**COMP3354 - Statistical Learning**

**Assignment 1**

**Name: Pranav Talwar**

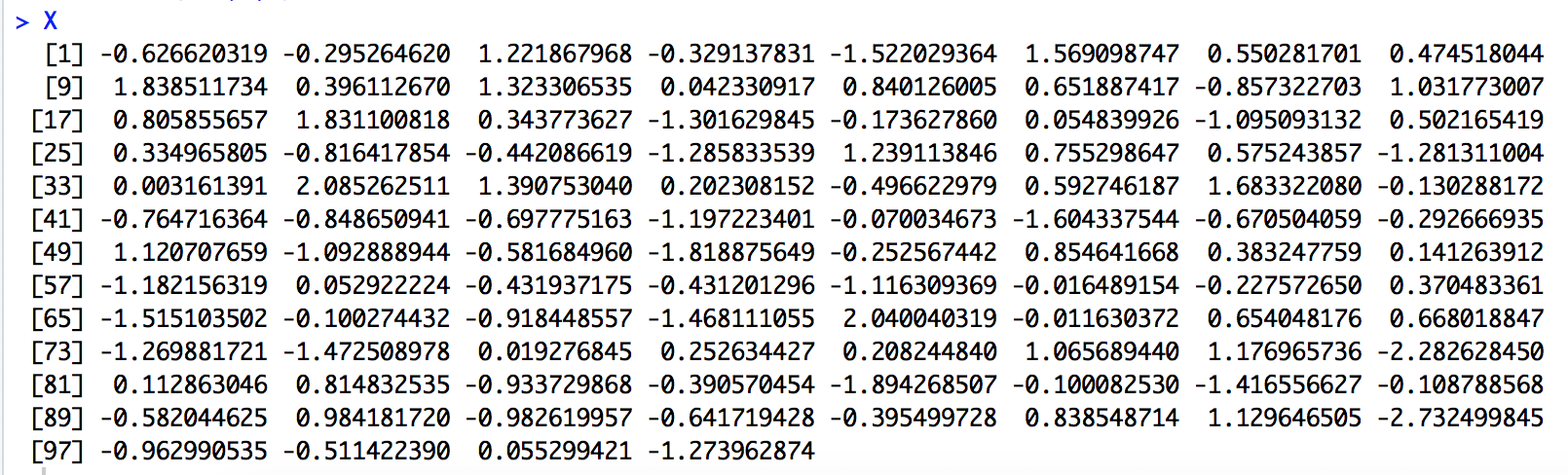
**UID: 3035435462**

Q1. > set.seed(5462)

1. R command : > X = rnorm(100,0,1)

> X

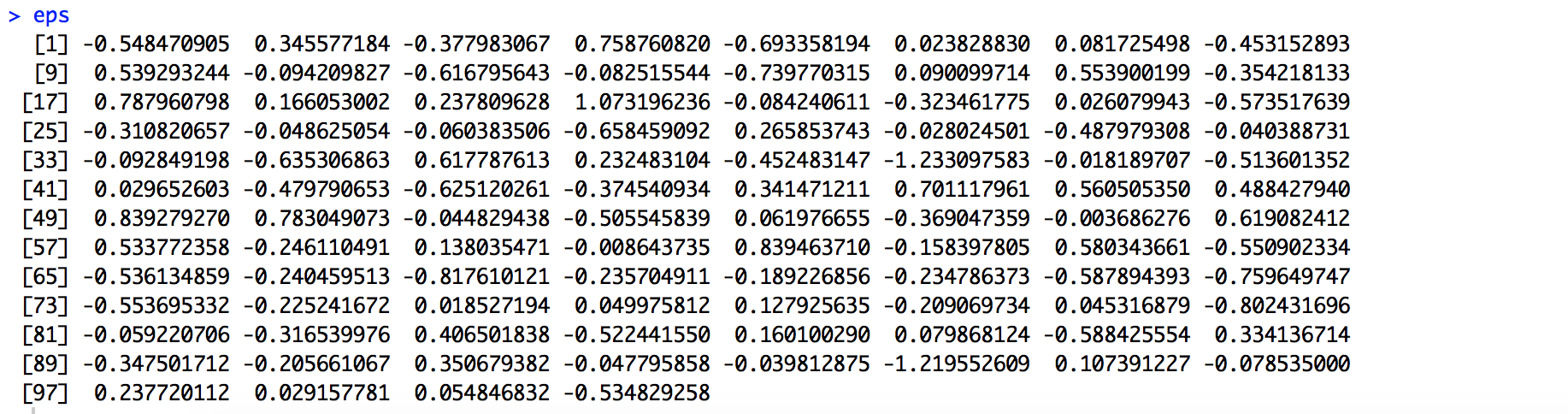
Output:



1. R command: > eps = rnorm(100,0,0.5)

> eps

Output:

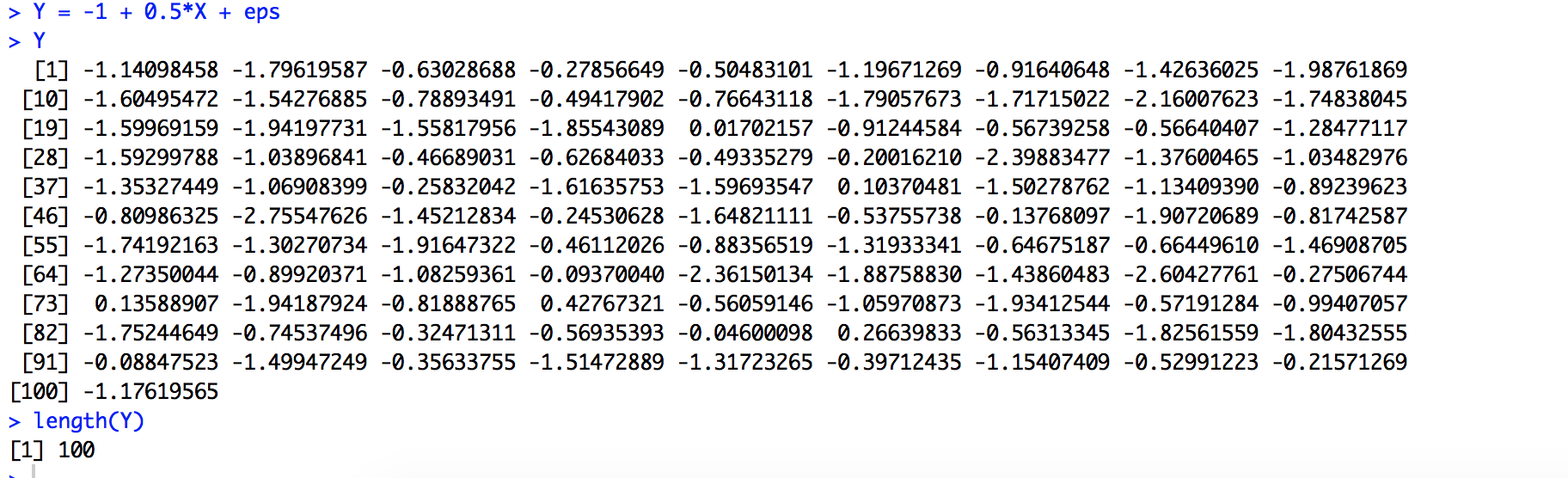


1. R command: > Y = -1 + 0.5\*X + eps

> Y

> length(Y)

