First steps with R

Contents

1 Getting started

1.1 Reset

If you want to start over or if you want to change the dataset you're working with there is a simple command that tells R to erase from the memory all the data, variables, regression models, etc. Just type

```
rm(list=ls())
```

1.2 Launching R and working directory

Before doing anything we need to **set the working directory**, i.e., tell R in which directory the data is. You can either go to the menu or set the command

```
setwd("PATH")
```

where `PATH` is the full path of the directory. For instance, with windows, if the data is in the folder `C:/Documents and Settings/Data` we type

```
setwd("C:/Documents and Settings/Data")
```

For a Mac, if the data is in the folder `/Users/john/data/` then we type

```
setwd("/Users/john/data/")
```

You can check that R got the working directory right by typing the command

```
getwd()
```

1.3 Loading the data

Data can come in several formats:

- `RData`: A datafile format that is adapted to R
- `xlsx` or `xls`: Excel spreadsheets. To read such files on R is a little bit more involved. We need to install some additional packages to R in order to read those files. Reading such files is not covered here.

- `csv` or `txt`: simple files (readable by almost any software). Usually in a `txt` file the columns are separated by a space or a tabulation. As for `csv`, it stands for "comma separated values and thus, as it names indicates, the columns are separated by a comma.

  Those files are easily readable by R. If we don't have such files but have instead Excel files then the best is to

  1. Open the Excel file with Exel
  2. Choose the menu "**File/Save as**" and select the format "**Comma Separated Values (.csv)**"

- files from other statistical softwares. Reading such files is not covered here.

1.3.1 Loading `RData` files

Just type

```
load("wage1.RData")
```

Here the data file we load is `wage1.RData`. It is important to write the full name of the file with its extension.

1.3.2 What does an "RData" file contain?

To get to now what is in the RData file you type

```
ls()
```

You get this:

```
"data" "desc" "self"
```

`data` is the data itself. If you type

```
data
```

then R shows you all the data. `desc` is the description of the variables. It is the same as what's in the file `WAGE1_description.txt` that you use for Excel. Finally, `self` is simply the name of the data set for R. You type

```
self
```

Then you get

```
"wage1.dta"
```

If you type

```
desc
```

then you get a description of the data (the name of the variables and their explanations):

```
> desc
   variable                            label
1      wage          average hourly earnings
2      educ                years of education
3     exper       years potential experience
4    tenure      years with current employer
5  nonwhite                    =1 if nonwhite
6    female                      =1 if female
7   married                     =1 if married
8    numdep              number of dependents
9      smsa                 =1 if live in SMSA
10 northcen =1 if live in north central U.S
11    south   =1 if live in southern region
12     west    =1 if live in western region
13 construc  =1 if work in construc. indus.
14  ndurman  =1 if in nondur. manuf. indus.
15 trcommpu  =1 if in trans, commun, pub ut
16    trade    =1 if in wholesale or retail
17 services        =1 if in services indus.
18 profserv     =1 if in prof. serv. indus.
19  profocc    =1 if in profess. occupation
20  clerocc    =1 if in clerical occupation
21  servocc     =1 if in service occupation
22    lwage                       log(wage)
```

23	expersq	exper^2
24	tenursq	tenure^2

1.3.3 Reading txt or csv files

For such files we need to distinguish between two cases, depending on whether the columns have names, called **headers** .

1.3.3.1 Datafiles with headers

It often occurs that a dataset comes with two files: one file containing only the data and another file (usually a text file) that indicates the names of the columns.

We start with the easiest case, when the columns have names. Let `mydata.txt` be the name of the file. In this case to load the data we type this command:

```
data <- read.table(file = "data.txt", header = TRUE)
```

Here the word `data` at the beginning of the command is the name we want to give to the dataset. It does not need to be the same name as the file. For instance, we could write instead `thedatasetwewilluse <- read.table(file = "data.txt", header = TRUE)` and thus the name of the dataset in R will be `thedatasetwewilluse`.

