Lab04

Rachel Brunner

11:59PM March 12, 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"))) #These will be the three y_hats for the model, we hope that
they are y_bar. Now, let's see if they are actually y_bar
```

```
##     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`.

```
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
```

```
#we're regressing species onto lengths. Species are categorical, so we need
dummy variables. If we use an intercept, we will have one intercept and two
dummies. (Because three variables, using one as intercept, leaves two
dummies)
```

Find the hat matrix H for this regression.

```
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]
## [1,]    0      0 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [2,]    0      0 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [3,]    0      0 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [4,]    0      0 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [5,]    0      0 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [6,]    0      0 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
##             [,105]        [,106]        [,107]        [,108]        [,109]
## [1,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [2,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [3,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [4,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [5,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [6,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
##             [,110]        [,111]        [,112]        [,113]        [,114]
## [1,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [2,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [3,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [4,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
```

```
## [5,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [6,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
##             [,115]        [,116]        [,117]        [,118]        [,119]
## [1,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [2,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [3,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [4,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [5,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [6,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
##             [,120]        [,121]        [,122]        [,123]        [,124]
## [1,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [2,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [3,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [4,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [5,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [6,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
##             [,125]        [,126]        [,127]        [,128]        [,129]
## [1,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [2,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [3,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [4,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [5,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [6,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
##             [,130]        [,131]        [,132]        [,133]        [,134]
## [1,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [2,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [3,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [4,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [5,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [6,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
##             [,135]        [,136]        [,137]        [,138]        [,139]
## [1,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [2,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [3,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [4,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [5,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [6,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
##             [,140]        [,141]        [,142]        [,143]        [,144]
## [1,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [2,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [3,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [4,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [5,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [6,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
##             [,145]        [,146]        [,147]        [,148]        [,149]
## [1,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [2,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [3,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [4,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
## [5,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
```

```
## [6,] -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18 -3.469447e-18
##             [,150]
## [1,] -3.469447e-18
## [2,] -3.469447e-18
## [3,] -3.469447e-18
## [4,] -3.469447e-18
## [5,] -3.469447e-18
## [6,] -3.469447e-18
```

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`.

```
pacman ::p_load(testthat)
expect_equal(H %*% H, H)
```

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

```
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 = y - y_hat
t(e) %*% y_hat
```

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

```
#This is numerically (Basically) zero, so they are orthogonal.
```

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

```
y_bar = mean(y)
SST = sum((y - y_bar)^2)
SSR = sum((y_hat - y_bar)^2)
SSE = sum((e)^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.

```
u = y - y_bar
v = y_hat - y_bar
normSq = function(d){
  sum(d^2)
}
norm = function(d){
  sqrt(sum(d^2))
}
theta = acos(norm(t(u) %*% v)/(norm(u)*norm(v)))
theta #it's in radians
```

```
## [1] 0.2445634
```

```
cos(theta)^2
```

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

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 = 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,H_prime) #with or without an intercept
```

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

```
Hy = H_prime %*% y
expect_equal(Hy, y_hat)
```

Convert this design matrix into Q, an orthonormal matrix.

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

##         q_1
##   [1,] 0.00 0.02 0.00
##   [2,] 0.00 0.02 0.00
##   [3,] 0.00 0.02 0.00
##   [4,] 0.00 0.02 0.00
```

```
##   [5,] 0.00 0.02 0.00
##   [6,] 0.00 0.02 0.00
##   [7,] 0.00 0.02 0.00
##   [8,] 0.00 0.02 0.00
##   [9,] 0.00 0.02 0.00
##  [10,] 0.00 0.02 0.00
##  [11,] 0.00 0.02 0.00
##  [12,] 0.00 0.02 0.00
##  [13,] 0.00 0.02 0.00
##  [14,] 0.00 0.02 0.00
##  [15,] 0.00 0.02 0.00
##  [16,] 0.00 0.02 0.00
##  [17,] 0.00 0.02 0.00
##  [18,] 0.00 0.02 0.00
##  [19,] 0.00 0.02 0.00
##  [20,] 0.00 0.02 0.00
##  [21,] 0.00 0.02 0.00
##  [22,] 0.00 0.02 0.00
##  [23,] 0.00 0.02 0.00
##  [24,] 0.00 0.02 0.00
##  [25,] 0.00 0.02 0.00
##  [26,] 0.00 0.02 0.00
##  [27,] 0.00 0.02 0.00
##  [28,] 0.00 0.02 0.00
##  [29,] 0.00 0.02 0.00
##  [30,] 0.00 0.02 0.00
##  [31,] 0.00 0.02 0.00
##  [32,] 0.00 0.02 0.00
##  [33,] 0.00 0.02 0.00
##  [34,] 0.00 0.02 0.00
##  [35,] 0.00 0.02 0.00
##  [36,] 0.00 0.02 0.00
##  [37,] 0.00 0.02 0.00
##  [38,] 0.00 0.02 0.00
##  [39,] 0.00 0.02 0.00
##  [40,] 0.00 0.02 0.00
##  [41,] 0.00 0.02 0.00
##  [42,] 0.00 0.02 0.00
##  [43,] 0.00 0.02 0.00
##  [44,] 0.00 0.02 0.00
##  [45,] 0.00 0.02 0.00
##  [46,] 0.00 0.02 0.00
##  [47,] 0.00 0.02 0.00
##  [48,] 0.00 0.02 0.00
##  [49,] 0.00 0.02 0.00
##  [50,] 0.00 0.02 0.00
##  [51,] 0.00 0.00 0.02
##  [52,] 0.00 0.00 0.02
##  [53,] 0.00 0.00 0.02
##  [54,] 0.00 0.00 0.02
```

