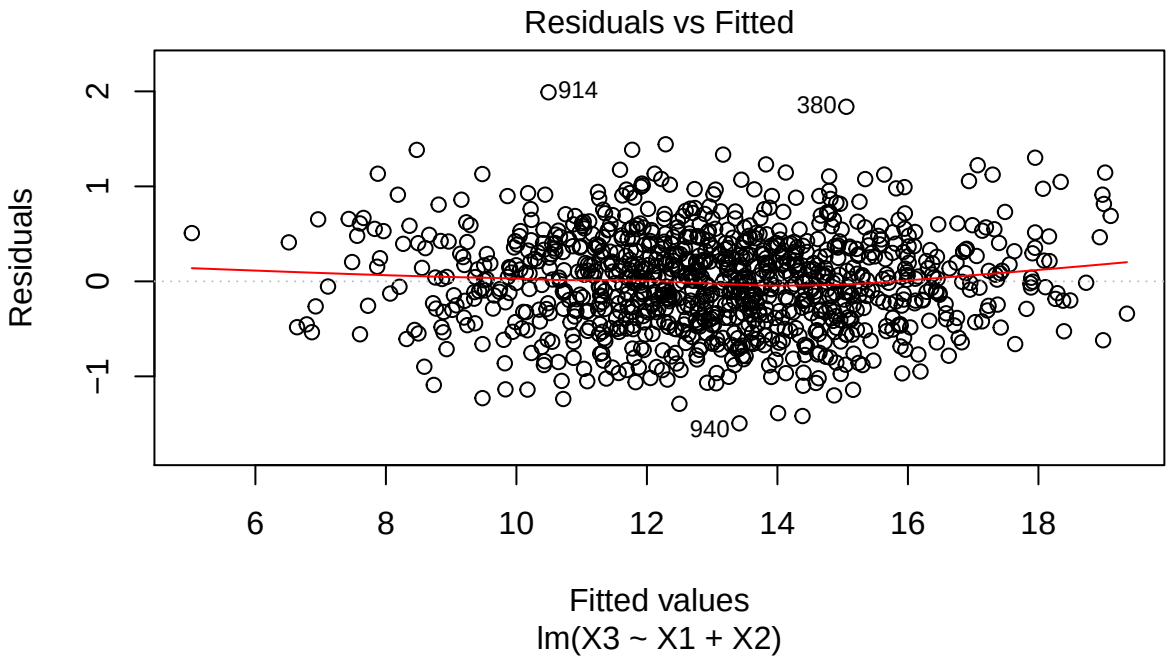
```
#Summary
summary(mod1)
```

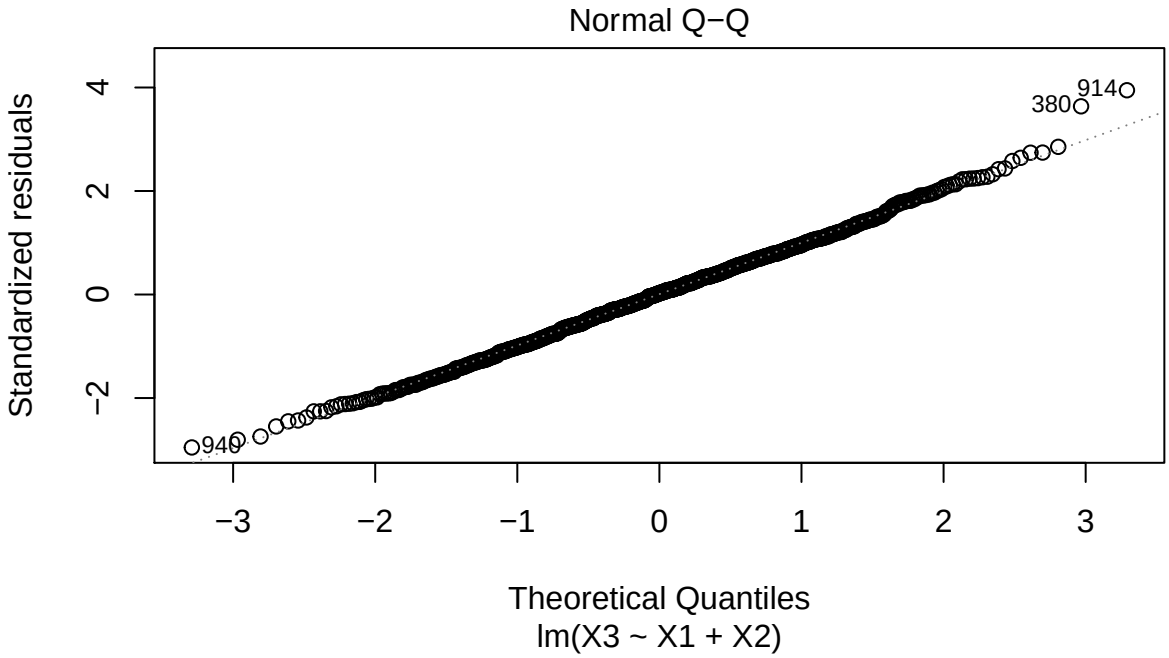
```
##
## Call:
## lm(formula = X3 ~ X1 + X2, data = data_1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.49428 -0.33401  0.01176  0.34182  1.99049
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.942306   0.026033   574.0   <2e-16 ***
## X1          -1.143661   0.008236  -138.9   <2e-16 ***
## X2           1.513424   0.013318   113.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.506 on 997 degrees of freedom
## Multiple R-squared:  0.9556, Adjusted