Lecture 3: The Classical Regression Model

Thomas Chadefaux

Contents

1 Why not simple regression? 2

2 Regression model in matrix form 2

3 Finding the line of best fit 3

4 Derivation of the least square estimator 3

4.1 K=2 3

4.2 K > 2, matrix notation 4

5 An Example 5

6 Interpreting the results 6

7 Geometric Interpretation 7

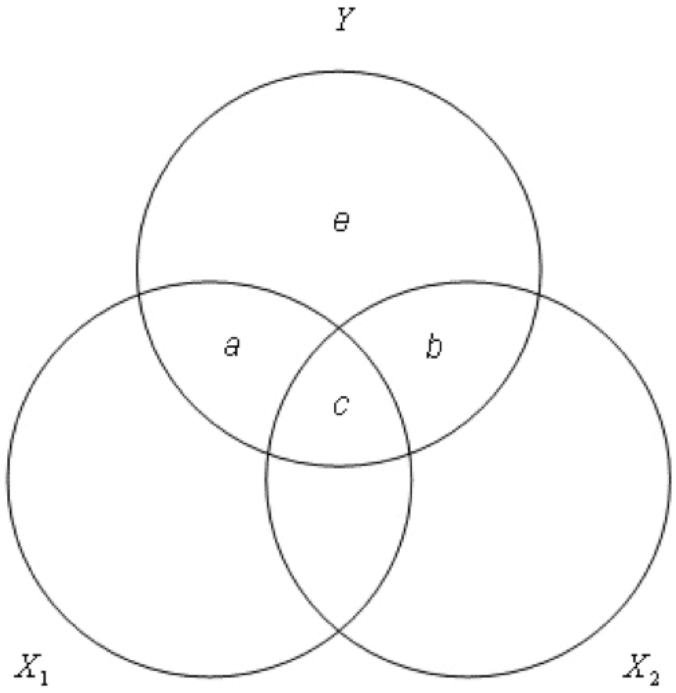
8 Intuition using Bootstrapping 8

1 Why not simple regression?

The effect of each variable could be studied in isolation. But the results may be misleading if the explanatory variables are mutually related. For example, voting might be affected by age and by wealth. But if we regress 'vote' on 'age' and then vote on wealth, we are likely to overestimate the importance of both age and wealth, because both are correlated. In other words, some of the effect of age on vote is accounted for by wealth (e.g., older people might simply be wealthier). Attributing the entire effect to age will lead to a biased coefficient.

The multiple linear regression model is used to study the relationship between a dependent variable y and one or more independent variables $x_2, x_3, \ldots, x_K$.

Why not just use the correlation coefficient? The correlation coefficient gives a measure of whether two variables are associated with one another, but not of the kind of relationship involved. It is also limited to two variables.



2 Regression model in matrix form

Our model is

$$y_i = \beta_1 + \beta_2 x_{2i} + \beta_3 x_{3i} + \ldots + \beta_k x_{ki} + \varepsilon_i. \qquad (1)$$

This model can be expressed in matrix form as follows: $y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$, $X = \begin{pmatrix} 1 & x_{12} & \ldots & x_{1k} \\ 1 & x_{22} & \ldots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n2} & \ldots & x_{nk} \end{pmatrix}$, $\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{pmatrix}$, $\varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$, or as

$$y = X\beta + \varepsilon$$

and the estimated model as

$$y = Xb + e$$

Alternatively, we can also write it as:

$$y_i = \mathbf{x}_i'\mathbf{b} + e_i,$$

where $\mathbf{x}_i' = \begin{pmatrix} 1 & x_{1i} & x_{2i} & \ldots & x_{ki} \end{pmatrix}$. The **disturbance** associated with the ith data point is:

$$\varepsilon_i = y_i - \mathbf{x}_i'\boldsymbol{\beta},$$

and the **residual** is

$$e_i = y_i - \mathbf{x}_i'\mathbf{b},$$

Another way to put it:

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + \varepsilon_i = \mathbf{x}_i'\mathbf{b} + e_i = \hat{y}_i + e_i,$$

