Beginning\_R\_Joshua\_C01

Beginning R An Introduction to Statistical Programming Second Edition

p18

Introduction

This book is about the R programming language. Maybe more important, this book is for you.

These days, R is an impressively robust language for solving problems that lend themselves to statistical programming methods. There is a large community of users and developers of this language, and together we are able to accomplish things that were not possible before we virtually met.

—----------------------------------------------

En estos días, R es un lenguaje impresionantemente robusto para resolver problemas que se prestan a métodos de programación estadística. Hay una gran comunidad de usuarios y desarrolladores de este lenguaje, y juntos podemos lograr cosas que no eran posibles antes de conocernos virtualmente.

—----------------------------------------------

Of course, to leverage this collective knowledge, we have to start somewhere. Chapters 1 through 5 focus on gaining familiarity with the R language itself. If you have prior experience in programming, these chapters will be very easy for you. If you have no prior programming experience, that is perfectly fine. We build from the ground up, and let us suggest you spend some thoughtful time here. Thinking like a programmer has some very great advantages. It is a skill we would want you to have, and this book is, after all, for you.

—----------------------------------------------

Por supuesto, para aprovechar este conocimiento colectivo, tenemos que empezar por alguna parte. Los capítulos 1 a 5 se centran en familiarizarse con el propio lenguaje R. Si tienes experiencia previa en programación, estos capítulos te resultarán muy fáciles. Si no tienes experiencia previa en programación, está perfectamente bien. Construimos desde cero, y permítanos sugerirle que pase un tiempo reflexivo aquí. Pensar como un programador tiene grandes ventajas. Es una habilidad que nos gustaría que tuvieras y, después de todo, este libro es para ti.

—----------------------------------------------

Chapters 6 through 10 focus on what might be termed elementary statistical methods in R. We did not have the space to introduce those methods in their entirety—we are supposing some knowledge of statistics. An introductory or elementary course for nonmajors would be more than enough. If you are already familiar with programming and statistics, we suggest you travel through these chapters only briefly.

—----------------------------------------------

Los capítulos 6 a 10 se enfocan en lo que podría llamarse métodos estadísticos elementales en R. No tuvimos el espacio para presentar esos métodos en su totalidad; suponemos algún conocimiento de estadística. Un curso introductorio o elemental para no mayores sería más que suficiente. Si ya está familiarizado con la programación y las estadísticas, le sugerimos que viaje a través de estos capítulos solo brevemente.

—----------------------------------------------

With Chapter 11, we break into the last part of the book. For someone with both a fair grasp of traditional statistics and some programming experience, this may well be a good place to start. For our readers who read through from the first pages, this is where it starts to get very exciting. From bootstrapping to logistic regression to data visualization to high-performance computing, these last chapters have hands-on examples that work through some much applied and very interesting examples.

—----------------------------------------------

Con el Capítulo 11, entramos en la última parte del libro. Para alguien con un buen conocimiento de las estadísticas tradicionales y algo de experiencia en programación, este puede ser un buen lugar para comenzar. Para nuestros lectores que leen desde las primeras páginas, aquí es donde comienza a ser muy emocionante. Desde el arranque hasta la regresión logística, desde la visualización de datos hasta la computación de alto rendimiento, estos últimos capítulos tienen ejemplos prácticos que funcionan a través de algunos ejemplos muy aplicados y muy interesantes.

—----------------------------------------------

One final note: While we wrote this text from Chapter 1 to Chapter 19 in order, the chapters are fairly independent of each other. Don't be shy about skipping to the chapter you're most interested in learning. We show all our code, and you may well be able to modify what we have to work with what you have.

—----------------------------------------------

Una nota final: si bien escribimos este texto desde el Capítulo 1 hasta el Capítulo 19 en orden, los capítulos son bastante independientes entre sí. No dude en saltar al capítulo que más le interese aprender. Mostramos todo nuestro código, y es posible que puedas modificar lo que tenemos para trabajar con lo que tienes.

—----------------------------------------------

Happy reading

pag19

**Chapter 1**

**Getting Started**

There are compelling reasons to use R. Enthusiastic users, programmers, and contributors support R and its development. A dedicated core team of R experts maintains the language. R is accurate, produces excellent graphics, has a variety of built-in functions, and is both a functional language and an object-oriented one. There are (literally) thousands of contributed packages available to R users for specialized data analyses.

—------------------------------------------

Existen razones convincentes para usar R. Los usuarios, programadores y colaboradores entusiastas respaldan R y su desarrollo. Un equipo central dedicado de expertos en R mantiene el lenguaje. R es preciso, produce excelentes gráficos, tiene una variedad de funciones integradas y es un lenguaje funcional y orientado a objetos. Hay (literalmente) miles de paquetes contribuidos disponibles para los usuarios de R para análisis de datos especializados.

—------------------------------------------

Developing from a novice into a more competent user of R may take as little as three months by only using R on a part-time basis (disclaimer: n = 1). Realistically, depending on background, your development may take days, weeks, months, or even a few years, depending on how often you use R and how quickly you can learn its many intricacies. R users often develop into R programmers who write R functions, and R programmers sometimes want to develop into R contributors, who write packages that help others with their data analysis needs. You can stop anywhere on that journey you like, but if you finish this book and follow good advice, you will be a competent R user who is ready to develop into a serious R programmer if you want to do it. We wish you the best of luck!

—------------------------------------------

Pasar de ser un novato a un usuario más competente de R puede tomar tan solo tres meses usando R solo a tiempo parcial (descargo de responsabilidad: n = 1). Siendo realistas, dependiendo de los antecedentes, su desarrollo puede llevar días, semanas, meses o incluso algunos años, dependiendo de la frecuencia con la que use R y la rapidez con la que pueda aprender sus muchas complejidades. Los usuarios de R a menudo se convierten en programadores de R que escriben funciones de R, y los programadores de R a veces quieren convertirse en colaboradores de R, que escriben paquetes que ayudan a otros con sus necesidades de análisis de datos. Puede detenerse en cualquier parte de ese viaje que desee, pero si termina este libro y sigue los buenos consejos, será un usuario de R competente que está listo para convertirse en un programador de R serio si así lo desea. ¡Te deseamos la mejor de las suertes!

—------------------------------------------

1.1 What is R, Anyway?

¿Qué es R, de todos modos?

