

Lab 4

Kennly Weerasinghe

11:59PM March 11, 2021

Load up the famous iris dataset. We are going to do a different prediction problem. Imagine the only input x is Species and you are trying to predict y which is Petal.Length. A reasonable prediction is the average petal length within each Species. Prove that this is the OLS model by fitting an appropriate `lm` and then using the `predict` function to verify.

```
data(iris)
mod = lm(Petal.Length ~ Species, iris)
mean(iris$Petal.Length[iris$Species == "setosa"] )

## [1] 1.462
mean(iris$Petal.Length[iris$Species == "versicolor"] )

## [1] 4.26
mean(iris$Petal.Length[iris$Species == "virginica"] )

## [1] 5.552
predict(mod, data.frame(Species = c("setosa")))

##      1
## 1.462
predict(mod, data.frame(Species = c("versicolor")))

##      1
## 4.26
predict(mod, data.frame(Species = c("virginica")))

##      1
## 5.552
```

Construct the design matrix with an intercept, X , without using `model.matrix`.

```
X = cbind(1, iris$Species == "versicolor", iris$Species == "virginica")
head(X)

##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    1    0    0
## [3,]    1    0    0
## [4,]    1    0    0
## [5,]    1    0    0
## [6,]    1    0    0
```

Find the hat matrix H for this regression.

```
H = X %%% solve(t(X) %%% X) %%% t(X)
Matrix :: rankMatrix(H)
```

```
## [1] 3
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 3.330669e-14
```

```
#head(H)
```

Verify this hat matrix is symmetric using the `expect_equal` function in the package `testthat`.

```
pacman :: p_load(testthat)
expect_equal(H, t(H))
```

Verify this hat matrix is idempotent using the `expect_equal` function in the package `testthat`.

```
expect_equal(H, H %%% H)
```

Using the `diag` function, find the trace of the hat matrix.

```
sum(diag(H))
```

```
## [1] 3
```

It turns out the trace of a hat matrix is the same as its rank! But we don't have time to prove these interesting and useful facts..

