Inputting and Exporting Files

Luke Chang

Last Revised July 16, 2010

Beginning R users often feel overwhelmed by how difficult it can be just getting started. The first task that many people struggle with is learning how to import their data into R . This section will illustrate how many different types of data (e.g., delimited and SPSS) can be imported into R with relative ease. In addition, we will also discuss how files can be saved and exported.

1 Reading in Data Files

1.1 Delimited Files

The first type of data that we will learn how to import is simple tab or comma delimited text files. These files can be exported using programs such as Excel or Matlab and are easily imported into R using the `read.table` function. As an example, we will import a comma delimited dataset on the relationship between brain size and IQ in monozygotic twins. If we are connected to the interenet, we can directly read this file from the course website. To shorten the text we will use the `paste()` function to concatenate the `website` variable with the data filename.

```
> website="http://sites.google.com/site/uarworkshop/file-cabinet/"
> brainIQ.data<-read.table(paste(website,"BrainIQ.csv",sep=""),
    sep=",",header=TRUE)
```

This command creates the data frame object, `brainIQ.data`. To ensure that R can locate the data file, it is important to specify its file path. The `read.table` function has several user defined options, which assist in correctly reading in the data file. The type of delimiter can be specifed using `sep = ","`, which we use here

to indicate that the file is comma delimited. The default is tab delimited. Users can also specify whether R should use the first row of the file to indicate the variable names using `header = TRUE`. Finally, it is also common to tell R the missing value indicator. For example, `na.strings=999999` indicates that any values of `999999` should be considered missing values.

1.2 Interrogating Your Data

One of the things that can potentially be initially psychologically difficult about switching to R from other analysis programs such as SPSS is that you don't have a spreadsheet of your data on display at all times. This can give R the feel of a black box. However, it is important to remember that your data can be queried with much greater sophistication then scrolling through a gigantic spreadsheet. For example to quickly look at your entire data.frame use the `print` command.

```
> print(brainIQ.data)
```

```
   CCMIDSA FIQ   HC ORDER PAIR    SEX   TOTSA TOTVOL  WEIGHT
1     6.08  96 54.7     1    1 Female 1913.88   1005  57.607
2     5.73  89 54.2     2    1 Female 1684.89    963  58.968
3     6.22  87 53.0     1    2 Female 1902.36   1035  64.184
4     5.80  87 52.9     2    2 Female 1860.24   1027  58.514
5     7.99 101 57.8     1    3 Female 2264.25   1281  63.958
6     8.42 103 56.9     2    3 Female 2216.40   1272  61.690
7     7.44 103 56.6     1    4 Female 1866.99   1051 133.358
8     6.84  96 55.3     2    4 Female 1850.64   1079 107.503
9     6.48 127 53.1     1    5 Female 1743.04   1034  62.143
10    6.43 126 54.8     2    5 Female 1709.30   1070  83.009
11    7.99 101 57.2     2    6   Male 1689.60   1173  61.236
12    8.76  96 57.2     1    6   Male 1806.31   1079  61.236
13    6.32  93 57.2     2    7   Male 2136.37   1067  83.916
14    6.32  88 57.2     1    7   Male 2018.92   1104  79.380
15    7.60  94 55.8     2    8   Male 1966.81   1347  97.524
16    7.62  85 57.2     1    8   Male 2154.67   1439  99.792
17    6.03  97 57.2     1    9   Male 1767.56   1029  81.648
18    6.59 114 56.5     2    9   Male 1827.92   1100  88.452
19    7.52 113 59.2     2   10   Male 1773.83   1204  79.380
20    7.67 124 58.5     1   10   Male 1971.63   1160  72.576
```

Sometimes you may be interested in just viewing a column or two at a time. Perhaps you are interested in quickly looking at the weight data for females. If you don't remember the column name you can use the `names` command. Then you can grab just the female's weight using the `subset` command.

```
> names(brainIQ.data)

[1] "CCMIDSA" "FIQ"     "HC"      "ORDER"   "PAIR"
[6] "SEX"     "TOTSA"   "TOTVOL"  "WEIGHT"

> subset(brainIQ.data$WEIGHT,brainIQ.data$SEX=="Female")

 [1]  57.607  58.968  64.184  58.514  63.958  61.690 133.358
 [8] 107.503  62.143  83.009
```

1.2.1 Editing Data

Data frames can easily be edited in `R` using two different approaches. The first approach opens up a table editor that is similar to an Excel spreadsheet interface.

```
> fix(brainIQ.data)
```

The second approach is to directly change the values of the data frame through the command line interface. For example, suppose the first subject's IQ score of 96 was incorrect and needed to be changed to 104. This can be done directly by telling `R` to replace the data value in the first row and second column to 104.

```
> brainIQ.data[1,2]<-104
```

You can confirm that the value has been correctly changed by simply displaying the first subject's data.

```
> brainIQ.data[1,]

  CCMIDSA FIQ   HC ORDER PAIR    SEX   TOTSA TOTVOL WEIGHT
1    6.08 104 54.7     1    1 Female 1913.88   1005 57.607
```

1.3 SPSS Files

Files saved in the SPSS format can be imported into R using the `read.spss()` function from the `foreign` package. Using this function it is important to indicate

that you want the data saved to a data frame (`to.data.frame=TRUE`). You may also want to tell `R` to use the value labels from the SPSS file (`use.value.labels = TRUE`) and whether or not to use the SPSS missing value codes (`use.missings=TRUE`).

```
> library(foreign)
> data<-read.spss(paste(website,"BrainIQ.sav",sep=""),
    to.data.frame=TRUE,use.value.labels = TRUE,use.missings=TRUE)
```

2 Exporting Data Files

2.1 Saving Data Frames

Often we are interested in saving a data frame that we have created in `R`This might be to be able to easily load it into another program such as Matlab or Excel or just to have a record of the data. Text files of a data frame can be saved using the `write.table` command. For example, perhaps we want to save our modified version of the brainIQ dataset to a new file.

```
> write.table(brainIQ.data,file="brainIQdata.txt")
```

2.2 Saving Objects

Sometimes you might want to save an object such as a data.frame or an analysis. This can be accomplished using the `save` command. Simply indicate the object and the name of the file. For example, we will create an object comprised of the participant's full scale IQ and save it to a file.

```
> FIQ<-brainIQ.data$FIQ
> save(FIQ,file="FIQ.rda")
```

Later if we wanted to work with that object again we can load it into the `R` workspace using the `attach` function

```
> load("FIQ.rda")
```

2.3 Saving Sessions

Sometimes you may want to save an entire session, particularly if you have created many different variables and a number of analyses that you would like to return to. This can be done using the `save.session` function. Alternatively, when you quit `R` using the `q()` command, you will be automatically prompted if you want to save your session.

```
> save.image(file="RSession_20100713.rda")
```

If you want to recover your saved session, simply use the `load` command and indicate the name of the session.

```
> load(file="RSession_20100713.rda")
```

2.4 Exporting to SPSS

Data frames in `R` can also be exported to SPSS. It is important to note that for whatever reason this process does not create a .sav file, which can easily be imported into SPSS. However, `R` will generate a simple text file and syntax code to import the file into SPSS using their datalist function. This is helpful if you have a lot of labels for your factors and want SPSS to automatically import them as labels. However, it seems fairly straightforward to export delimited files using the `write.table()` function, which can then be imported into SPSS.

```
> write.foreign(brainIQ.data,"brainIQ_data.txt","brainIQ_datalist.sps",
    package="SPSS")
```