Question 13:

In this exercise you will create some simulated data and will fit simple linear regression models to it. Make sure to use set.seed(1) prior to starting part (a) to ensure consistent result

(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)
```

(b) Using the rnorm() function, create a vector, eps, containing 100 observations drawn from a N (0, 0.25) distribution—a normal distribution with mean zero and variance 0.25.

```
eps <- rnorm(100, 0, sqrt(0.25))
```

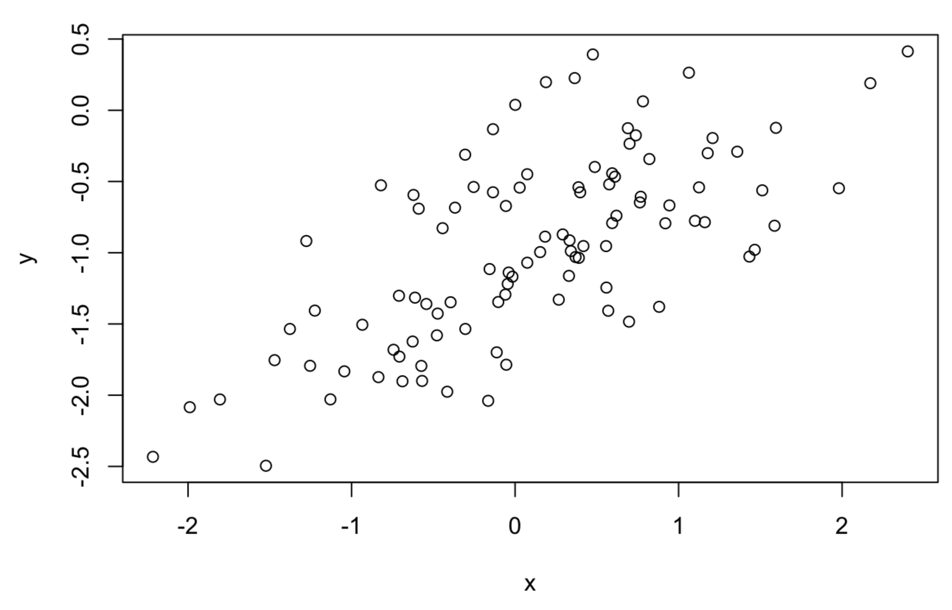
(c) Using x and eps, generate a vector y according to the model Y = −1 + 0.5X + ε. What is the length of the vector y? What are the values of β_0 and β_1 in this linear model?

```
y = -1 + (0.5 * x) + eps
```

The length of vector y is length of 100. The values of B0 is -1, and B1 is 0.5.

(d) Create a scatterplot displaying the relationship between x and y. Comment on what you observe.

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



From this plot, we observe a weak, positive, linear relationship between x and y.

(e) Fit a least squares linear model to predict y using x . Comment on the model obtained. How do $\hat\beta_0$ and $\hat\beta_1$ compare to β_0 and β_1 ?

```
lm.fit <-  lm(y ~ x)
summary(lm.fit)
```

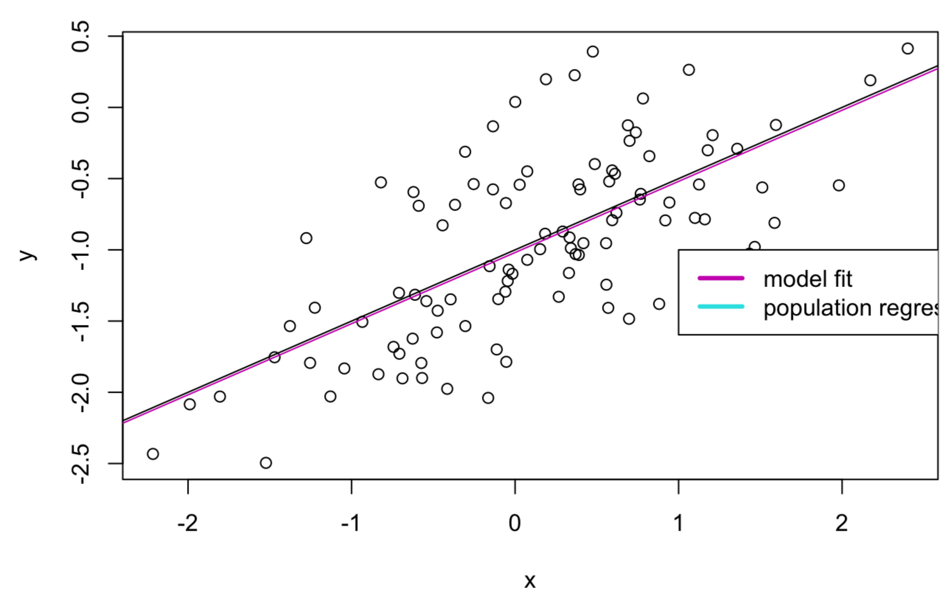
```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.93842 -0.30688 -0.06975  0.26970  1.17309
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.01885    0.04849 -21.010  < 2e-16 ***
## x            0.49947    0.05386   9.273 4.58e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4814 on 98 degrees of freedom
## Multiple R-squared:  0.4674, Adjusted R-squared:  0.4619
## F-statistic: 85.99 on 1 and 98 DF,  p-value: 4.583e-15
```

A simple linear regression was used to test if x significantly predicted y. The fitted regression model was: y = -1.01885 + 5.2503(x). The overall regression was statistically significant (R^2 = 0.47, F(1, 98)= 85.99, p<0.005). It was found that x significantly predicted y (B=0.49947, p<0.005).

Thus, the linear regression model closely fits the true coefficient values.

(f) Display the least squares line on the scatterplot obtained in (d). Draw the population regression line on the plot, in a different color. Use the legend() command to create an appropriate legend.

