Lab 4

Nasif Khan

11:59PM March 10, 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 =="Satosa"] )
```

```
## [1] NaN
```

```
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`.

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

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

```r
#head(H)
```

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)
```

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

```r
diag(H)
```

```
##   [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 0.02
##  [16] 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 0.02
##  [31] 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 0.02
##  [46] 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 0.02
##  [61] 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 0.02
##  [76] 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 0.02
##  [91] 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 0.02
## [106] 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 0.02
## [121] 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 0.02
## [136] 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 0.02
```

```
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-perpendicular.

```
#TO-DO
```

Using the hat matrix, compute the yhat 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
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)

Rsq = 1 - SSE/SST
Rsq
```

```
##           [,1]
## [1,] 0.9413717
```

```
SSR = t(y_hat-y_bar) %*% (y_hat-y_bar)
SSR
```

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

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

Find the angle theta between y - ybar 1 and yhat - ybar 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)
```

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
```

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

```
B <- cbind(0,iris$Species == "versicolor", iris$Species == "virginica" )
```

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

```
mean(B)
```

```
## [1] 0.2222222
```

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).

```
expect_equal(B, H%*%B)
```

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

```
proj4 = ( x[,1] %*% t(x[,1]) / as.numeric( t(x[,1]) %*% x[,1]) ) %*% B
proj5 = ( x[,2] %*% t(x[,2]) / as.numeric( t(x[,2]) %*% x[,2]) ) %*% B
proj6 = ( x[,3] %*% t(x[,3]) / as.numeric( t(x[,3]) %*% x[,3]) ) %*% B
```

Convert this design matrix into Q, an orthonormal matrix.

```
Q = (diag(nrow(iris))-y) %*% B
Q
```

```
##        [,1] [,2] [,3]
##   [1,]    0  -70  -70
##   [2,]    0  -70  -70
##   [3,]    0  -65  -65
##   [4,]    0  -75  -75
##   [5,]    0  -70  -70
##   [6,]    0  -85  -85
##   [7,]    0  -70  -70
##   [8,]    0  -75  -75
##   [9,]    0  -70  -70
##  [10,]    0  -75  -75
##  [11,]    0  -75  -75
##  [12,]    0  -80  -80
##  [13,]    0  -70  -70
##  [14,]    0  -55  -55
##  [15,]    0  -60  -60
##  [16,]    0  -75  -75
##  [17,]    0  -65  -65
##  [18,]    0  -70  -70
##  [19,]    0  -85  -85
##  [20,]    0  -75  -75
##  [21,]    0  -85  -85
##  [22,]    0  -75  -75
##  [23,]    0  -50  -50
##  [24,]    0  -85  -85
##  [25,]    0  -95  -95
##  [26,]    0  -80  -80
##  [27,]    0  -80  -80
##  [28,]    0  -75  -75
##  [29,]    0  -70  -70
```

```
##  [30,]    0  -80  -80
##  [31,]    0  -80  -80
##  [32,]    0  -75  -75
##  [33,]    0  -75  -75
##  [34,]    0  -70  -70
##  [35,]    0  -75  -75
##  [36,]    0  -60  -60
##  [37,]    0  -65  -65
##  [38,]    0  -70  -70
##  [39,]    0  -65  -65
##  [40,]    0  -75  -75
##  [41,]    0  -65  -65
##  [42,]    0  -65  -65
##  [43,]    0  -65  -65
##  [44,]    0  -80  -80
##  [45,]    0  -95  -95
##  [46,]    0  -70  -70
##  [47,]    0  -80  -80
##  [48,]    0  -70  -70
##  [49,]    0  -75  -75
##  [50,]    0  -70  -70
##  [51,]    0 -234 -235
##  [52,]    0 -224 -225
##  [53,]    0 -244 -245
##  [54,]    0 -199 -200
##  [55,]    0 -229 -230
##  [56,]    0 -224 -225
##  [57,]    0 -234 -235
##  [58,]    0 -164 -165
##  [59,]    0 -229 -230
##  [60,]    0 -194 -195
##  [61,]    0 -174 -175
##  [62,]    0 -209 -210
##  [63,]    0 -199 -200
##  [64,]    0 -234 -235
##  [65,]    0 -179 -180
##  [66,]    0 -219 -220
##  [67,]    0 -224 -225
##  [68,]    0 -204 -205
##  [69,]    0 -224 -225
##  [70,]    0 -194 -195
##  [71,]    0 -239 -240
##  [72,]    0 -199 -200
##  [73,]    0 -244 -245
##  [74,]    0 -234 -235
##  [75,]    0 -214 -215
##  [76,]    0 -219 -220
##  [77,]    0 -239 -240
##  [78,]    0 -249 -250
##  [79,]    0 -224 -225
##  [80,]    0 -174 -175
##  [81,]    0 -189 -190
##  [82,]    0 -184 -185
##  [83,]    0 -194 -195
```