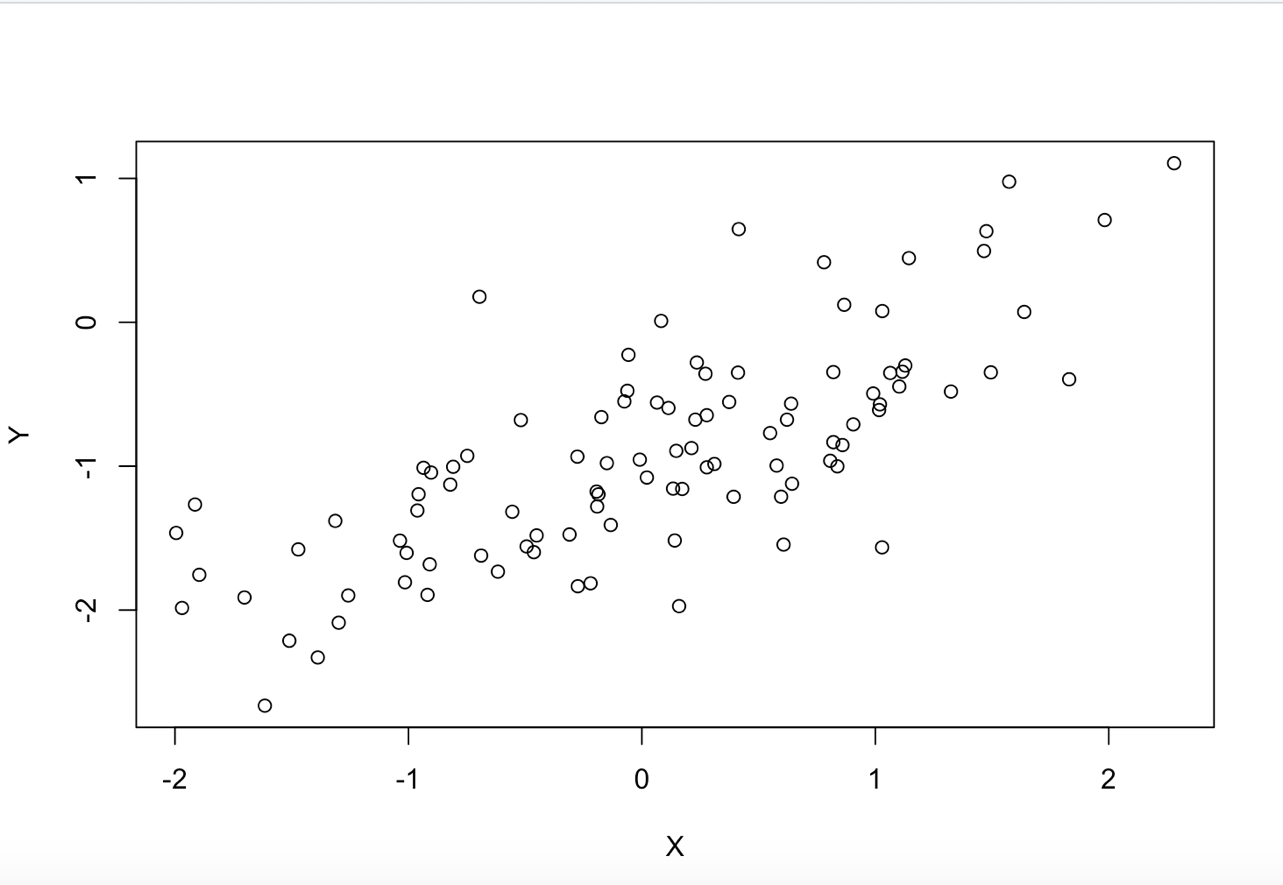
Output:



The length of vector Y is 100. The value of  is -1 and  is 0.5.

1. R command: > plot(X,Y)

Output:



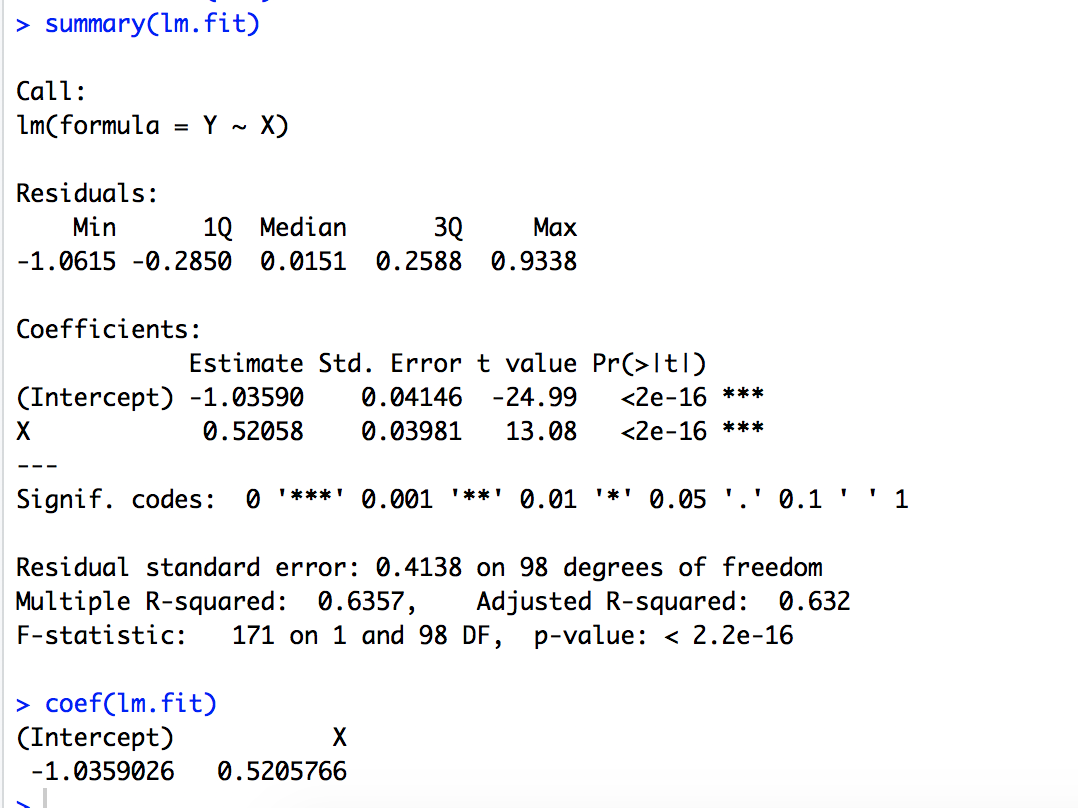
On observing the scatterplot between X and Y we can observe that there seems to be some sort of correlation between the two variables. We can observe that Y maybe dependent on X because when X increases, Y also increases. Hence, we can conclude that they have positive correlation. On viewing the scatterplot, we can also notice that the relationship between Y and X could be modelled using a linear model.

1. R command: > lm.fit = lm(Y~X)

> summary(lm.fit)

> coef(lm.fit)

Output:



We can observe from the linear model that the values of ^ and ^ are -1.0359026

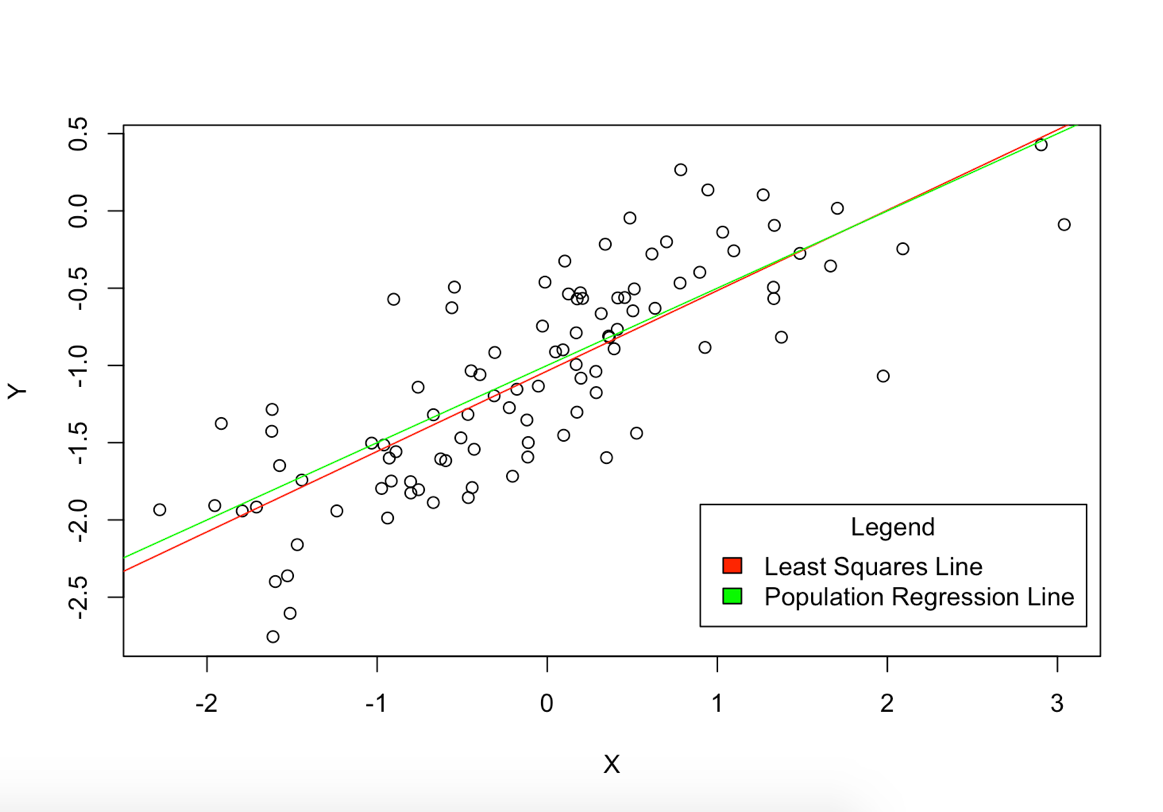
and 0.5205766 respectively. From these two values we can say that Y increases by 0.5205766 when X is increased by one unit. Furthermore, the p-value of x is <0.05 hence we reject the null hypothesis and conclude that there is significant correlation between X and Y. These values are very close to the original values of and hence we can say that the least squares line is a good fit to the data (Multiple R-Squared statistic is above 0.5).