3 Finding the line of best fit

The question we want to ask is: which linear combination of $x_2, x_3, \ldots, x_K$ and a constant gives a good approximation of y? For example, is

$$\tilde{\beta}_1 + \tilde{\beta}_2 x_2 + \ldots + \tilde{\beta}_K x_K$$

a good approximation of y? One way to determine this is to evaluate

$$y_i - \left(\tilde{\beta}_1 + \tilde{\beta}_2 x_{i2} + \ldots + \tilde{\beta}_K x_{iK}\right),$$

which can be rewritten as

$$y_i - x_i'\tilde{\beta},$$

where $x_i = \begin{pmatrix} 1 \\ x_{i2} \\ x_{i3} \\ \vdots \\ x_{iK} \end{pmatrix}$ and $\tilde{\beta} = \begin{pmatrix} \tilde{\beta}_1 \\ \tilde{\beta}_2 \\ \tilde{\beta}_3 \\ \vdots \\ \tilde{\beta}_K. \end{pmatrix}$

Obviously we want to choose values for $\tilde{\beta}_1, \ldots \tilde{\beta}_K$ such that these differences are small. The most common way is to choose $\tilde{\beta}$ such that the sum of squared differences is as small as possible. I.e., we want to find a vector $\tilde{\beta}$ such that

$$\tilde{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} \left(y_i - x_i'\beta\right)^2$$

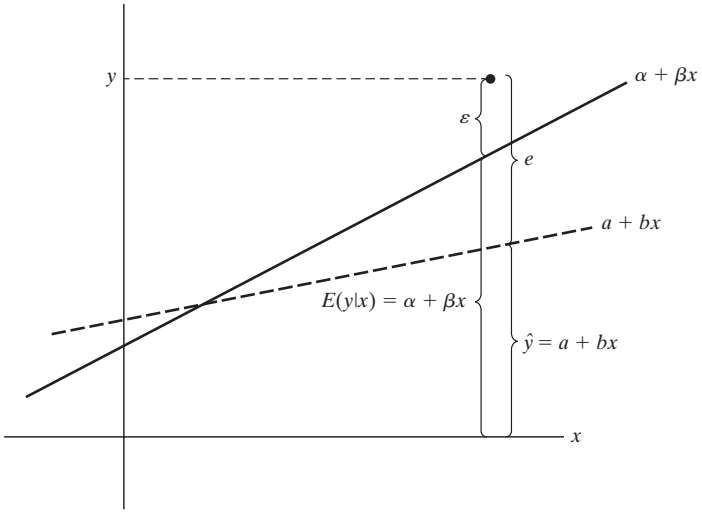


Figure 1: Population and Sample Regression (Source: Greene, p. 67)

4 Derivation of the least square estimator

4.1 K=2

Consider the case where $K = 2$. Then we want to minimise

$$S(\tilde{\beta}_0, \tilde{\beta}_1) = \sum_{i=1}^{N} \left(y_i - \tilde{\beta}_0 - x_i\tilde{\beta}_1\right)^2 = \sum_{i=1}^{N} e_i^2$$

To do that, we solve

$$\frac{\partial S(\tilde{\beta}_0, \tilde{\beta}_1)}{\partial \tilde{\beta}_0} = -2\sum_{i=1}^{N}(y_i - \tilde{\beta}_0 - x_i\tilde{\beta}_1) = 0$$
$$\frac{\partial S(\tilde{\beta}_0, \tilde{\beta}_1)}{\partial \tilde{\beta}_1} = -2\sum_{i=1}^{N} x_i(y_i - \tilde{\beta}_0 - x_i\tilde{\beta}_1) = 0,$$

from which we can obtain b_1 and b_2, our estimates of the population parameters β_1 and β_2 as:

$$b_0 = \bar{y} - b_1\bar{x} \tag{3}$$
$$b_1 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{4}$$

(3) and (4) are called the normal equations.

Note that $b_1 = \frac{Cov(X,Y)}{Var(X)}$

How do we derive b_0 and b_1?

$$b_0 = \frac{1}{N}\sum_{i=1}^{N} y_i - b_1\frac{1}{N}\sum_{i=1}^{N} x_i = \bar{y} - b_1\bar{x} \tag{2}$$

and b_2, our estimate of the true population parameter β_1, as:

$$\frac{1}{N}\sum_{i=1}^{N} x_i y_i - b_0\frac{1}{N}\sum_{i=1}^{N} x_i - \left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right) b_1 = 0$$

Substituting (2), we get:

$$\frac{1}{N}\sum_{i=1}^{N} x_i y_i - N\bar{x}\bar{y} - \left(\sum_{i=1}^{N} x_i^2 - N\bar{x}^2\right) b_1 = 0$$

