Lecture 9: The Linear Regression Model

Thomas Chadefaux

Quantitative Methods I

Simple Linear Regression

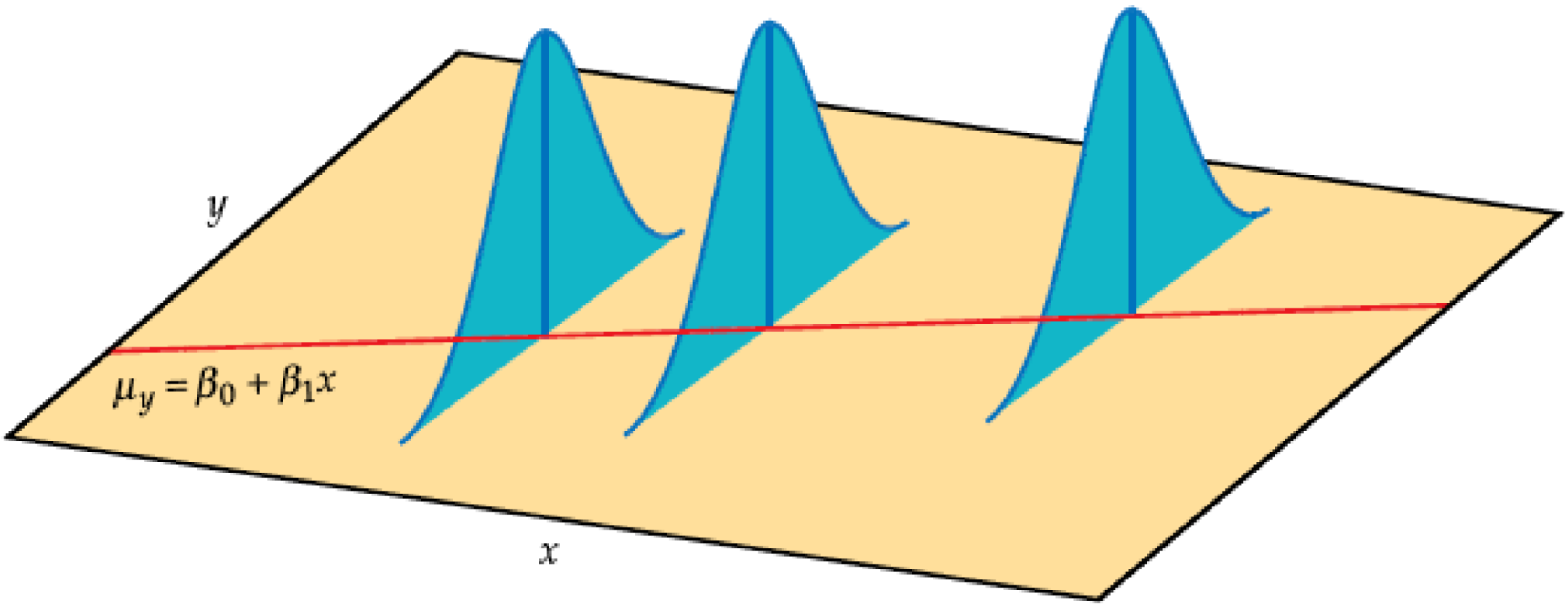
- Relationship between y and x
- Different values of $x \longrightarrow$ different values of y.
- We assume that y is a linear function of x
- Basic model:

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

- This is the population regression line.

Simple Linear Regression (cont'd)

- The regression line (red line in graph below) describes how the mean response changes with x.
- Observed y's will vary around these means. The model assumes that the variation is the same for all x.



Simple Linear Regression

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

- There are two parameters to estimate: β_0 and β_1.
- β_0 is the intercept of the population regression line. I.e., it is the expected value that y takes when $x = 0$.
- β_1 is the slope of the regression line. I.e., for a one unit increase in x, the expected value of y increases by β_1.

