**The matrix algebra of linear regression**

**Regression in matrix form**

Here is the matrix algebra formula for linear regression: **y** = **Xb**. A matrix is designated by an upper-case bold letter, while a vector (i.e., matrix with only one column) is designated by a lower-case bold letter. Therefore, **X** is a matrix and **b** and **y** are vectors. Although mathematical equations make problems appear complicated, you already know everything in this equation. **X** is the matrix of your predictors. Imagine an SPSS file in “data view” with 3 predictor variables next to each other. That is your **X** matrix. The only difference is that matrix algebra thinks of the intercept as a “predictor”. Even though it conceptually is not, mathematically is it. The intercept is represented by a constant – specifically a column of all 1s. A column of any constant (say 9s) will give the same regression slopes, but a different intercept. Essentially the intercept will be on a different axis if the values are not all 1. Let’s look at what this **X** matrix looks like with an example dataset of 10 cases (header rows added for clarity):

The **y** vector is just the column for your outcome and again looks like the SPSS “data view”.

The **b** vector contains your regression coefficients: one intercept (b0) and three partial regression slopes (b1, b2, b3).

If we put them together into our overall matrix equation **y = Xb**, we get:

You will notice there is a multiplication sign in between the predictor matrix **X** and the vector of regression coefficients **b**. This is not your standard multiplication operator as we are multiplying two matrices (i.e., matrix multiplication). If we were multiplying a matrix by a single value (e.g., 5\***X**), then we would simply multiple each element, or number, in the matrix by that value (i.e., scalar multiplication). With matrix multiplication you are taking linear combinations of columns and rows. While the multiplication of two full matrices can get complicated, multiplying a matrix and vector is relatively easy. It is simply the addition of each element of the vector multiplied by a column of the matrix. Thus, after the matrix multiplication we have the following formula:

**The column space of y**

The above formula might appear familiar to you. It is essentially the “usual” formula for linear regression:

However, notice there is no error term in this equation and y is “actual y” not “predicted y” (i.e., y-hat). This equation with a literal “= sign” assumes we have perfect prediction: that our three predictors can perfectly predict y (i.e., R2 = 1). And that is a very rarely the case. From an algebra point of view, there are no values for the **b** vector - b0, b1, b2, and b3 – that can satisfy the equality. There is no solution. You can see why this is from our application of matrix multiplication. If a linear combination of the predictors can equal the outcome, then you have perfect prediction.

So if the matrix equation **y = Xb** has no solution, we cannot directly solve for **b**. Instead, we find the component of **y** in the column space of **X**. “Spaces” in matrix algebra are groups of matrices or vectors that share some property. A column space contains all possible vectors that could result from all possible **X\*b**, where **b** can take on any real numbers. Vector spaces are a little abstract, so let’s look at an example. If we ignore **y** for the moment (i.e., the left hand side) we have the following matrix equation:

Now, **X** stays the same, but we plug in any values we want for **b**. Here is one of the simplest possibilities: b0 = 1, b1 = 1, b2 = 1, and b3 = 1. If we plug those values in for **b**, we get the following left hand side, *which is in the column space of* ***X***:

Here is another vector in the column space of **X**: b0 = 2, b1 = 3, b2 = 4, and b3 = 5.

And here is another vector in the column space of X: b0 = -50.237, b1 = 10.761, b2 = 0.042, and b3 = -0.876.

I only list three, but there are an infinite number of vectors in the column space **X** because there are an infinite number of different **b**s. However, that does not include every vector. For example, the following vector is not in the column space of **X**:

Although there are an infinite number of vectors in the column space of **X**, not all possible vectors are in the column space of **X**. This is analogous to the fact that there are an infinite number of real numbers between 0 and 1 (e.g., 0.53643…); however, not all real numbers are between 0 and 1 (e.g, 1.23425…). Another vector not in the column space of X is **y**:

This is expected. If **y**, was in the column space of **X**, then a linear combination of the predictors could equal the outcome, and we would have perfect prediction (i.e., R2 = 1). Instead **y** is almost always not in the column space of **X** and no linear combination of the predictors can equal the outcome. As stated above, this is why there is no solution to the equation **y** = **Xb** and we have to find the *component* of **y** in the column space of **X**.

**The projection vector**

The process of finding the component of a vector (or matrix) in a column space is called projection. It is a rather common operation in matrix algebra and used in many other mathematical applications. Projecting **y** will split the vector into two orthogonal components. In the context of multiple regression, one component will be the predicted values of **y**, also known as **y-hat**, and one component will be the residual values of **y**, also known as **e**. These two components will add up to **y** (i.e., **y** = **y-hat** + **e**). Therefore, if we know one component, we can easily solve for the other. For example, we usually estimate **y-hat** and then calculate **e** thereafter (e.g., **e** = **y** – **y-hat**).

