Regression Using Matrices

Andrew Zieffler

We will use the following toy data set to illustrate how regression is carried out via matrix algebra.

```
> myData = data.frame(
    wage   = c(12, 8, 16.26, 13.65, 8.5, 8.5),
    age    = c(32, 33, 32, 33, 26, 28),
    sex    = c("M", "F", "M", "M", "M", "M"),
    collar = c("Pink", "Blue", "Pink", "White", "White", "Blue")
  )

> myData



   wage age sex collar
1 12.00  32   M   Pink
2  8.00  33   F   Blue
3 16.26  32   M   Pink
4 13.65  33   M  White
5  8.50  26   M  White
6  8.50  28   M   Blue
```

In practice, we use built-in R functions to fit a linear regression model; for example, `lm()`, `summary()`, `anova()`, `vcov()`, `fitted()`, and `residuals()`.

```
> lm.1 = lm(wage ~ 1 + age, data = myData)

> summary(lm.1)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -7.1516    15.0840  -0.474     0.66
age           0.5968     0.4900   1.218     0.29

Residual standard error: 3.226 on 4 degrees of freedom
Multiple R-squared:  0.2706,    Adjusted R-squared:  0.08821
F-statistic: 1.484 on 1 and 4 DF,  p-value: 0.2901

> anova(lm.1)
Analysis of Variance Table

Response: wage
          Df Sum Sq Mean Sq F value Pr(>F)
age        1 15.436  15.436  1.4837 0.2901
Residuals  4 41.616  10.404
```

```
> vcov(lm.1)
            (Intercept)        age
(Intercept)  227.525576 -7.3627693
age           -7.362769  0.2400903

> fitted(lm.1)
        1         2         3         4         5         6 
11.947462 12.544308 11.947462 12.544308  8.366385  9.560077 

> residuals(lm.1)
          1           2           3           4           5           6 
 0.05253846 -4.54430769  4.31253846  1.10569231  0.13361538 -1.06007692 
```

- `lm()` — fits linear model
- `summary()` — outputs information from fitted model
- `anova()` — outputs decomposition of variation
- `vcov()` — outputs variance–covariance matrix of the fitted model
- `fitted()` — outputs fitted values (y-hats)
- `residuals()` — outputs residuals

These same computations can be produced using matrix algebra; which underlies R's built-in function. By understanding this, you can begin to see where computational problems will arise. Begin with the equation for the simple linear regression model.

$$Y_i = \beta_0 + \beta_1(X_i) + \epsilon_i \qquad \text{where } i = 1, 2, \ldots, n$$

We gives an equation for each individual *i*.

$$\begin{aligned} Y_1 &= \beta_0(1) + \beta_1(X_1) + \epsilon_1 \\ Y_2 &= \beta_0(1) + \beta_1(X_2) + \epsilon_2 \\ Y_3 &= \beta_0(1) + \beta_1(X_3) + \epsilon_3 \\ &\vdots \\ Y_n &= \beta_0(1) + \beta_1(X_n) + \epsilon_n \end{aligned}$$

$$
\begin{aligned}
Y_1 &= \beta_0(1) + \beta_1(X_1) + \epsilon_1 \\
Y_2 &= \beta_0(1) + \beta_1(X_2) + \epsilon_2 \\
Y_3 &= \beta_0(1) + \beta_1(X_3) + \epsilon_3 \\
&\vdots \\
Y_n &= \beta_0(1) + \beta_1(X_n) + \epsilon_n
\end{aligned}
$$

In these equations, we can see that the Y-values, the X-values, and the residuals **VARY** by individual. As such, we say these values are **RANDOM.**

The regression coefficient are **FIXED**. They are the same for every individual.

The **FIXED** regression coefficients give us the same fitted/predicted value of Y when individuals have the same X-value.