We can finally solve for b_1 as

$$b_1 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

4.2 K > 2, matrix notation

More generally, in matrix form, we have:

$$e = y - Xb,$$

and so

$$\begin{aligned} S(b) =& \sum_i e_i^2 \\ =& e'e \\ =& (y - Xb)'(y - Xb) \\ =& y'y - y'Xb - b'X'y + b'X'Xb \\ =& y'y - 2b'X'y + b'X'Xb \end{aligned} \tag{5}$$

Why the last step? Since $y'Xb$ is a scalar, it is equal to its transpose and so we can write $y'Xb = (y'Xb)' = b'X'y$

Now we want to minimise (5) with respect to b. Remember that b is a vector of the form $b = \begin{pmatrix} b_1 & b_2 & \dots & b_K \end{pmatrix}'$. So when we calculate $\frac{\partial S(b)}{\partial b}$, we'll get K first order conditions. So let's calculate $\frac{\partial S(b)}{\partial b}$:

$$\begin{aligned} \frac{\partial S(b)}{\partial b} =& 0 - 2X'y + 2X'Xb = 0 \\ & \rightarrow X'Xb = X'y \\ & \boxed{\rightarrow b = (X'X)^{-1}X'y} \end{aligned} \tag{6}$$

That we get K first order conditions for K variables is intuitive. Remember the case where we had $y_i = \alpha + \beta x_i$, we obtained two normal equations by calculating the derivative with respect to α, and then the derivative wrt β.

This, of course, assumes that $(X'X)^{-1}$ exists, and hence that X has full rank (see lecture 2). Practically, this means that we need the number of parameters k to be smaller than the number of observations n.[1]

5 An Example

To gain more intuition for (6), let us see how it produces the same results as (3) and (4) for the case where $k = 2$. Consider in particular a model of political affiliation on a left/right scale from 0 to 10. Our explanatory variable is the individual's yearly income in 1,000s of Euros.

$$y_i = \beta_0 + \beta_1 income_i + \varepsilon_i$$

Consider the following sample data:

L/R scale (y)	constant	Income (x)
2	1	10
3	1	42
7	1	39
1	1	30
9	1	80

Let us now use our formula (3) and (4) to calculate the coefficients b_0 and b_1.

$$\begin{cases} b_0 & = \bar{y} - b_1\bar{x} = 4.4 - b_1 40.2 \\ & \text{and} \\ b_1 & = \frac{\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{N}(x_i-\bar{x})^2} \\ & = \frac{(10-40.2)(2-4.4)+(42-40.2)(3-4.4)+(39-40.2)(7-4.4)+(30-40.2)(1-4.4)+(80-40.2)(9-4.4)}{(10-40.2)^2+(42-40.2)^2+(39-40.2)^2+(30-40.2)^2+(80-40.2)^2} \\ & = 0.1092598, \end{cases}$$

which gives us

$$b_0 = 4.4 - b_1 40.2 = 4.4 - 0.1092598 \times 40.2 = 0.007755$$

Now let us do the same thing using the matrix notation formula:

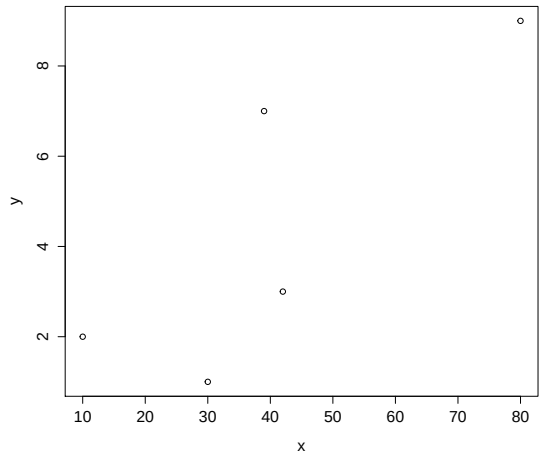
$$\begin{aligned} b &= (X'X)^{-1}X'y \\ &= \left(\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 10 & 42 & 39 & 30 & 80 \end{pmatrix} \begin{pmatrix} 1 & 10 \\ 1 & 42 \\ 1 & 39 \\ 1 & 30 \\ 1 & 80 \end{pmatrix} \right)^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 10 & 42 & 39 & 30 & 80 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \\ 7 \\ 1 \\ 9 \end{pmatrix} \\ &= \begin{pmatrix} 5 & 201 \\ 201 & 10685 \end{pmatrix}^{-1} \begin{pmatrix} 22 \\ 1169 \end{pmatrix} \\ &= \begin{pmatrix} 0.82040848 & -0.0154330467 \\ -0.01543305 & 0.0003839066 \end{pmatrix} \begin{pmatrix} 22 \\ 1169 \end{pmatrix} \\ &= \begin{pmatrix} 0.007755 \\ 0.109260 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \end{aligned}$$

