Gov 2000: 10. Multiple Linear Regression: Regression in Matrix Form

Matthew Blackwell
Harvard University
mblackwell@gov.harvard.edu

November 7, 2016

Where are we? Where are we going?

- Last few weeks: regression estimation and inference with one and two independent variables
- This week: the general regression model with arbitrary covariates
- Next few weeks: regression odds and ends and what to do when the regression assumptions go wrong

Nunn & Wantchekon

Nunn and Wantchekon (2011) ask the following question: are there long-term, persistent effects of slave trade on Africans today? Their basic idea is to compare levels of interpersonal trust (dependent variable) across different levels of historical slave exports from a respondent's ethnic group (main independent variable). The essential problem with this type of analysis is that ethnic groups and respondents might differ in their interpersonal trust in ways that correlate with the severity of slavery.

One way they attempt to get around this problem is to try to control for relevant differences between groups via multiple regression. Here's their specification of the regression

III. Estimating Equations and Empirical Results

A. *OLS Estimates*

We begin by estimating the relationship between the number of slaves that were taken from an individual's ethnic group and the individual's current level of trust. Our baseline estimating equation is:

$$(1)\quad trust_{i,e,d,c} = \alpha_c + \beta slave\ exports_e + \mathbf{X}'_{i,e,d,c}\Gamma + \mathbf{X}'_{d,c}\Omega + \mathbf{X}'_e\Phi + \varepsilon_{i,e,d,c},$$

where i indexes individuals, e ethnic groups, d districts, and c countries. The variable $trust_{i,e,d,c}$ denotes one of our five measures of trust, which vary across individuals. α_c denotes country fixed effects, which are included to capture country-specific factors, such as government regulations, that may affect trust (e.g., Philippe Aghion et al. 2010; Aghion, Algan, and Cahuc 2008). $slave\ exports_e$ is a measure of the number of slaves taken from ethnic group e during the slave trade. (We discuss this variable in more detail below.) Our coefficient of interest is β, the estimated relationship between the slave exports of an individual's ethnic group and the individual's current level of trust.

Our goal is to be able to understand what's we're saying in this equation. Here is the actual regression that they ran:

```
nunn <- foreign::read.dta("../data/Nunn_Wantchekon_AER_2011.dta")
mod <- lm(trust_neighbors ~ exports + age + male + urban_dum + malaria_ecology, data = nunn)
summary(mod)
```

```
##
## Call:
## lm(formula = trust_neighbors ~ exports + age + male + urban_dum +
##     malaria_ecology, data = nunn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5954 -0.7491  0.1440  0.8735  1.9964
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1.503e+00  2.183e-02  68.844   <2e-16 ***
## exports         -1.021e-03  4.094e-05 -24.935   <2e-16 ***
## age              5.045e-03  4.724e-04  10.680   <2e-16 ***
## male             2.784e-02  1.382e-02   2.015   0.0439 *
```

```
## urban_dum       -2.739e-01  1.435e-02 -19.079   <2e-16 ***
## malaria_ecology  1.941e-02  8.712e-04  22.279   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9778 on 20319 degrees of freedom
##   (1497 observations deleted due to missingness)
## Multiple R-squared:  0.06039,    Adjusted R-squared:  0.06016
## F-statistic: 261.2 on 5 and 20319 DF,  p-value: < 2.2e-16
```

It's pretty easy to extend the two-variable regression to arbitrary independent variables. We just keep adding them in a linear fashion:

$$y_i = \beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2 + \cdots + x_{ik}\beta_k + u_i$$

Why not just write it this way? First, the notation is going to get needlessly messy as we add variables. Second, there are a lot of concepts that are easier to define and discuss once we have matrix notation.

MATRIX ALGEBRA REVIEW

Vectors

A **vector** is a list of numbers arranged, usually represented as a column. Specifically, a **column vector** is a list of k numbers arranged as a column:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix}$$

A **row vector**, on the other hand, is $1 \times k$ list of numbers arranged as a row:

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_K \end{bmatrix}$$

Unless otherwise stated, we'll assume that a vector is column vector and vectors will be written with lowercase bold lettering ($\mathbf{b}$). For reasons that will become clear, we'll refer to column vector as $k \times 1$ and a row vector as $1 \times k$.

