Linear Regression - Intuition

Pouria

1/24/2022

Contents

Objective

The objective of this project is to practice the use of closed-from representation of a simple univariate linear regression. We aim to understand the underlying structure of what linear regression does under the hood and how it does it.

To this end, we use simulated data. What that means is we start with a given map of a set of values representing variable `X` to those representing the variable `y`. This way, we already know what the parameters of the map between `X` and `y` should be. From there, we simulate the data based off of the map equation and try to recover the true given equation using linear regression.

Part 1: Closed-Form Representation

In this first exercise, we will simulate different datasets with the following given map.

$$y = 2 + 3X + \epsilon,$$

with X randomly generated via a uniform distribution from 0 to 10, and ϵ generated via a normal (Gaussian) distribution of 0 mean and variance 1 (which we note, $\epsilon \sim N(\mu, \sigma^2)$, where $\mu = 0$ and $\sigma = 1$). Our goals and questions to address in this part are as follows:

a. Generate 10 "experiments" with 5 observations each. Compute the slopes and the intercepts using the closed-form equations shown below, check that the values are the same as given by the `lm()` function in R , and plot the 10 different lines. Also plot in bold the "true" line.

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2},$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

b. Repeat 10 experiments, but with 20 observations each. Plot the 10 different lines. What do you conclude?

c. For each 5 and 20 observations, use the formulas below to compute the SE for the slope.

- Plot the SEs in both conditions.
- Is this this difference statistically significant?
- Why is the SE smaller for 20 observations?

Equations for computing SE:

$$SE(\hat{\beta}_0)^2 = \sigma^2 [\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^n (x_i - \bar{x})^2}],$$
$$SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

Functions

The following is the closed-form univariate regression formula

```r
Eq3.4 <- function(X, y){

  # Equations to estimate beta0 and beta1
  beta1 = sum((X-mean(X))*(y-mean(y))) / sum((X-mean(X))^2)
  beta0 = mean(y) - beta1*mean(X)

  return(list(beta0, beta1))
}
```

In the following, I will code up the function for finding the fits and stacking them in a structure for any given number of observations and experiments:

```r
Experiment <- function(N.Obs, N.Exp, X.min, X.max, epsilon.mu, epsilon.sd){

  beta0 <- beta1 <- 0
  beta0_lm <- beta1_lm <- 0
  SE_beta1 <- 0

  for (i in 1:N.Exp){
    X = runif(N.Obs, min = X.min, max = X.max)
    epsilon = rnorm(N.Obs, mean = epsilon.mu, sd = epsilon.sd)
    y = 2 + 3*X + epsilon

#    n.exp = i*rep(N.Obs, 1)

#    data.points <- rbind(data.points, data.frame(n.exp, X, y))

    coeff <- Eq3.4(X, y)
    beta0[i] <- coeff[[1]]
    beta1[i] <- coeff[[2]]

    SE_beta1[i] <- sqrt(epsilon.sd^2 / sum((X-mean(X))^2) )

    fit.lm <- lm(y~X)
    beta0_lm[i] <- fit.lm$coefficients[1]
    beta1_lm[i] <- fit.lm$coefficients[2]
  }


  return(data.frame(beta0, beta1,beta0_lm, beta1_lm, SE_beta1))
}
```

The following function uses `ggplot` functionality to visualize the results of the linear regression.

```r
Visualize <- function(Coeff.df){
  # data.points = data.frame(X, y)
  ggplot() +
    geom_abline(data = Coeff.df, aes(slope=beta1 , intercept=beta0), col=4) +
    geom_abline(aes(slope = 3, intercept = 2), size=1.5, col=2) +
    scale_x_continuous(name="X", limits=c(-2,2)) +
    scale_y_continuous(name="y", limits=c(-10,10))
    # geom_point(data.points, aes(X,y, col = n.exp))

}
```

Part 1.a

Set up the parameters

```r
N.Exp = 10
N.Obs = 5
X.min = 0
X.max = 10
epsilon.mu = 0
epsilon.sd = 1
```