1. R command: > abline(lm.fit, col="red")

> legend(0.9,-1.9,c("Least Squares Line","Population Regression Line"), fill=c("red", "green"), title="Legend")

> abline(-1,0.5, col = "green")

Output:



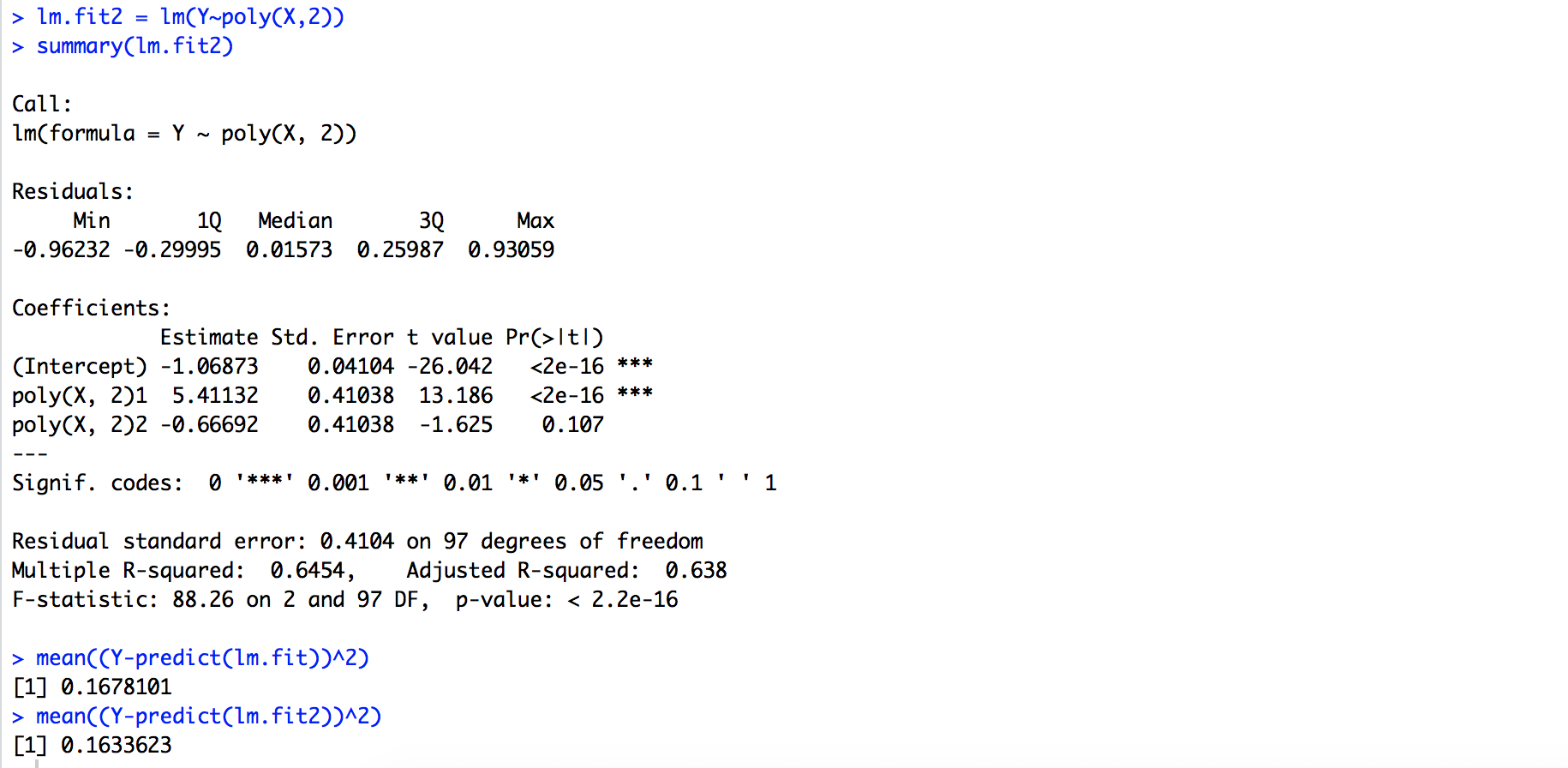
1. R command : > lm.fit2 = lm(Y~poly(X,2))

> summary(lm.fit2)

> mean((Y-predict(lm.fit))^2)

> mean((Y-predict(lm.fit2))^2)

Output:



The training MSE of the two models is 0.1678101 (linear model) and 0.1633623 (polynomial model). Hence, we can see that the polynomial model does not provide a significant improvement as compared to the linear model. The same is displayed by the Residual standard error of both models, where there is not much of a difference. Furthermore, the coefficient of the quadratic term is not statistically significant, because the p-value of the X-squared term is >0.05 which means the null hypothesis is true and that there is no significant relation between the X-squared term and Y. Hence, there is no evidence that the quadratic term improves the model.

1. R command: > X = rnorm(100,0,1)

> eps = rnorm(100,0,0.1)

> Y = -1 + 0.5\*X + eps

> plot(X,Y)

> lm.fit2 = lm(Y~X)