One way that we can already make our notation more compact is by writing the list of covariate values for unit i as a vector. That is, if we have k covariates, then we'll

define the $(k+1) \times 1$ vector of covariates, $\mathbf{x}_i$, as:

$$\mathbf{x}_i = \begin{bmatrix} 1 \\ x_{i1} \\ x_{i2} \\ \vdots \\ x_{ik} \end{bmatrix} \qquad \mathbf{x}_i = \begin{bmatrix} 1 \\ \text{exports}_i \\ \text{age}_i \\ \text{male}_i \end{bmatrix}$$

Notice that we can also write all of the coefficients as a $(k+1) \times 1$ vector as well:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}$$

It is useful to be able to switch a column vector to a row vector and vice versa. This is exactly what the transpose operation does. For instance, if $\mathbf{a}$ is a $k \times 1$ column vector, then its **transpose** $\mathbf{a}'$ is the same vector in $1 \times k$ row form. For instance, we can write the vector of independent variables in row form with the transpose:

$$\mathbf{x}_i' = \begin{bmatrix} 1 & x_{i1} & x_{i2} & \cdots & x_{ik} \end{bmatrix}$$

Vector Addition and Multiplication

How can we extend the ideas of addition and multiplication to vectors? To add two vectors together, it only makes sense if they have the same length. For example, let $\mathbf{a}$ and $\mathbf{b}$ be $k \times 1$ vectors. We define their sum as the sum of each entry:

$$\mathbf{a} + \mathbf{b} = \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix} = \begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \vdots \\ a_k + b_k \end{bmatrix}$$

The first type of multiplication we can do with vectors is **scalar multiplication** of a vector $b\mathbf{a}$, where we just multiply each element of the vector by b:

$$b\mathbf{a} = \mathbf{a}b = \begin{bmatrix} ba_1 \\ ba_2 \\ \vdots \\ ba_k \end{bmatrix}$$

There is a special way to multiply two (column) vectors of equal length that will allow us to write the linear model very compactly. The **inner product** of a two column vectors $\mathbf{a}$ and $\mathbf{b}$ (of equal dimension, $k \times 1$) is just the transpose of the first multiplied by the second:

$$\mathbf{a}'\mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots + a_k b_k$$

The intuition here is that this gives the length of the $\mathbf{a}$ vector in the "direction" of the $\mathbf{b}$ vector. With this in hand, we can write the linear model in a very compact way:

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + u_i$$

For instance, last week, we had the two variables (plus a constant). We could write things this way:

$$\mathbf{x}_i'\boldsymbol{\beta} = \begin{bmatrix} 1 & x_{i1} & x_{i2} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2$$

Here, $\mathbf{x}_i'\boldsymbol{\beta}$ can be thought of as the "length" vector of independent variable in the direction of the covariates. More generally, you can also think of this as generalizing the multiplication of $X_i\beta_1$ to multiple covariates.

Another important example of where we use the inner product is with the residuals. Let $\widehat{\mathbf{u}}$ be the $n \times 1$ vector of residual. Then, the inner product of the residuals is:

$$\widehat{\mathbf{u}}'\widehat{\mathbf{u}} = \begin{bmatrix} \widehat{u}_1 & \widehat{u}_2 & \cdots & \widehat{u}_n \end{bmatrix} \begin{bmatrix} \widehat{u}_1 \\ \widehat{u}_2 \\ \vdots \\ \widehat{u}_n \end{bmatrix}$$

$$\widehat{\mathbf{u}}'\widehat{\mathbf{u}} = \widehat{u}_1\widehat{u}_1 + \widehat{u}_2\widehat{u}_2 + \cdots + \widehat{u}_n\widehat{u}_n = \sum_{i=1}^{n} \widehat{u}_i^2$$

It's just the sum of the squared residuals!

Matrices

A **matrix** is just a rectangular array of numbers. We say that a matrix is $n \times k$ if it has n rows and k columns. We write a matrix with uppercase bold letters:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nk} \end{bmatrix}$$

We will often need to refer to some generic entry (or cell) of a matrix and we can do this with a_{ij} where this is the entry in row i and column j. There is nothing special about these matrices. They are basically just like spreadsheets in Excel or the like. It's a way to group numbers.

One example of a matrix that we'll use a lot is the **design matrix**, which has a column of ones, and then each of the subsequent columns is each independent variable in the regression.

$$\mathbf{X} = \begin{bmatrix} 1 & \text{exports}_1 & \text{age}_1 & \text{male}_1 \\ 1 & \text{exports}_2 & \text{age}_2 & \text{male}_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \text{exports}_n & \text{age}_n & \text{male}_n \end{bmatrix}$$

It is often useful to refer to matrices as a group of column or row vectors. For instance, we can write a 2×3 matrix as a two 1×3 row vectors:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1' \\ \mathbf{a}_2' \end{bmatrix}$$

with row vectors

$$\mathbf{a}_1' = \begin{bmatrix} a_{11} & a_{12} & a_{13} \end{bmatrix} \qquad \mathbf{a}_2' = \begin{bmatrix} a_{21} & a_{22} & a_{23} \end{bmatrix}$$

We can define the following 3×2 matrix in terms of two 3×1 column vectors:

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} \mathbf{b_1} & \mathbf{b_2} \end{bmatrix}$$

where $\mathbf{b_1}$ and $\mathbf{b_2}$ represent the columns of $\mathbf{B}$.