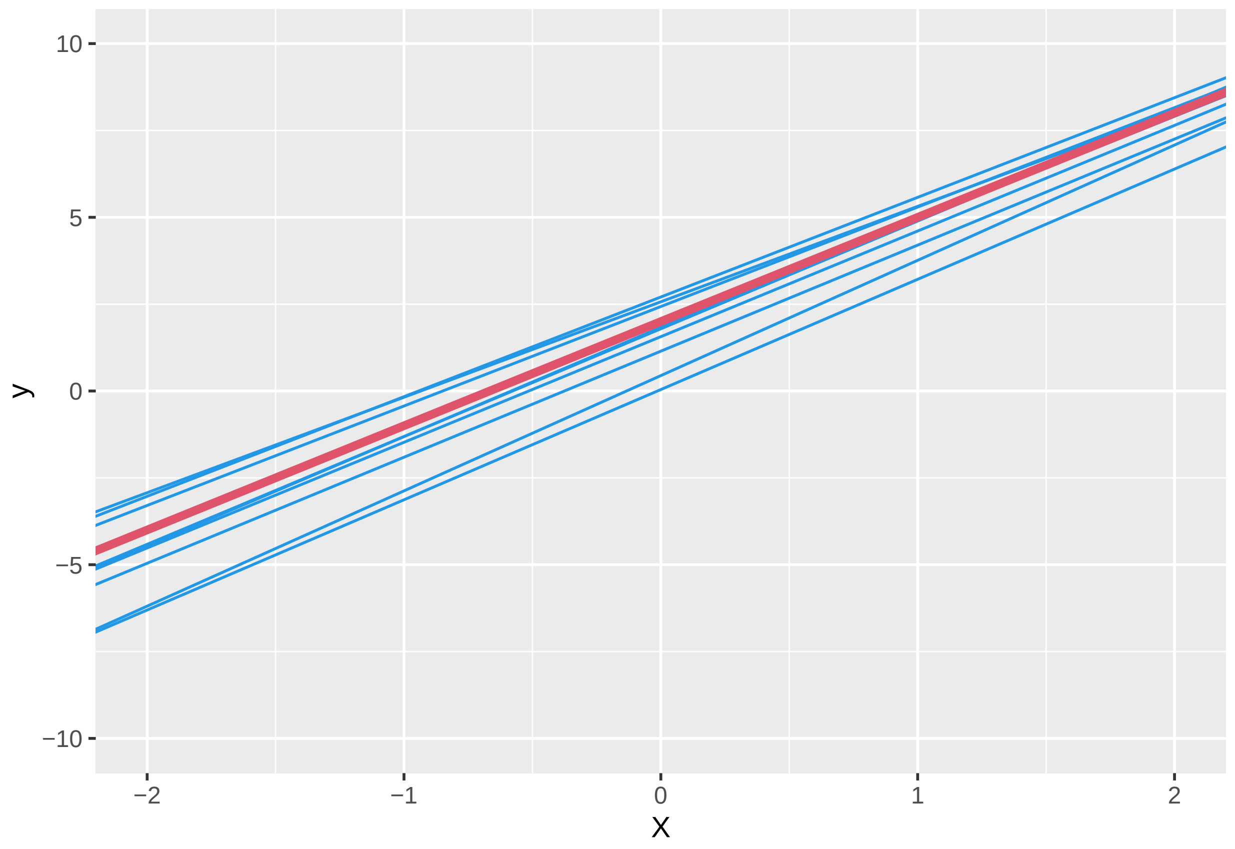
Coefficients for 5 observations from 10 experiments

```r
Coeff.df <- Experiment(N.Obs, N.Exp, X.min, X.max, epsilon.mu, epsilon.sd)

Coeff.df
```

```
##         beta0    beta1   beta0_lm beta1_lm  SE_beta1
## 1  1.56386277 3.040981 1.56386277 3.040981 0.2223013
## 2  1.14664587 3.052570 1.14664587 3.052570 0.1608245
## 3  2.43087530 2.862905 2.43087530 2.862905 0.2451380
## 4  0.04173686 3.173441 0.04173686 3.173441 0.1930411
## 5  0.44306329 3.318226 0.44306329 3.318226 0.1765996
## 6  2.70574896 2.868510 2.70574896 2.868510 0.1285646
## 7  1.83085544 3.142933 1.83085544 3.142933 0.1719720
## 8  2.56776817 2.747558 2.56776817 2.747558 0.1497124
## 9  1.79055497 3.101656 1.79055497 3.101656 0.2201951
## 10 1.97767126 2.964289 1.97767126 2.964289 0.1386754
```

Visualize for 5 observations from 10 experiments

```r
Visualize(Coeff.df)
```



Part 1.b

Set up the parameters according to 1.b now:

```
N.Exp = 10
N.Obs = 20
X.min = 0
X.max = 10
epsilon.mu = 0
epsilon.sd = 1
```

