

2

Computing with R

Data tables are often large, so it is not possible to undertake paper-and-pencil operations on them. The `BabyNames` data table 1.1 provides a case in point. `BabyNames` has 1,790,091 rows, far too many to carry out by hand even a simple sum or count. Data science relies on computers.

The human role in the process is managerial, to decide what forms of analysis are called for by the purpose at hand and to instruct the computer to carry out the operations needed. The process of instructing a computer is called *computer programming*. Programming is done using a language that is accessible both to humans and the computer; the human writes the instructions and the computer carries them out. These notes use a computer language called R.

This chapter introduces concepts that underlie the use of R. The emphasis here will be on the *kinds of things* that R commands involve. Only a few R commands will be covered, but these will illustrate the most important patterns in writing with R. Chapter 3 will start to introduce the commands themselves that are used for working with data.

2.1 R and RStudio

The R language includes many useful operations for working with data, drawing graphics, and writing documents, among other things.

RStudio is a computer application that gives ready access to the resources of R. RStudio provides a powerful way for a person to interact with R. There are other ways to work with R, but for good reason RStudio has become the most popular. RStudio offers facilities both for newbies and for professionals; it's good for everybody.

RStudio comes in two versions:

1. A *desktop* that runs on the user's own computer.
2. A *server* that runs on a computer in the cloud and is accessed by a

COMPUTER PROGRAMMING: The process of writing instructions for a computer. Sometimes the unfortunate word "coding" is used. The point of programming is to make instructions clear, not to obscure them with code.

R: A language for working with data.

RSTUDIO: An interface for organizing your work and interacting with R.

DESKTOP: An application that runs on your own computer.

SERVER: An application that is being run remotely, as a service to many users.

web browser.

You can install R and the desktop version of RStudio on all the most common operating systems: OS-X, Windows, Linux. There are now many text and video guides to installation. See `marin-stats-lectures` for an example.

If you are to use the server version of RStudio, someone has set up a server for you and provided a login ID. You access the server through an ordinary web browser — Chrome, Firefox, Safari, etc. — that may be running on a laptop, a tablet, or even a smartphone.

The two RStudio versions, desktop or server, are almost identical from a user's point of view. The desktop version is often faster but can only be used on one computer; the server version requires almost no set-up and is available from *any* computer browser, but it can be slower.

2.2 Components of the RStudio application

Just as a window in a house has several individual panes of glass, so the RStudio application appears in an graphical interface window divided up into several *panes*. All panes can be viewed at the same time. Each of the panes can contain several *tabs*.

Setting up an RStudio server? This requires some information technology (IT) knowledge and knowledge of the UNIX operating system. If you're comfortable with such things, there are many guides on the web.

The server version slows down when there are many simultaneous users, as might happen in a classroom.

PANES: Regions of the RStudio application that are displayed side-by-side.

TABS: Subdivisions of a pane, only one of which can be viewed at one time.

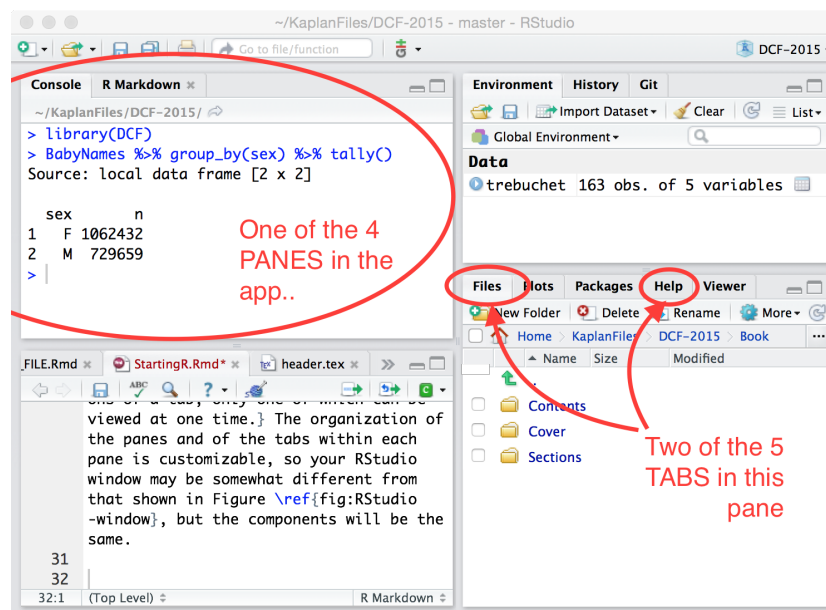


Figure 2.1: The RStudio window is divided into four panes. Each panes can contain multiple tabs.

