ex03: Simple regression

Kenji Sato
Kobe University*

June 21, 2017

1 Overview

Purpose

Become familiar with simple linear regression with R.

Instructions

In this assignment, you will

- clone the assignment repository and make a working branch (eg. `solution` branch);
- solve the problems in Section 3;
- write the solutions in `R/solution.R` and knit `solution.Rmd` the file;
- commit `solution.Rmd` and `solution.pdf`; and
- open a Pull Request.

2 Simple regression

Although this course does not dig deep into statistics and econometrics, you might want to have basic understandings of linear regressions.

In this exercise, we will use **tidyverse**. Run the following command.

```
library(tidyverse)
```

*Email: mail@kenjisato.jp

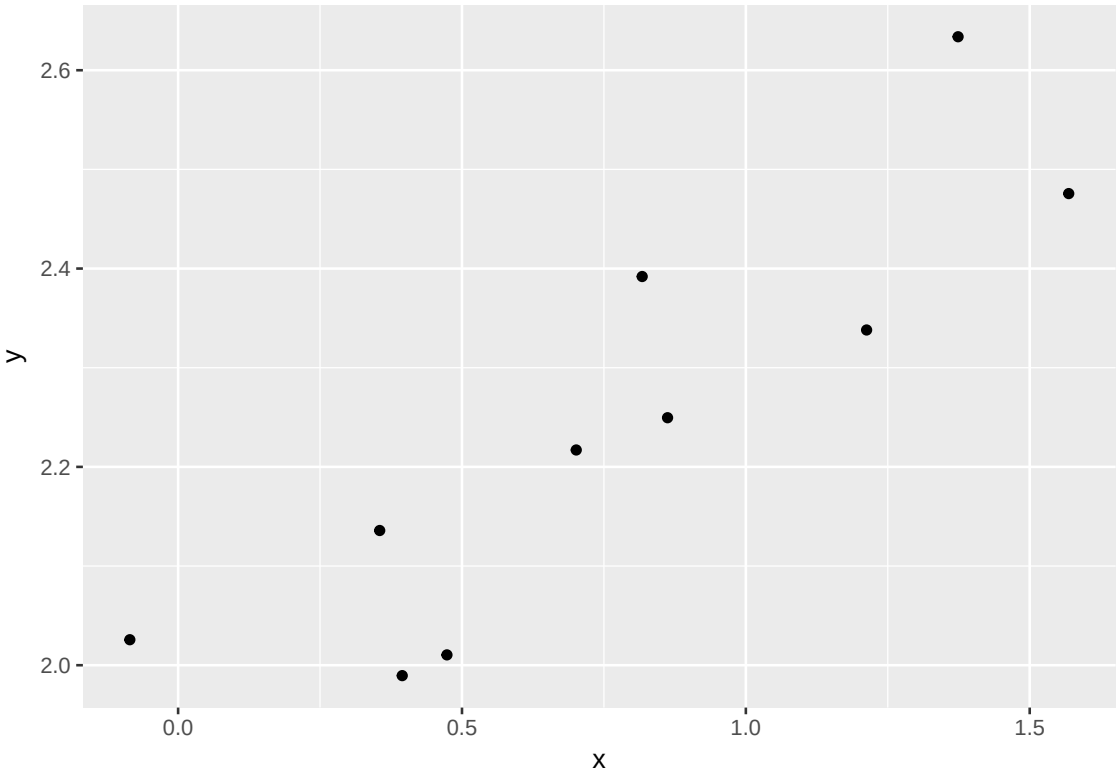
2.1 Purely linear case

The first data we play with is the following:

```
df_lin <-
  tibble(
    x = c(1.37379088091951, 0.862106871728239, 0.473489104395616,
          0.701262814146907, -0.0850552741258509, 1.56870212341502,
          0.817391973588289, 0.394768602709471, 1.21269855279069,
          0.35508065933832),
    y = c(2.63376468487512, 2.24957802290499, 2.0103939202368,
          2.21703232314625, 2.02570067955709, 2.4756136111241,
          2.39195731014444, 1.98947852908374, 2.33804041458729,
          2.1358293292897))
```

Any analysis bigins with a simple scatter plot.

```
(p_lin <- ggplot(df_lin, aes(x, y)) + geom_point())
```



Do you see a linear relation between x and y? You guessed correctly.

Usually, we do not know the data generating process but in this exercise I reveal how I created the data.

```r
set.seed(300)
(x <- rnorm(10))
```

```
##  [1]  1.37379088  0.86210687  0.47348910  0.70126281 -0.08505527
##  [6]  1.56870212  0.81739197  0.39476860  1.21269855  0.35508066
```

```r
e <- rnorm(10, 0, 0.1)
(y <- 2 + 0.3 * x + e)
```

```
##  [1] 2.633765 2.249578 2.010394 2.217032 2.025701 2.475614 2.391957
##  [8] 1.989479 2.338040 2.135829
```

The data was generated by the following rule:

$$y = 0.3x + e, \tag{1}$$

where the error term, e, is drawn from the normal distribution with mean = 0 and standard deviation = 0.1.