It should be clear what is what: matrices defined by column will be written horizontally, whereas matrices defined by row will be written vertically. Also, we'll use j and s as subscripts for columns of a matrix: $\mathbf{x}_j$ or $\mathbf{x}_s$, whereas i and t will be used for rows $\mathbf{x}_i'$.

The **transpose** of a matrix $\mathbf{A}$ is the matrix created by switching the rows and columns of the data and is denoted $\mathbf{A}'$. That is, the kth column becomes the kth row:

$$\mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \\ q_{31} & q_{32} \end{bmatrix} \quad \mathbf{Q}' = \begin{bmatrix} q_{11} & q_{21} & q_{31} \\ q_{12} & q_{22} & q_{32} \end{bmatrix}$$

If $\mathbf{A}$ is $j \times k$, then $\mathbf{A}'$ will be $k \times j$.

Note, though, that R always prints a vector in row form, even if it is a column in the original data.

```
head(nunn$trust_neighbors)
```

```
## [1] 3 3 0 0 1 1
```

One thing to watch out for is that R doesn't really distinguish between row vectors and columns vectors by default. In R, a vector is just a list of numbers. For instance, if you try to use `nrow()` or `ncol()` on a vector, R is confused:

```
ncol(nunn$trust_neighbors)
```

```
## NULL
```

```
nrow(nunn$trust_neighbors)
```

```
## NULL
```

Vectors in R are special constructs and you have to use `length()` to see how many entries there are in the vector:

```
length(nunn$trust_neighbors)
```

```
## [1] 21822
```

You can convert vectors to have explicit dimensions by using `as.matrix()`, but beware that R assumes all vectors are column vectors:

```
nrow(as.matrix(nunn$trust_neighbors))
```

```
## [1] 21822
```

```
ncol(as.matrix(nunn$trust_neighbors))
```

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

We can transpose in R using the `t()` function:

```r
a <- matrix(1:6, ncol = 3, nrow = 2)
a
```

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

```r
t(a)
```

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

Finally, we can see the design matrix using the `model.matrix()` function in R. And the `dim()` function can help us get the dimensions of the matrix quickly:

```r
head(model.matrix(mod), 5)
```

```
##   (Intercept)  exports age male urban_dum malaria_ecology
## 1           1 854.9581  40    0         0        28.14704
## 2           1 854.9581  25    1         0        28.14704
## 3           1 854.9581  38    1         1        28.14704
## 4           1 854.9581  37    0         1        28.14704
## 5           1 854.9581  31    1         0        28.14704
```

```r
dim(model.matrix(mod))
```

```
## [1] 20325     6
```

Addition and subtraction for matrices

How do we add or subtract matrices? Like vectors, matrices have to have the same dimensions to be added together. When this is true, we say they are **comformable**, meaning that the dimensions have to be the same. Let $\mathbf{A}$ and $\mathbf{B}$ both be 2×2 matrices. Then, let $\mathbf{A} + \mathbf{B}$, where we add each cell together:

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$$

Scalar multiplication for matrices is very similar to vectors and we just multiply each element/cell by that scalar:

$$b\mathbf{A} = b \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} b \times a_{11} & b \times a_{12} \\ b \times a_{21} & b \times a_{22} \end{bmatrix}$$

With these tools in hand, we can develop another matrix-based notation for the linear model. Remember that we wrote the linear model as the following for all $i \in [1, \ldots, n]$:

$$y_i = \beta_0 + x_i\beta_1 + z_i\beta_2 + u_i$$

Imagine we had an n of 4. We could write out each formula:

$$\begin{aligned} y_1 &= \beta_0 + x_1\beta_1 + z_1\beta_2 + u_1 \quad \text{(unit 1)} \\ y_2 &= \beta_0 + x_2\beta_1 + z_2\beta_2 + u_2 \quad \text{(unit 2)} \\ y_3 &= \beta_0 + x_3\beta_1 + z_3\beta_2 + u_3 \quad \text{(unit 3)} \\ y_4 &= \beta_0 + x_4\beta_1 + z_4\beta_2 + u_4 \quad \text{(unit 4)} \end{aligned}$$

We can write this with vectors form like so:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \beta_0 + \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \beta_1 + \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} \beta_2 + \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

