Week5\_Assignment

Mohit Mehndiratta

28/08/2021

## Question 1

### (a) Upload the Auto dataset and view first few rows.

library(ISLR)  
attach(Auto)  
dim(Auto)

## [1] 392 9

head(Auto)

## mpg cylinders displacement horsepower weight acceleration year origin  
## 1 18 8 307 130 3504 12.0 70 1  
## 2 15 8 350 165 3693 11.5 70 1  
## 3 18 8 318 150 3436 11.0 70 1  
## 4 16 8 304 150 3433 12.0 70 1  
## 5 17 8 302 140 3449 10.5 70 1  
## 6 15 8 429 198 4341 10.0 70 1  
## name  
## 1 chevrolet chevelle malibu  
## 2 buick skylark 320  
## 3 plymouth satellite  
## 4 amc rebel sst  
## 5 ford torino  
## 6 ford galaxie 500

str(Auto)

## 'data.frame': 392 obs. of 9 variables:  
## $ mpg : num 18 15 18 16 17 15 14 14 14 15 ...  
## $ cylinders : num 8 8 8 8 8 8 8 8 8 8 ...  
## $ displacement: num 307 350 318 304 302 429 454 440 455 390 ...  
## $ horsepower : num 130 165 150 150 140 198 220 215 225 190 ...  
## $ weight : num 3504 3693 3436 3433 3449 ...  
## $ acceleration: num 12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...  
## $ year : num 70 70 70 70 70 70 70 70 70 70 ...  
## $ origin : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ name : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 2 ...

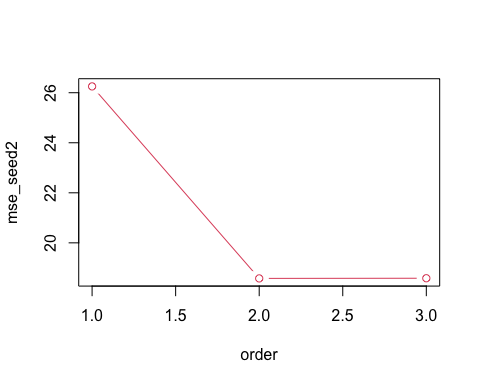
There are 392 observations of 9 variables. All variables are numeric except “Name”. Name is a factor variable.

### (b) Using validation-set approach, select the best model from the polynomial models of order 1, 2 and 3 to to explain mpg in terms of horsepower . (You may need to devide the dataset into two parts; one for training and the other for validation. Use the 3 models of order 1, 2 and 3 and select the model with minimum validation error) Use set.seed(2) to get reproducible results.

set.seed(2)  
tr1 = sample(1:nrow(Auto), nrow(Auto) \* 0.7)  
length(tr1)

## [1] 274

# splitting the training data  
auto\_training\_data = Auto[tr1,]  
# building models  
model1 <- lm(mpg~horsepower, data = Auto, subset = tr1)  
mse1 <- mean((mpg - predict(model1, Auto))[-tr1]^2)  
  
model2 <- lm(mpg~poly(horsepower, 2), data = Auto, subset = tr1)  
mse2 <- mean((mpg - predict(model2, Auto))[-tr1]^2)  
  
model3 <- lm(mpg~poly(horsepower, 3), data = Auto, subset = tr1)  
mse3 <- mean((mpg - predict(model3, Auto))[-tr1]^2)  
  
mse\_seed2 <- c(mse1, mse2, mse3)  
order <- c(1,2,3)  
  
plot(order, mse\_seed2, type="b", col = 2)



Here, we can see that Mean Square Estimate for order 2 has the lower error and it is less complex as compared to order 3. Therefore, Model 2 is our best model.

