Statistical Learning Assignment 1

Tarun Sudhams

22nd October 2018

## Question 1

*In this exercise you will create some simulated data and will fit simple linear regression models to it. Make sure to use set.seed(Your UID) prior to starting part (a) to ensure 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(3876)  
x=rnorm(100, mean = 0, sd = 1)

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

eps = rnorm(100,mean=0,sd = 0.25)

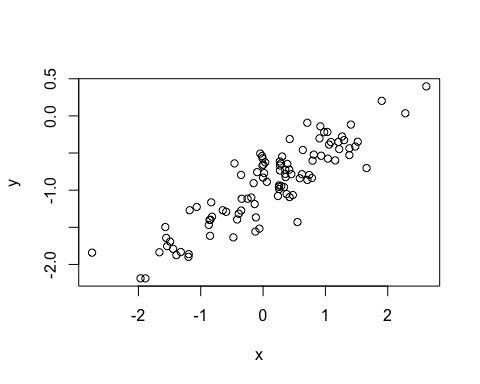
*(c) Using x and eps, generate a vector y according to the model Y = -1+0.5X+ε.*

y = -1+0.5\*x + eps  
length(y)

## [1] 100

The length of Y is 100 and value of β^o is -1 and β^1 is 0.5. *(d) Create a scatterplot displaying the relationship between x and y. Comment on what you observe.*

plot(y~x)

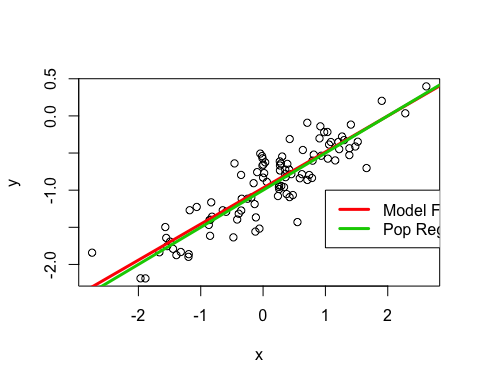
 The graph ploted above shows a linear relationship between Y and X. *(e) Fit a least squares linear model to predict y using x. Comment on the model obtained. How do βˆ0 and ?βˆ1 compare to β^o and β^1 ?*

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

##   
## Call:  
## lm(formula = y ~ x)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.72668 -0.15521 -0.01708 0.19065 0.55342   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.97065 0.02542 -38.18 <2e-16 \*\*\*  
## x 0.48541 0.02590 18.74 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2528 on 98 degrees of freedom  
## Multiple R-squared: 0.7819, Adjusted R-squared: 0.7797   
## F-statistic: 351.3 on 1 and 98 DF, p-value: < 2.2e-16

The linear regression fits very closely to the true value of the coefficients it was constructed from. The model has a large F-Statistic value and with a near zero p-value. *(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=3,col=2)  
abline(-1,0.5,lwd=3,col=3)  
legend(-1, legend = c("Model Fit", "Pop Reg"), col=2:3,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   
## -0.72806 -0.15612 -0.01772 0.18870 0.55172   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.968547 0.031411 -30.835 <2e-16 \*\*\*  
## x 0.485291 0.026048 18.631 <2e-16 \*\*\*  
## I(x^2) -0.002171 0.018842 -0.115 0.908   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2541 on 97 degrees of freedom  
## Multiple R-squared: 0.7819, Adjusted R-squared: 0.7774   
## F-statistic: 173.9 on 2 and 97 DF, p-value: < 2.2e-16

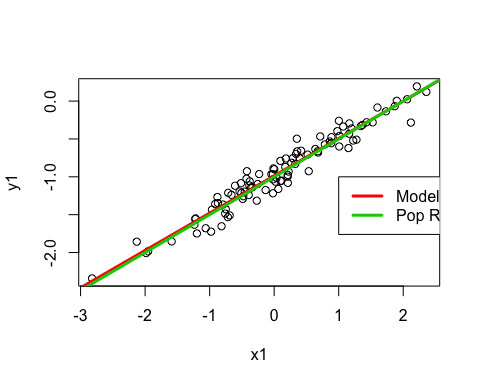
The coefficient of X^2 is not significant as p-value is greater 0.05. So it’s clearly that there is not enough evidence to show that the quadratic term imporves the model fit even if R^2 and RSE values are a little lower than the linear model.