R is an open-source implementation of the S language created and developed at Bell Labs. S is also the basis of the commercial statistics program S-PLUS, but R has eclipsed S-PLUS in popularity. If you do not already have R on your system, the quickest way to get it is to visit the CRAN (Comprehensive R Network Archive) website and download and install the precompiled binary files for your operating system. R works on Windows, Mac OS, and Linux systems. If you use Linux, you may already have R with your Linux distribution. Open your terminal and type $ R --version. If you do not already have R, the CRAN website is located at the following URL:

—-------------------------------------------------------------------

R es una implementación de código abierto del lenguaje S creada y desarrollada en Bell Labs. S también es la base del programa de estadísticas comerciales S-PLUS, pero R ha eclipsado a S-PLUS en popularidad.

Si aún no tiene R en su sistema, la forma más rápida de obtenerlo es visitar el sitio web CRAN (Comprehensive R Network Archive) y descargar e instalar los archivos binarios precompilados para su sistema operativo. R funciona en sistemas Windows, Mac OS y Linux. Si usa Linux, es posible que ya tenga R con su distribución de Linux. Abra su terminal y escriba $ R --version. Si aún no tiene R, el sitio web de CRAN se encuentra en la siguiente URL:

—-----------------------------------------------------------------

http://cran.r-project.org/

Download and install the R binaries for your operating system, accepting all the defaults. At this writing, the current version of R is 3.2.0, and in this book, you will see screenshots of R working in both Windows 7 and Windows 8.1. Your authors run on 64-bit operating systems, so you will see that information displayed in the screen captures in this book. Because not everything R does in Unix-based systems can be done in Windows, I often switch to Ubuntu to do those things, but we will discuss only the Windows applications here, and leave you to experiment with Ubuntu or other flavors of Unix. One author runs Ubuntu on the Amazon Cloud, but that is way beyond our current needs.

—-----------------------------------------------------------------

Descargue e instale los binarios de R para su sistema operativo, aceptando todos los valores predeterminados. En el momento de escribir este artículo, la versión actual de R es 3.2.0 y, en este libro, verá capturas de pantalla de R funcionando tanto en Windows 7 como en Windows 8.1. Sus autores se ejecutan en sistemas operativos de 64 bits, por lo que verá esa información en las capturas de pantalla de este libro. Debido a que no todo lo que hace R en los sistemas basados en Unix se puede hacer en Windows, a menudo cambio a Ubuntu para hacer esas cosas, pero aquí solo discutiremos las aplicaciones de Windows y lo dejaremos experimentar con Ubuntu u otras versiones de Unix. Un autor ejecuta Ubuntu en Amazon Cloud, pero eso va mucho más allá de nuestras necesidades actuales.

—-----------------------------------------------------------------

Go ahead and download Rstudio (current version as of this writing is 0.98.1103) now too, again, accepting all defaults from the following URL:

<http://www.rstudio.com/products/rstudio/download/>

Rstudio is a very forgiving environment for the novice user, and code written in here will work just as well in R itself.

—-----------------------------------------------------------

Rstudio es un entorno muy indulgente para el usuario novato, y el código escrito aquí funcionará igual de bien en R mismo.

—-----------------------------------------------------------

Launch Rstudio and examine the resulting interface. Make sure that you can identify the following parts of the R interface shown in Figure 1-1: the menu bar, the script editing area, the R console, and the R command prompt, which is >.

—-----------------------------------------------------------

Inicie Rstudio y examine la interfaz resultante. Asegúrese de que puede identificar las siguientes partes de la interfaz de R que se muestran en la Figura 1-1: la barra de menú, el área de edición de secuencias de comandos, la consola de R y el símbolo del sistema de R, que es >.

—-----------------------------------------------------------

Before we continue our first R session, let’s have a brief discussion of how R works. R is a high-level vectorized computer language and statistical computing environment. You can write your own R code, use R code written by others, and use R packages you write and those written by you or by others. You can use R in batch mode, terminal mode, in the R graphical user interface (RGui), or in Rstudio, which is what we will do in this book. As you learn more about R and how to use it effectively, you will find that you can integrate R with other languages such as Python or C++, and even with other statistical programs such as SPSS.

—-----------------------------------------------------------

Antes de continuar con nuestra primera sesión de R, analicemos brevemente cómo funciona R. R es un lenguaje informático vectorizado de alto nivel y un entorno informático estadístico. Puede escribir su propio código R, usar código R escrito por otros y usar paquetes R que usted escribe y los escritos por usted o por otros. Puede usar R en modo por lotes, modo terminal, en la interfaz gráfica de usuario de R (RGui) o en Rstudio, que es lo que haremos en este libro. A medida que aprenda más sobre R y cómo usarlo de manera efectiva, descubrirá que puede integrar R con otros lenguajes como Python o C++, e incluso con otros programas estadísticos como SPSS.

—-----------------------------------------------------------

In some computer languages, for instance, C++, you have to declare a data type before you assign a value to a new variable, but that is not true in R. In R, you simply assign a value to the object, and you can change the value or the data type by assigning a new one. There are two basic assignment operators in R. The first is < −, a left-pointing assignment operator produced by a less than sign followed by a “minus” sign, which is really a hyphen. You can also use an equals sign = for assignments in R. I prefer the < − assignment operator, and will use it throughout this book.

—--------------------------------------------------------

En algunos lenguajes de programación, por ejemplo, C++, debe declarar un tipo de datos antes de asignar un valor a una nueva variable, pero eso no es cierto en R. En R, simplemente asigna un valor al objeto y puede cambiar el valor o el tipo de dato asignando uno nuevo. Hay dos operadores de asignación básicos en R. El primero es < −, un operador de asignación que apunta hacia la izquierda producido por un signo menor que seguido de un signo “menos”, que en realidad es un guión. También puede usar un signo igual = para asignaciones en R. Prefiero el operador de asignación < − y lo usaré a lo largo de este libro.

—--------------------------------------------------------

You must use the = sign to assign the parameters in R functions, as you will learn. R is not sensitive to white space the way some languages are, and the readability of R code is benefited from extra spacing and indentation, although these are not mandatory. R is, however, case-sensitive, so to R, the variables x and X are two different things. There are some reserved names in R, which I will tell you about in Chapter 5.