### (c) Repeat part (b) with set.seed values as 5 and 8. Compare your results.

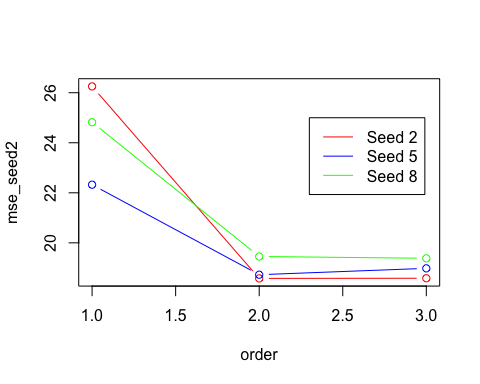
## Now for seed 5  
set.seed(5)  
tr1 = sample(1:nrow(Auto), nrow(Auto) \* 0.7)  
length(tr1)

## [1] 274

# splitting the training data  
auto\_training\_data = Auto[tr1,]  
# building models  
model1 <- lm(mpg~horsepower, data = Auto, subset = tr1)  
mse1 <- mean((mpg - predict(model1, Auto))[-tr1]^2)  
  
model2 <- lm(mpg~poly(horsepower, 2), data = Auto, subset = tr1)  
mse2 <- mean((mpg - predict(model2, Auto))[-tr1]^2)  
  
model3 <- lm(mpg~poly(horsepower, 3), data = Auto, subset = tr1)  
mse3 <- mean((mpg - predict(model3, Auto))[-tr1]^2)  
  
mse\_seed5 <- c(mse1, mse2, mse3)  
  
  
## Now for seed 8  
set.seed(8)  
tr1 = sample(1:nrow(Auto), nrow(Auto) \* 0.7)  
length(tr1)

## [1] 274

# splitting the training data  
auto\_training\_data = Auto[tr1,]  
# building models  
model1 <- lm(mpg~horsepower, data = Auto, subset = tr1)  
mse1 <- mean((mpg - predict(model1, Auto))[-tr1]^2)  
  
model2 <- lm(mpg~poly(horsepower, 2), data = Auto, subset = tr1)  
mse2 <- mean((mpg - predict(model2, Auto))[-tr1]^2)  
  
model3 <- lm(mpg~poly(horsepower, 3), data = Auto, subset = tr1)  
mse3 <- mean((mpg - predict(model3, Auto))[-tr1]^2)  
  
mse\_seed8 <- c(mse1, mse2, mse3)  
  
  
order <- c(1,2,3)  
plot(order, mse\_seed2, type="b", col = "red")  
lines(mse\_seed5, type = "b", col = "blue")  
lines(mse\_seed8, type = "b", col = "green")  
legend(2.3, 25, legend=c("Seed 2", "Seed 5", "Seed 8"),  
 col=c("red", "blue", "green"),lty = 1)



From the above graph, we can see that all the model has slightly different values with different seed values but has similar representation. Seed 2 gives the lowest error value for mpg vs horsepower at order 2.

### (d) Give a drawback of using validation set approach to select the best model?

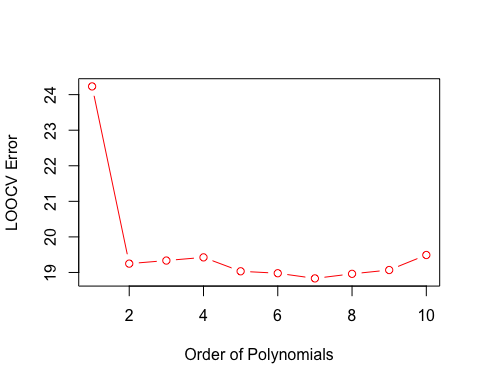
The major drawback of using validation set approach is that when we use different training set/ split we gets different values of our MSE values.

### (e) Using LOOCV method, select the best model from the polynomial models of order 1 to 10. (for mpg in terms of horsepower)

### Using LOOCV  
library(boot)  
poly\_order <- c(1:10)  
cv\_error <- rep(0,10)  
for(i in poly\_order){  
 m <- glm(mpg~poly(horsepower, i), data = Auto)  
 cv\_error[i] <- cv.glm(Auto, m)$delta[1]  
}  
  
cv\_error

## [1] 24.23151 19.24821 19.33498 19.42443 19.03321 18.97864 18.83305 18.96115  
## [9] 19.06863 19.49093

plot(poly\_order, cv\_error, type = "b", xlab = "Order of Polynomials",   
 col = "red", ylab = "LOOCV Error")



From the above graph, we can observe that there is a significant drop of error for order 2 and for the rest of the order there are slight increase and decrease for the error.