For masters students: create a matrix X_{\perp} .

```
I = diag(nrow(H))
X_perp = (I - H) %%% X
X_perp
```

```
##           [,1]      [,2]      [,3]
## [1,] -7.979728e-16  0.000000e+00  0.000000e+00
## [2,] -7.979728e-16  0.000000e+00  0.000000e+00
## [3,] -7.979728e-16  0.000000e+00  0.000000e+00
## [4,] -7.979728e-16  0.000000e+00  0.000000e+00
## [5,] -7.979728e-16  0.000000e+00  0.000000e+00
## [6,] -7.979728e-16  0.000000e+00  0.000000e+00
## [7,] -6.869505e-16  0.000000e+00  0.000000e+00
## [8,] -6.869505e-16  0.000000e+00  0.000000e+00
## [9,] -6.869505e-16  0.000000e+00  0.000000e+00
## [10,] -6.869505e-16  0.000000e+00  0.000000e+00
## [11,] -6.869505e-16  0.000000e+00  0.000000e+00
## [12,] -6.869505e-16  0.000000e+00  0.000000e+00
## [13,] -6.869505e-16  0.000000e+00  0.000000e+00
## [14,] -6.869505e-16  0.000000e+00  0.000000e+00
## [15,] -6.869505e-16  0.000000e+00  0.000000e+00
## [16,] -6.869505e-16  0.000000e+00  0.000000e+00
## [17,] -6.869505e-16  0.000000e+00  0.000000e+00
## [18,] -6.869505e-16  0.000000e+00  0.000000e+00
## [19,] -6.869505e-16  0.000000e+00  0.000000e+00
## [20,] -6.869505e-16  0.000000e+00  0.000000e+00
## [21,] -6.869505e-16  0.000000e+00  0.000000e+00
```

```

## [22,] -6.869505e-16 0.000000e+00 0.000000e+00
## [23,] -6.869505e-16 0.000000e+00 0.000000e+00
## [24,] -6.869505e-16 0.000000e+00 0.000000e+00
## [25,] -6.869505e-16 0.000000e+00 0.000000e+00
## [26,] -6.869505e-16 0.000000e+00 0.000000e+00
## [27,] -6.869505e-16 0.000000e+00 0.000000e+00
## [28,] -6.869505e-16 0.000000e+00 0.000000e+00
## [29,] -6.869505e-16 0.000000e+00 0.000000e+00
## [30,] -6.869505e-16 0.000000e+00 0.000000e+00
## [31,] -6.869505e-16 0.000000e+00 0.000000e+00
## [32,] -6.869505e-16 0.000000e+00 0.000000e+00
## [33,] -6.869505e-16 0.000000e+00 0.000000e+00
## [34,] -6.869505e-16 0.000000e+00 0.000000e+00
## [35,] -6.869505e-16 0.000000e+00 0.000000e+00
## [36,] -6.869505e-16 0.000000e+00 0.000000e+00
## [37,] -6.869505e-16 0.000000e+00 0.000000e+00
## [38,] -6.869505e-16 0.000000e+00 0.000000e+00
## [39,] -6.869505e-16 0.000000e+00 0.000000e+00
## [40,] -6.869505e-16 0.000000e+00 0.000000e+00
## [41,] -6.869505e-16 0.000000e+00 0.000000e+00
## [42,] -6.869505e-16 0.000000e+00 0.000000e+00
## [43,] -6.869505e-16 0.000000e+00 0.000000e+00
## [44,] -7.008283e-16 0.000000e+00 0.000000e+00
## [45,] -7.147061e-16 0.000000e+00 0.000000e+00
## [46,] -7.285839e-16 0.000000e+00 0.000000e+00
## [47,] -7.424616e-16 0.000000e+00 0.000000e+00
## [48,] -7.563394e-16 0.000000e+00 0.000000e+00
## [49,] -7.667478e-16 0.000000e+00 0.000000e+00
## [50,] -7.771561e-16 0.000000e+00 0.000000e+00
## [51,] -3.330669e-16 -5.551115e-16 0.000000e+00
## [52,] -3.330669e-16 -5.551115e-16 0.000000e+00
## [53,] -3.330669e-16 -5.551115e-16 0.000000e+00
## [54,] -3.330669e-16 -4.440892e-16 0.000000e+00
## [55,] -3.330669e-16 -4.440892e-16 0.000000e+00
## [56,] -3.330669e-16 -4.440892e-16 0.000000e+00
## [57,] -2.220446e-16 -4.440892e-16 0.000000e+00
## [58,] -2.220446e-16 -3.330669e-16 0.000000e+00
## [59,] -2.220446e-16 -3.330669e-16 0.000000e+00
## [60,] -1.110223e-16 -3.330669e-16 0.000000e+00
## [61,] -1.110223e-16 -3.330669e-16 0.000000e+00
## [62,] -1.110223e-16 -3.330669e-16 0.000000e+00
## [63,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [64,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [65,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [66,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [67,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [68,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [69,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [70,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [71,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [72,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [73,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [74,] -1.110223e-16 -2.220446e-16 0.000000e+00
## [75,] -1.110223e-16 -2.220446e-16 0.000000e+00

