COMPUTER LAB 3

Today's Lab is divided in two parts:

- First Part: how to measure the relationship between two numerical variables.
- Second Part: Linear regression.

Before we start, we need some preparations. We will be using the datasets `shoevsheight.csv`, `salary.xslx`, `salary2.csv`, `FevChildren.RData` and `marketing.csv` that can be found in Athena under the folder for the third computer lab. Download the files into a folder in your computer. Then change R's working directory to that folder: click on the `Session` menu at the top of the screen, then `Set working directory`, then `Choose directory` and then navigate to the folder where you downloaded the datasets.

Let us install and load the packages `alr4`, `openxlsx` and `carData`. By now you should be familiar with the process of installing and loading a package in R. Recall that the installation is needed only once, whereas the packages have to be loaded (through the command `library()`) every time we start Rstudio.

```
install.packages("alr4") #this is needed just for the first time
install.packages("openxlsx") #this is needed just for the first time
install.packages("carData") #this is needed just for the first time
library(alr4)
library(openxlsx)
library(carData)
```

1 First Part: relationship between two numerical variables

For this part of the computer lab we are going to use the `iris` dataset that we used in Computer Lab 2. It is a built-in dataset. Therefore, in order to load it, we can simply use the function `data()`.

```
data(iris)
```

This dataset consists of measurements of four variables made on three species of flowers iris.

```
head(iris)
str(iris)
```

In order to study the relationship between two numerical variables we will make use of scatter plots and the correlation coefficient.

1.1 Scatter plots

Creating a scatter plot in R is pretty straightforward by means of the function `plot()`. For example, we may be interested in studying the relationship between the two variables `Petal.Length` and `Petal.Width`.

```
plot(iris$Petal.Length, iris$Petal.Width)
```

It is important to rename the labels of the axes. In order to do that we can specify the arguments `xlab` and `ylab`. If we want to change the colors of the points, we can provide the argument `col`. We can add the main title with the argument `main`. Further arguments are presented in the appendix.

```
plot(iris$Petal.Length, iris$Petal.Width, xlab = "Petal Length",
     ylab = "Petal Width", col = "navy")
```

We should be very cautious when placing the variables on the x and y axis. If one of the variables can be considered to be dependent on the another one, it is common practice to place this dependent variable on the y axis and the another one on the x axis. Consider, for instance, the variables *salary* and *years of education*. It is natural to think that the salary depends on the years of education, not the other way around. Therefore, the variable salary should be placed on the y axis and the variable years of education on the x axis.

There is an alternative use of the function `plot()` for creating scatter plots. Instead of using two variables as input, we can input an entire dataset *containing only numerical variables*. In this case, the output will be the scatter plots for all pair of variables.

We illustrate this use of the function `plot()` with the `iris` dataset. It contains five variables, four of them are numerical and one is categorical, namely, `Species`. Therefore, first we need to "get rid" of this categorical variable. We can do that as follows:

```r
iris.num = iris[, 1:4]
plot(iris.num)
```

> **Subsetting in R**
>
> R allows to retrieve some specific cells of a dataset (or a matrix, a vector or an object, in general) in a quite simple way. For instance, consider the `iris` dataset,
>
> - we can retrieve the value of the second variable for the fifth observation (so fifth row, second column) by using `iris[5,2]`;
> - we can retrieve the value of the fourth variable for the third, fourth and fifth observations by using `iris[3:5,4]`;
> - we can retrieve the value of the first variable for the third, seventh and tenth observations by using `iris[c(3, 7, 10),1]`;
> - we can retrieve the value of all observations for the fifth variable by using `iris[,1]`;
> - we can retrieve the value of all observations for the first four variables by using `iris[,1:4]` (as we did in the example before);
> - and so on...

From the plot, we "know" that there is a positive relationship between `Petal.Length` and `Petal.Width`. So, let us quantify it.