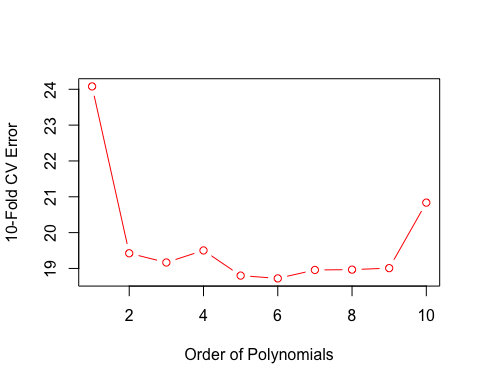
For order 7, the error is the lowest but there is no quite significant improvement in the error as compared to order 2. Therefore, we can use order 2 to make our model less complex and hence this is the best model.

### (f) Use 10-fold cross-validation method for the same model in part (e).

### Using K-Fold CV  
  
cv\_error\_K\_fold <- rep(0,10)  
for(i in 1:10){  
 mod <- glm(mpg~poly(horsepower, i), data = Auto)  
 cv\_error\_K\_fold[i] <- cv.glm(Auto, mod, K = 10)$delta[1]  
}  
cv\_error\_K\_fold

## [1] 24.08050 19.42367 19.16607 19.50262 18.80053 18.72215 18.95710 18.96687  
## [9] 19.00893 20.83688

plot(poly\_order, cv\_error\_K\_fold, type = "b", xlab = "Order of Polynomials",   
 col = "red", ylab = "10-Fold CV Error")



we can also see that there is a significant drop of error for order 2 and for the rest of the order there are slight increase and decrease for the error.

There are lower error for order 5 and 7 as compared to order 2 but those are not much significant in numbers.

Therefore, we can use order 2 to make our model less complex and hence this is the best model.

## Question 2

### Consider linear models with all posible combinations of variables in the Advertising dataset to explain Sales variable.

### Using 10-fold cross-validation select the best model. (use glm function)

First uploading and viewing the data set.

Advertising <- read.csv("../datasets/advertising.csv")  
dim(Advertising)

## [1] 200 4

attach(Advertising)  
str(Advertising)

## 'data.frame': 200 obs. of 4 variables:  
## $ TV : num 230.1 44.5 17.2 151.5 180.8 ...  
## $ Radio : num 37.8 39.3 45.9 41.3 10.8 48.9 32.8 19.6 2.1 2.6 ...  
## $ Newspaper: num 69.2 45.1 69.3 58.5 58.4 75 23.5 11.6 1 21.2 ...  
## $ Sales : num 22.1 10.4 9.3 18.5 12.9 7.2 11.8 13.2 4.8 10.6 ...

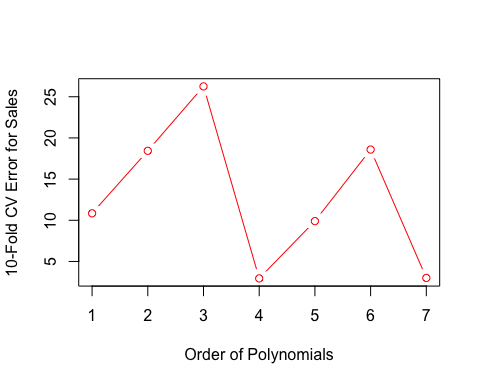
head(Advertising)

## TV Radio Newspaper Sales  
## 1 230.1 37.8 69.2 22.1  
## 2 44.5 39.3 45.1 10.4  
## 3 17.2 45.9 69.3 9.3  
## 4 151.5 41.3 58.5 18.5  
## 5 180.8 10.8 58.4 12.9  
## 6 8.7 48.9 75.0 7.2

There are 200 observations of 4 variables. All variables are numeric.

