Class 5. Introduction to econometric analysis in R®

Advanced Econometrics I

Lectures: Elena Kotyrlo
Classes: Yuri Trifonov, Elena Semerikova

Fall 2022

Problem 1

This problem requires you to use data.xlsx database. The data represents prices for two-bedroom apartments in Moscow, year 2005 (2040 observations). The table with description of all variables is provided below:

Variables	Description
n	Id of the apartment
price	Price in $1000
totsp	Total space, sqm
livesp	Living space, sqm
kitsp	Kitchen space, sqm
dist	Distance from the center of Moscow
metrdist	Distance to the closest metro station
walk	From metro: on foot (1) or on public transport (0)
brick	Type of house: brick / cast reinforced concrete house (1) or other (0)
floor	Floor: first / top (0) or other (1)
code	District:
	North, orange (1)
	North, gray (2)
	North-west, green (3)
	North-west, purple (4)
	South-east, light-green (5)
	South-east, purple (6)
	East, yellow (7)
	East, blue (8)

(a) Import the data file to R and load the required libraries.

(b) Have a look at the histogram of *price*, compare the distribution with normal.

(c) Generate a new variable *price_msq*, representing the price of apartments per square meter. How can it be useful?

(d) Compute descriptive statistics, make a brief comment.

(e) Provide the graphical analysis of the relationship of *price_msq* with continuous and discrete variables.

(f) Compute correlations of *price_msq* with other variables.

(g) Compare the mean of *price_msq* by floor.

(h) Estimate different regression models via the OLS. Start with the model: $price\_msq_i = \alpha + \beta_1 \times dist_i + \varepsilon_i$.

(i) Discuss the results and give economic interpretation.

(j) Compare the effect of distance on price for two types of housing: those in walking distance with the rest.

Problem 2* (Using matrix notation)

This problem is designed for those who want to learn how to program OLS estimator by hand. Consider the following multiple linear regression model:

$$y_i = \alpha + \beta_1 \times x_1 + \beta_2 \times x_2 + \varepsilon_i, \;\; \varepsilon_i \sim N(0, 1).$$

Step 1. Simulate the data according to the above-mentioned model. Consider the following distributions of independent variables: $x_1 \sim N(2, 1)$, $x_2 \sim t(4)$. Let the true values of parameters be equal to $\alpha = 0.8$, $\beta_1 = 1.5$, $\beta_2 = 2.1$. The number of observations is equal to $n = 1000$.
Step 2. Create a data frame with the name *'df'*, containing dependent and independent variables.
Step 3. Write your own function for obtaining OLS estimates for the discussed model using matrix notation. Call this function *'ols_matr'*.
Step 4. Estimate the simulated model via the developed function and ensure that the the obtained estimates are very similar to the true values of parameters α, β_1, β_2.

Problem 3* (Using numerical optimization)

This problem is designed for those who want to learn how to program OLS estimator by hand. Consider the following multiple linear regression model:

$$y_i = \alpha + \beta_1 \times x_1 + \beta_2 \times x_2 + \varepsilon_i, \;\; \varepsilon_i \sim N(0, 1).$$

Repeat **Steps 1-2** from the previous problem.
Step 3. Write your own function for calculating RSS (without matrix notation). Call this function *'rss_func'*.
Step 4. Obtain the values of estimates through the minimization of the RSS using the implemented in R optimization function **optim()**. This process is called 'numerical optimization' since you actually taking the derivative not through the analytical expressions, but via numerical methods.
Step 5. Ensure that the the obtained estimates are very similar to the true values of parameters α, β_1, β_2.

Some hints for Problems 2-3

- To generate pseudo-random samples from normal and Student t distributions use functions rnorm() and rt() correspondingly. Remember that R always gives you hints what each value in the input of the function means .
- To construct a data frame containing your variables use the function data.frame(). For example: df = data.frame(y, x_1, x_2) will construct a data frame for you. Notice that after you create it, you can reach the variables inside it as follows: dfx_1, dfy.
- To construct X matrix with a first column of ones use functions cbind() and rep(1, n). First function allows you to combine columns of data into one matrix, while function rep(1,n) allows to create a column of n ones. *Example: cbind(rep(1,n), x_1, x_2).*
- To multiply matrices according to the rules of linear algebra in R, you need to use the operator **%*%**. For example: A%*%B, where A and B are matrices.
- Function t(A) allows you to compute the transpose of a matrix. Function solve(A) computes the inverse of a matrix. Where A is a matrix.
- Use $x_0 = (0, 0, 0)$ as a vector of starting points for optimization in Problem 3.

- Examples of how to correctly specify the optimizer and RSS function are provided below:

```
rss_func <- function(par, # input arguments are vector of
                     df)  # estimated parameters and your data
{
    alpha <- par[1]    # like this you specify each specific
    beta_1 <- par[2]   # parameter in the vector, that you are
    beta_2 <- par[3]   # going to estimate

    ... # here the function continues
}

x0 <- c(0,0,0)                       # vector of initial points
optim(par = x0,                      # initial point
      method = "BFGS",               # optimization algorithm
      fn = rss_func,                 # rss function
      df = df)                       # we should tell to the optimizer
                                     # which data we should use
optim$par                            # return values of estimates
```