Hopefully it's clear in this notation that the column vector of the outcomes is a linear combination of the independent variables and the error, with the β coefficients acting as the weights. Can we write this in a more compact form? Yes, but we need to develop a notation and idea of matrix multiplication. Let $\mathbf{X}$ and $\boldsymbol{\beta}$ be the following:

$$\underset{(4\times3)}{\mathbf{X}} = \begin{bmatrix} 1 & x_1 & z_1 \\ 1 & x_2 & z_2 \\ 1 & x_3 & z_3 \\ 1 & x_4 & z_4 \end{bmatrix} \qquad \underset{(3\times1)}{\boldsymbol{\beta}} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

Matrix multiplication by a vector

We will define multiplication of a matrix by a vector in the following way:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \beta_0 + \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \beta_1 + \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} \beta_2 = \mathbf{X}\boldsymbol{\beta}$$

Thus, multiplication of a matrix by a vector is just the **linear combination** of the columns of the matrix with the vector elements as weights/coefficients. And the left-hand side here only uses scalars times vectors, which is easy! In general, let's say that we have a $n \times k$ matrix $\mathbf{A}$ and a $k \times 1$ column vector $\mathbf{b}$ (notice that the number of columns of the matrix is the same as the number of rows of the vector)

Let $\mathbf{a}_j$ be the jth column of $\mathbf{A}$. Then we can write:

$$\underset{(n\times 1)}{\mathbf{c}} = \mathbf{A}\mathbf{b} = b_1\mathbf{a}_1 + b_2\mathbf{a}_2 + \cdots + b_k\mathbf{a}_k$$

Thus, now let $\mathbf{X}$ be the $n \times (k+1)$ matrix of independent variables and $\boldsymbol{\beta}$ be the $(k+1) \times 1$ column vector of coefficients. Then:

$$\mathbf{X}\boldsymbol{\beta} = \beta_0 + \beta_1\mathbf{x}_1 + \beta_2\mathbf{x}_2 + \cdots + \beta_k\mathbf{x}_k$$

Thus, we can compactly write the linear model as the following:

$$\underset{(n\times 1)}{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta} + \underset{(n\times 1)}{\mathbf{u}}$$

An equivalent way to define matrix multiplication is with the inner product. Let $\mathbf{a}_i'$ be the ith row of the matrix, $\mathbf{A}$. Then, we can write each entry as:

$$c_i = \mathbf{a}_i'\mathbf{b}$$

With this definition, it is clear that our statement of the linear model is equivalent to:

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + u_i$$

Matrix multiplication

What if, instead of a column vector $\mathbf{b}$, we have a matrix $\mathbf{B}$ with dimensions $k \times m$. How do we do multiplication like so $\mathbf{C} = \mathbf{A}\mathbf{B}$? Each column of the new matrix is just matrix by vector multiplication:

$$\mathbf{C} = [\mathbf{c}_1 \quad \mathbf{c}_2 \quad \cdots \quad \mathbf{c}_m] \qquad \mathbf{c}_j = \mathbf{A}\mathbf{b}_j$$

Thus, each column of $\mathbf{C}$ is a linear combination of the columns of $\mathbf{A}$.

Special matrices

A **square matrix** is one with equal numbers of rows and columns. The **diagonal** of a square matrix are the values in which the row number is equal to the column number:

a_{11} or a_{22}, etc.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

To get the diagonal of a matrix in R, use the `diag()` function:

```
b <- matrix(1:4, nrow = 2, ncol = 2)
b
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

```
diag(b)
```

```
## [1] 1 4
```

The **identity matrix**, $\mathbf{I}_k$ is a square $k \times k$ matrix, with 1s along the diagonal and 0s everywhere else.

$$\mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The identity matrix multiplied by any matrix just returns the matrix: $\mathbf{AI} = \mathbf{A}$. Here, as is often the case, the dimension of the identity matrix is left implicit—from the context of the equation we know it must be equal to the number of columns of $\mathbf{A}$.

To create an identity matrix in R, you can also use the `diag()` function, but this time just pass it a number instead of a matrix:

```
diag(3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

OLS IN MATRIX FORM

Multiple linear regression in matrix form

Let $\widehat{\boldsymbol{\beta}}$ be the $(k+1) \times 1$ vector of estimated regression coefficients:

$$\widehat{\boldsymbol{\beta}} = \begin{bmatrix} \widehat{\beta}_0 \\ \widehat{\beta}_1 \\ \vdots \\ \widehat{\beta}_k \end{bmatrix}$$