Now creating the models for all the combination of variables against Sales.

model1 <- glm(Sales~TV, data = Advertising)  
model2 <- glm(Sales~Radio, data = Advertising)  
model3 <- glm(Sales~Newspaper, data = Advertising)  
model4 <- glm(Sales~TV+Radio, data = Advertising)  
model5 <- glm(Sales~TV+Newspaper, data = Advertising)  
model6 <- glm(Sales~Radio+Newspaper, data = Advertising)  
model7 <- glm(Sales~TV+Radio+Newspaper, data = Advertising)  
  
cv\_error\_Sales <- rep(0,7)  
cv\_error\_Sales[1] = cv.glm(Advertising, model1, K = 10)$delta[1]  
cv\_error\_Sales[2] = cv.glm(Advertising, model2, K = 10)$delta[1]  
cv\_error\_Sales[3] = cv.glm(Advertising, model3, K = 10)$delta[1]  
cv\_error\_Sales[4] = cv.glm(Advertising, model4, K = 10)$delta[1]  
cv\_error\_Sales[5] = cv.glm(Advertising, model5, K = 10)$delta[1]  
cv\_error\_Sales[6] = cv.glm(Advertising, model6, K = 10)$delta[1]  
cv\_error\_Sales[7] = cv.glm(Advertising, model7, K = 10)$delta[1]  
  
plot(1:7, cv\_error\_Sales, type = "b", xlab = "Order of Polynomials",   
 col = "red", ylab = "10-Fold CV Error for Sales")



By looking at this graph, the best model is TV + Radio because it has lower error. TV + Radio + Newspaper has also the lowest error but it is not much significantly lower and is more complex than TV + Radio model.

So, we will select the model 4, Sales vs TV + Radio as the best model.

## Question 3

### (a) Generate 20 random numbers using the following R code:

### x = 10\*rexp(20)

set.seed(100)  
x = 10\*rexp(20)

### (b) Calculate the mean of x.

mean(x)

## [1] 7.762101

Mean for the above sample x is **7.762101**.

### (c) Generate 1000 bootstrap samples using the above dataset and calculate the means.

count\_BS <- 1000  
mean\_samples <- rep(0,count\_BS)  
mean\_samples[1] <- mean(x)  
  
for(i in 2:count\_BS) {  
 mean\_samples[i] <- mean(sample(x, replace = TRUE))  
}  
  
