A Simple Example: Using lm() in R

Ishan Paul,bs2033

August 2022

In this section, we show how to use the lm() function, in R, to find "solutions" of approximate linear systems. We consider the following example, introduced in section 2.

$$\begin{bmatrix} 3 & 4 \\ 4 & 1 \\ 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} A \\ B \end{bmatrix} \approx \begin{bmatrix} 9.8 \\ 9.1 \\ 7.0 \end{bmatrix}$$

Let us denote

$$X = \begin{bmatrix} 3 & 4 \\ 4 & 1 \\ 2 & 3 \end{bmatrix}$$

$$\boldsymbol{\beta} = \begin{bmatrix} A \\ B \end{bmatrix}$$

$$\boldsymbol{y} = \begin{bmatrix} 9.8 \\ 9.1 \\ 7.0 \end{bmatrix}$$

Then the equation can be written as

$$X\boldsymbol{\beta} \approx \boldsymbol{y}$$

We already know that the system of linear equations given by $X\boldsymbol{\beta} = \boldsymbol{y}$ is inconsistent, i.e. it has no solution for $\boldsymbol{\beta}$. So we instead find $\hat{\boldsymbol{\beta}} \in \mathbb{R}^2$ such that $\|X\hat{\boldsymbol{\beta}} - \boldsymbol{y}\|$ is minimum. Here $\|.\|$ is the Euclidian norm in $\mathbb{R}^3$. We have seen in section 4 that such a $\hat{\boldsymbol{\beta}}$ is a solution of the corresponding normal system of equations, $X'X\boldsymbol{\beta} = X'\boldsymbol{y}$. We have stated the result, without proving that the normal system of equations is always consistent, i.e., there is always a $\hat{\boldsymbol{\beta}}$ satisfying the system of equations. In this case, columns of X are linearly independent. So, $X'X$ is non-singular and hence, $\hat{\boldsymbol{\beta}}$ is unique.

Now we show how to use lm() function in R to find the required $\hat{\boldsymbol{\beta}}$. We use RStudio as IDE with R version 4.1.1 . So, let's open R console and start coding!

Note that lm() function and other functions, we will use, are present in the R base packages. So, we do not need to load any additional package.

1. First, we create the design matrix and store it in variable X. To do so we use the "matrix()" function.

```
> X <- matrix(c(3,4,2,4,1,3), nrow = 3, ncol = 2)
> X
     [,1] [,2]
[1,]    3    4
[2,]    4    1
[3,]    2    3
>
```

   The function "matrix(body, nrow, ncol, byrow = FALSE)" creates a matrix of dimension $nrow \times ncol$ using elements of the vector, "body", arranged by column major. In this case, we have body $= (3, 4, 2, 4, 1, 3)$. The first three elements, 3,4,2, are assigned to column 1 in order and next three, 4,1,3, to column 2 in order.

2. Now we store the observed weights in the vector y.

```
> y <- c(9.8, 9.1, 7.0)
> y
[1] 9.8 9.1 7.0
>
```

   the "c()" function in R stands for combine and is used to make a vector of objects of the same type. Here it makes a vector of numeric type.

3. Now we use the "lm()" function to make a linear model using the data.

```
> fit <- lm(formula = y~X-1)
> fit

Call:
lm(formula = y ~ X - 1)

Coefficients:
    X1      X2
2.0378  0.9411

>
```

   The lm() function takes a formula as argument. Here, the formula is $y \sim X - 1$. The $\sim$ stands for approximate equality. The placement of $\boldsymbol{y}$ and X around $\sim$ may seem unusual. It may serve to show

how the matrix X describes $\boldsymbol{y}$ as a linear function. The values of the coefficients, X1 and X2 are such that $(X1, X2)$ is a solution of the equation, $X'X\boldsymbol{\beta} = X'\boldsymbol{y}$, i.e.,

$$X'X. \begin{bmatrix} X1 \\ X2 \end{bmatrix} = X'\boldsymbol{y}$$

Now, we finally address the -1 in the formula. Usually it would mean an operation resulting in a new matrix with same dimensions as X, whose every element is one less than the corresponding element in X. However, in this case, it has a completely different meaning.

Before proceeding further,we observe using the command, "model.matrix()" that the design matrix in the linear model, "fit", described by the formula $y \sim X - 1$ is

$$X = \begin{bmatrix} 3 & 4 \\ 4 & 1 \\ 2 & 3 \end{bmatrix}$$

```
> model.matrix(fit)
  X1 X2
1  3  4
2  4  1
3  2  3
attr(,"assign")
[1] 1 1
>
```

This is exactly what we wanted.

Now if we make another linear model "fit2", with the formula $y \sim X$

```
> fit2 = lm(formula = y ~ X)
> fit2

Call:
lm(formula = y ~ X)

Coefficients:
(Intercept)           X1           X2
      0.525        1.925        0.875

> model.matrix(fit2)
  (Intercept) X1 X2
1           1  3  4
2           1  4  1
3           1  2  3
attr(,"assign")
[1] 0 1 1
>
```

Observe that the values of X1, X2 are different and there is an additional term called intercept. However the most surprising observation is that the design matrix is not what we wanted. There is an additional column of ones, appended to the left of the matrix, X. At this point, we reveal that the "lm()" function appends an additional column of ones to the left of the argument X, unless specified otherwise. The "-1" at the end of the formula does precisely that. It stops "lm()" from appending the extra column to X. Although this behaviour may seem to be quite vexing, it may prove to be helpful as we progress.

We have learnt how to use lm() to create linear models in R. However, to use it properly, we must keep in mind that arithmetic operators do not have their usual meaning, when used in "formulae". Later in the book, we will learn the syntax of these "formulae" in greater detail. For now, we must be content with knowing that the lm() function can be used to solve the least square problem, as shown, when the measurement vector and design matrix are known.