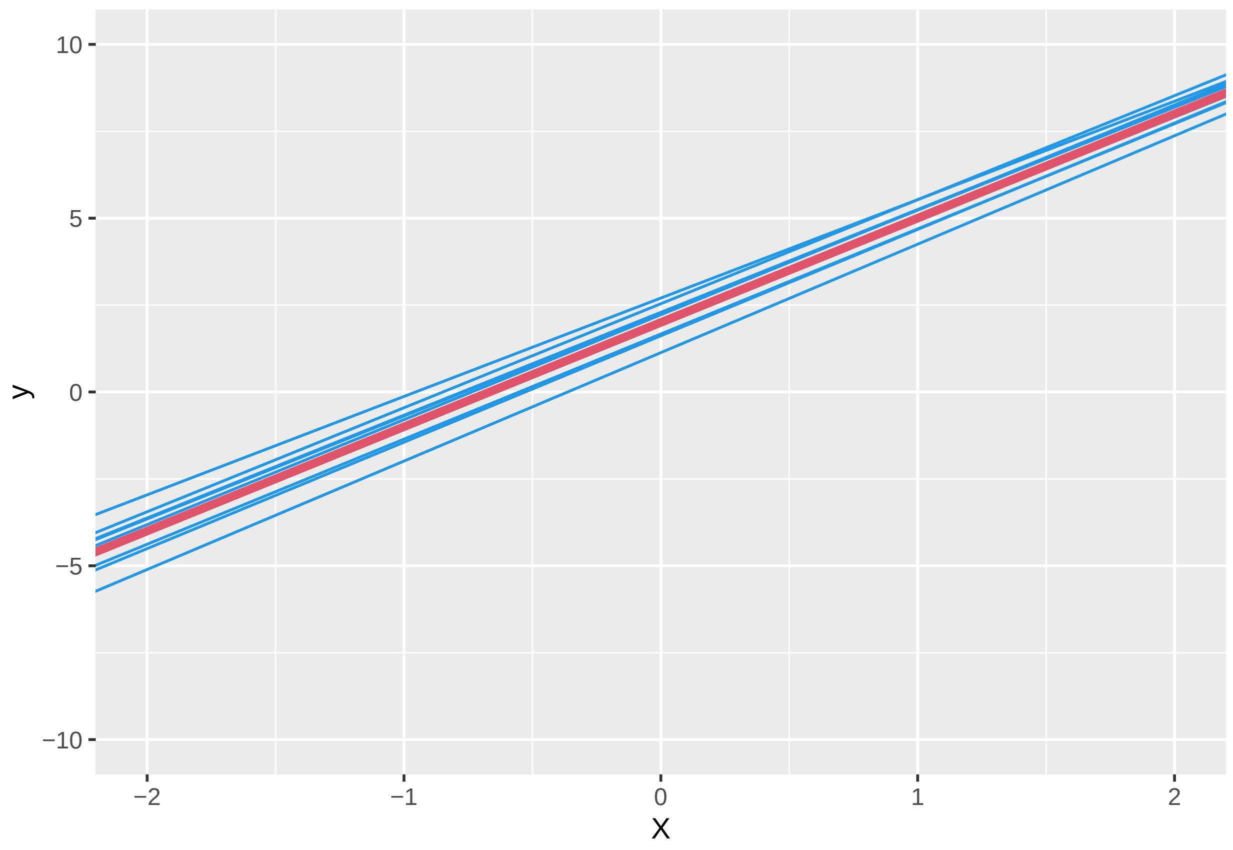
Coefficients for 20 observations from 10 experiments;

```
Coeff.df <- Experiment(N.Obs, N.Exp, X.min, X.max, epsilon.mu, epsilon.sd)
Coeff.df
```

```
##       beta0    beta1 beta0_lm beta1_lm   SE_beta1
## 1  1.131753 3.119721 1.131753 3.119721 0.07986821
## 2  2.290789 2.957960 2.290789 2.957960 0.09091350
## 3  1.962219 2.999278 1.962219 2.999278 0.07696685
## 4  1.616000 3.060789 1.616000 3.060789 0.09142783
## 5  1.953258 2.976565 1.953258 2.976565 0.09376113
## 6  2.268626 2.961368 2.268626 2.961368 0.08043199
## 7  1.668107 3.024440 1.668107 3.024440 0.06865076
## 8  2.538956 2.992570 2.538956 2.992570 0.07477152
## 9  2.220970 3.017968 2.220970 3.017968 0.07623758
## 10 2.702189 2.830943 2.702189 2.830943 0.08219383
```

Now let's visualize for 20 observations from 10 experiments

```
Visualize(Coeff.df)
```



