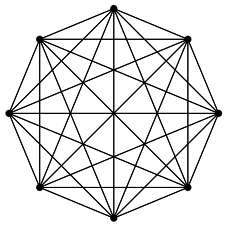
Chapter 11

Character Strings



Data structures, such as vectors, matrices and arrays, have elements that have thus far been assumed to be either numeric values or complex numbers. This chapter introduces a third possibility for these elements: character strings.

A single character in R is stored in memory in eight consecutive bits that computer scientists call a *byte*. A *bit* is a binary digit—a 0 or a 1. The American Standard Code for Information Interchange, which is abbreviated by ASCII, is a character encoding standard that uses seven of the eight bits in a byte to store $2^7 = 128$ symbols (256 symbols in extended ASCII), most of which are on the keys on a standard keyboard. The 26 lowercase letters, the 26 uppercase letters, the 10 digits, and dozens of symbols are defined as particular strings of seven bits in ASCII. A single such letter or symbol is often referred to as a *character*. A sequence of characters is known as a *character string*.

This chapter introduces character strings in R. The topics considered in this chapter are (a) simple character strings, (b) automatic coercion, (c) built-in strings, (d) string manipulation, (e) names, and (f) factors.

11.1 Simple character strings

Placing characters in single or double quotes constitutes a character string. For example, in order to assign the character string `"R"` to the object `x` and display the value of `x`, type

```
> x = "R"                                  # a character: ASCII 1010010
> x                                        # display x
[1] "R"
```

The `[1]` is again displayed because `x` is a vector of length one consisting of a single element that has been assigned the character string `"R"`. Using single quotes to assign the character string `'S'` to `x` is done in the same fashion:

```
> x = 'S'                                  # a character: ASCII 1010011
> x                                        # display x
[1] "S"
> nchar(x)                                 # character count
[1] 1
```

R displays strings surrounded by double quotes, regardless of whether they are entered with single or double quotes. The `nchar` function accepts a string as its argument and returns the number of

characters in the string, which in this case is one. Strings with multiple characters are input in the same fashion:

```
> x = "Hello world!"                    # a string of characters
> x                                     # display x
[1] "Hello world!"
> nchar(x)                              # character count
[1] 12
```

This time the nchar function returns 12, which is a count of the number of characters in the string "Hello world!" (not including the quotes, but including the space and the exclamation point). The substr (substring) function allows a portion of a string to be extracted.

```
> substr(x, start = 2, stop = 10)       # substring function
[1] "ello worl"
```

Occasions may arise in which you would like to have a quotation mark or apostrophe within a character string. This can be accomplished in R by placing a backslash before the quotation mark or apostrophe. When nchar is called in the example below, it counts the apostrophe as just a single character.

```
> x = "I can\'t go there"               # a string of characters
> x                                     # display x
[1] "I can't go there"
> nchar(x)                              # character count
[1] 16
```

Octal (base 8) is a compact way to write a binary number. Beginning a string with a backslash indicates an octal number in a string. The octal number 007 (binary 0000111) corresponds to the chime sound in ASCII on a laptop or desktop computer. So one way to sound this chime is to use the cat (concatenate) function to output this particular octal code:

```
> x = "\007"                            # bell sound as octal ASCII code
> cat(x)                                # ring the bell
```

The uppercase R is binary code 1010010 in ASCII, which is octal 122. Calling the print function below prints R as a one-element vector that consists of a character string as an element, but calling the cat function just prints the letter R (not as a vector and without the quotation marks).

```
> x = "\122"                            # letter R as octal ASCII code
> print(x)                              # prints the object x
[1] "R"
> cat(x)                                # display R
R
```

Finally, calling the cat function to display the letter R, then a tab, then another letter R, and then a newline character (to move the next command prompt to the subsequent line) results in

```
> cat(x, "\t", x, "\n")                 # display R, tab, R, newline
R       R
```

11.2 Automatic coercion

R does not allow you to mix data types within the data structures that have been introduced so far. For example, you cannot mix numeric values, complex values, and character strings together in a vector. (There is a data structure known as a *list*, introduced in Chapter 15, which allows this mixing.) If you do mix these data types, R will perform what is known as *automatic coercion*, which coerces the elements to all have the same type. In setting `x` to a vector of character strings and numeric values, for example, R will coerce all elements to be character strings:

```
> x = c("R", 4.7, 3, -8, 6.02e23,  pi)     # mixed data types coerced
> x                                         # display x
[1] "R"                "4.7"              "3"                "-8"
[5] "6.02e+23"         "3.14159265358979"
```

Coercing numerics to be character strings is more sensible than coercing character strings to be numerics (for example, it is not clear what numerical value should be assigned to `"Hello world!"`). Applying the `nchar` function to the vector `x` results in the following six counts:

```
> nchar(x)                                  # character count
[1]  1  3  1  2  8 16
```

Applying the `substr` function with `start = 2` and `stop = 4` (the arguments are referenced by position here) to the `x` vector results in

```
> substr(x, 2, 4)                           # substring function
[1] ""    ".7"  ""    "8"   ".02" ".14"
```

The character strings displayed are the second through the fourth characters in the each of the six elements in `x`. You are not limited to just automatic coercion in R. Chapter 14 will introduce functions that can be used for custom coercion of objects.

11.3 Built-in objects of character strings

R contains some built-in objects consisting of character strings as elements that can often be useful as labels in graphics. The objects `letters` and `LETTERS`, for instance, are each 26-element vectors that contain the lowercase and uppercase letters, respectively.

```
> letters                                   # lowercase letters
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r"
[19] "s" "t" "u" "v" "w" "x" "y" "z"
> LETTERS                                   # uppercase letters
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R"
[19] "S" "T" "U" "V" "W" "X" "Y" "Z"
```

A specific letter can be accessed using a subscript in the usual fashion:

```
> LETTERS[18]                               # one uppercase letter
[1] "R"
```

Two other vectors of built-in character strings are `month.name` and `month.abb`, which contain the months of the year and their abbreviations, respectively.