THE EXPERT'S VOICE® IN R

R Quick Syntax Reference

A QUICK, HANDY GUIDE TO USING R

Margot Tollefson

Apress®

For your convenience Apress has placed some of the front matter material after the index. Please use the Bookmarks and Contents at a Glance links to access them.

Apress®

Contents at a Glance

Introduction

R is a programming language that provides the user with powerful data and graphical analysis options. R is both flexible and broad. From tasks as simple as adding two numbers to tasks as complex as fitting an ARIMA model, R is capable of crunching the numbers.

The purpose of *R Quick Syntax Reference* is to provide the reader with the basic syntax of R. Often an R user gets stuck if, for example, a mode is incorrect or a logical test does not work. Because the full spectrum of R packages uses the same fairly simple syntax, *R Quick Syntax Reference* provides the reader with the necessary information to get unstuck and run and create all R functions and code.

The R language is based on the language S, a high-level programming language developed mainly by Richard A. Becker, John M. Chambers, and Allan R. Wilks in the AT&T laboratories in 1975. The R version of the language first became available in 1993 and was developed by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand.

R is open source and is a GNU project. As open-source code, the R language is free and constantly being improved. The R Development Core Team currently does the development. Packages for specific analysis techniques are added often. At the present time, there are 4,986 packages available in R. Most users will use only a few packages. Although GUI versions of R are available, we discuss using R at the command prompt in *R Quick Syntax Reference*.

This book is about the S3 version of R—S3 standing for the third version of S, the commercial program on which R is based. The developers of R have a new version, S4—the fourth version of S—running concurrently with S3. Even though version S4 is quite different from S3, it is necessary to know the syntax of S3 in order to use S4. And S3 remains a powerful, flexible language in its own right—hence, this book.

Part I covers the basics of R. Chapter 1 describes how to download and install R for the Windows, Mac, and Linux operating systems and also how to download packages. Because keeping separate folders for different projects is very useful, Chapter 1 gives instructions for running R from different folders. It also gives the methods for updating the R program itself.

Chapter 2 introduces the R prompt, gives a sample calculation, and describes the three parts of R—objects, operators, and assignments. Chapter 3 covers the assignment of names to objects, demonstrates the `ls()` function that allows you to see the objects in a folder, and discusses the operators in R.

Part II describes R objects. Objects have modes, classes, and types. Chapter 4 lists the modes and describes some of them. It also shows how modes and types differ. Chapter 5 discusses some of the classes.

Part III covers functions. Chapter 6 starts with a list of the 30 default packages in R and follows with instructions on how to use functions. Because packaged functions all have help pages, the chapter provides instructions on how to access and use the help page of a function. Chapter 7 describes how to create a function. Chapter 8 explains how to run a function—with a detailed approach to the argument list.

Part IV focuses on importing and exporting data in R and methods for creating and manipulating some kinds of object. Chapter 9 describes several methods for importing data, gives a number of functions to create data objects, and discusses some random-number generators. Chapter 10 gives several methods for exporting from R. Chapter 11 gives a number of functions that operate on objects—to bind objects together, to find descriptive qualities of an object, to assign qualities to an object, to aggregate an object in some way, or to apply functions to portions of an object.

Part V covers flow conditioning commands and functions. Chapter 12 presents the flow conditioning statements, and Chapter 13 supplies examples of them. Chapter 14 describes the two flow conditioning functions and gives examples.

Part VI discusses functions related to formatting and outputting output, looks at the results from packaged functions and at what some of the default packages contain, and provides some tips for using R. Chapter 15 gives some rounding functions and some functions for outputting from a function. It also gives some functions that vary according to the class of the object on which the function operates and that summarize the results of the function, either textually or visually. Chapter 16 takes a look at the contents of the packages **base**, **stats**, and **graphics** and glances at the **datasets**, **grDevices**, **methods**, and utils packages. Chapter 17 describes how to deal with some common frustrations in R. More information is given on outputting from functions, plus an example of a recursive function and some advice on using R.