```
plot(x, y)
abline(lm.fit, lwd = 1, col = 6)
abline(-1, 0.5, lwd = 1, col = 1)
legend(-1, legend = c("model fit", "population regression"), col = 6:1, lwd = 3)
```



(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? Explain your answer.

```
lm.fit_sq <- lm(y ~  x + I(x^2))
summary(lm.fit_sq)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98252 -0.31270 -0.06441  0.29014  1.13500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.97164    0.05883 -16.517  < 2e-16 ***
## x            0.50858    0.05399   9.420  2.4e-15 ***
## I(x^2)      -0.05946    0.04238  -1.403    0.164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.479 on 97 degrees of freedom
## Multiple R-squared:  0.4779, Adjusted R-squared:  0.4672
## F-statistic:  44.4 on 2 and 97 DF,  p-value: 2.038e-14
```

There is a slight increase in model fit on the training data in the polynomial proven by the slight increase in the adjusted R^2 value.

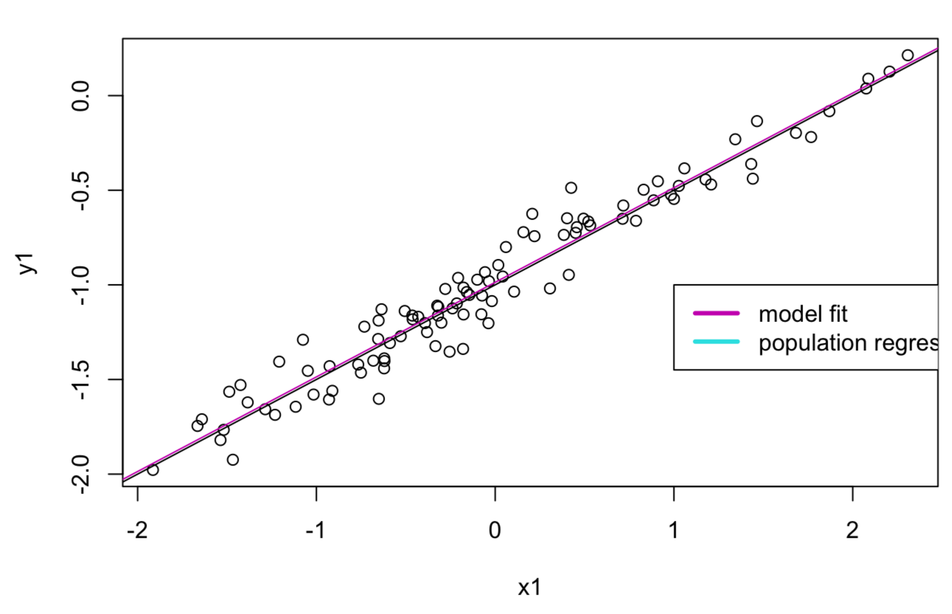
Interestingly enough, the model output suggests the y and x^2 relationship is not significant (B=-0.05946, p>0.05).

(h) Repeat (a)–(f) after modifying the data generation process in such a way that there is less noise in the data. The model (3.39) should remain the same. You can do this by decreasing the variance of the normal distribution used to generate the error term ε in (b). Describe your results.