```

```

## [76,] -5.551115e-17 -2.220446e-16 0.000000e+00
## [77,] 0.000000e+00 -2.220446e-16 0.000000e+00
## [78,] 0.000000e+00 -2.220446e-16 0.000000e+00
## [79,] 0.000000e+00 -2.220446e-16 0.000000e+00
## [80,] 0.000000e+00 -2.220446e-16 0.000000e+00
## [81,] 0.000000e+00 -2.220446e-16 0.000000e+00
## [82,] 0.000000e+00 -2.220446e-16 0.000000e+00
## [83,] 0.000000e+00 -2.220446e-16 0.000000e+00
## [84,] 0.000000e+00 -2.220446e-16 0.000000e+00
## [85,] 0.000000e+00 -2.220446e-16 0.000000e+00
## [86,] 0.000000e+00 -2.220446e-16 0.000000e+00
## [87,] 0.000000e+00 -2.220446e-16 0.000000e+00
## [88,] -2.775558e-17 -2.498002e-16 0.000000e+00
## [89,] -5.551115e-17 -2.775558e-16 0.000000e+00
## [90,] -8.326673e-17 -3.053113e-16 0.000000e+00
## [91,] -1.110223e-16 -3.330669e-16 0.000000e+00
## [92,] -1.387779e-16 -3.608225e-16 0.000000e+00
## [93,] -1.665335e-16 -3.885781e-16 0.000000e+00
## [94,] -1.942890e-16 -4.163336e-16 0.000000e+00
## [95,] -2.220446e-16 -4.440892e-16 0.000000e+00
## [96,] -2.498002e-16 -4.718448e-16 0.000000e+00
## [97,] -2.706169e-16 -4.926615e-16 0.000000e+00
## [98,] -2.914335e-16 -5.134781e-16 0.000000e+00
## [99,] -3.122502e-16 -5.342948e-16 0.000000e+00
## [100,] -3.330669e-16 -5.551115e-16 0.000000e+00
## [101,] -5.689893e-16 0.000000e+00 -7.910339e-16
## [102,] -5.689893e-16 0.000000e+00 -7.910339e-16
## [103,] -5.689893e-16 0.000000e+00 -7.910339e-16
## [104,] -5.689893e-16 0.000000e+00 -7.910339e-16
## [105,] -5.689893e-16 0.000000e+00 -7.910339e-16
## [106,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [107,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [108,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [109,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [110,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [111,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [112,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [113,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [114,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [115,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [116,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [117,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [118,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [119,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [120,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [121,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [122,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [123,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [124,] -5.689893e-16 0.000000e+00 -6.800116e-16
## [125,] -5.134781e-16 0.000000e+00 -6.800116e-16
## [126,] -4.579670e-16 0.000000e+00 -6.800116e-16
## [127,] -4.579670e-16 0.000000e+00 -6.800116e-16
## [128,] -4.579670e-16 0.000000e+00 -6.800116e-16
## [129,] -4.579670e-16 0.000000e+00 -6.800116e-16

```

```
## [130,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [131,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [132,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [133,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [134,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [135,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [136,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [137,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [138,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [139,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [140,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [141,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [142,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [143,] -4.579670e-16  0.000000e+00 -6.800116e-16
## [144,] -4.718448e-16  0.000000e+00 -6.938894e-16
## [145,] -4.857226e-16  0.000000e+00 -7.077672e-16
## [146,] -4.996004e-16  0.000000e+00 -7.216450e-16
## [147,] -5.134781e-16  0.000000e+00 -7.355228e-16
## [148,] -5.273559e-16  0.000000e+00 -7.494005e-16
## [149,] -5.412337e-16  0.000000e+00 -7.632783e-16
## [150,] -5.551115e-16  0.000000e+00 -7.771561e-16
```

