Introduction to the mathematics of mixed models

Brad Duthie

03/06/2020

Some quick background on matrix multiplication

Matrix algebra is extremely useful, but it can be confusing at first because it differs in some key ways from normal algebra. Say we have two matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix},$$

and

$$\mathbf{B} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}.$$

If we want to just multiply the two matrices ($\mathbf{A} \circ \mathbf{B}$), for example, using the familiar rules of scalar multiplication (i.e., get the Hadamard product), then we would do this for two matrices,

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \circ \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 5 & 12 \\ 21 & 32 \end{bmatrix}.$$

Matrix multiplication works different. To get **A** times **B**, we need to take the first element of the row of **A** and multiply it by the first element of the column of **B**, then add this to the second element of the row of **A** and multiply it by the second element of the column of **B**. See below for a clearer explanation of what this looks like in practice,

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} [(1 \times 5) + (2 \times 7)] & [(1 \times 6) + (2 \times 8)] \\ [(3 \times 5) + (4 \times 7)] & [(3 \times 6) + (4 \times 8)] \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}.$$

This might seem arbitrary or strange at first, but there are a lot of good reasons that matrix multiplication works this way. In statistics, one really cool property is that a matrix times its transpose (rows swapped for columns) is the sum of squares,

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = 1^2 + 2^2 + 3^2 + 4^2 = 30.$$

The reason for introducing this is just because we will need the matrix algebra for understanding the mixed model. I will keep this as simple as possible to give an idea of what is going on. Note that matrix addition works as you would expect,

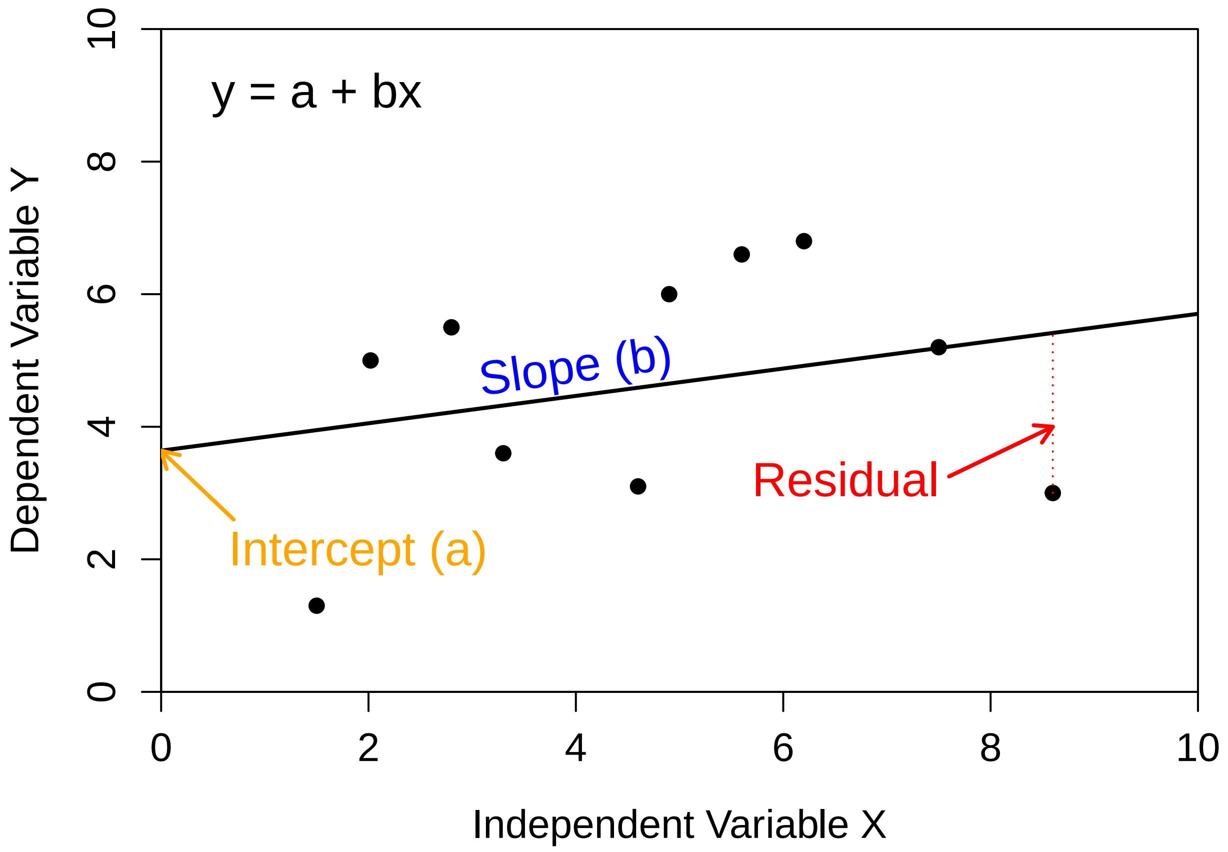
$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} + \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}6 & 8\\10 & 12\end{bmatrix}.$$

