

# Assignment 10

Kornel Kovacs

11/15/2019

## Exercise 1:

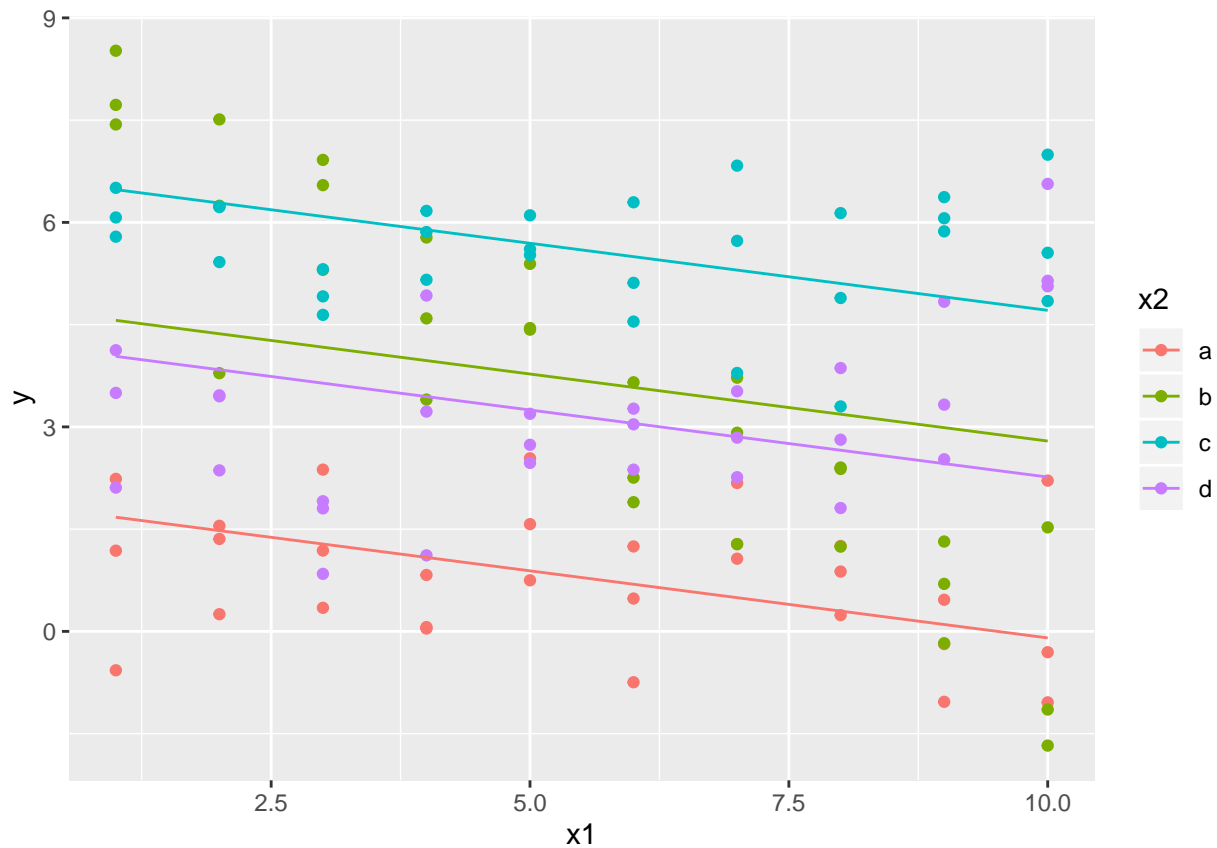
To make a conclusion and be able to interpret coefficients, I first need to replicate the models with the `sim3` dataset. We have a categorical variable, `x2` and a continuous one, `x1`. This is important for understanding the equations we estimate.

```
mod1 <- lm(y ~ x1 + x2, data = sim3)
mod2 <- lm(y ~ x1 * x2, data = sim3)

sim3 <- sim3 %>%
  add_predictions(mod1, var = "mod1") %>%
  add_predictions(mod2, var = "mod2")
```

Plotting `mod1`

```
ggplot(sim3, aes(x = x1, y = y, color = x2)) +
  geom_point() +
  geom_line(aes(y = mod1))
```



