Lab 4

Brendan Gubbins

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)
mod
```

```
##
## Call:
## lm(formula = Petal.Length ~ Species, data = iris)
##
## Coefficients:
##       (Intercept)  Speciesversicolor   Speciesvirginica
##             1.462              2.798              4.090
```

```
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
```

```r
predict(mod, data.frame(Species = c("virginica")))
```

```
##     1
## 5.552
```

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

```r
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.

```r
H = X %*% solve(t(X) %*% X) %*% t(X)
head(H)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02  0.02  0.02  0.02  0.02  0.02
## [2,] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02  0.02  0.02  0.02  0.02  0.02
## [3,] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02  0.02  0.02  0.02  0.02  0.02
## [4,] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02  0.02  0.02  0.02  0.02  0.02
## [5,] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02  0.02  0.02  0.02  0.02  0.02
## [6,] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02  0.02  0.02  0.02  0.02  0.02
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
## [1,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [2,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [3,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [4,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [5,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [6,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
##      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38]
## [1,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [2,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [3,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [4,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [5,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [6,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
##      [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
## [1,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [2,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [3,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [4,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
## [5,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
```

```
## [6,]  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02  0.02
##      [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
## [1,]     0     0     0     0     0     0     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0     0     0     0     0     0     0
## [3,]     0     0     0     0     0     0     0     0     0     0     0     0
## [4,]     0     0     0     0     0     0     0     0     0     0     0     0
## [5,]     0     0     0     0     0     0     0     0     0     0     0     0
## [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##      [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74]
## [1,]     0     0     0     0     0     0     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0     0     0     0     0     0     0
## [3,]     0     0     0     0     0     0     0     0     0     0     0     0
## [4,]     0     0     0     0     0     0     0     0     0     0     0     0
## [5,]     0     0     0     0     0     0     0     0     0     0     0     0
## [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##      [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86]
## [1,]     0     0     0     0     0     0     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0     0     0     0     0     0     0
## [3,]     0     0     0     0     0     0     0     0     0     0     0     0
## [4,]     0     0     0     0     0     0     0     0     0     0     0     0
## [5,]     0     0     0     0     0     0     0     0     0     0     0     0
## [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##      [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98]
## [1,]     0     0     0     0     0     0     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0     0     0     0     0     0     0
## [3,]     0     0     0     0     0     0     0     0     0     0     0     0
## [4,]     0     0     0     0     0     0     0     0     0     0     0     0
## [5,]     0     0     0     0     0     0     0     0     0     0     0     0
## [6,]     0     0     0     0     0     0     0     0     0     0     0     0
##      [,99] [,100] [,101] [,102] [,103] [,104] [,105] [,106] [,107] [,108]
## [1,]     0      0      0      0      0      0      0      0      0      0
## [2,]     0      0      0      0      0      0      0      0      0      0
## [3,]     0      0      0      0      0      0      0      0      0      0
## [4,]     0      0      0      0      0      0      0      0      0      0
## [5,]     0      0      0      0      0      0      0      0      0      0
## [6,]     0      0      0      0      0      0      0      0      0      0
##      [,109] [,110] [,111] [,112] [,113] [,114] [,115] [,116] [,117] [,118]
## [1,]      0      0      0      0      0      0      0      0      0      0
## [2,]      0      0      0      0      0      0      0      0      0      0
## [3,]      0      0      0      0      0      0      0      0      0      0
## [4,]      0      0      0      0      0      0      0      0      0      0
## [5,]      0      0      0      0      0      0      0      0      0      0
## [6,]      0      0      0      0      0      0      0      0      0      0
##      [,119] [,120] [,121] [,122] [,123] [,124] [,125] [,126] [,127] [,128]
## [1,]      0      0      0      0      0      0      0      0      0      0
## [2,]      0      0      0      0      0      0      0      0      0      0
## [3,]      0      0      0      0      0      0      0      0      0      0
## [4,]      0      0      0      0      0      0      0      0      0      0
## [5,]      0      0      0      0      0      0      0      0      0      0
## [6,]      0      0      0      0      0      0      0      0      0      0
##      [,129] [,130] [,131] [,132] [,133] [,134] [,135] [,136] [,137] [,138]
## [1,]      0      0      0      0      0      0      0      0      0      0
## [2,]      0      0      0      0      0      0      0      0      0      0
## [3,]      0      0      0      0      0      0      0      0      0      0
```

```
## [4,]      0      0      0      0      0      0      0      0      0      0
## [5,]      0      0      0      0      0      0      0      0      0      0
## [6,]      0      0      0      0      0      0      0      0      0      0
##      [,139] [,140] [,141] [,142] [,143] [,144] [,145] [,146] [,147] [,148]
## [1,]      0      0      0      0      0      0      0      0      0      0
## [2,]      0      0      0      0      0      0      0      0      0      0
## [3,]      0      0      0      0      0      0      0      0      0      0
## [4,]      0      0      0      0      0      0      0      0      0      0
## [5,]      0      0      0      0      0      0      0      0      0      0
## [6,]      0      0      0      0      0      0      0      0      0      0
##      [,149] [,150]
## [1,]      0      0
## [2,]      0      0
## [3,]      0      0
## [4,]      0      0
## [5,]      0      0
## [6,]      0      0
```

```r
Matrix::rankMatrix(H)
```