R-squared:  0.9555
## F-statistic: 1.073e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

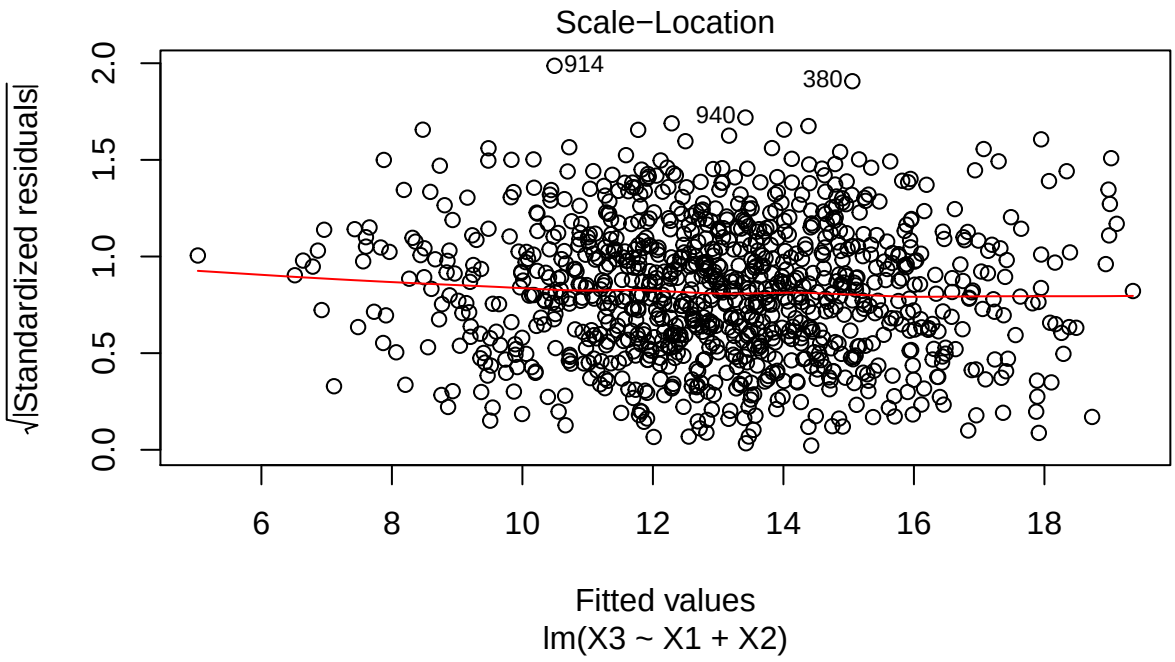
When performing a linear model first and foremost we have to look at the t-value and p-values. Higher absolute t-values allow us to reject the null hypothesis. Lower p-values tell us how trustworthy are the results that we get. In this case the t-values allow us to reject the null hypothesis, and the p-values indicate that this assumption can be made safely. The ajusted R^2 is close to one $R^2 \approx 0.96$, therefore the model fits the data well.

