The objective of this section is to (1) study the behavior of the centered R^2 as cofactors are incrementally included in the regression, and (2) use `R` to check the behavior of Problem 2 in the optional section of the first problem set.

Centered R^2

First, we create a random matrix, where each element is drawn from a standard uniform distribution — another context to practice the `function()` structure. The function `randomMat()` generates a long vector of length $n \cdot k$ and then reshapes it into an $n \times k$ matrix.

```
randomMat <- function(n, k) {
  v <- runif(n*k)
  matrix(v, nrow=n, ncol=k)
}
```

The function bound to `randomMat` behaves as we would expect:

```
randomMat(3,2)
```

```
          [,1]      [,2]
[1,] 0.4688696 0.8325487
[2,] 0.3989330 0.4329744
[3,] 0.3218055 0.2732040
```

Another useful function for this section will be to create a square demeaning matrix $\mathbf{A}$ of dimension n. The following function just wraps a few algebraic maneuvers, so that subsequent code is easier to read.

```
demeanMat <- function(n) {
  ones <- rep(1, n)
  diag(n) - (1/n) * ones %*% t(ones)
}
```

As is described in the notes, pre-multiplying a matrix $\mathbf{B}$ by $\mathbf{A}$ will result in a matrix $\mathbf{C} = \mathbf{AB}$ of deviations from the column means of $\mathbf{B}$. Check that this is true. This may seem like a roundabout way to check the equivalence of the matrices; but it provides the opportunity to practice the `apply` function.

```
A <- demeanMat(3)
B <- matrix(1:9, nrow=3)
col.means <- apply(B, 2, mean)
C <- apply(B, 1, function(x) {x - col.means})
all.equal(A %*% B, t(C))
```

```
[1] TRUE
```

Alright, we're ready to apply the functions to real data in order to calculate the centered R^2. First, read in the data to conform to equation (2.37) on page 14 of the lecture notes, and identify the number of observations n for later use:

```
data <- read.csv("../data/auto.csv", header=TRUE)
names(data) <- c("price", "mpg", "weight")
y <- matrix(data$price)
X2 <- cbind(data$mpg, data$weight)
n <- nrow(X2)
```

The centered R^2 is defined according to equation (2.41) as follows:

$$R^2 = \frac{\mathbf{b}_2' \mathbf{X}_2^{*\prime} \mathbf{X}_2^* \mathbf{b}_2}{\mathbf{y}^{*\prime} \mathbf{y}^*}, \tag{1}$$

where $\mathbf{y}^* = \mathbf{A}\mathbf{y}$, $\mathbf{X}_2^* = \mathbf{A}\mathbf{X}_2$, and $\mathbf{b}_2 = (\mathbf{X}_2^{*\prime}\mathbf{X}_2^*)^{-1}\mathbf{X}_2^{*\prime}\mathbf{y}^*$. Noting that $\mathbf{A}$ is both symmetric and idempotent, we can rewrite Equation (1) in terms of matrices already defined, thereby simplifying the subsequent code dramatically. From my limited experience with programming, the best code is that which reflects the core idea of the procedure; more time spent with a pen and paper and not in `R` will almost always yield more readable code, and more readable code yields fewer errors and suggests quick extensions. That said, note that $\mathbf{X}_2^{*\prime}\mathbf{X}_2^* = \mathbf{X}_2'\mathbf{A}'\mathbf{A}\mathbf{X}_2 = \mathbf{X}_2'\mathbf{A}\mathbf{A}\mathbf{X}_2 = \mathbf{X}_2'\mathbf{A}\mathbf{X}_2$ and similarly that $\mathbf{y}^{*\prime}\mathbf{y}^* = \mathbf{y}'\mathbf{A}\mathbf{y}$ and $\mathbf{X}_2^{*\prime}\mathbf{y}^* = \mathbf{X}_2'\mathbf{A}\mathbf{y}$. If we write a more general function, though, we can apply it to an arbitrary dependent vector and associated cofactor matrix:

```
R.squared <- function(y, X) {
  n <- nrow(X)
  A <- demeanMat(n)
  xtax <- t(X) %*% A %*% X
  ytay <- t(y) %*% A %*% y
  b2 <- solve(xtax) %*% t(X) %*% A %*% y
  return(t(b2) %*% xtax %*% b2 / ytay)
}

R.squared(y, X2)
```

```
          [,1]
 [1,] 0.2933891
```

Without some penalty for addtional cofactors, the centered R^2 will monotonically increase with the number of columns in the cofactor matrix $\mathbf{X}$. This function is plotted in Figure (1), mostly as an introduction to very simple plots in `R`.

```
n <- nrow(X2); k.max <- 70
X.rnd <- randomMat(n, k.max)
res <- rep(0, k.max)

for (i in 1:70) {
  X.ext <- cbind(X2, X.rnd[, seq(i)])
  res[i] <- R.squared(y, X.ext)
}

plot(res, type = "l", lwd = 3, col = "blue",
     xlab = "num. of additional columns",
     ylab = "R-squared value")
```

It may be difficult to get a sense of the shape of the curve based on a single draw for the random matrix. We can calculate the relationship between R^2 and the number of cofactors — or we can bootstrap an estimate for each index, which we will do in a subsequent section to illustrate bootstrapping in `R`.