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

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

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

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

```r
expect_equal(H, H %*% H) # wont work on large matrix, use tolerance
```

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

```r
sum(diag(H)) # sum of trace is rank
```

```
## [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 $X_\perp$.

```r
#TO-DO
# rows n, cols = n - (p + 1)
# all orthogonal to X columns
# full-rank matrix n - (p+1) cols, spans residual space
# bind X, X_perp spans the full space
```

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
```

```
I = diag(nrow(iris))
e = (I - 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
```

```r
t(e) %*% y_hat # orthogonal
```

```
##             [,1]
## [1,] -2.2915e-13
```

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

```r
SSE = t(e) %*% e
y_bar = mean(y)
SST = t(y - y_bar) %*% (y - y_bar)
Rsq = 1 - SSE/SST
SSR = t(y_hat - y_bar) %*% (y_hat - y_bar)

expect_equal(SST, SSR + SSE)
```

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.

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

```
##          [,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 yhat.

```r
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) NOT EQUAL!!!
```

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

```r
X = X[,2:ncol(X)]
```

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

```r
b = solve(t(X) %*% X) %*% t(X) %*% y
b
```

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

```
X_model = lm(Petal.Length ~ X, iris)
X_model
```

```
##
## Call:
## lm(formula = Petal.Length ~ X, data = iris)
##
## Coefficients:
## (Intercept)           X1           X2
##       1.462        2.798        4.090
```

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).

```
X = cbind(as.integer(iris$Species == "setosa"), as.integer(iris$Species == "versicolor"), as.integer(iri
H_new = X %*% solve(t(X) %*% X) %*% t(X)
expect_equal(H_new, H)
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
##   [7,]    1    0    0
##   [8,]    1    0    0
##   [9,]    1    0    0
##  [10,]    1    0    0
##  [11,]    1    0    0
##  [12,]    1    0    0
##  [13,]    1    0    0
##  [14,]    1    0    0
##  [15,]    1    0    0
##  [16,]    1    0    0
##  [17,]    1    0    0
##  [18,]    1    0    0
##  [19,]    1    0    0
##  [20,]    1    0    0
##  [21,]    1    0    0
##  [22,]    1    0    0
##  [23,]    1    0    0
##  [24,]    1    0    0
##  [25,]    1    0    0
##  [26,]    1    0    0
##  [27,]    1    0    0
##  [28,]    1    0    0
##  [29,]    1    0    0
```