The organization of the panes and of the tabs within each pane is customizable, so your RStudio window may be somewhat different from that shown in Figure 2.1, but the components will be the same.

You will often be using the Console tab and tabs for editing documents. Each document-editing tab has the name of the document it contains. In Figure 2.1, three such editing tabs are shown, one of which is labeled `StartingR`. Other important tabs: Packages for installing new R software and Help for displaying documentation.

2.3 Commands and the Console

This is a good time to start up your version of RStudio. Once RStudio is running:

- Enter R-language commands into the *console*. Find the console tab in your RStudio app and the prompt, `>`.
- Move the cursor to the console tab so that anything you type shows up there. Type the simple command shown in Figure 2.2.
- After you press *enter*, \leftrightarrow , R will *execute* your command and print out a response.

Notice that after the response, there is another prompt. You're ready to go again.

PRACTICE. Enter each of the following arithmetic commands at the command prompt, one at a time. Confirm that the response is the right arithmetic answer.

```
16 * 9
sqrt(2)
20 / 5
18.5 - 7.21
```

2.4 Sessions

Your work in R occurs in *sessions*. A session is a kind of ongoing dialog with the R system.

A new session is begun every time you start RStudio. The session is terminated only when you close or *quit* the RStudio program. If you are using RStudio server (via your browser), the R session will be maintained for days or weeks or months, and will be retained even when you login to the server from a different computer.

When a session is first started, it is in an empty environment. RStudio displays this in the *Environment* tab. (Figure 2.3)



Figure 2.2: The console tab showing the command prompt `>` (top), a command composed but not yet entered (middle), and after entering the command. (bottom)

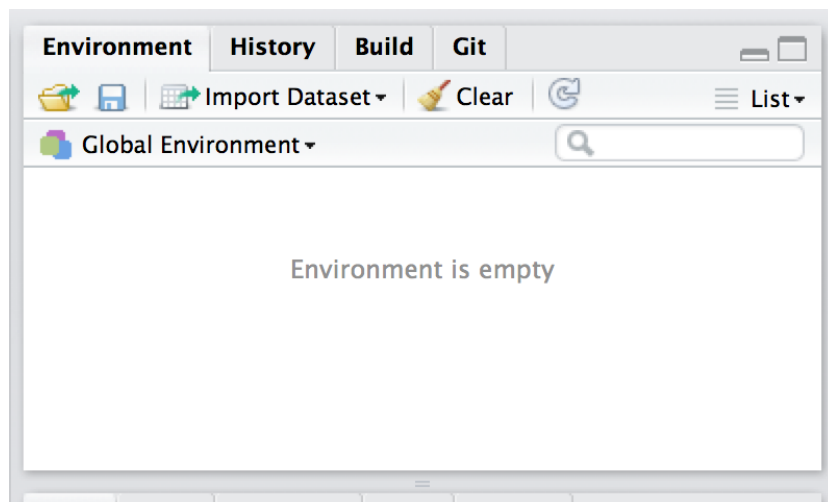


Figure 2.3: The "Environment" tab in RStudio lists the objects in the "session environment." At the start of a session, the environment is empty.

2.5 Packages

Many people work to provide and maintain new software and new capabilities for R. Such additions are called *packages*. Among the packages you will be using, the *DataComputing* package contains data for the examples in this book. In order to make these data accessible, you have to *load* the package. The command is:

```
library(DataComputing)
```

Usually you will give this command at the start of a session or at the top of a document.

Did something go wrong?

In response to the above command, you might get a response from R like this:

```
Error in library(DataComputing) : there is no package called 'DataComputing'
```

If this happens, it means only that your R account is not up to date with the *DataComputing* package and you will have to *install* the package. To do so, use these two commands in sequence. (Copy the commands verbatim into your R console. The installation may take some minutes depending on the speed of your Internet connection)

```
install.packages("devtools")
devtools::install_github("DataComputing/DataComputing")
```

Then try again with `library(DataComputing)`.

PACKAGE: The standard way of distributing new software to run within R.

DATA COMPUTING PACKAGE: R software written as a companion to this book. You will need to install it once, then load it each time you start up R to work with the book.

LOAD: Instructing R to refer to commands and data in an R package.

INSTALL: A one-time process that copies the contents of a package from its distribution site onto your computer or your account on the RStudio server.

Note the repeated "DataComputing/DataComputing".