Just the general linear model first

To start off, let us consider the familiar linear model,

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i.$$

Note that in the above, we are predicting the value of our data point i (y_i) from i's value of x (x_i) with the intercept (β_0), slope (β_1), and error of data point i (ϵ_i). Note that ϵ_i is just the residual value for i; that is, its distance from the regression line (positive values mean that the data point is above the line, negative values mean that it is below).



We need to start out with a small hypothetical data set. We are not looking for something realistic to test a hypothesis; we only need something that will show what is going on mathematically, so we can consider the data below.

Y	X
9.98	5.78
9.55	2.57
11.39	6.58
11.79	5.69
10.62	6.43

Y	X
8.46	5.85

If we do a simple linear regression model, then we find an intercept of $\beta_0 = 8.6704261$ and a slope of $\beta_1 = 0.2968828$. Our six module residuals are -0.4064086, 0.1165852, 0.7660852, 1.4303109, 0.0406176, -1.9471904. We can put all of these into a table.

y_i	beta_0	beta_1	x_i	epsilon_i
9.98	8.670426	0.2968828	5.78	-0.4064086
9.55	8.670426	0.2968828	2.57	0.1165852
11.39	8.670426	0.2968828	6.58	0.7660852
11.79	8.670426	0.2968828	5.69	1.4303109
10.62	8.670426	0.2968828	6.43	0.0406176
8.46	8.670426	0.2968828	5.85	-1.9471904

I have written it the way I did above so that you can see how each of the values correspond to our general linear model above,

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i.$$

To get a the y value for $i = 1$ (y_1), we then just need to use the values in the first row,

$$y_i = 8.6704261 + 0.2968828(5.78) + -0.4064086.$$

You can confirm that the maths works here. Another way to write that general linear model is using matrix notation,

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{e}.$$

For the above example, this would look as follows,

$$\begin{bmatrix} 9.98 \\ 9.55 \\ 11.39 \\ 11.79 \\ 10.62 \\ 8.46 \end{bmatrix} = \begin{bmatrix} 1 & 5.78 \\ 1 & 2.57 \\ 1 & 6.58 \\ 1 & 5.69 \\ 1 & 6.43 \\ 1 & 5.85 \end{bmatrix} \begin{bmatrix} 8.6704261 \\ 0.2968828 \end{bmatrix} + \begin{bmatrix} -0.4064086 \\ 0.1165852 \\ 0.7660852 \\ 1.4303109 \\ 0.0406176 \\ -1.9471904 \end{bmatrix}.$$

We can do the multiplication of $\mathbf{X}\beta$ first,

$$\begin{bmatrix} 9.98 \\ 9.55 \\ 11.39 \\ 11.79 \\ 10.62 \\ 8.46 \end{bmatrix} = \begin{bmatrix} 10.3864086 \\ 9.4334148 \\ 10.6239148 \\ 10.3596891 \\ 10.5793824 \\ 10.4071904 \end{bmatrix} + \begin{bmatrix} -0.4064086 \\ 0.1165852 \\ 0.7660852 \\ 1.4303109 \\ 0.0406176 \\ -1.9471904 \end{bmatrix}.$$

Now we just need to add the two matrices on the right hand side together,

$$\begin{bmatrix} 9.98 \\ 9.55 \\ 11.39 \\ 11.79 \\ 10.62 \\ 8.46 \end{bmatrix} = \begin{bmatrix} 9.98 \\ 9.55 \\ 11.39 \\ 11.79 \\ 10.62 \\ 8.46 \end{bmatrix}.$$