1.2 The correlation coefficient

The correlation coefficient is a parameter that indicates the degree of lilnear association between two numerical variables. It indicates both, the direction and the strength of the linear association between the two variables. By construction, it takes values between -1 and 1. Values close to 1 indicate a strong positive linear association. Values close to -1 indicate a strong negative linear association. When the correlation is equal to 0 there is no linear relationship between the two variables.

To get the correlation between two variables, the function `cor()` is used as follows:

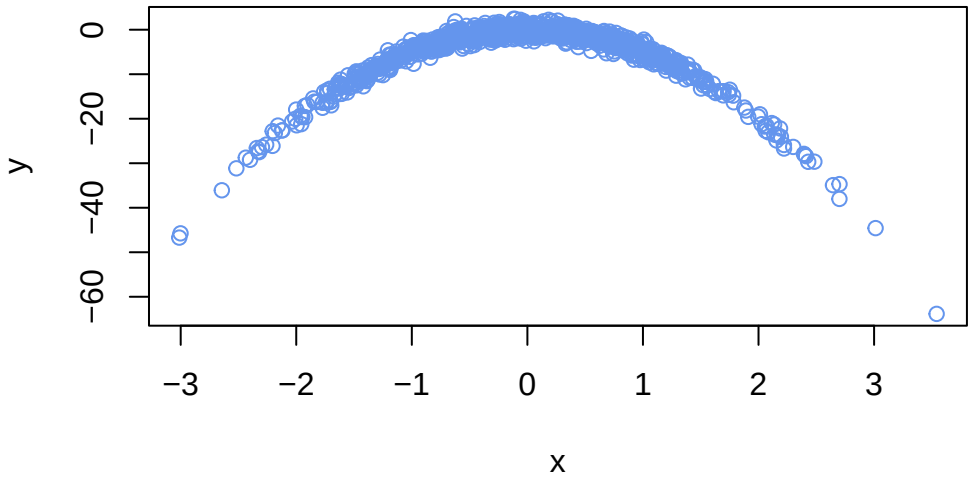
```r
cor(iris$Petal.Length, iris$Petal.Width)
```

Similarly to what we did with scatter plots, it is possible to calculate the correlation between all pairs of variables in a dataset that contains *only numerical variables*. The result is known as the correlation matrix:

```r
cor(iris[, 1:4])
```

It is extremely important to remind the fact that the correlation measures the **linear** relationship between two variables. If the correlation is equal 0, it does not necessarily mean that there is no relationship between the two variables. It means that there is no **linear** relationship between them.

Let us consider the following two variables x and y.



The correlation between the two variables x and y is basically 0, indicating that there is no linear association between them. This is evident from the scatter plot, but it is also evident that *there is* some relationship between the two variables x and y, a quadratic relationship in this case.

Spearman's and Kendall's correlation coefficients

By default, the function `cor()` calculates Pearson's correlation coefficient. But it can also be used for calculating Spearman's and Kendall's correlation coefficients by means of the argument `method` as follows:

```
cor(iris$Petal.Length, iris$Petal.Width)
cor(iris$Petal.Length, iris$Petal.Width, method="spearman")
cor(iris$Petal.Length, iris$Petal.Width, method="kendall")
```

EXERCISE

- Obtain scatter plots between all pairs of numerical variables in the built-in dataset `mtcars`. Take into account that the variables `cyl`, `vs`, `am`, `gear` and `carb` are categorical.
- Obtain the (Pearson's) correlation coefficient between all pairs of numerical variables.
- Obtain the Spearman's correlation coefficient between all pairs of numerical variables.
- Obtain the Kendall's correlation coefficient between all pairs of numerical variables.

2 Second part: Linear regression

In this second part we are going to talk about linear regression. Linear regression is a very useful tool that allows us to describe the relationship between a dependent variable (response) and some independent variables (explanatory variables).