PART 1

R Basics

Part I introduces you to the basics of the R language.

To use R, you must first download the program from the Internet. Chapter 1 describes how to install R on the Windows, OS X, and Linux operating systems. It also describes how to install and update packages and how to update R and use R within a file system.

Once you've installed and opened R, you are faced with an R prompt and little else. Chapter 2 presents the parts of R (objects, operators, and assignments), the R prompt, and an example of using R as a calculator from the R prompt.

Chapter 3 shows you how to assign names to expressions to create R objects and describes two functions: ls() for listing the objects in the workspace and rm() for removing objects from the workspace. It then discusses the operators that operate on objects and expressions: logical, arithmetic, matrix, relational, and subscripting operators, plus a few other special operators.

CHAPTER 1

Downloading R and Setting Up a File System

The first step in using R is to download R from the Internet. R can be downloaded for the modern operating systems Windows, OS X, and Linux. In this chapter, you will learn how to download and install R and the 30 basic packages as well as how to install other packages and update R. You will also learn how to use R in individual folders within the file system of the computer.

Downloading R

You can download R from the web site of the Comprehensive R Archive Network (CRAN). CRAN updates the installation process from time to time; however, the instructions in this book are for the current steps at time of publication. CRAN provides instructions on the web site if the process has changed.

Begin the download process by going to the web site `http://cran.r-project.org`. At the web site, links to current versions for Windows, OS X, and Linux are listed at the top of the opening window. Select the appropriate link.

Windows

On the page that opens with the Windows link, select the link **base**, which is the top link. In the next window, click on the **download** link for the given Windows version. (Currently, the link is **Download R 3.0.2 for Windows**.) If R has not already been installed on the computer, the downloader will create a default folder in the **Documents** folder to hold R files. Unless there is a reason to change the folder name or location, accept the default. R will begin to download.

When the program finishes downloading, find the downloaded file in your file system. Downloads are put in `C://Users/User_folder/Downloads`, where **User_folder** is the folder of the user, unless another folder was specified earlier in the installation. Click on the downloaded file, which is an `.exe` installation file (currently `R-3.0.2-win.exe`.) A question about the safety of the program may pop up. The installation program is safe, so run the program.

The installation wizard will open. The installation process steps through several pages. On the first page, read the GNU GENERAL PUBLIC LICENSE; then click on **Next**. For the rest of the pages, accepting the defaults on each page is fine, so click on **Next** on each page.

At the page of additional choices, click on **Next**, and the program will begin to install. When the installation is finished, click on **Finish** to complete the installation. The program and the 30 base packages are now installed. An icon for R will be on the computer desktop and, for Windows 8, in the charms. To run R, click on the icon or charm.

OS X

On the page that opens from the OS X link, first read the section under **R for Mac OS X**. The R project gives the advice to check the files for viruses and other problems.

Under **Files:** there are two package choices: the current version and latest version. Selecting the current version (the `.pkg` link, currently `R-3.0.2.pkg`) will download both packages. When the packages have finished downloading, open the download box on the icon bar (the yellow and brown box) or the downloads folder under the username in **Finder**.

Select an R version in the download box. Opening the version will open the installer. With the installer open, click on **Continue** to go to the next page of the installer. Read the message from CRAN; then click **Continue**. Again, read the message from CRAN; then click **Continue**.

On the next page, you will find the license. After reading the license, click **Agree** to download R. On the next page, select either of the choices; then click on **Continue**. (The **Continue** button will not light up until a choice is made.)

On the next page, select **Install**. The installation program will ask for a password. After you have entered a password, the installation will begin. When the installation is finished, click on **Close**. R will now be in the applications folder and on the dock and the 30 base packages will be loaded. Select **R** on the dock or in the applications folder to start R.

Linux

At the CRAN site, CRAN provides source code for R for the Linux distributions Debian, Red Hat, Suse, and Ubuntu. The developers state that R is available through the package management system for most distributions of Linux.