Estimating the regression parameters

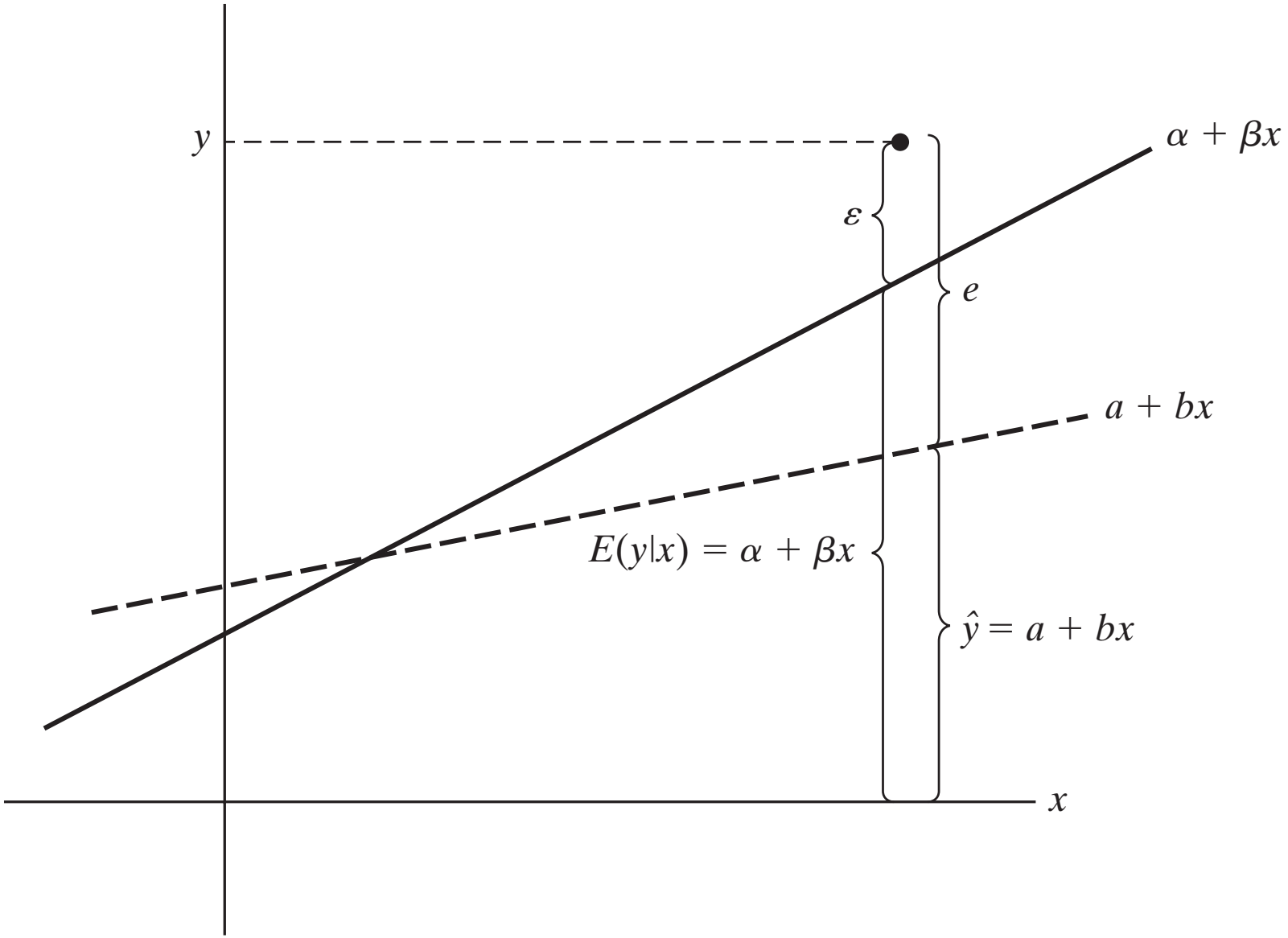
- The population model is

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

- Our estimated model is:

$$y_i = b_0 + b_1 x_i + \varepsilon_i$$

Regression



Estimating the parameters

- But how do we obtain b_0 and b_1 (a and b in the plot above)?
- 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

$$\tilde{\beta}_0 + \tilde{\beta}_1 x_1$$

a good approximation of y?