2.1 Simple Linear Regression

In this part of the lab we are going to make use of the dataset `shoevsheight.csv` which we downloaded at the beginning of the session. It consists of three variables: the shoe size (variable `shoe`), the height (variable `height`, in cm) and the sex (variable `gender`, 1=male, 2=female) measured on 94 students. We are mainly interested in the first two variables. We can read the dataset using

```
shoe_data <- read.csv("shoevsheight.csv")
```

Let us take a quick look at the structure of the dataset

```
head(shoe_data)
str(shoe_data)
```

We are going to illustrate how to use simple linear regression to study the relationship between height and shoe size. In particular, we are interested in studying how the height affects the shoe size. This means that `height` is the explanatory (independent) variable and `shoe` is the dependent variable.

Before fitting the model, it is always useful to create a scatterplot. Recalling what we have said before, `height` should be placed on the x-axis and `shoe` should be placed on the y-axis.

```r
plot(shoe_data$height, shoe_data$shoe, ylab = "Shoe size", xlab = "Height")
cor(shoe_data$height, shoe_data$shoe)
```

The relationship looks linear and the correlation is almost 0.8. Therefore, the simple linear regression is an adequate model for these data. Later, we will see how the linear regression model can still be used when the association between the variables is not linear.

The function `lm()` can be used to fit a linear regression. The syntax is the following: `lm(dep. variable ~ indep. variable, data = dataset)`.

```r
simple_reg <- lm(shoe ~ height, data = shoe_data)
```

The function `summary()` can be used to get the main results from the fitted model in a single output.

```r
summary(simple_reg)
```

> **Note**
>
> For the moment, you are able to recognize only few things: the residuals, the estimates of the coefficients, Multiple R-squared and the Adjusted R-squared. Don't worry! At the end of the second part of the course you will recognise everything!

> **EXERCISE**
>
> Calculate the coefficients of the regression above using the formulas from the Lecture notes!

Let us interpret a bit these results:

- Almost 60% of the variation of shoe size is explained by the height;
- The fitted regression line is
$$\hat{y} = -1.823 + 0.246 \cdot x$$
- This means that -1.823 is the expected shoe size when height is equal to 0 cm. Of course, this interpretation is meaningless. It is always important to pay attention when interpreting the intercept, as, in some cases, it does not make any sense.
- For each additional cm in height, the shoe size increases on average by 0.246 units.

It is always useful to plot the estimated regression line with the data to see how the model fits. The function `abline()` can be used to add a straight line to an existing plot. A useful property of this function is that we just need to provide the estimated model and it automatically plots the estimated line.

```
plot(shoe_data$height, shoe_data$shoe, ylab = "Shoe size", xlab = "Height")
abline(simple_reg, col = "orange")
```

It is also possible to get the estimated parameters individually. In order to do so, the `$` sign is used.

> **Note**
>
> So far, we have used the `$` sign to select a certain variable from a dataset. We do not delve into the details here, but everything that is created in R is called an object. This means that a dataset, a table and even the model that we have just created are objects. Formally the `$` sign is used to retrieve the attributes of an object. In order to see the attributes of an object, we can use the function `attribute()`.
>
> ```
> attributes(simple_reg)
> ```
>
> To summarize, the `$` sign is used to retrieve properties of an object. In the case of a dataframe (which is an object), we can use the `$` sign to get the variables contained in the dataframe. In the case of the linear regression model, the `$` sign can be used to retrieve the coefficients of the model, the fitted values, the residuals, etc.

Let us get the coefficients.

```
simple_reg$coefficients
```

Once having estimated the coefficients, we can obtain the fitted values

$$\hat{y}_i = b0 + b1 \cdot x_i = -1.823 + 0.246 \cdot x_i.$$

These can be obtained as

```
simple_reg$fitted.values
```

We can also find the residuals:

$$e_i = y_i - \hat{y}_i$$

as follows

```
simple_reg$residuals
```

EXERCISE

Calculate the fitted values and the residuals using the formulas from the Lecture notes!