mean\_samples

## [1] 7.762101 8.577788 7.991328 5.937333 6.567463 9.999951 6.638912  
## [8] 9.406114 5.793509 7.026296 3.972550 8.131962 9.821821 7.330084  
## [15] 7.535175 9.783117 6.162282 8.692149 5.203706 9.151571 8.977726  
## [22] 6.443510 10.796797 7.853595 6.049840 7.829891 7.081807 8.199763  
## [29] 7.713897 4.610884 10.827877 4.268021 8.878589 6.560727 9.300601  
## [36] 12.998184 10.280431 6.441082 6.263721 9.439128 9.707276 10.321523  
## [43] 8.372421 8.726439 6.184214 8.794553 8.005751 10.564076 7.974452  
## [50] 6.736324 7.040119 7.023563 5.358525 8.356277 8.918972 6.954887  
## [57] 6.287888 6.031327 6.623931 7.118135 7.722979 7.029963 7.079678  
## [64] 5.820102 11.381927 8.604633 8.118863 5.279993 6.915954 9.036776  
## [71] 11.013988 7.482252 7.265260 9.227192 11.182023 9.445121 6.405125  
## [78] 13.626026 7.545253 7.692385 5.892178 8.665812 9.917661 5.749845  
## [85] 10.789030 7.926459 9.453156 9.302299 8.574005 8.813682 6.386329  
## [92] 7.855773 7.743479 7.559802 7.945371 6.515649 8.101266 7.778176  
## [99] 12.302207 9.406938 10.544706 8.211431 8.110574 7.158019 6.538802  
## [106] 5.817379 6.245368 7.543900 5.509443 6.691621 6.851717 6.924677  
## [113] 7.538225 5.756068 7.043722 5.865806 5.094961 6.073103 7.926463  
## [120] 7.162857 5.409626 7.902200 4.516264 8.664949 7.346141 5.860811  
## [127] 7.185812 6.448764 6.808426 6.509931 7.477338 6.957475 8.477760  
## [134] 7.315531 6.558423 7.583559 8.575376 9.308363 8.405448 9.420376  
## [141] 10.834117 8.283568 9.277640 6.987303 6.847793 5.691622 8.128215  
## [148] 6.524899 8.200242 5.836579 5.168013 8.063300 7.069727 7.243004  
## [155] 6.516077 7.302867 6.819375 7.681604 7.620683 7.455485 8.037340  
## [162] 5.969670 6.695689 5.016880 7.300621 5.621421 9.659524 10.815410  
## [169] 6.871843 8.904358 7.155561 7.722516 4.427226 10.206410 8.229758  
## [176] 6.526146 8.338720 8.839200 8.622432 11.321498 7.148037 5.584699  
## [183] 7.220654 9.074400 5.196425 9.881144 7.986474 8.453665 6.900730  
## [190] 4.981293 8.545669 6.738937 7.208234 3.560306 7.138658 8.134436  
## [197] 4.950314 10.888291 8.851351 9.291871 9.387773 6.173664 6.529204  
## [204] 7.686994 5.856149 9.296766 11.983700 8.006289 7.117686 6.041512  
## [211] 6.299175 6.244619 8.420583 11.447100 5.160895 7.609316 9.560264  
## [218] 6.551086 11.509229 10.238365 7.087147 6.663579 5.984819 5.523602  
## [225] 10.791726 10.380332 5.933850 6.098002 9.160464 6.736868 7.665385  
## [232] 6.597335 7.334115 8.011170 8.516628 9.002427 8.350399 4.897758  
## [239] 6.811801 9.384457 8.618273 6.084942 9.128455 8.930863 6.011802  
## [246] 8.321514 7.323468 8.685426 6.740530 7.814648 7.991226 3.724373  
## [253] 7.638492 8.779779 8.237264 7.985541 9.754270 10.561154 7.662096  
## [260] 7.473686 7.716503 10.483359 7.942795 7.688384 9.361071 6.662817  
## [267] 8.131083 7.961400 9.123973 7.366301 10.680272 9.191010 7.162266  
## [274] 9.189284 7.522831 10.278427 10.520318 7.553585 5.968987 8.870155  
## [281] 8.091836 9.956674 8.291886 8.810053 11.151909 6.923833 7.178124  
## [288] 9.939940 9.303927 6.361842 6.187910 7.042229 9.175286 6.638770  
## [295] 6.394837 5.342302 10.392076 8.212743 8.916319 6.243433 5.925517  
## [302] 7.022095 5.610156 11.433861 7.028650 9.306264 6.522062 7.105479  
## [309] 8.037311 11.071162 7.632180 6.925334 7.555773 6.752641 4.730556  
## [316] 8.294439 8.697608 6.055525 8.206201 10.017647 5.097989 8.423212  
## [323] 9.882940 6.612311 9.018149 9.613897 7.468135 7.806145 5.629834  
## [330] 9.180913 9.024573 5.921662 9.664812 8.359314 10.531621 8.122603  
## [337] 11.167266 5.829304 6.673175 7.354442 