Exercise 1

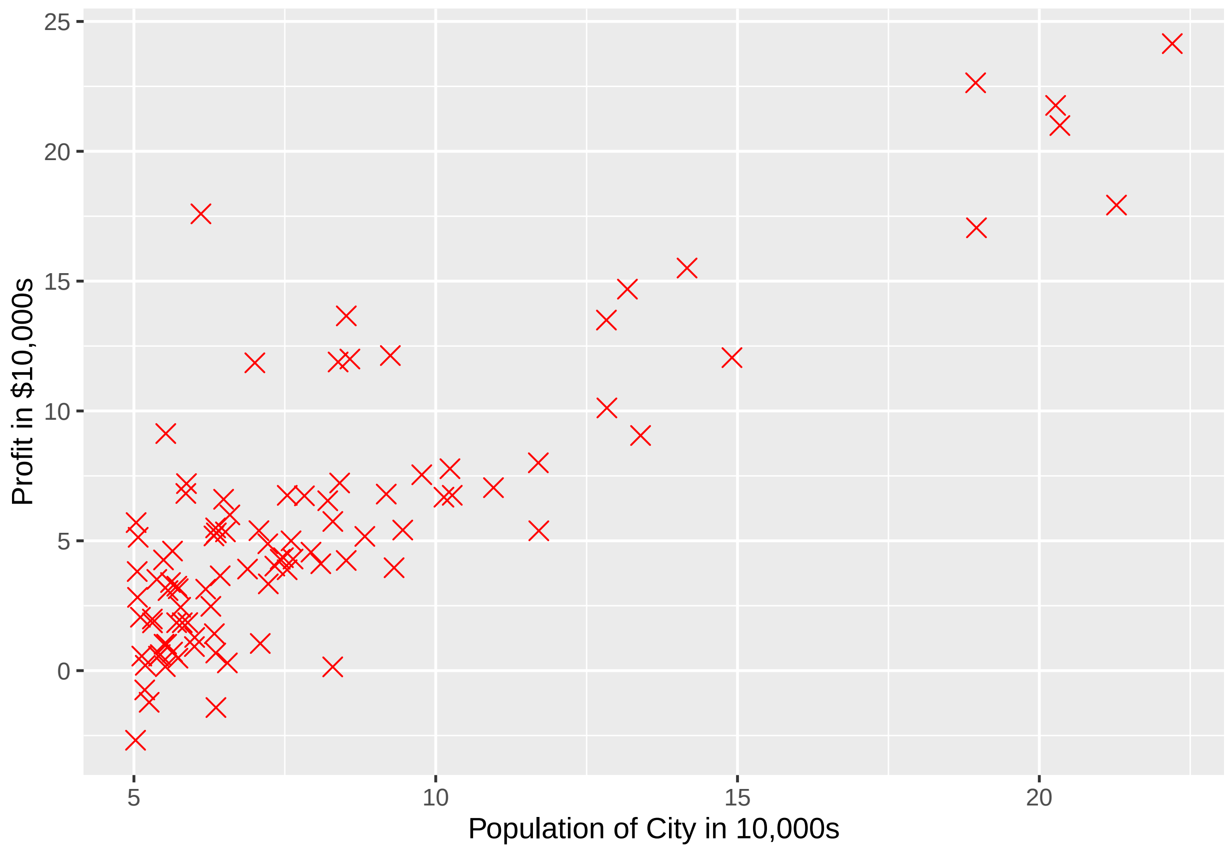
Jai Broome

May 21, 2016

```r
require(ggplot2)
require(dplyr)
require(knitr)
```

```r
q1data <- read.table("../data/ex1data1.txt",
                     sep = ",",
                     col.names = c("population", "profit"))
q1data$ones <- 1
q1data <- q1data %>% select(ones, population, profit)
X <- select(q1data, -profit)
y <- q1data$profit
m <- length(y)
theta <- rep(0, times = 2)
```

```r
g1 <- ggplot(q1data, aes(x = population, y = profit)) +
    geom_point(shape = 4, color = "red", size = 3) +
    xlab("Population of City in 10,000s") +
    ylab("Profit in $10,000s")

g1
```



```
iterations <- 1500
alpha <- 0.01
```

Functions are defined in a separate script so they can be accessed for other assignments. See `?knitr::read_chunk`

```
read_chunk("ex1_chunks.R")
```