Up to this point we have used the linear regression for describing the relation between the dependent and the independent variable. In practice, linear regression is commonly used also for predictive purposes, i.e. we can predict the value of the dependent variable for an element with a value of x_0 for the independent variable. For example, we would like to predict the shoe size for two persons with heights 168 and 183. The predicted values are given by the formula

$$\hat{y} = b_0 + b_1 \cdot x_0,$$

where b_0 is the estimated intercept, b_1 is the estimated slope and x_0 is the value of the explanatory variable for which we want to predict the value of the dependent variable. As we want to predict the shoe size for two persons with heights 168 and 183, we have two values $x_0 = 168$ and $x_0 = 183$.

The function `predict()` can be used to this end. This function takes as input the estimated model and the argument `newdata` needs to be specified. The latter should be a dataframe containing one variable having the same name as the independent variable that we used to fit the model (`height`), with the values that we want to predict at. In fact, this function is able to give us predictions for more than one x_0 at the same time.

```
new_x <- data.frame(height = c(168, 183))
predict(simple_reg, newdata = new_x)
```

EXERCISE

Calculate the predicted values using the formulas from the Lecture notes!

EXERCISE

Load the built-in dataset `cars`. Fit a simple linear regression model with the stopping distance as dependent variable and speed as independent variable.

- Create a scatterplot to study the relationship between the two variables.
- Interpret the coefficients and R^2.
- Add the regression line to the scatterplot.
- Predict the stopping distance when speed is 5, 6 and 21.

2.2 Modelling non-linear associations

In the previous subsection, we have seen that the regression line fits quite well the data because the relationship between the two variables was basically linear. However, what can we do when the relationship is not linear?

We can still use the linear regression model! In fact, we can just apply a non-linear transformation to the data trying to "linearise" the association between both variables.

2.2.1 Transformation of the response variable

In this section we will make use of the dataset `salary.xlsx` which we downloaded at the beginning of the session. This dataset contains data on years of education (variable `years_education`) and salary (variable `salary`, in EUR) for 300 people. We are interested in studying how the years of education affect the salary. We can read the dataset as follows:

```
salary_data <- read.xlsx("salary.xlsx")
head(salary_data)
```

Let us create the scatterplot for these two variables.

```
plot(salary_data$years_education, salary_data$salary)
```

We can clearly see that the relationship between years of education and salary is not linear. Therefore, the linear regression model is not adequate. Let us add the regression line to the scatter plot.

```
my.reg <- lm(salary ~ years_education, data = salary_data)
summary(my.reg)
plot(salary_data$years_education, salary_data$salary, xlab = "Years of education",
     ylab = "Salary")
abline(my.reg, col = "orange")
```

When the dependent variable y has a non-linear relationship with the independent variable x, it may be useful to transform the response variable. The most common transformation is the log-transformation.

Let us present the scatterplot between x and $log(y)$.

```
plot(salary_data$years_education, log(salary_data$salary),
     xlab = "Years of education", ylab = "log(Salary)")
```

We can see now that, after having applied the log transformation, the relationship looks more linear!

The new model is then the following:

$$\widehat{log(y)} = \hat{\beta}_0 + \hat{\beta}_1 \cdot x$$

Fitting this model requires a minor change in the specification:

```
log_response <- lm(log(salary) ~ years_education, data = salary_data)
summary(log_response)
```

As you can see, the syntax is almost the same as before, the only difference being that now we use `log(salary)` as the response variable . This means that we can still fit a linear regression model, after having transformed the response variable!

The estimated model is then:

$$\widehat{log(y)} = 6.85 + 0.109 \cdot x$$

```
plot(salary_data$years_education, log(salary_data$salary),
     xlab = "Years of education", ylab = "log(Salary)")
abline(log_response, col = "orange")
```

Everything has a price and therefore we should pay attention when interpreting the results of this model. We are not explaining the salary any more, we are explaining the logarithm of the salary and the interpretation of the model has to take this fact into account. For instance, we see that almost 15% of the variability of `log(salary)` is explained by the years of education. The same applies to the coefficients.

2.2.1.1 * Plotting the regression line on the original scale

In the previous plot we have shown the fitted linear regression for log(salary) and years of education. We may be interested in plotting the function on the original scale of salary. `abline()` cannot be used to plot the function on the original scale, as it will be a non-linear function and this function is just able to plot straight lines.