—--------------------------------------------------------

Debe usar el signo = para asignar los parámetros en las funciones R, como aprenderá. R no es sensible a los espacios en blanco como lo son algunos lenguajes, y la legibilidad del código R se beneficia del espacio adicional y la sangría, aunque estos no son obligatorios. Sin embargo, R distingue entre mayúsculas y minúsculas, por lo que para R, las variables x y X son dos cosas diferentes. Hay algunos nombres reservados en R, de los que hablaré en el Capítulo 5.

—--------------------------------------------------------

The best way to learn R is to use R, and there are many books, web-based tutorials, R blog sites, and videos to help you with virtually any question you might have. We will begin with the basics in this book but will quickly progress to the point that you are ready to become a purposeful R programmer, as mentioned earlier.

—---------------------------------------------------------

La mejor manera de aprender R es usar R, y hay muchos libros, tutoriales basados en la web, sitios de blogs de R y videos para ayudarlo con prácticamente cualquier pregunta que pueda tener. Comenzaremos con los conceptos básicos en este libro, pero progresaremos rápidamente hasta el punto en que esté listo para convertirse en un programador de R con propósito, como se mencionó anteriormente.

—--------------------------------------------------------

Let us complete a five-minute session in R, and then delve into more detail about what we did, and what R was doing behind the scenes. The most basic use of R is as a command-line interpreted language. You type a command or statement after the R prompt and then press , and R attempts to implement the command. If R can do what you are asking, it will do it and return the result in the R console. If R cannot do what you are asking, it will return an error message. Sometimes R will do something but give you warnings, which are messages concerning what you have done and what the impact might be, but that are sometimes warnings that what you did was not what you probably wanted to do. Always remember that R, like any other computer language, cannot think for you.

—-----------------------------------------------------------------

Completemos una sesión de cinco minutos en R y luego profundicemos en más detalles sobre lo que hicimos y lo que R estaba haciendo detrás de escena. El uso más básico de R es como un lenguaje interpretado de línea de comandos. Escribe un comando o declaración después de la solicitud de R y luego presiona , y R intenta implementar el comando. Si R puede hacer lo que le pides, lo hará y devolverá el resultado en la consola de R. Si R no puede hacer lo que le pide, devolverá un mensaje de error. A veces, R hará algo pero le dará advertencias, que son mensajes sobre lo que ha hecho y cuál podría ser el impacto, pero a veces son advertencias de que lo que hizo no era lo que probablemente quería hacer. Recuerde siempre que R, como cualquier otro lenguaje informático, no puede pensar por usted.

—-------------------------------------------------------------------

1.2 A First R Session

Okay, let’s get started. In the R console, type Ctrl + L to clear the console in order to have a little more working room. Then type the following, pressing the key at the end of each command you type. When you get to the personal information, substitute your own data for mine:

—-----------------------------------------

Bien, comencemos. En la consola R, escriba Ctrl+ L para despejar la consola y tener un poco más de espacio de trabajo. Luego escriba lo siguiente, presionando la tecla al final de cada comando que escriba. Cuando llegue a la información personal, sustituya sus propios datos por los míos:

—-------------------------------------------

> 1 + 1

[ 1 ] 2

> 1 ^ 1

[ 1 ] 1

> 1 \* 1

[ 1 ] 1

> 1 - 1

[ 1 ] 0

> 1 : 1 0

[ 1 ] 1 2 3 4 5 6 7 8 9 1 0

> ( 1 : 1 0 ) ^ 2

[ 1 ] 1 4 9 1 6 2 5 3 6 4 9 6 4 8 1 1 0 0

> myName <- "Joshua Wiley"

> myAlmaMater <- "University of California, Los Angeles"

> myURL <- "www.JoshuaWiley.com"

> myPhone <- "1.260.673.5518"

> myData <- list(myName, myAlmaMater, myURL, myPhone)

> myData [[1]]

[1] "Joshua Wiley" [[2]]

[1] "University of California, Los Angeles" [[3]] [1] "www.JoshuaWiley.com" [[4]] [1] "1.260.673.5518"

We began with the use of R as a basic calculator. We can create sequences of integers by using the colon operator. Using the exponentiation operator (1:10) ˆ2 gives us the squares of all the numbers in the vector 1 to 10. Observe that when you type a command and press the key, R will return the result on the next line, prefaced by an index, such as [1]. You can assign values to variables without declaring the variable type, as we discussed, so you can just type myName < − "Joshua Wiley" to give the variable a value.

—---------------------------------------

Empezamos con el uso de R como calculadora básica. Podemos crear secuencias de enteros usando el operador de dos puntos. Usando el operador de exponenciación (1:10) ˆ2 nos da los cuadrados de todos los números en el vector 1 a 10. Observe que cuando escribe un comando y presiona la tecla, R devolverá el resultado en la línea siguiente, precedido por un índice, como [1]. Puede asignar valores a las variables sin declarar el tipo de variable, como comentamos, por lo que puede simplemente escribir myName < − "Joshua Wiley" para darle un valor a la variable.

—---------------------------------------

This might have seemed a strange way to start, but it shows you some of the things you can enter into your R workspace simply by assigning them. Character strings must be enclosed in quotation marks, and you can use either single or double quotes. Numbers can be assigned as they were with the myPhone variable. With the name and address, we created a list, with is one of the basic data structures in R. Unlike vectors, lists can contain multiple data types. We also see square brackets [ and ], which are R’s way to index the elements of a data object, in this case our list. We can also create vectors, matrices, and data frames in R. Let’s see how to save a vector of the numbers from 1 to 10. We will call the vector x. We will also create a “constant” called y:

—---------------------------------------------------

Esta puede haber parecido una forma extraña de comenzar, pero le muestra algunas de las cosas que puede ingresar en su espacio de trabajo de R simplemente asignándolas. Las cadenas de caracteres deben estar entre comillas y puede usar comillas simples o dobles. Los números se pueden asignar tal como estaban con la variable myPhone. Con el nombre y la dirección, creamos una lista, que es una de las estructuras de datos básicas en R. A diferencia de los vectores, las listas pueden contener varios tipos de datos. También vemos corchetes [ y ], que son la forma en que R indexa los elementos de un objeto de datos, en este caso nuestra lista. También podemos crear vectores, matrices y marcos de datos en R. Veamos cómo guardar un vector de los números del 1 al 10. Llamaremos al vector x. También crearemos una “constante” llamada y:

—--------------------------------------------------

> x < - 1 : 1 0

> x [ 1 ] 1 2 3 4 5 6 7 8 9 1 0

> y < - 5 >

y

[ 1 ] 5

See that R starts its listing of both x and y with an index [1]. This is because R does not recognize a scalar value. To R, even a single number is a vector. The object y is a vector with one element. The [1] in front of x means that the first element of the vector appears at the beginning of the line. Let’s make another vector, z, containing a sequence of 33 randomly generated numbers from a normal distribution with a mean of 70 and a standard deviation of 10. Because the numbers are random, your z vector will not be the same as mine, though if we wanted to, we could set the seed number in R so that we would both get the same vector:

—---------------------------------------------------

Vea que R comienza su listado de x e y con un índice [1]. Esto se debe a que R no reconoce un valor escalar. Para R, incluso un solo número es un vector. El objeto y es un vector con un elemento. El [1] delante de x significa que el primer elemento del vector aparece al principio de la línea. Hagamos otro vector, z, que contenga una secuencia de 33 números generados aleatoriamente a partir de una distribución normal con una media de 70 y una desviación estándar de 10. Debido a que los números son aleatorios, su vector z no será el mismo que el mío, aunque si queríamos, podríamos establecer el número semilla en R para que ambos obtuviéramos el mismo vector:

—----------------------------------------------------

> z < - r n o r m ( 3 3 , 70 , 1 0 )

> z < - r o u n d ( z , 2 )

> z [ 1 ] 8 1 . 5 6 7 0 . 8 5 7 7 . 4 8 6 4 . 0 2 6 8 . 9 4 8 0 . 2 4 6 0 . 8 4 7 0 . 9 3 7 5 . 2 1 7 5 . 0 5 5 2 . 1 7 5 2 . 2 9 [ 1 3 ] 7 0 . 2 0 7 9 . 2 9 8 4 . 7 5 6 4 . 8 8 7 3 . 7 4 7 1 . 1 9 6 1 . 0 1 6 3 . 4 3 5 5 . 7 4 7 1 . 5 4 6 9 . 7 1 8 2 . 5 2 [ 2 5 ] 7 3 . 4 0 7 5 . 3 9 7 9 . 2 8 8 0 . 3 6 6 5 . 7 9 7 3 . 1 5 7 5 . 4 1 6 9 . 5 6 8 5 . 8 7

When R must wrap to a new line in the console to print additional output, it shows the index of the first element of each line.

—-----------------------------------------------------------

Cuando R debe ajustarse a una nueva línea en la consola para imprimir una salida adicional, muestra el índice del primer elemento de cada línea.

—-----------------------------------------------------------

To see a list of all the objects in your R session, type the command ls():

Para ver una lista de todos los objetos en su sesión de R, escriba el comando ls():

> ls()

[1] "myAlmaMater" "myData" "myName" "myPhone" "myURL" "x" "y" "z"

To see the current working directory, type the command getwd().

Para ver el directorio de trabajo actual, escriba el comando getwd().

You can change the working directory by typing setwd(), but I usually find it easier to use the File menu. Just select File > Change dir... and navigate to the directory you want to become the new working directory. As you can see from the code listing here, the authors prefer working in the cloud. This allows us to gain access to our files from any Internetconnected computer, tablet, or smartphone.

Similarly, our R session is saved to the cloud, allowing access from any of several computers at home or office computers.

—----------------------------------------------------------------

Para ver el directorio de trabajo actual, escriba el comando getwd(). Puede cambiar el directorio de trabajo escribiendo setwd(), pero generalmente me resulta más fácil usar el menú Archivo. Simplemente seleccione Archivo > Cambiar directorio... y navegue hasta el directorio que desea convertir en el nuevo directorio de trabajo. Como puede ver en la lista de códigos aquí, los autores prefieren trabajar en la nube. Esto nos permite obtener acceso a nuestros archivos desde cualquier computadora, tableta o teléfono inteligente conectado a Internet. De manera similar, nuestra sesión de R se guarda en la nube, lo que permite el acceso desde cualquiera de varias computadoras en el hogar o en la oficina.

—---------------------------------------------------------------------

> getwd()

[1] "C:/Users/Joshua Wiley/Google Drive/Projects/Books/Apress\_BeginningR/BeginningR"

In addition to ls(), another helpful function is dir(), which will give you a list of the files in your current working directory.

—----------------------------------------------

Además de ls(), otra función útil es dir(), que le dará una lista de los archivos en su directorio de trabajo actual.

—---------------------------------------------

To quit your R session, simply type **q()** at the command prompt, or if you like to use the mouse, select File > Exit or simply close Rstudio by clicking on the X in the upper right corner. In any of these cases, you will be prompted to save your R workspace.

—----------------------------------------

Para salir de su sesión de R, simplemente escriba q() en el símbolo del sistema, o si desea usar el mouse, seleccione Archivo > Salir o simplemente cierre Rstudio haciendo clic en la X en la esquina superior derecha. En cualquiera de estos casos, se le pedirá que guarde su espacio de trabajo de R.

—----------------------------------------

Go ahead and quit the current R session, and save your workspace when prompted. We will come back to the same session in a few minutes. What was going on in the background while we played with R was that R was recording everything you typed in the console and everything it wrote back to the console. This is saved in an R history file. When you save your R session in an RData file, it contains this particular workspace. When you find that file and open it, your previous workspace will be restored. This will keep you from having to reenter your variables, data, and functions.

—-----------------------------------------------

Continúe y salga de la sesión R actual y guarde su espacio de trabajo cuando se le solicite. Volveremos a la misma sesión en unos minutos. Lo que sucedía en segundo plano mientras jugábamos con R era que R grababa todo lo que escribías en la consola y todo lo que escribía en la consola. Esto se guarda en un archivo de historial de R. Cuando guarda su sesión R en un archivo RData, contiene este espacio de trabajo en particular. Cuando encuentre ese archivo y lo abra, se restaurará su espacio de trabajo anterior. Esto evitará que tenga que volver a ingresar sus variables, datos y funciones.