Figure 2:

```
#--- Input data
x = c(10, 42, 39, 30,80)
y = c(2,3,7,1,9)

#--- Plot it
plot(x, y)
```

Listing 1: BasicRegPlot.R



Let us check our results using R (normal equations version):

```
#--- Input data
x = c(10, 42, 39, 30,80)
y = c(2,3,7,1,9)

#--- calculate b0 and b1 using the normal equations
numerator <- sum((x - mean(x))  *  (y - mean(y)))
denominator <- sum((x - mean(x))^2)
b1 <- numerator / denominator

b0 <- mean(y) - b1 * mean(x)
print(c(b0,b1))
```

Listing 2: normalEqns.R

Let us check our results using R (matrix version):

```
#--- Input data
x0 = c(1,1,1,1,1)
x1 = c(10, 42, 39, 30,80)
X <- cbind(x0,x1)
y = c(2,3,7,1,9)

# Calculate (X'X)^{-1}X'y
XprimeX <- t(X)%*%X
XprimeXinverse <- solve(XprimeX)
Xprimey <- t(X)%*%y

XprimeXinverse %*% Xprimey

# or the equivalent, in one line:
solve(t(X)%*%X) %*% t(X)%*%y
```

Listing 3: bmatrix.R

We get the exact same result. Why? Let's take a look at what exactly $b = (X'X)^{-1}X'y$ does. For clarity we can rewrite it as $X'Xb = X'y$. Note that it can be rewritten as

$$b_0 n + b_1 \sum_i x_i = \sum_i y_i$$
$$b_0 \sum_i x_i + b_1 \sum_i x_i^2 = \sum_i y_i x_i$$

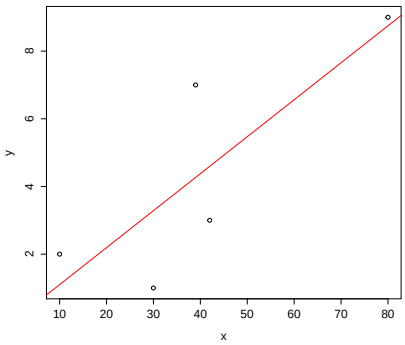
Now solving for b_0 and b_1 , and after a bit of algebraic manipulation, we get (3) and (4) back.

6 Interpreting the results

What we have calculated is the intercept (b_0) and the slope (b_1). So we have

$$\begin{aligned}\hat{y}_i &= b_0 + b_1 x_i \\ &= 0.007755 + 0.109260 \times income\end{aligned}$$

We can now plot our fitted line.



Note that we can also calculate our fitted values and residuals and plot them:

L/R scale (y)	constant	Income	fitted	residuals
2	1	10	1.1	0.9
3	1	42	4.6	-1.6
7	1	39	4.3	2.7
1	1	30	3.3	-2.3
9	1	80	8.7	0.3

Finally, let's check our results using R's canned function, lm(y x):

```
#--- Input data
x = c(10, 42, 39, 30, 80)
y = c(2,3,7,1,9)

#--- run the regression
lm(y ~ x)
```

Listing 4: basicReg.R

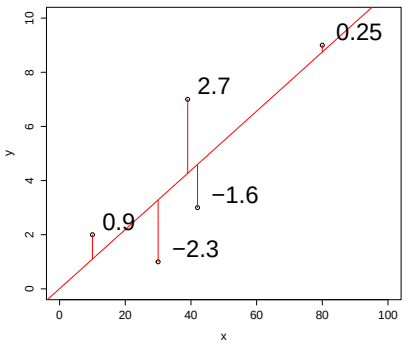
```
#--- Input data
x = c(10, 42, 39, 30, 80)
y = c(2,3,7,1,9)

#--- run the regression
lm(y ~ x)

#--- Plot it
plot(x, y)
abline(a = 0.007755, b = 0.109260, col=2)

fitted(lm(y ~ x))
```