In order to get back to the original scale, we need to apply the inverse transformation `exp()`. Therefore:

$$exp(\widehat{log(y)}) = exp(b_0 + b_1 \cdot x) = exp(6.85 + 0.109 \cdot x)$$

which leads to

$$\hat{y} = exp(b_0) \cdot exp(b_1 \cdot x) = exp(6.85) \cdot exp(0.109 \cdot x)$$

Let us write a function in R that returns the value of $\hat{y}$ for any desired value of x:

```
reg_function <- function(x){exp(6.85) * exp(0.109*x)}
```

The function `reg_function()` takes `x` as input and returns `exp(6.85) * exp(0.109 * x)` as output.

Most of the functions in R are "vectorised". This means that if we provide a vector, then we can compute the function for all the values of the vectors simultaneously. Then we are going to define a grid of values for x and we will evaluate the function at those values:

```
x.grid <- seq(from = min(salary_data$years_education),
              to = max(salary_data$years_education), length = 1000)
yhat = reg_function(x.grid)
```

Now we can add the curve defined by `x.grid` and `yhat` to the scatter plot. We can use The function `lines()` is then used to add a line to an existing plot. The arguments are the same as the function `plot()`. We just need to provide the x and y-values of the function that we would like to plot.

```
plot(salary_data$years_education, salary_data$salary,
     xlab = "Years of education", ylab = "Salary")
lines(x.grid, yhat, col = "orange")
```

As we can see, the model is not linear on the original scale.

2.2.1.2 * Predicting values on the original scale

Now, let us say that we want to make a prediction for the response variable for a certain value x_0 and the model is fitted on the log-scale:

$$\widehat{log(y)} = 6.85 + 0.109 \cdot x_0$$

We would like to get the prediction on the original scale. We proceed in the same way we did before. So, let us say that we want to predict the salary for a person with ten years of education, i.e. we want to find $\hat{y}$ when $x_0 = 10$, we use:

$$\hat{y} = exp(6.85) \cdot exp(0.109 \cdot x_0)$$

We can use our previously defined function to this end:

```
yhat = reg_function(20)
```

Note

Other transformations than the log are possible. Other common transformations and their inverse are summarised in the following table.

Transformation of the response	Original scale
y^2	$(y^2)^{1/2}$
y	$\hat{y}$
$y^{1/2}$	$(y^{1/2})^2$
$\log(y)$	$\exp(\log(y))$
$-y^{-1/2}$	$(-y^{-1/2})^{-2}$
$-y^{-1}$	$-(-y^{-1})^{-1}$

EXERCISE

Download the file `salary2.csv` on Athena. This dataset contains the years of experience and the salary for the employees of a certain company.

- Fit a simple linear regression that explains the salary in terms of the years of experience.
- Plot the estimated regression line. Do you think that the relationship is linear?
- Apply now the log-transformation to the response variable `salary` and estimate the model.
- Create the scatterplot for *log(salary)* and *years of experience* and add the estimated line. What do you think now? The relationship is more linear?
- Interpret the coefficients of the transformed model.
- Plot the estimated curve on the original scale.
- Predict the salary of two employees having 5 and 4.2 years of experience.

2.2.2 Transformation of the independent variable

In some situations, it is necessary to transform also the dependent variable.

Let us consider another example. For this part we can load the built-in dataset `mtcars`, that you have used throughout the Computer Labs.

```
data(mtcars)
head(mtcars)
```

We are interested in the regression that explains the miles per gallon (variable `mpg`) in terms of the horsepower (variable `hp`).

Let us investigate the relationship between the two variables.

```
plot(mtcars$hp, mtcars$mpg, xlab = "hp", ylab = "mpg")
```

We can see some non-linear association between both variables. So, let us apply a log transformation to the response.

```
plot(mtcars$hp, log(mtcars$mpg), xlab = "hp", ylab = "log(mpg)")
```

There has been an improvement with respect to the previous case. But, let us try to apply the logarithmic transformation also to the independent variable.

```
plot(log(mtcars$hp), log(mtcars$mpg), xlab = "log(hp)", ylab = "log(mpg)")
```

We can see that now the relationship looks linear!