If the data set is a `csv` file, and assume that the name of the file is `anotherdata.csv`, then we type the following command

```
theotherdata <- read.csv(file = "anotherdata.csv", header = TRUE)
```

where `theotherdata` is again the name we decided to give to that dataset.

1.3.3.2 Datafiles without headers

In this case we should have two files:

- the dataset
- a text file with the names of the columns.

If the data is a text file, and let `mydata.txt` be the name of the file, then we type

```
data <- read.table(file = "data.txt")
```

Similarly, if the datafile is a `csv` file (with name `anotherdata.csv`, then we type the following command

```
theotherdata <- read.csv(file = "anotherdata.csv")
```

Again, for these two cases `data` and `theotherdata` are the names we gave to the datasets.

Unlike the case when the datafiles have headers there is an additional step: give names to the columns. Suppose that our data set has 4 columns, and that the names of those columns should be (in that order)

myFIRSTcolumn mySECONDcolumn myTHIRDcolumn myFOURTHcolumn

Then we type the following command in R:

```
colnames(THE-NAME-OF-THE-DATA) <- C( ‘‘myFIRSTcolumn’’,’’mySECONDcolumn’’,
                ‘‘myTHIRDcolumn’’, ‘‘myFOURTHcolumn’’)
```

where `THE-NAME-OF-THE-DATA` is the name you chose when loading the dataset (e.g., in the above examples those were `data` and `theotherdata`).

1.4 Declaring the variable names

R does not interpret automatically the variable names. This is so for all the cases reviewed in the previous sections:

- you loaded an `RData` file;
- you loaded a `txt` or `csv` file with headers;
- you loaded a `txt` or `csv` file without headers but then assigned names to the columns.

R knows that columns have names, but that does not allow you to use those names directly. For instance, if you loaded the datafile `wage1`, which has a variable called `educ`, then R will not be able to understand the command

```
mean(educ)
```

(which gives the average of the column `educ`). To do this we need to tell R that we will use the names of the columns as the names of the variables, too.

To do this you type

```
attach(data)
```

where `data` is the name of the data when you type `ls()`. So, if you called your data when loading it, say, `theotherdata`, then you would type `attach(theotherdata)`.

Once you have done that you can invoke the variables by simply typing their names. For instance, if you type

```
mean(wage)
```

then you get the average wage, 5.896103.

If you want to see how the data looks like, type the following command

```
str(data)
```

You should see this

```
'data.frame':        526 obs. of  24 variables:
 $ wage    : num  3.1 3.24 3 6 5.3 ...
 $ educ    : int  11 12 11 8 12 16 18 12 12 17 ...
 $ exper   : int  2 22 2 44 7 9 15 5 26 22 ...
 $ tenure  : int  0 2 0 28 2 8 7 3 4 21 ...
 $ nonwhite: int  0 0 0 0 0 0 0 0 0 0 ...
 $ female  : int  1 1 0 0 0 0 0 1 1 0 ...
 $ married : int  0 1 0 1 1 1 0 0 0 1 ...
 $ numdep  : int  2 3 2 0 1 0 0 0 2 0 ...
 $ smsa    : int  1 1 0 1 0 1 1 1 1 1 ...
 $ northcen: int  0 0 0 0 0 0 0 0 0 0 ...
 $ south   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ west    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ construc: int  0 0 0 0 0 0 0 0 0 0 ...
 $ ndurman : int  0 0 0 0 0 0 0 0 0 0 ...
 $ trcommpu: int  0 0 0 0 0 0 0 0 0 0 ...
 $ trade   : int  0 0 1 0 0 0 1 0 1 0 ...
```

```
$ services: int  0 1 0 0 0 0 0 0 0 0 ...
$ profserv: int  0 0 0 0 0 1 0 0 0 0 ...
$ profocc : int  0 0 0 0 0 1 1 1 1 1 ...
$ clerocc : int  0 0 0 1 0 0 0 0 0 0 ...
$ servocc : int  0 1 0 0 0 0 0 0 0 0 ...
$ lwage   : num  1.13 1.18 1.1 1.79 1.67 ...
$ expersq : int  4 484 4 1936 49 81 225 25 676 484 ...
$ tenursq : int  0 4 0 784 4 64 49 9 16 441 ...
- attr(*, "datalabel")= chr ""
- attr(*, "time.stamp")= chr "25 Jun 2011 23:03"
- attr(*, "formats")= chr  "%8.2g" "%8.0g" "%8.0g" "%8.0g" ...
- attr(*, "types")= int  254 251 251 251 251 251 251 251 251 251 ...
- attr(*, "val.labels")= chr  "" "" "" "" ...
- attr(*, "var.labels")= chr  "average hourly earnings" "years of education" "years pot
- attr(*, "version")= int 10
```