Given the above derivations, it is clear that we can write the fitted values of the outcome as:

$$\widehat{\mathbf{y}} = \mathbf{X}\widehat{\boldsymbol{\beta}}$$

It might be helpful to see this again more written out:

$$\widehat{\mathbf{y}} = \begin{bmatrix} \widehat{y}_1 \\ \widehat{y}_2 \\ \vdots \\ \widehat{y}_n \end{bmatrix} = \mathbf{X}\widehat{\boldsymbol{\beta}} = \begin{bmatrix} 1\widehat{\beta}_0 + x_{11}\widehat{\beta}_1 + x_{12}\widehat{\beta}_2 + \cdots + x_{1K}\widehat{\beta}_k \\ 1\widehat{\beta}_0 + x_{21}\widehat{\beta}_1 + x_{22}\widehat{\beta}_2 + \cdots + x_{2K}\widehat{\beta}_k \\ \vdots \\ 1\widehat{\beta}_0 + x_{n1}\widehat{\beta}_1 + x_{n2}\widehat{\beta}_2 + \cdots + x_{nK}\widehat{\beta}_k \end{bmatrix}$$

Residuals

With these fitted values, we can easily write the **residuals** in vector form:

$$\widehat{\mathbf{u}} = \mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}$$

The goal of OLS remains the same: to minimize the sum of the squared residuals. Earlier, we saw that we can write this compactly with the inner product:

$$\widehat{\mathbf{u}}'\widehat{\mathbf{u}} = (\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})'(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})$$

Using matrix calculus to minimize this function, we arrive at the following first-order condition:

$$0 = \mathbf{X}'(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})$$

Rearranging, we have the following formula for the OLS estimator:

$$\mathbf{X}'\mathbf{X}\widehat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y}$$

In order to isolate $\widehat{\boldsymbol{\beta}}$, we need to move the $\mathbf{X}'\mathbf{X}$ term to the other side of the equals sign. We've learned about matrix multiplication, but what about matrix "division"?

Matrix Inversion

What is division in its simplest form? $\frac{1}{a}$ is the value such that $a\frac{1}{a} = 1$. For some algebraic expression: $au = b$, let's solve for u:

$$\begin{aligned} \frac{1}{a}au &= \frac{1}{a}b \\ u &= \frac{b}{a} \end{aligned}$$

We need a matrix version of this: $\frac{1}{a}$.

Definition If it exists, the **inverse** of square matrix $\mathbf{A}$, denoted $\mathbf{A}^{-1}$, is the matrix such that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$.

We can use the inverse to solve (systems of) equations:

$$\begin{aligned} \mathbf{Au} &= \mathbf{b} \\ \mathbf{A^{-1}Au} &= \mathbf{A^{-1}b} \\ \mathbf{Iu} &= \mathbf{A^{-1}b} \\ \mathbf{u} &= \mathbf{A^{-1}b} \end{aligned}$$

If the inverse exists, we say that $\mathbf{A}$ is **invertible** or **nonsingular**.

Let's assume, for now, that the inverse of $\mathbf{X}'\mathbf{X}$ exists (we'll come back to this) Then we can write the OLS estimator as the following:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

Memorize this: "x prime x inverse x prime y" sear it into your soul.

OLS by hand in R

Let's skip the `lm()` function and compute the coefficients directly. First we need to get the design matrix:

```
X <- model.matrix(trust_neighbors ~ exports + age + male + urban_dum + malaria_ecology, data = nunn)
dim(X)
```

```
## [1] 20325     6
```

```
## model.frame always puts the response in the first column
y <- model.frame(trust_neighbors ~ exports + age + male + urban_dum + malaria_ecology, data = nunn)[,1]
```

```
## solve() does inverses
## and %*% is matrix multiplication
solve(t(X) %*% X) %*% t(X) %*% y
```

```
##                         [,1]
## (Intercept)      1.503037046
## exports         -0.001020836
## age              0.005044682
## male             0.027836875
## urban_dum       -0.273871917
## malaria_ecology  0.019410561
```

```
coef(mod)
```

```
##     (Intercept)         exports             age            male
##     1.503037046    -0.001020836     0.005044682     0.027836875
##       urban_dum malaria_ecology
##    -0.273871917     0.019410561
```

Intuition for the OLS in matrix form

What's the intuition here for this formula? First, note that the "numerator" $\mathbf{X}'\mathbf{y}$ is roughly composed of the covariances between the columns of X and y. Next, the "denominator" $\mathbf{X}'\mathbf{X}$ is roughly composed of the sample variances and covariances of variables within $\mathbf{X}$. Thus, we have something like:

$$\widehat{\boldsymbol{\beta}}_{-0} \approx (\text{variance of } \mathbf{X})^{-1} (\text{covariance of } \mathbf{X} \ \& \ \mathbf{y})$$

Here, $\widehat{\boldsymbol{\beta}}_{-0}$ is the vector of coefficients without the intercept/constant. We're sidestepping the exact definitions of the variance and covariance of a matrix here.