Remember that we can have any number of β coefficients that we want (assuming that we are not overparameterising the model). Here I have used an example with only one continuous x variable (5.78, 2.57, 6.58, 5.69, 6.43, 5.85). We could just as well add a categorical variable x_2 that might distinguish two groups (e.g., 'species_1' versus 'species_2'), making x_2 a binary column that would be added to $\mathbf{X}$ with its own coefficient β_2,

$$\mathbf{X} = \begin{bmatrix} 1 & 5.78 & 0 \\ 1 & 2.57 & 0 \\ 1 & 6.58 & 0 \\ 1 & 5.69 & 1 \\ 1 & 6.43 & 1 \\ 1 & 5.85 & 1 \end{bmatrix}$$

Its coefficiencts would then, be $\beta = [\beta_0, \beta_1, \beta_2]$. The details are not so important, but I just want to emphasise that $\mathbf{X}$ (and $\mathbf{Y}$) can have any number of columns. Remember that typically all we have are the $\mathbf{X}$ and $\mathbf{Y}$ to start with, and we need to solve for the β values,

$$\begin{bmatrix} 9.98 \\ 9.55 \\ 11.39 \\ 11.79 \\ 10.62 \\ 8.46 \end{bmatrix} = \begin{bmatrix} 1 & 5.78 \\ 1 & 2.57 \\ 1 & 6.58 \\ 1 & 5.69 \\ 1 & 6.43 \\ 1 & 5.85 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{bmatrix}.$$

Note that $\mathbf{X}$ is sometimes called the 'design matrix'.

Now we can talk about the mixed model

The model discussed above all focused on **fixed effects**. Now we will add **random effects** to the above to build a **mixed model**. I am going to deliberately avoid defining 'fixed effect' and 'random effect' and instead just focus on what changes mathematically. We are going to add a new term **Zu** the above model (note that a lot of papers use **b** instead of **u**, but I want to avoid potential confusion with β). Our mixed model equation just looks like the below,

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{Z}\mathbf{u} + \mathbf{e}.$$

What we are doing is adding the product of group indicators ($\mathbf{Z}$) and their random effects ($\mathbf{u}$, their deviation u_i from the mean 0). We can read in the `nlme` R library and the data set 'dung', which looks like the below.

```
library(nlme);
dung <- read.csv("dung.csv");
print(dung);
```

```
##      pr_dung flies oven_block
## 1  0.1433747   223          1
## 2  0.1551917     0          1
## 3  0.1413354   100          1
## 4  0.1380987    97          1
## 5  0.1488016   130          1
## 6  0.1443227   169          1
## 7  0.1428871   183          1
## 8  0.1705165    77          2
## 9  0.1593878     0          2
## 10 0.1651354    51          2
## 11 0.1747078    49          2
## 12 0.1904174    81          2
## 13 0.1806727    45          3
## 14 0.1700109    98          3
## 15 0.1654107   168          3
## 16 0.1803931    22          3
## 17 0.1727390     4          3
## 18 0.1701061    98          3
```

This is a subset of data from a set of experiments that I ran in which containers (rows) of wet cow dung were placed out for a parent generation of dungflies to lay their eggs. The mass of the wet dung was recorded before the experiment for each container. Offspring that emerged from each dung container were counted (`flies`), and then the wet dung was placed in the oven to get the dry mass. The proportion of mass still remaining after drying was calculated (`pr_dung`), but due to experimental constraints (timing of experiments and the size of the oven), dung from containers needed to be placed in the oven in blocks (`oven_block`). Hence, the block in which the container went into the oven could affect `pr_dung`, so we want to include this as a random effect in the model. To summarise, the three columns of data above are as follows:

- **pr_dung**: Oven-dried mass of a dung container divided by its original wet mass
- **flies**: The number of offspring flies that emerged from a container
- **oven_block**: The group (i.e., 'block') in which the dung container was placed in the oven

We can place the known values from the data set above into our mixed model equation,