In a typical data analysis task, we want to reconstruct (1) from given data. More precisely, we start from assuming that the data is generated by a linear relationship

$$y = \beta_0 + \beta_1 x$$

and then determine what values of coefficients (β_0, β_1) best explain the data.

If we could know the coefficient 0.3, we can interpret predict that an increase in x by 1 would increase y by 0.3.

This problem of linear regression is easily solved with R by `lm()` function.[1]

```r
(fit_lin <- lm(y ~ x, data = df_lin))
```

```
##
## Call:
## lm(formula = y ~ x, data = df_lin)
##
## Coefficients:
## (Intercept)            x
##      1.9629       0.3699
```

Expressions like `y ~ x` are called model formulas.

- Use `y ~ x` to regress `y` on `x`,

[1] That it is easy does not mean that you can skip serious study of econometrics. You must understand, at least in the long run, what is going on under the hood.

- Use `z ~ x + y` to regress `z` on `x` and `y`, etc.

R (or its statistical procedure) estimates that the above data is generated by

$$y = 1.9629 + 0.36990x$$

This answer is not particularly appealing but considering the small number of data, we can say that linear regression is doing a good job.

You can study the statistical properties in more detail with `summary()` function.

```
summary(fit_lin)
```

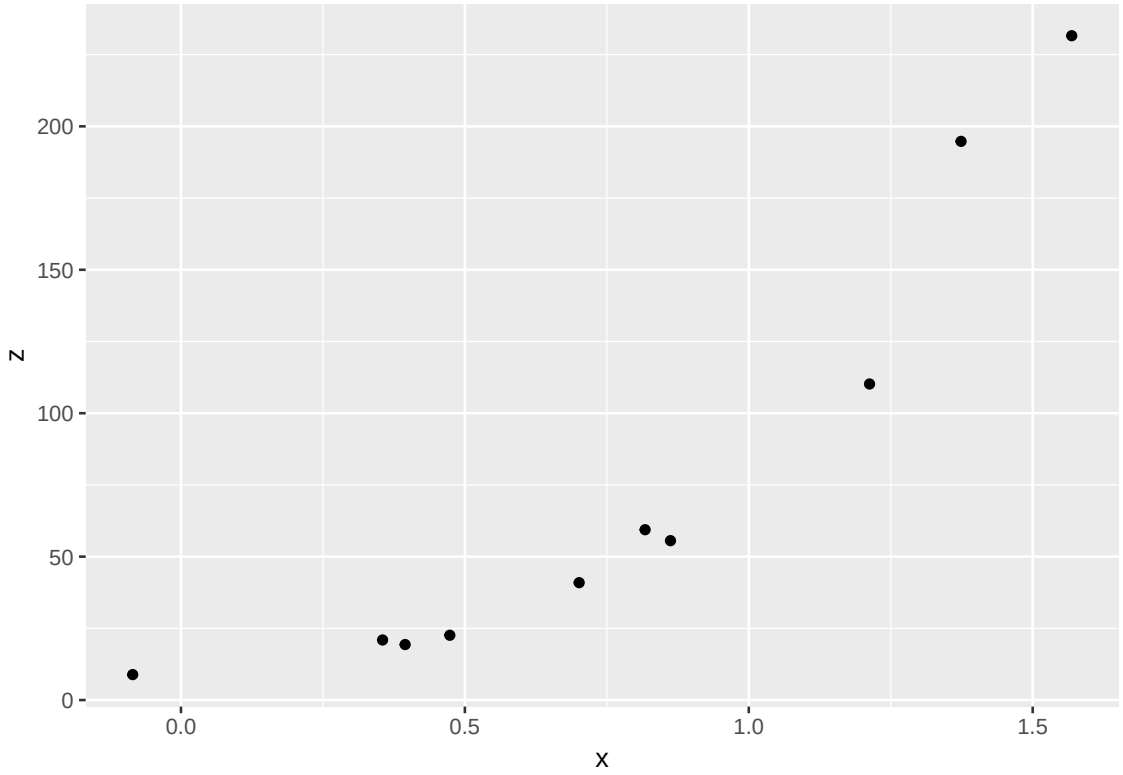
```
##
## Call:
## lm(formula = y ~ x, data = df_lin)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12762 -0.07193 -0.01871  0.08112  0.16273
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.96287    0.06447  30.445 1.47e-09 ***
## x            0.36990    0.07104   5.207 0.000816 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1088 on 8 degrees of freedom
## Multiple R-squared:  0.7721, Adjusted R-squared:  0.7436
## F-statistic: 27.11 on 1 and 8 DF,  p-value: 0.0008159
```

2.2 Log linear case

Besides linear case, we often encounter log linear case. Observe the following data.