OLS INFERENCE IN MATRIX FORM

A **random vector** is a vector of random variables:

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}$$

Here, $\mathbf{x}_i$ is a random vector and x_{i1} and x_{i2} are random variables. When we talk about the distribution of $\mathbf{x}_i$, we are talking about the joint distribution of x_{i1} and x_{i2}. The

expectation of this random vector is just the expectation of each r.v. in the vector:

$$\mathbb{E}[\mathbf{x}_i] = \begin{bmatrix} \mathbb{E}[x_{i1}] \\ \mathbb{E}[x_{i2}] \end{bmatrix}$$

The variance of random vectors has to tell us both the variance of each r.v., but also the covariance between them. Thus, we often call this the **variance-covariance matrix**:

$$\mathbb{V}[\mathbf{x}_i] = \mathbb{E}\left[(\mathbf{x}_i - \mathbb{E}[\mathbf{x}_i])(\mathbf{x}_i - \mathbb{E}[\mathbf{x}])'\right] = \begin{bmatrix} \mathbb{V}[x_{i1}] & \text{Cov}[x_{i1}, x_{i2}] \\ \text{Cov}[x_{i1}, x_{i2}] & \mathbb{V}[x_{i2}] \end{bmatrix}$$

Combined, these describe the joint distribution of $\mathbf{x}_i$.

Most general OLS assumptions

1. Linearity: $y_i = \mathbf{x}_i'\boldsymbol{\beta} + u_i$
2. Random/iid sample: $(y_i, \mathbf{x}_i')$ are a iid sample from the population.
3. No perfect collinearity: $\mathbf{X}$ is an $n \times (k+1)$ matrix with rank $k+1$
4. Zero conditional mean: $\mathbb{E}[u_i|\mathbf{x}_i] = \mathbf{0}$
5. Homoskedasticity: $\mathbb{V}[u_i|\mathbf{x}_i] = \sigma_u^2$
6. Normality: $u_i|\mathbf{x}_i \sim N(0, \sigma_u^2)$

No perfect collinearity

In matrix form: $\mathbf{X}$ is an $n \times (k+1)$ matrix with rank $k+1$. The **rank** of a matrix is the maximum number of linearly independent columns. If $\mathbf{X}$ has rank $k+1$, then all of its columns are linearly independent and none of its columns are linearly dependent implies no perfect collinearity between any of the columns.

Why do we care about this? If $\mathbf{A}$ is a square $k \times k$ matrix and has rank k, then there exists a unique inverse, $\mathbf{A}'$. Furthermore, if $\mathbf{X}$ has rank $k+1$, then $(\mathbf{X}'\mathbf{X})$ has rank $k+1$. Thus, in order for the OLS coefficients to even be computed, we need full rank of $\mathbf{X}$. This is very similar to needing variation in X to be able to divide by the variance in simple OLS.

Bias and Variance of OLS

It is useful to write the OLS estimator in the following way:

$$\begin{aligned} \widehat{\beta} &= \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{y} \\ &= \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'(\mathbf{X}\beta + \mathbf{u}) \\ &= \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{X}\beta + \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{u} \\ &= \beta + \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{u} \end{aligned}$$

With this in hand, under matrix assumptions 1-4, OLS is conditionally unbiased for β:

$$\begin{aligned}\mathbb{E}[\widehat{\beta}|\mathbf{X}] &= \beta + \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbb{E}[\mathbf{u}|\mathbf{X}] \\ &= \beta + \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{0} \\ &= \beta\end{aligned}$$

This also implies that OLS is unconditionally unbiased: $\mathbb{E}[\widehat{\beta}] = \beta$

What about the sampling variance of the OLS estimator, $\mathbb{V}[\widehat{\beta}|\mathbf{X}]$? First, note that if we have a vector of constants, $\mathbf{b}$, a matrix of constants, $\mathbf{A}$ and a random vector $\mathbf{x}$, then $\mathbb{V}[\mathbf{b} + \mathbf{A}\mathbf{x}] = \mathbf{A}\mathbb{V}[\mathbf{x}]\mathbf{A}'$. This is the matrix version of "squaring" a constant in the variance of a single random variable.

Plugging in the expression for the OLS estimator and using this fact, we find:

$$\begin{aligned}\mathbb{V}[\widehat{\beta}|\mathbf{X}] &= \mathbb{V}[\beta + \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{u}|\mathbf{X}] \\ &= \mathbb{V}[\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{u}|\mathbf{X}] \\ &= \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbb{V}[\mathbf{u}|\mathbf{X}](\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}')'\end{aligned}$$

Can we make progress here? Yes, if use the property that if $\mathbf{A}$ is symmetric, then $\left(\mathbf{A}^{-1}\right)' = \mathbf{A}^{-1}$. Thus, $((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}')' = \mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}$ and plugging back into the variance:

$$\mathbb{V}[\widehat{\beta}|\mathbf{X}] = \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbb{V}[\mathbf{u}|\mathbf{X}]\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}$$