```
##  [55,] 0.00 0.00 0.02
##  [56,] 0.00 0.00 0.02
##  [57,] 0.00 0.00 0.02
##  [58,] 0.00 0.00 0.02
##  [59,] 0.00 0.00 0.02
##  [60,] 0.00 0.00 0.02
##  [61,] 0.00 0.00 0.02
##  [62,] 0.00 0.00 0.02
##  [63,] 0.00 0.00 0.02
##  [64,] 0.00 0.00 0.02
##  [65,] 0.00 0.00 0.02
##  [66,] 0.00 0.00 0.02
##  [67,] 0.00 0.00 0.02
##  [68,] 0.00 0.00 0.02
##  [69,] 0.00 0.00 0.02
##  [70,] 0.00 0.00 0.02
##  [71,] 0.00 0.00 0.02
##  [72,] 0.00 0.00 0.02
##  [73,] 0.00 0.00 0.02
##  [74,] 0.00 0.00 0.02
##  [75,] 0.00 0.00 0.02
##  [76,] 0.00 0.00 0.02
##  [77,] 0.00 0.00 0.02
##  [78,] 0.00 0.00 0.02
##  [79,] 0.00 0.00 0.02
##  [80,] 0.00 0.00 0.02
##  [81,] 0.00 0.00 0.02
##  [82,] 0.00 0.00 0.02
##  [83,] 0.00 0.00 0.02
##  [84,] 0.00 0.00 0.02
##  [85,] 0.00 0.00 0.02
##  [86,] 0.00 0.00 0.02
##  [87,] 0.00 0.00 0.02
##  [88,] 0.00 0.00 0.02
##  [89,] 0.00 0.00 0.02
##  [90,] 0.00 0.00 0.02
##  [91,] 0.00 0.00 0.02
##  [92,] 0.00 0.00 0.02
##  [93,] 0.00 0.00 0.02
##  [94,] 0.00 0.00 0.02
##  [95,] 0.00 0.00 0.02
##  [96,] 0.00 0.00 0.02
##  [97,] 0.00 0.00 0.02
##  [98,] 0.00 0.00 0.02
##  [99,] 0.00 0.00 0.02
## [100,] 0.00 0.00 0.02
## [101,] 0.02 0.00 0.00
## [102,] 0.02 0.00 0.00
## [103,] 0.02 0.00 0.00
## [104,] 0.02 0.00 0.00
```

```
## [105,] 0.02 0.00 0.00
## [106,] 0.02 0.00 0.00
## [107,] 0.02 0.00 0.00
## [108,] 0.02 0.00 0.00
## [109,] 0.02 0.00 0.00
## [110,] 0.02 0.00 0.00
## [111,] 0.02 0.00 0.00
## [112,] 0.02 0.00 0.00
## [113,] 0.02 0.00 0.00
## [114,] 0.02 0.00 0.00
## [115,] 0.02 0.00 0.00
## [116,] 0.02 0.00 0.00
## [117,] 0.02 0.00 0.00
## [118,] 0.02 0.00 0.00
## [119,] 0.02 0.00 0.00
## [120,] 0.02 0.00 0.00
## [121,] 0.02 0.00 0.00
## [122,] 0.02 0.00 0.00
## [123,] 0.02 0.00 0.00
## [124,] 0.02 0.00 0.00
## [125,] 0.02 0.00 0.00
## [126,] 0.02 0.00 0.00
## [127,] 0.02 0.00 0.00
## [128,] 0.02 0.00 0.00
## [129,] 0.02 0.00 0.00
## [130,] 0.02 0.00 0.00
## [131,] 0.02 0.00 0.00
## [132,] 0.02 0.00 0.00
## [133,] 0.02 0.00 0.00
## [134,] 0.02 0.00 0.00
## [135,] 0.02 0.00 0.00
## [136,] 0.02 0.00 0.00
## [137,] 0.02 0.00 0.00
## [138,] 0.02 0.00 0.00
## [139,] 0.02 0.00 0.00
## [140,] 0.02 0.00 0.00
## [141,] 0.02 0.00 0.00
## [142,] 0.02 0.00 0.00
## [143,] 0.02 0.00 0.00
## [144,] 0.02 0.00 0.00
## [145,] 0.02 0.00 0.00
## [146,] 0.02 0.00 0.00
## [147,] 0.02 0.00 0.00
## [148,] 0.02 0.00 0.00
## [149,] 0.02 0.00 0.00
## [150,] 0.02 0.00 0.00
```

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