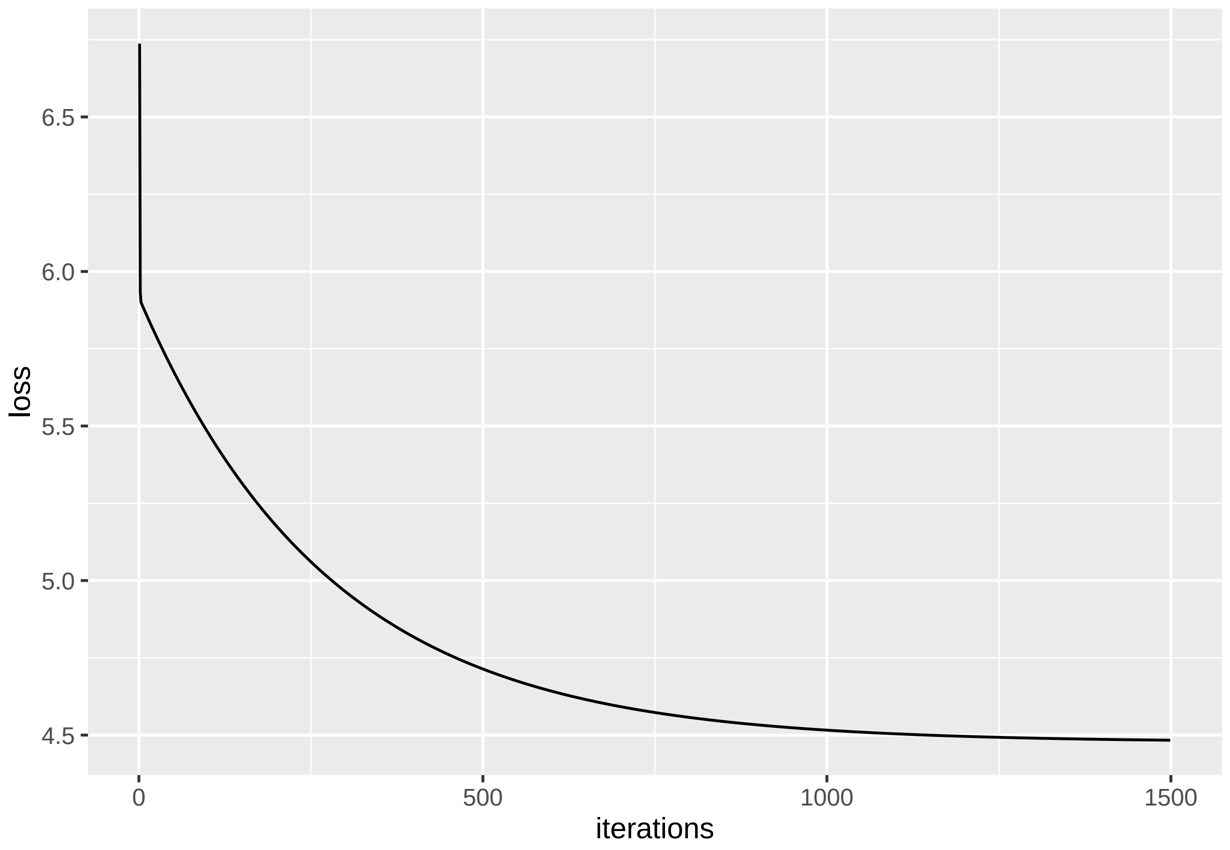
```
## I should go through and make m an argument to these functions
computeCost <- function(X, y, theta, lambda = 0){
    pred <- theta %*% t(X)
    sqError <- (pred - y) ^ 2
    cost <- (1 / (2 * m)) * sum(sqError)
    reg <- (lambda / (2 * m)) * theta[-1]^2
    J <- cost + reg

    gradient <- vector()
    for(j in 1:length(theta)){
        gradient <- c(gradient, ((1 / m) * sum((pred - y) * X[, j]) + lambda * theta[j] / m))
    }
    return(list(J=J, gradient=gradient))
}
```

```
gradStep <- function(thetaj, alpha, gradj){
    thetaj - alpha * gradj
}
```