If the command line version of R is not available using the package management system, installing R directly from the terminal is an option. At `http://cran.r-project.org/bin/linux/distribution`, where *distribution* is Debian, Suse, or Ubuntu, you can find instructions for installing R from the terminal command prompt under the `ReadMe` files.

For Red Hat, `http://cran.r-project.org/bin/linux/redhat`, there is no `ReadMe` file. Follow instructions on the CRAN site to install R for Red Hat. Once you have installed R, command line R will be available by typing **R** in the terminal window.

Installing and Updating Packages

When initially installed, R comes with 30 packages. Often the user will want to use the power of the many other packages available in R. Installing and updating a package is straightforward.

For any of the operating systems, if the name of a package is known, typing

```
install.packages("package name")
```

at the R command prompt, where *package name* is the name of the package, will install the package. To update packages, typing

```
update.packages()
```

at the R command prompt will find those packages with updates and update the packages. To see which packages are already installed on the computer, enter

```
installed.packages()
```

at the R prompt.

If the name of the package is not known (also for known names), using the installer for the operating systems Windows and OS X is easy. For Linux, instructions can be found at the CRAN web site, http://cran.r-project.org. Here you can find instructions for Windows and OS X.

Windows

To install a package in Windows not using the command line, start by opening R. On the menu bar at the top of the screen, select **Packages**. A menu will drop down. Select **Install package (s)...** . Either the CRAN mirror window or the Packages window will come up. If the CRAN mirror window comes up, select a close mirror and click **OK**, which will bring up the Packages window.

The Packages window consists of a list of all of the available packages. Scroll down the list to find the package(s) you wish to install and select the package(s). Click on **OK** to begin the installation. As the installation proceeds, the steps of the installation will scroll on the R console. When the R prompt returns to the screen, the installation is complete.

To update packages not using the command line, select **Packages** on the menu bar and then select **Update packages...** . The Packages window to be updated will open, and it will have a list of all of the installed packages with updates. If there are none, the window will be empty. Choose the packages for updating and click on the **OK** button. If a question about using a personal library pops up, choose **Yes**. The packages will update. When the R prompt returns to the screen, the updates are complete.

OS X

To install packages in OS X, start by opening R. On the menu bar at the top of the screen, select **Packages & Data**. From the drop-down menu, select **Package Installer**, which brings up the R Package Installer. Click on **Get List** for a full list of packages or use the **Package Search** option to search for a package. Under either option, select the package(s) to be installed from the list.

Below the list of packages are choices for the location to put the packages. Hover over the list of location options for more information. Usually, one of the first two options will be correct. To the right of the location options are the **Install Selected** and **Update All** buttons. Before clicking on **Install Selected**, check the **Install Dependencies** box to make sure that any necessary packages are installed. Click on **Install Selected** to start the installation process. The selected packages will install.

To update packages, select **Packages & Data** from the menu bar at the top of the screen**.** From the drop-down menu, select **Package Installer**, which opens up the R Package Installer. At the bottom right of the Installer, select **Update All** and follow instructions.

Updating R

Since CRAN does not provide automatic updates for R, you must update it manually. The processes for Windows and OS X are easy. For the Linux distributions Debian, Suse, and Ubuntu, instructions can be found in the `ReadMe` files at `http://cran.r-project/bin/linux/distribution`, where *distribution* is either Debian, Suse, or Ubuntu. For Red Hat Linux, look elsewhere on the CRAN web site.

Windows

The first step in updating R in Windows is to open R and install the package **installr** if the package has not already been installed. Next, use the function **library** to provide access to **installr**. Type

```
library(installr)
```

at the command prompt and press **enter**. Then, to update R, type

```
updateR()
```

at the command prompt and press **enter**. R will either do an update or give a message that the program is up-to-date and return **False**.

Once **installr** has been installed, **installr** does not need to be installed again. The library must be accessed every time R is run.