—-------------------------------------------------

Before we go back to our R session, let’s see how to use R for some mathematical operators and functions (see Table 1-1). These operators are vectorized, so they will apply to either single numbers or vectors with more than one number, as we will discuss in more detail later in this chapter. According to the R documentation, these are “unary and binary generic functions” that operate on numeric and complex vectors, or vectors that can be coerced to numbers. For example, logical vectors of TRUE and FALSE are coerced to integer vectors, with TRUE = 1 and FALSE = 0.

—-------------------------------------------------

Antes de volver a nuestra sesión de R, veamos cómo usar R para algunos operadores y funciones matemáticos (consulte la Tabla 1-1). Estos operadores están vectorizados, por lo que se aplicarán a números únicos o vectores con más de un número, como veremos con más detalle más adelante en este capítulo. De acuerdo con la documentación de R, estas son "funciones genéricas unarias y binarias" que operan en vectores numéricos y complejos, o vectores que pueden convertirse en números. Por ejemplo, los vectores lógicos de VERDADERO y FALSO se convierten en vectores enteros, con VERDADERO = 1 y FALSO = 0.

—------------------------------------------------

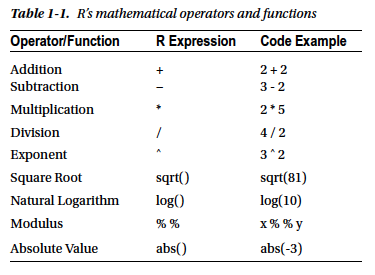


Table 1-2 shows R’s comparison operators. Each of these evaluates to a logical result of

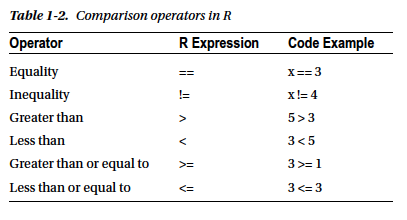
TRUE or FALSE.

We can abbreviate TRUE and FALSE as T and F, so it would be unwise to name a variable T or F, although R will let you do that. Note that the equality operator == is different from the = used as an assignment operator. As with the mathematical operators and the logical operators (see Chapter 4), these are also vectorized.

—-------------------------------------------

Podemos abreviar VERDADERO y FALSO como V y F, por lo que sería imprudente nombrar una variable T o F, aunque R te permitirá hacerlo. Tenga en cuenta que el operador de igualdad == es diferente del = utilizado como operador de asignación. Al igual que con los operadores matemáticos y los operadores lógicos (ver Capítulo 4), estos también están vectorizados.

—---------------------------------------------------



R has six “atomic” vector types (meaning that they cannot be broken down any further), including logical, integer, real, complex, string (or character), and raw.

R tiene seis tipos de vectores "atómicos" (lo que significa que no se pueden desglosar más), incluidos lógicos, enteros, reales, complejos, cadenas (o caracteres) y sin formato.

Vectors must contain only one type of data, but lists can contain any combination of data types. A data frame is a special kind of list and the most common data object for statistical analysis. Like any list, a data frame can contain both numerical and character information. Some character information can be used for factors. Working with factors can be a bit tricky because they are “like” vectors to some extent, but they are not exactly vectors.

—---------------------------------------------------------

R tiene seis tipos de vectores "atómicos" (lo que significa que no se pueden desglosar más), incluidos lógicos, enteros, reales, complejos, cadenas (o caracteres) y sin formato. Los vectores deben contener solo un tipo de datos, pero las listas pueden contener cualquier combinación de tipos de datos.

Un marco de datos es un tipo especial de lista y el objeto de datos más común para el análisis estadístico. Como cualquier lista, un marco de datos puede contener información tanto numérica como de caracteres. Parte de la información de los caracteres se puede utilizar para los factores. Trabajar con factores puede ser un poco complicado porque son vectores "similares" hasta cierto punto, pero no son exactamente vectores.

—---------------------------------------------------------------

My friends who are programmers who dabble in statistics think factors are evil, while statisticians like me who dabble in programming love the fact that character strings can be used as factors in R, because such factors communicate group membership directly rather than indirectly. It makes more sense to have a column in a data frame labeled sex with two entries, male and female, than it does to have a column labeled sex with 0s and 1s in the data frame. If you like using 1s and 0s for factors, then use a scheme such as labeling the column female and entering a 1 for a woman and 0 for a man. That way the 1 conveys meaning, as does the 0. Note that some statistical software programs such as SPSS do not uniformly support the use of strings as factors, whereas others, for example, Minitab, do.

—------------------------------------------------

Mis amigos que son programadores que incursionan en las estadísticas piensan que los factores son malos, mientras que a los estadísticos como yo que incursionan en la programación les encanta el hecho de que las cadenas de caracteres se pueden usar como factores en R, porque dichos factores comunican la pertenencia a un grupo directamente en lugar de indirectamente. Tiene más sentido tener una columna en un marco de datos etiquetada como sexo con dos entradas, masculino y femenino, que tener una columna etiquetada como sexo con 0 y 1 en el marco de datos. Si le gusta usar 1 y 0 para los factores, use un esquema como etiquetar la columna como mujer e ingresar un 1 para una mujer y un 0 para un hombre. De esa manera, el 1 transmite significado, al igual que el 0. Tenga en cuenta que algunos programas de software estadístico como SPSS no admiten uniformemente el uso de cadenas como factores, mientras que otros, por ejemplo, Minitab, sí lo hacen.

—----------------------------------------------

In addition to vectors, lists, and data frames, R has language objects including calls, expressions, and names. There are symbol objects and function objects, as well as expression objects. There is also a special object called NULL, which is used to indicate that an object is absent. Missing data in R are indicated by NA, which is also a valid logical object.

—---------------------------------------------------------

Además de vectores, listas y marcos de datos, R tiene objetos de lenguaje que incluyen llamadas, expresiones y nombres. Hay objetos de símbolo y objetos de función, así como objetos de expresión. También hay un objeto especial llamado NULL, que se usa para indicar que un objeto está ausente. Los datos que faltan en R se indican mediante NA, que también es un objeto lógico válido.

—--------------------------------------------------------

1.3 Your Second R Session

Reopen your saved R session by navigating to the saved workspace and launching it in R. We will put R through some more paces now that you have a better understanding of its data types and its operators, functions, and “constants.” If you did not save the session previously, you can just start over and type in the missing information again. You will not need the list with your name and data, but you will need the x, y, and z variables we created earlier.

