Chapter 3 Problem 13

Andira Putri

In this exercise, you will create some simulated data and will fit simple linear regression models to it.

*Note - When working on problems that uses randomly-generated data sets, set.seed() reproduces the exact same set of random numbers. This ensures consistent results.

a. Using the rnorm() function, create a vector x containing 100 observations drawn from a N(0,1) distribution. This represents a feature, X.

```
set.seed(1)
x=rnorm(100,0,1)
```

b. Using the rnorm() function, create a vector eps containing 100 observations drawn from a N(0,0.25) distribution.

```
eps=rnorm(100,0,0.25)
```

c. Using x and eps, generate a vector y according to the model Y=-1+0.5X+??. What is the length of vector y? What are the values of the coefficients in the model?

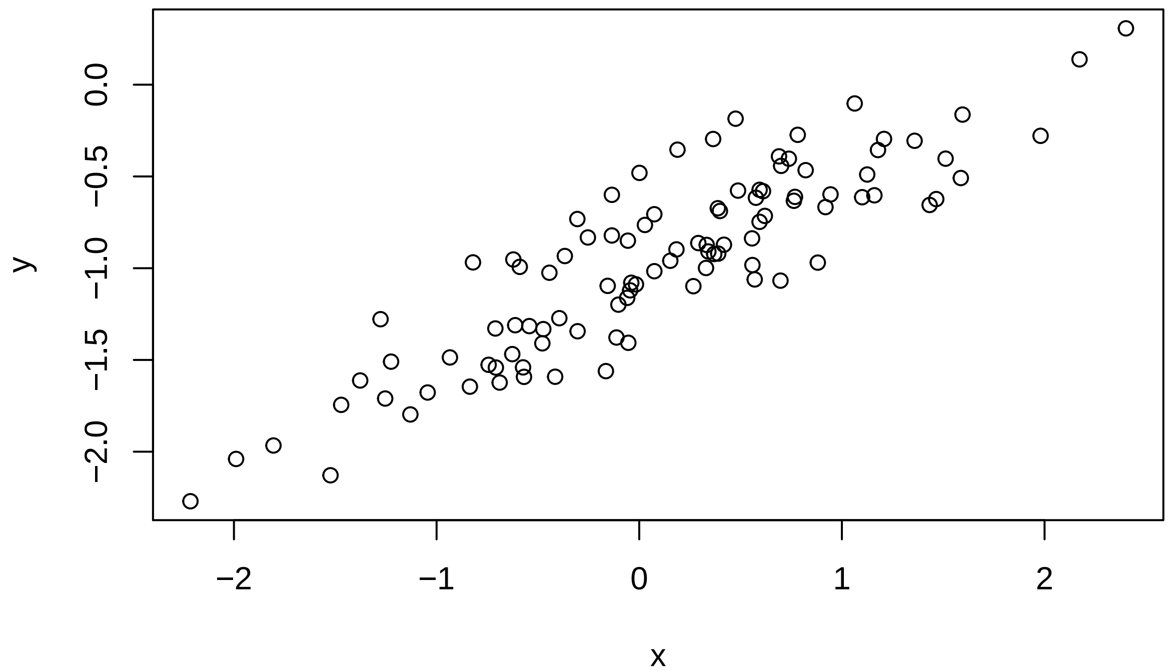
```
y=0.5*x-1+eps
length(y) #length of y vector
```

```
## [1] 100
```

$\beta_0 = -1$ and $\beta_1 = 0.5$.

d. Create a scatterplot displaying the relationship between x and y.

```
plot(x,y)
```



e. Fit a least squares model to predict y using x. Comment on the model obtained. How do the estimated coefficients compare to those in part c?