The formula for the projection of **y**, **y-hat**, is found by simply pre-multiplying XT (i.e, multiply **X**T to the left) by both sides of the linear regression matrix equation (i.e., **y** = **Xb**). **X**T is the transpose of the **X** matrix (sometimes written **X**’). The transpose is actually very simple – you just switch the rows and columns of a matrix: The first row is now the first column and the first column is now the first row; the second row is now the second column and the second column is now the second row; etc. Therefore **X**T is (first column is a header):

We then get, **X**T**y** = **X**T**Xb**. If you wanted to solve for the projection vector of **y**, we could simply divide both sides by **X**T and get **y** = (**X**T)-1\*(**X**T**X)\*b.** However, 1) that’s not really what we want (the projection vector would be the predicted values of y, or **y-hat**) and 2) we need **b** before we can solve for it. Now to solve for **b**, we need to divide by **XTX** and get **b** alone on one side of the equal sign. In matrix algebra, division is multiplying be inverses. This is still valid in scaler algebra. For example, if we had the formula d = cb, we could solve for **b**, by either dividing by c *or* multiplying each side by c-1 (i.e., 1/c). In matrix algebra we always use the c-1 method. Therefore, to get **b** alone on the right side of the equation, we multiply each side by (**X**T**X)**-1. This gives us the final matrix algebra equation for the regression coefficients: **b** = (**X**T**X)**-1 \***(X**T**y).** The reason the (**XTX)**-1 goes on the left side of the **X**T**y** expression (i.e., (**XTX)**-1 \***(X**T**y)** instead of **(X**T**y)**\* (**X**T**X)**-1**)** is because the **X**T**X** was on the left side of b. Whatever side the expression leaves from, is the same side it’s inverse is placed on.

**Solving for b**

Now, in order to find **b** we need to be able to calculate (**X**T**X)**-1 \***(X**T**y)**, which involves matrix multiplication and division (i.e., inverses). Let’s start with matrix multiplication: **X**T**X**. Below, you will see the matrix multiplication of **X**T by **X**. In the second equation, the matrices are lined up such that the right matrix, **X**, is above the left, **X**T. This makes more clear the mechanisms of matrix multiplication: you take the dot product of a row of the left (i.e. pre-multiplied) matrix times a column of the right (i.e., post-multiplied) matrix. The dot product is the sum of each component of the row multiplied by its associated component of the column. For example, the dot product of the first row of **X**T and the first column of **X** is simply 1\*1 + 1\*1 + 1\*1 + 1\*1 + 1\*1 + 1\*1 + 1\*1 + 1\*1 + 1\*1 + 1\*1 = 10. Therefore, 10 is the value of the upper-left element in the new matrix **X**T**X**. As a more complicated calculation, the dot product of the second row of **X**T and the third column of **X** is 6\*24 + 3\*45 + 7\*33 + 5\*31 + 2\*26 + 4\*44 + 7\*47 + 4\*38 + 4\*34 + 5\*29 = 1655. This is done for every combination of rows and columns.

\*

The interesting thing you might notice about the resulting **X**T**X** is that it’s symmetrical because the values in **X**T and **X** are the same. The final matrix **X**T**X** is then:

Now that we have **X**T**X** we can think about how to compute (**X**T**X**)-1. Division in matrices is the same concept of 1/ **X**T**X**. However, when dealing with matrixes, 1 is not just a number, but the identity matrix (**I**), which is a matrix with 1s on the diagonal and 0s on the off-diagonals. It has the “identity property” where a matrix times its identity matrix, equals the exact same matrix – similar to how any number times 1 equals that number. The associated identity matrix for our **X**T**X** matrix is below. Notice the number of rows and columns have to be the same (4x4):

There are multiple ways to find the inverse of a matrix, but I will present “Gauss-Jordan elimination” because it is the simplest to understand. It involves doing three “elementary” row operations to the matrix you seek to invert along with its identify matrix juxtaposed next to it. For **X**T**X**, this new matrix looks like so:

The three elementary row operations are 1) row substitutions and 2) row multiplication by a scalar and 3) row addition/subtraction from one another. Our goal is to convert the left half of the matrix (**X**T**X**) to the identity matrix (**I**). However, any elementary row operation we do to the left half of our matrix, we also have to do to the right half. Therefore, as we convert the left half of the matrix to the identify matrix, the right half of the matrix will change. Specifically, it will change to the inverse of our original matrix. By doing so we have found the matrix that will satisfy the equation **I** = **A**\***X**T**X,** where I is the identify matrix and A is the inverse of **X**T**X**. Essentially, we are using the matrix form of the scalar equation 1 = 1/X\*X.