OS X

The first step in updating R in OS X is to open R and select **R** from the menu bar at the top of the page. To run the updater, select **Check for R Updates** in the drop-down menu under **R** and follow instructions.

Using R in Separate Folders

Separate workspace images for R can be maintained in separate folders for Windows, OS X, and Linux. This property of R is very handy for using R on separate projects. While the process of opening R in a given folder varies by the operating system, once in a folder, saving the workspace image is straightforward. When closing an R session, the program asks if the user would like to save the workspace image. If **Yes** is selected, then `.RData` and `.Rhistory` (`.Rapp.history` for OS X) files are saved in the current directory. (For OS X, the files are hidden, but the files are there.)

The `.RData` file contains the objects that were in R at the beginning of the session plus any objects that were added during the session minus any objects that were erased during the session. The `.Rhistory` (`.Rapp.history` for OS X) file contains the history of the lines input at the R console. By default, all lines up to the last 512 lines are saved in Windows. For OS X and Linux, the default is 250 lines. Access to the lines carries over from session to session if the history is saved.

Windows

To initially set up R in a folder, open R at the desktop. (Click on the **R** icon on the desktop or click on **R** in the list of programs or, in Windows 8, the list of charms.) Select **File** on the menu bar at the top of the screen. From the drop-down menu, select **Change dir...**. The **Browse to folder** window will open. Navigate to the folder of choice.

When exiting R, save the workspace image and R will create `.RData` and `.Rhistory` files in the folder. The `.RData` file will have a blue **R** icon associated with the file. In the future, going to the folder and clicking on the **R** icon will open R and the history and objects saved within the folder will be present.

As a note for the initial setup, any objects in the desktop R will still be in R when the folder is changed. You can easily remove the objects. Type **`rm(list=ls())`** at the command prompt to remove all objects from the folder.

OS X

Working within different folders in OS X is also easy. There are two ways: dragging and dropping or using the terminal. If R is on the dock and R is not open, dragging the folder from either **Finder** or **Documents** to the **R** icon on the dock will open R in the folder using the `.RData` and `.Rapp.history` for that folder.

To open R using the terminal, open the terminal (located under **Applications**/**Utilities** in **Finder**.) and type

```
open -a R folder
```

where *folder* is the location of the folder. R will open in the folder using the `.RData` and `.Rapp.history` files for that folder.

Linux

To open R in a given folder in Linux, change the directory to the folder and type **R** at the command prompt.

CHAPTER 2

The R Prompt

This chapter covers the R prompt. It starts with descriptions of the three parts of R: objects, operators, and assignments. It continues with a discussion of working with the R prompt, followed by an example of doing a calculation at the R prompt.

In Windows and OS X, R runs in GUIs: *RGUI* in Windows and *R.app GUI* in OS X. Both RGUI and R.app GUI open an R Console and run from the R prompt in the R Console. GUIs are available in Linux, but this book covers only running R from the terminal window R prompt.

The Three Parts of R: Objects, Operators, and Assignments

There are basically three parts of R: objects, operators, and assignments.

> *Objects* contain information and can be data, functions, or the results of functions. Objects always have a name. Users create some objects, which are automatically saved on creation. Other objects are functions and datasets contained in the packages of R.
>
> *Operators* manipulate the objects, numbers, strings, and/or logical variables. For example, entering **a = 2*b** at the R prompt would multiply **b** by two and assign the result to **a**. The objects **a** and **b** are numeric objects and * is the multiplication operator. The equal sign makes an assignment of two times **b** to **a**.
>
> *Assignments* assign an expression to an object. *Expressions* consist of objects, numbers, logical variables, and/or strings, which are operated on by operators.

Expressions can be evaluated from the R prompt without an assignment. (The other places where assignments and operations occur are within functions and within flow control.)

The R Prompt

All of R flows from the R prompt. R is essentially the running of functions and the doing of calculations. Functions and calculations can be run at the R prompt with or without an assignment to an object. Functions and calculations can also be run as part of a function, but everything starts at the R prompt.