Sum of squared residuals

Suppose that $\mathbf{b}$ is the 2×1 least squared coefficient vector in the regression of $\mathbf{y}$ on $\mathbf{X}_2$. Suppose that $\mathbf{c}$ is some other 2×1 vector. We are asked to show that

$$(\mathbf{y} - \mathbf{X}\mathbf{c})'(\mathbf{y} - \mathbf{X}\mathbf{c}) - (\mathbf{y} - \mathbf{X}\mathbf{b})'(\mathbf{y} - \mathbf{X}\mathbf{b}) = (\mathbf{c} - \mathbf{b})'\mathbf{X}'\mathbf{X}(\mathbf{c} - \mathbf{b}) \tag{2}$$

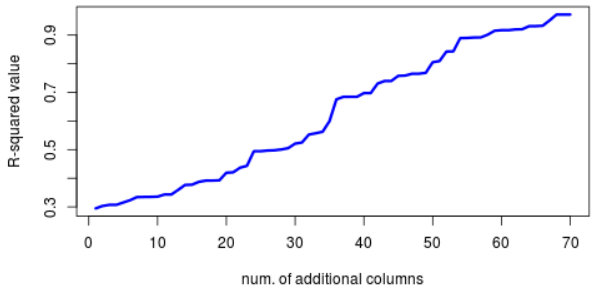

Figure 1: R^2 rises monotonically as a function of columns

We could prove this with matrix algebra. In fact, we are asked to prove this fact with matrix algebra in the problem set. But matrix algebra is for chumps — or very smart and kind people. Let's check the equality using `R` by choosing any arbitrary $\mathbf{c}$.

```
b <- solve(t(X2) %*% X2) %*% t(X2) %*% y
c <- c(-3, 5)
```

For simplicity of notation, define $\mathbf{M}$ and $\mathbf{N}$ to be the following:

```
M <- y - X2 %*% b
N <- y - X2 %*% c
```

Now, we can check both sides of the equality:

```
lhs <- floor(t(N) %*% N - t(M) %*% M)
rhs <- floor(t(c - b) %*% t(X2) %*% X2 %*% (c - b))
all(lhs == rhs)
```

```
[1] TRUE
```

Note, however, that the order of $\mathbf{c}$ and $\mathbf{b}$ doesn't matter:

```
rhs.alt <- floor(t(b - c) %*% t(X2) %*% X2 %*% (b - c))
all(lhs == rhs.alt)
```

```
[1] TRUE
```

This result is because we are effectively looking at the sum of the squared difference between the two vectors. The ordering in the difference calculation doesn't matter if it is subsequently squared. Consider the property $\mathbb{V}(\mathbf{cX}) = \mathbf{c}\mathbb{V}(\mathbf{X})\mathbf{c}'$ for a vector $\mathbf{c}$ or $\mathbb{V}(a\mathbf{X}) = a^2\mathbb{V}(\mathbf{X})$ for a scalar a. The right-hand side of Equation (**??**) is effectively a nested, squared matrix, which has to yield positive entries (and a postive scalar if the result is 1×1):

```
G <- X2 %*% (b - c)
t(G) %*% G
```

```
           [,1]
[1,] 6209641706
```

Additional puzzles

1. **Partitioned regression**: Generate a 100×4 matrix $\mathbf{X}$ *including* a column of ones for the intercept. Additionally, generate a vector $\mathbf{y}$ according to the generating process:

$$y_i = 1 + x_{1i} + 2x_{2i} + 3x_{3i} + \epsilon_i,$$

where $\epsilon_i \sim N(0, 1)$. Let $\mathbf{Q}$ be the first three columns of $\mathbf{X}$ and let $\mathbf{N}$ be the final column. In addition, let

$$\begin{aligned} \hat{\gamma}_1 &= (\mathbf{Q}'\mathbf{Q})^{-1}\mathbf{Q}'\mathbf{y} \quad \text{and} \quad \mathbf{f} = \mathbf{y} - \mathbf{Q}\hat{\gamma}_1 \\ \hat{\gamma}_2 &= (\mathbf{Q}'\mathbf{Q})^{-1}\mathbf{Q}'\mathbf{N} \quad \text{and} \quad \mathbf{g} = \mathbf{N} - \mathbf{Q}\hat{\gamma}_2 \\ \hat{\gamma}_3 &= \mathbf{f} \cdot \mathbf{g}/||\mathbf{g}||^2 \quad \text{and} \quad \mathbf{e} = \mathbf{f} - \mathbf{g}\hat{\gamma}_3 \end{aligned}$$

Show that $\hat{\beta} = [\hat{\gamma}_1 - \hat{\gamma}_2\hat{\gamma}_3 \quad \hat{\gamma}_3]$. Note that the total dimension of $\hat{\beta}$ is 4.

Answer:

```
X <- cbind(1, randomMat(100, 3))
e <- rnorm(100)

beta <- c(1, 1, 2, 3)
y <- X %*% beta + e

Q <- X[, 1:3]
N <- X[, 4]
gamma.1 <- solve(t(Q) %*% Q) %*% t(Q) %*% y
gamma.2 <- solve(t(Q) %*% Q) %*% t(Q) %*% N
f <- y - Q %*% gamma.1
g <- N - Q %*% gamma.2
gamma.3 <- as.numeric(crossprod(f,g)/crossprod(g,g))
e <- f - g * gamma.3

(b <- c(gamma.1 - gamma.2 * gamma.3, gamma.3))
```

```
[1] 1.2704760 0.8533298 1.6991382 3.0427751
```