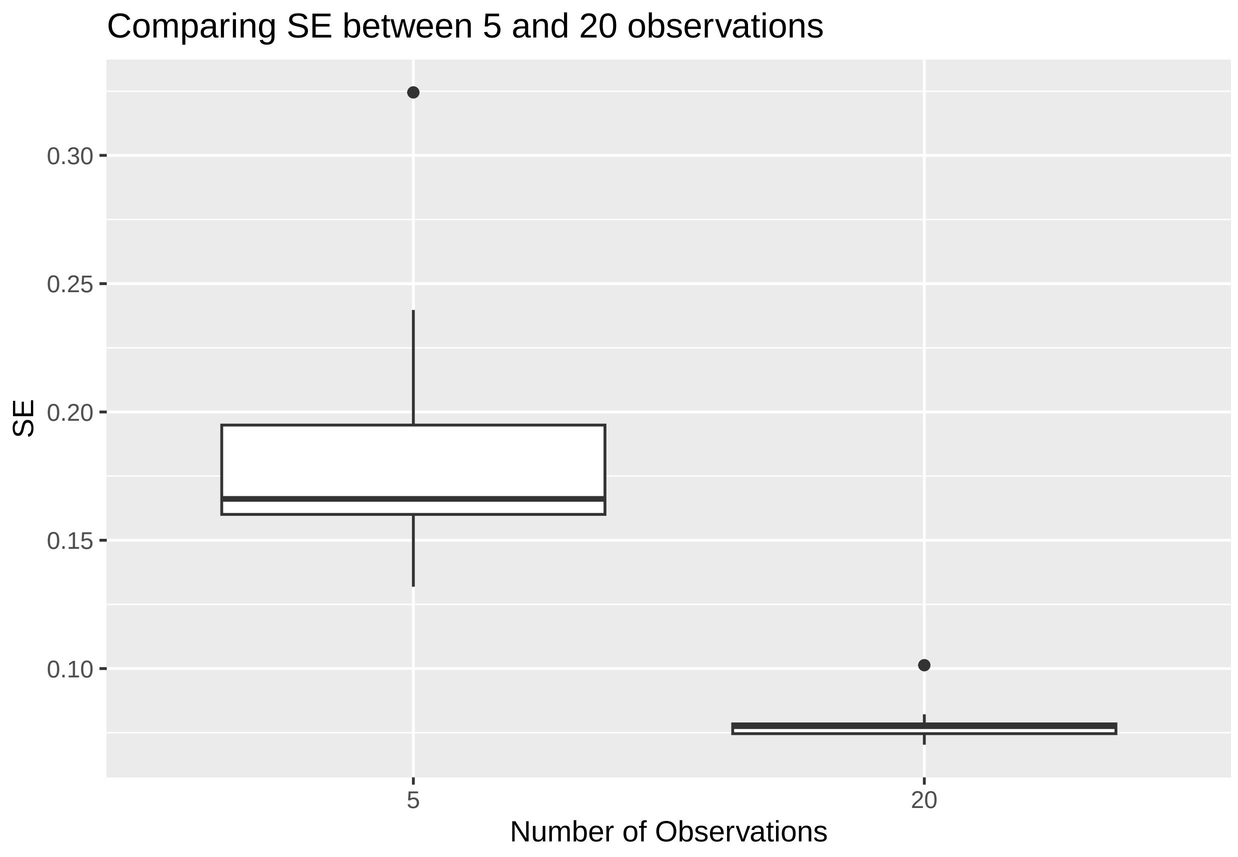
Part 1.c

No, for each of the cases with 5 and 20 observations, we use the formula below to compute the SE for the slope.

$$SE(\hat{\beta}_0)^2 = \sigma^2[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}],$$
$$SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Let's plot the SEs from both conditions.

```r
ggplot(SE.df, aes(x = N.Obs, y = SE)) +
geom_boxplot() +
scale_x_discrete(limits = rev) +
labs(
title = "Comparing SE between 5 and 20 observations",
x = 'Number of Observations',
y = 'SE'
)
```



Let’s see if this difference is statistically significant:

```r
t.test(SE.5, SE.20, alternative = "two.sided", var.equal = F)
```

```
##
##  Welch Two Sample t-test
##
## data:  SE.5 and SE.20
## t = 5.8755, df = 9.3977, p-value = 0.0001994
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.06659275 0.14910852
## sample estimates:
##  mean of x  mean of y
## 0.18681810 0.07896747
```

We can see that the SE computed for 20 observations is significantly smaller than the case with 5 observations.

You may ask why. Well, intuitively, more observations will add more reliability to your estimation of the fit parameters. The more observations you have, the more confident you can be in your obtained fit, and the smaller you standard error (SE) would be. Moreover, technically, looking closely at the SE equations above, you can see that the number of terms in the SE equation denominator is proportionate with the number of observations.