Using the hat matrix, compute the \hat{y} vector and using the projection onto the residual space, compute the e vector and verify they are orthogonal to each other.

```
y = iris$Petal.Length
y_hat = H %*% y
table(y_hat)

## y_hat
## 1.462  4.26 5.552
##    50    50    50

e = (diag(nrow(iris))-H) %*% y
e

##      [,1]
## [1,] -0.062
## [2,] -0.062
## [3,] -0.162
## [4,]  0.038
## [5,] -0.062
## [6,]  0.238
## [7,] -0.062
## [8,]  0.038
## [9,] -0.062
## [10,]  0.038
## [11,]  0.038
## [12,]  0.138
## [13,] -0.062
## [14,] -0.362
## [15,] -0.262
## [16,]  0.038
## [17,] -0.162
## [18,] -0.062
## [19,]  0.238
```

```
## [20,] 0.038
## [21,] 0.238
## [22,] 0.038
## [23,] -0.462
## [24,] 0.238
## [25,] 0.438
## [26,] 0.138
## [27,] 0.138
## [28,] 0.038
## [29,] -0.062
## [30,] 0.138
## [31,] 0.138
## [32,] 0.038
## [33,] 0.038
## [34,] -0.062
## [35,] 0.038
## [36,] -0.262
## [37,] -0.162
## [38,] -0.062
## [39,] -0.162
## [40,] 0.038
## [41,] -0.162
## [42,] -0.162
## [43,] -0.162
## [44,] 0.138
## [45,] 0.438
## [46,] -0.062
## [47,] 0.138
## [48,] -0.062
## [49,] 0.038
## [50,] -0.062
## [51,] 0.440
## [52,] 0.240
## [53,] 0.640
## [54,] -0.260
## [55,] 0.340
## [56,] 0.240
## [57,] 0.440
## [58,] -0.960
## [59,] 0.340
## [60,] -0.360
## [61,] -0.760
## [62,] -0.060
## [63,] -0.260
## [64,] 0.440
## [65,] -0.660
## [66,] 0.140
## [67,] 0.240
## [68,] -0.160
## [69,] 0.240
## [70,] -0.360
## [71,] 0.540
## [72,] -0.260
## [73,] 0.640
```

```
## [74,] 0.440
## [75,] 0.040
## [76,] 0.140
## [77,] 0.540
## [78,] 0.740
## [79,] 0.240
## [80,] -0.760
## [81,] -0.460
## [82,] -0.560
## [83,] -0.360
## [84,] 0.840
## [85,] 0.240
## [86,] 0.240
## [87,] 0.440
## [88,] 0.140
## [89,] -0.160
## [90,] -0.260
## [91,] 0.140
## [92,] 0.340
## [93,] -0.260
## [94,] -0.960
## [95,] -0.060
## [96,] -0.060
## [97,] -0.060
## [98,] 0.040
## [99,] -1.260
## [100,] -0.160
## [101,] 0.448
## [102,] -0.452
## [103,] 0.348
## [104,] 0.048
## [105,] 0.248
## [106,] 1.048
## [107,] -1.052
## [108,] 0.748
## [109,] 0.248
## [110,] 0.548
## [111,] -0.452
## [112,] -0.252
## [113,] -0.052
## [114,] -0.552
## [115,] -0.452
## [116,] -0.252
## [117,] -0.052
## [118,] 1.148
## [119,] 1.348
## [120,] -0.552
## [121,] 0.148
## [122,] -0.652
## [123,] 1.148
## [124,] -0.652
## [125,] 0.148
## [126,] 0.448
## [127,] -0.752
```

```
## [128,] -0.652
## [129,]  0.048
## [130,]  0.248
## [131,]  0.548
## [132,]  0.848
## [133,]  0.048
## [134,] -0.452
## [135,]  0.048
## [136,]  0.548
## [137,]  0.048
## [138,] -0.052
## [139,] -0.752
## [140,] -0.152
## [141,]  0.048
## [142,] -0.452
## [143,] -0.452
## [144,]  0.348
## [145,]  0.148
## [146,] -0.352
## [147,] -0.552
## [148,] -0.352
## [149,] -0.152
## [150,] -0.452
```

Compute SST, SSR and SSE and R^2 and then show that $SST = SSR + SSE$.

```
SSE = t(e) %*% e
y_bar = mean(y)
SST = t(y-y_bar) %*% (y - y_bar)
SSR = t(y_hat - y_bar) %*% (y_hat - y_bar)
SSR
```

```
##           [,1]
## [1,] 437.1028
```

```
expect_equal(SSR + SSE, SST)
```

Find the angle θ between $y - \bar{y}1$ and $\hat{y} - \bar{y}1$ and then verify that its cosine squared is the same as the R^2 from the previous problem.

```
theta = acos(t(y-y_bar) %*% (y_hat - y_bar) / sqrt(SST * SSR))
theta * 180/pi
```

```
##           [,1]
## [1,] 14.01245
```

Project the y vector onto each column of the X matrix and test if the sum of these projections is the same as \hat{y} .

```
proj1 = (X[,1] %*% t(X[,1]) / as.numeric(t(X[,1]) %*% X[,1])) %*% y
proj2 = (X[,2] %*% t(X[,2]) / as.numeric(t(X[,2]) %*% X[,2])) %*% y
proj3 = (X[,3] %*% t(X[,3]) / as.numeric(t(X[,3]) %*% X[,3])) %*% y
#expect_equal(proj1 + proj2 + proj3, y_hat)
```

Construct the design matrix without an intercept, X , without using `model.matrix`.

```
x = cbind(as.numeric(iris$Species == "setosa"), iris$Species == "versicolor", iris$Species == "virginica")
y = iris$Petal.Length
```



```
head(x)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    1    0    0
## [3,]    1    0    0
## [4,]    1    0    0
## [5,]    1    0    0
## [6,]    1    0    0
```