Finding the line of best fit

One way to determine this is to evaluate

$$\sum_i (y_i - \left(\tilde{\beta}_0 + \tilde{\beta}_1\right)),$$

which can be rewritten as $\sum_i (y_i - x_i' \tilde{\beta})$, where

$$x_i = \begin{pmatrix} 1 \\ x_{i1} \end{pmatrix}, \tilde{\beta} = \begin{pmatrix} \tilde{\beta}_0 \\ \tilde{\beta}_1 \end{pmatrix} \tag{1}$$

Obviously we want to choose values for $\tilde{\beta}_1, \ldots, \tilde{\beta}_K$ such that these differences are small.

Finding the line of best fit

$$y_i - \left(\tilde{\beta}_0 + \tilde{\beta}_1\right),$$

would not be a great way, however, since the errors would cancel out.
Two solutions: take the absolute value or the square
E.g.,

$$(y_i - \left(\tilde{\beta}_0 + \tilde{\beta}_1\right))^2,$$

Finding the line of best fit

We are going to choose $\tilde{\beta}$ such that the sum of squared differences is as small as possible. I.e., we want to find a vector $\tilde{\beta} = (\tilde{\beta}_0, \tilde{\beta}_1)$ such that

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

or, equivalently:

$$\tilde{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - \tilde{\beta}_0 - x_i\tilde{\beta}_1)^2$$

or, in short,

$$\tilde{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} e_i^2$$

How do we do that?

Derivation of the least square estimator

K=2

- Consider the case where $K = 2$. I.e., we have two parameters to estimate: β_0 and β_1 (and hence only 1 IV. Why?)
- 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

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

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

K=2 (cont'd)

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

$$b_0 = \bar{y} - b_1\bar{x} \tag{2}$$

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

(2) and (3) are called the normal equations.
Note that $b_1 = \frac{Cov(X,Y)}{Var(X)}$

(note: How did we derive b_0 and b_1?)

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

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 (4), 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

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

Finding the OLS estimates: An Example

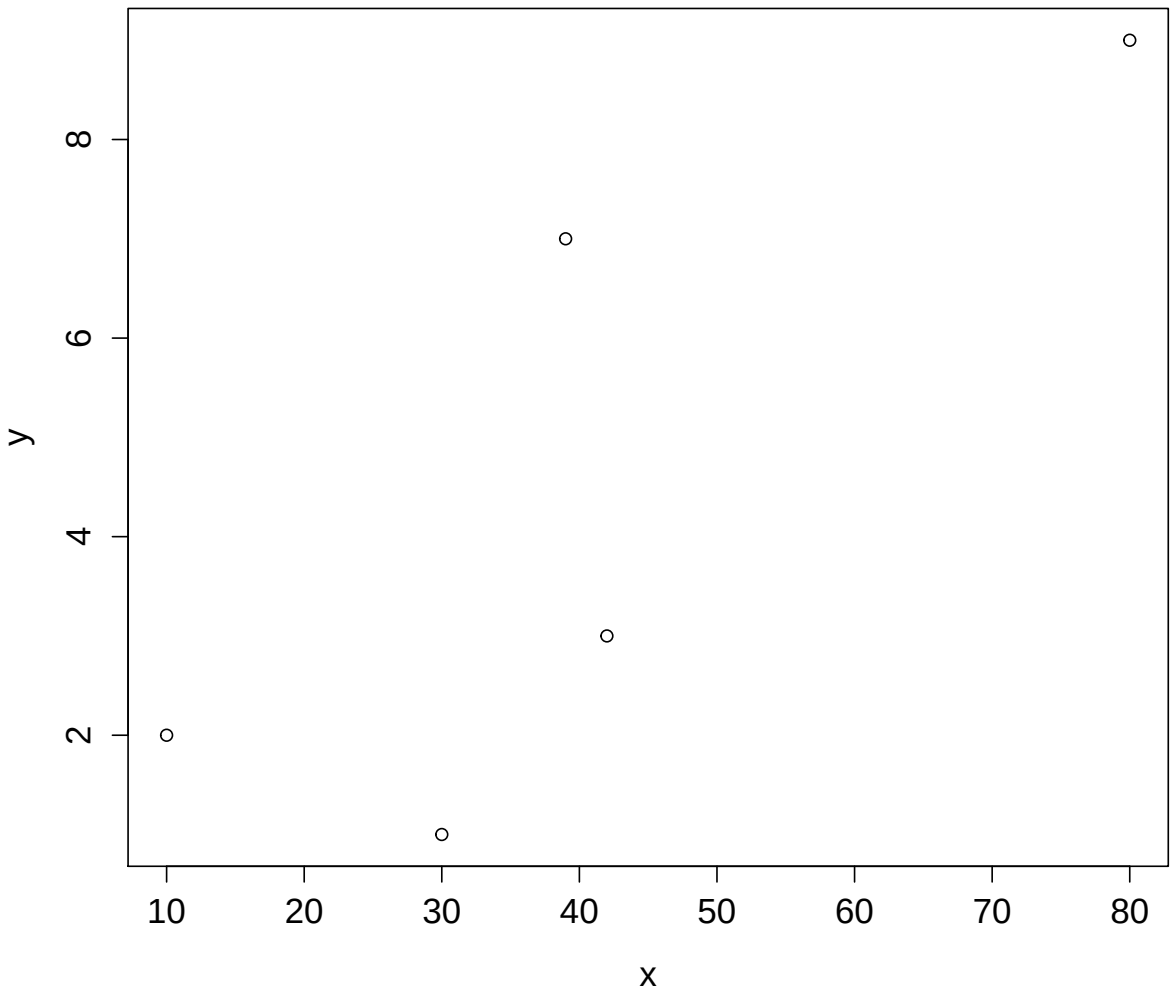
Consider a model of political affiliation on a left/right scale from 0 to 10. Our explanatory variable is the individual's yearly income in 1,000s of Euros.