If you type

```
summary(data)
```

Then you have a summary of the data (for each variable the minimum, maximum, mean, etc.). With the data `xxx.txt` you should see this

```
      wage             educ           exper          tenure          nonwhite
 Min.   : 0.530   Min.   : 0.00   Min.   : 1.00   Min.   : 0.000   Min.   :0.0000
 1st Qu.: 3.330   1st Qu.:12.00   1st Qu.: 5.00   1st Qu.: 0.000   1st Qu.:0.0000
 Median : 4.650   Median :12.00   Median :13.50   Median : 2.000   Median :0.0000
 Mean   : 5.896   Mean   :12.56   Mean   :17.02   Mean   : 5.105   Mean   :0.1027
 3rd Qu.: 6.880   3rd Qu.:14.00   3rd Qu.:26.00   3rd Qu.: 7.000   3rd Qu.:0.0000
 Max.   :24.980   Max.   :18.00   Max.   :51.00   Max.   :44.000   Max.   :1.0000
     female          married           numdep           smsa           northcen
 Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :0.000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.000
```

```
 Median :0.0000   Median :1.0000   Median :1.000   Median :1.0000   Median :0.000
 Mean   :0.4791   Mean   :0.6084   Mean   :1.044   Mean   :0.7224   Mean   :0.251
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:0.750
 Max.   :1.0000   Max.   :1.0000   Max.   :6.000   Max.   :1.0000   Max.   :1.000
     south            west           construc          ndurman          trcommpu
 Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
 Median :0.0000   Median :0.0000   Median :0.00000   Median :0.0000   Median :0.00000
 Mean   :0.3555   Mean   :0.1692   Mean   :0.04563   Mean   :0.1141   Mean   :0.04373
 3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000   Max.   :1.00000
     trade           services         profserv          profocc           clerocc
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :0.2871   Mean   :0.1008   Mean   :0.2586   Mean   :0.3669   Mean   :0.1673
 3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
    servocc           lwage            expersq          tenursq
 Min.   :0.0000   Min.   :-0.6349   Min.   :   1.0   Min.   :   0.00
 1st Qu.:0.0000   1st Qu.: 1.2030   1st Qu.:  25.0   1st Qu.:   0.00
 Median :0.0000   Median : 1.5369   Median : 182.5   Median :   4.00
 Mean   :0.1407   Mean   : 1.6233   Mean   : 473.4   Mean   :  78.15
 3rd Qu.:0.0000   3rd Qu.: 1.9286   3rd Qu.: 676.0   3rd Qu.:  49.00
 Max.   :1.0000   Max.   : 3.2181   Max.   :2601.0   Max.   :1936.00
```

2 Ordinary Least Squares

2.1 Running the regression

When you run a regression you will "assign" the results of a regression to an object that will be stored in R's memory.

Suppose we want to do the following regression

$$Wage = \beta_0 + \beta_1 Education + \beta_2 Female + u$$

(to see how the wage depends on the years of education (captured by the variable `school`) and the gender (captured by binary variable **female**).

Let's call the "object" of the regression `model`. Here "`model`" is just a name. You can choose the name you want. You can call that regression "`myfirstregression`" or "`whateveryouwant`."

So the command we have to type to run the regression is

```
model <- lm(wage ~ educ + female)
```

If you want to see the results of the regression type

```
summary(model)
```

(If you called the regression "`myfirstregression`" then you should type "`summary(myfirstregression)`".)