Find the OLS estimates using this design matrix. It should be the sample averages of the petal lengths within species.

```
H2 = x %>% solve(t(x) %>% x) %>% t(x)
yhatnew = H2 %>% y
```

```
H = X %>% solve(t(X) %>% X) %>% t(X)
y_hat = H %>% y
```

```
unique(yhatnew)
```

```
##      [,1]
## [1,] 1.462
## [2,] 4.260
## [3,] 5.552
```

```
unique(y_hat)
```

```
##      [,1]
## [1,] 1.462
## [2,] 4.260
## [3,] 5.552
```

```
mean(iris$Petal.Length[iris$Species == "setosa"])
```

```
## [1] 1.462
```

```
mean(iris$Petal.Length[iris$Species == "versicolor"])
```

```
## [1] 4.26
```

```
mean(iris$Petal.Length[iris$Species == "virginica"])
```

```
## [1] 5.552
```

Verify the hat matrix constructed from this design matrix is the same as the hat matrix constructed from the design matrix with the intercept. (Fact: orthogonal projection matrices are unique).

```
pacman::p_load(testthat)
expect_equal(H, H2)
```

Project the y vector onto each column of the X matrix and test if the sum of these projections is the same as $yhat$.

```
HY = H2 %>% y
expect_equal(HY, yhatnew)
```

Convert this design matrix into Q , an orthonormal matrix.

```

Q = qr.Q(qr(x))

sum(Q[, 1]^2) #check if normalized

## [1] 1
sum(Q[, 2]^2)

## [1] 1
sum(Q[, 3]^2)

## [1] 1
Q[, 1] %*% Q[, 2] #check if orthogonal

##      [,1]
## [1,]    0
Q[, 1] %*% Q[, 3]

##      [,1]
## [1,]    0
Q[, 2] %*% Q[, 3]

##      [,1]
## [1,]    0

```

Project the y vector onto each column of the Q matrix and test if the sum of these projections is the same as \hat{y} .

```

proj1 = (Q[,1] %*% t(Q[,1]) / as.numeric(t(Q[,1]) %*% Q[,1])) %*% y
proj2 = (Q[,2] %*% t(Q[,2]) / as.numeric(t(Q[,2]) %*% Q[,2])) %*% y
proj3 = (Q[,3] %*% t(Q[,3]) / as.numeric(t(Q[,3]) %*% Q[,3])) %*% y

yhatQ = Q %*% t(Q) %*% y

expect_equal(yhatQ, yhatnew)

```

Find the $p = 3$ linear OLS estimates if Q is used as the design matrix using the `lm` method. Is the OLS solution the same as the OLS solution for X ?

```

mod_Q = lm(Petal.Length ~ 0+Q, iris)
mod_Q

##
## Call:
## lm(formula = Petal.Length ~ 0 + Q, data = iris)
##
## Coefficients:
##      Q1      Q2      Q3
## -10.34  -30.12  -39.26

```

Use the `predict` function and ensure that the predicted values are the same for both linear models: the one created with X as its design matrix and the one created with Q as its design matrix.

```

colnames(X) <- c("setosa", "versicolor", "virginica")
mod3 = lm(y~0+x)
unique(predict(mod3, data.frame(x)))

```

```
## [1] 1.462 4.260 5.552
```

```
mod4 = lm(y~0+Q)
unique(predict(mod4, data.frame(Q)))
```

```
## [1] 1.462 1.462 4.260 5.552
```