```
## [30,]    1    0    0
## [31,]    1    0    0
## [32,]    1    0    0
## [33,]    1    0    0
## [34,]    1    0    0
## [35,]    1    0    0
## [36,]    1    0    0
## [37,]    1    0    0
## [38,]    1    0    0
## [39,]    1    0    0
## [40,]    1    0    0
## [41,]    1    0    0
## [42,]    1    0    0
## [43,]    1    0    0
## [44,]    1    0    0
## [45,]    1    0    0
## [46,]    1    0    0
## [47,]    1    0    0
## [48,]    1    0    0
## [49,]    1    0    0
## [50,]    1    0    0
## [51,]    0    1    0
## [52,]    0    1    0
## [53,]    0    1    0
## [54,]    0    1    0
## [55,]    0    1    0
## [56,]    0    1    0
## [57,]    0    1    0
## [58,]    0    1    0
## [59,]    0    1    0
## [60,]    0    1    0
## [61,]    0    1    0
## [62,]    0    1    0
## [63,]    0    1    0
## [64,]    0    1    0
## [65,]    0    1    0
## [66,]    0    1    0
## [67,]    0    1    0
## [68,]    0    1    0
## [69,]    0    1    0
## [70,]    0    1    0
## [71,]    0    1    0
## [72,]    0    1    0
## [73,]    0    1    0
## [74,]    0    1    0
## [75,]    0    1    0
## [76,]    0    1    0
## [77,]    0    1    0
## [78,]    0    1    0
## [79,]    0    1    0
## [80,]    0    1    0
## [81,]    0    1    0
## [82,]    0    1    0
## [83,]    0    1    0
```

```
##  [84,]    0    1    0
##  [85,]    0    1    0
##  [86,]    0    1    0
##  [87,]    0    1    0
##  [88,]    0    1    0
##  [89,]    0    1    0
##  [90,]    0    1    0
##  [91,]    0    1    0
##  [92,]    0    1    0
##  [93,]    0    1    0
##  [94,]    0    1    0
##  [95,]    0    1    0
##  [96,]    0    1    0
##  [97,]    0    1    0
##  [98,]    0    1    0
##  [99,]    0    1    0
## [100,]    0    1    0
## [101,]    0    0    1
## [102,]    0    0    1
## [103,]    0    0    1
## [104,]    0    0    1
## [105,]    0    0    1
## [106,]    0    0    1
## [107,]    0    0    1
## [108,]    0    0    1
## [109,]    0    0    1
## [110,]    0    0    1
## [111,]    0    0    1
## [112,]    0    0    1
## [113,]    0    0    1
## [114,]    0    0    1
## [115,]    0    0    1
## [116,]    0    0    1
## [117,]    0    0    1
## [118,]    0    0    1
## [119,]    0    0    1
## [120,]    0    0    1
## [121,]    0    0    1
## [122,]    0    0    1
## [123,]    0    0    1
## [124,]    0    0    1
## [125,]    0    0    1
## [126,]    0    0    1
## [127,]    0    0    1
## [128,]    0    0    1
## [129,]    0    0    1
## [130,]    0    0    1
## [131,]    0    0    1
## [132,]    0    0    1
## [133,]    0    0    1
## [134,]    0    0    1
## [135,]    0    0    1
## [136,]    0    0    1
## [137,]    0    0    1
```

```
## [138,]    0    0    1
## [139,]    0    0    1
## [140,]    0    0    1
## [141,]    0    0    1
## [142,]    0    0    1
## [143,]    0    0    1
## [144,]    0    0    1
## [145,]    0    0    1
## [146,]    0    0    1
## [147,]    0    0    1
## [148,]    0    0    1
## [149,]    0    0    1
## [150,]    0    0    1
```

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

```
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)
```

Convert this design matrix into Q, an orthonormal matrix.

```
qrX = qr(X)
Q = qr.Q(qrX)
```

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

```
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

expect_equal(proj1 + proj2 + proj3, y_hat)
```

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?