Can we make more progress? Yes! What the covariance matrix of the errors, $\mathbb{V}[\mathbf{u}|\mathbf{X}]$?

$$\mathbb{V}[\mathbf{u}|\mathbf{X}] = \begin{bmatrix} \mathbb{V}[u_1|\mathbf{X}] & \text{cov}[u_1, u_2|\mathbf{X}] & \dots & \text{cov}[u_1, u_n|\mathbf{X}] \\ \text{cov}[u_2, u_1|\mathbf{X}] & \mathbb{V}[u_2|\mathbf{X}] & \dots & \text{cov}[u_2, u_n|\mathbf{X}] \\ \vdots & & \ddots & \\ \text{cov}[u_n, u_1|\mathbf{X}] & \text{cov}[u_n, u_2|\mathbf{X}] & \dots & \mathbb{V}[u_n|\mathbf{X}] \end{bmatrix}$$

Note that this matrix is symmetric since $\text{cov}(u_i, u_j) = \text{cov}(u_i, u_j)$. By homoskedasticity and iid, , we have constant variance $\mathbb{V}[u_i|\mathbf{X}] = \mathbb{V}[u_i|\mathbf{x}_i] = \sigma_u^2$ and uncorrelated errors, $\text{cov}[u_s, u_t|\mathbf{X}] = 0$. Then, the covariance matrix of the errors is simply:

$$\mathbb{V}[\mathbf{u}|\mathbf{X}] = \sigma_u^2\mathbf{I}_n = \begin{bmatrix} \sigma_u^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_u^2 & 0 & \dots & 0 \\ & & & \vdots & \\ 0 & 0 & 0 & \dots & \sigma_u^2 \end{bmatrix}$$

Puttting this back into our expression for the OLS sampling variance, we get:

$$\begin{aligned}\mathbb{V}[\widehat{\beta}|\mathbf{X}] &= \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbb{V}[\mathbf{u}|\mathbf{X}]\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1} \\ &= \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'(\sigma_u^2\mathbf{I}_n)\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1} \\ &= \sigma_u^2\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1} \\ &= \sigma^2\left(\mathbf{X}'\mathbf{X}\right)^{-1}\end{aligned}$$

Inference in the general setting

Under assumptions 1-5, the sampling variance of the OLS estimator can be written in matrix form as the following:

$$\mathbb{V}\left[\widehat{\boldsymbol{\beta}}|\mathbf{X}\right] = \sigma_u^2(\mathbf{X}'\mathbf{X})^{-1}$$

This variance matrix of the OLS estimator looks like this:

$$\begin{bmatrix} \mathbb{V}\left[\widehat{\beta}_0|\mathbf{X}\right] & \text{Cov}\left[\widehat{\beta}_0,\widehat{\beta}_1|\mathbf{X}\right] & \text{Cov}\left[\widehat{\beta}_0,\widehat{\beta}_2|\mathbf{X}\right] & \cdots & \text{Cov}\left[\widehat{\beta}_0,\widehat{\beta}_k|\mathbf{X}\right] \\ \text{Cov}\left[\widehat{\beta}_0,\widehat{\beta}_1|\mathbf{X}\right] & \mathbb{V}\left[\widehat{\beta}_1|\mathbf{X}\right] & \text{Cov}\left[\widehat{\beta}_1,\widehat{\beta}_2|\mathbf{X}\right] & \cdots & \text{Cov}\left[\widehat{\beta}_1,\widehat{\beta}_k|\mathbf{X}\right] \\ \text{Cov}\left[\widehat{\beta}_0,\widehat{\beta}_2|\mathbf{X}\right] & \text{Cov}\left[\widehat{\beta}_1,\widehat{\beta}_2|\mathbf{X}\right] & \mathbb{V}\left[\widehat{\beta}_2|\mathbf{X}\right] & \cdots & \text{Cov}\left[\widehat{\beta}_2,\widehat{\beta}_k|\mathbf{X}\right] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \text{Cov}\left[\widehat{\beta}_0,\widehat{\beta}_k|\mathbf{X}\right] & \text{Cov}\left[\widehat{\beta}_k,\widehat{\beta}_1|\mathbf{X}\right] & \text{Cov}\left[\widehat{\beta}_k,\widehat{\beta}_2|\mathbf{X}\right] & \cdots & \mathbb{V}\left[\widehat{\beta}_k|\mathbf{X}\right] \end{bmatrix}$$

With this sampling variance in hand, we can appeal to the usual results to do inference, get confidence intervals, and perform hypothesis tests. For instance, under assumption 1-5, by the central limit theorem, in large samples:

$$\frac{\widehat{\beta}_j - \beta_j}{\widehat{\mathsf{se}}[\widehat{\beta}_j]} \sim N(0,1)$$

