Goodness of Fit **ARE212**: Section 04

The idea behind this section is to study the behavior of the centered and uncentered R^2 as cofactors are incrementally included in the regression. First, we must create a random matrix, where each variable is drawn from a standard uniform distribution. All elements are independent and identically distributed, so we can create a very long, random vector and reshape it into a rectangular matrix. For convenience and practice writing functions in R, we show a general function that accepts the dimensions (n rows and k columns) of the matrix. The function generates a long vector of length $n \cdot k$ and then reshapes it into an $n \times k$ matrix.

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

The function bound to `random.mat()` behaves as we would expect:

```
random.mat(3,2)
```

```
           [,1]      [,2]
[1,] 0.02103311 0.2653310
[2,] 0.87201441 0.3937495
[3,] 0.33811370 0.4776009
```

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.

```
demean.mat <- 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.

```
A <- demean.mat(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:

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

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 Eq. (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 $\mathbf{X}_2^{*\prime}\mathbf{X}_2^* = \mathbf{X}_2^\prime\mathbf{A}^\prime\mathbf{A}\mathbf{X}_2 = \mathbf{X}_2^\prime\mathbf{A}\mathbf{A}\mathbf{X}_2 = \mathbf{X}_2^\prime\mathbf{A}\mathbf{X}_2$ and similarly that $\mathbf{y}^{*\prime}\mathbf{y}^* = \mathbf{y}^\prime\mathbf{A}\mathbf{y}$ and $\mathbf{X}_2^{*\prime}\mathbf{y}^* = \mathbf{X}_2^\prime\mathbf{A}\mathbf{y}$. 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 <- demean.mat(n)
  xtax <- t(X) %*% A %*% X
  ytay <- t(y) %*% A %*% y
  b2 <- solve(xtax) %*% t(X) %*% A %*% y
  t(b2) %*% xtax %*% b2 / ytay
}

R.squared(y, X2)
```

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

Without some penalty for addtional cofactors, the 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)
X.rnd <- random.mat(n, 70)
res <- rep(0, 70)
for(i in 1:70) {
  X.ext <- cbind(X2, X.rnd[, 1:i])
  res[i] <- R.squared(y, X.ext)
}
plot(res)
```

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`.

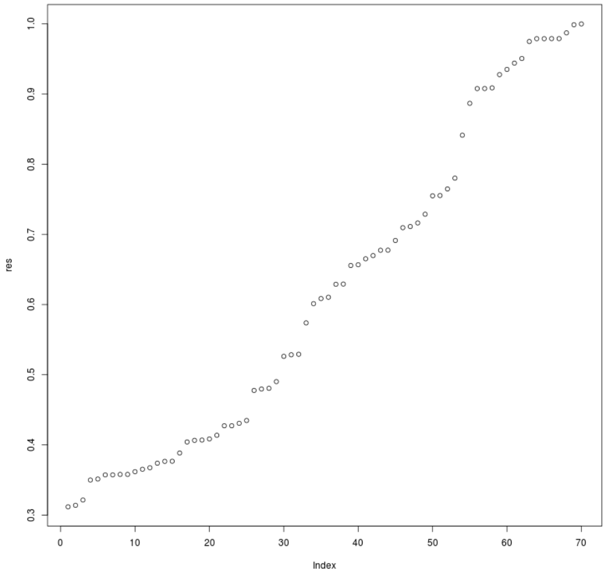

Figure 1: R^2 rises monotonically as a function of columns