Although the residuals are allowed to **VARY**, the regression model puts distributional assumptions on how they are allowed to vary as a whole (if we want to do inference)…at each X-value, the residuals are normally distributed, have a mean of zero, and have constant variance.

$$\begin{aligned} Y_1 &= \beta_0(1) + \beta_1(X_1) + \epsilon_1 \\ Y_2 &= \beta_0(1) + \beta_1(X_2) + \epsilon_2 \\ Y_3 &= \beta_0(1) + \beta_1(X_3) + \epsilon_3 \\ &\vdots \\ Y_n &= \beta_0(1) + \beta_1(X_n) + \epsilon_n \end{aligned}$$

We can now arrange parts of these equations into vectors and matrices.

$$\mathbf{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_n \end{pmatrix} \qquad \mathbf{X} = \begin{pmatrix} 1 & X_1 \\ 1 & X_2 \\ 1 & X_3 \\ \vdots & \\ 1 & X_n \end{pmatrix} \qquad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \qquad \boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

The model can then be expressed very compactly using matrix algebra.

$$\underset{n\times 1}{\mathbf{Y}} = \underset{n\times 2}{\mathbf{X}}\ \underset{2\times 1}{\boldsymbol{\beta}} + \underset{n\times 1}{\boldsymbol{\epsilon}}$$

We can also express the residual assumptions using matrix algebra.

The mean error is zero. We typically write this as an expectation:

$$E(\epsilon_i) = 0$$

Since each observation is an instantiation of the larger population, we say there is an expected value for every case.

$$\underset{n\times 1}{\begin{pmatrix} E(\epsilon_1) \\ E(\epsilon_2) \\ E(\epsilon_3) \\ \vdots \\ E(\epsilon_n) \end{pmatrix}} = \underset{n\times 1}{\begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}$$

We also make the assumption that the errors have constant variance (σ_ε^2) and that they are independent of one another. Independence indicates that the covariance (or correlation) between any two residuals is zero:

$\sigma(\varepsilon_i, \varepsilon_j) = 0$ for all $i \neq j$.

The variance–covariance matrix of the error terms can therefore be expressed as

$$\boldsymbol{\sigma^2}\{\boldsymbol{\epsilon}\} = \underset{n\times n}{\begin{pmatrix} \sigma_\epsilon^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_\epsilon^2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_\epsilon^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_\epsilon^2 \end{pmatrix}}$$

This is a diagonal matrix which can be expressed as the product of a scalar, σ_ε^2, and the Identity matrix.

$$= \sigma_\epsilon^2 \underset{n\times n}{\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}} = \sigma_\epsilon^2 \underset{n\times n}{\mathbf{I}}$$

We can express the entire regression model, including assumptions as:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where ε is a vector of independent random variables with

$$\mathbf{E}\{\boldsymbol{\varepsilon}\} = \mathbf{0}$$

and

$$\boldsymbol{\sigma}^2\{\boldsymbol{\varepsilon}\} = (\sigma^2{}_{\varepsilon})\mathbf{I}$$

Great. We can compactly write the regression equation. How does matrix algebra help us with actually doing regression???

First, it helps us find the OLS coefficient estimates.

Re-expressing the equation, we get:

$$\boldsymbol{\varepsilon} = \mathbf{Y} - \mathbf{X}\boldsymbol{\beta}$$

The method of least squares minimizes the sum of squared errors. We can express the SSE as the product between the transpose of $\boldsymbol{\varepsilon}$ and $\boldsymbol{\varepsilon}$.

$$\boldsymbol{\epsilon}'\boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_1 & \epsilon_2 & \epsilon_3 & \dots & \epsilon_n \end{pmatrix} \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_n \end{pmatrix} = \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \dots + \epsilon_n^2$$

However, rather than using $\boldsymbol{\varepsilon}$, we will use $\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}$

$$\boldsymbol{\epsilon}'\boldsymbol{\epsilon} = (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})$$

We can expand this via the rules of matrix algebra.

