Lab 5

Vincent Miceli

11:59PM March 16, 2019

Load the Boston housing data frame and create the vector y (the median value) and matrix X (all other features) from the data frame. Name the columns the same as Boston except for the first name it "(Intercept)".

```
data(Boston, package = "MASS")
summary(Boston)
```

```
##       crim                zn              indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm              age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00
```

```
y = Boston$medv
X = cbind(rep(1, nrow(Boston)), as.matrix(Boston[, 1:13]))
colnames(X)[1] = "(Intercept)"
```

Run the OLS linear model to get b, the vector of coefficients. Do not use `lm`.

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

```
##                      [,1]
## (Intercept)  3.645949e+01
## crim        -1.080114e-01
## zn           4.642046e-02
## indus        2.055863e-02
```

```
## chas         2.686734e+00
## nox         -1.776661e+01
## rm           3.809865e+00
## age          6.922246e-04
## dis         -1.475567e+00
## rad          3.060495e-01
## tax         -1.233459e-02
## ptratio     -9.527472e-01
## black        9.311683e-03
## lstat       -5.247584e-01
```

Find the hat matrix for this regression `H` and find its rank. Is this rank expected?

```
H = X %*% solve(t(X) %*% X) %*% t(X)
dim(H)
```

```
## [1] 506 506
```

```
pacman::p_load(Matrix)
rankMatrix(H)
```

```
## [1] 14
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 1.123546e-13
```

Verify this is a projection matrix by verifying the two sufficient conditions. Use the `testthat` library's `expect_equal(matrix1, matrix2, tolerance = 1e-2)`.

```
pacman::p_load(testthat)
expect_equal(H, t(H), tolerance = 1e-2)
expect_equal(H, H %*% H, tolerance = 1e-2)
```

Find the matrix that projects onto the space of residuals `H_comp` and find its rank. Is this rank expected?

```
I = diag(nrow(H))
H_comp = (I - H)
rankMatrix(H_comp)
```

```
## [1] 497
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 1.123546e-13
```

Verify this is a projection matrix by verifying the two sufficient conditions. Use the `testthat` library.

```
expect_equal(H_comp, t(H_comp))
expect_equal(H_comp, H_comp %*% H_comp )
```

Calculate $\hat{y}$.

```
yhat = H %*% y
```

Calculate e as the difference of y and $\hat{y}$ and the projection onto the space of the residuals. Verify the two

means of calculating the residuals provide the same results.

```
e = y - yhat
e_2 = H_comp %*% y
expect_equal(e, e_2)
```

Calculate R^2 and RMSE.

```
sse = sum(e^2)
sst = sum((y - mean(y))^2)
r2 = 1 - (sse / sst)
r2
```

```
## [1] 0.7406427
```

```
mse = sse / (nrow(X) - ncol(X))
RMSE = sqrt(mse)
RMSE
```

```
## [1] 4.745298
```

Verify $\hat{y}$ and e are orthogonal.

```
t(e) %*% yhat
```

```
##                [,1]
## [1,] -4.991142e-08
```

Verify $\hat{y} - \bar{y}$ and e are orthogonal.

```
t(e) %*% (yhat - mean(y))
```

```
##               [,1]
## [1,] 2.832162e-09
```

Find the cosine-squared of $y - \bar{y}$ and $\hat{y} - \bar{y}$ and verify it is the same as R^2.

```
y_minus_y_bar = y - mean(y)
yhat_minus_y_bar = yhat - mean(y)
len_y_minus_y_bar = sqrt(sum(y_minus_y_bar^2))
len_yhat_minus_y_bar = sqrt(sum(yhat_minus_y_bar^2))

theta = acos(t(y_minus_y_bar) %*% yhat_minus_y_bar / (len_y_minus_y_bar * len_yhat_minus_y_bar))
theta * (180 / pi)
```

```
##          [,1]
## [1,] 30.61531
```

```
cos_theta_sqr = cos(theta)^2
cos_theta_sqr
```

```
##           [,1]
## [1,] 0.7406427
```

Verify the sum of squares identity which we learned was due to the Pythagorean Theorem (applies since the projection is specifically orthogonal).

```
len_y_minus_y_bar^2 - len_yhat_minus_y_bar^2 - sse
```

```
## [1] 5.666152e-09
```

