HW5_arflowers

Anna Flowers

11/8/2021

Homework 5

Problem 2

Part A

The original code supplied does not work because when you use the cbind command on a data frame, it does not automatically make the column names the same as the original vector names. This means that when using the lm command in the loop and calling the logapple08 and logrm08 variables, they are both the original values instead of the resampled ones. So every iteration will give an estimate the same as the original one, since the command is always on the original data.

Part B

```r
set.seed(1834)
n <- 100
x <- c(seq(0.5, 10, 0.5), seq(0.5, 10, 0.5), seq(0.5, 10, 0.5))
y <- 1 + 2*x + rnorm(60)
beta.boot <- matrix(0, n, 2)
system.time({
for(i in 1:n) {
  samp <- sample(seq(1, 60, by = 1), 60, replace = TRUE)
  boot.set <- data.frame(xb = x[samp], yb = y[samp])
  fit <- lm(yb ~ xb, boot.set)
  beta.boot[i,] <- fit$coefficients
}})
```

```
##    user  system elapsed
##   0.077   0.004   0.081
```

```r
mean(beta.boot[,1])
```

```
## [1] 1.531828
```

```r
mean(beta.boot[,2])
```

```
## [1] 1.904912
```

Problem 3

Part A

From the graph it appears there are 4 roots.

```r
f1 <- function(x) {
  y <- x - (3^(x) - sin(x) + cos(5*x) + x^2 - 1.5) / (log(3)*3^(x) - cos(x) - 5*sin(5*x) + 2*x)
  n <- 0
  while(abs(x - y) > .00001 && n < 100) {
    x <- y
    y <- x - (3^x - sin(x) + cos(5*x) + x^2 - 1.5) / (log(3)*3^x - cos(x) - 5*sin(5*x) + 2*x)
    n <- n + 1
  }
  if(n < 100) {return(y)}
  return(NA)
}
```

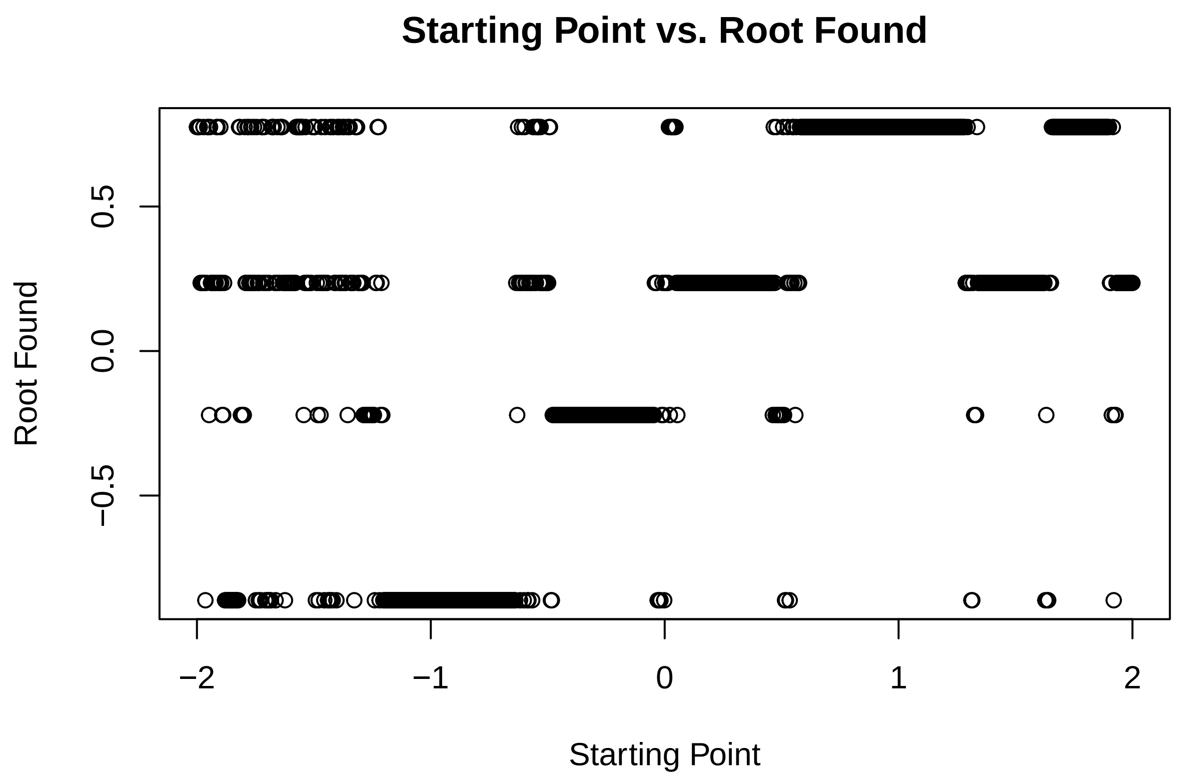
If the function does not converge in 100 iterations, it returns a value of NA.