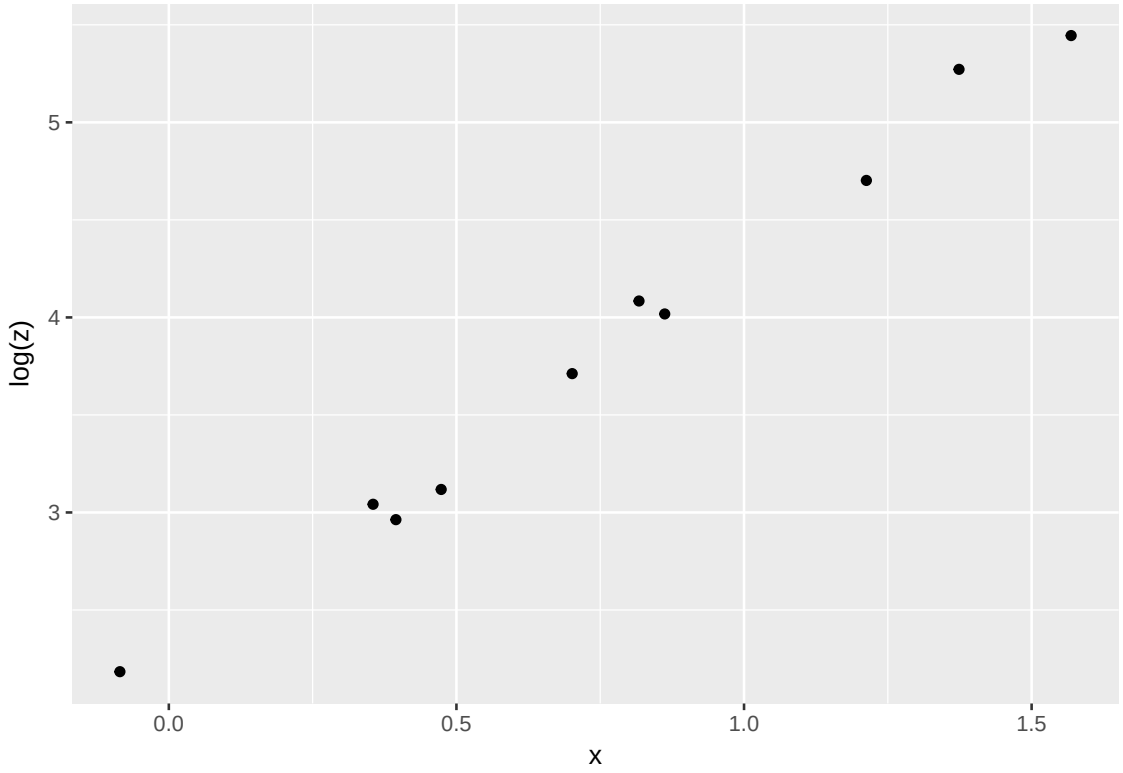
```
z <- 10 * exp(2 * x + e)
df_loglin <- tibble(x = x, z = z)

ggplot(df_loglin, aes(x, z)) + geom_point()
```



This data doesn't seem to follow a linear law but if taken log, linearity comes out.

```
ggplot(df_loglin, aes(x, log(z))) + geom_point()
```



Mathematically, if the data is generated by

$$z = ae^{bx+e},$$

by taking log of both size we obtain

$$\ln z = \ln a + bx + e.$$

These coefficients can easily be estimated with `lm()` function.

```
lm(log(z) ~ x, data = df_loglin)
```

```
##
## Call:
## lm(formula = log(z) ~ x, data = df_loglin)
##
## Coefficients:
## (Intercept)            x
##       2.265        2.070
```

The coefficients for `x` is estimated at 2.070. This means that an increase of x by 1 will cause an increase of z by $100 \times b$ percent. You can observe this percentage rule with

$$b = \frac{d(\ln z)}{dx} = \frac{1}{z}\frac{dz}{dx}$$

or

$$\frac{dz}{z} = b \cdot dx$$

2.3 Other cases

We have seen two cases for which we applied the following formulas.

- `y ~ x`
- `log(y) ~ x`

As you might expect, the following formulas are also valid. Construct simulation data yourself and experiment with these specifications.

- `y ~ log(x)`
- `log(y) ~ log(x)`

3 Problems

Conduct simple linear regression exercises with the following datasets, which are bundled with R:

- `women`
- `cars`
- `pressure`

Interpret each result.

Write your report in `solution.Rmd`, knit it to produce `solution.pdf` and commit the two files.