Exercise 11

Zoltan Kekecs

15 november 2018

Exercise 11 - The basics of linear regression

This exercise is related to learning the basic logic behind making predictions with linear regression models, and to quantifying the effectiveness of our predictions.

The latest version of this document and the code the document refers to can be found in the GitHub repository of the class at: https://github.com/kekecsz/PSYP13_Data_analysis_class-2018

Loading packages

You will need to load the following packages for this exercise:

```r
library(psych)  # for describe
library(dplyr)  # for data management
library(gsheet)  # to read data from google sheets
library(ggplot2)  # for ggplot
```

Custom functions

The following custom functions will come in handy during the visualization of the error of the predictions of the regressions line. This is not essential for the analysis and you don't need to learn how this custom function works if you don't want to.

```r
error_plotter <- function(mod, col = "black", x_var = NULL) {
    mod_vars = as.character(mod$call[2])
    data = eval(parse(text = as.character(mod$call[3])))
    y = substr(mod_vars, 1, as.numeric(gregexpr(pattern = "~",
        mod_vars)) - 2)
    x = substr(mod_vars, as.numeric(gregexpr(pattern = "~", mod_vars)) +
        2, nchar(mod_vars))

    data$pred = predict(mod)

    if (x == "1" & is.null(x_var)) {
        x = "response_ID"
        data$response_ID = 1:nrow(data)
    } else if (x == "1") {
        x = x_var
    }

    plot(data[, y] ~ data[, x], ylab = y, xlab = x)
    abline(mod)

    for (i in 1:nrow(data)) {
        clip(min(data[, x]), max(data[, x]), min(data[i, c(y,
            "pred")]), max(data[i, c(y, "pred")]))
        abline(v = data[i, x], lty = 2, col = col)
    }
}
```

Data management and descriptive statistics

Collect some data

Lets say we are a complany that sells shows, and we would like to be able to tell people's shoe size just by knowing their height. We collect some data using this simple survey:

https://goo.gl/forms/nNXWwCPTbdHxGrBO2

You can load the data we just collected with the following code

```
# data from
# https://github.com/kekecsz/PSYP13_Data_analysis_class-2018/blob/master/Shoe%20and%20height%20data%20P
mydata = read.csv("https://bit.ly/2Q0qZgP")
```

Check the dataset for irregularities

You should always check the dataset for coding errors or data that does not make sense.

View data in the data viewer tool

```
View(mydata)
```

or display simple descriptive statistics and plots.

You can use the describe() function from the psych package for continuous data, and the table() function for categorical data.

```
describe(mydata$height)
hist(mydata$height, breaks = 20)

describe(mydata$shoe_size)
hist(mydata$shoe_size, breaks = 10)

# scatterplot
plot(shoe_size ~ height, data = mydata)
```

There are several irregularties in the variable 'gender'. We need to handle these before going forward.

```
table(mydata$gender)
```

```
##
##    etc.  female female     male
##       1      33       1       9
```

You should clean your data before doing the analyses. Make sure to keep a record of the corrections. You should keep the raw datafile when possible, and make changes in the code, so the corrections and omissions are trackable in your code.

```
mydata_cleaned = mydata  # create a copy of the data where which will be cleaned
mydata_cleaned[mydata_cleaned[, "height"] == "168cm", "height"] = 168
mydata_cleaned[, "height"] = as.numeric(as.character(mydata_cleaned[,
    "height"]))
mydata_cleaned[mydata_cleaned[, "height"] == 64, "height"] = 164
```

```
mydata_cleaned = mydata_cleaned[-which(mydata_cleaned[, "gender"] ==
    "etc."), ]  # exclude invalid value
# this code might be required to correct the dataframe as
# well if you have a cell where 'female' has a space after
# it.  mydata_cleaned[,'gender'][mydata_cleaned[,'gender'] ==
# 'female '] = 'female' # unify different coding variations
```

Check whether the data cleaning had the desired effect

```
table(mydata_cleaned$gender)
```

```
##
##    etc.  female female     male
##       0      33       1       9
```