*(h) Repeat (a)–(f) after modifying the data generation process in such a way that there is less noise in the data. The model (1) 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(3876)  
eps1 = rnorm(100, 0, 0.125)  
x1 = rnorm(100)  
y1 = -1 + .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   
## -0.34198 -0.06718 0.01616 0.07991 0.31600   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.98624 0.01238 -79.65 <2e-16 \*\*\*  
## x1 0.49308 0.01227 40.17 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1231 on 98 degrees of freedom  
## Multiple R-squared: 0.9428, Adjusted R-squared: 0.9422   
## F-statistic: 1614 on 1 and 98 DF, p-value: < 2.2e-16

abline(lm.fit1, lwd=3,col=2)  
abline(-1, 0.5, lwd=3,col=3)  
legend(-1, legend = c("Model Fit", "Pop Reg"), col=2:3, lwd=3)

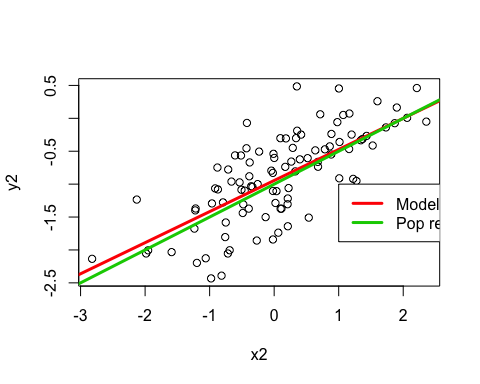
 With the decrease in variance, the noise also decreased and now we have higher values of R^2 and the relationship is more linear. Furthermore, the RSE has also decreased.

*(i) Repeat (a)–(f) after modifying the data generation process in such a way that there is more noise in the data. The model (1) 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(3876)  
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   
## -1.36790 -0.26871 0.06464 0.31962 1.26401   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.94496 0.04953 -19.078 < 2e-16 \*\*\*  
## x2 0.47234 0.04909 9.621 8.04e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4923 on 98 degrees of freedom  
## Multiple R-squared: 0.4857, Adjusted R-squared: 0.4805   
## F-statistic: 92.57 on 1 and 98 DF, p-value: 8.043e-16

abline(lm.fit2, lwd = 3, col = 2)  
abline(-1, 0.5, lwd = 3, col = 3)  
legend(-1, legend = c("Model Fit", "Pop reg"), col=2:3, lwd = 3)

 On comparing the previous vales of R^2 and RSE, we can R^2 and RSE have increased by a considerable amount. *(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 data set? Comment on your results.*

confint(lm.fit)

## 2.5 % 97.5 %  
## (Intercept) -1.0211076 -0.9201974  
## x 0.4340156 0.5367998

confint(lm.fit1)

## 2.5 % 97.5 %  
## (Intercept) -1.010814 -0.9616677  
## x1 0.468728 0.5174405

confint(lm.fit2)

## 2.5 % 97.5 %  
## (Intercept) -1.043256 -0.8466707  
## x2 0.374912 0.5697622

Seems like all the confidence intervals are approximately around the 0.5 mark. fit1’s interval is observed to be slightly narrow than fit’s interval and fit2’s interval is wider than fit’s interval.

