Lab 4

Michael Velez

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 == "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
```

```
#trace same as the rank
```

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 $X_\perp$.

```
#TO-DO
```

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
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
```

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

```
SSE = t(e) %*% e
SSE
```

```
##         [,1]
## [1,] 27.2226
```

```
y_bar = mean(y)
SST = t(y - y_bar) %*% (y - y_bar)
SST
```

```
##          [,1]
## [1,] 464.3254
```

```
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 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 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`.

```
X_2 = cbind(as.integer(iris$Species == "setosa"), as.integer(iris$Species == "versicolor"), as.integer(iris$Speci
es == "virginica"))
head(X_2)
```

```
##      [,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.

```
y = iris$Petal.Length
H = X_2 %*% solve(t(X_2) %*% X) %*% t(X_2)
y_hat = H %*% y
unique(y_hat)
```

```
##       [,1]
## [1,] 1.462
## [2,] 2.798
## [3,] 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(iris$Species
== "virginica"))
H = X_2 %*% solve(t(X_2) %*% X) %*% t(X_2)
H_new = X %*% solve(t(X) %*% X) %*% t(X)
expect_equal(H_new, H)
```

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
y_hat = H %*% y

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

Convert this design matrix into Q, an orthonormal matrix.

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

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?

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

```
##
## 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")
first_mod = lm(y ~ 0 + X)
unique(predict(first_mod, data.frame(X)))
```

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

```
second_mod = lm(y ~ 0 + Q)
unique(predict(second_mod, 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
X = cbind(1, as.matrix(boston[,1:13]))
y = boston[,14]
p_add_one = ncol(X)

matrix_p_add_one = matrix(NA, nrow = p_add_one, ncol = p_add_one)
colnames(matrix_p_add_one) = c(colnames(boston[1:13]), "full OLS")
for (i in 1:ncol(X)) {
  X_i = X[,1:i]
  matrix_p_add_one[i,1:i] = solve(t(X_i) %*% X_i) %*% t(X_i) %*% y
}
matrix_p_add_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
```

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

This is because each row is adding another feature which changes the estimates' value.

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

```
Rsq_vector = c(1:14)
for (i in 1:ncol(X)) {
  mod = lm(y ~ X[, 1:i])
  Rsq_vector[i] = summary(mod)$r.squared
}
Rsq_vector
```

```
##  [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?

This is because as the amount of featurs goes up, the value of R^2 will increase (be more accurate).