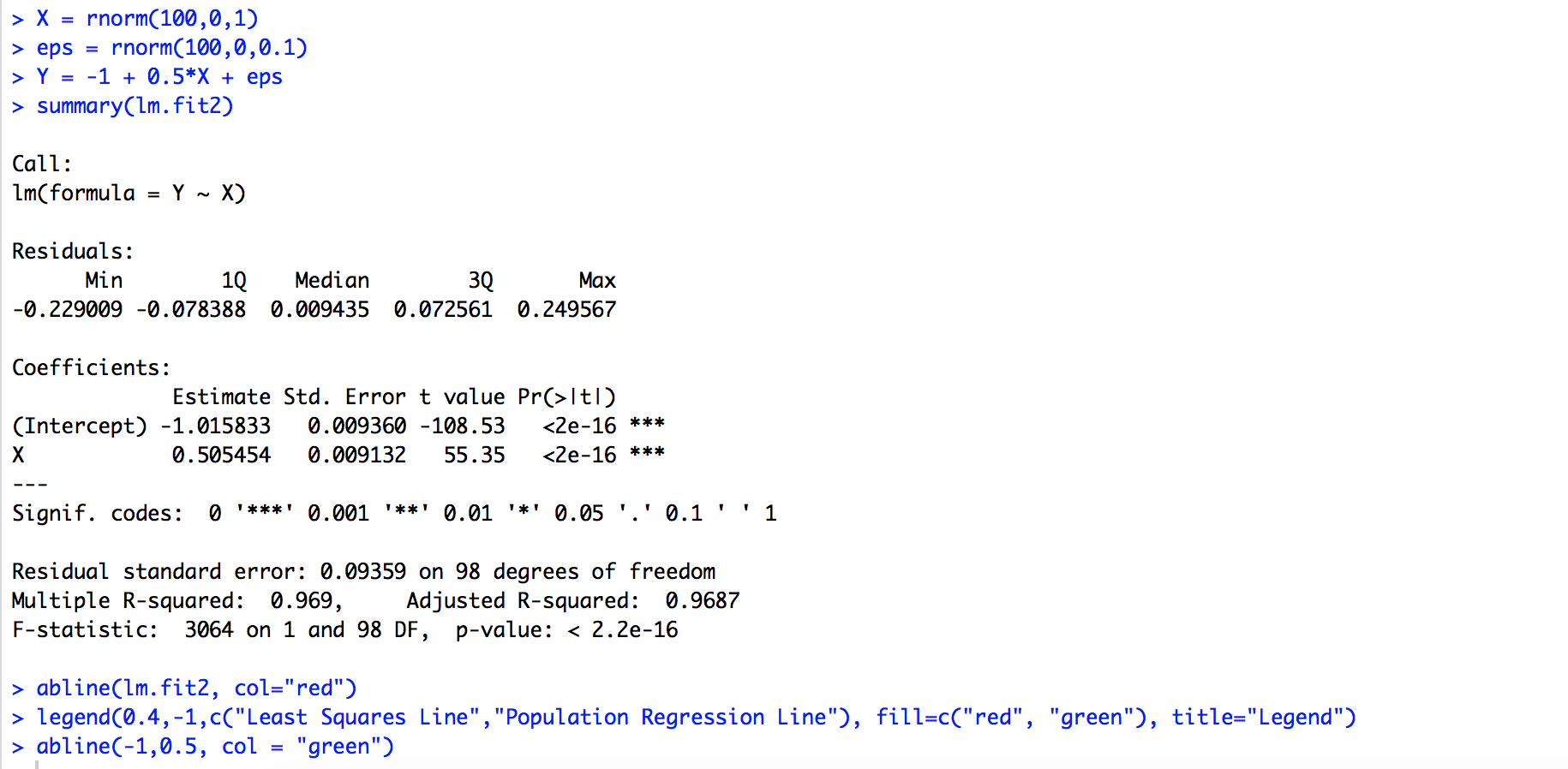
> summary(lm.fit2)

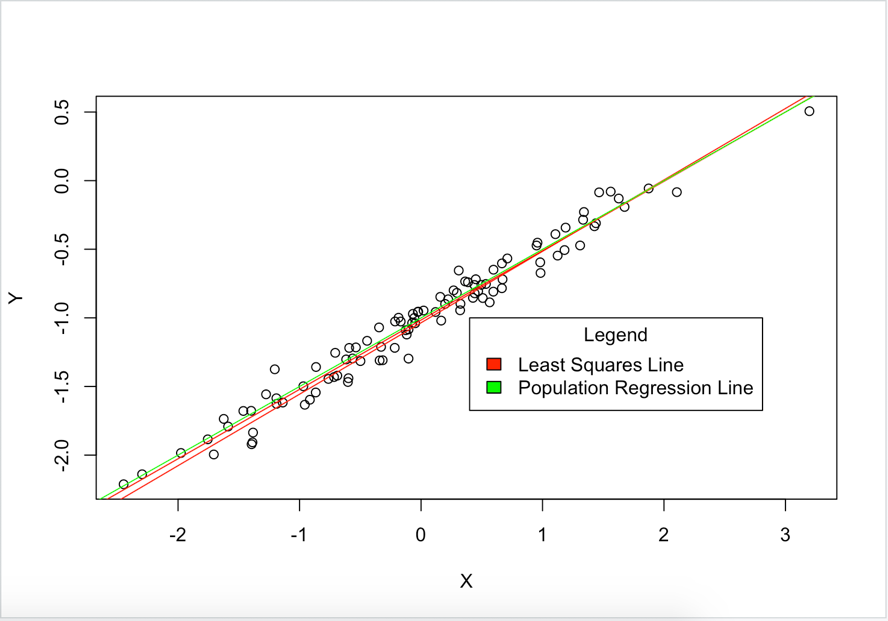
> abline(lm.fit2, col="red")

> legend(0.4,-1,c("Least Squares Line","Population Regression Line"), fill=c("red", "green"), title="Legend")

> abline(-1,0.5, col = "green")

Output:





In this case, the population regression and the least squares line are very close to each other and almost overlap. Due to a decrease in the variance of the error term eps, we can observe that data points are closer to each other and fit the linear model better. This can be seen by the values of the RSE, which has decreased considerably as compared to the case where the variance of the error term was high and also by the increase in the R-squared statistic which is 0.969 (when the variance in the error term eps is 0.01).

1. R command : > X = rnorm(100,0,1)

> eps = rnorm(100,0,0.6)

> Y = -1 + 0.5\*X + eps

> plot(X,Y)

> lm.fit3 = lm(Y~X)

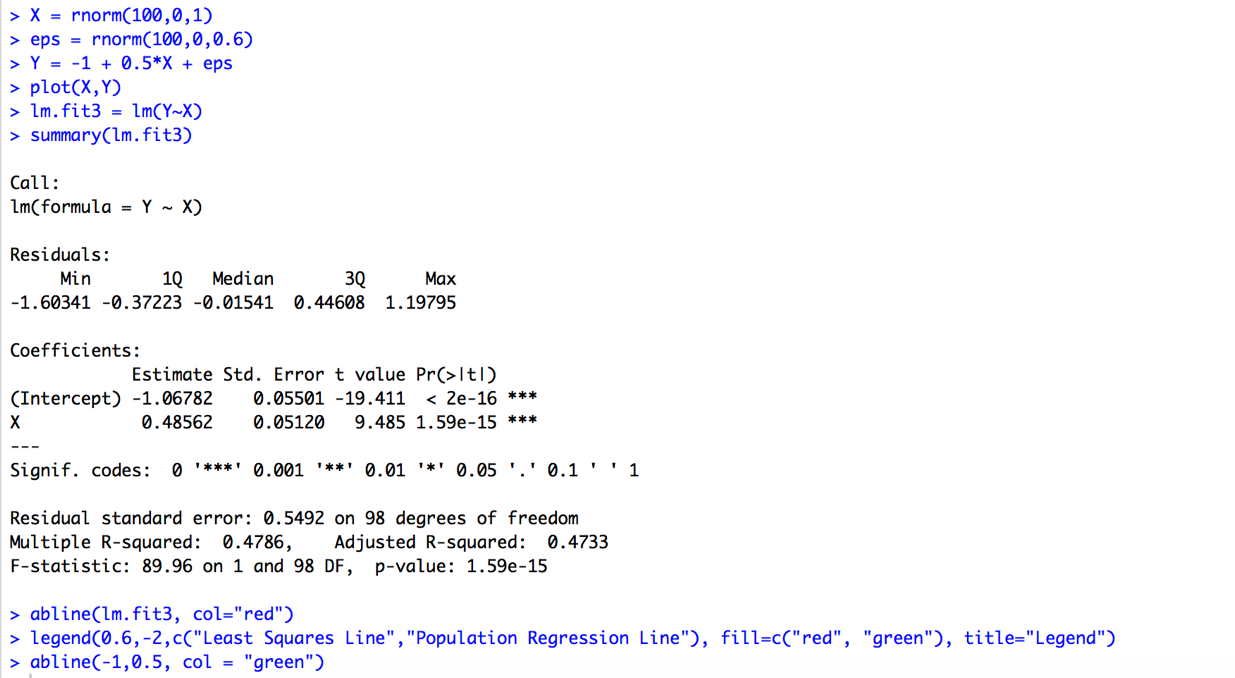
> summary(lm.fit3)