## Question 2

*In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.*

*(a) Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables.*

library(ISLR)  
summary(Weekly)

## Year Lag1 Lag2 Lag3   
## Min. :1990 Min. :-18.1950 Min. :-18.1950 Min. :-18.1950   
## 1st Qu.:1995 1st Qu.: -1.1540 1st Qu.: -1.1540 1st Qu.: -1.1580   
## Median :2000 Median : 0.2410 Median : 0.2410 Median : 0.2410   
## Mean :2000 Mean : 0.1506 Mean : 0.1511 Mean : 0.1472   
## 3rd Qu.:2005 3rd Qu.: 1.4050 3rd Qu.: 1.4090 3rd Qu.: 1.4090   
## Max. :2010 Max. : 12.0260 Max. : 12.0260 Max. : 12.0260   
## Lag4 Lag5 Volume   
## Min. :-18.1950 Min. :-18.1950 Min. :0.08747   
## 1st Qu.: -1.1580 1st Qu.: -1.1660 1st Qu.:0.33202   
## Median : 0.2380 Median : 0.2340 Median :1.00268   
## Mean : 0.1458 Mean : 0.1399 Mean :1.57462   
## 3rd Qu.: 1.4090 3rd Qu.: 1.4050 3rd Qu.:2.05373   
## Max. : 12.0260 Max. : 12.0260 Max. :9.32821   
## Today Direction   
## Min. :-18.1950 Down:484   
## 1st Qu.: -1.1540 Up :605   
## Median : 0.2410   
## Mean : 0.1499   
## 3rd Qu.: 1.4050   
## Max. : 12.0260

data("Auto")  
mpg01 <- rep(0,length(Auto$mpg))  
mpg01[Auto$mpg > median(Auto$mpg)] <- 1  
Auto <- data.frame(Auto, mpg01)  
summary(Auto)

## mpg cylinders displacement horsepower   
## Min. : 9.00 Min. :3.000 Min. : 68.0 Min. : 46.0   
## 1st Qu.:17.00 1st Qu.:4.000 1st Qu.:105.0 1st Qu.: 75.0   
## Median :22.75 Median :4.000 Median :151.0 Median : 93.5   
## Mean :23.45 Mean :5.472 Mean :194.4 Mean :104.5   
## 3rd Qu.:29.00 3rd Qu.:8.000 3rd Qu.:275.8 3rd Qu.:126.0   
## Max. :46.60 Max. :8.000 Max. :455.0 Max. :230.0   
##   
## weight acceleration year origin   
## Min. :1613 Min. : 8.00 Min. :70.00 Min. :1.000   
## 1st Qu.:2225 1st Qu.:13.78 1st Qu.:73.00 1st Qu.:1.000   
## Median :2804 Median :15.50 Median :76.00 Median :1.000   
## Mean :2978 Mean :15.54 Mean :75.98 Mean :1.577   
## 3rd Qu.:3615 3rd Qu.:17.02 3rd Qu.:79.00 3rd Qu.:2.000   
## Max. :5140 Max. :24.80 Max. :82.00 Max. :3.000   
##   
## name mpg01   
## amc matador : 5 Min. :0.0   
## ford pinto : 5 1st Qu.:0.0   
## toyota corolla : 5 Median :0.5   
## amc gremlin : 4 Mean :0.5   
## amc hornet : 4 3rd Qu.:1.0   
## chevrolet chevette: 4 Max. :1.0   
## (Other) :365

*(b) Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.*

cor(Auto[,-9])