When invoking the `summary(...)` commands you should see

```
Call:
lm(formula = wage ~ educ + female)

Residuals:
    Min      1Q  Median      3Q     Max
-5.9890 -1.8702 -0.6651  1.0447 15.4998

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.62282    0.67253   0.926    0.355
educ         0.50645    0.05039  10.051  < 2e-16 ***
female      -2.27336    0.27904  -8.147 2.76e-15 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
Residual standard error: 3.186 on 523 degrees of freedom
Multiple R-squared:  0.2588,        Adjusted R-squared:  0.256
F-statistic: 91.32 on 2 and 523 DF,  p-value: < 2.2e-16
```

2.2 Interpreting the results

The first thing that you see is to what the data refers:

```
Call:
lm(formula = wage ~ educ + female)
```

This simply says that it is a **L**inear regression **M**odel (hence the command `lm`), where `wage` is the dependent variable and `educ` and `female` are the explanatory variables. The intercept β_0 is included by default.

Next you see

```
Residuals:
    Min      1Q  Median      3Q     Max
-5.9890 -1.8702 -0.6651  1.0447 15.4998
```

This table displays information about the residuals (i.e., the realized values of the error terms).

The most interesting part comes after.

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.62282    0.67253   0.926    0.355
educ         0.50645    0.05039  10.051  < 2e-16 ***
female      -2.27336    0.27904  -8.147 2.76e-15 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

The first column is the name of the regressors (+ the intercept $\hat{\beta}_0$). The second column gives the values of the coefficients. So we have here

$$\hat{\beta}_0 = 0.62282$$
$$\hat{\beta}_1 = 0.50645$$
$$\hat{\beta}_2 = -2.27336$$

Here β_0 is the value of the intercept, i.e., when `educ=0` and `female=0`. This could be interpreted as the minimum wage (the least qualified job, when no education is required) for a male.

The third column gives the standard errors of the coefficients. Those are the ones you would use to compute the confidence intervals of the t-statistics.

$$s_{\hat{\beta}_0} = 0.67253$$
$$s_{\hat{\beta}_1} = 0.05039$$
$$s_{\hat{\beta}_2} = 0.27904$$

For instance, if you want to compute the 95% confidence interval for β_1 you would do

$$[0.50645 - 1.96 \times 0.05039\,,\, 0.50645 + 1.96 \times 0.05039]$$

The third columns gives you the t-statistic for the coefficients and the fourth column gives the p-value.

The most valuable column is the last one, that contains the symbol $*$. Below the table we have the interpretation of the stars symbols. If there is $***$ in the row corresponding to the coefficient β_h then we can reject with a near 100 confidence level the null hypothesis in the test $H_0 : \beta_h = 0$ vs. $H_1 : \beta_h \neq 0$. If there is only one star then it means we can reject at the 95% confidence level, but not at the 1% level (so the p-value would be between 0.01 and 0.05).

Finally we have the following information

```
Residual standard error: 3.186 on 523 degrees of freedom
Multiple R-squared:  0.2588,        Adjusted R-squared:  0.256
F-statistic: 91.32 on 2 and 523 DF,  p-value: < 2.2e-16
```

The `Residual standard error` is the SER of the regression, equal to 24.82. Here we are running a regression with multiple regressors so the relevant R^2 is the adjusted R^2, which is equal to 0.1967.

3 Extracting results and creating new variables

3.1 Extracting OLS estimates

You can store the variables that R calculates. The most interesting ones are the OLS estimates. You can obtain the coefficients by typing

```
coefficients
```

You can store them in a vector, just have to choose a name. For instance, let us call that vector `coeffs`. So we type

```
coeffs = coefficients(model)
```

To get the first coefficient, the intercept you just type

```
coeffs[1]
```

so we have

$$\widehat{\beta}_0 = 0.62282 = \text{coeffs}[1].$$

To get the second coefficient, the $\widehat{\beta}$ for `educ` you just type To get the first coefficient, the intercept you just type

```
coeffs[2]
```

And thus we have

$$\widehat{\beta}_1 = 0.50645 = \text{coeffs}[2]$$
$$\widehat{\beta}_2 = -2.27336 = \text{coeffs}[3].$$

3.2 Calculating fitted values and residuals

We can create the predicted wage with this command:

```
predictedwage = coeffs[1]+coeffs[2]*educ+coeffs[3]*female
```

He again, "`predictedwage`" is only a name. You can call it "`myfittedvalue`" or "`whatever`."

