R objects as storage entities

While you can accomplish many things entering commands directly into the RStudio command console, there will be times when you want to store the results of an operation so you can apply multiple techniques without recomputing the original result as you apply each technique. You can store your computational results in an R object and then apply your new computations using that object.

An R object can be thought of as a container for your data. The concept is similar to the variables in C++ or Java. An R object is a user-defined name for the container. You can use any character string as the name for an object. Some examples of possible object names are:

x, abc, myData, LongObjectName, x2, 45_data

Any combination of letters and numbers can be used for object names, but there are some characters and combinations you should avoid. You cannot use spaces in your object names. R treats spaces as the separator between different objects and commands. The object name

My Object

will produce and error from the R interpreter. It will treat this as two character strings and not a single object name. Most special characters [like: @, # , and $] are acceptable, it is best to use only the underline _ and period . in your object names. These symbols are often used in place of spaces in longer, more descriptive, object names. It is advisable, for fairly obvious reasons, to not use object names consisting of only numbers. While most of these recommendations are mainly stylistic, they help maintain the clarity of your R code.

An earlier topic discussed the R functional programming model. It explained how R treats all expressions as functions that return a value. Any return value can be assigned [stored in] an R object. You assign data to an object using one of two possible symbols. Both of these statements assign the results of a computation to an object.

```
x = 2 + 5
x <- 2 + 5
```

The **=** symbol and the **<-** symbol assign the result of **2 + 5** to the object **x**. You will see both symbols used R literature and forums. Throughout these instructional topics, you will see the symbol **=** used most often. This symbol takes only one keystroke [the lazy option]. The **<-** symbol can potentially produce a confusing error if you accidentally place a space between the **<** and the **-** symbols. R interprets this expression

```
x < - 8
```

as a logical test whether **x** is less than **- 8** rather than assigning **8** to the object **x**. the correct assignment expression is

```
x <- 8
```

If you find that your typing style often inserts the extra space [thereby producing an error], you might consider using **=** instead of **<-**.

R objects and data typing

If you are familiar with programming languages such as C++ or Java, you are used to their strong data-typing model. In C++, a variable is assigned a storage data type [like int] and that variable will only be permitted to hold that type of data. While this model prevents operational errors, it also reduces flexibility. R uses a more flexible dynamic data-typing model. It determines the data type of an object when data is assigned to that object. This means that an object can store one data type and, later, store another data type. Here is an example of this dynamic data-typing model in use:

```
x = 3 / 5            # x contains a decimal value
x = "An example text string"    # x now contains a character string
x = c(1, 2, 3, 4, 5, 6)      # x now contains a vector of integer values
```

In each case in the example above, x is assigned the data type of the data it contains. The flexibility of this model allows you to work on your computing problem without worrying about the details of data types. The unfortunate side effect of this model is that you can incorrectly use an object in a computation. When this occurs, R will respond with an error statement that identifies the data type mismatch. Most computational data type problems can be avoided. All of your active objects, and descriptions of the data they contain, are shown in the RStudio environment panel. The environment panel object references should help you remember and use your active objects appropriately.