Prediction with linear regressioin

how to set up and interpret simple regression

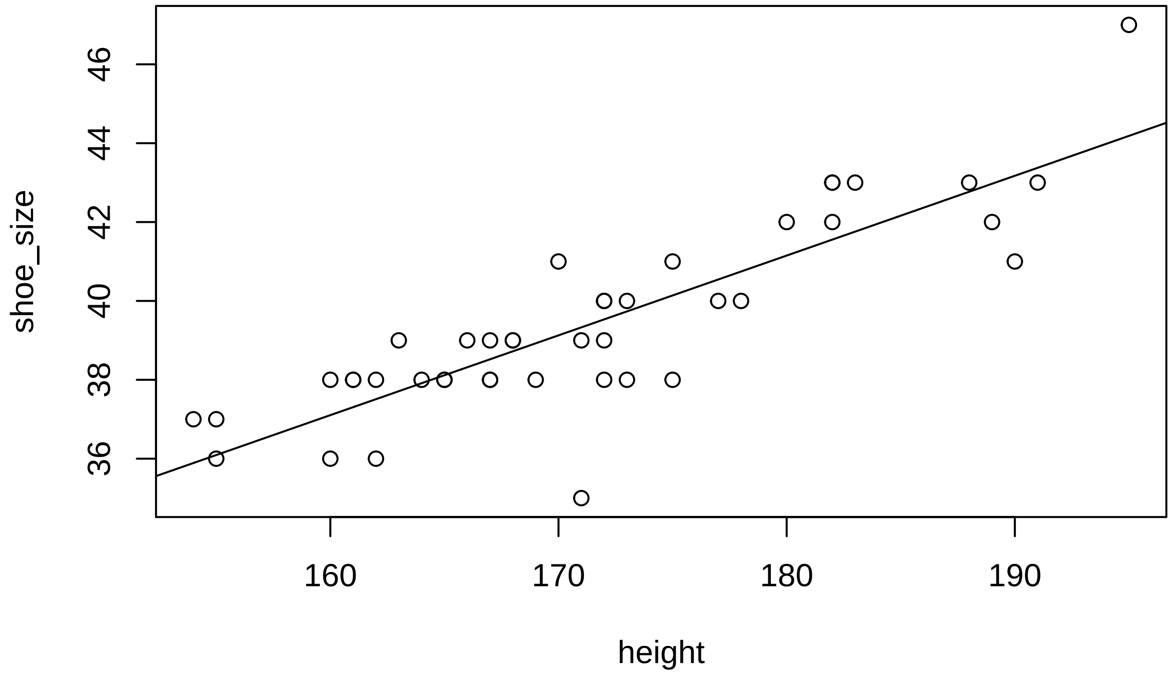
Regression is all about predicting an outcome by knowing the value of predictor variables that are associated with the outcome.

You can set up a simple regression model by using the code below. In the code we first specify an object where the results will be saved (mod1), then we specify the regression formula. In the formula, we start with specifying the outcome variable, what we are interested in predicting (shoe_size), then, after a ~ we specify the predictors. In simple regresison we only have one predictor (here this is height), but we could add more predictor variables, spearated with a + sign. In the end of the code we specify the data file where these variables originate from. Remember to use the cleaned dataset (mydata_cleaned).

```
mod1 <- lm(shoe_size ~ height, data = mydata_cleaned)
```

In simple regression, we identifies the underlying pattern of the data, by fitting a single straight line that is closest to all data points.

```
plot(shoe_size ~ height, data = mydata_cleaned)
abline(mod1)
```



Regression provides a matematical equation (called the regression equation) with which you can predict the outcome by knowing the value of the predictors.