And the residuals (that we call "`res`"):

```
res=wage-predictedwage
```

where `res` is the name we give to the variable residuals. Theoretically, the sum of the residuals is 0. In practice, because of the rounding errors, it is not always the case, but it is always very very very very very very close to 0:

```
sum(res)
[1] 9.574119e-12
```

Calculating the fitted/predicted values and the residuals is a frequent exercise and R has commands for those variables.

The fitted values can be obtained with the command `fitted`. For instance, if we wanted the fitted values of a regression that we called `model` we would do this:

```
fitted(model)
```

But if we do this then R will display all the fitted values. It is better to store them in a new variable, that we can call, say, `myfittedvalues`. So we would do

```
myfittedvalues = fitted(model)
```

For the residuals, the command is `residuals`. Again, it is better to store them in a new variable that we call `myresiduals`:

```
myresiduals = residuals(model)
```

Note that once we have created these new variables R has them in memory. If we type

```
ls()
```

Then R will display

```
[1] "data"        "desc"        "m"           "myfitted"    "myresiduals" "self"
```

4 Partialling out: calculating OLS estimates for multi-linear regressions

4.1 The problem

How do we calculate with Excel the OLS estimates for a regression of the following type?

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + u \ .$$

We'll proceed using the dataset `wage1.xls`, with the following model,

$$\text{wage} = \beta_0 + \beta_1 \text{educ} + \beta_2 \text{exper} + u \ . \tag{1}$$

If we run this regression (with this data) with a statistical software we obtain the following estimates:

$$\widehat{\beta}_0 = -3.39054 \tag{2}$$

$$\widehat{\beta}_1 = 0.64427 \tag{3}$$

$$\widehat{\beta}_2 = 0.07010 \tag{4}$$

There is another method to obtain these estimates, known as **partialling out**. It is important to know this method because we will need it when studying the problem of *omitted variable bias*.

4.2 Calculating $\widehat{\beta}_1$

We will proceed in two steps. I explain here how to compute $\widehat{\beta}_1$ (and I let you reproduce the results for $\widehat{\beta}_2$. The coefficient β_1 in the model in equation (1) is the *net effect* of `educ` on `wage`. That is, it is the effect of `educ` on wage that eliminates the influence that `exper` can have on `educ` and `wage`.

4.2.1 Step 1: regressing educ on exper

The intuition for the method is the following: the data about the variable `educ` is a bit correlated with `exper`. So we will "purify" that data by eliminating from `educ` any influence on that variable made by `exper`. We we run the following regression:

$$\text{educ} = \gamma_0 + \gamma_1 \text{exper} + v \ . \tag{5}$$

Here I use γ (= gamma, the Greek letter for g) instead of β. That's just a notation to avoid confusion (if I use educ $= \beta_0 + \beta_1$exper $+ u$ then there is the risk that when we talk about β_1 we don't know if we refer to regression (1) or (5)). For the same reason in the model of Eq. (5) we denote the error term by v (instead of u).