Part 2: Iterative Search

Now we will try to estimate coefficients of a linear regression model via systematic simulations

a. We now use the following model

$$y = 7 + 0.05 * X + \sigma,$$

with $\sigma \sim N(0,1)$)

b. Plot the regression line with 100 observations
c. Now using 2 for-loops over the parameters with about 100 steps for each parameter, generate a figure like figure 3.2A in the book (use similar parameter ranges).
d. Find the minimum of RSS curve using `min()`? Plot the minimum on the figure. How does this compare with the parameters given by the `lm()`?
e. Now repeat with about 5 steps for each parameter. Generate the figure again. How does this compare with the parameters given by the `lm()`? What do you conclude?
f. *Bonus*: generate 3D plots like figure 3.2B

Part 2.a

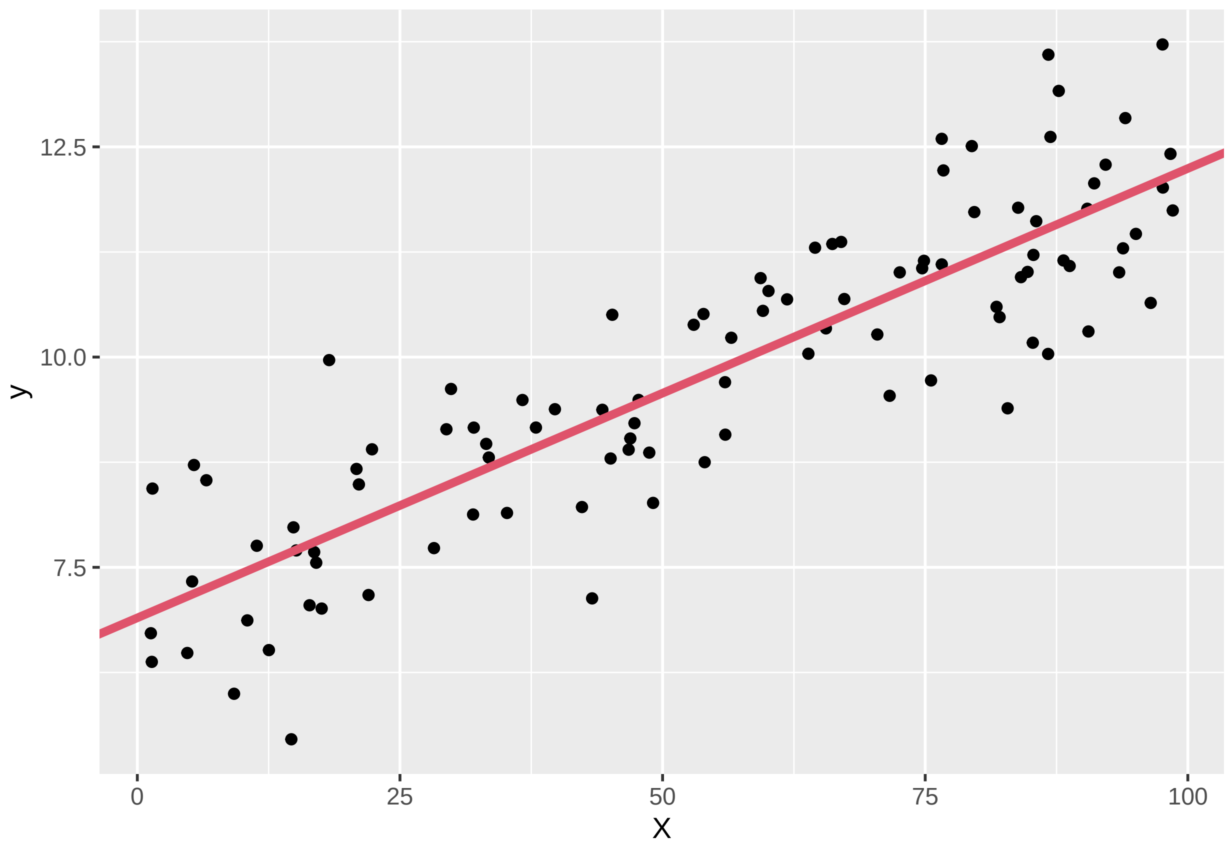
```r
RSS_func <- function(b0, b1, X, y){
  y_hat = b0 + b1*X
  RSS = sum( ( y-y_hat )^2 )

  return(RSS)
}
```

```r
i <- 0

N.Obs = 100
X = runif(N.Obs, min = 0, max = 100)
epsilon = rnorm(N.Obs, mean = 0, sd = 1)
#y = 2 + 3*X + epsilon
y = 7.0 + .05*X + epsilon

beta1 <- beta0 <- 0
RSS <- 0

for (bet0 in seq(5,9,length.out=100)){
  for (bet1 in seq(.03,.07,length.out=100)){
    i <- i+1
    beta0[i] <- bet0
    beta1[i] <- bet1
    RSS[i]   <- RSS_func(bet0, bet1, X, y)
  }
}
```

```
beta0.opt <- beta0[RSS==min(RSS)]
beta1.opt <- beta1[RSS==min(RSS)]
```