$$\begin{bmatrix}0.1433747\\0.1551917\\0.1413354\\0.1380987\\0.1488016\\0.1443227\\0.1428871\\0.1705165\\0.1593878\\0.1651354\\0.1747078\\0.1904174\\0.1806727\\0.1700109\\0.1654107\\0.1803931\\0.172739\\0.1701061\end{bmatrix} = \begin{bmatrix}1 & 223\\1 & 0\\1 & 100\\1 & 97\\1 & 130\\1 & 169\\1 & 183\\1 & 77\\1 & 0\\1 & 51\\1 & 49\\1 & 81\\1 & 45\\1 & 98\\1 & 168\\1 & 22\\1 & 4\\1 & 98\end{bmatrix} \begin{bmatrix}\beta_0\\\beta_1\end{bmatrix} + \begin{bmatrix}1 & 0 & 0\\1 & 0 & 0\\1 & 0 & 0\\1 & 0 & 0\\1 & 0 & 0\\1 & 0 & 0\\1 & 0 & 0\\0 & 1 & 0\\0 & 1 & 0\\0 & 1 & 0\\0 & 1 & 0\\0 & 1 & 0\\0 & 0 & 1\\0 & 0 & 1\\0 & 0 & 1\\0 & 0 & 1\\0 & 0 & 1\\0 & 0 & 1\end{bmatrix} \begin{bmatrix}u_1\\u_2\\u_3\end{bmatrix} + \begin{bmatrix}\epsilon_1\\\epsilon_2\\\epsilon_3\\\epsilon_4\\\epsilon_5\\\epsilon_6\\\epsilon_7\\\epsilon_8\\\epsilon_9\\\epsilon_{10}\\\epsilon_{11}\\\epsilon_{12}\\\epsilon_{13}\\\epsilon_{14}\\\epsilon_{15}\\\epsilon_{16}\\\epsilon_{17}\\\epsilon_{18}\end{bmatrix}.$$

Here is how we would run a mixed effects model in R.

```r
mod  <- lme(pr_dung ~ flies, random = ~ 1|oven_block, data = dung);
```

I do not want to focus on the R notation here, or the output, but we can grab the values of β and $\mathbf{u}$, and $\mathbf{e}$ to show how they relate to our equation. We can pull these out of `mod` and put them in matrix form as follows.

```r
beta <- as.matrix(fixed.effects(mod));
uu   <- as.matrix(random.effects(mod));
ee   <- as.matrix(residuals(mod));
```

Here are our `beta` (β) values.

```r
print(beta);
```

```
##                      [,1]
## (Intercept)  1.662317e-01
## flies       -3.466743e-05
```

Here are our `uu` ($\mathbf{u}$) values.

```r
print(uu);
```

```
##    (Intercept)
## 1 -0.016207919
## 2  0.007158668
## 3  0.009049251
```

Note that the $\mathbf{u}$ values sum to zero. Each u_i is a deviation from the mean of u, where u is normally distributed around zero. Here are our `ee` (ϵ) values (residuals).

```r
print(ee);
```

```
##            [,1]
## 1  0.0010817597
## 1  0.0051679006
## 1 -0.0052216740
## 1 -0.0085623576
## 1  0.0032845742
## 1  0.0001577298
## 1 -0.0007925669
## 2 -0.0002044497
## 2 -0.0140025103
## 2 -0.0064868708
## 2  0.0030161277
## 2  0.0198351347
## 3  0.0069518305
## 3 -0.0018725961
## 3 -0.0040461441
## 3  0.0058748860
## 3 -0.0024033180
## 3 -0.0017774556
```

Now we can put these values back into the equation above and confirm that the right side of the equation equals the left side,

$$\begin{bmatrix}0.1433747\\0.1551917\\0.1413354\\0.1380987\\0.1488016\\0.1443227\\0.1428871\\0.1705165\\0.1593878\\0.1651354\\0.1747078\\0.1904174\\0.1806727\\0.1700109\\0.1654107\\0.1803931\\0.172739\\0.1701061\end{bmatrix} = \begin{bmatrix}1 & 223\\1 & 0\\1 & 100\\1 & 97\\1 & 130\\1 & 169\\1 & 183\\1 & 77\\1 & 0\\1 & 51\\1 & 49\\1 & 81\\1 & 45\\1 & 98\\1 & 168\\1 & 22\\1 & 4\\1 & 98\end{bmatrix}\begin{bmatrix}0.1662317\\-0.00003466743\end{bmatrix} + \begin{bmatrix}1&0&0\\1&0&0\\1&0&0\\1&0&0\\1&0&0\\1&0&0\\1&0&0\\0&1&0\\0&1&0\\0&1&0\\0&1&0\\0&1&0\\0&0&1\\0&0&1\\0&0&1\\0&0&1\\0&0&1\\0&0&1\end{bmatrix}\begin{bmatrix}-0.016207919\\0.007158668\\0.009049251\end{bmatrix} + \begin{bmatrix}0.0010818\\0.0051679\\-0.0052217\\-0.0085624\\0.0032846\\1.5772976\times10^{-4}\\-7.9256689\times10^{-4}\\-2.0444972\times10^{-4}\\-0.0140025\\-0.0064869\\0.0030161\\0.0198351\\0.0069518\\-0.0018726\\-0.0040461\\0.0058749\\-0.0024033\\-0.0017775\end{bmatrix}.$$