The simple linear model can still be used in this situation. In fact, we just need to provide the two transformed variables!

```
my.reg2 <- lm(log(mpg) ~ log(hp), data = mtcars)
summary(my.reg2)
```

The fitted model is:

$$\widehat{\log(y)} = 5.545 - 0.53 \cdot \log(x)$$

> **Note**
>
> When the independent variable x is transformed, there is no simple interpretation for the estimated coefficient b_1, regardless of the fact the response variable y is transformed. The interpretation will not be covered in this course!

Let us plot the linear regression line.

```
plot(log(mtcars$hp), log(mtcars$mpg), xlab = "log(hp)", ylab = "log(mpg)")
abline(my.reg2)
```

It fits well the data!

If we want to obtain the miles per gallon on the original scale, the inverse transformation must be applied:

$$\widehat{log(y)} = 5.545 - 0.53 \cdot log(x) \Longrightarrow \hat{y} = exp(5.545) \cdot x^{-0.53}$$

Note

The logarithmic transformation is the most common transformation that we can apply to our variables. Of course, it is possible to apply all the combinations of transformations that we have presented before to both the response and independent variable!
Another useful "transformation" will be shown in the next paragraph!

EXERCISE

Plot the estimated line on the original scale of the two variables.

EXERCISE

Load the dataset `brains` from the package `alr4`. This dataset contains the average body weight in kilograms and the average brain weight in grams for 62 species of mammals. We want to fit the regression that explains the brain weight in terms of the body weight.

- Present an adequate scatterplot and fit a simple linear regression with the brain weight as the dependent variable and plot the estimated line. Do you think that the relationship is linear?
- Apply now the log-transformation just to the response variable. Create the scatterplot for *log(brain weight)* and *body weight*. Fit the model and add the estimated line. What do you think now? Has the situation improved?
- Apply now the log-transformation to both the response and the independent variable. Create the scatterplot for *log(brain weight)* and *log(body weight)*. Fit the model and add the estimated line. And now?
- Plot the estimated curve on the original scale of both variables.
- Predict average brain weight of two species with average body weight equal to 20 and 50.

2.3 Multiple Linear Regression

It is now time to use more than one independent variable in our model. In order to illustrate the use of multiple linear regression model, we are going to use the dataset `FevChildren.RData` that we used in Lab 2. It contains data on the forced expiratory volume (variable `fev`) measured for 606 children and adolescents.

```
load("FevChildren.RData")
head(FevChildren)
str(FevChildren)
```

The dataset contains data about the forced expiratory volume (variable `fev`), the height (in cm, variable `height`), the age (in years, variable `age`), smoking status (variable `smoking`, 1="Yes", 0="No"), sex (variable `gender`) and age group (variable `age.group`) measured on 606 children and adolescents. We are interested in fitting a regression that explains `fev` in terms of `height`, `gender` and `age.group`.

Before fitting the model, it is important to make sure that all the categorical variables that we would like to include into the model are stored as factor. Luckily, the two categorical variables `gender` and `age.group` are already stored as a `factor`.

> **Note**
>
> In order for a categorical variable with K categories to be included in the model, it should be rewritten as K dummy variables and then, one of them should be dropped. The category that is dropped is taken as a reference. This process is automatically carried out by R (if the categorical variables are stored as `factor`). By default, the reference level is chosen as the first level. Thus, the reference category for the variable `gender` will be `f` and the reference category for the variable `age.group` will be `6-9` and all the interpretations will be with respect to these two categories.
> It is possible to change the reference category by using the function `relevel()`. We just need to provide the categorical variable and the new reference level. For example, if we want to change the reference category for `gender` to `m` the code will be the following:
>
> ```
> FevChildren2 = FevChildren
> FevChildren2$gender <- relevel(FevChildren2$gender, ref = "m")
> ```
>
> **Note** in the code above, first we created a copy of the dataset `FevChildren` and we changed the reference category in the copy, not the original dataset.