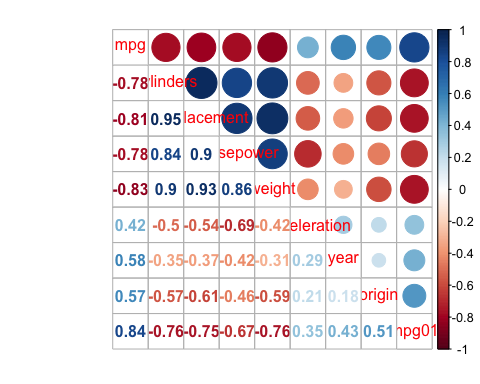
## mpg cylinders displacement horsepower weight  
## mpg 1.0000000 -0.7776175 -0.8051269 -0.7784268 -0.8322442  
## cylinders -0.7776175 1.0000000 0.9508233 0.8429834 0.8975273  
## displacement -0.8051269 0.9508233 1.0000000 0.8972570 0.9329944  
## horsepower -0.7784268 0.8429834 0.8972570 1.0000000 0.8645377  
## weight -0.8322442 0.8975273 0.9329944 0.8645377 1.0000000  
## acceleration 0.4233285 -0.5046834 -0.5438005 -0.6891955 -0.4168392  
## year 0.5805410 -0.3456474 -0.3698552 -0.4163615 -0.3091199  
## origin 0.5652088 -0.5689316 -0.6145351 -0.4551715 -0.5850054  
## mpg01 0.8369392 -0.7591939 -0.7534766 -0.6670526 -0.7577566  
## acceleration year origin mpg01  
## mpg 0.4233285 0.5805410 0.5652088 0.8369392  
## cylinders -0.5046834 -0.3456474 -0.5689316 -0.7591939  
## displacement -0.5438005 -0.3698552 -0.6145351 -0.7534766  
## horsepower -0.6891955 -0.4163615 -0.4551715 -0.6670526  
## weight -0.4168392 -0.3091199 -0.5850054 -0.7577566  
## acceleration 1.0000000 0.2903161 0.2127458 0.3468215  
## year 0.2903161 1.0000000 0.1815277 0.4299042  
## origin 0.2127458 0.1815277 1.0000000 0.5136984  
## mpg01 0.3468215 0.4299042 0.5136984 1.0000000

### Corrplot

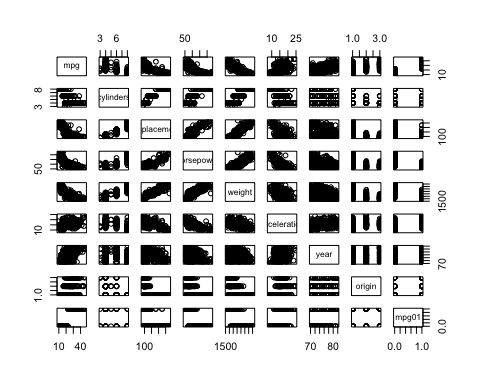
library(corrplot)

## corrplot 0.84 loaded

corrplot::corrplot.mixed(cor(Auto[,-9]), upper="circle")

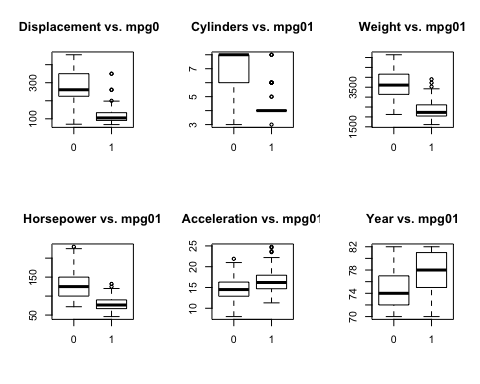
 ### Scatterplot Matrix

pairs(Auto[, -9])



### Boxplots

par(mfrow=c(2,3))  
boxplot(displacement ~ mpg01, data = Auto, main = "Displacement vs. mpg01")  
boxplot(cylinders ~ mpg01, data = Auto, main = "Cylinders vs. mpg01")  
boxplot(weight ~ mpg01, data = Auto, main = "Weight vs. mpg01")  
boxplot(horsepower ~ mpg01, data = Auto, main = "Horsepower vs. mpg01")  
boxplot(acceleration ~ mpg01, data = Auto, main = "Acceleration vs. mpg01")  
boxplot(year ~ mpg01, data = Auto, main = "Year vs. mpg01")

 There exists a clear anti-correlation between “mpg01” and cyclinders“,”displacement“,”horsepower" and “weight”.

*(c) Split the data into a training set and a test set.*

set.seed(123)  
train <- sample(1:dim(Auto)[1], dim(Auto)[1]\*.7, rep=FALSE)  
test <- train  
training\_data = Auto[train, ]  
testing\_data = Auto[test, ]  
mpg01.test <- mpg01[test]

*(f) Perform logistic regression on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?*

glm.fit = glm(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,   
 family = binomial, subset = train)  
glm.probs = predict(glm.fit, testing\_data, type = "response")  
glm.pred = rep(0, length(glm.probs))  
glm.pred[glm.probs > 0.5] = 1  
mean(glm.pred != mpg01.test)

## [1] 0.09489051

The above logistic regression model has a 9.48% test error rate.