6.318229 6.429398 6.110695  
## [344] 10.655263 9.875212 7.372604 8.760032 8.493794 7.867130 6.280776  
## [351] 8.211324 7.605635 7.720784 6.019720 9.124869 7.342443 5.361450  
## [358] 7.735748 7.442175 6.939926 6.959059 8.423835 4.534081 9.649682  
## [365] 5.675431 8.197999 9.885240 6.601708 7.298005 10.188164 6.225194  
## [372] 6.993679 8.684516 5.832294 8.411376 4.760859 12.049310 9.301579  
## [379] 5.457583 7.990754 5.761381 10.184414 7.092637 9.248753 8.049814  
## [386] 6.988281 7.098611 8.941277 4.720123 6.116742 6.944646 7.811585  
## [393] 7.633035 6.620862 7.655982 13.272368 6.564051 6.501468 9.788663  
## [400] 8.527893 8.050310 10.010876 6.763495 5.320327 6.001497 10.416077  
## [407] 9.169271 6.720879 7.715273 6.788256 10.949034 6.303480 5.128540  
## [414] 9.031677 8.116977 8.414823 10.651072 8.553334 5.965390 7.329304  
## [421] 8.553277 8.406510 7.797239 8.602009 7.714022 11.205317 5.340334  
## [428] 6.821606 8.540344 7.392186 6.827782 8.397909 5.900667 8.318899  
## [435] 7.932319 8.208678 10.420079 3.853161 8.158935 9.311975 10.318705  
## [442] 7.936155 3.621346 5.788604 5.691884 8.373619 11.181597 5.078731  
## [449] 6.832059 6.638107 8.938796 9.545039 8.554630 7.476683 5.702475  
## [456] 7.982922 5.582696 8.839304 11.045707 7.016962 6.049637 6.938160  
## [463] 5.511442 5.335322 8.521019 10.369666 6.653218 6.652937 7.373510  
## [470] 7.482612 6.763867 5.224904 4.857121 8.694380 5.729489 8.120412  
## [477] 6.880781 6.384068 6.482522 6.733531 5.970531 6.085210 7.048968  
## [484] 10.583915 7.633961 5.739298 9.929903 9.173963 6.821567 9.787730  
## [491] 6.572128 10.194644 9.943224 6.633185 10.375018 7.697461 6.668762  
## [498] 8.025153 7.573778 6.674419 10.106567 7.594199 6.595087 6.201544  
## [505] 8.215271 8.472712 7.162945 10.594351 8.021850 4.492262 6.890863  
## [512] 7.449338 8.807559 4.966551 6.016171 6.339306 5.738614 5.416744  
## [519] 10.872019 8.247442 6.628426 8.102897 4.811819 6.675569 5.961753  
## [526] 10.009129 9.122725 4.565411 7.698290 7.947169 8.869368 8.846101  
## [533] 10.799642 7.521998 7.182684 7.350403 7.811783 4.852015 7.480427  
## [540] 7.077119 7.964058 6.489389 6.245432 9.246758 6.879318 8.041012  
## [547] 10.332834 8.231915 9.384255 6.315896 6.175067 6.003866 8.240961  
## [554] 6.363678 10.958028 7.439152 5.852099 6.341776 11.876112 8.355642  
## [561] 5.092095 7.566243 6.204782 7.713300 9.127531 7.386604 8.712687  
## [568] 5.438868 7.885662 9.354703 8.914917 8.946859 6.120395 7.385056  
## [575] 6.787525 10.099264 10.074428 6.222363 9.130097 6.152848 5.585892  
## [582] 7.038376 9.216631 5.245352 6.259476 11.139247 5.310519 5.356038  
## [589] 7.895667 7.415319 10.307519 6.459727 8.078653 6.537447 8.734979  
## [596] 8.708095 5.990136 5.154672 8.073249 9.993911 7.808277 4.740248  
## [603] 8.419115 9.217894 5.727030 6.688899 6.950779 9.990495 10.581893  
## [610] 13.051339 8.290964 7.993538 5.696644 6.855143 9.847551 8.050925  
## [617] 6.041163 7.397261 7.625127 6.499432 7.018359 8.980037 6.897602  
## [624] 8.158444 5.467772 5.619249 8.917108 7.127548 7.662338 9.993750  
## [631] 8.372983 7.968807 8.369824 7.225292 5.530501 8.299692 6.738940  
## [638] 7.977627 5.037135 7.773820 8.731026 5.164338 12.735983 5.963145  
## [645] 7.057555 7.773124 8.610710 10.419180 4.750193 9.489167 11.350094  
## [652] 7.290142 6.341547 7.729733 11.951618 8.335786 7.025792 9.406264  
## [659] 8.864654 4.894576 7.123072 10.010467 7.094450 12.161260 9.207517  
## [666] 7.078015 9.920935 8.487626 8.994771 8.240851 7.375409 7.837869  
## [673] 7.630098 6.442196 7.143037 8.597501 5.738561 5.829716 6.104893  
## [680] 