We can do some of the calculations on the right hand side to simplify,

$$\begin{bmatrix}0.1433747\\0.1551917\\0.1413354\\0.1380987\\0.1488016\\0.1443227\\0.1428871\\0.1705165\\0.1593878\\0.1651354\\0.1747078\\0.1904174\\0.1806727\\0.1700109\\0.1654107\\0.1803931\\0.172739\\0.1701061\end{bmatrix} = \begin{bmatrix}0.1585009\\0.1662317\\0.1627649\\0.162869\\0.1617249\\0.1603729\\0.1598876\\0.1635623\\0.1662317\\0.1644637\\0.164533\\0.1634236\\0.1646717\\0.1628343\\0.1604076\\0.165469\\0.166093\\0.1628343\end{bmatrix} + \begin{bmatrix}-0.0162079\\-0.0162079\\-0.0162079\\-0.0162079\\-0.0162079\\-0.0162079\\-0.0162079\\0.0071587\\0.0071587\\0.0071587\\0.0071587\\0.0071587\\0.0090493\\0.0090493\\0.0090493\\0.0090493\\0.0090493\\0.0090493\end{bmatrix} + \begin{bmatrix}0.0010818\\0.0051679\\-0.0052217\\-0.0085624\\0.0032846\\1.5772976\times10^{-4}\\-7.9256689\times10^{-4}\\-2.0444972\times10^{-4}\\-0.0140025\\-0.0064869\\0.0030161\\0.0198351\\0.0069518\\-0.0018726\\-0.0040461\\0.0058749\\-0.0024033\\-0.0017775\end{bmatrix}.$$

The way that we could do this in R is as follows, first setting up **X** and **Z**.

```
X    <- cbind(rep(x = 1, times = dim(dung)[1]), dung$flies);
Z    <- matrix(data = 0, nrow = dim(dung)[1], ncol = dim(uu)[1]);
for(i in 1:dim(Z)[1]){
  Z[i, dung$oven_block[i]] <- 1;
}
```

The for loop is just a concise way of getting Z in the form we need. Now `X` looks like the below.

```
##        [,1] [,2]
##   [1,]    1  223
##   [2,]    1    0
##  [3,]    1  100
##  [4,]    1   97
##  [5,]    1  130
##  [6,]    1  169
##  [7,]    1  183
##  [8,]    1   77
##  [9,]    1    0
## [10,]    1   51
## [11,]    1   49
## [12,]    1   81
## [13,]    1   45
## [14,]    1   98
## [15,]    1  168
## [16,]    1   22
## [17,]    1    4
## [18,]    1   98
```

This is what `Z` looks like.

```
##       [,1] [,2] [,3]
##  [1,]    1    0    0
##  [2,]    1    0    0
##  [3,]    1    0    0
##  [4,]    1    0    0
##  [5,]    1    0    0
##  [6,]    1    0    0
##  [7,]    1    0    0
##  [8,]    0    1    0
##  [9,]    0    1    0
## [10,]    0    1    0
## [11,]    0    1    0
## [12,]    0    1    0
## [13,]    0    0    1
## [14,]    0    0    1
## [15,]    0    0    1
## [16,]    0    0    1
## [17,]    0    0    1
## [18,]    0    0    1
```

We can do the matrix multiplication to get back `pr_dung` using the following.

```r
Y <- X %*% beta + Z %*% uu + ee;
print(cbind(Y, dung$pr_dung));
```

```
##       (Intercept)
##  [1,]   0.1433747 0.1433747
##  [2,]   0.1551917 0.1551917
##  [3,]   0.1413354 0.1413354
##  [4,]   0.1380987 0.1380987
##  [5,]   0.1488016 0.1488016
##  [6,]   0.1443227 0.1443227
##  [7,]   0.1428871 0.1428871
##  [8,]   0.1705165 0.1705165
##  [9,]   0.1593878 0.1593878
## [10,]   0.1651354 0.1651354
## [11,]   0.1747078 0.1747078
## [12,]   0.1904174 0.1904174
## [13,]   0.1806727 0.1806727
## [14,]   0.1700109 0.1700109
## [15,]   0.1654107 0.1654107
## [16,]   0.1803931 0.1803931
## [17,]   0.1727390 0.1727390
## [18,]   0.1701061 0.1701061
```

Models can of course get a lot more complex than what we just did, but this should give you a general idea of what is going on in R when you run a mixed model.