There are several different ways to enter data into R. This document will describe several of them.

Entering data using `c` . The easiest way to enter small data sets is with the function `c` that *concatenates* numbers (or vectors) together. For example, we could create an object named 'glucose' containing the 31 measures as follows. This data comes from Exercise 2.10 in the third edition of *Statistics for the Life Sciences* by Samuels and Witmer.

```
> glucose = c(81, 85, 93, 93, 99, 76, 75, 84, 78, 84, 81, 82, 89,
+     81, 96, 82, 74, 70, 84, 86, 80, 70, 131, 75, 88, 102, 115,
+     89, 82, 79, 106)
```

You do not need to type in the '+' symbols, which are prompts that indicates R is waiting for a command to be completed. As shown here, this command was so long it did not fit onto a single line, so R wrapped to the following line. You could have typed it in without breaking the command into several lines. You may also press `[Enter]` to continue a command on the next line.

Warning—if you type a '`(`' and then do not complete the command by typing a '`)`', R will continue to wait for the command to be completed and show a string of '`+`' prompts even if you continue to press `[Enter]`. If you get in trouble, you can press `[Esc]`, the Escape key to break back to a regular prompt.

Entering data using `scan` . The previous example is a bit of a pain, because you need to enter all of the commas. The `scan` function reads in data separated by "white space", spaces, tabs, and newline characters. If you press `[Enter]` on a blank line, you end the input. Here is the same example.

```
> glucose = scan()
1: 81 85 93 93 99 76 75 84 78 84 81 82 89 81
15: 96 82 74 70 84 86 80 70 131 75 88 102
27: 115 89 82 79 106
32:
Read 31 items
```

Entering data using `read.table` . Your textbook has a CD with all of the data sets used in the textbook. It is worth learning how to read in these data sets.

You will probably want to begin by creating a folder that will contain the data from the CD. I will assume that you know how to copy the data files from the CD to a folder on your hard drive. There is a folder on the CD called 'DataFiles'. Under this, there is a folder for each chapter. For each chapter, there are folders for the data sets in different formats. The easiest to read into R are the ASCII formatted files, or plain text files. The data for Exercise 2.10 is in the 'DataFiles/Chpt 2/ASCII' folder and is called 'glucose.txt'.

If you have copied this file to your hard drive *and if you have changed directories through the File menu so that your working directory contains this data file*, this command will read in the data.

```
> x = read.table("glucose.txt", header = T)
```

(If you see an error that the file is not found, it is probably the case that the file is not in your working directory. You may want to try the `file.choose` method described below.)

The file "glucose.txt" is typical for ASCII data sets from the textbook. The first row contains a "header", the variable names for each variable. In addition ot the header, there is one row for each observational unit. This example has only one variable and one column of data. Other examples can have multiple columns of data. The default behavior of R is to read in data frames without headers. That is why it was necessary to add the argument `header=T` to the command.

The R commands above create `x`, an object known in R as a *data frame*. The name `x` is arbitrary. You may use any valid variable name (which does not begin with a digit or use characters with other meaning). A data frame is a data matrix, but can include both categorical and numerical variables.

To ensure that you have read in what you think you have, it is useful to use the command `str` to check the structure of the data frame.

```
> str(x)

'data.frame':        31 obs. of  1 variable:
 $ glucose: int  81 85 93 93 99 76 75 84 78 84 ...
```

The output tells you that the data frame `x` has one quantitative variable called `glucose` and it shows the first several values.

After reading in a data frame, R will recognize the name `x` as the name of the entire data frame, but R will not recognize the variable `glucose`. There are two ways to get around this. First, the variables of each data frame may be accessed using the `$` operator. Here is an example.

```
> x$glucose

 [1]  81  85  93  93  99  76  75  84  78  84  81  82  89  81  96  82  74  70  84
[20]  86  80  70 131  75  88 102 115  89  82  79 106
```

Alternatively, if you use the `attach` command, you can refer to each variable in `x` by name without the dollar sign.

```
> attach(x)
> glucose

 [1]  81  85  93  93  99  76  75  84  78  84  81  82  89  81  96  82  74  70  84
[20]  86  80  70 131  75  88 102 115  89  82  79 106
```

You only need to type in the attach command once per session — once you have done it, R will know that `glucose` means the same thing as `x$glucose` until you quit R.

Reading in data using `file.choose`. As an alternative to typing in the file name, you may use the function `file.choose` which will open up a dialog box that you can use to move through your folders to find the data file you want to read in. The syntax is

```
> x = read.table(file.choose(),header=T)
```

Reading in data from an Excel file. R does not read in data directly from an Excel file, but you can use Excel to export data in a format that R can read. To do this, create an Excel spread sheet. Use the first row of the spread sheet as the header row and put the variable values below. Then, save this spread sheet as a CSV file, for comma-separated-variable file. This format is a plain text file, but there will be commas instead of spaces between variable values. Then, in R you would read in the data using `read.table` with the additional argument `sep=","` to indicate that commas are used as separators between fields. When there is only one variable, this is not necessary.

If you have a data file called "students.csv" that looks like this,

```
First,Last,Height,BloodType
Adam,Ant,71.5,A
Jon,Bon Jovi,70,O
Marshal,Mathers,69,B
Elvis,Presley,73,AB
```

you could read in the data and check it with these commands.

```
> students = read.table("students.csv", header = T, sep = ",")
> str(students)

'data.frame':        4 obs. of  4 variables:
 $ First    : Factor w/ 4 levels "Adam","Elvis",..: 1 3 2 4
 $ Last     : Factor w/ 4 levels "Ant","Bon Jovi",..: 1 2 4 3
 $ Height   : num  71.5 70 73 69
 $ BloodType: Factor w/ 4 levels "A","AB","B","O": 1 4 2 3
```