title: “Lab 4” author: “Namisha Singh” output: pdf_document date: March 7, 2022 —

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)
table(iris$Species)

##
##     setosa versicolor  virginica
##         50         50         50

predict(mod,newdata =
data.frame(Species=c("setosa","versicolor","virginica")))

##     1     2     3
## 1.462 4.260 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
```

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

```
table(iris$Species)

##
##     setosa versicolor  virginica
##         50         50         50

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

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

tail(X)
```

```
##        [,1] [,2] [,3]
## [145,]    1    0    0
## [146,]    1    0    0
## [147,]    1    0    0
## [148,]    1    0    0
## [149,]    1    0    0
## [150,]    1    0    0

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

Find the hat matrix H for this regression.

```
table(iris$Species)

## 
##     setosa versicolor  virginica 
##         50         50         50

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

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

tail(X)

##        [,1] [,2] [,3]
## [145,]    1    0    0
## [146,]    1    0    0
## [147,]    1    0    0
## [148,]    1    0    0
## [149,]    1    0    0
## [150,]    1    0    0

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

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

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

```r
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.

```r
#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.

```r
y <- iris$Petal.Length
yhat = H%*%y
evec <- y-yhat
t(evec) %*% yhat

##                [,1]
## [1,] -6.514789e-13
```

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

```r
ybar <- mean(y)
SST <-  sum((y-ybar)^2)
SSR <- sum((yhat-ybar)^2)
SSE <- sum((evec)^2)
Rsq <- SSR/SST

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.

```r
u <- y-ybar
v <- yhat - ybar
normSquared = function(d) {
  sqrt(sum(d^2))
}
norm <- function(d) {
  sqrt(normSquared(d))
}
theta <- acos(norm(t(u)%*%v)/(norm(u)*norm(v)))
theta

## [1] 0.173369

cos(theta)^2

## [1] 0.9702431
```

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

```
yhat_prime = rep(0,length(yhat))
ncol(X)

## [1] 3

for(j in 1:ncol(X)) {
  yhat_prime = yhat_prime + (X[,j]%*%t(X[,j])/normSquared(X[,j])) %*% y
}
head(yhat)

##       [,1]
## [1,] 1.462
## [2,] 1.462
## [3,] 1.462
## [4,] 1.462
## [5,] 1.462
## [6,] 1.462

head(yhat_prime)

##          [,1]
## [1,] 56.36381
## [2,] 56.36381
## [3,] 56.36381
## [4,] 56.36381
## [5,] 56.36381
## [6,] 56.36381
```

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

```
X = cbind(
  as.numeric(iris$Species == "virginica"),
  as.numeric(iris$Species == "setosa"),
  as.numeric(iris$Species == "versicolor"))
colSums(X)

## [1] 50 50 50
```

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

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

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

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).

```
H_prime = X%*%solve(t(X)%*%X)%*%t(X)
expect_equal(H_prime,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.

```
yhat_prime = rep(0,length(yhat))
ncol(X)

## [1] 3

for(j in 1:ncol(X)) {
  yhat_prime = yhat_prime + (X[,j]%*%t(X[,j])/normSquared(X[,j])) %*% y
}
head(yhat)

##       [,1]
## [1,] 1.462
## [2,] 1.462
## [3,] 1.462
## [4,] 1.462
## [5,] 1.462
## [6,] 1.462

head(yhat_prime)

##         [,1]
## [1,] 10.3379
## [2,] 10.3379
## [3,] 10.3379
## [4,] 10.3379
## [5,] 10.3379
## [6,] 10.3379
```

Convert this design matrix into Q, an orthonormal matrix.

```
v_1 = X[,1]
v_2 = X[,2] - (v_1 %*%t(v_1)/normSquared(v_1)) %*% X[,2]
v_3 = X[,3] - (v_1 %*%t(v_1)/normSquared(v_1)) %*% X[,3] - (v_2
%*%t(v_2)/normSquared(v_2)) %*% X[,3]
q_1 = v_1/norm(v_1)
q_2 = v_1/norm(v_2)
q_3 = v_1/norm(v_3)
Q = cbind(q_1,q_2,q_3)
```

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
yhat_prime = rep(0,length(yhat))
ncol(X)

## [1] 3

for(j in 1:ncol(Q)) {
  yhat_prime = yhat_prime + (Q[,j]%*%t(Q[,j])/normSquared(Q[,j])) %*% y
}
head(yhat)

##       [,1]
## [1,] 1.462
## [2,] 1.462
## [3,] 1.462
## [4,] 1.462
## [5,] 1.462
## [6,] 1.462

head(yhat_prime)

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

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
mod =lm(y ~ 0 + .,data=data.frame(X))
b = coef(mod)
mod_Q =lm(y ~ 0 + .,data=data.frame(X))
b_Q = coef(mod_Q)
cbind(b, b_Q)

##        b   b_Q
## X1 5.552 5.552
## X2 1.462 1.462
## X3 4.260 4.260

b_Q / b

## X1 X2 X3
##  1  1  1
```

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
mod$fitted.values
```

```
##     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
cbind(mod$fitted.values,mod_Q$fitted.values)
```

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