If we run this regression you should find

$$\begin{aligned} \text{cov}(\texttt{educ},\texttt{exper}) &= -11.25726598 \\ \text{var}(\texttt{exper}) &= 184.2035162 \\ \text{average}(\texttt{educ}) &= 12.5627 \\ \text{average}(\texttt{exper}) &= 17.01711 \end{aligned}$$

and thus,

$$\begin{aligned} \widehat{\gamma}_0 &= 13.60271 \\ \widehat{\gamma}_1 &= -0.061113 \end{aligned}$$

4.2.2 Step 2: calculating the residuals

Now we look at the residuals $\widehat{v}$. The residuals are calculated with this formula:

$$\widehat{v} = \text{educ} - \widehat{\text{educ}} = \text{educ} - \widehat{\gamma}_0 - \widehat{\gamma}_1 \text{exper} \, . \tag{6}$$

The residuals $\widehat{v}$ are the part of `educ` that is <u>not</u> explained by `exper`. So the residuals can be interpreted somehow as our "purified" `educ`.

In Excel you should find this (for the first 4 lines):

`wage`	`educ`	`exper`	$\widehat{\text{educ}}$	$\widehat{v}$
3.1	11	2	13.48	-2.4805
3.24	12	22	12.26	-0.2582
3	11	2	13.4805	-2.4805
6	8	44	10.9137	-2.9137

Table 1: Data + predicted values + residuals for regression (5)

4.2.3 Step 3: regressing wage on the residuals

Now we consider this second regression,

$$\text{wage} = \delta_0 + \delta_1 \widehat{v} + w \ . \tag{7}$$

In the model of Eq. (7) $\widehat{v}$ is now a regressor and the error term is denoted w. Again, I use δ (= delta, the Greek letter for "d") to avoid confusion with the other regressions (and the same for the error term, denoted here w).

Since $\widehat{v}$ is a "purified" `educ`, the coefficient δ_1 will capture the net effect of educ on wage. So will obtain

$$\widehat{\delta}_1 = \widehat{\beta}_1 \ . \tag{8}$$

That is, the coefficient $\widehat{\beta}_1$ we're after in (1) is in fact equal to the coefficient $\widehat{\delta}_1$ of model (7). Let's check this.

We know that

$$\widehat{\delta}_1 = \frac{\text{cov}(\widehat{v},\texttt{wage})}{\text{var}(\widehat{v})} \tag{9}$$

So we have

$$\widehat{\delta}_1 = \frac{4.4967}{6.9795} = 0.64427 \tag{10}$$

We can see that the coefficient we obtain is the same as the coefficient given in (3).

4.3 Calculating $\widehat{\beta}_2$

To calculate the estimate $\widehat{\beta}_2$ we have to proceed in the same way, but now interchanging the roles of `educ` and `exper`. Now that you are experts I write the two regressions that you have to do using always the notation β for the coefficients and u for the parameters. That is, you have to do this:

Step 1 Run the regression $\text{exper} = \beta_0 + \beta_1 \text{educ} + u$.

Step 2 Calculate the residuals $\widehat{u}$ (with the OLS estimates you found in Step 1).

Step 1 Run the regression $\text{wage} = \beta_0 + \beta_1 \widehat{u} + v$, where $\widehat{u}$ are the residuals you calculated in Step 2.

4.4 Calculating $\widehat{\beta}_0$

Now, to calculate the coefficient $\widehat{\beta}_0$ of (2), you just have to solve this equation:

$$\overline{\text{wage}} = \widehat{\beta}_0 + \widehat{\beta}_1 \overline{\text{educ}} + \widehat{\beta}_2 \overline{\text{exper}} \tag{11}$$

where, I recall, $\overline{\text{wage}}$ is the average of the wage column (and similarly for $\overline{\text{educ}}$ and $\overline{\text{exper}}$), and $\widehat{\beta}_1$ and $\widehat{\beta}_2$ are the OLS estimates we have just calculated.

4.5 Using R

I suppose you load the data set `wage1.RData`.

You first run the following regression model

```
model <- lm(educ ~ exper)
```

Then you extract the residuals. There's the slow method (see the notes on R), or there's a faster way (there should, because extracting residuals is a classic operation in econometrics), which consists of entering the following command:

```
res = residuals(model)
```

Here we are creating a new variable that we call `res`.