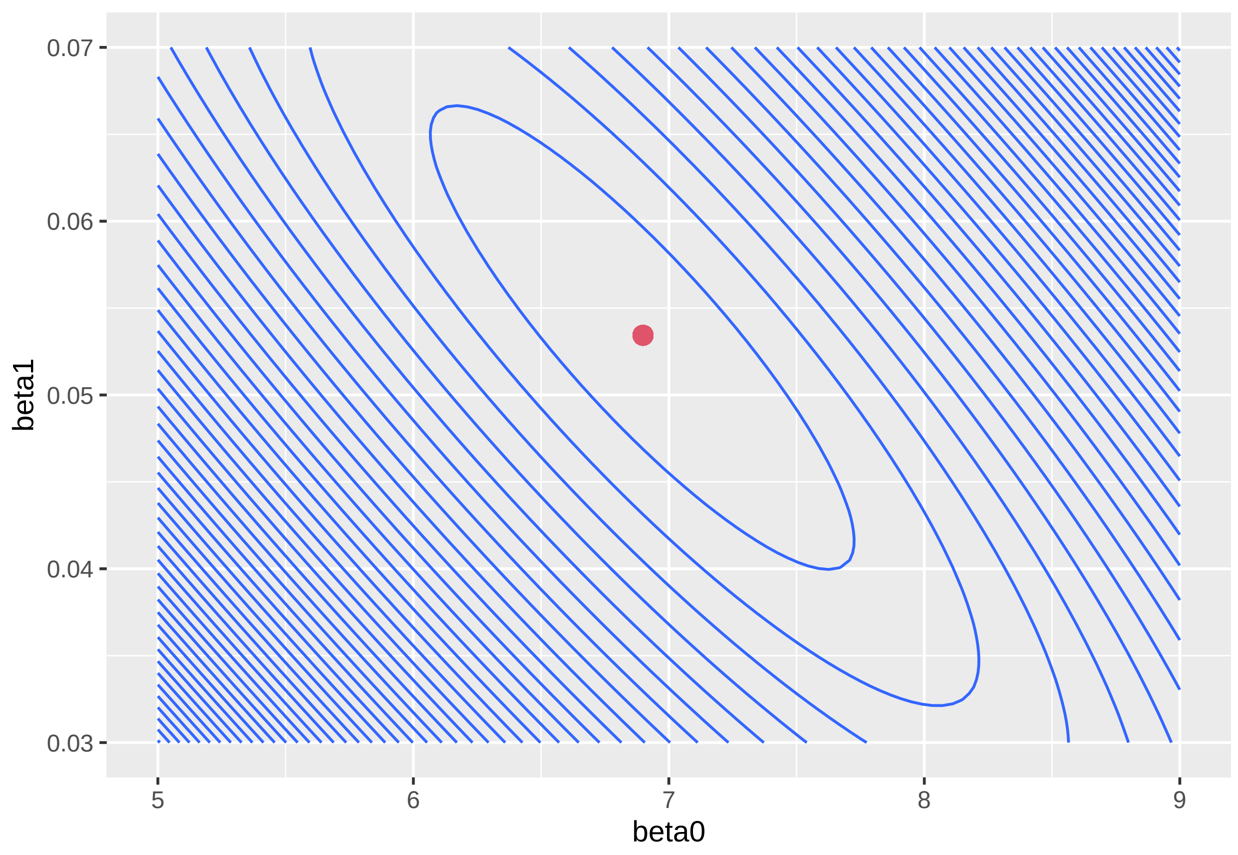
Part 2.b

```
datapoints = data.frame(X,y)
ggplot(datapoints, aes(X, y))+
  geom_point() +
  geom_abline(aes(slope = beta1.opt, intercept=beta0.opt), color = 2, size = 1.5)
```



Part 2.c

```
df.RSS = data.frame(beta0, beta1, RSS)
var.opt = data.frame(beta0.opt, beta1.opt)

ggplot(df.RSS, aes(beta0, beta1, z=RSS)) +
  geom_contour(bins = 50) +
  geom_point(aes(x = beta0.opt, y = beta1.opt), color=2, size=3)
```



Part 2.d

```
fit = lm(y~X)

beta0.lm = fit$coefficients[1]
beta1.lm = fit$coefficients[2]

data.frame(beta0.opt, beta1.opt, beta0.lm, beta1.lm)
```

```
##             beta0.opt  beta1.opt beta0.lm   beta1.lm
## (Intercept)   6.89899 0.05343434 6.895859 0.05330652
```