The regression equation is formalized as: Y = b0 + b1*X1, where Y is the predicted value of the outcome, b0 is the intercept, b1 is the regression coefficient for predictor 1, and X1 is the value of predictor 1.

```
mod1
```

```
##
## Call:
## lm(formula = shoe_size ~ height, data = mydata_cleaned)
##
## Coefficients:
## (Intercept)       height
##      4.7307       0.2023
```

This means that the regression equiation for predicting shoe size is:

shoe size = 4.73 + 0.2 * height

that is, for a person who is 170 cm tall, the predicted shoe size is calcluated this way:

4.73 + 0.2 * 170 = 38.73

You don't have to do the calculations by hand, you can get the predicted values by using the predict() function.

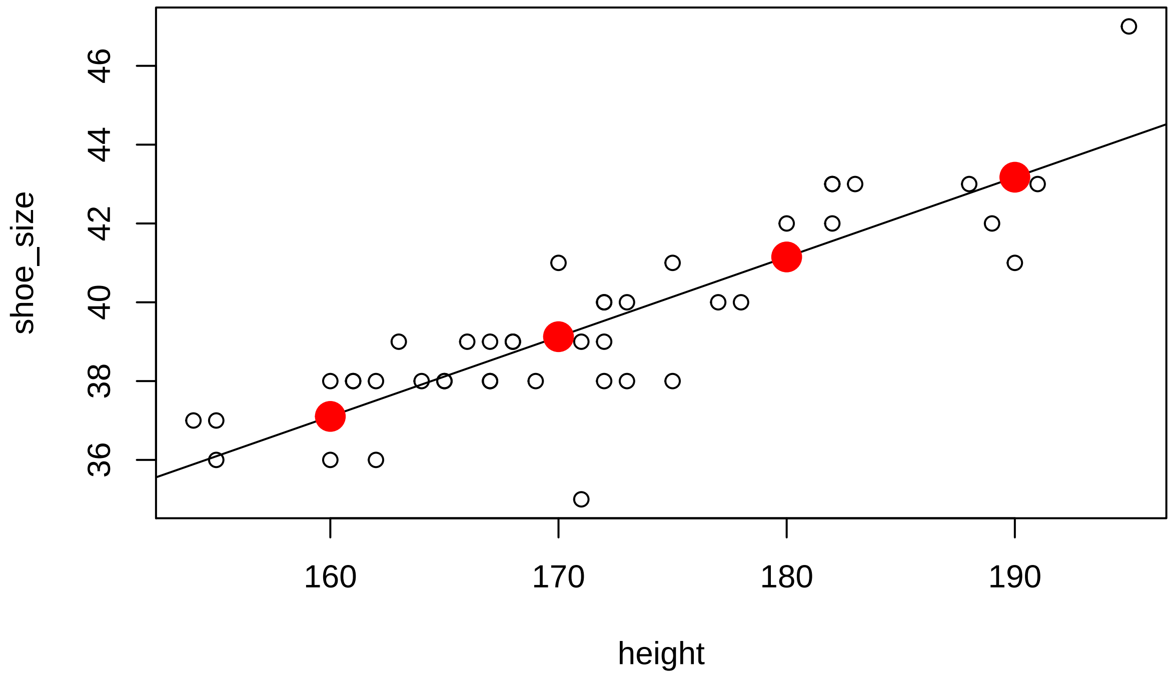
The predictors have to have the same variable name as in the regression formula, and they need to be in a dataframe

```
height = c(150, 160, 170, 180, 190)
height_df = as.data.frame(height)
predictions = predict(mod1, newdata = height_df)
height_df_with_predicted = cbind(height_df, predictions)
height_df_with_predicted
```

```
##   height predictions
## 1    150    35.08026
## 2    160    37.10357
## 3    170    39.12687
## 4    180    41.15018
## 5    190    43.17348
```

Predicted values all fall on the regression line

```
plot(shoe_size ~ height, data = mydata_cleaned)
abline(mod1)
points(height_df_with_predicted, col = "red", pch = 19, cex = 2)
```