## Question 3

*We continue to consider the use of a logistic regression model to predict the probability of default using income and balance on the Default data set. In particular, we will now compute estimates for the standard errors of the income and balance logistic regression coefficients in two different ways: (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the glm() function. Do not forget to set a random seed before beginning your analysis.*

*(a) Using the summary() and glm() functions, determine the estimated standard errors for the coefficients associated with income and balance in a multiple logistic regression model that uses both predictors.*

set.seed(3876)  
attach(Default)  
glm.fit = glm(default ~ income + balance, data = Default, family = binomial)  
summary(glm.fit)

##   
## Call:  
## glm(formula = default ~ income + balance, family = binomial,   
## data = Default)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.4725 -0.1444 -0.0574 -0.0211 3.7245   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.154e+01 4.348e-01 -26.545 < 2e-16 \*\*\*  
## income 2.081e-05 4.985e-06 4.174 2.99e-05 \*\*\*  
## balance 5.647e-03 2.274e-04 24.836 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2920.6 on 9999 degrees of freedom  
## Residual deviance: 1579.0 on 9997 degrees of freedom  
## AIC: 1585  
##   
## Number of Fisher Scoring iterations: 8

The glm() estimates of standard errors for the coefficients β0, β1 and β2 are 0.434756, 4.9841572 x 10^{-6} and 2.2737314 x 10^{-4}

*(b) Write a function, boot.fn(), that takes as input the Default data set as well as an index of the observations, and that outputs the coefficient estimates for income and balance in the multiple logistic regression model.*

boot.fn <- function(data,index)  
{  
 fit <-glm(default ~ income + balance, data = data, family = "binomial", subset = index)  
 return (coef(fit))  
}

*(c) Write a function, boot.fn(), that takes as input the Default data set as well as an index of the observations, and that outputs the coefficient estimates for income and balance in the multiple logistic regression model.*

library(boot)  
boot(Default, boot.fn, 1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Default, statistic = boot.fn, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* -1.154047e+01 -1.451662e-02 4.281430e-01  
## t2\* 2.080898e-05 2.160095e-08 4.690553e-06  
## t3\* 5.647103e-03 7.653440e-06 2.288706e-04

The bootstrap estimates of the standard errors for the coefficients β0,β1 and β2 are 0.4239, 4.583 x 10^(-6) and 2.268 x 10^(-4)

*(d) Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.*

The glm() estimates and bootstrap estimates of the standard errors are very similar to each other.

## Question 4

*We will now consider the Boston housing data set, from the MASS library.*

*(a) Based on this data set, provide an estimate for the population mean of medv. Call this estimate μˆ.*

library(MASS)  
summary(Boston)

## crim zn indus chas   
## Min. : 0.00632 Min. : 0.00 Min. : 0.46 Min. :0.00000   
## 1st Qu.: 0.08204 1st Qu.: 0.00 1st Qu.: 5.19 1st Qu.:0.00000   
## Median : 0.25651 Median : 0.00 Median : 9.69 Median :0.00000   
## Mean : 3.61352 Mean : 11.36 Mean :11.14 Mean :0.06917   
## 3rd Qu.: 3.67708 3rd Qu.: 12.50 3rd Qu.:18.10 3rd Qu.:0.00000   
## Max. :88.97620 Max. :100.00 Max. :27.74 Max. :1.00000   
## nox rm age dis   
## Min. :0.3850 Min. :3.561 Min. : 2.90 Min. : 1.130   
## 1st Qu.:0.4490 1st Qu.:5.886 1st Qu.: 45.02 1st Qu.: 2.100   
## Median :0.5380 Median :6.208 Median : 77.50 Median : 3.207   
## Mean :0.5547 Mean :6.285 Mean : 68.57 Mean : 3.795   
## 3rd Qu.:0.6240 3rd Qu.:6.623 3rd Qu.: 94.08 3rd Qu.: 5.188   
## Max. :0.8710 Max. :8.780 Max. :100.00 Max. :12.127   
## rad tax ptratio black   
## Min. : 1.000 Min. :187.0 Min. :12.60 Min. : 0.32   
## 1st Qu.: 4.000 1st Qu.:279.0 1st Qu.:17.40 1st Qu.:375.38   
## Median : 5.000 Median :330.0 Median :19.05 Median :391.44   
## Mean : 9.549 Mean :408.2 Mean :18.46 Mean :356.67   
## 3rd Qu.:24.000 3rd Qu.:666.0 3rd Qu.:20.20 3rd Qu.:396.23   
## Max. :24.000 Max. :711.0 Max. :22.00 Max. :396.90   
## lstat medv   
## Min. : 1.73 Min. : 5.00   
## 1st Qu.: 6.95 1st Qu.:17.02   
## Median :11.36 Median :21.20   
## Mean :12.65 Mean :22.53   
## 3rd Qu.:16.95 3rd Qu.:25.00   
## Max. :37.97 Max. :50.00