—--------------------------------------

Vuelva a abrir su sesión de R guardada navegando al espacio de trabajo guardado e iniciándolo en R. Pondremos a R a prueba un poco más ahora que tiene una mejor comprensión de sus tipos de datos y sus operadores, funciones y "constantes". Si no guardó la sesión anteriormente, puede comenzar de nuevo y escribir la información que falta nuevamente. No necesitará la lista con su nombre y datos, pero necesitará las variables x, y y z que creamos anteriormente.

—---------------------------------------

As you have learned, R treats a single number as a vector of length 1. If you create a vector of two or more objects, the vector must contain only a single data type. If you try to make a vector with multiple data types, R will coerce the vector into a single type.

—-------------------------------------------------

Como ha aprendido, R trata un solo número como un vector de longitud 1. Si crea un vector de dos o más objetos, el vector debe contener un solo tipo de datos. Si intenta crear un vector con múltiples tipos de datos, R convertirá el vector en un solo tipo.

—---------------------------------------------------

1.3.1 Working with Indexes

Trabajar con índices

R’s indexing is quite flexible. We can use it to add elements to a vector, to substitute new values for old ones, and to delete elements of the vector. We can also subset a vector by using a range of indexes. As an example, let’s return to our x vector and make some adjustments:

—---------------------------------------------

—--------------------------------------------------

> x [ 1 ] 1 2 3 4 5 6 7 8 9 1 0

> x [ 1 ] [ 1 ] 1

> x [ 2 : 4 ] [ 1 ] 2 3 4

> x [ - 1 0 ] [ 1 ] 1 2 3 4 5 6 7 8 9 > x [ 1 ] 1 2 3 4 5 6 7 8 9 1 0 > x [ 2 : 4 ] < - c ( 9 8 , 99 , 1 0 0 ) > x [ 1 ] 1 9 8 9 9 1 0 0 5 6 7 8 9 1 0

Note that if you simply ask for subsets, the x vector is not changed, but if you reassign the subset or modified vector, the changes are saved. Observe that the negative index removes the selected element or elements from the vector but only changes the vector if you reassign the new vector to x. We can, if we choose, give names to the elements of a vector, as this example shows:

> x < - 1 : 1 0 > x [ 1 ] 1 2 3 4 5 6 7 8 9 1 0 > n a m e s ( x ) < - c ( " A " , " B " , " C " , " D " , " E " , " F " , " G " , " H " , " I " , " J " ) > x A B C D E F G H I J 1 2 3 4 5 6 7 8 9 1 0

This showcases the difference between thinking as a user versus thinking as a programmer! R has a variety of built-in functions that automate even the simplest kind of operations. You just saw me waste our time by typing in the letters A through J. R already knows the alphabet, and all you have to do is tell R you want the first 10 letters. The more you know about R, the easier it is to work with, because it keeps you from having to do a great deal of repetition in your programming. Take a look at what happens when we ask R for the letters of the alphabet and use the power of built-in character manipulation functions to make something a reproducible snippet of code. Everyone starts as an R user and (ideally) becomes an R programmer, as discussed in the introduction:

> x [ 1 ] 1 2 3 4 5 6 7 8 9 1 0 > n a m e s ( x ) < - t o u p p e r ( l e t t e r s [ 1 : 1 0 ] ) > x A B C D E F G H I J 1 2 3 4 5 6 7 8 9 1 0

The toupper function coerces the letters to uppercase, and the letters[1:10] subset gives us A through J. Always think like a programmer rather than a user. If you wonder if something is possible, someone else has probably thought the same thing. Over two million people are using R right now, and many of those people write R functions and code that automates the things that we use on such a regular basis that we usually don’t even have to wonder whether but simply need to ask where they are and how to use them. You can find many examples of efficient R code on the web, and the discussions on StackExchange are very helpful.

If you are trying to figure something out that you don’t know how to do, don’t waste much time experimenting. Use a web search engine, and you are very likely to find that someone else has already found the solution, and has posted a helpful example you can use or modify for your own problem. The R manual is also helpful, but only if you already have a strong programming background. Otherwise, it reads pretty much like a technical manual on your new toaster written in a foreign language.

It is better to develop good habits in the beginning than it is to develop bad habits and then having to break them first before you can learn good ones. This is what Dr. Lynda McCalman calls a BFO. That means a blinding flash of the obvious. I have had many of those in my experience with R.

1.3.2 Representing Missing Data in R

Now let’s see how R handles missing data. Create a simple vector using the c() function (some people say it means combine, while others say it means concatenate ). I prefer combine because there is also a cat() function for concatenating output. For now, just type in the following and observe the results. The built-in function for the mean returns NA because of the missing data value. The na.rm = TRUE argument does not remove the missing value but simply omits it from the calculations. Not every built-in function includes the na.rm option, but it is something you can program into your own functions if you like. We will discuss functional programming in Chapter 5, in which I will show you how to create your own custom function to handle missing data. We will add a missing value by entering NA as an element of our vector. NA is a legitimate logical character, so R will allow you to add it to a numeric vector:

> w < - c ( 1 0 , NA , 10 , 25 , 30 , 15 , 10 , 18 , 16 , 1 5 ) > w [ 1 ] 1 0 N A 1 0 2 5 3 0 1 5 1 0 1 8 1 6 1 5 > m e a n ( w ) [ 1 ] N A > m e a n ( w , n a . r m = T R U E ) [ 1 ] 1 6 . 5 5 5 5 6

Observe that the mean is calculated when you omit the missing value, but unless you were to use the command w <- w[-2], the vector will not change.

1.3.3 Vectors and Vectorization in R

Remember vectors must contain data elements of the same type. To demonstrate this, let us make a vector of 10 numbers, and then add a character element to the vector. R coerces the data to a character vector because we added a character object to it. I used the index [11] to add the character element to the vector. But the vector now contains characters and you cannot do math on it. You can use a negative index, [-11], to remove the character and the R function as.integer() to coerce the vector back to integers.