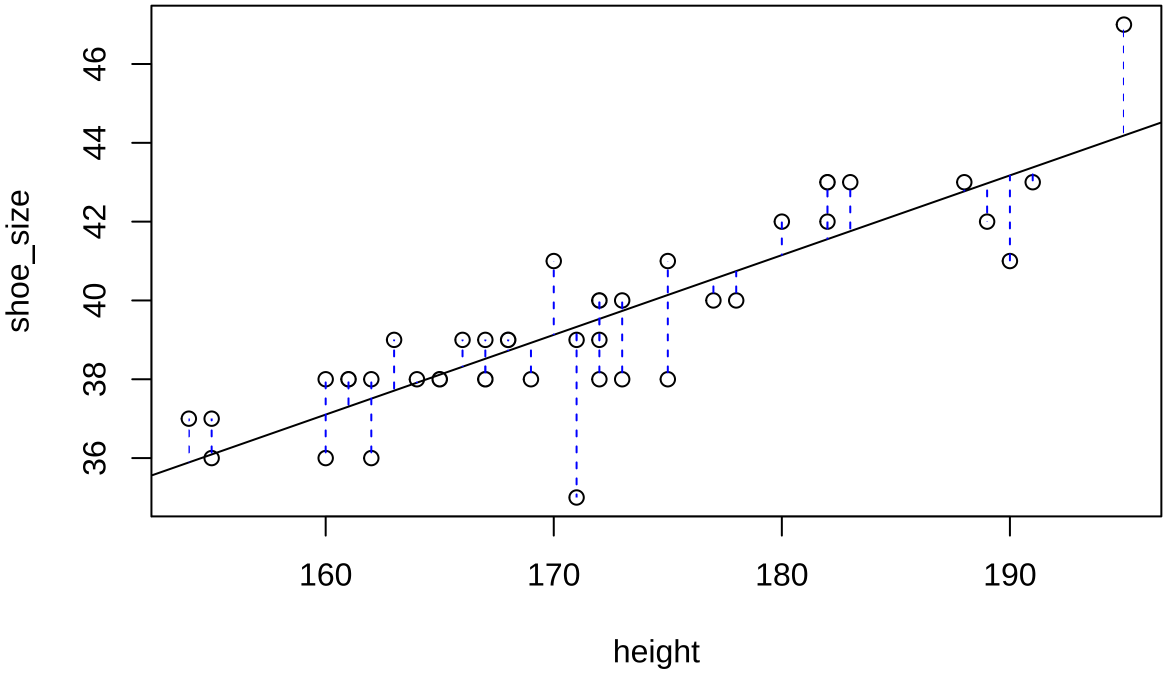
How good is my model

How to measure prediction efficiency?

You can measure how effective your model is by measureing the difference between the actual outcome values and the predicted values. We call this residual error in regression.

The residual error for each observed shoe size can be seen on this plot (this will only work if you ran the code in the top of the script containing the error_plotter() custom function). This code is only for visualization purposes.

```
error_plotter(mod1, col = "blue")
```



You can simply add up all the residual error (the length of these lines), and get a good measure of the overall efficiency of your model. This is called the residual absolute difference (RAD). However, this value is rarely used. More common is the residual sum of squared differences (RSS).

```
RAD = sum(abs(mydata_cleaned$shoe_size - predict(mod1)))
RAD
```

```
## [1] 40.60293
```

```
RSS = sum((mydata_cleaned$shoe_size - predict(mod1))^2)
RSS
```

```
## [1] 66.6071
```

Is the predictor useful?