$$\boldsymbol{\epsilon}'\boldsymbol{\epsilon} = (\boldsymbol{Y} - \boldsymbol{X\beta})'(\boldsymbol{Y} - \boldsymbol{X\beta})$$

$$= \boldsymbol{Y'Y} - \boldsymbol{\beta' X' Y} - \boldsymbol{Y' X\beta} + \boldsymbol{\beta' X' X\beta}$$

Each term is a 1x1 matrix (check this!), which means that each term is equal to its transpose. We will re-write the third term **Y'X β** as its transpose **β'X'Y**.

$$= \boldsymbol{Y'Y} - \boldsymbol{\beta' X' Y} - \boldsymbol{\beta' X' Y} + \boldsymbol{\beta' X' X\beta}$$

Now we can combine the two middle terms.

$$= \boldsymbol{Y'Y} - 2\boldsymbol{\beta' X' Y} + \boldsymbol{\beta' X' X\beta}$$

Now we have expressed the SSE as a function of the coefficients, Xs and Ys. The goal in OLS is to find the coefficient values that MINIMIZE the SSE. To do so, we need to differentiate the equation with respect to β$_0$ and β$_1$ (**β**); set the derivative equal to zero, and solve for the coefficients.

$$\frac{\partial}{\partial \boldsymbol{\beta}}(\boldsymbol{Y}'\boldsymbol{Y} - 2\boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{Y} + \boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta})$$

Differentiating and setting this equal to 0 we get

$$0 = -2\boldsymbol{X}'\boldsymbol{Y} + 2\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta}$$

Dividing everything by 2 and adding **X'Y** to both sides, we get

$$\boldsymbol{X}'\boldsymbol{Y} = \boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta}$$

To isolate **β**, we pre-multiply both sides of the equation by **(X'X)**$^{-1}$

$$(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta}$$

$$(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y} = \boldsymbol{I}\boldsymbol{\beta}$$

$$\boxed{\boldsymbol{\beta} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}}$$

$$\boldsymbol{\beta} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

The vector of regression coefficients can be obtained directly through manipulation of the design matrix (**X**) and the vector of the outcomes (**Y**).

One crucial point is that the product of **X'X** must have an inverse. To check this, recall we compute the determinant and make sure that it is not zero.

Let's set up the outcome vector (**Y**) and the design matrix (**X**) for our toy data set based on regressing income on age.

```
# Outcome vector
> Y = myData$wage
> Y
[1] 12.00  8.00 16.26 13.65  8.50  8.50

# Design matrix
> X = matrix(c(rep(1, 6), myData$age), ncol = 2)
> X
     [,1] [,2]
[1,]    1   32
[2,]    1   33
[3,]    1   32
[4,]    1   33
[5,]    1   26
[6,]    1   28
```

We can check to see if the determinant of **X'X** is 0.

```
> det( t(X) %*% X)
[1] 260
```

Since the determinant is not 0, the matrix is invertible.

Now we will apply the following matrix algebra to obtain the coefficients.

$$\boldsymbol{\beta} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

```
> b = solve(t(X) %*% X) %*% t(X) %*% Y
> b
           [,1]
[1,] -7.1516154
[2,]  0.5968462
```

Note that these values are equivalent to the coefficient output obtained using the `summary()` function on the `lm` object.

Note: When we estimate the $\boldsymbol{\beta}$ matrix from sample data we refer to it as **b**.

Once we have the vector of coefficients, we can use those, and the design matrix to obtain fitted/predicted values.

$$\hat{\boldsymbol{Y}} = \boldsymbol{X}\boldsymbol{b}$$

```
> fitted_values = X %*% b
> fitted_values
          [,1]
[1,] 11.947462
[2,] 12.544308
[3,] 11.947462
[4,] 12.544308
[5,]  8.366385
[6,]  9.560077
```

We can also compute residuals.

$$\boldsymbol{e} = \hat{\boldsymbol{Y}} - \boldsymbol{X}\boldsymbol{b}$$

```
> residuals = Y - fitted_values
> residuals
            [,1]
[1,]  0.05253846
[2,] -4.54430769
[3,]  4.31253846
[4,]  1.10569231
[5,]  0.13361538
[6,] -1.06007692
```

We can also get uncertainty estimates (SEs) for the coefficients. In order to do this, we first need to compute the Mean Squared Error (MSE) based on the residuals.