Then you just call the following regression (that we call `model2`):

```
model2 <- lm(wage ~ res)
```

And you get

```
Call:
lm(formula = wage ~ res)

Residuals:
    Min      1Q  Median      3Q     Max
-5.4885 -2.0468 -0.8275  1.2669 16.1159

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)   5.8961     0.1430   41.22   <2e-16 ***
res           0.6443     0.0542   11.89   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 3.281 on 524 degrees of freedom
Multiple R-squared:  0.2124,        Adjusted R-squared:  0.2109
F-statistic: 141.3 on 1 and 524 DF,  p-value: < 2.2e-16
```

You can see that we get the same estimate as in Eq. (10). Then you store that value:

```
coeffBeta1 = coefficients(model2)
```

You need to remember that $\widehat{\beta}_1 =$ `coeffBeta1[2]`. If you want, you can enter the command

```
beta1 = coeffBeta1[2]
```

To get $\widehat{\beta}_1$ you do this (there's no need to type "`summary(model3)`" or "`summary(model4)`".

```
model3 <- lm(exper ~ educ)
residualsModel3 = residuals(model3)
model4 <- lm(wage ~ residualsModel3)
summary(model4)
coeffBeta2 = coefficients(model4)
beta2=coeffBeta2[2]
```

You can check that beta2 is the value you wanted: You just enter `beta2` and you get this:

```
beta2
residualsModel3
      0.0700954
```

It says "`residualsModel3` because the value of beta2 is the coefficient of the regressor named `residualsModel3` in the regression `model4`.

Then to get $\widehat{\beta}_0$ you just do

```
beta0 = mean(wage)-beta1*mean(educ)-beta2*mean(exper)
```

And we can check:

```
beta0
      res
-3.390539
```

which corresponds to the value found in Eq. (2).

5 Logarithms

If you need to calculate the log of a variable, say, `educ`, you just type the following command:

```
logEduc = log(educ)
```

It will calculate the **natural logarithm**, and store the results in a new variable called here "`logEduc`" (again, you can choose whatever name you like).

Since the logarithm is used frequently in econometrics many data sets already include the logarithms of some variables. Often (but not always!), if the variable in the data set is called "`theSuperCoolVariable`" then the logarithm of that variable will be called "`ltheSuperCoolVariable`" (i.e., adding the letter "l" at the beginning of the name of the variable). This is why it is worth looking at the list of variables included in the data set before doing anything.

6 Plots

You can plot the data and draw the regression line, too.

Here is what you have to do if you want to plot:

- the data, with wages in the vertical axis and education in the horizontal axis
- the regression line, for the regression

$$wage = \beta_0 + \beta_1 educ + u$$

If you do that regression you find

```
model2 <- lm(wage ~ educ)

summary(model2)

Call:
lm(formula = wage ~ educ)

Residuals:
    Min      1Q  Median      3Q     Max
-5.3396 -2.1501 -0.9674  1.1921 16.6085

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.90485    0.68497  -1.321    0.187
educ         0.54136    0.05325  10.167   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 3.378 on 524 degrees of freedom
Multiple R-squared:  0.1648,        Adjusted R-squared:  0.1632
F-statistic: 103.4 on 1 and 524 DF,  p-value: < 2.2e-16
```

So that means

$$\widehat{\beta}_0 = -0.90485 \qquad \text{and} \qquad \widehat{\beta}_1 = -0.54136$$

To plot the data you do the following sequence

```
png(filename="wage-educ.png")

plot(educ,wage, pch = 16, cex = 1.3, col = "blue", main = "Hourly wage
against education", xlab = "education (years)", ylab = "hourly wage ($)")

abline(0.62282,0.50645, col = "black")
```

```
dev.off()
```

The first command,

```
png(filename="wage-educ.png")
```

says that you will save a picture (with format `png`, a standard image file, the same that your phone is using when you make a screen capture) with the name `wage-educ.png`. It will be stored in the **working directory**. If you want to save it in another directory you have to specify the path.

The next command is the one creating the plot:

```
plot(educ,wage, pch = 16, cex = 1.3, col = "blue", main = "Hourly wage
against education", xlab = "education (years)", ylab = "hourly wage ($)")
```

Important: This is only 1 line. Here you see 2 lines because it's a long command, it wouldn't fit on the page.