2.6 Data and Objects

The data you work with will be organized into *data tables*. Data tables are generally stored in files or database systems or even web pages. To use a data table in R, you need to read the data into your R session. There are many ways to do this. The simplest, and the one we will use in this introduction, is via the `data()` function.

You can list the data in a package with this command:

```
data(package = "DataComputing")
```

The command shows a table like the following in an editing tab:



Item	Title
BabyNames	Names of children as recorded by the US Social Security Administration.
CountryGroups	Membership in Country Groups
HappinessIndex	World Happiness Report Data
MedicareCharges	Charges to and Payments from Medicare
... and so on for 19 rows	

To *read* a data table into R from the DataComputing package, use the `data()` function with the name of the data table and the name of the package, as in:

```
data("NCHS", package = "DataComputing")
```

To take a peak at the data that has already been read into an ongoing session, use `View()`. This will display the object in a new tab as in Figure 2.4.

```
View(NCHS)
```

Another way to get information about the data is to click on the expand icon,  next to the NCHS listing in the environment tab. You can look at the data table more closely by clicking on the small spreadsheet icon, . Finally, for data tables contained in a package, you can display a narrative description — the codebook — using the `help()` command:

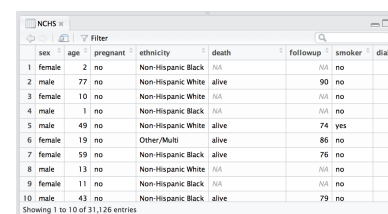
```
help("NCHS")
```

2.7 Functions, arguments and commands

The examples above demonstrate many of the most important components needed to use R. Giving these components names will help in communicating about using R.

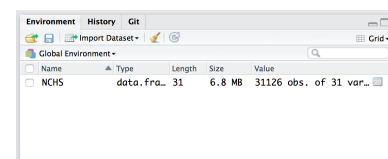
DATA TABLE: A spreadsheet-like configuration of tidy data. Another term used by R users is data frame

READ: To bring a data table into an R session. There are many ways to read data tables; `data()` will suffice for the present.





	sex	age	pregnant	ethnicity	death	followup	smoker	diabe
1	female	2	no	Non-Hispanic Black	NA	NA	no	
2	male	77	no	Non-Hispanic White	alive	90	no	
3	female	10	no	Non-Hispanic White	NA	NA	no	
4	male	1	no	Non-Hispanic Black	NA	NA	no	
5	male	49	no	Non-Hispanic White	alive	74	yes	
6	female	19	no	Other/Multi	alive	86	no	
7	female	59	no	Non-Hispanic Black	alive	76	no	
8	male	13	no	Non-Hispanic White	NA	NA	no	
9	female	11	no	Non-Hispanic Black	NA	NA	no	
10	male	43	no	Non-Hispanic Black	alive	79	no	

Figure 2.4: The tab opened by RStudio in response to `View(NCHS)`.



Name	Type	Length	Size	Value
NCHS	data.frame	31	6.8 MB	31126 obs. of 31 var...

Figure 2.5: The session environment after the NCHS data table has been loaded.

If you haven't already done so, go back to the beginning and type each command into the R console, find the environment tab, and press the spreadsheet icon  and the summary icon .

FUNCTIONS ARE THE MECHANISM THAT R USES TO CARRY OUT AN OPERATION. Functions take one or more inputs and produce an output. You've seen several functions already: `sqrt()`, `library()`, `data()`, and `help()`. Functions have names. In this text, function names will be written followed by open and close parentheses simply to signal to you, the human reader, that the name refers to a function, as opposed to, say, `NCHS`, which is a data table, not a function.

Think of a function as a **verb**. It tells *what* to do.

A COMMAND IS A COMPLETE STATEMENT OF A COMPUTATION. Commands are usually constructed by giving inputs to a function. Think of a command as a **sentence**.

The grammar of R sentences is straightforward. Follow the *name* of the function with a pair of parentheses. Inside the parentheses, you put the input on which the operation is to be performed.

All of the commands above have this form: a function name followed by parentheses containing an argument.

```
sqrt(2)
library("DataComputing")
library("mosaicData")
data(package="DataComputing")
data(package="mosaicData")
data("NCHS")
help("NCHS")
```

You can also use an object name itself as a command.

```
NCHS
```

This is useful when you want to get a quick display of the value stored under that name.