Recall that a *Mean Square* (or variance estimate) is the ratio of the sum of squares to its degrees of freedom. Thus the MSE is the ratio of the SS_{Error} to df_{Error}.

$$\text{MSE} = \frac{\text{SS}_{\text{Error}}}{df_{\text{Error}}} = \frac{\sum \epsilon_i^2}{n - c}$$

where

$$\sum \epsilon_i^2 = \boldsymbol{\epsilon}'\boldsymbol{\epsilon}$$

n =sample size, and c = number of coefficients being estimated in the regression.

To compute MSE

```
# Compute SS for the residuals
> SSE = t(residuals) %*% residuals
> SSE
         [,1]
[1,] 41.61565


# Compute df for the residuals
> n = 6
> c = 2


# Compute MSE
> MSE = SSE / (n - c)
> MSE
         [,1]
[1,] 10.40391
```

The SEs for the coefficients represent the uncertainty in the estimates. This uncertainty is correlated between the coefficients (unless you have a balanced design). Most regression analyses do not have a balanced design, unless random assignment was used to assign predictor values.

We represent the uncertainty and the correlation in the estimated variance–covariance matrix of the coefficients,

$$(\text{MSE})\boldsymbol{V_\beta}$$

where

$$\boldsymbol{V_\beta} = (\boldsymbol{X}'\boldsymbol{X})^{-1}$$

```
> V_b = solve(t(X) %*% X)

# Compute variance--covariance matrix
> var_cov = as.numeric(MSE) * V_b
> var_cov

           [,1]       [,2]
[1,] 227.525576 -7.3627693
[2,]  -7.362769  0.2400903
```

This is the same output as the vcov() function produces.

$$\begin{pmatrix} 227.5 & -7.36 \\ -7.36 & 0.24 \end{pmatrix}$$

The variances for B₀ and B₁, respectively, are on the main diagonal. The SEs can be computed by taking the square roots of these elements.

```
> sqrt(diag(var_cov))

[1] 15.0839509  0.4899901
```

The covariance between B_0 and B_1 is the off-diagonal element. It is often easier to interpret correlations rather than covariances. To convert the variance–covariance matrix to a correlation matrix, use the `cov2cor()` function.

```
> cov2cor(var_cov)

           [,1]       [,2]
[1,]  1.0000000 -0.9961822
[2,] -0.9961822  1.0000000
```

Here we can see that the two coefficients are highly correlated.

We can use the `model.matrix()` function on an lm object to obtain the design matrix directly.

```
> model.matrix(lm.1)
  (Intercept) age
1           1  32
2           1  33
3           1  32
4           1  33
5           1  26
6           1  28
attr(,"assign")
[1] 0 1
```

As mentioned previously, when the matrix **X'X** is not invertible, none of these calculations can be performed. This happens in practice when (1) the determinant is zero because the design matrix's columns are not linearly independent, and (2) when the determinant is very close to zero because the model is over-specified (too many predictors for the sample size).

Consider the model that includes age and sex as main-effects and the interaction between age and sex to predict wage.

```
> lm.2 = lm(wage ~ 1 + age + sex + age:sex, data = myData)

> Y = myData$wage
> X = model.matrix(lm.2)
> X
  (Intercept) age sexM age:sexM
1           1  32    1       32
2           1  33    0        0
3           1  32    1       32
4           1  33    1       33
5           1  26    1       26
6           1  28    1       28
attr(,"assign")
[1] 0 1 2 3
attr(,"contrasts")
attr(,"contrasts")$sex
[1] "contr.treatment"
```

Note the design matrix includes a column for the intercept, both main-effects, and the interaction (product).