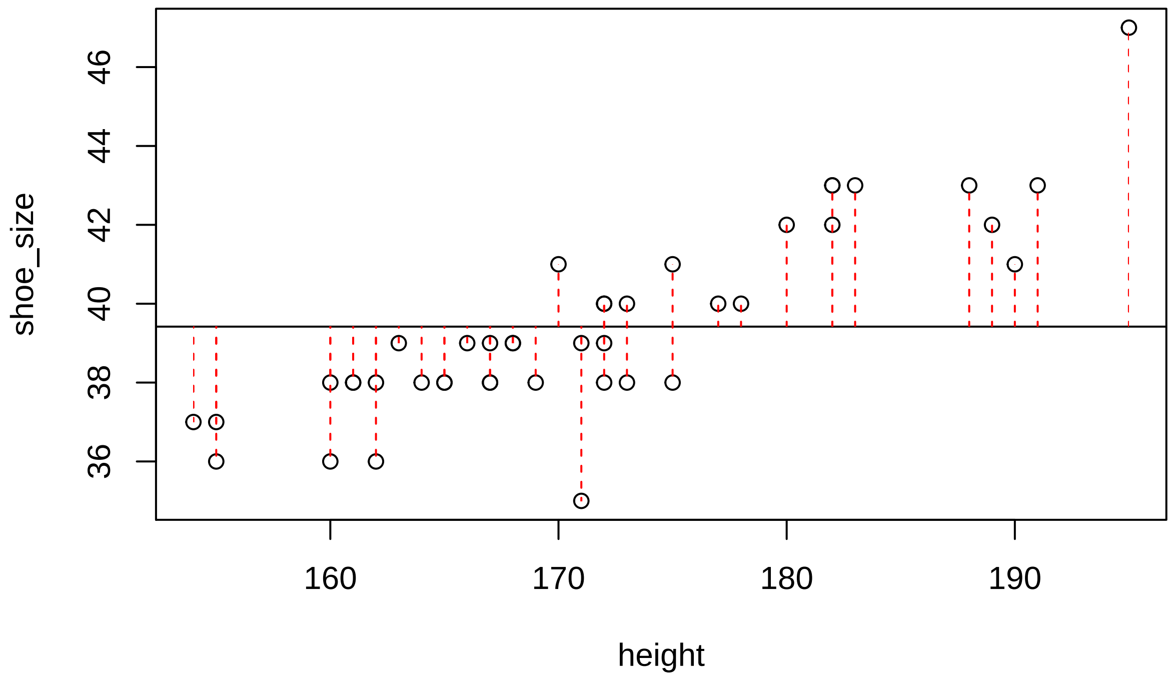
To establish how much benefit did we get by taking into account the predictor, we can compare the residual error when using out best guess (mean) without taking into account the predictor, with the residual error when the predictor is taken into account.

Below you can find regression model where we only use the mean of the outcome variable to predict the outcome value.

We can calculate the sum of squared differences the same way as before, but for the model without any predictors, we call this the total sum of squared differences (TSS).

```
mod_mean <- lm(shoe_size ~ 1, data = mydata_cleaned)

error_plotter(mod_mean, col = "red", x_var = "height")  # visualize error
```



```
TSS = sum((mydata_cleaned$shoe_size - predict(mod_mean))^2)
TSS
```

```
## [1] 244.4651
```

The total amount of information gained about the variability of the outcome is shown by the R squared statistic (R^2).

```
R2 = 1 - (RSS/TSS)
R2
```

```
## [1] 0.7275395
```

This means that by using the regression model, we are able to explain 72.75% of the variability in the outcome.

R^2 = 1 means all variablility of the outcome is perfectly predicted by the predictor(s)

R^2 = 0 means no variablility of the outcome is predicted by the predictor(s)

Is the model with the predictor significantly better than a model without the predictor?

You can do an anova to find out, comparing the amount of variance explained by the two models.

```
anova(mod_mean, mod1)
```

```
## Analysis of Variance Table
```

```
##
## Model 1: shoe_size ~ 1
## Model 2: shoe_size ~ height
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1     42 244.465
## 2     41  66.607  1    177.86 109.48 3.819e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Or, you can get all this information and more from the model summary

