1. Input

Contents

Here we explore how to define a data set in an R session. Only two commands are explored. The first is for simple assignment of data, and the second is for reading in a data file. There are many ways to read data into an R session, but we focus on just two to keep it simple.

1.1. Assignment

The most straight forward way to store a list of numbers is through an assignment using the c command. (c stands for “combine.”) The idea is that a list of numbers is stored under a given name, and the name is used to refer to the data. A list is specified with the c command, and assignment is specified with the “<-” symbols. Another term used to describe the list of numbers is to call it a “vector.”

The numbers within the c command are separated by commas. As an example, we can create a new variable, called “bubba” which will contain the numbers 3, 5, 7, and 9:

```
> bubba <- c(3,5,7,9)
>
```

When you enter this command you should not see any output except a new command line. The command creates a list of numbers called “bubba.” To see what numbers is included in bubba type “bubba” and press the enter key:

```
> bubba
[1] 3 5 7 9
```

If you wish to work with one of the numbers you can get access to it using the variable and then square brackets indicating which number:

```
> bubba[2]
[1] 5
> bubba[1]
[1] 3
> bubba[0]
numeric(0)
> bubba[3]
[1] 7
> bubba[4]
[1] 9
```

Notice that the first entry is referred to as the number 1 entry, and the zero entry can be used to indicate how the computer will treat the data. You can store strings using both single and double quotes, and you can store real numbers.

You now have a list of numbers and are ready to explore. In the chapters that follow we will examine the basic operations in R that will allow you to do some of the analyses required in class.

1.2. Reading a CSV file

Unfortunately, it is rare to have just a few data points that you do not mind typing in at the prompt. It is much more common to have a lot of data points with complicated relationships. Here we will examine how to read a data set from a file using the read.csv function but first discuss the format of a data file.

We assume that the data file is in the format called “comma separated values” (csv). That is, each line contains a row of values which can be numbers or letters, and each value is separated by a comma. We also assume that the very first row contains a list of labels. The idea is that the labels in the top row are used to refer to the different columns of values.

trial	mass	velocity

First we read a very short, somewhat silly, data file. The data file is called simple.csv and has three columns of data and six rows. The three columns are labeled "trial," "mass," and "velocity." We can pretend that each row comes from an observation during one of two trials labeled "A" and "B." A copy of the data file is shown below and is created in defiance of Werner Heisenberg:

silly.csv¶

trial	mass	velocity
A	10	12
A	11	14
B	5	8
B	6	10
A	10.5	13
B	7	11

The command to read the data file is *read.csv*. We have to give the command at least one arguments, but we will give three different arguments to indicate how the command can be used in different situations. The first argument is the name of file. The second argument indicates whether or not the first row is a set of labels. The third argument indicates that there is a comma between each number of each line. The following command will read in the data and assign it to a variable called "heisenberg:"

```
> heisenberg <-
read.csv(file="simple.csv",head=TRUE,sep=",")
> heisenberg
trial mass velocity
1     A 10.0       12
2     A 11.0       14
3     B  5.0        8
4     B  6.0       10
5     A 10.5       13
6     B  7.0       11
> summary(heisenberg)
trial      mass          velocity
A:3   Min.   : 5.00   Min.   : 8.00
B:3   1st Bu.: 6.25   1st Qu.:10.25
      Median : 8.50   Median :11.50
      Mean   : 8.25   Mean   :11.33
      3rd Qu.:10.38   3rd Qu.:12.75
      Max.   :11.00   Max.   :14.00
```

(Note that if you are using a Microsoft system the file naming convention is different from what we use here. If you want to use a backslash it needs to be escaped, i.e. use two backslashes together "\." Also you can specify what folder to use by clicking on the "File" option in the main menu and choose the option to specify your working directory.)

To get more information on the different options available you can use the help command:

```
> help(read.csv)
```

If R is not finding the file you are trying to read then it may be looking in the wrong folder/directory. If you are using the graphical interface you can change the working directory from the file menu. If you are not sure what files are in the current working directory you can use the *dir()* command to list the files and the *getwd()* command to determine the current working directory:

```
> dir()
[1] "fixedWidth.dat" "simple.csv"     "trees91.csv"
"trees91.wk1"
[5] "w1.dat"
> getwd()
[1] "/home/black/write/class/stat/stat383-13F/dat"
```

The variable “heisenberg” contains the three columns of data. Each column is assigned a name based on the header (the first line in the file). You can now access each individual column using a “$” to separate the two names:

```
> heisenberg$trial
[1] A A B B A B
Levels: A B
> heisenberg$mass
[1] 10.0 11.0  5.0  6.0 10.5  7.0
> heisenberg$velocity
[1] 12 14  8 10 13 11
```

If you are not sure what columns are contained in the variable you can use the names command:

```
> names(heisenberg)
[1] "trial"    "mass"     "velocity"
```

We will look at another example which is used throughout this tutorial. we will look at the data found in a spreadsheet located at http://cdiac.ornl.gov/ftp/ndp061a/trees91.wk1 . A description of the data file is located at http://cdiac.ornl.gov/ftp/ndp061a/ndp061a.txt . The original data is given in an excel spreadsheet. It has been converted into a csv file, trees91.csv , by deleting the top set of rows and saving it as a “csv” file. This is an option to save within excel. (You should save the file on your computer.) It is a good idea to open this file in a spreadsheet and look at it. This will help you make sense of how R stores the data.