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.

```
M = matrix(NA, nrow = ncol(X), ncol = ncol(X))
colnames(M) = colnames(X)
M
```

```
##       (Intercept) crim zn indus chas nox rm age dis rad tax ptratio black
##  [1,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
##  [2,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
##  [3,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
##  [4,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
##  [5,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
##  [6,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
##  [7,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
##  [8,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
##  [9,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
## [10,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
## [11,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
## [12,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
## [13,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
## [14,]          NA   NA NA    NA   NA  NA NA  NA  NA  NA  NA      NA    NA
##       lstat
##  [1,]    NA
##  [2,]    NA
##  [3,]    NA
##  [4,]    NA
##  [5,]    NA
##  [6,]    NA
##  [7,]    NA
##  [8,]    NA
##  [9,]    NA
## [10,]    NA
## [11,]    NA
## [12,]    NA
## [13,]    NA
## [14,]    NA
```

```
X_j = X[, 1, drop=FALSE]
b = solve(t(X_j) %*% X_j) %*% t(X_j) %*% y
b
```

```
##                 [,1]
## (Intercept) 22.53281
```

```
mean(y)
```

```
## [1] 22.53281
```

```
M[1,1] = b

X_j2 = X[, 1:2, drop = FALSE]
b = solve(t(X_j2) %*% X_j2) %*% t(X_j2) %*% y
b
```

```
##                   [,1]
## (Intercept) 24.0331062
## crim        -0.4151903
```

```
for (j in 1:ncol(M)){
  X_j = X[, 1:j, drop=FALSE]
  b = solve(t(X_j) %*% X_j) %*% t(X_j) %*% y
  M[j, 1:j] = b
}

round(M,2)
```

```
##       (Intercept)  crim   zn indus chas    nox   rm   age   dis   rad
##  [1,]       22.53    NA   NA    NA   NA     NA   NA    NA    NA    NA
##  [2,]       24.03 -0.42   NA    NA   NA     NA   NA    NA    NA    NA
##  [3,]       22.49 -0.35 0.12    NA   NA     NA   NA    NA    NA    NA
##  [4,]       27.39 -0.25 0.06 -0.42   NA     NA   NA    NA    NA    NA
##  [5,]       27.11 -0.23 0.06 -0.44 6.89     NA   NA    NA    NA    NA
##  [6,]       29.49 -0.22 0.06 -0.38 7.03  -5.42   NA    NA    NA    NA
##  [7,]      -17.95 -0.18 0.02 -0.14 4.78  -7.18 7.34    NA    NA    NA
##  [8,]      -18.26 -0.17 0.01 -0.13 4.84  -4.36 7.39 -0.02    NA    NA
##  [9,]        0.83 -0.20 0.06 -0.23 4.58 -14.45 6.75 -0.06 -1.76    NA
## [10,]        0.16 -0.18 0.06 -0.21 4.54 -13.34 6.79 -0.06 -1.75 -0.05
## [11,]        2.99 -0.18 0.07 -0.10 4.11 -12.59 6.66 -0.05 -1.73  0.16
## [12,]       27.15 -0.18 0.04 -0.04 3.49 -22.18 6.08 -0.05 -1.58  0.25
## [13,]       20.65 -0.16 0.04 -0.03 3.22 -20.48 6.12 -0.05 -1.55  0.28
## [14,]       36.46 -0.11 0.05  0.02 2.69 -17.77 3.81  0.00 -1.48  0.31
##         tax ptratio black lstat
##  [1,]    NA      NA    NA    NA
##  [2,]    NA      NA    NA    NA
##  [3,]    NA      NA    NA    NA
##  [4,]    NA      NA    NA    NA
##  [5,]    NA      NA    NA    NA
##  [6,]    NA      NA    NA    NA
##  [7,]    NA      NA    NA    NA
##  [8,]    NA      NA    NA    NA
##  [9,]    NA      NA    NA    NA
## [10,]    NA      NA    NA    NA
## [11,] -0.01      NA    NA    NA
## [12,] -0.01   -1.00    NA    NA
## [13,] -0.01   -1.01  0.01    NA
## [14,] -0.01   -0.95  0.01 -0.52
```

Examine this matrix. Why are the estimates changing from row to row as you add in more predictors?

As we add more predictors, the weights have to adjust to create the best fit linear model of the data. These weight values are not permanent, they are just used to create the OLS fit line.