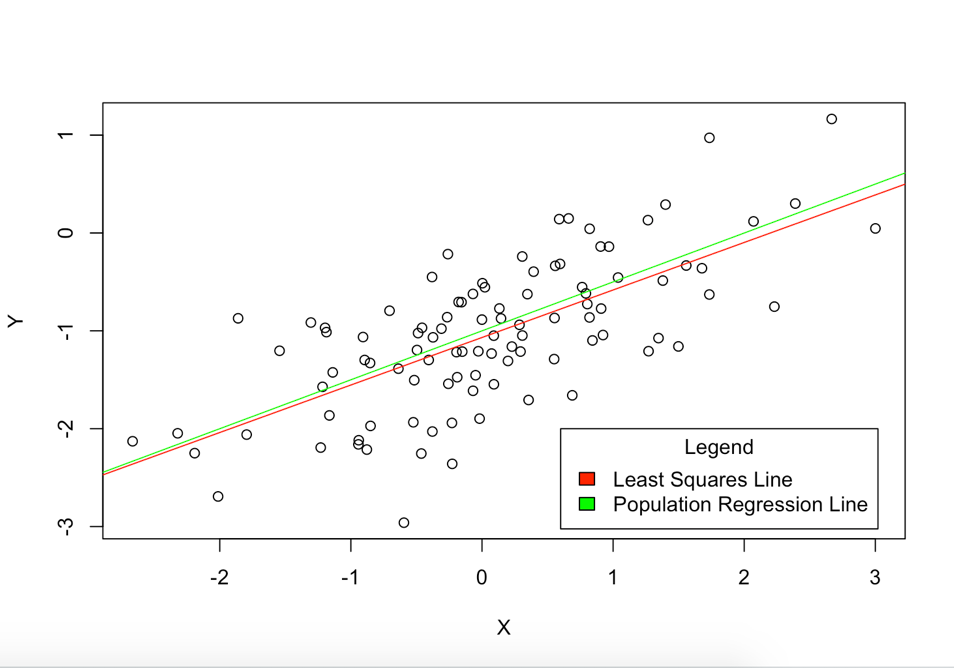
> abline(lm.fit3, col="red")

> legend(0.9,-1.9,c("Least Squares Line","Population Regression Line"), fill=c("red", "green"), title="Legend")

> abline(-1,0.5, col = "green")

Output:





In this case, the population line and the least squares line are not as close to each other and do not overlap as compared to previous cases. Due to an increase in the variance of the error term eps we can see that data points are far apart from each other and might not fit a linear model that well. The following is also represented by the value of the RSE, which has increased considerably as compared to when the variance of the error term was lower. It is also reflected by the decrease in the R-squared statistic which is 0.4786 (when the variance of the error term is 0.36 ).

1. R command:> confint(lm.fit) # original dataset

> confint(lm.fit2) # less noisy dataset

> confint(lm.fit3) # more noisy dataset

Output:



The range of the confidence interval is higher when the noise in the relation between X and Y is more. From the above data we can see that when the range of the confidence interval is low we can predict the value of the intercept and the coefficient of x with greater accuracy as compared to when the range of the confidence interval is high.

Q2. > set.seed(5462)

> library(ISLR)

> attach(Auto)

1. R commands:> mpg01 = rep(0, length(mpg))

> for (i in 1:length(mpg)){

+ if(mpg[i] > median(mpg)){

+ mpg01[i] = 1

+ }

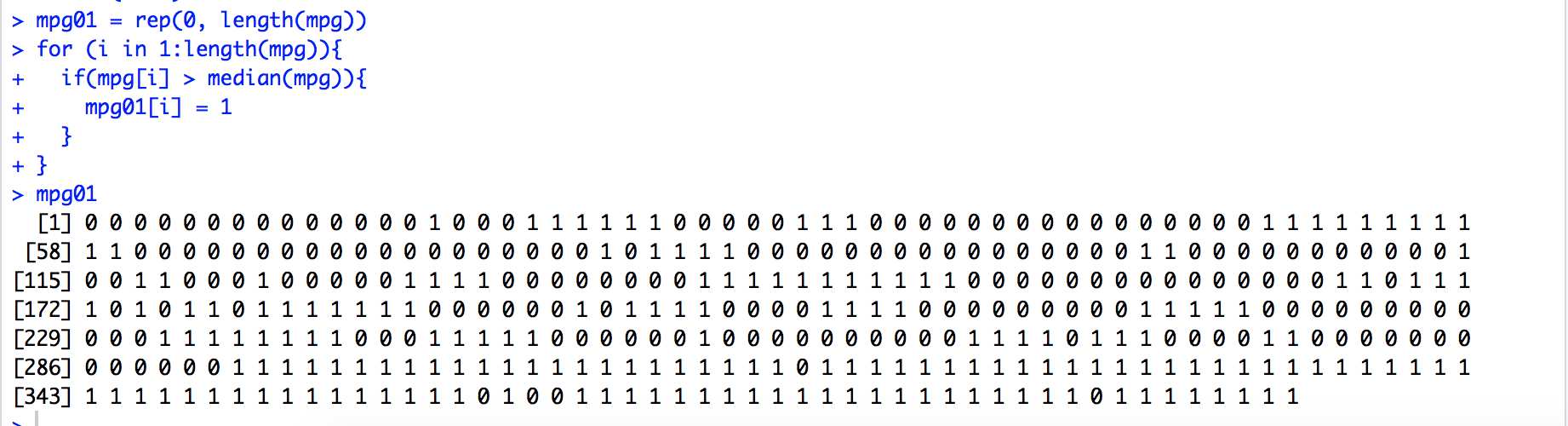
+ }

> mpg01

> Auto01 = data.frame(Auto, mpg01)

> View(Auto01)

Output:





1. R command: > pairs(Auto01[,-9]) # scatterplots of all variables in Auto01 except name

> cor(Auto01[,-9]) #correlation matrix of variables in Auto01 except name

> par(mfrow=c(2,3)) #exploring boxplots

> boxplot(cylinders ~ mpg01, data = Auto, main = "Cylinders vs mpg01")

> boxplot(displacement ~ mpg01, data = Auto, main = "Displacement vs mpg01")

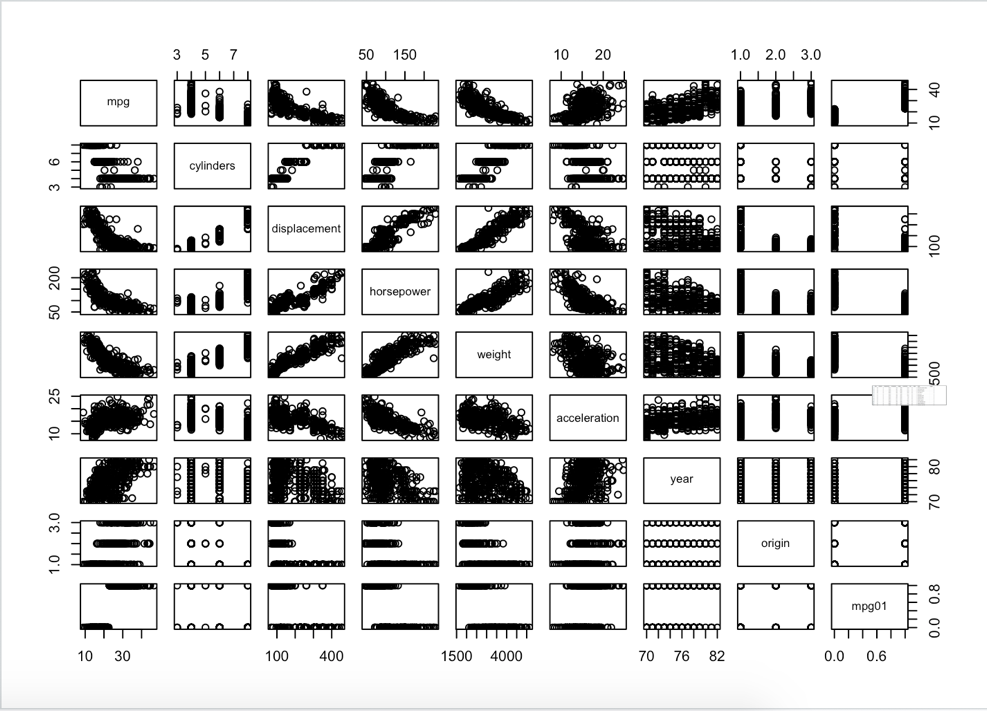
> boxplot(horsepower ~ mpg01, data = Auto, main = "Horsepower vs mpg01")