The syntax to fit the model is not really different with respect to simple linear regression.

```
mult_reg <- lm(fev ~ height + gender + age.group, data = FevChildren)
summary(mult_reg)
```

The fitted model in this case is:

$$\hat{y} = -4.897 + 0.047 \cdot x_1 + 0.134 \cdot x_2 + 0.162 \cdot x_3 + 0.454 \cdot x_4$$

where

- x_1 = "height";
- x_2 = "gender: male";
- x_3 = "age group: 10-14";

- x_4 = "age group: 15-17";

Recall the interpretation of the coefficients for multiple linear regression:

- the intercept tells us the expected expiratory volume when height is equal to 0 cm, the gender is female and when the age group is 6-9. (Does the interpretation make sense?)
- When the variable height increase by 1 cm, the forced expiratory volume increases on average by 0.0472 units, holding all the other variables constant.
- the forced expiratory volume for males is expected to be 0.134 units higher with respect to females, holding all the other variables constant.
- the forced expiratory volume for the age group 10-14 is expected to be 0.162 units higher with respect to the age group 6-9, holding all the other variables constant.
- the forced expiratory volume for the age group 15-17 is expected to be 0.454 units higher with respect to the age group 6-9, holding all the other variables constant.

75% of the variability of the forced expiratory volume is explained by the height, gender and age group.

As for the simple linear regression, it is possible to get the coefficients, the residuals and the fitted values by typing `mult_reg$coefficients`, `mult_reg$residuals` and `mult_reg$fitted.values`, respectively.

Of course, it is possible to make predictions with this model. For example, let us predict the forced expiratory volume for a female child that is 12 years old and 150 cm tall, i.e. $x_1^* = 150$, $x_2^* = 0$, $x_3^* = 1$ and $x_4^* = 0$:

$$\hat{y} = -4.897 + 0.047x_1^* + 0.134x_2^* + 0.162x_3^* + 0.454x_4^* = -4.897 + 0.047 \cdot 150 + 0.134 \cdot 0 + 0.162 \cdot 1 + 0.454 \cdot 0 = 2.315.$$

The function `predict()` can still be used. Remember that the new dataset should have the same names as the independent variables that we used to fit the model and the new values for the categorical should be stored as factors.

```
new_x <- data.frame(height = 150, gender = factor("f"), age.group = factor("10-14"))
predict(mult_reg, newdata = new_x)
```

Of course, we can have multiple predictions at a time. For instance, let us predict the forced expiratory volume for a female child that is 12 years old and 150 cm tall, as before, but also for a eight years old female who is 170 cm tall:

```
new_x <- data.frame(height = c(150, 170), gender = factor(c("f", "f")),
                    age.group = factor(c("10-14", "6-9")))
predict(mult_reg, newdata = new_x)
```

EXERCISE

It is also possible to apply a non-linear transformation to the multiple linear regression. Fit the the model with same independent variables, but transform the response variable `fev` by using the log-transformation. Interpret the coefficients.

EXERCISE

- Load the built-in dataset `Salaries` from the library `carData`. This dataset consists of the academic salary for Assistant Professors, Associate Professors and Professors in a college in the US.
- Check if the categorical variables are stored as factor.
- Fit a multiple linear regression model with salary as dependent variable and considering all the other variables as independent variables.
- Interpret the coefficients. Is there a difference between male and female professors?

Polynomial regression

Let us return to the situation considered in simple linear regression, that is, we want to explain a response variable y in terms of **one** explanatory variable x. When the relationship between the response variable and the independent variable is not linear, we have seen that we can apply some non-linear transformation to make the relationship linear. Another very common technique is to use a polynomial regression. This means that polynomial terms are added to the models.