To determine the structure of a data object in R, you can use the str() function. You can also check to see if our modified vector is integer again, which it is:

> x < - 1 : 1 0 > x [ 1 1 ] < - " A " > x [ 1 ] " 1 " " 2 " " 3 " " 4 " " 5 " " 6 " " 7 " " 8 " " 9 " " 1 0 " " A " > s t r ( x ) c h r [ 1 : 1 1 ] " 1 " " 2 " " 3 " " 4 " " 5 " " 6 " " 7 " " 8 " " 9 " " 1 0 " . . . > i s . c h a r a c t e r ( x ) [ 1 ] T R U E > x < - x [ - 1 1 ] > x < - a s . i n t e g e r ( x ) > i s . i n t e g e r ( x ) [ 1 ] T R U E

Add y to x as follows. See that R recycles y for each value of x, so that the addition operation results in a new vector. No explicit looping was required:

> x + y [ 1 ] 6 7 8 9 1 0 1 1 1 2 1 3 1 4 1 5

The way vectorization works when you use operations with two vectors of unequal length is that the shorter vector is recycled. If the larger vector’s length is a multiple of the length of the shorter vector, this will produce the expected result. When the length of the longer vector is not an exact multiple of the shorter vector’s length, the shorter vector is recycled until R reaches the end of the longer vector. This can produce unusual results. For example, divide z by x. Remember that z has 33 elements and x has 10:

> z [ 1 ] 8 1 . 5 6 7 0 . 8 5 7 7 . 4 8 6 4 . 0 2 6 8 . 9 4 8 0 . 2 4 6 0 . 8 4 7 0 . 9 3 7 5 . 2 1 7 5 . 0 5 5 2 . 1 7 5 2 . 2 9 [ 1 3 ] 7 0 . 2 0 7 9 . 2 9 8 4 . 7 5 6 4 . 8 8 7 3 . 7 4 7 1 . 1 9 6 1 . 0 1 6 3 . 4 3 5 5 . 7 4 7 1 . 5 4 6 9 . 7 1 8 2 . 5 2 [ 2 5 ] 7 3 . 4 0 7 5 . 3 9 7 9 . 2 8 8 0 . 3 6 6 5 . 7 9 7 3 . 1 5 7 5 . 4 1 6 9 . 5 6 8 5 . 8 7 > x [ 1 ] 1 2 3 4 5 6 7 8 9 1 0 > r o u n d ( z / x , 2 ) [ 1 ] 8 1 . 5 6 3 5 . 4 2 2 5 . 8 3 1 6 . 0 0 1 3 . 7 9 1 3 . 3 7 8 . 6 9 8 . 8 7 8 . 3 6 7 . 5 0 5 2 . 1 7 2 6 . 1 4 [ 1 3 ] 2 3 . 4 0 1 9 . 8 2 1 6 . 9 5 1 0 . 8 1 1 0 . 5 3 8 . 9 0 6 . 7 8 6 . 3 4 5 5 . 7 4 3 5 . 7 7 2 3 . 2 4 2 0 . 6 3 [ 2 5 ] 1 4 . 6 8 1 2 . 5 6 1 1 . 3 3 1 0 . 0 4 7 . 3 1 7 . 3 2 7 5 . 4 1 3 4 . 7 8 2 8 . 6 2 W a r n i n g m e s s a g e : I n z / x : l o n g e r o b j e c t l e n g t h i s n o t a m u l t i p l e o f s h o r t e r o b j e c t l e n g t h

R recycled the x vector three times, and then divided the last three elements of z by 1, 2, and 3, respectively. Although R gave us a warning, it still performed the requested operation.

1.3.4 A Brief Introduction to Matrices

Matrices are vectors with dimensions. We can build matrices from vectors by using the cbind() or rbind() functions. Matrices have rows and columns, so we have two indexes for each cell of the matrix. Let’s discuss matrices briefly before we create our first matrix and do some matrix manipulations with it.

A matrix is an m × n (row by column) rectangle of numbers. When n = m, the matrix is said to be “square.” Square matrices can be symmetric or asymmetric. The diagonal of a square matrix is the set of elements going from the upper left corner to the lower right corner of the matrix. If the off-diagonal elements of a square matrix are the same above and below the diagonal, as in a correlation matrix, the square matrix is symmetric.

A vector (or array) is a 1-by-n or an n-by-1 matrix, but not so in R, as you will soon see. In statistics, we most often work with symmetric square matrices such as correlation and variance-covariance matrices. An entire matrix is represented by a boldface letter, such as A:

A a a a a a a a a a m n n n m m m n , , , , , , , , , , = × × × é ë ê ê ê ê ê ù û ú 1 1 1 2 1 2 1 2 2 2 1 2 … … … ú ú ú ú

Matrix manipulations are quite easy in R. If you have studied matrix algebra, the following examples will make more sense to you, but if you have not, you can learn enough from these examples and your own self-study to get up to speed quickly should your work require matrices.

Some of the most common matrix manipulations are transposition, addition and subtraction, and multiplication. Matrix multiplication is the most important operation for statistics. We can also find the determinant of a square matrix, and the inverse of a square matrix with a nonzero determinant.

You may have noticed that I did not mention division. In matrix algebra, we write the following, where B−1 is the inverse of B. This is the matrix algebraic analog of division (if you talk to a mathematician, s/he would tell you this is how regular ‘division’ works as well. My best advice, much like giving a mouse a cookie, is don’t):

AB C A B C = = -1 (1, 1)

We define the inverse of a square matrix as follows. Given two square matrices, A and B, if AB = I, the identity matrix with 1s on the diagonals and 0s on the off-diagonals, then B is the right-inverse of A, and can be represented as A−1 . With this background behind us, let’s go ahead and use some of R’s matrix operators. A difficulty in the real world is that some matrices cannot be inverted. For example, a so-called singular matrix has no inverse. Let’s start with a simple correlation matrix:

A = é ë ê ê ê ê ù û ú ú ú ú 1 00 0 14 0 35 0 14 1 00 0 09 0 35 0 98 1 00 . . .. . . . .. . . . .. .