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

We have 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

An Example



An Example

Let us now use our formula (2) and (3) to calculate the coefficients b_0 and b_1.

$$\begin{cases} b_0 & = \bar{y} - b_1\bar{x} = 4.4 - b_1 40.2 \\ & \text{and} \\ b_1 & = \frac{\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{N}(x_i-\bar{x})^2} \\ & = \frac{(10-40.2)(2-4.4)+(42-40.2)(3-4.4)+(39-40.2)(7-4.4)+(30-40.2)(1-4.4)+(80-4}{(10-40.2)^2+(42-40.2)^2+(39-40.2)^2+(30-40.2)^2+(80-40.2)^2} \\ & = 0.1092598, \end{cases}$$

which gives us

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

Check our results with R (normal equations version)

```r
#--- 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))
```

```
## [1] 0.007754914 0.109259828
```

Check our results using R (canned version):

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)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##    0.007755     0.109260
```

Interpreting the results

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

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

Interpretation

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 $b_1 \approx 0.1$.
    - Since income is measured in thousands of euros here, one 'unit' of income is $1,000$ euros.
    - So if your income increases by $1,000$, we expect you to move up one point on the L/R scale (More formally, $\frac{\partial y}{\partial income} = 0.109260$)
    - In-class exercise: what happens to the coefficient if you multiply income by $1,000$?

Now let us take a look at the regression output

The regression output

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

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##    0.007755     0.109260
```

Plot our fitted line.