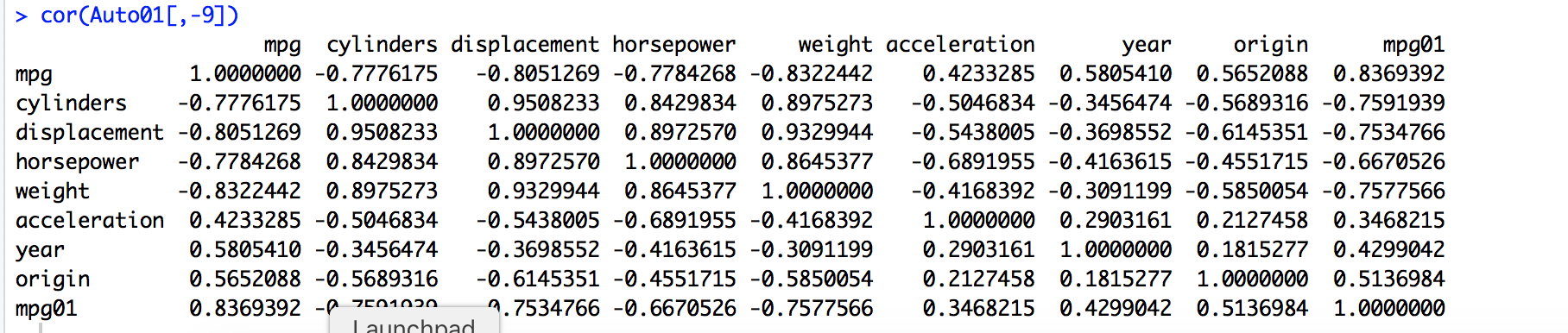
> boxplot(weight ~ mpg01, data = Auto, main = "Weight vs mpg01")

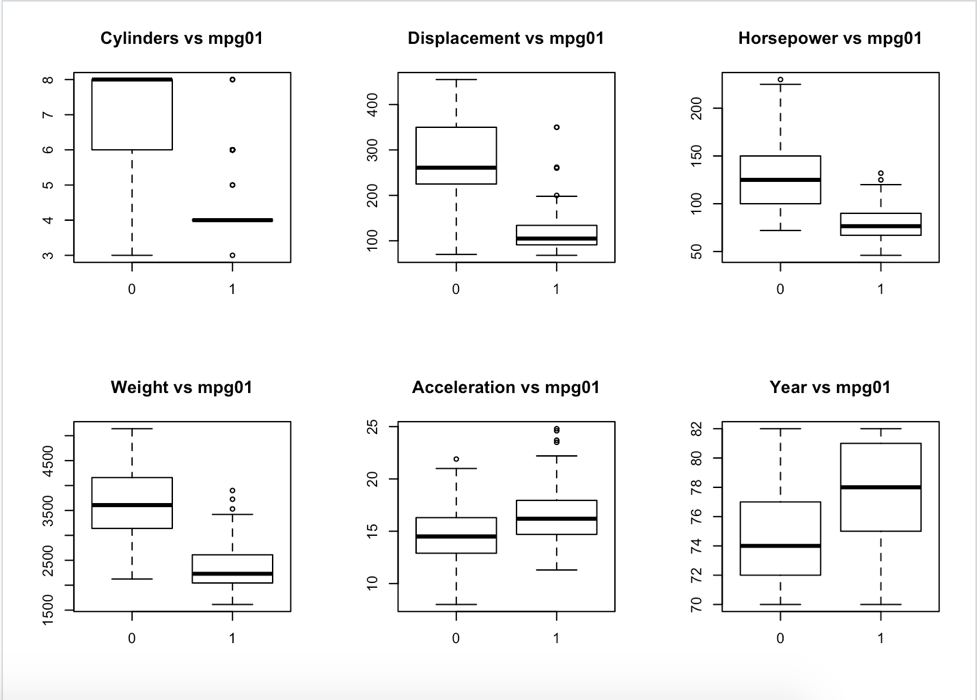
> boxplot(acceleration ~ mpg01, data = Auto, main = "Acceleration vs mpg01")

> boxplot(year ~ mpg01, data = Auto, main = "Year vs mpg01")

Output:







Using the analysis tools givenabove we can see that mpg01 is strongly related to "cylinders", "displacement", "horsepower" and "weight". This can be seen from the correlation matrix ("cylinders", "displacement", "horsepower" and "weight" have the highest correlation amongst all the predictor variables) and also from the boxplots given above.

1. R commands: > train = sample(392, 0.8\*dim(Auto01)[1])

> test = -train

> training\_data = Auto01[train,]

> length(training\_data[,1])

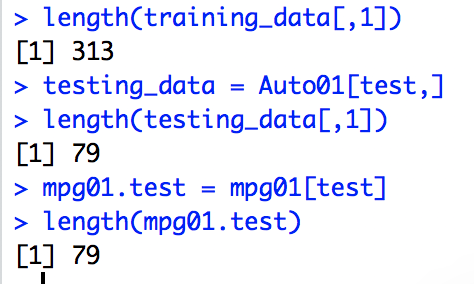
> testing\_data = Auto01[test,]

> length(testing\_data[,1])

> mpg01.test = mpg01[test]

> length(mpg01.test)

Output: Splitting the data intro training and testing data (80:20 ratio)



1. R command: > glm.fits=glm(mpg01~cylinders+displacement+horsepower+weight , data=training\_data ,family=binomial)

> summary(glm.fits)

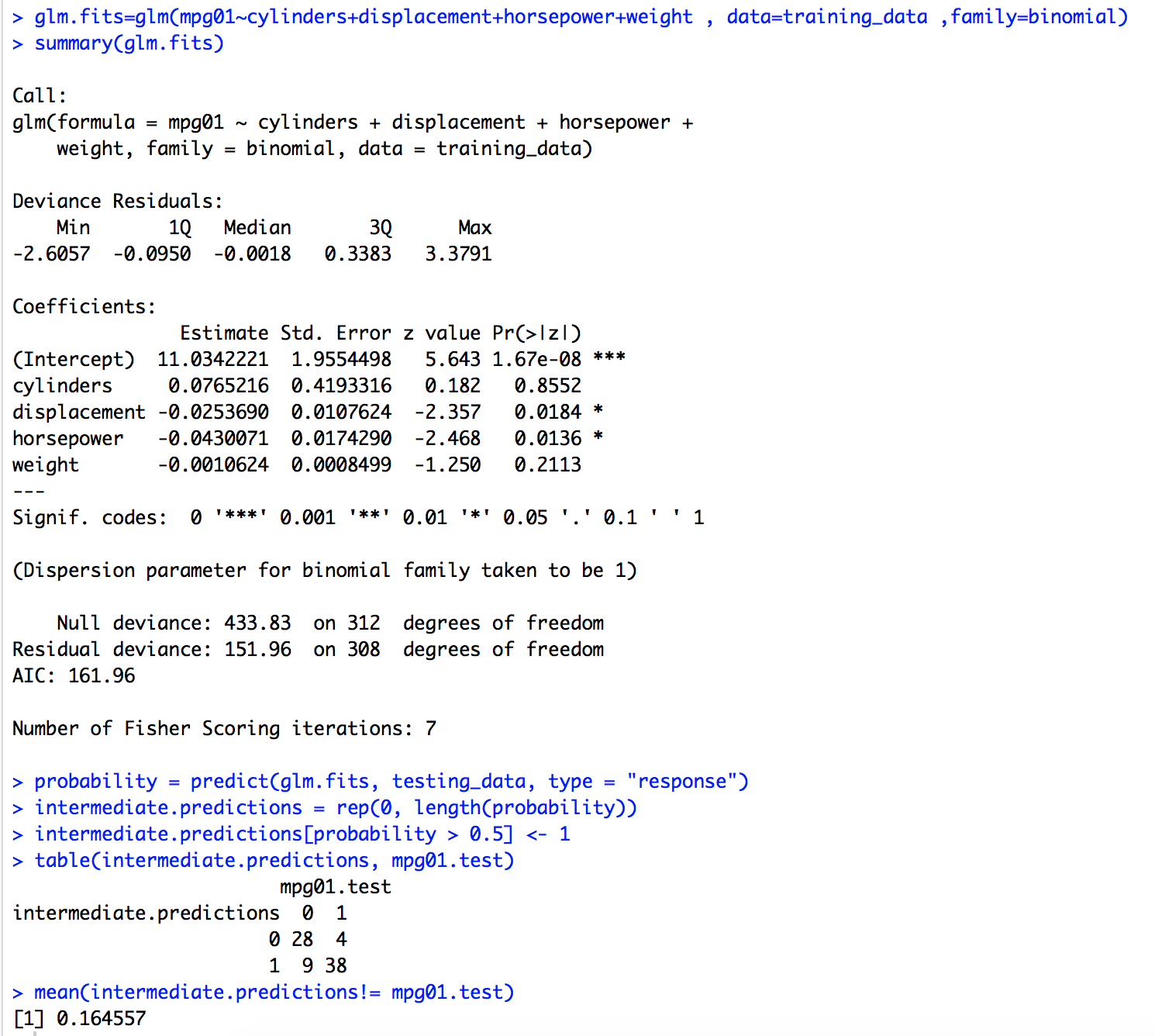
> probability = predict(glm.fits, testing\_data, type = "response")

> intermediate.predictions = rep(0, length(probability))

> intermediate.predictions[probability > 0.5] <- 1

> table(intermediate.predictions, mpg01.test)

> mean(intermediate.predictions!= mpg01.test)



The test error rate is 16.4557 percent.