Using R from the R prompt may seem daunting at first. R opens with some script, and then a lonely little greater-than sign (>) is the R prompt. The opening script gives the R version number and some other information about the program, including the fact that the program runs with no warranty.

R remembers every line that is entered into the program, up to a set number of lines. A very handy side of R is that the up and down arrows on the keyboard will step through the lines. You only need to enter an expression once. Corrections to expressions are easy to do without typing the entire expression again.

To close R, enter **q()** at the R prompt or, for Windows and OS X, close the window. R will close with the option to save the workspace. In Linux, if the terminal window is closed without using **q()**, the current workspace will be lost.

The workspace consists of any objects present in R at the time the program is closed and the current history. Closing R without saving the workspace will result in reverting to the workspace present at the time the R session started.

An Example of a Calculation

The simplest use of R is as a calculator. The following calculation was done from the R prompt. There is no assignment in the calculation, so the result is returned on the screen.

```
> (1 + 3 + 7)/5
[1] 2.2
>
```

The first line gives the expression to be evaluated and the second line gives the result. The **[1]** in the second line is a label that tells the user that the result is the first value returned from the expression. Many expressions return more than one value. At the third line, the R prompt comes back and R is ready for another task.

CHAPTER 3

Assignments and Operators

R works with objects. Objects can include vectors, matrices, functions, the results from a function, or a number of other kinds of objects. Objects make working with information easier. This chapter covers assigning names to objects, listing and removing objects, and object operations. Part II (Chapters 4 and 5) covers the possible forms of objects.

Some objects come with the packages in R. Other objects are user-created. User-created objects have names that are assigned by the user. Knowing how to create, list, and remove user-created objects is basic to R.

Types of Assignment

Names in R must begin with a letter or a period, cannot have breaks, and can contain letters, numeric digits, periods, and underscores. The names that begin with a period are hidden and are used by R for startup defaults, the random seed, and other such things. The indexing symbols **[]**,**[[]]**,**$**, and **@** have special meanings with regard to R names, as explained the "Subscripting Operators" section of this chapter.

R originally used five types of assignment, four of which are still current. The four types are

```
a <- b,
```

which assigns **b** to **a**,

```
a -> b,
```

which assigns **a** to **b**,

```
a <<- b,
```

which assigns **b** to **a** and can be used inside a function to bring the assignment up to the workspace level, and

```
a ->> b,
```

which assigns **a** to **b** and brings an assignment in a function up to the workspace level.

Recently, the developers at R have included the more standard

```
a = b,
```

which assigns **b** to **a**. While any of the types of assignment can be used, the use of the equal sign is easiest to type.

When R makes an assignment, the name is automatically saved in the workspace. Note that no warning is given if the assigned name already exists. The assignment will overwrite the object in the workspace with the assigned object.

R is interesting in that a function of an object can be assigned to the original object. For example,

```
a = 2*a,
```

where the object **a** is replaced by the original **a** times two.

For more information about assignment operators, enter **?"Assignment Operators"** at the R prompt.

Example of Three Types of Assignment

An example of some of the types of assignment follows. Three objects are created: **abc**, **bcd**, and **cde**. You create the objects by assigning sequences to the objects. The sequences are generated when you put a colon between two integers, which creates a sequence of integers starting with the first integer and ending with the second integer.

To show that the objects actually contain the assigned sequence, the contents of the three objects are displayed below. Note that entering the name of an object at the R prompt will always display the contents of the object. The **[1]** refers to the first element of the objects.

```
> abc = 1:10

> abc
 [1]  1  2  3  4  5  6  7  8  9 10

> bcd <- 11:20

> bcd
 [1] 11 12 13 14 15 16 17 18 19 20

> 21:30 -> cde

> cde
 [1] 21 22 23 24 25 26 27 28 29 30
```

As you can see, the assignment operators <- and = give the same result. The assignment operator -> works in the opposite direction.

