Statistical Learning and Linear Regression

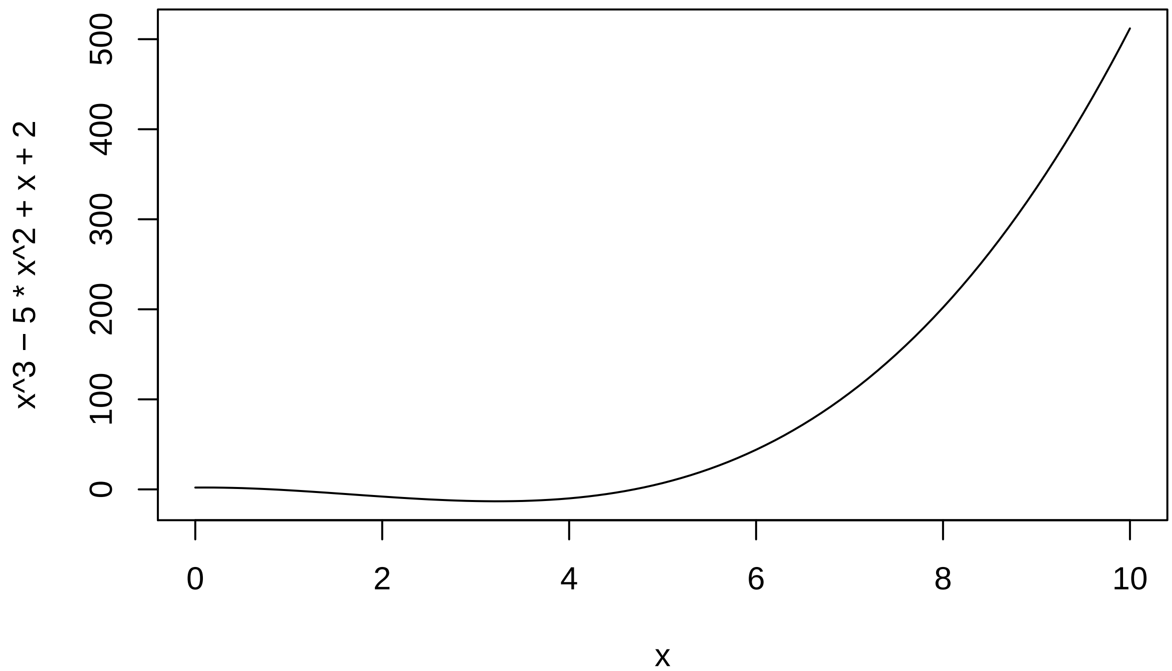
Naeem Chowdhury and Simona Rahi

2/14/2020

1. Reproduce slide 18, using R markdown.

- Let $f(x) = x^3 - 5x^2 + x + 2$. That's the truth. Draw $f(x)$ in range $x \in (0, 10)$.

```
curve(x^3 - 5*x^2 + x + 2, from = 0, to = 10)
```



- Make 10 training sets. To make each training set, pick 15 random values of x in the range. Generate 15 responses $f(x) + \epsilon$, with $\epsilon \sim \mathbb{N}(0, 2)$.