Q3) set.seed(5462)

1. R command: > library(ISLR)

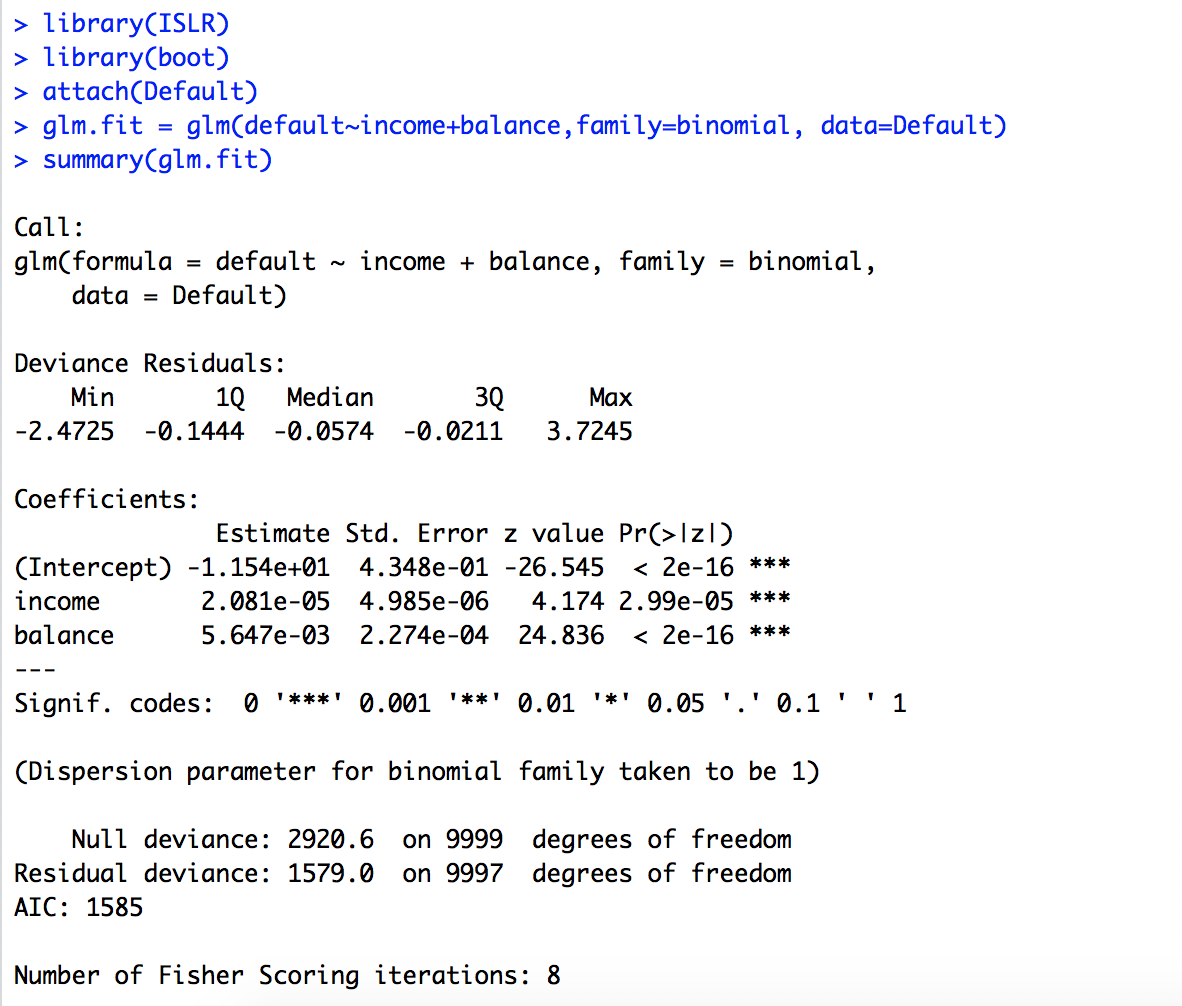
> library(boot)

> attach(Default)

> glm.fit = glm(default~income+balance,family=binomial, data=Default)

> summary(glm.fit)

Output:



The standard error for the coefficient of income is 4.985e-06 and that of the coefficient of balance is 2.274e-04.

1. R command: > boot.fn = function(data,index){

+ glm.fit= + glm(default~income+balance,data=data,family=binomial,subset=index)

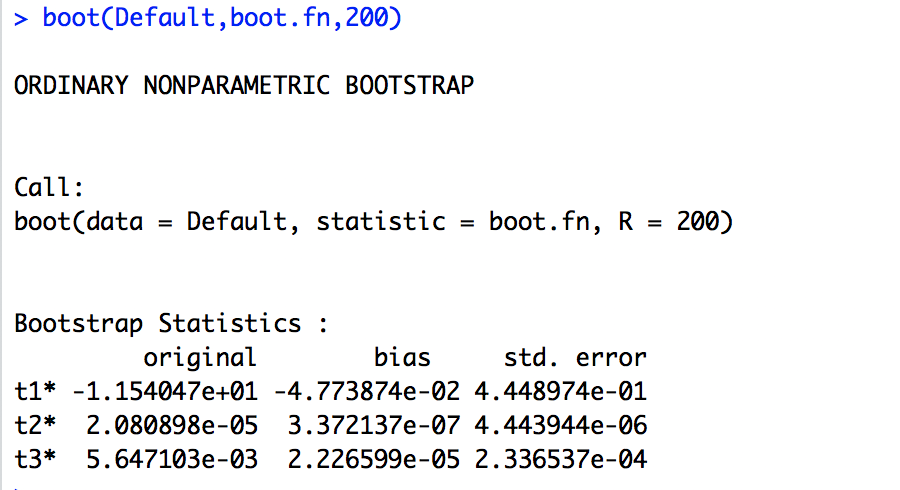
+ return(coef(glm.fit))

+ }

Output: No output

1. R command: > boot(Default,boot.fn,200)

Output:



The standard error of coefficient estimate is calculated using library boot and boot.fn function defined above. The standard error for the coefficient of income is 4.443944e-06 and that of the coefficient of balance is 2.336537e-04 using this method.

1. The standard error of coefficient estimates found from the glm() method and the bootstrap method are extremely close to each other.

Q4) set.seed(5462)

library(MASS)

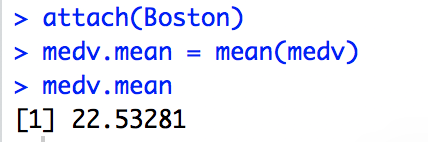
library(stringr)

1. R command: > attach(Boston)

> medv.mean = mean(medv)

> medv.mean

Output:

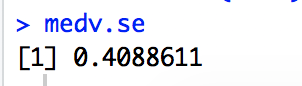


^ (Population mean of medv) is 22.5381.

1. R command: > medv.se = sd(medv)/sqrt(length(medv))

> medv.se

Output:



The estimate of the standard error of ^ is 0.4088611. Since the standard error of the mean is small as compared to the mean itself, the mean does not vary much when different samples are taken from the population.