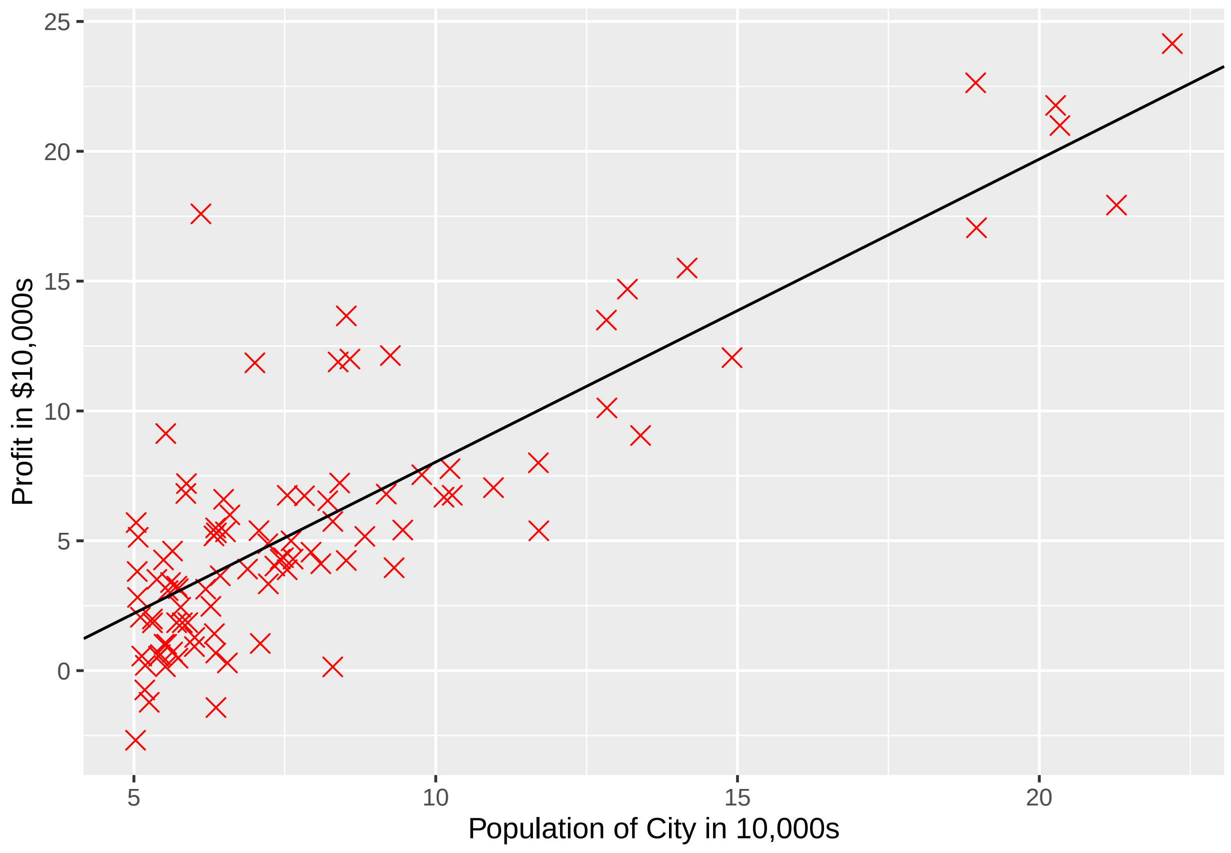
```r
gradientDescent <- function(X, y, theta, alpha, iterations, lambda = 0){
    if(length(theta) != ncol(X)){stop("theta and X are nonconformable")}
    J_history <- data.frame()
    for (iteration in 1:iterations){

        a <- computeCost(X = X, y = y, theta = theta, lambda = lambda)
        J <- a$J
        gradient <- a$gradient
        J_history <- rbind(J_history, c(J, iteration - 1, theta))

        thetaTemp <- vector()
        for(j in 1:length(theta)){
            thetaTemp <- c(thetaTemp, gradStep(theta[j], alpha, gradient[j]))
        }
        theta <- thetaTemp
    }
    # for final iteration
    J <- computeCost(X = X, y = y, theta = theta, lambda = lambda)$J
    J_history <- rbind(J_history, c(J, iteration, theta))
    thetanames <- rep("theta", times = length(theta) - 1)
    for(i in length(thetanames)){thetanames[i] <- paste("theta", i, sep = "")}
    colnames(J_history) <- c("loss", "iterations", "theta0", thetanames)
    return(J_history)
}
```

```r
jhist <- gradientDescent(X, y, theta, alpha, iterations)
jhist <- jhist[2:(nrow(jhist) - 1),] #makes graph more interpretible
```

```r
ggplot(jhist, aes(x = iterations, y = loss)) + geom_line()
```



```r
g1 + geom_abline(slope = tail(jhist$theta1, n = 1), intercept = tail(jhist$theta0, n = 1))
```


25
20
15
10
5
0
Profit in $10,000s
5
10
15
20
Population of City in 10,000s