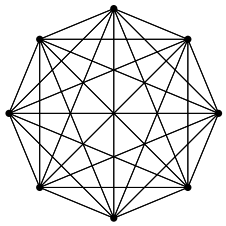
Chapter 6

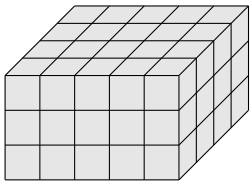
Arrays



This chapter introduces the *array* data structure in R, which is the next logical step after introducing vectors and matrices. As was the case with vectors and matrices, we assume for now that the elements in an array are numerical values. There is a geometric interpretation of the progression of data structures surveyed so far.

- Chapter 3 introduced *simple objects* consisting of just a single element that can assume a numeric value. These objects are analogous to *points* in geometry.
- Chapter 4 introduced *vectors* whose elements can assume a number of numeric values arranged linearly. Vectors are analogous to *lines* in geometry.
- Chapter 5 introduced *matrices* whose elements can assume a number of numeric values arranged in a rectangular fashion. Matrices are analogous to *planes* in geometry.
- This chapter introduces *arrays* whose elements can assume a number of numeric values arranged in a fashion that requires three or more subscripts to access an individual element. Arrays in three dimensions are analogous to *rectangular solids* in geometry.

The figure below depicts the arrangement of elements in a three-dimensional array. A numerical element is stored in each cube. The front face of the array is a 3×5 matrix, and there are several layers of identically-dimensioned matrices lying just behind the first matrix. An element in such a matrix can be accessed by three subscript values: the first is the row number, the second is the column number, and the third is, for lack of a better term, the layer number. The dimensions of the three-dimensional array depicted in the figure are $3 \times 5 \times 4$. There are a total of $3 \cdot 5 \cdot 4 = 60$ elements in the array.



Before diving into the specifics of how to create and manipulate an array in R, two examples will illustrate instances of their use. The first example is from meteorology; the second example is from photography.

- Yuxin is a meteorologist. She has collected hourly meteorological data in a rectangular grid of locations for a particular day associated with
  - ten longitudes,
  - ten latitudes, and
  - 24 hourly measurements.

  The numerical values collected are
  - surface temperature,
  - surface humidity,
  - surface wind speed, and
  - surface barometric pressure.

  What is the best way for Yuxin to store this data in R? The array data structure in her case would be a four-dimensional array with dimensions $10 \times 10 \times 24 \times 4$, with the first dimension for the longitudes, the second dimension for the latitudes, the third dimension for the hour of the day, and the fourth dimension for the particular meteorological quantity being stored.

- Taylor is a photographer. He would like to store 100 color digital photographs in R. Each digital photograph is 1000 by 1500 pixels, and all photos are taken in the landscape mode. The color associated with each pixel in a photograph is captured by
  - the amount of red, ranging from 0 to 255,
  - the amount of green, ranging from 0 to 255, and
  - the amount of blue, ranging from 0 to 255.

  What is the best way for Taylor to store the digital photographs in R? The array data structure in his case would also be a four-dimensional array with dimensions $100 \times 1000 \times 1500 \times 3$, with the first dimension for the image number, the second dimension for the pixel row number, the third dimension for the pixel column number, and the fourth dimension for storing the color parameters. Each element in the array contains a numerical value between 0 and 255.

The three topics introduced in this chapter are (a) creating an array, (b) extracting elements of an array by using subscripts, and (c) performing arithmetic operations on an array.

6.1 Creating an array

Arrays can be created in R with the `array` function, which has the following syntax.

```
array(data = NA, dim, dimnames = NULL)
```

The `array` function is a generalization of the `matrix` function. The first argument is the vector `data`, which contains the elements that will populate the array. As with the `matrix` function, the default for `data` is `NA`. The second argument `dim` is a vector of integer elements that establishes the largest indexes in each dimension. As was the case with vectors and matrices, these subscript values begin at 1. The third argument `dimnames` can be used to place labels on the various dimensions of the array. The first example of the use of the `array` function creates an array of zeros (recycling is used) to form a $3 \times 5 \times 2$ array.