```
lm(Petal.Length ~ Q[,3], iris)
```

```
##
## Call:
## lm(formula = Petal.Length ~ Q[, 3], data = iris)
##
## Coefficients:
## (Intercept)       Q[, 3]
##       2.861      -19.028
```

```r
Q_model = lm(Petal.Length ~ Q, iris) # not the same
Q_model
```

```
##
## Call:
## lm(formula = Petal.Length ~ Q, data = iris)
##
## Coefficients:
## (Intercept)           Q1           Q2           Q3
##       5.552       28.921        9.136           NA
```

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.

```r
predict(X_model)
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    14    15    16    17    18    19    20    21    22    23    24    25    26
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    27    28    29    30    31    32    33    34    35    36    37    38    39
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    40    41    42    43    44    45    46    47    48    49    50    51    52
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 4.260 4.260
##    53    54    55    56    57    58    59    60    61    62    63    64    65
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    66    67    68    69    70    71    72    73    74    75    76    77    78
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    79    80    81    82    83    84    85    86    87    88    89    90    91
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##    92    93    94    95    96    97    98    99   100   101   102   103   104
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 5.552 5.552 5.552 5.552
##   105   106   107   108   109   110   111   112   113   114   115   116   117
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   118   119   120   121   122   123   124   125   126   127   128   129   130
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   131   132   133   134   135   136   137   138   139   140   141   142   143
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   144   145   146   147   148   149   150
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552
```

```r
predict(Q_model)
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    14    15    16    17    18    19    20    21    22    23    24    25    26
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    27    28    29    30    31    32    33    34    35    36    37    38    39
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462
##    40    41    42    43    44    45    46    47    48    49    50    51    52
## 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 1.462 4.260 4.260
```

```
##     53    54    55    56    57    58    59    60    61    62    63    64    65
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##     66    67    68    69    70    71    72    73    74    75    76    77    78
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##     79    80    81    82    83    84    85    86    87    88    89    90    91
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260
##     92    93    94    95    96    97    98    99   100   101   102   103   104
## 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 4.260 5.552 5.552 5.552 5.552
##   105   106   107   108   109   110   111   112   113   114   115   116   117
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   118   119   120   121   122   123   124   125   126   127   128   129   130
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   131   132   133   134   135   136   137   138   139   140   141   142   143
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552 5.552
##   144   145   146   147   148   149   150
## 5.552 5.552 5.552 5.552 5.552 5.552 5.552
```

```
expect_equal(predict(X_model), predict(Q_model))
```

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
X = cbind(1, as.matrix(boston[,1:13]))
y = boston[,14]
p_plus_one = ncol(X)

matrix_p_plus_one = matrix(NA, nrow = p_plus_one, ncol = p_plus_one)
colnames(matrix_p_plus_one) = c(colnames(boston[1:13]), "full OLS")

for (i in 1:ncol(X)) {
  X_i = X[,1:i]
  matrix_p_plus_one[i,1:i] = solve(t(X_i) %*% X_i) %*% t(X_i) %*% y
}

matrix_p_plus_one
```

```
##              crim         zn      indus        chas      nox         rm
##  [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
##            age           dis       rad         tax     ptratio      black
##  [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
##             lstat   full OLS
##  [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
```

```r
View(matrix_p_plus_one)
```

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

Estimates change from row to row because each row is adding one more predictor/feature than the previous row. The model adjusts based on this new information.

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

```r
rsq_vec = c(1:14)

for (i in 1:ncol(X)) {
  mod = lm(y ~ X[, 1:i])
  rsq_vec[i] = summary(mod)$r.squared
}

rsq_vec
```

```
##  [1] 0.0000000 0.1507805 0.2339884 0.2937136 0.3295277 0.3313127 0.5873770
```

```
##  [8] 0.5894902 0.6311488 0.6319479 0.6396628 0.6703141 0.6842043 0.7406427
```

Is R^2 monotonically increasing? Why?

R^2 is monotonically increasing because as the model predicts based on more features, it makes sense that the model will get better at explaining the variance.