In small samples, under assumptions 1-6,

$$\frac{\widehat{\beta}_j - \beta_j}{\widehat{\mathsf{se}}[\widehat{\beta}_j]} \sim t_{n-(k+1)}$$

Thus, under the null of $H_0 : \beta_j = 0$, we know that $\widehat{\beta}_j/\widehat{\mathsf{se}}[\widehat{\beta}_j] \sim t_{n-(k+1)}$. Here, the estimated SEs come from: $\widehat{\mathbb{V}}[\widehat{\boldsymbol{\beta}}] = \widehat{\sigma}_u^2(\mathbf{X}'\mathbf{X})^{-1}$, where

$$\widehat{\sigma}_u^2 = \frac{\widehat{\mathbf{u}}'\widehat{\mathbf{u}}}{n-(k+1)} = \frac{\sum_{i=1}^n u_i^2}{n-(k+1)}.$$

We can access this estimated covariance matrix in R:

```
vcov(mod)
```

```
##                   (Intercept)       exports           age          male
## (Intercept)      4.766593e-04  1.163698e-07 -7.956151e-06 -6.675717e-05
## exports          1.163698e-07  1.676040e-09 -3.658689e-10  7.282947e-09
## age             -7.956151e-06 -3.658689e-10  2.231299e-07 -7.764680e-07
## male            -6.675717e-05  7.282947e-09 -7.764680e-07  1.908894e-04
## urban_dum       -9.658428e-05 -4.861159e-08  7.107867e-07 -1.711373e-06
## malaria_ecology -6.909410e-06 -2.124140e-08  2.324132e-10 -1.017404e-07
##                      urban_dum malaria_ecology
## (Intercept)     -9.658428e-05   -6.909410e-06
## exports         -4.861159e-08   -2.124140e-08
## age              7.107867e-07    2.324132e-10
## male            -1.711373e-06   -1.017404e-07
## urban_dum        2.060633e-04    2.723938e-09
## malaria_ecology  2.723938e-09    7.590439e-07
```

Note that the diagonal are the variances. So the square root of the diagonal is are the standard errors:

```
sqrt(diag(vcov(mod)))
```

```
##     (Intercept)         exports             age            male
##    2.183253e-02    4.093947e-05    4.723663e-04    1.381627e-02
##       urban_dum malaria_ecology
##    1.435491e-02    8.712313e-04
```

```
coef(summary(mod))[, "Std. Error"]
```

```
##     (Intercept)         exports             age            male
##    2.183253e-02    4.093947e-05    4.723663e-04    1.381627e-02
##       urban_dum malaria_ecology
##    1.435491e-02    8.712313e-04
```

APPENDIX

Covariance/variance interpretation of matrix OLS

$$\mathbf{X}'\mathbf{y} = \sum_{i=1}^{n} \begin{bmatrix} y_i \\ y_i x_{i1} \\ y_i x_{i2} \\ \vdots \\ y_i x_{iK} \end{bmatrix} \approx \begin{bmatrix} n\overline{y} \\ \widehat{\text{Cov}}[y_i, x_{i1}] \\ \widehat{\text{Cov}}[y_i, x_{i2}] \\ \vdots \\ \widehat{\text{Cov}}[y_i, x_{ik}] \end{bmatrix}$$

$$\mathbf{X}'\mathbf{X} = \sum_{i=1}^{n} \begin{bmatrix} 1 & x_{i1} & x_{i2} & \cdots & x_{ik} \\ x_{i1} & x_{i1}^2 & x_{i2}x_{i1} & \cdots & x_{i1}x_{ik} \\ x_{i2} & x_{i1}x_{i2} & x_{i2}^2 & \cdots & x_{i2}x_{ik} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{ik} & x_{i1}x_{ik} & x_{i2}x_{ik} & \cdots & x_{ik}x_{ik} \end{bmatrix}$$

$$\approx \begin{bmatrix} n & n\overline{x}_1 & n\overline{x}_2 & \cdots & n\overline{x}_k \\ n\overline{x}_1 & \widehat{\mathbb{V}}[x_{i1}] & \widehat{\text{Cov}}[x_{i1}, x_{i2}] & \cdots & \widehat{\text{Cov}}[x_{i1}, x_{ik}] \\ n\overline{x}_2 & \widehat{\text{Cov}}[x_{i2}, x_{i1}] & \widehat{\mathbb{V}}[x_{i2}] & \cdots & \widehat{\text{Cov}}[x_{i2}, x_{ik}] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ n\overline{x}_k & \widehat{\text{Cov}}[x_{ik}, x_{i1}] & \widehat{\text{Cov}}[x_{ik}, x_{i2}] & \cdots & \widehat{\mathbb{V}}[x_{ik}] \end{bmatrix}$$