Now, we will begin doing elementary row operations to convert the left half of the matrix (**X**T**X**) to the identity matrix (**I**). First, we will try to convert the 10 in the upper-left corner of the matrix into a 1, to correspond to the identity matrix. To do so, I am going to multiply the top row by 4.8 and then subtract the second row from it.

So now we have one of the 16 elements of the left half of the matrix converted to its identity matrix value. And while we did the row operations, the right of the half matrix begins to change from the identity matrix into the inverse matrix.

Now, I am not actually going to do all the elementary row operations needed to solve for the inverse. Here is where modern computing power has really changed applied statistics. Although having a computer add, subtract, multiply, and divide is convenient, having a computer solve for the inverse of matrices saves relatively much more time. There is no “formula” to find inverses, rather the computer has to do the elementary row operations as an algorithm. Now mathematicians and computer scientists have found ways to greatly decrease the time and iterations of the algorithms (remember there are other ways to find the inverse of a matrix other than Guass-Jordan elimination) so that when SPSS runs your linear regression, it finds the inverse in milliseconds. So here is the inverse matrix that my computer spit out:

To check we have the correct answer, we can do matrix multiplication of the original matrix **X**T**X** and inverse matrix (**X**T**X**)-1, which should equal the identity matrix. And indeed it does…

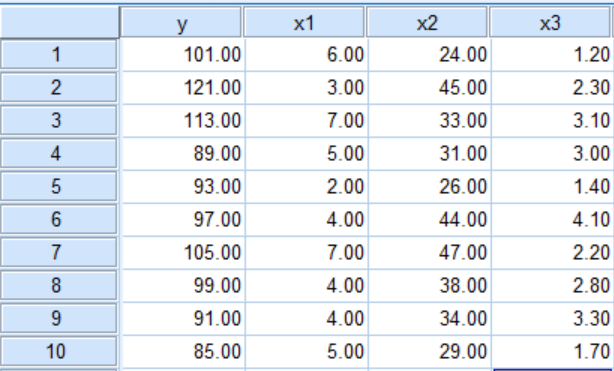
Now we have the first part of our expression, (**X**T**X)**-1, but we still need to calculate the second part - **(X**T**y)**. We follow the same procedure for matrix multiplication, although now we are multiplying a matrix by a vector, which simplifies the calculation a bit.

The matrix **X**T**y** is in bold. Now we can go ahead and do the final matrix multiplication, (**X**T**X)**-1\***(X**T**y)** to calculate the coefficient vector **b**. Notice, due to the dimensions of the two matrices being multiplied, the final solution with be only a vector, rather than another matrix.

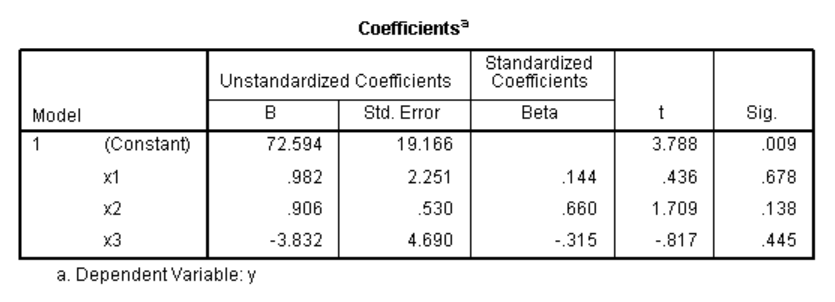
And there you have it folks! – the intercept and 3 partial regression slopes in bold: Intercept b0 = 72.59; partial regression slope for x1 (b1 = 0.98); partial regression slope for x2 (b2 = 0.91); partial regression slope for x3 (b3 = -3.83).

**Checking in SPSS**

While, you might have faith in my matrix algebra ability, I sure didn’t. I wanted to “check my work” like my middle school math teachers taught me. So let’s do that with SPSS. I created an SPSS dataset that is identical to the initial dataset above.



I asked SPSS to do a linear regression with x1, x2, and x3 as predictors with y as the outcome. Notice, we get the exact same regression coefficients:



Of course, we still need to calculate the standard errors, t-values, p-values, residuals, r-squared, etc.; however, I am sick of math at the moment. But now you know the math behind calculating the regression coefficients in a linear regression model!

It really is amazing how much computing power have changed the world of applied statistics. Things never possible before, such as bootstrapping and Gibbs sampling, now are. In addition, it makes linear regression quite a bit easier. I would much rather point and click in SPSS rather than calculate all of my regression coefficients by hand using matrix algebra. Nevertheless, doing it once or twice satisfies the curiosity of a first year graduate student ☺.