1. R command: > boot.fn = function(data, index){

+ return (mean(data[index]))

+ }

> medv.boot = boot(medv,boot.fn,1000)

> x = capture.output(medv.boot) # store the output as text

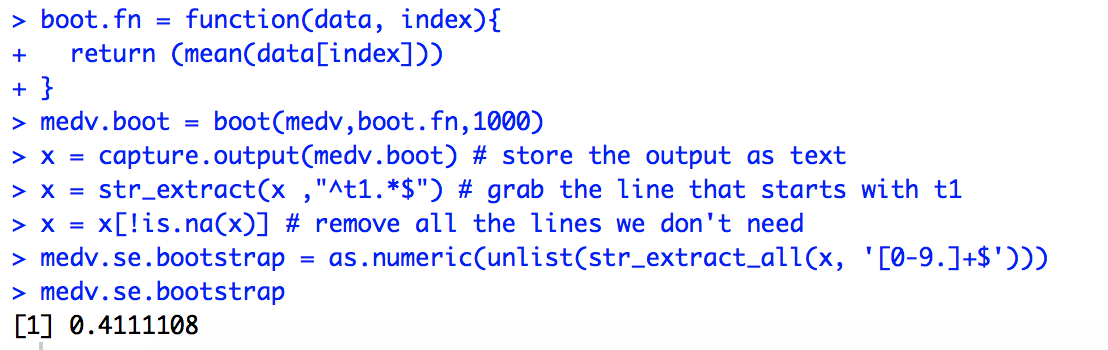
> x = str\_extract(x ,"^t1.\*$") # grab the line that starts with t1

> x = x[!is.na(x)] # remove all the lines we don't need

> medv.se.bootstrap = as.numeric(unlist(str\_extract\_all(x, '[0-9.]+$')))

> medv.se.bootstrap

Output:



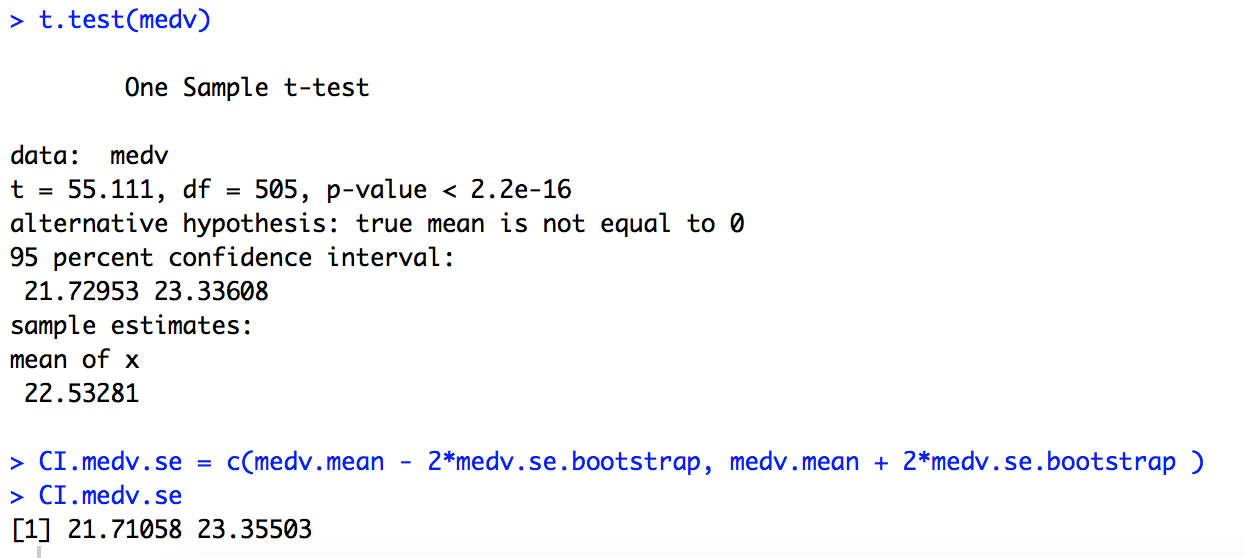
The estimate of standard error of ^ is 0.4111108 using the bootstrap method which is extremely close to 0.4088611 which was found in part B.

1. R command: > t.test(medv)

> CI.medv.se = c(medv.mean - 2\*medv.se.bootstrap, medv.mean + + 2\*medv.se.bootstrap )

> CI.medv.se

Output:

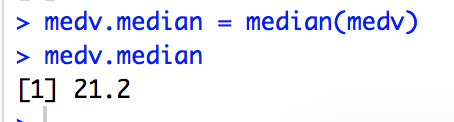


The confidence interval generated by the t-test is (21.72953, 23.33608) while that generated with the help of the bootstrap estimate of the standard error is (21.71058, 23.35503). Hence we can see that the bootstrap confidence interval is very close to the one provided by the t.test(medv) function.

1. R command: > medv.median = median(medv)

> medv.median

Output:



^ med (Population median of medv) is 21.2.

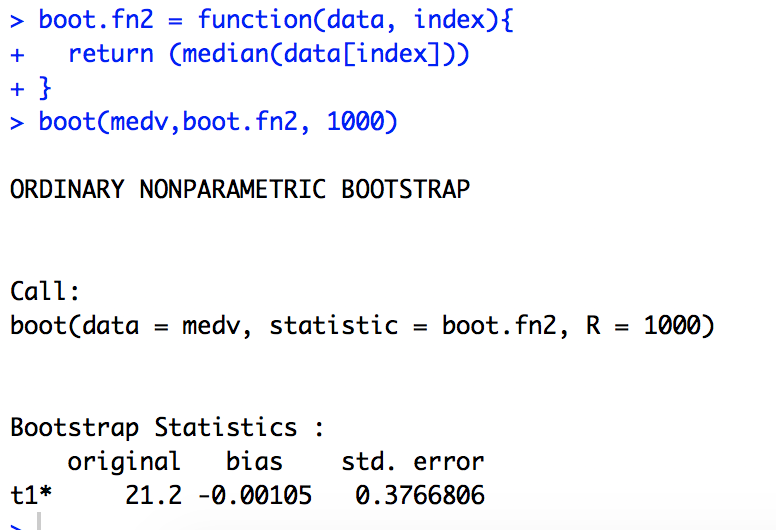
1. R command: > boot.fn2 = function(data, index){

+ return (median(data[index]))

+ }

> boot(medv,boot.fn2, 1000)

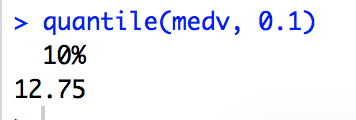
Output:



Using bootstrap the standard error^ med of the comes out to be 0.3766906 which is extremely small when compared to the value of the median which is 21.2 (given by bootstrap). This means that the mean does not vary too much when different samples are taken from a population dataset.

1. R command: > quantile(medv, 0.1)

Output:



The tenth percentile for medv (^0.1) in the Boston suburbs is 12.75.

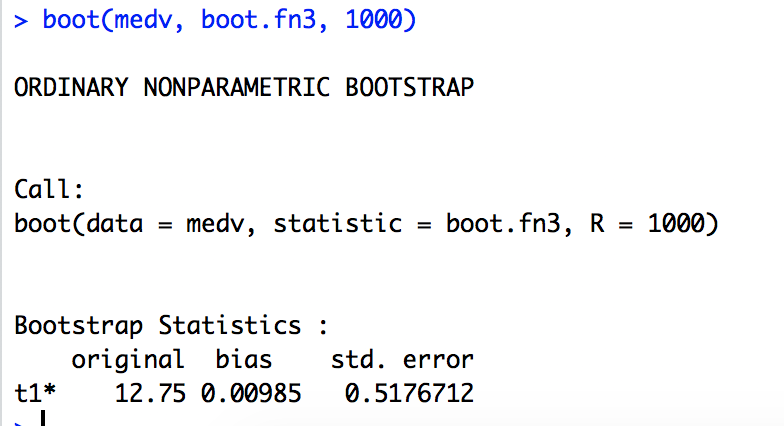
1. R commands: > boot.fn3 = function(data, index){

+ return (quantile(data[index], 0.1))

+ }

> boot(medv, boot.fn3, 1000)

Output:



^0.1 is 12.75 with a standard error of 0.5176712. The standard error is small relative to the value of ^0.1. Hence the tenth percentile of ^0.1 does not vary much when different samples of the data are considered from the population dataset.