```
## 50  1.462 1.462
## 51  4.260 4.260
## 52  4.260 4.260
## 53  4.260 4.260
## 54  4.260 4.260
## 55  4.260 4.260
## 56  4.260 4.260
## 57  4.260 4.260
## 58  4.260 4.260
## 59  4.260 4.260
## 60  4.260 4.260
## 61  4.260 4.260
## 62  4.260 4.260
## 63  4.260 4.260
## 64  4.260 4.260
## 65  4.260 4.260
## 66  4.260 4.260
## 67  4.260 4.260
## 68  4.260 4.260
## 69  4.260 4.260
## 70  4.260 4.260
## 71  4.260 4.260
## 72  4.260 4.260
## 73  4.260 4.260
## 74  4.260 4.260
## 75  4.260 4.260
## 76  4.260 4.260
## 77  4.260 4.260
## 78  4.260 4.260
## 79  4.260 4.260
## 80  4.260 4.260
## 81  4.260 4.260
## 82  4.260 4.260
## 83  4.260 4.260
## 84  4.260 4.260
## 85  4.260 4.260
## 86  4.260 4.260
## 87  4.260 4.260
## 88  4.260 4.260
## 89  4.260 4.260
## 90  4.260 4.260
## 91  4.260 4.260
## 92  4.260 4.260
## 93  4.260 4.260
## 94  4.260 4.260
## 95  4.260 4.260
## 96  4.260 4.260
## 97  4.260 4.260
## 98  4.260 4.260
## 99  4.260 4.260
```

```
## 100 4.260 4.260
## 101 5.552 5.552
## 102 5.552 5.552
## 103 5.552 5.552
## 104 5.552 5.552
## 105 5.552 5.552
## 106 5.552 5.552
## 107 5.552 5.552
## 108 5.552 5.552
## 109 5.552 5.552
## 110 5.552 5.552
## 111 5.552 5.552
## 112 5.552 5.552
## 113 5.552 5.552
## 114 5.552 5.552
## 115 5.552 5.552
## 116 5.552 5.552
## 117 5.552 5.552
## 118 5.552 5.552
## 119 5.552 5.552
## 120 5.552 5.552
## 121 5.552 5.552
## 122 5.552 5.552
## 123 5.552 5.552
## 124 5.552 5.552
## 125 5.552 5.552
## 126 5.552 5.552
## 127 5.552 5.552
## 128 5.552 5.552
## 129 5.552 5.552
## 130 5.552 5.552
## 131 5.552 5.552
## 132 5.552 5.552
## 133 5.552 5.552
## 134 5.552 5.552
## 135 5.552 5.552
## 136 5.552 5.552
## 137 5.552 5.552
## 138 5.552 5.552
## 139 5.552 5.552
## 140 5.552 5.552
## 141 5.552 5.552
## 142 5.552 5.552
## 143 5.552 5.552
## 144 5.552 5.552
## 145 5.552 5.552
## 146 5.552 5.552
## 147 5.552 5.552
## 148 5.552 5.552
```

```
## 149 5.552 5.552
## 150 5.552 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) 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.

```
rm(list=ls())
Boston=MASS::Boston
X=cbind(1, as.matrix(Boston[,1:13]))
y=Boston[,14]
p1=ncol(X)
matrixp1=matrix(NA, nrow=p1, ncol=p1)
for(j in 1:ncol(X)){
  Xj=X[,1:j]
  matrixp1[j,1:j]=solve(t(Xj)%*%Xj)%*%t(Xj)%*%y
}
```

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

As I add in more predictors, the estimates change because there is more data to estimate for.

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

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

Is R^2 monotonically increasing? Why?

Yes, r^2 does monotonically increase because as there is more data the model starts to explain it better and it can explain more of the variation in the dependent variable can be explained by independent.

Create a 2x2 matrix with the first column 1's and the next column iid normals. Find the absolute value of the angle (in degrees, not radians) between the two columns in absolute difference from 90 degrees.

```
n=100
norm_squared=function(v){
  sqrt(sum(v^2))
}
X = matrix(rnorm(2 * n), ncol = 2)
```

```
cos_theta= t(X[,1]%*%X[,2]) / (norm_squared(X[,1])*norm_squared(X[,2]))
abs(90-acos(cos_theta)*180/pi)

##          [,1]
## [1,] 3.568114
```

Repeat this exercise `Nsim = 1e5` times and report the average absolute angle.

```
Nsim = 1e5
angles=array(NA,Nsim)
for (i in 1:Nsim){
  X = matrix(rnorm(2 * n), ncol = 2)
  cos_theta= t(X[,1]%*%X[,2]) / (norm_squared(X[,1])*norm_squared(X[,2]))
  angles[i]=abs(90-acos(cos_theta)*180/pi)
}
mean(angles)

## [1] 4.59935
```

Create a n x 2 matrix with the first column 1's and the next column iid normals. Find the absolute value of the angle (in degrees, not radians) between the two columns. For n = 10, 50, 100, 200, 500, 1000, report the average absolute angle over `Nsim = 1e5` simulations.

```
N=c(2,5,10, 50, 100, 200, 500, 1000)
Nsim = 1e5
angles=matrix(NA,nrow=Nsim,ncol=length(N))
for (j in 1:length(N)){
  for (i in 1:Nsim){
    X=matrix(1,nrow=N[j],ncol=2)
    X[,2]=rnorm(N[j])
    cos_theta= t(X[,1]%*%X[,2]) / (norm_squared(X[,1])*norm_squared(X[,2]))
    angles[i,j]=abs(90-acos(cos_theta)*180/pi)
  }
}
colMeans(angles)

## [1] 44.892337 23.232803 15.357415  6.548025  4.594742  3.236010  2.047147
## [8]  1.444577
```

What is this absolute angle difference from 90 degrees converging to? Why does this make sense?

The absolute angle difference from 90 degrees is converging to zero. This makes sense because the directions are orthogonal.