That command says:

- you will plot `educ` in the vertical axis and `wage` in the horizontal axis.
- `pch=16` is a code that specifies the symbol used to plot the data. Here 16 means that each entry will be a big, solid points. If you type `pch=1` then you'll have circles instead of big, solid points. If you type `pch=0` then each entry will be represented by a square.

  You have a list of possible options here (look for the section "Plotting Symbols"):

  `https://www.statmethods.net/advgraphs/parameters.html`

- `cex=1.3` is a scaling factor for the text and symbols. Here it means we want it 30% bigger than the default size.
- `col= "blue"` means that I want the data points in blue.
- `main = "Hourly wage against education"` is the title of my picture

- `xlab = "education (years)"` says that I want "education (years)" as the title of the horizontal axis.
- `ylab = "hourly wage ($)"`. Same as above but for the vertical axis.

Then the command

```
abline(0.62282,0.50645, col = "black")
```

is invoked to add to the graph the regression line:

- 0.62282 is the intercept ($\widehat{\beta}_0$)
- 0.50645 is the slope ($\widehat{\beta}_1$)
- `col = "black"` sets the color of the regression line.

If you want to avoid errors it is perhaps best to ask R to use the OLS estimates instead of typing them. In this case you would first do the following:

```
CoeffsSecondRegression = coefficients(model2)
```

and then instead of `abline(0.62282,0.50645, col = "black")` you type

```
abline(CoeffsSecondRegression[1], CoeffsSecondRegression[2], col = "black")
```

Finally, the last command:

```
dev.off()
```

It tells R to close the file and write it on the hard drive (where we asked him with the first command).

When you type `dev.off()`, you have as an answer

```
null device
          1
```

That's not an error. It just says that everything went fine. Then you can see the file and you obtain the picture in Figure 1.

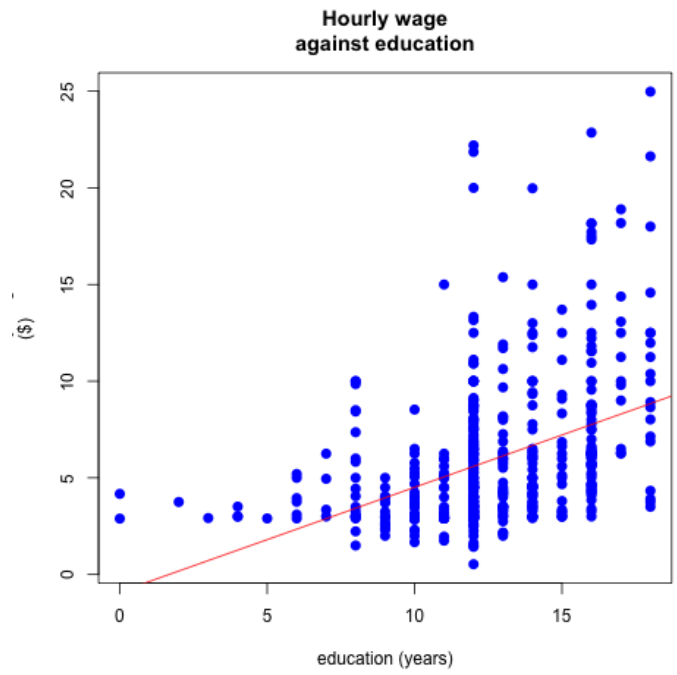

Figure 1: Wage and Education data with the regression line

```
Call:
lm(formula = wage ~ res)

Residuals:
    Min      1Q  Median      3Q     Max
-5.4885 -2.0468 -0.8275  1.2669 16.1159

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.8961     0.1430   41.22   <2e-16 ***
res           0.6443     0.0542   11.89   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 3.281 on 524 degrees of freedom
```

```
Multiple R-squared:  0.2124,        Adjusted R-squared:  0.2109
F-statistic: 141.3 on 1 and 524 DF,  p-value: < 2.2e-16
```