Part B

```r
vec <- cbind(seq(-2, 2, length.out = 1000))
system.time({solutions <- apply(vec, 1, f1)})
```

```
##    user  system elapsed
##   0.020   0.001   0.021
```

```r
plot(vec, solutions, xlab = "Starting Point", ylab = "Root Found", main = "Starting Point vs. Root Foun
```



Problem 4

Part A

```r
n <- length(x)
f2 <- function(beta1, beta2) {
  x <- c(seq(0.5, 10, 0.5), seq(0.5, 10, 0.5), seq(0.5, 10, 0.5))
  y <- 1 + 2*x + rnorm(60)
  yhat <- beta1 + x * beta2
  MSE2 <- sum((y - yhat)^2) / n
  MSE1 <- MSE2 - 1
  step <- 0.00001
  tolerance <- 0.000001
  count <- 0
  while(abs(MSE2 - MSE1) > tolerance && count < 50000) {
    MSE1 <- MSE2
    beta1 <- beta1 + step * ((2/n) * (sum(y - yhat)))
    beta2 <- beta2 + step * ((2/n) * (sum((y - yhat) * x)))
    yhat <- beta1 + x * beta2
    MSE2 <- sum((y - yhat)^2) / n
    count <- count + 1
  }
  return(cbind(beta1, beta2))
}
```

Part B

My algorithm stops once the mean square error of successive iterations are within 0.000001 of each other or once the algorithm repeats 50000 times, whichever comes first. We could change the stopping criterion so that it stops within a certain number of decimal places of the actual value, but with such a small stepping size this would take a long time if we started with a value far from beta. So this is not a good way to run this algorithm. The best guess for starting would be the true values for each beta.

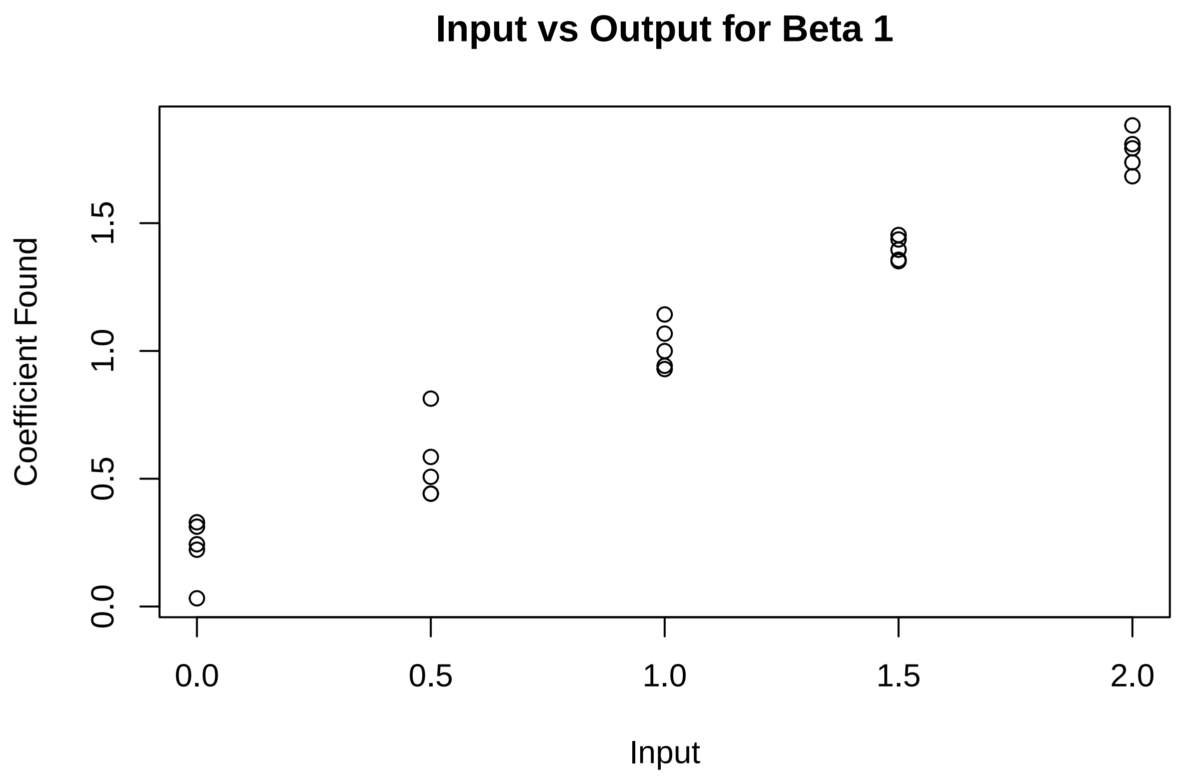
Part C

```
f3 <- function(beta1, beta2) {
  x <- c(seq(0.5, 10, 0.5), seq(0.5, 10, 0.5), seq(0.5, 10, 0.5))
  y <- 1 + 2*x + rnorm(60)
  yhat <- beta1 + x * beta2
  MSE2 <- sum((y - yhat)^2) / n
  MSE1 <- MSE2 - 1
  step <- 0.0000001
  tolerance <- 0.000000001
  count <- 0
  while(abs(MSE2 - MSE1) > tolerance && count < 5000000) {
    MSE1 <- MSE2
    beta1 <- beta1 + step * ((2/n) * (sum(y - yhat)))
    beta2 <- beta2 + step * ((2/n) * (sum(x * (y - yhat))))
    yhat <- beta1 + x * beta2
    MSE2 <- sum((y - yhat)^2) / n
    count <- count + 1
  }
  return(cbind(beta1, beta2))
}
vec <- expand.grid(seq(0,2,0.5), seq(1,3,0.5))
system.time(solutions <- mapply(f3, vec[,1], vec[,2]))
```