```
y_hat_prime = rep(0, length(y_hat))
for (j in 1:ncol(X)){
  y_hat_prime = y_hat_prime + (Q[,j] %*% t(Q[,j])/normSq(Q[,j])) %*% y
} #Works because columns of Q are orthogonal
```

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?

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

##         b   b_Q
## X1 5.552 277.6
## X2 1.462  73.1
## X3 4.260 213.0

b_Q / b

## q_1  V2  V3
##  50  50  50
```

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.

```
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
intercept = rep(1, nrow(Boston))
X = as.matrix(cbind(intercept, Boston[,1:13]))
y = Boston[,14]
Matrix = matrix(data = NA, nrow = 14, ncol = 14)
colnames(Matrix) = c(colnames(X))

for (i in 1:ncol(Matrix)){
  b = array(NA, dim = ncol(Matrix))
```

```
  X_2 = X[,1:i]
  X_2 = as.matrix(X_2)
  XtX_inv = solve(t(X_2) %*% X_2)
  b[1:i] = XtX_inv %*% t(X_2) %*% y
  Matrix[i,] = b
}
Matrix
```

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

The estimates change as we add in more predictors because the algorithm is taking in more data to return a better fit line.

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

```
Rsq = array(dim = 14)
y_bar = mean(y)
SST = sum((y-y_bar)^2)

for(i in 1:nrow(Matrix)){
  b = c(Matrix[i, 1:i], rep(0, nrow(Matrix) - i))
  y_hat = X %*% b
  SSR = sum((y_hat - y_bar)^2)
  Rsqd = SSR/SST
  Rsq[i] = Rsqd
}
Rsq
```

```
##  [1] 5.382448e-30 1.507805e-01 2.339884e-01 2.937136e-01 3.295277e-01
##  [6] 3.313127e-01 5.873770e-01 5.894902e-01 6.311488e-01 6.319479e-01
## [11] 6.396628e-01 6.703141e-01 6.842043e-01 7.406427e-01
```

Is R^2 monotonically increasing? Why?

Yes, because as you add more parameters, you explain away more of the variance of your error.

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

X = matrix(rnorm(2 * n), ncol = 2)
acos(t(X[,1]) %*% X[,2] / sqrt(sum(X[, 1]^2) * sum(X[, 2]^2))) * 180 / pi
```

```
##          [,1]
## [1,] 86.03027
```

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)
  X[,2] = rnorm(2)
```

```
  cos_theta = (t(X[,1]) %*% X[,2] / sqrt(sum(X[, 1]^2) * sum(X[, 2]^2))) *
180 / pi
  angles[i] = abs(90 - cos(cos_theta)*180/pi)
}
mean(angles)

## [1] 90.09652
```

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_s = c(10,50,100,200,500,1000)
Nsim = 1e5
angles = matrix(NA, nrow = Nsim, ncol = length(N_s))
for(j in 1:length(N_s)){
  for(i in 1:Nsim){
    X = matrix(1, nrow = N_s[j], ncol = 2)
    X[,2] = rnorm(N_s[j])
    cos_theta = (t(X[,1]) %*% X[,2] / sqrt(sum(X[, 1]^2) * sum(X[, 2]^2)))
    angles[i,j] = abs(90 - cos(cos_theta)*180/pi)
  }
}
colMeans(angles)

## [1] 35.50713 33.27175 32.99075 32.84726 32.76172 32.73273
```

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

I am not really sure what it is converging to, but the numbers are getting lower so maybe zero? There is probably something wrong with my code, because I feel like it should be more obvious.