Clear the workspace and load the diamonds dataset.

```
pacman::p_load(ggplot2)
data(diamonds)
```

Extract y, the price variable and "c", the nominal variable "color" as vectors.

```
y = diamonds$price
c = diamonds$color
```

Convert the "c" vector to X which contains an intercept and an appropriate number of dummies. Let the color G be the refernce category as it is the modal color. Name the columns of X appropriately. The first should be "(Intercept)". Delete c.

```
X = rep(1, nrow(diamonds))

for (level in levels(c)){
  if(level!="G"){
    X = cbind(X, c == level)
  }
}
colnames(X) = c("Intercept", "Is_D", "Is_E","Is_F","Is_H","Is_I","Is_J")
head(X)
```

```
##      Intercept Is_D Is_E Is_F Is_H Is_I Is_J
## [1,]         1    0    1    0    0    0    0
## [2,]         1    0    1    0    0    0    0
## [3,]         1    0    1    0    0    0    0
## [4,]         1    0    0    0    0    1    0
## [5,]         1    0    0    0    0    0    1
## [6,]         1    0    0    0    0    0    1
```

Repeat the iterative exercise above we did for Boston here.

Why didn't the estimates change as we added more and more features?

TO-DO

```
#TO-DO
```

Create a vector y by simulating $n = 100$ standard iid normals. Create a matrix of size 100 x 2 and populate the first column by all ones (for the intercept) and the second column by 100 standard iid normals. Find the R^2 of an OLS regression of `y ~ X`. Use matrix algebra.

```
y = rnorm(100)
X = rep(1, 100)
M = cbind(X,y)

b = solve(t(X) %*% X) %*% t(X) %*% y
yhat = X%*%b

ybar = mean(y)

e = y-yhat
sse = sum(e^2)
sst = sum((y-ybar)^2)
R2 = 1- sse/sst
R2
```

```
## [1] 0
```

from the last problem. Find the R^2 of an OLS regression of `y ~ X`. You can use the `summary` function of an `lm` model.

Write a for loop to each time bind a new column of 100 standard iid normals to the matrix X and find the R^2 each time until the number of columns is 100. Create a vector to save all R^2's. What happened??

```
model = lm(y ~ X)
summary(model)$r.squared
```

```
## [1] 0
```

```
v = rep(NA, 100)

M = rep(1, 100)

for (i in 1:100){
  M = cbind(M, rnorm(100))
  model = lm(y ~ M)
  v[i] = summary(model)$r.squared
}

v
```

```
##   [1] 0.0003458395 0.0041625669 0.0328887089 0.0391524936 0.0929566332
##   [6] 0.0938982529 0.1033213676 0.1075655856 0.1115674312 0.1135155330
##  [11] 0.1137318080 0.1410391458 0.1420969877 0.1497502452 0.1508899011
##  [16] 0.1628475050 0.2070649528 0.2094021024 0.2095223134 0.2382946021
##  [21] 0.2534618673 0.2662495309 0.2663659679 0.2667954240 0.2683509459
##  [26] 0.2865866852 0.2956499401 0.3008413409 0.3359850418 0.3581322138
##  [31] 0.3630431978 0.3684566576 0.3759200326 0.3859570064 0.3877155123
##  [36] 0.3972863034 0.4149028552 0.4161429152 0.4272452173 0.4343520482
##  [41] 0.4525020956 0.4527169962 0.4748923531 0.5195309640 0.5199854255
##  [46] 0.5212661234 0.5246600574 0.5431410881 0.5587627923 0.5753714261
##  [51] 0.5763127290 0.6130938574 0.6131116150 0.6178822240 0.6364467327
##  [56] 0.6499265815 0.6516787491 0.6517871167 0.6625459502 0.6911360524
##  [61] 0.7498171519 0.7501454588 0.7503810309 0.7503848003 0.7512343152
##  [66] 0.7608080420 0.7673181199 0.7754830437 0.7766779008 0.7775065057
##  [71] 0.7988966184 0.8116371807 0.8129702506 0.8139593175 0.8142178464
##  [76] 0.8159730064 0.8163683079 0.8194120459 0.8194120490 0.8204268023
##  [81] 0.8204809767 0.8225895599 0.8281145141 0.8291998838 0.8476024936
##  [86] 0.8548109649 0.8548187355 0.8781768829 0.8839466286 0.8866399114
##  [91] 0.9443714147 0.9443714147 0.9558749094 0.9773260834 0.9776889862
##  [96] 0.9955388534 0.9956290089 0.9997712851 1.0000000000 1.0000000000
```

Add one final column to X to bring the number of columns to 101. Then try to compute R^2. What happens and why?

```
M = cbind(M, rnorm(1))

model = lm(M[, 2] ~ M[, 1])
summary(model)$r.squared
```

```
## [1] 0
```

The more random features you add into the matrix, the in sample R squared value will approach 1.