set.seed(3876)  
attach(Boston)  
mu\_hat <- mean(medv)  
mu\_hat

## [1] 22.53281

*(b) Provide an estimate of the standard error of μˆ. Interpret this result.*

x.hat <- sd(medv) / sqrt(dim(Boston)[1])  
x.hat

## [1] 0.4088611

It tells us the average amount that this estimate μˆ differs from the actual value of μ. So μˆ differs by 0.488611 times by the actual value of μ.

*(c) Now estimate the standard error of μˆ using the bootstrap. How does this compare to your answer from (b)?*

boot.fn = function(data, index) return(mean(data[index]))  
library(boot)  
bstrap = boot(medv, boot.fn, 1000)  
bstrap

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = medv, statistic = boot.fn, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 22.53281 0.007578063 0.4114338

The bootsrap estimated standard error of μ̂of 0.40881 is very close to the estimate found in (b) of 0.4114.

*(d) Based on your bootstrap estimate from (c), provide a 95 % confidence interval for the mean of medv. Compare it to the results obtained using t.test(Boston$medv).*

t.test(medv)

##   
## One Sample t-test  
##   
## data: medv  
## t = 55.111, df = 505, p-value < 2.2e-16  
## alternative hypothesis: true mean is not equal to 0  
## 95 percent confidence interval:  
## 21.72953 23.33608  
## sample estimates:  
## mean of x   
## 22.53281

ConfidenceInterval.mu.hat <- c(22.53 - 2 \* 0.4114, 22.53 + 2 \* 0.4114)  
ConfidenceInterval.mu.hat

## [1] 21.7072 23.3528

The confidence interval of bootstrap is very close to the one returned test function.

*(e) Based on this dataset, provide an estimate, μˆ med , for the median value of medv in the population.*

med\_hat <- median(medv)  
med\_hat

## [1] 21.2

*(f) We now would like to estimate the standard error of μˆ med . Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error of the median using the bootstrap. Comment on your findings.*

boot.fn = function(data,index) return (median(data[index]))  
boot(medv, boot.fn, 1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = medv, statistic = boot.fn, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 21.2 0.01195 0.3808145

The estimated median value of 21.2 is exactly the same as what was obtained in (e) with a standard error of 0.3927222 which is still quite small when compared to the value of the median.

*(g) Based on this data set, provide an estimate for the tenth percentile of medv in Boston suburbs. Call this quantity μˆ 0.1 .*

percentile\_hat <- quantile(medv, c(0.1))  
percentile\_hat

## 10%   
## 12.75

*(h) Use the bootstrap to estimate the standard error of μ̂ 0.1 Comment on your findings.*

boot.fn = function(data, index) return(quantile(data[index], c(0.1)))  
boot(medv, boot.fn, 1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = medv, statistic = boot.fn, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 12.75 -0.0113 0.5168844

The estimated 10th percentile value of 12.75 is the same to the value obtained in (g) with a standard error of 0.5168 which is quite small when compared to the percentile value.