Making a summary of `mod1`

```
summary(mod1)
```

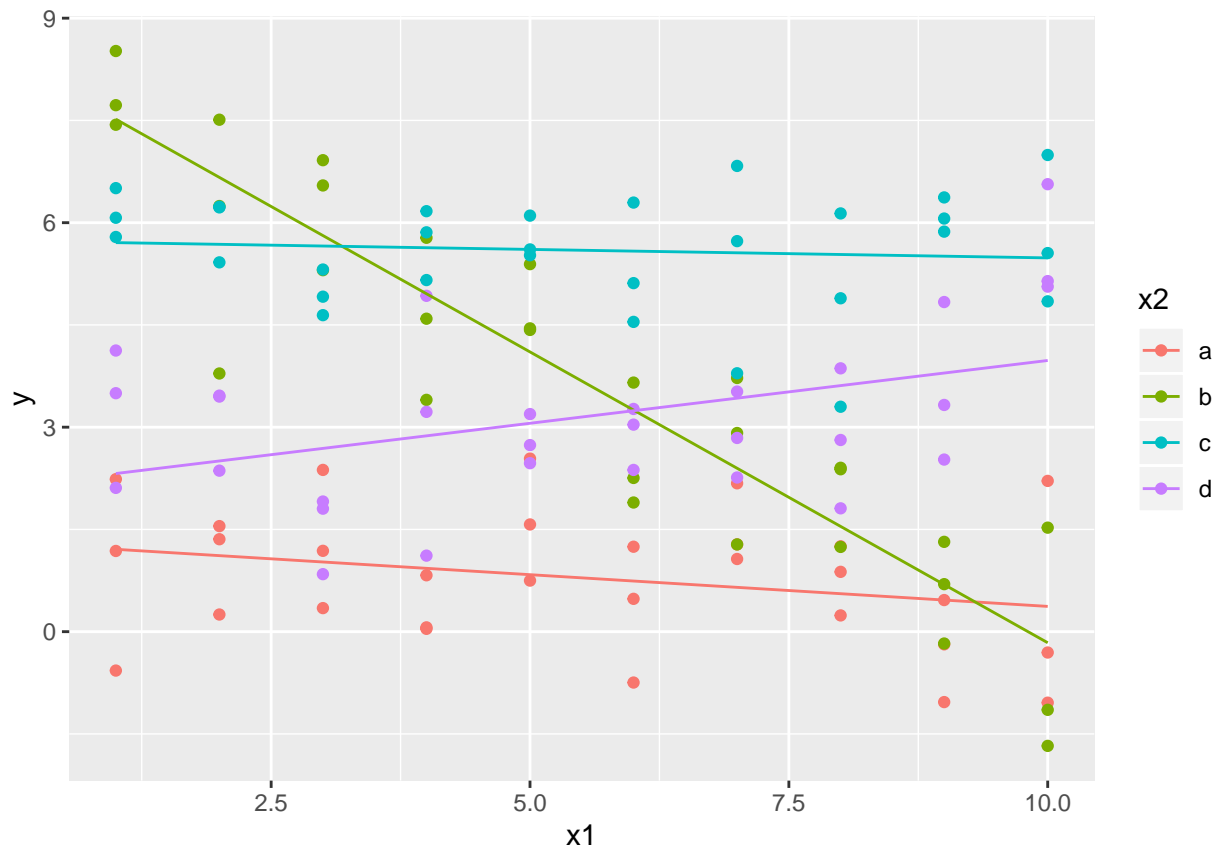
```
##
## Call:
## lm(formula = y ~ x1 + x2, data = sim3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.4674 -0.8524 -0.0729  0.7886  4.3005
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.87167    0.38738   4.832 4.22e-06 ***
## x1          -0.19674    0.04871  -4.039 9.72e-05 ***
## x2b           2.88781    0.39571   7.298 4.07e-11 ***
## x2c           4.80574    0.39571  12.145 < 2e-16 ***
## x2d           2.35959    0.39571   5.963 2.79e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.533 on 115 degrees of freedom
## Multiple R-squared:  0.5911, Adjusted R-squared:  0.5768
## F-statistic: 41.55 on 4 and 115 DF,  p-value: < 2.2e-16
```

#### Interpretation for mod1:

- **(Intercept):** It is the intercept when  $x_2$  is “a”, so the intercept of the red line.
- **$x_{2b}$ :** It shows how much the intercept is greater when  $x_2$  equals “b” compared to the baseline case (which is  $x_2 = \text{“a”}$ ). Therefore, the intercept of the green line is (Intercept) +  $x_{2b}$ , so  $1.87 + 2.89 = 4.76$ .
- **$x_{2c}$ :** It shows how much the intercept is greater when  $x_2$  equals “c” compared to the baseline case (which is  $x_2 = \text{“a”}$ ). Therefore, the intercept of the blue line is (Intercept) +  $x_{2c}$ , so  $1.87 + 4.8 = 6.67$ .
- **$x_{2d}$ :** It shows how much the intercept is greater when  $x_2$  equals “d” compared to the baseline case (which is  $x_2 = \text{“a”}$ ). Therefore, the intercept of the purple line is (Intercept) +  $x_{2d}$ , so  $1.87 + 4.8 = 6.67$ .
- **-0.20 (coefficient of  $x_1$ )** is the expected difference in  $y$  corresponding to one unit difference in  $x_1$  regardless of the value of  $x_2$  (so called *ceteris paribus*). It is also the slope of every single line in the graph. Since it is negative, the lines are descending.