The data is used to indicate an estimate of biomass of ponderosa pine in a study performed by Dale W. Johnson, J. Timothy Ball, and Roger F. Walker who are associated with the Biological Sciences Center, Desert Research Institute, P.O. Box 60220, Reno, NV 89506 and the Environmental and Resource Sciences College of Agriculture, University of Nevada, Reno, NV 89512. The data is consists of 54 lines, and each line represents an observation. Each observation includes measurements and markers for 28 different measurements of a given tree. For example, the first number in each row is a number, either 1, 2, 3, or 4, which signifies a different level of exposure to carbon dioxide. The sixth number in every row is an estimate of the biomass of the stems of a tree. Note that the very first line in the file is a list of labels used for the different columns of data.

The data can be read into a variable called "tree" in using the read.csv command:

```
> tree <-
read.csv(file="trees91.csv",header=TRUE,sep=",");
```

This will create a new variable called "tree." If you type in "tree" at the prompt and hit enter, all of the numbers stored in the variable will be printed out. Try this, and you should see that it is difficult to make any sense out of the numbers.

There are many different ways to keep track of data in R. When you use the *read.csv* command R uses a specific kind of variable called a "data frame." All of the data are stored within the data frame as separate columns. If you are not sure what kind of variable you have then you can use the attributes command. This will list all of the things that R uses to describe the variable:

```
> attributes(tree)
$names
[1] "C"      "N"      "CHBR"   "REP"    "LFBM"
"STBM"   "RTBM"   "LFNCC"
[9] "STNCC"  "RTNCC"  "LFBCC"  "STBCC"  "RTBCC"
"LFCACC" "STCACC" "RTCACC"
[17] "LFKCC"  "STKCC"  "RTKCC"  "LFMGCC" "STMGCC"
"RTMGCC" "LFPCC"  "STPCC"
[25] "RTPCC"  "LFSCC"  "STSCC"  "RTSCC"

$class
[1] "data.frame"

$row.names
[1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
"11" "12" "13" "14" "15"
[16] "16" "17" "18" "19" "20" "21" "22" "23" "24" "25"
"26" "27" "28" "29" "30"
[31] "31" "32" "33" "34" "35" "36" "37" "38" "39" "40"
"41" "42" "43" "44" "45"
[46] "46" "47" "48" "49" "50" "51" "52" "53" "54"
```

The first thing that R stores is a list of names which refer to each column of the data. For example, the first column is called “C”, the second column is called “N.” Tree is of type data.frame. Finally, the rows are numbered consecutively from 1 to 54. Each column has 54 numbers in it.

If you know that a variable is a data frame but are not sure what labels are used to refer to the different columns you can use the names command:

```
> names(tree)
[1]  "C"      "N"      "CHBR"   "REP"    "LFBM"
"STBM"   "RTBM"   "LFNCC"
[9]  "STNCC"  "RTNCC"  "LFBCC"  "STBCC"  "RTBCC"
"LFCACC" "STCACC" "RTCACC"
[17] "LFKCC"  "STKCC"  "RTKCC"  "LFMGCC" "STMGCC"
"RTMGCC" "LFPCC"  "STPCC"
[25] "RTPCC"  "LFSCC"  "STSCC"  "RTSCC"
```

If you want to work with the data in one of the columns you give the name of the data frame, a “$” sign, and the label assigned to the column. For example, the first column in tree can be called using “tree$C:”

```
> tree$C
[1]  1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 3 3 3 3 3 3 3
[39] 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4
```

1.3. Brief Note on Fixed Width Files

There are many ways to read data using R. We only give two examples, direct assignment and reading csv files. However, another way deserves a brief mention. It is common to come across data that is organized in flat files and delimited at preset locations on each line. This is often called a “fixed width file.”

The command to deal with these kind of files is *read.fwf*. Examples of how to use this command are not explored here, but a brief example is given. If you would like more information on how to use this command enter the following command:

```
> help(read.fwf)
```

The *read.fwf* command requires at least two options. The first is the name of the file and the second is a list of numbers that gives the length of each column in the data file. A negative number in the list indicates that the column should be skipped. Here we give the command to read the data file fixedWidth.dat . In this data file there are three columns. The first colum is 17 characters wide, the second column is 15 characters wide, and the last column is 7 characters wide. In the example below we use the optional *col.names* option to specify the names of the columns:

```
 > a =
read.fwf('fixedWidth.dat',widths=c(-17,15,7),col.names=c

 > a
  temp offices
1 17.0      35
2 18.0     117
3 17.5      19
4 17.5      28
```

Previous

Next

Sponsorship

This site generously supported by Datacamp. Datacamp offers a free interactive introduction to R coding tutorial as an additional resource. Already over 100,000 people took this free tutorial to sharpen their R coding skills.