Let's try to compute the coefficients.

$$\boldsymbol{\beta} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

```
> det(t(X) %*% X)
[1] 1.833627e-10
```

The determinant is very small…near zero…could be problematic.

```
> b = solve(t(X) %*% X) %*% t(X) %*% Y

Error in solve.default(t(X) %*% X) :
  system is computationally singular: reciprocal condition number = 4.18242e-20
```

The computation produces an error.

Let's try to compute the coefficients.

$$\boldsymbol{\beta} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

```
> det(t(X) %*% X)
[1] 1.833627e-10
```

The determinant is very small…near zero…could be problematic.

```
> b = solve(t(X) %*% X) %*% t(X) %*% Y

Error in solve.default(t(X) %*% X) :
  system is computationally singular: reciprocal condition number = 4.18242e-20
```

The computation produces an error. This is because the model is over-specified for the sample size. In reality we are trying to estimate five parameters: 4 coefficients, and the MSE using $n = 6$ observations!

Using summary() on the fitted lm object would produce the same issue.

```
> summary(lm.2)

Call:
lm(formula = wage ~ 1 + age + sex + age:sex, data = myData)

Residuals:
         1          2          3          4          5          6 
-1.479e+00 -9.992e-16  2.781e+00 -7.713e-01  6.770e-01 -1.208e+00 

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)  
(Intercept) -23.1061    11.2609  -2.052   0.1325  
age           0.9426     0.3356   2.809   0.0674 .
sexM          6.4213     2.4202   2.653   0.0768 .
age:sexM          NA         NA      NA       NA  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.036 on 3 degrees of freedom
Multiple R-squared:  0.782,	Adjusted R-squared:  0.6367 
F-statistic: 5.382 on 2 and 3 DF,  p-value: 0.1018
```

Singularity is the fancy word mathematicians use when the determinant is zero (or very near zero).

The other time we get singularities is when the design matrix's columns are not linearly independent. Linear independence loosely means that we cannot define any column using a linear equation of the other columns.

Consider using the categorical job predictor collar (with levels: pink, white, and blue) to predict wage. To dummy-code this predictor, we create three new predictors—pink, white, and blue—that indicate whether or not the person is a pink-collar worker, a white-collar worker, or a blue-collar worker, respectively.

Collar	Pink	White	Blue
Pink	1	0	0
Blue	0	0	1
Pink	1	0	0
White	0	1	0
White	0	1	0
Blue	0	0	1

To fit the model we include the intercept and any two (not all three) dummy-coded predictors. The design matrix if we included pink and white would be:

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

What if we would have included all three dummy-coded predictors? Then the design matrix would be:

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

```
> X = matrix(
 data = c(1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1),
 byrow = TRUE,
 ncol = 4
 )

> X
     [,1] [,2] [,3] [,4]
[1,]    1    1    0    0
[2,]    1    0    0    1
[3,]    1    1    0    0
[4,]    1    0    1    0
[5,]    1    0    1    0
[6,]    1    0    0    1
```

Compute the determinant of **X'X**.

```
> det(t(X) %*% X)
[1] 5.329071e-15
```

Again, this is very near zero.

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Notice that the intercept column (call it X_1) can be written as a linear combination of the other three columns:

$$X_1 = 1(X_2) + 1(X_3) + 1(X_4)$$

This will lead to a singularity. To fix this, we need to eliminate one of the columns.

Technically, the problem is that the model is again over-specified.

Eliminating the blue-collar column:

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

```
> X = matrix(
  data = c(1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0),
  byrow = TRUE,
  ncol = 3
  )
> X
     [,1] [,2] [,3]
[1,]    1    1    0
[2,]    1    0    0
[3,]    1    1    0
[4,]    1    0    1
[5,]    1    0    1
[6,]    1    0    0

> det(t(X) %*% X)
[1] 8
```

Now the determinant is further from 0, and we can invert the **X'X** matrix.

References and Source Material

Additional Resources

- Prasad, A. (2012). *Geometric definition of determinants*. YouTube video. https://www.youtube.com/watch?v=xX7qBVa9cQU&feature=youtu.be