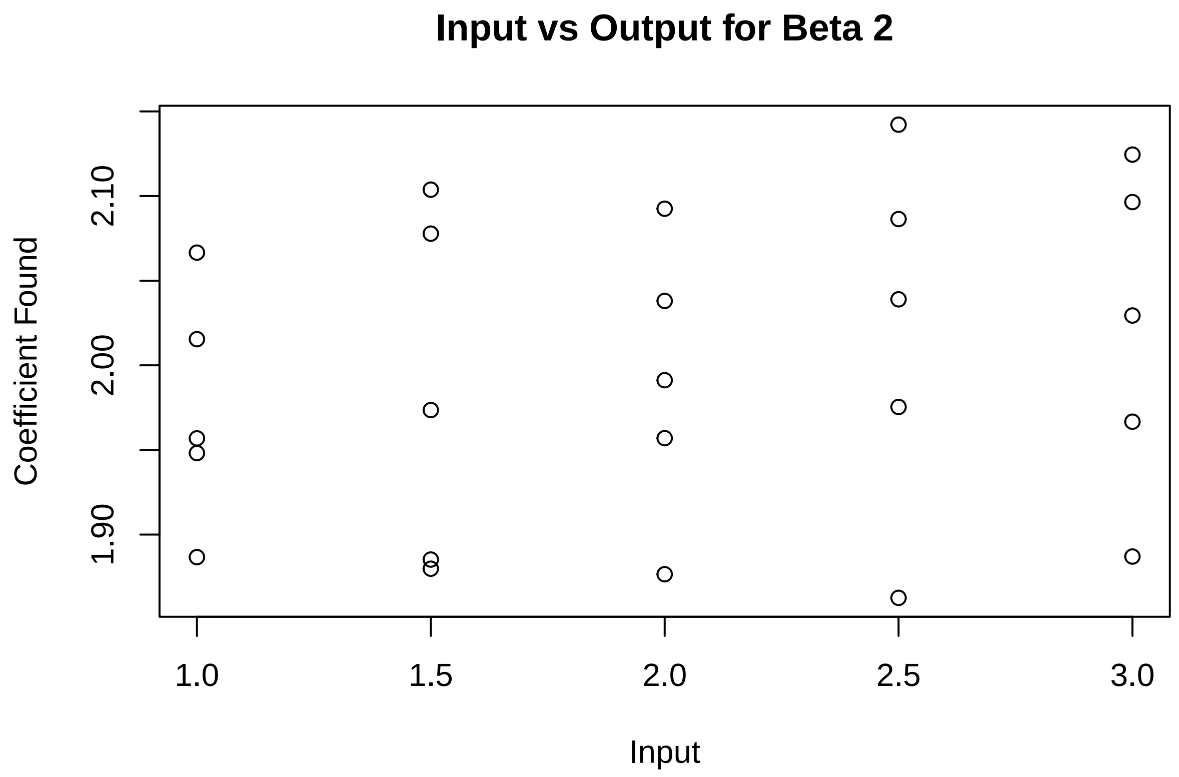
```
##     user   system  elapsed
## 2260.416   30.770 2292.750
```

Part D

```r
plot(vec[,1], solutions[1,], xlab = "Input", ylab = "Coefficient Found", main = "Input vs Output for Be
```



```r
plot(vec[,2], solutions[2,], xlab = "Input", ylab = "Coefficient Found", main = "Input vs Output for Be
```



This algorithm is slow and inefficient. I was unable to run 1000 iterations of the function on my personal computer due to time constraints.