The ls() and rm() Functions

To see the objects present in the workspace, use the function `ls()`. Entering **ls()** at the R prompt for the above example gives

```
> ls()
[1] "abc" "bcd" "cde"
>,
```

which are the three objects created above.

Although functions are covered in detail in Part III, one interesting property of functions to note here is they can have arguments that the user enters. Two of the possible arguments for `ls()` are **pattern** and **all.names**.

The first argument is entered as **pattern** = **"a string"**, where *"a string"* is any part of an object name. For example, in the above workspace, searching for those objects containing **bc** in the name gives **abc** and **bcd**, that is

```
> ls(pattern="bc")
[1] "abc" "bcd"
```

The argument **pattern** can be reduced to **pat**, as in **ls(pat="bc")**. The shortening of arguments of functions is a property of R. All arguments in R can be reduced to the shortest unique form, but they are usually given in the full form in manuals.

The second argument is **all.names=**, which can equal **TRUE** or **FALSE**. If set to **TRUE**, the **all.names** argument instructs R to list all of the files in the workspace, including those that begin with a period. **FALSE** is the default value and does not need to be entered. For the example workspace above, setting **all.names** equal to **TRUE** gives

```
> ls(all.n=T)
[1] ".commander.done" ".First"          ".Random.seed"    ".Traceback"
[5] "abc"             "bcd"             "cde"
.
```

The **[1]** refers to ".commander.done" since "commander.done" is the first element of the vector, and the **[5]** refers to "abc" since "abc" is the fifth element of the vector. In R, if the elements of a vector have not been given a name, the convention for listing the elements is to show the index of the first element in each line of the lines of listed elements.

The function `rm()` can be used to remove objects from the workspace. For `rm()`, the names of the objects to be deleted are put within the parentheses and separated by commas. For example,

```
rm(a,b,c)
```

will remove objects **a**, **b**, and **c**. To remove all objects,

```
rm(list=ls())
```

works.

For more information about `ls()` or `rm()`, enter **?ls** or **?rm** at the R prompt.

Operators

Operators operate on objects. Operators can be logical, arithmetic, matrix, relational, or subscripting, or they may have a special meaning. Each of the types of operators is described here.

For operators, *elementwise* refers to performing the operation on each element of an object or paired elements for two objects. If two objects do not have the same dimensions, the operator will cycle the smaller object against the larger object. The cycling proceeds through each dimension. For example, for matrices the first dimension is the rows and the second dimension is the columns, so the cycling is down rows starting with the first column.

The letters **NA** are used to indicate that an element is missing data. Most operators have rules for dealing with missing data and may return an **NA** if data is missing.

CRAN gives a help page of information about operation precedence. Enter **??"Operator Syntax and Precedence"** at the R prompt to see the page.

Logical Operators and Functions

Logical operators return the values **TRUE**, **FALSE**, or **NA**, where **NA** refers to a missing value. The logical operators are the **not** operator, two **or** operators, two **and** operators, the **exclusive or** function (which is a function that acts as an operator), and the **any** function (which is a function that operates on a logical object). For logical operators, if the two objects do not have the same dimensions, the number of elements in the larger object must be a multiple of the number of elements in the smaller object for cycling to occur. The logical operators and two logical functions are listed in Table 3-1.

Table 3-1. *The Logical Operators and Functions*

Operator	Operation	Description
!	not	negation operator—e.g., !a
|	or	elementwise **or** operator—e.g., a|b
||	or	**or** operator, just evaluates the first elements in the objects—e.g., a||b
&	and	elementwise **and** operator—e.g., a&b
&&	and	**and** operator, just evaluates the first elements in the objects—e.g., a&&b
xor()	exclusive or	**exclusive or** function—e.g., xor(a,b)
any()	logical test	tests if **TRUE** is present in a logical object—e.g., any(a)

The logical operators operate on objects that are logical, numeric, or raw. When a numeric object is coerced to logical, all of the nonzero values are set to **TRUE** and the zero values are set to **FALSE**. For raw vectors, the operators are applied bitwise.