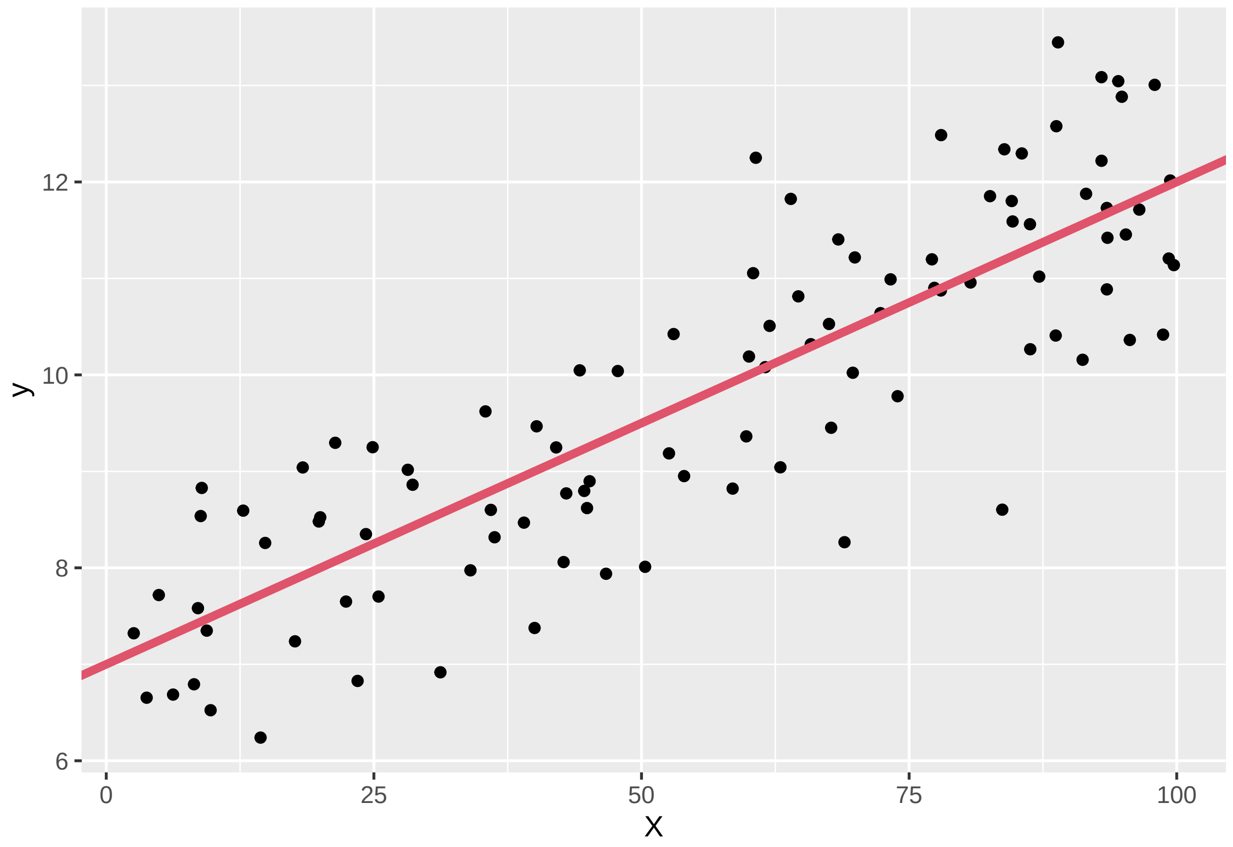
```
i <- 0

N.Obs = 100
X = runif(N.Obs, min = 0, max = 100)
epsilon = rnorm(N.Obs, mean = 0, sd = 1)
#y = 2 + 3*X + epsilon
y = 7.0 + .05*X + epsilon

beta1 <- beta0 <- 0
RSS <- 0
```

```r
for (bet0 in seq(5,9,length.out=5)){
  for (bet1 in seq(.03,.07,length.out=5)){
    i <- i+1
    beta0[i] <- bet0
    beta1[i] <- bet1
    RSS[i]   <- RSS_func(bet0, bet1, X, y)
  }
}

beta0.opt <- beta0[RSS==min(RSS)]
beta1.opt <- beta1[RSS==min(RSS)]
```

Part 2.e:

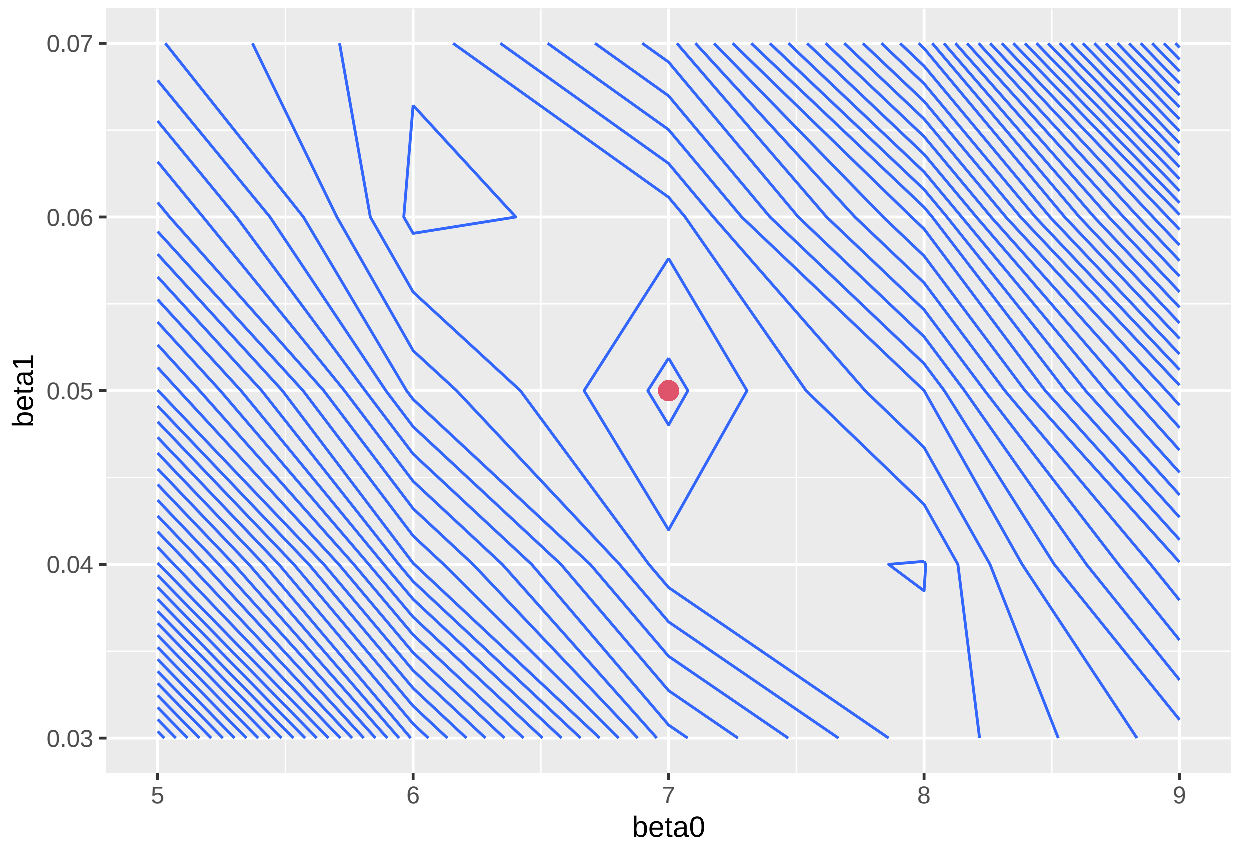
Let's visualize the regression

```r
datapoints = data.frame(X,y)
ggplot(datapoints, aes(X, y))+
  geom_point() +
  geom_abline(aes(slope = beta1.opt, intercept=beta0.opt), color = 2, size = 1.5)
```



Now we will take a look at the contour plot

```r
df.RSS = data.frame(beta0, beta1, RSS)
var.opt = data.frame(beta0.opt, beta1.opt)

ggplot(df.RSS, aes(beta0, beta1, z=RSS)) +
  geom_contour(bins = 50) +
  geom_point(aes(x = beta0.opt, y = beta1.opt), color=2, size=3)
```



Now, let's compare the coefficients from the two methods

```r
fit = lm(y~X)

beta0.lm = fit$coefficients[1]
beta1.lm = fit$coefficients[2]

data.frame(beta0.opt, beta1.opt, beta0.lm, beta1.lm)
```

```
##             beta0.opt beta1.opt beta0.lm    beta1.lm
## (Intercept)         7      0.05 6.946931 0.05060663
```

Part 3: Gradient Descent

Now we will try to estimate the parameters via a method called "gradient descent". Given a random starting point, we will "descend" along the steepest gradient in parameter space until we converge to the minimum.

$$\hat{y} = X\theta$$

$$L = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

$$L = \frac{1}{2m}(\mathbf{y} - X\theta)^T(\mathbf{y} - X\theta)$$

...