10.151119 7.113270 10.593876 9.027357 10.118022 8.698224 9.093621  
## [687] 4.832812 8.107430 8.118376 9.205431 7.883713 7.053403 7.369018  
## [694] 8.031845 4.110598 7.954652 6.706384 10.650741 7.917604 5.362220  
## [701] 6.731107 7.574118 8.625741 5.372010 8.521597 8.451130 6.451589  
## [708] 7.284193 5.960794 9.654182 8.060701 6.403576 12.456593 9.340022  
## [715] 4.263203 9.588885 8.815394 4.583209 9.229396 7.632717 8.772674  
## [722] 9.533542 8.918171 7.568812 10.125135 8.524722 10.119458 9.287468  
## [729] 8.001280 6.421528 8.030268 8.646838 8.643689 5.582522 7.609245  
## [736] 7.984632 6.064969 6.648581 6.118235 9.183482 6.625471 5.010628  
## [743] 4.864892 6.116458 10.065620 8.652507 4.843074 10.429284 5.397989  
## [750] 7.095893 10.904356 7.722553 6.090176 8.303314 5.813327 8.685345  
## [757] 9.289898 5.434131 8.628715 6.171754 6.141552 8.063891 10.183577  
## [764] 7.708318 8.059503 12.761888 8.073289 7.413744 8.284799 8.241749  
## [771] 11.595720 5.812660 5.559823 5.660946 6.973778 7.662880 6.801130  
## [778] 9.923365 7.119832 8.589424 9.415551 6.969307 7.376878 8.841992  
## [785] 7.363857 8.694163 10.805964 9.598725 7.112505 9.360754 6.907477  
## [792] 5.058200 8.880182 6.289303 5.119418 7.876760 11.372832 7.029407  
## [799] 7.835453 11.437377 7.217016 9.075896 6.908920 7.000885 6.747543  
## [806] 8.237044 6.407881 7.266758 7.404311 8.928394 7.919340 7.442679  
## [813] 7.716635 4.937951 10.400075 4.657055 7.795908 7.553333 7.500555  
## [820] 6.117880 8.484347 8.564127 8.948028 9.400397 9.052568 7.712800  
## [827] 9.963596 8.971031 8.710147 5.394342 6.020671 9.755422 8.483081  
## [834] 6.706248 10.168909 11.657603 6.397096 7.635229 6.352852 6.590955  
## [841] 5.889577 10.725963 6.463248 9.023572 5.827826 6.145838 8.164453  
## [848] 6.800567 5.567221 6.053600 8.300828 6.744902 7.773263 7.531289  
## [855] 5.968025 7.176191 11.223623 7.013188 8.772368 8.166197 8.343814  
## [862] 8.081309 8.621844 6.796989 8.296121 8.031854 5.964004 9.724548  
## [869] 8.578250 7.967702 13.272900 6.763958 8.910770 8.386059 5.243609  
## [876] 5.876543 6.262502 9.068119 6.563090 7.936517 7.802711 12.002736  
## [883] 8.263693 7.768738 8.734798 11.944924 8.451290 7.480653 5.718997  
## [890] 7.605741 5.653435 5.707855 7.522973 7.946004 8.375286 7.821561  
## [897] 7.810516 8.032095 10.071697 5.056460 9.914260 7.598621 7.212966  
## [904] 8.180317 9.718319 6.802188 7.158586 6.764727 8.455127 7.052044  
## [911] 8.186931 5.366453 7.387736 7.235503 7.353516 5.695514 8.170190  
## [918] 8.244945 6.356895 9.506076 5.692223 4.862863 7.591819 6.280419  
## [925] 7.180172 8.690266 7.822488 6.891495 6.248507 7.155413 9.529336  
## [932] 4.390554 6.719322 6.821564 9.519406 7.373728 7.486846 9.156049  
## [939] 8.362287 8.248036 8.201806 7.930221 6.061233 8.882228 8.035915  
## [946] 6.793559 9.834642 9.582833 7.390434 5.106908 8.753649 5.773225  
## [953] 9.092829 7.410035 4.519602 8.415407 10.471282 8.805505 9.415081  
## [960] 10.507755 5.356930 8.477700 6.925971 9.306367 6.511019 8.903425  
## [967] 5.892746 7.918928 8.255092 5.892792 10.436769 9.033167 8.591876  
## [974] 5.132736 6.533959 7.430657 8.127337 6.049879 8.821488 7.364225  
## [981] 5.916944 9.109740 8.694697 7.802224 6.062460 6.887911 9.197880  
## [988] 9.516192 7.973483 6.689162 7.806750 6.262550 7.479581 9.461654  
## [995] 6.839146 7.218343 5.664206 7.885113 7.819349 7.302978

Here we received 1000 samples of means for the sample x.

### (d) Draw the histogram of the means calculated in part (c).

hist(mean\_samples, main = "Mean of 1000 bootstrapped samples", xlab = "Mean samples")