The negation operator changes **TRUE** to **FALSE** and **FALSE** to **TRUE** in a logical object. The operator **|** compares the two objects elementwise and, for each pair of elements, returns **TRUE** if **TRUE** is present, and **FALSE** otherwise. The operator **||** compares the first element of the first object to the first element of the second object and returns **TRUE** if **TRUE** is present, or **FALSE** otherwise.

The operator & compares two objects elementwise and, for each pair of elements, returns **TRUE** if both elements are **TRUE**, and **FALSE** otherwise. The operator **&&** compares the first element of the first object to the first element of the second object and returns **TRUE** if the first elements are both **TRUE**, otherwise **FALSE**.

The `xor()` function compares objects elementwise and returns **TRUE** if the paired elements are different and **FALSE** if the paired elements are the same.

For a logical vector or a vector that can be coerced to logical, the function `any()` will return **TRUE** if any of the elements are **TRUE**, and **FALSE** otherwise.

For more information about the logical operators, the CRAN help pages for logical operators can be found by entering **??"logical operators"** at the R prompt. The help page for `any()` can be accessed by entering **?any** at the R prompt.

Arithmetic Operators

Arithmetic operators can have numeric operands or operands that can be coerced to numeric. For example, for logical objects **TRUE** coerces to **1** and **FALSE** coerces to **0**. For some types of objects, specific operators have a different meaning, but those types of objects will not be covered in this chapter.

Arithmetic expressions are evaluated elementwise. If the number of elements is not the same between the objects in an expression, the smaller object cycles through the larger one until the end of the larger one. The numbers of elements in the larger object do not have to be a multiple of the smaller object for cycling. Expressions are evaluated from left to right, under the rules of precedence.

The arithmetic operators are the standard * for multiplication, / for division, + for addition, and - for subtraction. The exponentiation symbol is ^. The operator **%%** gives the modulus of the first argument with respect to the second argument. The operator **%/%** performs integer division. Expressions can be grouped using parentheses, for example **(a+b)/c**. Table 3-2 lists the arithmetic operators.

Table 3-2. *Arithmetic Operators*

Operator	Operation	Example
*	multiplication	a*b
/	division	a/b
+	addition	a+b
-	subtraction	a-b
^	exponentiation	a^b
%%	modulus	a%%b
%/%	integer division	a%/%b

For more information, the CRAN help pages for arithmetic operators can be found by entering **??"arithmetic operators"** at the R prompt.

Matrix Operators and Functions

R provides operators and functions to manipulate matrices. A list of some matrix operators and functions can be found in Table 3-3.

Table 3-3. *Matrix Operators and Functions*

Operator / Function	Operation	Example
%*%	matrix multiplication	a%*%b
%o% or **outer( )**	outer product of two vectors, matrices, or arrays	a%*%b, outer(a,b)
t( )	transpose of a matrix	t(a)
crossprod( ) or **tcrossprod( )**	crossproduct of a matrix or two matrices	crossprod(a) or crossprod(a,b) or tcrossprod(a) or tcrossprod(a,b)
diag( )	diagonal of a matrix or a diagonal matrix	diag(a), **a** is a matrix or diag(a), **a** is a vector
solve()	inverse of a matrix or solution to **Xa=b**	solve(a), solve(X,b)

The matrix multiplication operator is **%*%**. R will return an error if the two matrices do not conform.

For two arrays (arrays include vectors and matrices), **%o%**, or outer(), gives the outer product of the arrays.

To transpose a matrix, use the function `t()`, for example, t(a).

To get the cross product of one matrix with another (or the original matrix), use either the function `crossprod()` or the function `tcrossprod()`. If **a** and **b** are conforming matrices, then

```
crossprod(a) = t(a)%*%a,

tcrossprod(a) = a%*%t(a),

crossprod(a,b) = t(a)%*%b,

tcrossprod(a,b) = a%*%t(b).
```