```r
summary(mod1)
```

```
##
## Call:
## lm(formula = shoe_size ~ height, data = mydata_cleaned)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3292 -0.5374  0.2661  0.7720  2.8149
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.73067    3.32090   1.425    0.162
## height       0.20233    0.01934  10.463 3.82e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.275 on 41 degrees of freedom
## Multiple R-squared:  0.7275, Adjusted R-squared:  0.7209
## F-statistic: 109.5 on 1 and 41 DF,  p-value: 3.819e-13
```

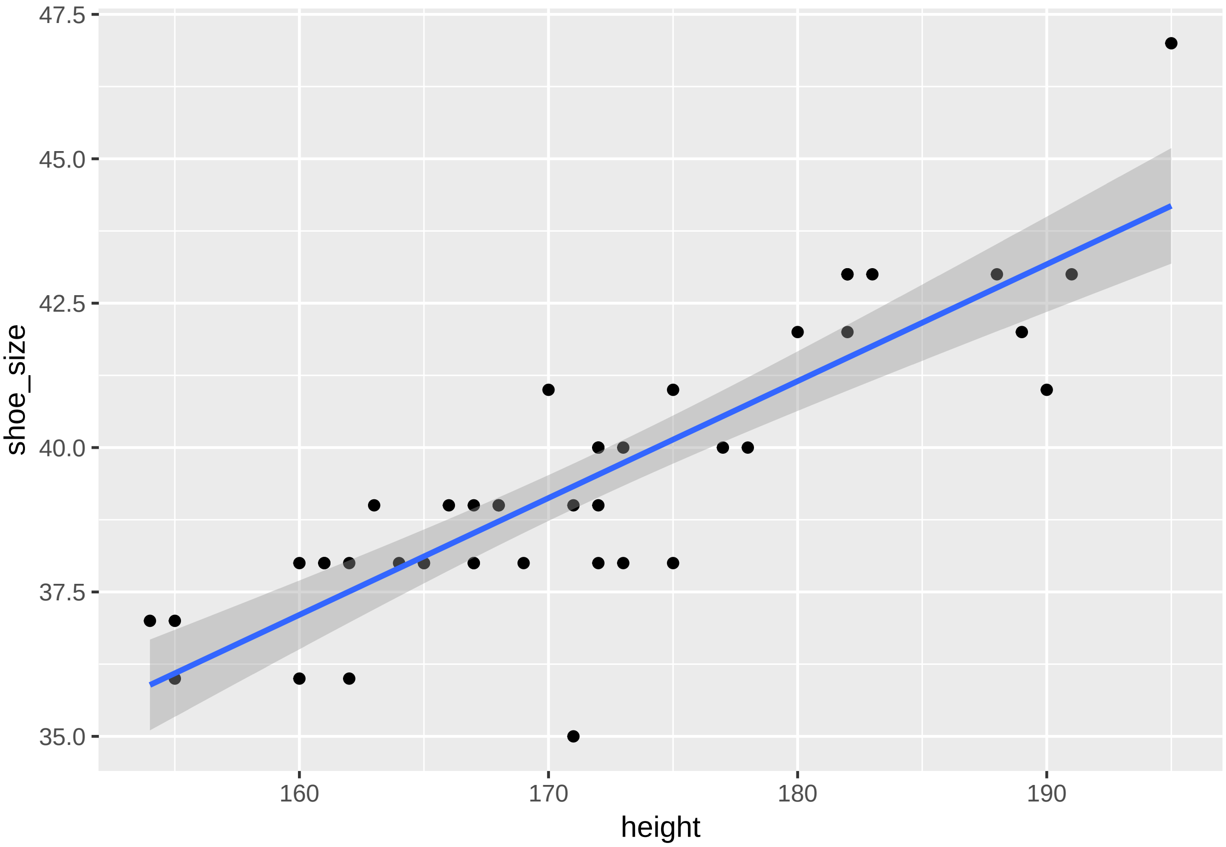
The confidence interval of the regression coefficient is given by the confint() function

```r
confint(mod1)
```

```
##                  2.5 %     97.5 %
## (Intercept) -1.9760167 11.4373534
## height       0.1632784  0.2413828
```

You can also plot the confidence interval of the prediction

```r
ggplot(mydata_cleaned, aes(x = height, y = shoe_size)) + geom_point() +
    geom_smooth(method = "lm", formula = y ~ x)
```


47.5
45.0
42.5
40.0
37.5
35.0
shoe_size
160
170
180
190
height