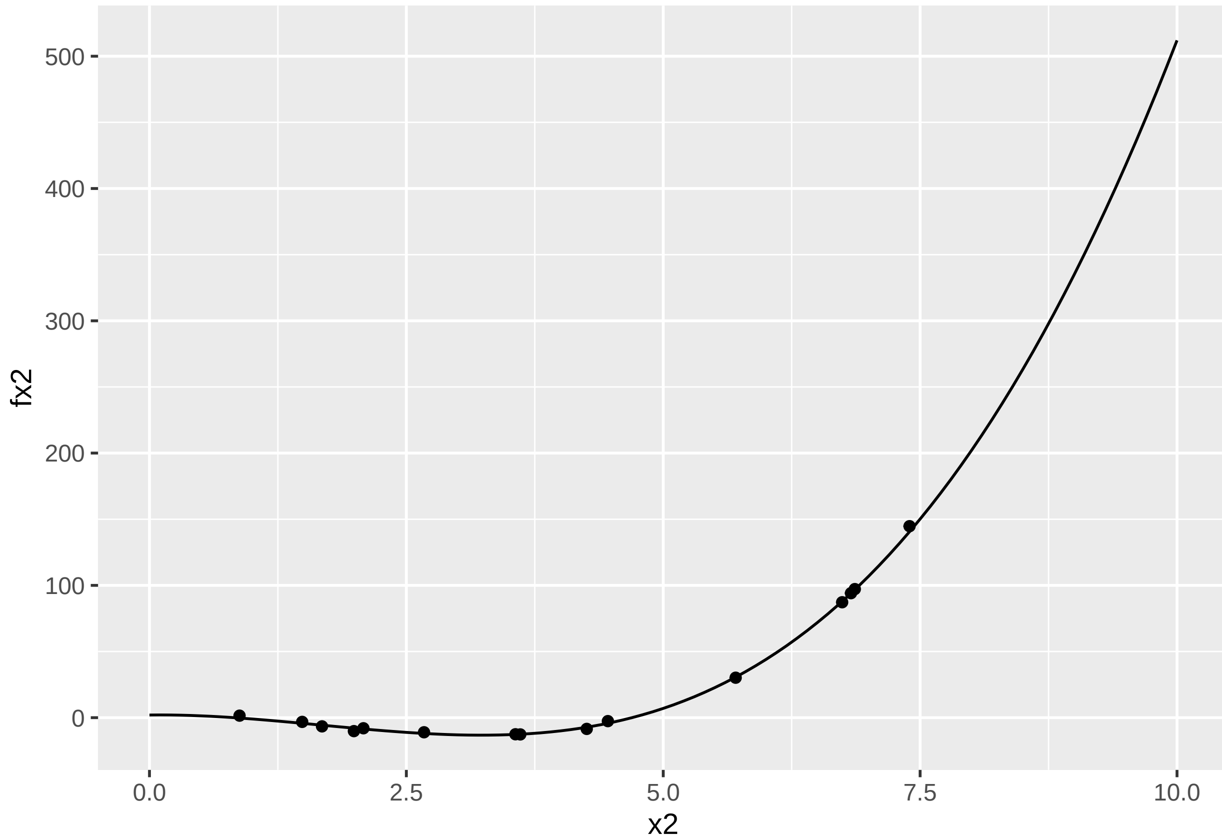
Produce training data, and resulting function values.

```
vecG <- c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
trainingData <- data.frame(id = vecG, stringsAsFactors=TRUE)
# Creating each traing seet
for (i in (1:10)){
```

```
  vecFx <- vector() # vector of random variables
  vecX <- vector() # vector of randomly generated x
  randEpsilon <- rnorm(15, 0, 2)
  randX <- runif(15, 0, 10)
  for(j in (1:15)){


    fx <- ((randX[j])^3) - (5*((randX[j])^2)) + randX[j] + 2 + randEpsilon[j]

    vecX[j] <- randX[j]
    vecFx[j] <- fx
  }
  trainingData <- cbind(trainingData, data.frame(name = vecX))
  trainingData <- cbind(trainingData, data.frame(name2 = vecFx))
}
colnames(trainingData) <- c("dummy", "x1", "fx1", "x2", "fx2", "x3", "fx3", "x4", "fx4", "x5", "fx5", ":
trainingData <- trainingData[,-1]

fun.1 <- function(x) x^3 - 5*x^2 + x + 2
```

Showing a sample plot

```
fun.1 <- function(x) x^3 - 5*x^2 + x + 2

ggplot(data = trainingData, mapping = aes(x = x2, y = fx2)) +
  geom_point() +
  stat_function(fun = fun.1) + xlim(0,10)
```




```r
## Creating Models and Calculating MSE for training and test
dummyVec <- c(1,2,3,4,5)
polynomialAvgs <- data.frame(id = dummyVec, stringsAsFactors=TRUE)
## Polynomial 1
polyMSEs <- data.frame(id = dummyVec, stringsAsFactors=TRUE)
modelDF <- data.frame(id = dummyVec, stringsAsFactors=TRUE)
testMSEs <- c()
trainedModels <- c()
avgMSEs <- c()
for (i in (1:10)){ # For each training set

  # build polynomial model
  xName <- paste("x",i, sep="")
  fxName <- paste("fx",i, sep="")
  trainingMSEs <- c()
  trainedModels <- c()

  for (k in (1:5)){
    lm.obj <- lm(trainingData[,(i*2)] ~ poly(trainingData[,i*2-1],k))

    # calculate the singlular training MSE for the model
    traingMSE <- mean((trainingData[,(i*2)] - lm.obj$fitted.values)^2)
    trainingMSEs <- append(trainingMSEs, traingMSE)
    trainedModels <- append(trainedModels, lm.obj$fitted.values)
  }
```

```
  polyMSEs <- cbind(polyMSEs, trainingMSEs)
  modelDF <- cbind(modelDF, trainedModels)

    #for (j in (1:10)){ # For each testing set

      # Run the testing data set on the model
      #linear.MSE <- mean((trainingData[,(i*2)] - predict(lm.obj,Auto.test))^2)

      # Calucluate the testing MSE
}

# colnames(polyMSEs) <- c('degree','trainMSE1','trainMSE2','trainMSE3','trainMSE4','trainMSE5','trainMS

# polyMSEs <- polyMSEs %>% mutate(avgMSE = Reduce("+",.)/length(.))

# colnames(modelDF) <- c('degree','mod1','mod2','mod3','mod4','mod5','mod6','mod7','mod8','mod9','mod10
```