THE INPUTS TO FUNCTIONS are called *arguments*. Here are some of the arguments in the above examples:

```
2
"DataComputing"
package = "DataComputing"
"NCHS"
```

The third example, `package="DataComputing"` is called a *named argument*. Named arguments are a way to signal clearly what role the argument is to play. In the command `data(package="DataComputing")` the name of the argument is `package` and the value `"DataComputing"` is the value given to that argument.

FUNCTION: The software container for an operation that transforms one or more inputs into an output.

COMMAND: In R, a command is a complete statement of what to do. Commands are made up of functions and arguments.

Behind the scenes: Using an object name as a command is just shorthand for a command using the `print()` function. The explicit command would be `print(NCHS)` in this example.

ARGUMENT: An input to a function. In R, the argument is enclosed in parentheses that follow the function name. If there is more than one argument, they are separated by commas, e.g.
`arrange(CountryData, gdp)`

It would be reasonable to call the inputs to functions simply "inputs," but that's not the convention.

NAMED ARGUMENT: An argument whose role in the function is marked by a name. When a function takes named arguments as inputs, the name rather than the order determines how the input is used.

When there is more than one argument to a function, put them all in the same set of parentheses with a comma between the different arguments.

2.8 Objects

An *object* is a packet of information. Just about everything you'll be using in R is an object. There are many different sorts of objects, just like there are many sorts of things in the world.

It helps to distinguish between the packet and the information contained in it. The information contained in the object is called its *value*. Here are some of the objects and their values that appear in the examples above:

Name	Value	Kind of object
NCHS	data	a data table
sqrt	computer commands	a function
	"DataComputing"	a string of characters
	2	a number
	"NCHS"	a string of characters

Most of the objects you will use have names, e.g. NCHS or sqrt. Sometimes you will use objects that don't have a name, for instance the number 2 or the quoted set of characters "DataComputing".

It will take a while for you to get used to the difference between an object and the *name* of an object. Object names are never in quotes and they never begin with a digit. When quotes are used, it is to identify characters as a string. Strings are used for labels, or to identify something outside of R such as a web location, file name, or caption on a graphic.

Giving Names to Objects

Analyzing or visualizing data often involves several steps, each of which creates a new objects. It's often useful to name these objects so that you can refer to them in the following steps.

Name an object with the <- notation. The syntax is simple:

```
name <- value
```

In the jargon of computer programming, naming an object is called *assignment*. You assign a name to an object.

EXAMPLE: RANDOM SAMPLING. It's often useful to take a random subset of the cases in a data table.¹ The function that does this is

OBJECT: a packet containing information. The word "object" is not any more descriptive than, say, "thing." At least, "object" makes it clear that you are referring to a thing in R.

VALUE: The information contained in an object

ASSIGNMENT: Storing a value along with a name that can be used later to refer to or access the value

¹ For instance, you might want to prototype with a small data table when developing your data wrangling and visualization before tackling the whole data table.

called `sample_n()`. It takes two arguments: the name of the data table from which the subset is to be drawn and a named argument, `size=`, that specifies how many cases should be in the sample. For example:

```
SmallNCHS <- sample_n(NCHS, size=100)
```

By assigning the output of `sample_n()` to a named object, `SmallNCHS` in the statement above, you can access the object when you need it.

You can have as many named objects as you like. RStudio helps you keep track of them by listing them in the Environment tab, as in Figure 2.6.

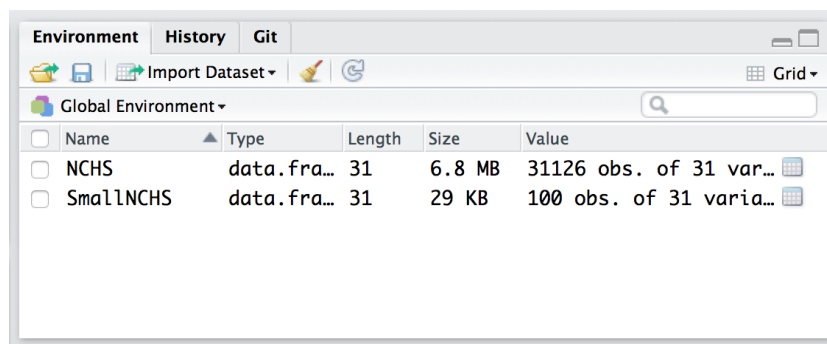


Figure 2.6: When the `SmallNCHS` object is created, it appears in the Environment tab.