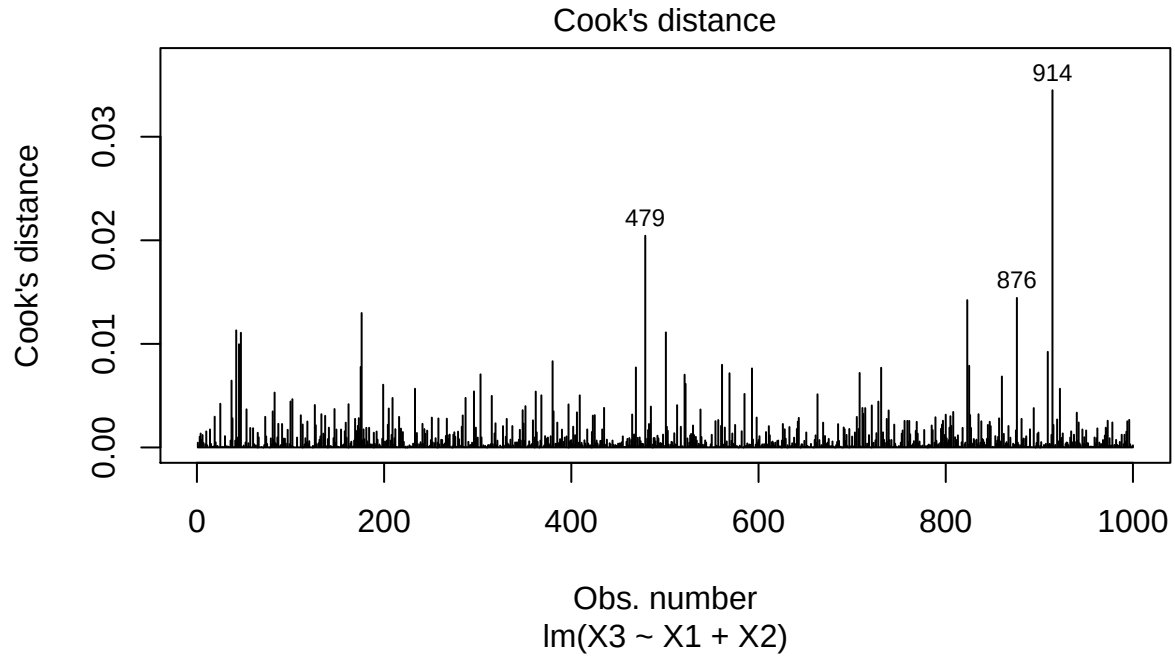
The residual plots for this model are:

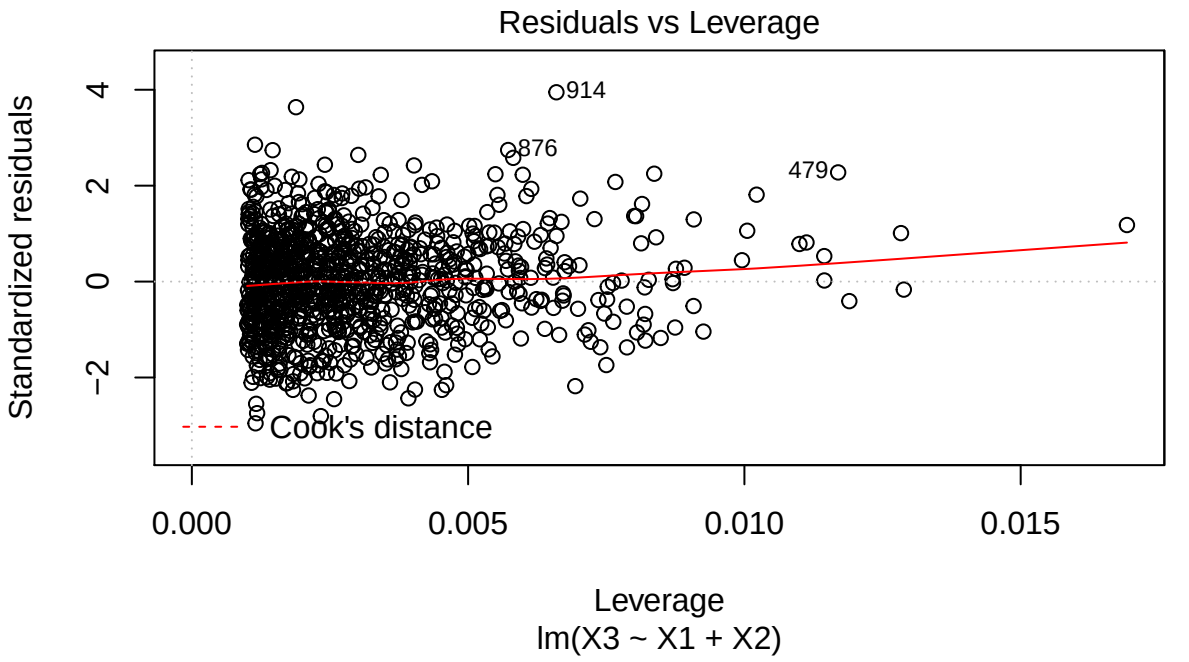
```
#Residual plots
plot(mod1, 1:6)
```

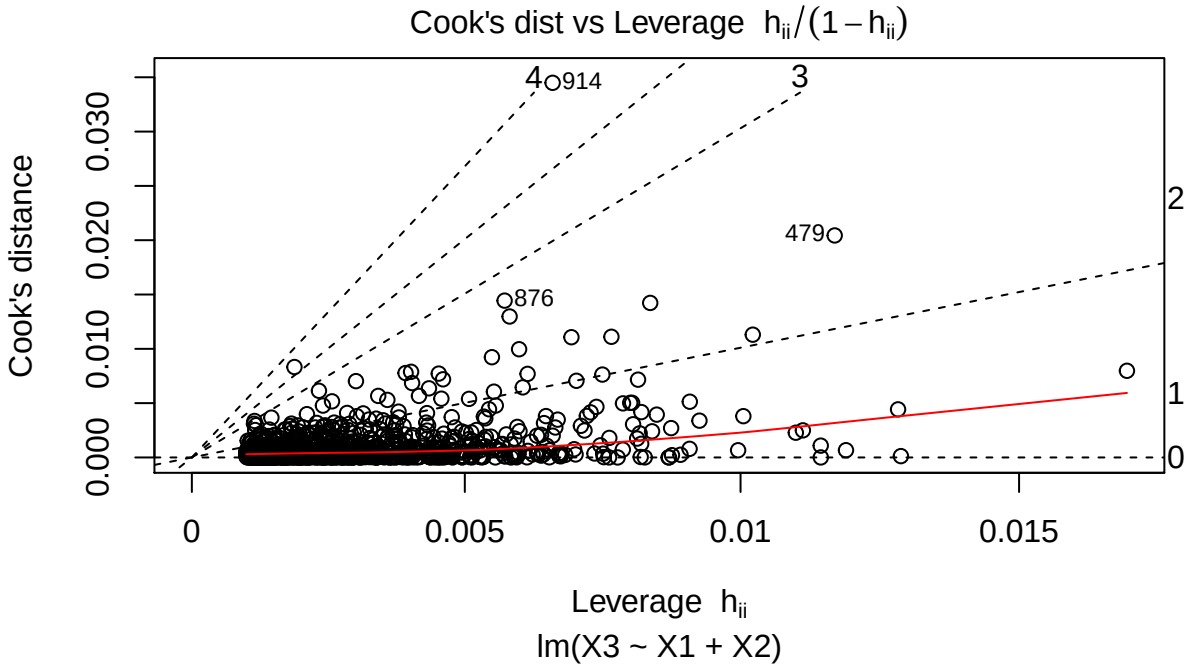

Residuals vs Fitted
914
380
940
Residuals
2
1
0
−1
6
8
10
12
14
16
18
Fitted values
lm(X3 ~ X1 + X2)


Normal Q−Q
914
380
940
Standardized residuals
4
2
0
−2
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(X3 ~ X1 + X2)


Scale−Location
√|Standardized residuals|