```
set.seed(1)
eps1 <-  rnorm(100, 0, 0.125)
x1 <-  rnorm(100)
y1 <-  -1 + 0.5*x1 + eps1
plot(x1, y1)
lm.fit1 <- lm(y1 ~ x1)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = y1 ~ x1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29052 -0.07545  0.00067  0.07288  0.28664
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.98639    0.01129  -87.34   <2e-16 ***
## x1           0.49988    0.01184   42.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1128 on 98 degrees of freedom
## Multiple R-squared:  0.9479, Adjusted R-squared:  0.9474
## F-statistic:  1782 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
plot(x1, y1)
abline(lm.fit1, lwd = 1, col = 6)
abline(-1, 0.5, lwd = 1, col = 1)
legend(-1, legend = c("model fit", "population regression"), col = 6:1, lwd = 3)
```



There is a slight increase in model fit on the training data in the less noisy data proven by the slight increase in the adjusted R^2 value.

A multiple linear regression was used to test if x1 significantly predicted y1. The fitted regression model was: y1 = -0.98639 + 0.49988(x1). The overall regression was statistically significant (R^2 = 0.95, F(1, 98)= 1782, p<0.005). It was found that x1 significantly predicted y1 (B=0.49988, p<0.005).

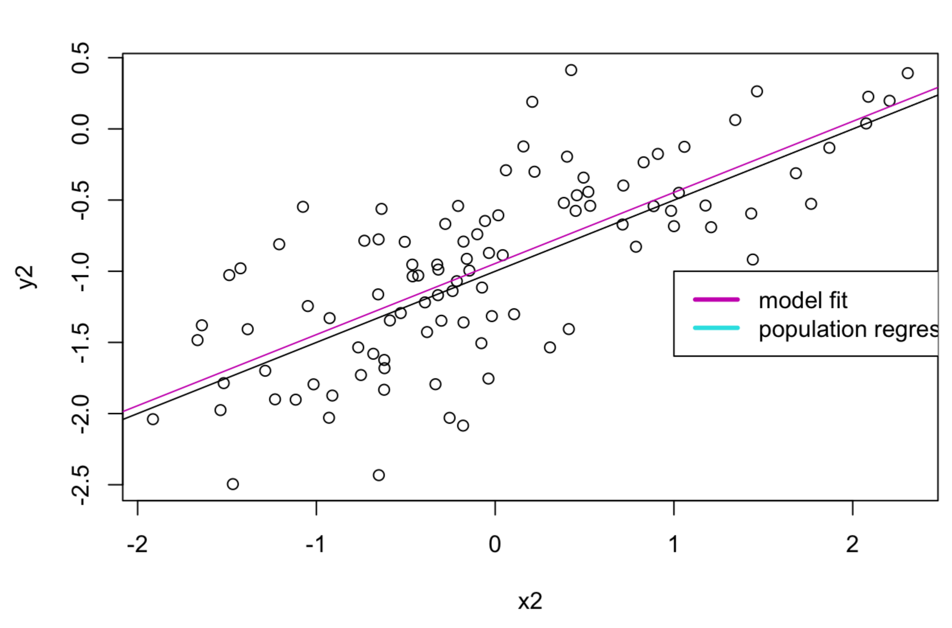
Thus, the linear regression model closely fits the true coefficient values.

(i) Repeat (a)–(f) after modifying the data generation process in such a way that there is more noise in the data. The model (3.39) should remain the same. You can do this by increasing the variance of the normal distribution used to generate the error term ε in (b). Describe your results.

```
set.seed(1)
eps2 <-  rnorm(100, 0, 0.5)
x2 <-  rnorm(100)
y2 <-  -1 + 0.5*x2 + eps2
plot(x2, y2)
lm.fit2 <-  lm(y2 ~ x2)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = y2 ~ x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.16208 -0.30181  0.00268  0.29152  1.14658
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.94557    0.04517  -20.93   <2e-16 ***
## x2           0.49953    0.04736   10.55   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4514 on 98 degrees of freedom
## Multiple R-squared:  0.5317, Adjusted R-squared:  0.5269
## F-statistic: 111.2 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
plot(x2, y2)
abline(lm.fit2, lwd = 1, col = 6)
abline(-1, 0.5, lwd = 1, col = 1)
legend(-1, legend = c("model fit", "population regression"), col = 6:1, lwd = 3)
```



There is a slight increase in model fit on the training data in the more noisy data proven by the large decrease in the adjusted R^2 value.

A simple linear regression was used to test if x2 significantly predicted y2. The fitted regression model was: y2 = -0.98639 + 0.49988(x2). The overall regression was statistically significant (R^2 = 0.53, F(1, 98)= 111.2, p<0.005). It was found that x2 significantly predicted y2 (B=0.49953, p<0.005).

Thus, the linear regression model closely fits the true coefficient values.

(j) What are the confidence intervals for β_0 and β_1 based on the original data set, the noisier data set, and the less noisy dataset? Comment on your result.

```
confint(lm.fit)
```

```
##                  2.5 %     97.5 %
## (Intercept) -1.1150804 -0.9226122
## x            0.3925794  0.6063602
```

```
confint(lm.fit1)
```

```
##                 2.5 %     97.5 %
## (Intercept) -1.008805 -0.9639819
## x1           0.476387  0.5233799
```

```
confint(lm.fit2)
```

```
##                  2.5 %     97.5 %
## (Intercept) -1.0352203 -0.8559276
## x2           0.4055479  0.5935197
```

All of the confidence intervals around x include 0.5. The range of the x CI's decrease in size from x1, x, x2. Additionally, all of the confidence intervals around the intercept include -1.0. These are all close to our original coefficients.