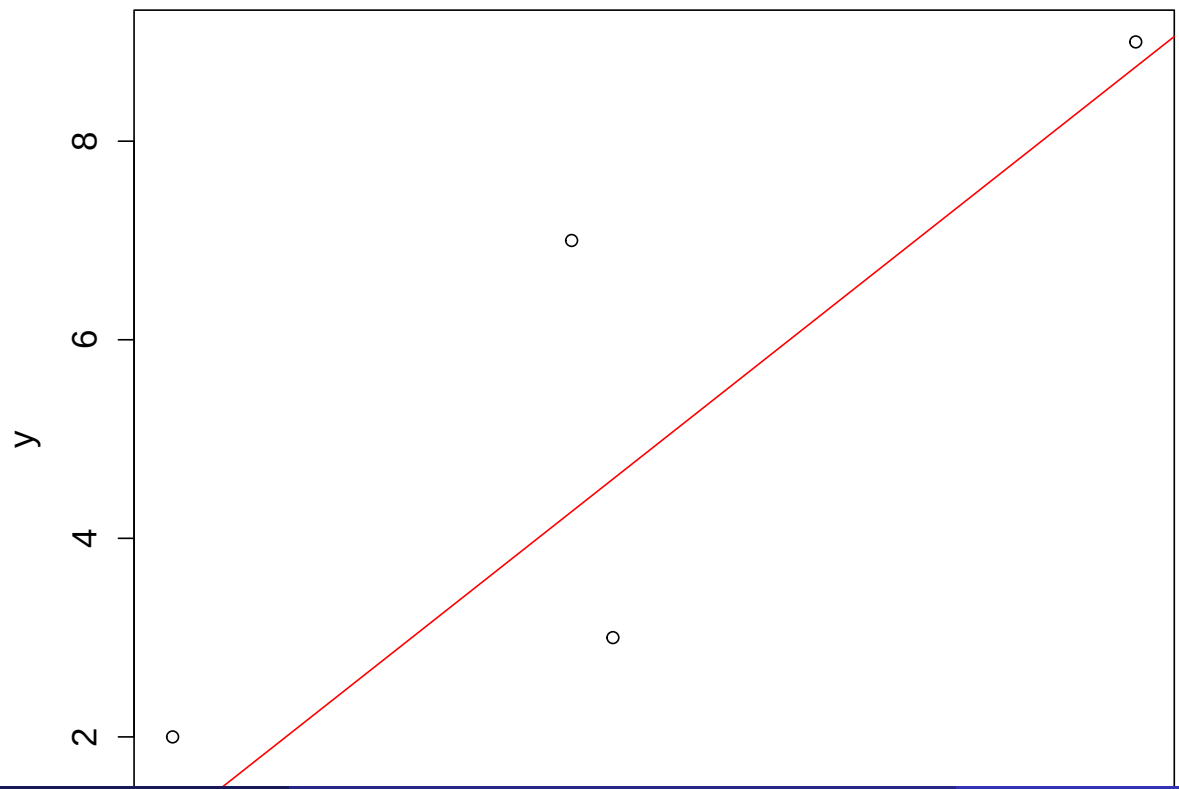
```
#--- Input data
x = c(10, 42, 39, 30, 80)
y = c(2,3,7,1,9)

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

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##    0.007755     0.109260
```

Plot our fitted line.