#### Plotting mod2

```
ggplot(sim3, aes(x = x1, y = y, color = x2)) +
  geom_point() +
  geom_line(aes(y = mod2))
```



Making a summary of mod2

```
summary(mod2)
```

```
##
## Call:
## lm(formula = y ~ x1 * x2, data = sim3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.87634 -0.67655  0.04837  0.69963  2.58607
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.30124    0.40400   3.221  0.00167 **
## x1            -0.09302    0.06511  -1.429  0.15587
## x2b             7.06938    0.57134  12.373 < 2e-16 ***
## x2c             4.43090    0.57134   7.755 4.41e-12 ***
## x2d             0.83455    0.57134   1.461  0.14690
## x1:x2b        -0.76029    0.09208  -8.257 3.30e-13 ***
## x1:x2c         0.06815    0.09208   0.740  0.46076
## x1:x2d         0.27728    0.09208   3.011  0.00322 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.024 on 112 degrees of freedom
## Multiple R-squared:  0.8221, Adjusted R-squared:  0.811
## F-statistic: 73.93 on 7 and 112 DF,  p-value: < 2.2e-16
```

## Interpretation for Mod2

- **(Intercept):** It is the intercept when  $x_2$  is “a”, so the intercept of the red line.
- **$x_2b$ :** It shows how much the intercept is greater when  $x_2$  equals “b” compared to the baseline case (which is  $x_2 = \text{“a”}$ ). Therefore, the intercept of the green line is (Intercept) +  $x_2b$ , so  $1.3 + 7.07 = 11.37$ .
- **$x_2c$ :** It shows how much the intercept is greater when  $x_2$  equals “c” compared to the baseline case (which is  $x_2 = \text{“a”}$ ). Therefore, the intercept of the blue line is (Intercept) +  $x_2c$ , so  $1.3 + 4.43 = 5.73$ .
- **$x_2d$ :** It shows how much the intercept is greater when  $x_2$  equals “d” compared to the baseline case (which is  $x_2 = \text{“a”}$ ). Therefore, the intercept of the purple line is (Intercept) +  $x_2d$ , so  $1.3 + 0.83 = 2.13$ .
- **-0.09 (coefficient of  $x_1$ )** is the expected difference in  $y$  corresponding to one unit difference in  $x_1$  in case  $x_2$  equals “a”. It is also the slope of the red line in the graph. Since it is negative, the line is descending.
- **$x_1:x_2b$ :** It shows how much the slope is steeper when  $x_2$  equals “b” compared to the baseline case (which is  $x_2 = \text{“a”}$ ). Therefore, the slope of the green line is  $x_1 + x_1:x_2b$ , so  $-0.09 + -0.76 = -0.87$ . Since it is negative, the line is descending.
- **$x_1:x_2c$ :** It shows how much the slope is steeper when  $x_2$  equals “c” compared to the baseline case (which is  $x_2 = \text{“a”}$ ). Therefore, the slope of the blue line is  $x_1 + x_1:x_2c$ , so  $-0.09 + 0.07 = -0.02$ . Since it is negative, the line is descending.
- **$x_1:x_2d$ :** It shows how much the slope is steeper when  $x_2$  equals “d” compared to the baseline case (which is  $x_2 = \text{“a”}$ ). Therefore, the slope of the purple line is  $x_1 + x_1:x_2d$ , so  $-0.09 + 0.28 = 0.19$ . Since it is positive, the line is ascending.

## Exercise 2:

`data_grid()` is a function to find all the unique values of  $x_1$  and  $x_2$  and generates all combinations.

