7.3 Sorting, ordering, and ranking

There are a number of R functions that sort, order, and rank numeric values. The `rev` (reverse) function reverses the order of the elements. The `order` function gives the permutation of the subscripts that results in a sorted vector. The `rank` function operates similarly, but averages the ranks for tied observations. These functions are illustrated for a vector `x`.

```
> x = c(2, 0, -3, 2)  # a vector of data values
> x                   # display x
[1]  2  0 -3  2
> rev(x)              # reverse the order of the elements
[1]  2 -3  0  2
> unique(x)           # unique elements
[1]  2  0 -3
> sort(x)             # sort into ascending order
[1] -3  0  2  2
> sort(x, decreasing = TRUE)   # sort into descending order
[1]  2  2  0 -3
> order(x)            # ordering permutation via subscripts
[1] 3 2 1 4
> rank(x)             # ordering permutation via subscripts adjusted for ties
[1] 3.5 2.0 1.0 3.5
```

The `sort` function places missing values (`NA`s) at the end of the returned vector by default. This can be altered by setting the `na.last` argument to `FALSE` to place the missing values at the beginning of the returned vector or setting the `na.last` argument to `NA` to remove the missing values.

As an illustration of the application of these sorting functions, consider an automobile company that has eight plants that manufacture the same car. The cost per car manufactured at the eight plants, in thousands of dollars, is given in the vector named `cost`. The associated monthly capacity of the eight plants, in thousands of cars, is given in the vector named `ncar`. The cumulative number of cars that can be manufactured, ordered by the cost of manufacturing the cars (lowest cost first), can be generated by using the `cumsum` and `order` functions.

```
> cost = c(12.4, 11.7, 13.5, 13.1, 12.0, 12.8, 11.6, 13.3)
> ncar = c(30.3, 24.7, 16.8, 30.0, 14.0, 33.0, 22.1, 29.4)
> cumsum(ncar[order(cost)])
[1]  22.1  46.8  60.8  91.1 124.1 154.1 183.5 200.3
```

7.4 Properties of an object

R has a group of built-in functions that determine certain properties of an object. To illustrate these functions, set `x` to a vector, `y` to a matrix, and `z` to an array.

```
> x = 1:8                        # x is a vector
> y = matrix(1:8, 2, 4)          # y is a 2 x 4 matrix
> z = array(1:8, c(2, 2, 2))     # z is a 2 x 2 x 2 array
```

The first function, `class`, returns a character string (which will be enclosed in quotes) that describes the type of data structure associated with a particular object.