Clear the workspace and load the boston housing data and extract X and y . The dimensions are $n = 506$ and $p = 13$. Create a matrix that is $(p + 1) \times (p + 1)$ full of NA's. Label the columns the same columns as X . Do not label the rows. For the first row, find the OLS estimate of the y regressed on the first column only and put that in the first entry. For the second row, find the OLS estimates of the y regressed on the first and second columns of X only and put them in the first and second entries. For the third row, find the OLS estimates of the y regressed on the first, second and third columns of X only and put them in the first, second and third entries, etc. For the last row, fill it with the full OLS estimates.

```
rm(list=ls())
Boston = MASS::Boston
intercept = rep(1, nrow(Boston))
X = as.matrix(cbind(intercept, Boston[, 1:13]))
y = Boston[,14]
Matrix = matrix(data = NA, nrow = 14, ncol = 14)
colnames(Matrix) = c(colnames(X))

for (i in 1:ncol(Matrix)){
  b=array(NA, dim = ncol(Matrix))
  X2 = X[, 1:i]
  X2 = as.matrix(X2)
  XTX_inv = solve(t(X2) %*% X2)
  b[1:i] = XTX_inv %*% t(X2) %*% y
  Matrix[i, ] <- b
}
Matrix
```

```
##      intercept      crim      zn      indus      chas      nox
## [1,] 22.5328063      NA      NA      NA      NA      NA
## [2,] 24.0331062 -0.4151903      NA      NA      NA      NA
## [3,] 22.4856281 -0.3520783 0.11610909      NA      NA      NA
## [4,] 27.3946468 -0.2486283 0.05850082 -0.41557782      NA      NA
## [5,] 27.1128031 -0.2287981 0.05928665 -0.44032511 6.894059      NA
## [6,] 29.4899406 -0.2185190 0.05511047 -0.38348055 7.026223 -5.424659
## [7,] -17.9546350 -0.1769135 0.02128135 -0.14365267 4.784684 -7.184892
## [8,] -18.2649261 -0.1727607 0.01421402 -0.13089918 4.840730 -4.357411
## [9,]  0.8274820 -0.1977868 0.06099257 -0.22573089 4.577598 -14.451531
## [10,] 0.1553915 -0.1780398 0.06095248 -0.21004328 4.536648 -13.342666
## [11,] 2.9907868 -0.1795543 0.07145574 -0.10437742 4.110667 -12.591596
## [12,] 27.1523679 -0.1840321 0.03909990 -0.04232450 3.487528 -22.182110
## [13,] 20.6526280 -0.1599391 0.03887365 -0.02792186 3.216569 -20.484560
## [14,] 36.4594884 -0.1080114 0.04642046  0.02055863 2.686734 -17.766611
##      rm      age      dis      rad      tax      ptratio
## [1,]  NA      NA      NA      NA      NA      NA
## [2,]  NA      NA      NA      NA      NA      NA
## [3,]  NA      NA      NA      NA      NA      NA
## [4,]  NA      NA      NA      NA      NA      NA
## [5,]  NA      NA      NA      NA      NA      NA
## [6,]  NA      NA      NA      NA      NA      NA
## [7,] 7.341586      NA      NA      NA      NA      NA
```

```
## [8,] 7.386357 -0.0236248493 NA NA NA NA
## [9,] 6.752352 -0.0556354540 -1.760312 NA NA NA
## [10,] 6.791184 -0.0562612189 -1.748296 -0.04529059 NA NA
## [11,] 6.664084 -0.0546675064 -1.727933 0.15926305 -0.01434060 NA
## [12,] 6.075744 -0.0451880522 -1.583852 0.25472196 -0.01221262 -0.9962062
## [13,] 6.123072 -0.0459320518 -1.554912 0.28157503 -0.01173838 -1.0142228
## [14,] 3.809865 0.0006922246 -1.475567 0.30604948 -0.01233459 -0.9527472
##          black      lstat
## [1,]          NA          NA
## [2,]          NA          NA
## [3,]          NA          NA
## [4,]          NA          NA
## [5,]          NA          NA
## [6,]          NA          NA
## [7,]          NA          NA
## [8,]          NA          NA
## [9,]          NA          NA
## [10,]          NA          NA
## [11,]          NA          NA
## [12,]          NA          NA
## [13,] 0.013620833          NA
## [14,] 0.009311683 -0.5247584
```

Why are the estimates changing from row to row as you add in more predictors?

The estimates are changing from row to row as you add more predictors because the model is adjusting as it takes in the additional data to return a better fitting line. Thus as the model takes

Create a vector of length $p + 1$ and compute the R^2 values for each of the above models.

```
Rsq = array(dim = 14)
ybar = mean(y)
SST = sum((y - ybar)^2)

for(i in 1:nrow(Matrix)){
  b = c(Matrix[i,1:i],rep(0, nrow(Matrix)-i))

  yhat = X %*% b
  SSR = sum((yhat - ybar)^2)
  Rsqd = SSR / SST
  Rsq[i] = Rsqd
}

Rsq
```

```
## [1] 5.382448e-30 1.507805e-01 2.339884e-01 2.937136e-01 3.295277e-01
## [6] 3.313127e-01 5.873770e-01 5.894902e-01 6.311488e-01 6.319479e-01
## [11] 6.396628e-01 6.703141e-01 6.842043e-01 7.406427e-01
```

Is R^2 monotonically increasing? Why?

R^2 is monotonically increasing because you are fitting more parameters and thus the model will fit the data better.