Listing 5: basicRegPlotWithLine.R




```r
#--- Input data
x = c(10, 42, 39, 30, 80)
y = c(2,3,7,1,9)

#--- plot scatterplot and the regression line
lm1 <- lm(y ~ x)
plot(x, y, xlim=c(0,100), ylim=c(0,10))
abline(a = 0.007755, b = 0.109260, col=2) #note: this is easier and does
      the same thing: abline(lm1)

# calculate residuals and predicted values
res <- signif(residuals(lm1), 2)
fit <- predict(lm1)

# plot distances between points and the regression line
segments(x, y, x, fit, col="red")

# add labels (res values) to points
library(calibrate)
textxy(x, y, res, cex=2)
```

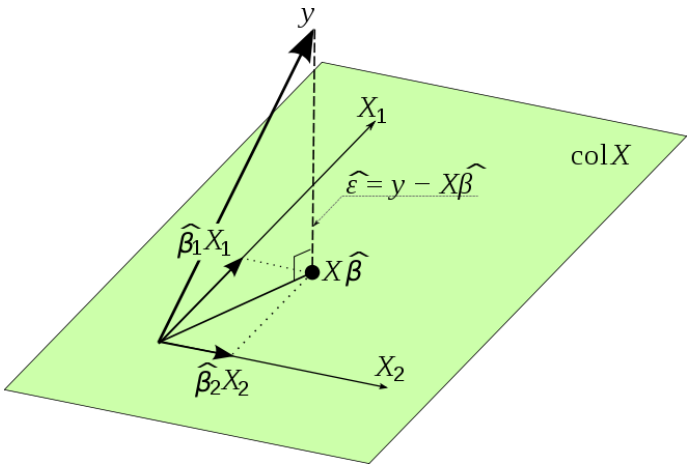
Listing 6: fitAndResid.R

What do these coefficients mean substantively? b_0 is the intercept. That is, it is the value y takes if $x = 0$. And b_1 is the effect of our variable x (income in this case). The interpretation is that if your income is 0, then we expect your position on the L/R scale to be very close to 0. And for every increase in one unit of income, your expected position on the scale will change by $b_1 = 0.1$. Since income is measured in thousands of euros here, one 'unit' of income is 1,000 euros. So if your income increases by 10,000, we expect you to move up one point on the L/R scale.[2]

In-class exercise: what happens to the coefficient if you multiply income by 1,000?

[2] More formally, $\frac{\partial y}{\partial income} = 0.109260$

7 Geometric Interpretation



8 Intuition using brute force

Another way of getting the same results involves bootstrapping (though it is mathematically less "clean"). Remember that the goal is to fit a line through a set of point, in such a way that this line fits the points as well as possible. I.e., we are looking for two coefficients (for a simple regression, more otherwise, obviously): an intercept (b_0) and a slope (b_1). One way to find is simply brute force. I.e., try every possible combination of b_0 and b_1 until you get a good fit. I.e., until the sum of squared residuals ($\sum(y - \hat{y})^2$) is as small as possible.

In **R**, for example:

```
#--- generate some data:

x1 <- rnorm(1000)
y <- 0.6 + 0.2*x1 + rnorm(1000)

#--- for each combination of b0 and b1, calculate the sum of squared res:
       -----
results <- NULL # creates an empty object where we will store our results
for(b0 in seq(-2, 2, 0.1)){
     print(b0)
     for(b1 in seq(-2, 2, 0.1)){
          e <- y - (b0 + b1*x1)
          sum.of.resid.squared <- sum(e^2)
          results.tmp <- data.frame(sum.of.res = sum.of.resid.squared,
                                    b0 = b0,
                                    b1 = b1)
          results <- rbind(results, results.tmp)
     }
}

# let's see what we got:
head(results)

# which combination of b0 and b1 gives us the best fit (i.e., smalles sum
      of residuals squared?)
best <- which(results$sum.of.res == min(results$sum.of.res))
results[best, ]

#let's check with the canned method:
lm(y ~ x1)
```

Listing 7: bootstrapping.R

Of course there are problems with this method: first, we might not know where to look, and so would need an enormous search space. Second, the computational challenges are such that our results risk being approximate. [3]

[3] Note that there are better ways than to simply go about the process randomly. I.e., you should not go through every possible combination, as there are various techniques to get to the result faster. This was just for illustration purposes

9 Graphical intuition: what does it mean to 'control for', or 'holding other variables constant'?

See plot3d.R