THERE ARE A FEW SIMPLE RULES that apply when creating a name for an object:

- The name cannot start with a digit. So ~~100~~NCHS is not allowed, although `NCHS100` is fine. This rule is to make it easy for R to distinguish between object names and numbers. It also helps you avoid mistakes such as writing `2pi` when you mean `2*pi`.
- The name cannot contain any punctuation symbols (with two exceptions). So `?NCHS` or `N*Hanes` are not legitimate names. The exceptions: You can use `.` and `_` in a name.
- The case of the letters in the name matters. So `NCHS`, `nchs`, `Nchs`, and `nChs`, etc. are all different names that only look similar to a human reader, not to R.

The strikethrough bar, like ~~this~~, is not part of the name. The strikethrough is just a device in the notes to remind you that the name is not allowed.

Occasionally, you will encounter function names like `data.table::fread()`. This refers to a function contained in a specific package. In this case, the name refers to the `fread()` function in the `data.table` package.

EXAMPLE: READING A FILE The following command consists of a function, a quoted *character string*, the assignment operator,

CHARACTER STRING: Letters, numbers, punctuation, etc. meant to be taken literally as a value. Character strings always start and end with quotes, for instance, "My name is ...". In contrast, quotes are not used with object names.

and an object name.

```
Motors <- read.file("http://tiny.cc/mosaic/engines.csv")
```

The effect of this command is to read some data about internal combustion motors from a web site into an R object called `Motors`. Note that the URL of the data file is a quoted character string, but the function and object names are *not* quoted.

2.9 Exercises

Problem 2.1

The following ideas should be meaningful to you from Chapter 2:

*package, function, command, argument,
assignment, object, object name, data table,
named argument, quoted character string, value*

Construct an example R command that makes use of at least four of the ideas. Label which part of your example R command corresponds to each of those ideas.

Problem 2.2

Which of these kinds of names should be wrapped with quotation marks when used in R?

1. function name
2. file name
3. the name of an argument in a named argument
4. object name

Problem 2.3

Look at the documentation for the CPS85 data table in the `mosaicData` package. From reading that documentation, what is the meaning of CPS?

Problem 2.4

What's wrong with this statement?

```
help(NHANES, package <- "NHANES")
```

Problem 2.5

Look at the help documentation for the `library()` function.

Without worrying about all the detail, answer these questions simply:

1. What is the other function listed under "Usage"?
2. In the "See Also" section of the documentation, what is the name of the function after `detach()`?

Problem 2.6

Some of these are legitimate object names, others are not. For the ones that are not legitimate, say what is wrong.

1. `essay14`
2. `first-essay`
3. `"MyData"`
4. `third_essay`
5. `small sample`
6. `functionList`
7. `FuNcTiOnLiSt`
8. `.MyData.`
9. `sqrt()`

Problem 2.7

Install the `nycflights13` package into R. (You can use the “Packages” tab which has an “install” button. If you are not using RStudio, given the R command `install.packages("nycflights13")`)

Once the package is installed, you can access the `flights` data table with this command:

```
data(flights, package="nycflights13")
```

The codebook is available with

```
help(flights)
```

Using the codebook and examining the data table with the `View()` command (hint: you’ll need to give `flights` as an argument to `View()`), answer these questions:

1. How many variables are there?
2. How many cases are there?
3. What is the meaning of a case? (“Meaning” refers to the kind of entity, for instance, “airport” or “airline” or “date”. Hint: the case in `flights` is not any of these things.)
4. For each variable, is the variable quantitative or categorical?
5. For the variables `air_time` and `distance`, what are the units?

Problem 2.8

Consider this list of some possible mistakes in an assignment operation:

1. No assignment operator
2. Unmatched quotes in character string
3. Improper syntax for function argument
4. Invalid object name
5. No mistake

For each of the following assignment statements, say what is the mistake.

- a. `ralph <- sqrt 10`
- b. `ralph2 <-- "Hello to you!"`
- c. `3ralph <- "Hello to you!"`
- d. `ralph4 <- "Hello to you!"`
- e. `ralph5 <- date()`

Problem 2.9

Here are a few characters: `.`, `,`, `;`, `_`, `-`, `^`, `[space]`, `(`, `)`

- Which of those characters can be used in the name of an R object?
- Which of those characters can be used in a quoted character string?

Problem 2.10

These questions should be easy to answer if you use the appropriate commands to load, view, or get documentation on the datasets.

- How many variables are there in `CountryData`?
- What does the variable `tfat` measure in the NCHS data table? (in package `DataComputing`)
- How many cases are there in `WorldCities`?
- What’s the third variable in `BabyNames`?
- What are the codes for the levels of the categorical variable `party` in the `RegisteredVoters` data table, and what does each code stand for?