This is where section materials with no home come to rest.

Projection and annihilation matrices

```
data <- read.csv("../data/auto.csv", header=TRUE)
names(data) <- c("price", "mpg", "weight")
y <- matrix(data$price)
X <- cbind(1, data$mpg, data$weight)
n <- nrow(X)
```

Digging deeper into the numbers, consider the projection matrix $\mathbf{P} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ and the residual maker matrix $\mathbf{M} = \mathbf{I}_n - \mathbf{P}$

```
n <- nrow(y)
P <- X %*% solve(t(X) %*% X) %*% t(X)
M <- diag(n) - P
```

`R` is useful for checking the properties of these matrices, including whether $\mathbf{M}$ is symmetric, that is, whether $\mathbf{M} = \mathbf{M}'$. The function `all.equal()` does not test **exact** equality, but instead whether the supplied objects are "close enough" to be considered the same. The problem is the limits of machine precision, and rounding at the tail ends of floating point numbers.

```
all.equal(M, t(M))
```

```
[1] TRUE
```

If we want to test for exact equality, we set the tolerance to zero, and the function will return a message with the mean relative difference between elements — which is clearly very close to zero.

```
all.equal(M, t(M), tol=0)
```

```
[1] "Mean relative difference: 5.547715e-15"
```

The residual maker matrix should also be idempotent, or $\mathbf{M} = \mathbf{MM}$.

```
all.equal(M, M %*% M)
```

```
[1] TRUE
```

We could also verify that $\mathbf{M} = \mathbf{M}'\mathbf{M} = \mathbf{M}'\mathbf{M}' = \mathbf{MMMM}$, and so on. Instead, let's move on to the sums of squares.

Calculating R^2_{uc}

Finally, we can use $\mathbf{M}$ and $\mathbf{P}$ to examine the different components of the variation in the dependent variable as they relate to the OLS estimate. We'll then use this to calculate the R_{2uc}. First, recall from lecture that:

$$\mathbf{y}'\mathbf{y} = \hat{\mathbf{y}}'\hat{\mathbf{y}} + \mathbf{e}'\mathbf{e} \tag{1}$$

First, define the relevant variables:

```
e <- M %*% y
y.hat <- P %*% y
epe <- t(e) %*% e
yhpyh <- t(y.hat) %*% y.hat
ypy <- t(y) %*% y
```

Then check the condition in Eq. (1):

```
all.equal(ypy, yhpyh + epe)
```

```
[1] TRUE
```

This is neat, but what

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

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

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 $(\mathbf{cX}) = \mathbf{c}(\mathbf{X})\mathbf{c}'$ for a vector $\mathbf{c}$ or $(a\mathbf{X}) = a^2(\mathbf{X})$ 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
```