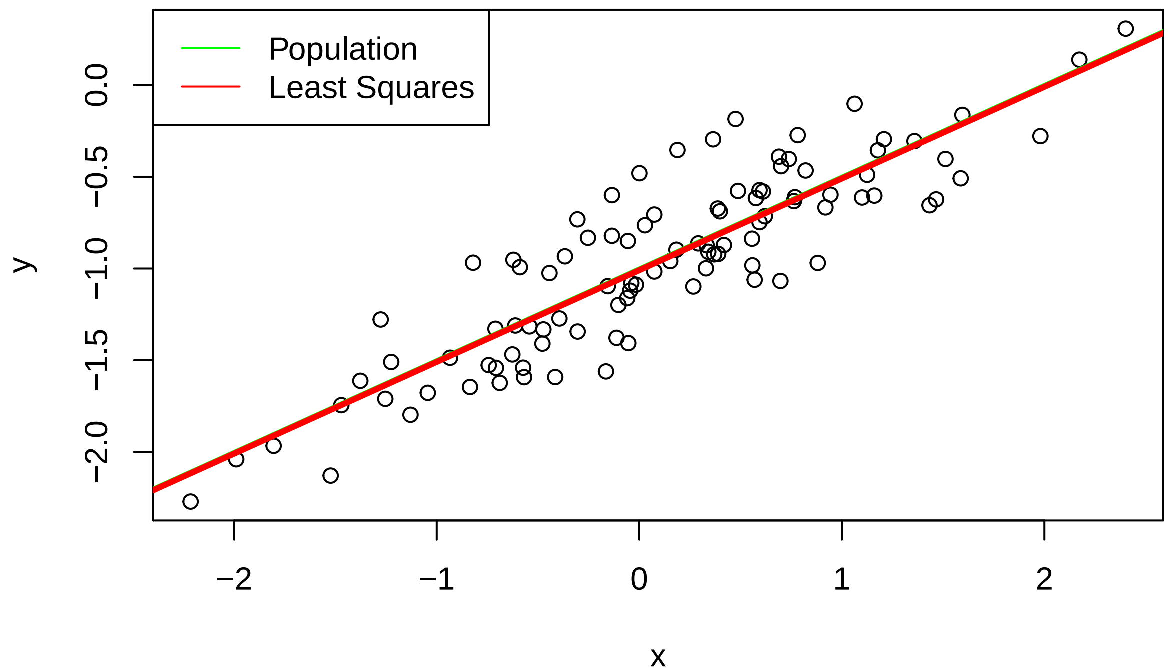
```
lm.fit=lm(y~x)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.46921 -0.15344 -0.03487  0.13485  0.58654
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.00942    0.02425  -41.63   <2e-16 ***
## x            0.49973    0.02693   18.56   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2407 on 98 degrees of freedom
## Multiple R-squared:  0.7784, Adjusted R-squared:  0.7762
## F-statistic: 344.3 on 1 and 98 DF,  p-value: < 2.2e-16
```

The model suggests a strong relationship between x and y. The estimated coefficients are quite close to the actual β_1 and β_0 values in part c.

f. Display the least squares and population regression line on the scatterplot in part d.

```r
plot(x,y)
abline(-1,0.5,col="green",lwd=2) #population
abline(lm.fit,col="red",lwd=3) #least squares
legend("topleft",c("Population","Least Squares"),col=c("green","red"),lty=c(1,1))
```



g. Now fit a polynomial regression model that predicts y using x and x^2. Is there evidence that the quadratic term improves the model fit?

```r
lm.fitpoly=lm(y~x+I(x^2))
summary(lm.fitpoly)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.4913 -0.1563 -0.0322  0.1451  0.5675
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.98582    0.02941 -33.516   <2e-16 ***
## x            0.50429    0.02700  18.680   <2e-16 ***
## I(x^2)      -0.02973    0.02119  -1.403    0.164
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2395 on 97 degrees of freedom
## Multiple R-squared:  0.7828, Adjusted R-squared:  0.7784
## F-statistic: 174.8 on 2 and 97 DF,  p-value: < 2.2e-16
```

There is no evidence suggesting that the quadratic term improves the model fit, given its high p-value