Expressing design formula in R

The Design Matrix

The Design Matrix

Here we will show how to use the two R functions, `formula` and `model.matrix`, in order to produce *design matrices* (also known as *model matrices*) for a variety of linear models. For example, in the mouse diet examples we wrote the model as

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, i = 1, \dots, N$$

with Y_i the weights and x_i equal to 1 only when mouse i receives the high fat diet. We use the term *experimental unit* to N different entities from which we obtain a measurement. In this case, the mice are the experimental units.

This is the type of variable we will focus on in this chapter. We call them *indicator variables* since they simply indicate if the experimental unit had a certain characteristic or not. As we described earlier, we can use linear algebra to represent this model:

$$\mathbf{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{pmatrix}, \mathbf{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_N \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \text{ and } \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{pmatrix}$$

as:

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_N \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{pmatrix}$$

Choice of design

The choice of design matrix is a critical step in linear modeling since it encodes which coefficients will be fit in the model, as well as the inter-relationship between the samples. A common misunderstanding is that the choice of design follows straightforward from a description of which samples were included in the experiment. This is not the case. The basic information about each sample (whether control or treatment group, experimental batch, etc.) does not imply a single 'correct' design matrix. The design matrix additionally encodes various assumptions about how the variables in $\mathbf{X}$ explain the observed values in $\mathbf{Y}$, on which the investigator must decide.

For the examples we cover here, we use linear models to make comparisons between different groups. Hence, the design matrices that we ultimately work with will have at least two columns: an *intercept* column, which consists of a column of 1's, and a second column, which specifies which samples are in a second group. In this case, two coefficients are fit in the linear model: the intercept, which represents the population average of the first group, and a second coefficient, which represents the difference between the population averages of the second group and the first group. The latter is typically the coefficient we are interested in when we are performing statistical tests: we want to know if there is a difference between the two groups.

We encode this experimental design in R with two pieces. We start with a formula with the tilde symbol `~`. This means that we want to model the observations using the variables to the right of the tilde. Then we put the name of a variable, which tells us which samples are in which group.

Let's try an example. Suppose we have two groups, control and high fat diet, with two samples each. For illustrative purposes, we will code these with 1 and 2 respectively. We should first tell R that these values should not be interpreted numerically, but as different levels of a *factor*. We can then use the paradigm `~ group` to, say, model on

More groups

Using the same formula, we can accommodate modeling more groups. Suppose we have a third diet:

```
group <- factor(c(1,1,2,2,3,3))
model.matrix(~ group)
```

```
##   (Intercept) group2 group3
## 1           1      0      0
## 2           1      0      0
## 3           1      1      0
## 4           1      1      0
## 5           1      0      1
## 6           1      0      1
## attr(,"assign")
## [1] 0 1 1
## attr(,"contrasts")
## attr(,"contrasts")$group
## [1] "contr.treatment"
```

Now we have a third column which specifies which samples belong to the third group.

An alternate formulation of design matrix is possible by specifying `+ 0` in the formula:

```
group <- factor(c(1,1,2,2,3,3))
model.matrix(~ group + 0)
```

```
##   group1 group2 group3
## 1      1      0      0
## 2      1      0      0
```

More variables

We have been using a simple case with just one variable (diet) as an example. In the life sciences, it is quite common to perform experiments with more than one variable. For example, we may be interested in the effect of diet and the difference in sexes. In this case, we have four possible groups:

```r
diet <- factor(c(1,1,1,1,2,2,2,2))
sex <- factor(c("f","f","m","m","f","f","m","m"))
table(diet,sex)
```

```
##     sex
## diet f m
##    1 2 2
##    2 2 2
```

If we assume that the diet effect is the same for males and females (this is an assumption), then our linear model is:

$$Y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \varepsilon_i$$

To fit this model in R, we can simply add the additional variable with a + sign in order to build a design matrix which fits based on the information in additional variables:

```r
diet <- factor(c(1,1,1,1,2,2,2,2))
sex <- factor(c("f","f","m","m","f","f","m","m"))
model.matrix(~ diet + sex)
```

```
##   (Intercept) diet2 sexm
## 1           1     0    0
## 2           1     0    0
```

Releveling

The level which is chosen for the *reference level* is the level which is contrasted against. By default, this is simply the first level alphabetically. We can specify that we want group 2 to be the reference level by either using the `relevel` function:

```
group <- factor(c(1,1,2,2))
group <- relevel(group, "2")
model.matrix(~ group)
```

```
##   (Intercept) group1
## 1           1      1
## 2           1      1
## 3           1      0
## 4           1      0
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$group
## [1] "contr.treatment"
```

or by providing the levels explicitly in the `factor` call:

```
group <- factor(group, levels=c("1","2"))
model.matrix(~ group)
```

```
##   (Intercept) group2
## 1           1      0
## 2           1      0
## 3           1      1
## 4           1      1
```

Where does model.matrix look for the data?

The `model.matrix` function will grab the variable from the R global environment, unless the data is explicitly provided as a data frame to the `data` argument:

```
group <- 1:4
model.matrix(~ group, data=data.frame(group=5:8))
```

```
##   (Intercept) group
## 1           1     5
## 2           1     6
## 3           1     7
## 4           1     8
## attr(,"assign")
## [1] 0 1
```

Note how the R global environment variable `group` is ignored.

Continuous variables

In this chapter, we focus on models based on indicator values. In certain designs, however, we will be interested in using numeric variables in the design formula, as opposed to converting them to factors first. For example, in the falling object example, time was a continuous variable in the model and time squared was also included:

```
tt <- seq(0,3.4,len=4)
model.matrix(~ tt + I(tt^2))
```

```
##   (Intercept)       tt   I(tt^2)
## 1           1 0.000000  0.000000
## 2           1 1.133333  1.284444
## 3           1 2.266667  5.137778
## 4           1 3.400000 11.560000
## attr(,"assign")
## [1] 0 1 2
```

The `I` function above is necessary to specify a mathematical transformation of a variable. For more details, see the manual page for the `I` function by typing `?I`.

In the life sciences, we could be interested in testing various dosages of a treatment, where we expect a specific relationship between a measured quantity and the dosage, e.g. 0 mg, 10 mg, 20 mg.

The assumptions imposed by including continuous data as variables are typically hard to defend and motivate than the indicator function variables. Whereas the indicator variables simply assume a different mean between two groups, continuous variables assume a very specific relationship between the outcome and predictor variables.

In cases like the falling object, we have the theory of gravitation supporting the model. In the father son height example, because the data is bivariate normal, it follows that