```
##  [84,]    0 -254 -255
##  [85,]    0 -224 -225
##  [86,]    0 -224 -225
##  [87,]    0 -234 -235
##  [88,]    0 -219 -220
##  [89,]    0 -204 -205
##  [90,]    0 -199 -200
##  [91,]    0 -219 -220
##  [92,]    0 -229 -230
##  [93,]    0 -199 -200
##  [94,]    0 -164 -165
##  [95,]    0 -209 -210
##  [96,]    0 -209 -210
##  [97,]    0 -209 -210
##  [98,]    0 -214 -215
##  [99,]    0 -149 -150
## [100,]    0 -204 -205
## [101,]    0 -300 -299
## [102,]    0 -255 -254
## [103,]    0 -295 -294
## [104,]    0 -280 -279
## [105,]    0 -290 -289
## [106,]    0 -330 -329
## [107,]    0 -225 -224
## [108,]    0 -315 -314
## [109,]    0 -290 -289
## [110,]    0 -305 -304
## [111,]    0 -255 -254
## [112,]    0 -265 -264
## [113,]    0 -275 -274
## [114,]    0 -250 -249
## [115,]    0 -255 -254
## [116,]    0 -265 -264
## [117,]    0 -275 -274
## [118,]    0 -335 -334
## [119,]    0 -345 -344
## [120,]    0 -250 -249
## [121,]    0 -285 -284
## [122,]    0 -245 -244
## [123,]    0 -335 -334
## [124,]    0 -245 -244
## [125,]    0 -285 -284
## [126,]    0 -300 -299
## [127,]    0 -240 -239
## [128,]    0 -245 -244
## [129,]    0 -280 -279
## [130,]    0 -290 -289
## [131,]    0 -305 -304
## [132,]    0 -320 -319
## [133,]    0 -280 -279
## [134,]    0 -255 -254
## [135,]    0 -280 -279
## [136,]    0 -305 -304
## [137,]    0 -280 -279
```

```
## [138,]    0 -275 -274
## [139,]    0 -240 -239
## [140,]    0 -270 -269
## [141,]    0 -280 -279
## [142,]    0 -255 -254
## [143,]    0 -255 -254
## [144,]    0 -295 -294
## [145,]    0 -285 -284
## [146,]    0 -260 -259
## [147,]    0 -250 -249
## [148,]    0 -260 -259
## [149,]    0 -270 -269
## [150,]    0 -255 -254
```

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

```r
proj7 = ( x[,1] %*% t(x[,1]) / as.numeric( t(x[,1]) %*% x[,1]) ) %*% Q
proj8 = ( x[,2] %*% t(x[,2]) / as.numeric( t(x[,2]) %*% x[,2]) ) %*% Q
proj9 = ( x[,3] %*% t(x[,3]) / as.numeric( t(x[,3]) %*% x[,3]) ) %*% Q
```

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?

```r
mean(B)
```

```
## [1] 0.2222222
```

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
#?
```

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) x (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.

```r
#TO-DO
```

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

#TO-DO

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

```r
#TO-DO
```

Is R^2 monotonically increasing? Why?

#TO-DO