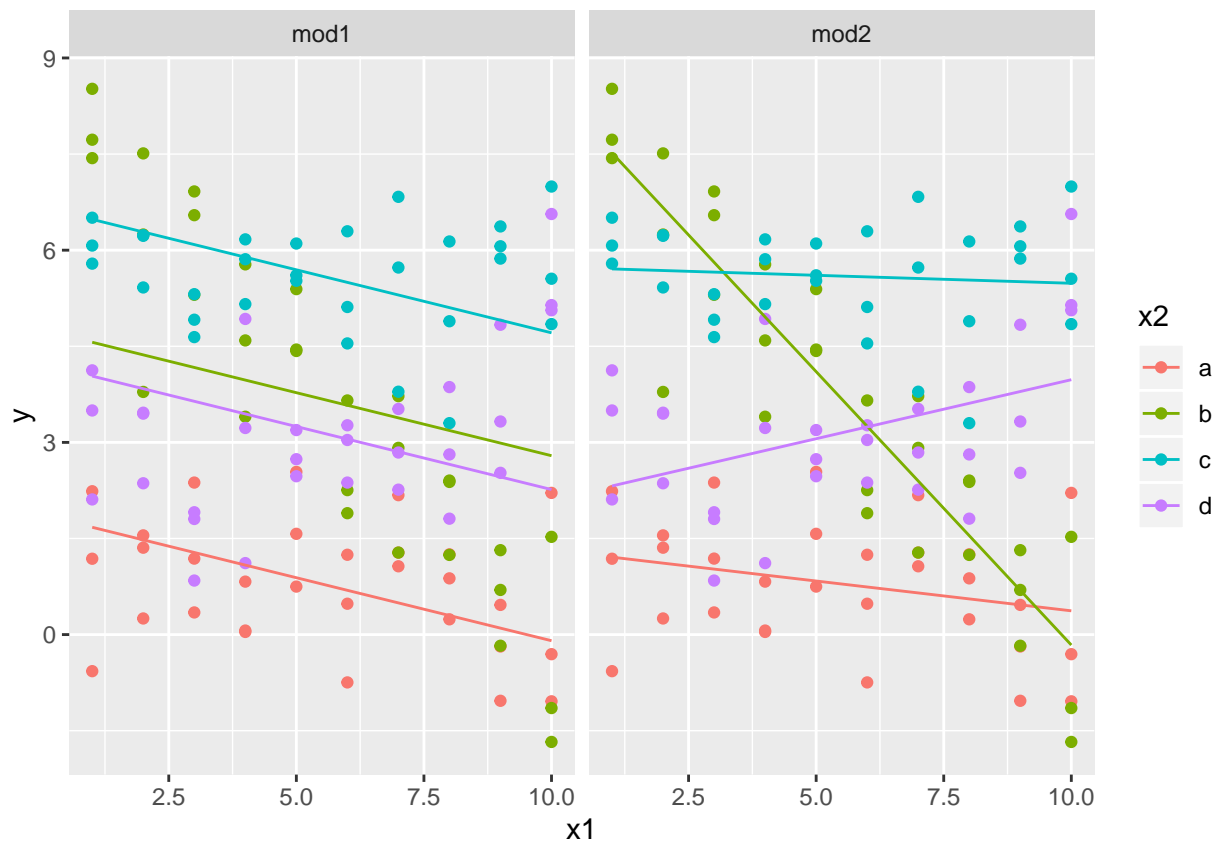
`gather_predictions()` is a function to add each prediction as a row.

I use their combination to create a grid to plot.

```
grid <- sim3 %>%  
  data_grid(x1, x2) %>%  
  gather_predictions(mod1, mod2)
```

Visualizing the models in one graph.

```
ggplot(sim3, aes(x1, y, colour = x2)) +  
  geom_point() +  
  geom_line(data = grid, aes(y = pred)) +  
  facet_wrap(~ model)
```



### Exercise 3:

In classes, we usually do not write codes that require heavy computations. However in real life, effective codes are of crucial importance. R is a very vector-oriented language. I suspect that running a `for` loop is slow compared to some sort of `apply` function. In this exercise, I intend to shed light on how much loops are slower in R.

First, I create a vector with 100 000 values. I want to create a new vector adding a uniform random number between 0 and 1 to the each and every value of the original vector. I add a random seed to ensure that we add the very same random numbers to the original vector.

```
numbers <- c(1:100000)
```

### Loops

This for loop iterates through the original vector appending a new element to the cumulating vector in each iteration.

```
set.seed(42)
start_time <- Sys.time()
vec <- c()
for (value in numbers) {
  vec <- c(vec, value + runif(1))
}
end_time <- Sys.time()
end_time - start_time
```

```
## Time difference of 13.11796 secs
```

### Apply-like functions

This is more like an R way of coding. I wrote a function that adds a random uniform number to a number. I applied this function to every element of my original vector.

```
set.seed(42)
start_time <- Sys.time()
random_maker <- function(x) {
  return (x + runif(1))
}

vec <- sapply(numbers, random_maker)
end_time <- Sys.time()
end_time - start_time
```

```
## Time difference of 0.2343984 secs
```

This is ridiculously faster than the for loop. R coders should use loops rarely as they are quite slow in this R environment, furthermore, theoretically not supported.