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



Fitted values and residuals

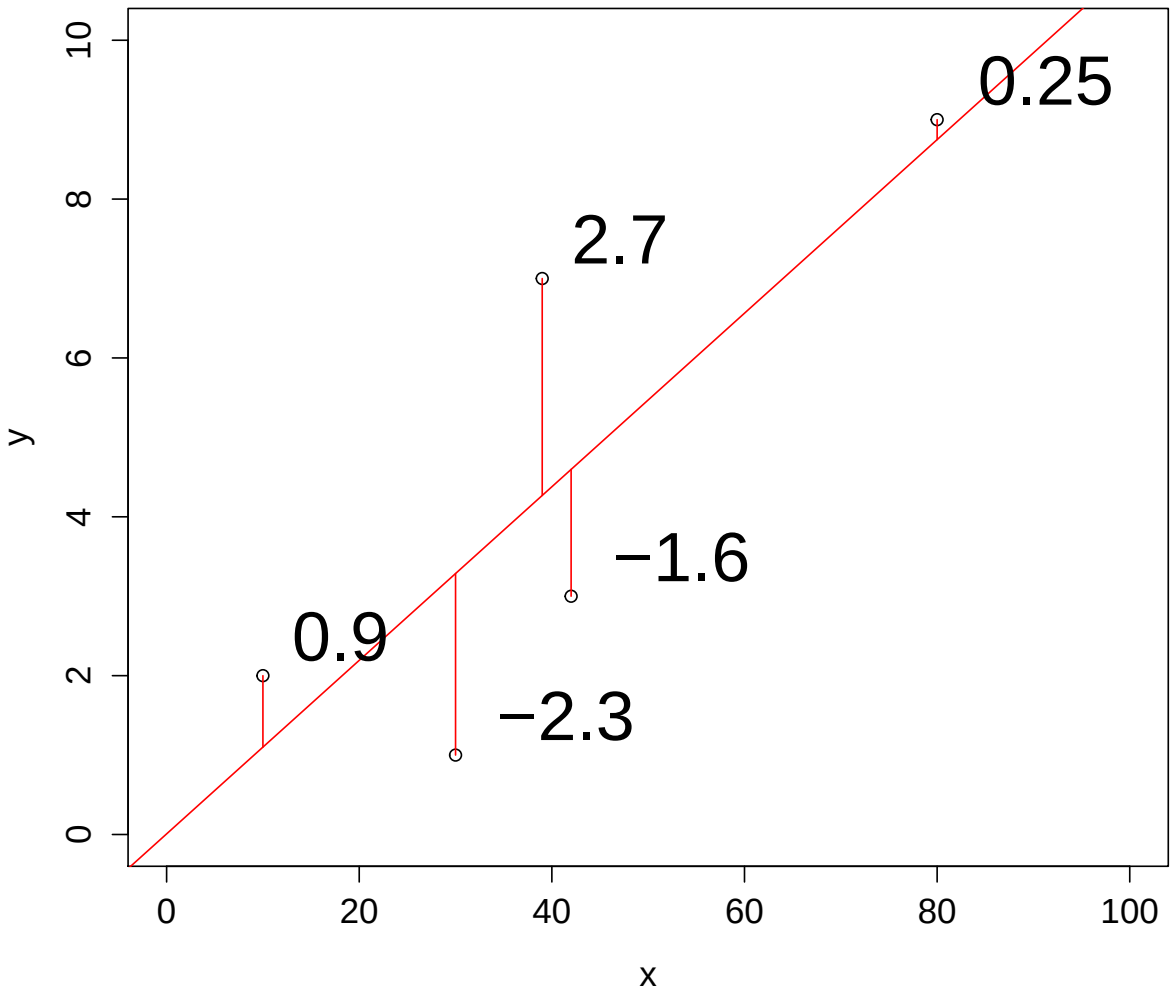
Note that we can also calculate our fitted values and residuals and plot them:

```
head(data.frame(x=x, 'filled values'= fitted(lm(y ~ x))),
```

```
##    x filled.values
## 1 10       1.100353
## 2 42       4.596668
## 3 39       4.268888
```

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

fitted values and residuals



Intuition using brute force

- Another way of getting the same results involves using brute (computational) force (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.
- More precisely, 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:

```r
#Generate some data and run the canned function:
x1 <- rnorm(1000)
y <- 0.6 + 0.2*x1 + rnorm(1000)
lm(y ~ x1)
```

```
##
## Call:
## lm(formula = y ~ x1)
##
## Coefficients:
## (Intercept)           x1
##      0.5637       0.2177
```

In R, for example:

For each combination (b_0, b_1), calculate the sum of squared residuals:

```
results <- NULL # creates an empty object
for(b0 in seq(-2, 2, 0.2)){
     for(b1 in seq(-2, 2, 0.2)){
          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)
          print(paste('b0 is ', b0,
                      'and b1 is ', b1,
                      ' and e2 is', sum.of.resid.squared)
     }}
```

Brute force results

```
# let's see what we got:
head(results, 4)
```

```
##   sum.of.res b0   b1
## 1   13001.69 -2 -2.0
## 2   12082.48 -2 -1.8
## 3   11248.51 -2 -1.6
## 4   10499.78 -2 -1.4
```

Brute force results

```
# Which combination of b0 and b1 gives us the best fit (
best <- which(results$sum.of.res == min(results$sum.of.re
results[best, ]
```

```
##     sum.of.res  b0  b1
## 285   1005.561 0.6 0.2
```

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

```
## (Intercept)          x1
##   0.5636911   0.2177118
```

Problems with brute force

- 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.
- 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