In R, we can create the matrix first as a vector, and then give the vector the dimensions 3 × 3, thus turning it into a matrix. Note the way we do this to avoid duplicating A; for very large data, this may be more compute efficient. The is.matrix(X) function will return TRUE if X has these attributes, and FALSE otherwise. You can coerce a data frame to a matrix by using the as.matrix function, but be aware that this method will produce a character matrix if there are any nonnumeric columns. We will never use anything but numbers in matrices in this book. When we have character data, we will use lists and data frames:

> A <- c(1.00, 0.14, 0.35, 0.14, 1.00, 0.09, 0.35, 0.09, 1.00) > dim(A)<-c(3,3)

> A [,1] [,2] [,3] [1,] 1.00 0.14 0.35 [2,] 0.14 1.00 0.09 [3,] 0.35 0.09 1.00 > d i m ( A ) [ 1 ] 3 3

Some useful matrix operators in R are displayed in Table 1-3. Table 1-3. Matrix operators in R

Operator Operator Code Example Transposition t t(A) Matrix Multiplication %\*% A %\*% B Inversion solve() solve(A)

Because the correlation matrix is square and symmetric, its transpose is the same as A. The inverse multiplied by the original matrix should give us the identity matrix. The matrix inversion algorithm accumulates some degree of rounding error, but not very much at all, and the matrix product of A−1 and A is the identity matrix, which rounding makes apparent:

> A i n v < - s o l v e ( A ) > m a t P r o d < - A i n v % \* % A > r o u n d ( m a t P r o d ) [ , 1 ] [ , 2 ] [ , 3 ] [ 1 , ] 1 0 0 [ 2 , ] 0 1 0 [ 3 , ] 0 0 1

If A has an inverse, you can either premultiply or postmultiply A by A−1 and you will get an identity matrix in either case.

1.3.5 More on Lists

Recall our first R session in which you created a list with your name and alma mater. Lists are unusual in a couple of ways, and are very helpful when we have “ragged” data arrays in which the variables have unequal numbers of observations. For example, assume that my coauthor, Dr Pace, taught three sections of the same statistics course, each of which had a different number of students. The final grades might look like the following:

> section1 <- c(57.3, 70.6, 73.9, 61.4, 63.0, 66.6, 74.8, 71.8, 63.2, 72.3, 61.9, 70.0) > section2 <- c(74.6, 74.5, 75.9, 77.4, 79.6, 70.2, 67.5, 75.5, 68.2, 81.0, 69.6, 75.6, 69.5, 72.4, 77.1) > section3 <- c(80.5, 79.2, 83.6, 74.9, 81.9, 80.3, 79.5, 77.3, 92.7, 76.4, 82.0, 68.9, 77.6, 74.6) > allSections <- list(section1,section2,section3) > allSections [[1]] [1] 57.3 70.6 73.9 61.4 63.0 66.6 74.8 71.8 63.2 72.3 61.9 70.0 [[2]] [1] 74.6 74.5 75.9 77.4 79.6 70.2 67.5 75.5 68.2 81.0 69.6 75.6 69.5 72.4 77.1 [[3]] [1] 80.5 79.2 83.6 74.9 81.9 80.3 79.5 77.3 92.7 76.4 82.0 68.9 77.6 74.6 > section\_means <- sapply(allSections, mean) > round(section\_means, 2) [1] 67.23 73.91 79.24 > section\_sdev <- sapply(allSections, sd) > round(section\_sdev,2) [1] 5.74 4.17 5.40

We combined the three classes into a list and then used the sapply function to find the means and standard deviations for the three classes. As with the name and address data, the list uses two square brackets for indexing. The [[1]] indicates the first element of the list, which is a number contained in another list. The sapply function produces a simplified view of the means and standard deviations. Note that the lapply function works here as well, as the calculation of the variances for the separate sections shows, but produces a different kind of output from that of sapply, making it clear that the output is yet another list:

> lapply(allSections,var) [[1]] [1] 32.99515 [[2]] [1] 17.3521 [[3]] [1] 29.18879

1.3.6 A Quick Introduction to Data Frames

As I mentioned earlier, the most common data structure for statistics is the data frame. A data frame is a list, but rectangular like a matrix. Every column represents a variable or a factor in the dataset. Every row in the data frame represents a case, either an object or an individual about whom data have been collected, so that, ideally, each case will have a score for every variable and a level for every factor. Of course, as we will discuss in more detail in Chapter 2, real data are far from ideal.

Here is the roster of the 2014-2015 Clemson University mens’ basketball team, which I downloaded from the university’s website. I saved the roster as a comma-separated value (CSV) file and then read it into R using the read.csv function. Please note that in this case, the file ‘roster.csv’ was saved in our working directory. Recall that earlier we discussed both getwd() and setwd(), these can be quite helpful. As you can see, when you create data using this method, the file will automatically become a data frame in R:

> r o s t e r < - r e a d . c s v ( " r o s t e r . c s v " ) > r o s t e r Jersey Name Position Inches Pounds Class 1 0 R o o k s , P a t r i c k G 7 4 1 9 0 f r e s h m a n 2 1 A j u k w a , A u s t i n G 7 8 2 0 5 s o p h o m o r e 3 3 H o l m e s , A v r y G 7 4 2 0 5 j u n i o r 4 5 B l o s s o m g a m e , J a r o n F 7 9 2 1 5 s o p h o m o r e 5 1 0 D e V o e , G a b e G 7 5 2 0 0 f r e s h m a n 6 1 2 H a l l , R o d G 7 3 2 0 5 s e n i o r 7 1 5 G r a n t h a m , D o n t e F 8 0 2 0 5 f r e s h m a n 8 2 0 R o p e r , J o r d a n G 7 2 1 6 5 j u n i o r 9 2 1 H a r r i s o n , D a m a r c u s G 7 6 2 0 5 s e n i o r (conti

> s t r ( r o s t e r ) ’ d a t a . f r a m e ’ : 1 3 o b s . o f 6 v a r i a b l e s : $ J e r s e y : i n t 0 1 3 5 1 0 1 2 1 5 2 0 2 1 3 3 . . . $ N a m e : F a c t o r w / 1 3 l e v e l s " A j u k w a , A u s t i n " , . . : 1 1 1 8 2 3 6 5 1 2 7 1 3 . . . $ P o s i t i o n : F a c t o r w / 3 l e v e l s " C " , " F " , " G " : 3 3 3 2 3 3 2 3 3 2 . . . $ I n c h e s : i n t 7 4 7 8 7 4 7 9 7 5 7 3 8 0 7 2 7 6 8 0 . . . $ P o u n d s : i n t 1 9 0 2 0 5 2 0 5 2 1 5 2 0 0 2 0 5 2 0 5 1 6 5 2 0 5 2 4 5 . . . $ C l a s s : F a c t o r w / 4 l e v e l s " f r e s h m a n " , " j u n i o r " , . . : 1 4 2 4 1 3 1 2 3 2 . . .