Let us consider an example. Let us use again the built-in dataset `mtcars` to study the relationship between **mpg** and **hp**.

```
plot(mtcars$hp, mtcars$mpg, xlab = "hp", ylab = "mpg")
```

Based on the scatter plot, it looks like the relationship between these two variables is not linear. Before, we have considered to apply the log transformation to both dependent and independent variable. Now, instead, we consider to add polynomial terms to the model.

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \cdot x + \hat{\beta}_2 \cdot x^2$$

This is basically a multiple linear regression with two explanatory variables are x and x^2. This means that we can still use the function `lm()`!

In principle, we are tempted to do add the variable `x^2` in this way in the model. Actually, we need to pay attention on that. In fact the operator `^` has another meaning inside the `lm()` function. We do not go into the specific details of that, but take into account that, inside the

function `lm()`, `x^2` will not create what we want! For this reason, the function `I()` is used. This function tells us that we actually would like to have a quadratic terms for x.

```
pol_reg <- lm(mpg ~ hp + I(hp^2), data = mtcars)
summary(pol_reg)
```

The problem of adding polynomial terms in the regression is that the parameter interpretation is not that easy. In fact, the coefficient $\hat{\beta}_1$ can no longer be interpreted as the average change of y when x increase by 1 unit, holding the other variable constant. This is because the other variable is x^2 and if x increases, so does x^2. For this reason, the polynomial regression is mostly used for prediction purposes.

The prediction for a value of x_0 is the following:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2$$

The last thing that remains to do is to plot the estimated function. As it is a non-linear function, `abline()` cannot be used. Therefore, the idea, is the same that we used when we talked about the log-transformation: we can create a grid of values for the variable x and then we use the model to predict the value of y at all the points of the grid. Then, we connect these points by using the function `lines()`.

```
x.grid <- seq(from = min(mtcars$hp), to = max(mtcars$hp), length=1000)
x.new = data.frame(hp = x.grid)
yhat <- predict(pol_reg, newdata = x.new)
plot(mtcars$hp, mtcars$mpg, xlab = "hp", ylab = "mpg")
lines(x.grid, yhat, col = "orange")
```

We can see that the polynomial regression fits quite well the data.

Of course, we can add extra polynomial terms. For example, let us add x^3 to our model. The process is the same as before.

```
pol_reg2 <- lm(mpg ~ hp + I(hp^2) + I(hp^3), data = mtcars)
summary(pol_reg2)

yhat2 <- predict(pol_reg2, newdata = x.new)
plot(mtcars$hp, mtcars$mpg, xlab = "hp", ylab = "mpg")
lines(x.grid, yhat, col = "orange")
lines(x.grid, yhat2, col = "blue")
```

EXERCISE

- The dataset `marketing.csv` contains sales and the advertising budget of a product in 10000 US dollars. We are interesting in studying how the advertsing budget affects the sales.
- Present a scatterplot of these two variables. Is the relationship linear?
- Fit a simple linear regression model and add the estimated line to the scatterplot.
- Fit a polynomial regression model including x^2 and add the estimated curve to the scatterplot. What do you think now?
- Try to add an extra polynomial term x^3. Which one is better visually?

3 Appendix

List of common arguments for the function `plot()`.

- col = colors for lines and points. It can be specified by using a number or typing the name of the color, usually in lowercase letters. Don't forget to use the quotes when writing the color!
- main = main title of the plot, string, within quotes.
- xlab = title on the x-axis, string, whiting quotes.
- ylab = title on the y-axis, string, whiting quotes.
- xlim = range of the x-axis. For example, xlim = c(0, 10) means that the lowest plotted value will be 0 and the highest value will be 10 (on the x-axis).
- ylim = range the y-axis. For example, ylim = c(-5, 5) means that the lowest plotted value will be -5 and the highest value will be 5 (on the y-axis).
- lwd = the thickness of lines (or points) in a line chart (scatterplots), specified by a number.
- lty = type of line (straight, dashed, dotted, etc.), specified by a number.
- pch = type of dot (round, square, etc), specified by a number.