914
380
940
0.0
0.5
1.0
1.5
2.0
6
8
10
12
14
16
18
Fitted values
lm(X3 ~ X1 + X2)


Cook's distance
Cook's distance
914
479
876
0.00
0.01
0.02
0.03
0
200
400
600
800
1000
Obs. number
lm(X3 ~ X1 + X2)





The residuals plots are very similar to the simple linear model. The Residuals vs Fitted values as well as the Normal Q-Q plot show us a linear dependency on the predictors.

As a last exercise we will predict a new observation using this model:

```r
input <- data.frame(X1 = 2, X2 = 0)
predict.lm(mod1, input, interval = "confidence")
```

```
##        fit     lwr      upr
## 1 12.65499 12.6167 12.69327
```

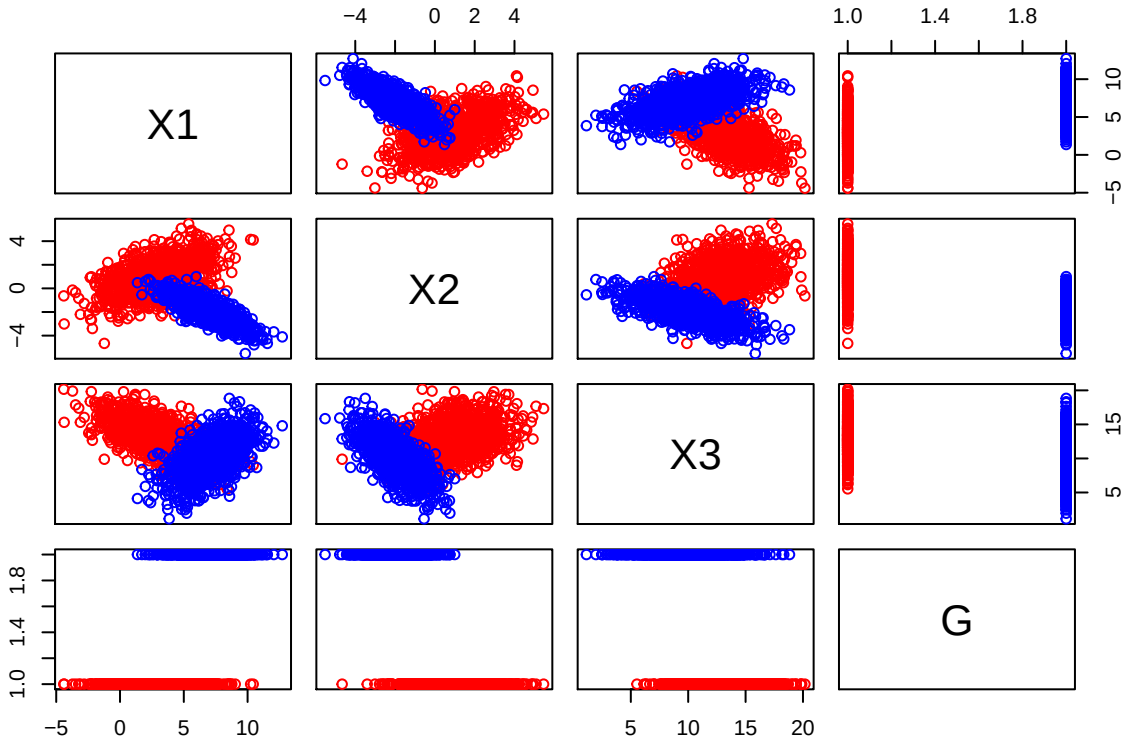
```r
predict.lm(mod1, input, interval = "prediction")
```

```
##        fit      lwr      upr
## 1 12.65499 11.66136 13.64861
```

Part 4: More Graphics

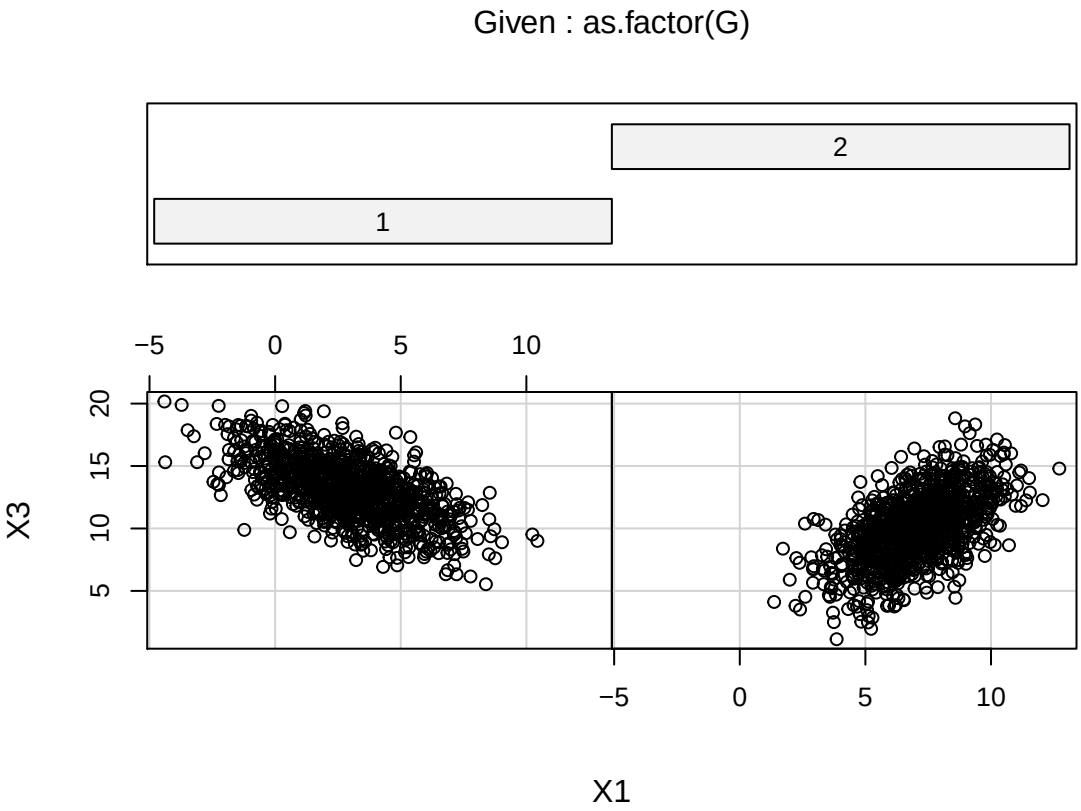
It is interesting to draw a scatterplot of each variable against each other in the dataframe `data`:

```
plot(data,col = c("red","blue")[data$G])
```



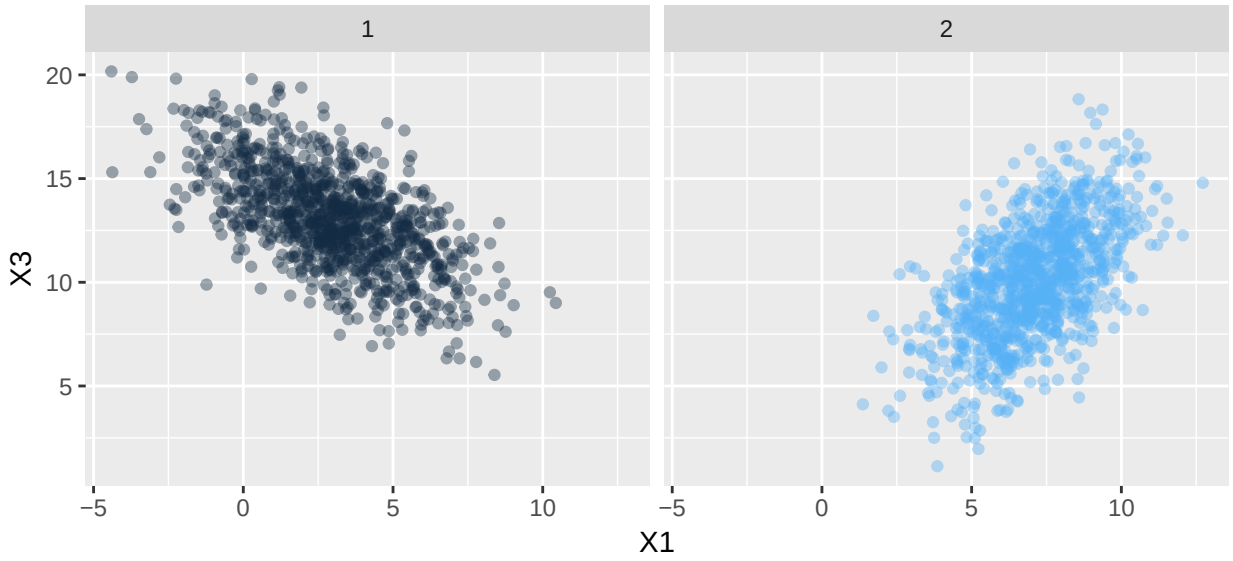
In each plot the difference between the two groups is that they always have different signs in the slope, in case we performed a linear model. We can separate the two groups into different plots using `coplot()` to compare them:

```
#Coplot
coplot(X3~X1|as.factor(G), data)
```



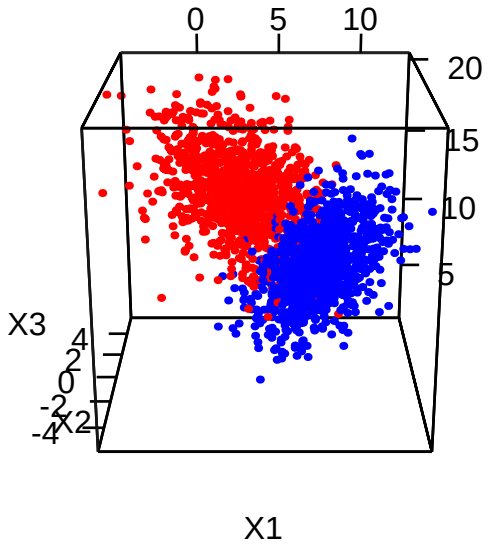
However we found that a neater display is given using `ggplot`:

```
#ggplot
ggplot(data, aes(x = X1, y = X3)) +
  geom_point(aes(colour = G), alpha = 0.4, show.legend = FALSE) +
  facet_wrap(~ G)
```



Finally we make a 3D plot displaying all three variables:

```
plot3d(data,col = c("red", "blue")[data$G])
rgl.postscript("dplot.pdf", "pdf")
```



The last exercise could not be performed because of problems with the necessary packages that include the functions that have to be used.