



SOCIETY OF ACTUARIES

Article from:

CompAct

January 2009 – Issue No. 30

R Cornerⁱ

by Steve Craighead

DATA COMMA DELIMITED FORMAT (CSV)

In this article, I'll describe how to get data into and out of R. The first approach will be the simplest in that we will move data in and out by the use of text files. (Specifically the comma delimited CSV format). In R, one can read a CSV file in by the use of the read.csv method.

If you use the help command in a R command window

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

Figure 1 displays what you get from the help browser.



Steve Craighead, ASA, MAAA, is an actuarial consultant at TowersPerrin in Atlanta, Ga. He can be reached at steven.craighead@towersperrin.com.

```
Usage

read.table(file, header = FALSE, sep = ",", quote = "\"'",
  dec = ".", row.names, col.names,
  as.is = !stringsAsFactors,
  na.strings = "NA", colClasses = NA, nrows = -1,
  skip = 0, check.names = TRUE, fill = !blank.lines.skip,
  strip.white = FALSE, blank.lines.skip = TRUE,
  comment.char = "#",
  allowEscapes = FALSE, flush = FALSE,
  stringsAsFactors = default.stringsAsFactors(),
  encoding = "unknown")

read.csv(file, header = TRUE, sep = ",", quote="\"", dec=".",
  fill = TRUE, comment.char="", ...)

read.csv2(file, header = TRUE, sep = ";", quote="\"", dec=";",
  fill = TRUE, comment.char="", ...)

read.delim(file, header = TRUE, sep = "\t", quote="\"", dec=".",
  fill = TRUE, comment.char="", ...)

read.delim2(file, header = TRUE, sep = "\t", quote="\"", dec=";",
  fill = TRUE, comment.char="", ...)
```

Figure 1. Help(read.csv)results

Say you have some data in the file "george.csv," that looks like the following:

	VarOne,	VarTwo,	VarThree
1.	9000,	6.8000,	4.7000
2.	0090,	7.0080,	6.0000
3.	0900,	8.0080,	7.0070
4.	0009,	9.0008,	8.0000
5.	0090,	3.0800,	9.0007
6.	0900,	2.8000,	4.7000

Also, if this data is in the same subdirectory that you started R with, you can get this into a dataframe, let's call MyData, by using this command:

```
MyData <- read.csv("george.csv")
```

Here we have assumed that the .CSV file has header names on the first row.

If you type MyData, R will display the following:

```
> MyData
  VarOne  VarTwo  VarThree
1  1.9000  6.8000  4.7000
2  2.0090  7.0080  6.0000
3  3.0900  8.0080  7.0070
4  4.0009  9.0008  8.0000
5  5.0090  3.0800  9.0007
6  6.0900  2.8000  4.7000
```

Note how R has added a row number to each entry. Suppose, however, that george.csv is in a different sub-directory than the one that you are in. Say, that george.

csv is in C:\R\RCorner\. To access this data you will use a strange non-Windows method to enter the path information:

```
MyData <- read.csv("C:\\R\\RCorner\\george.csv")
```

Note how, R requires the use of “\\” instead of “\” when entering filepaths.

If you want to examine the data in a full screen editor, type the following command:

```
fix(MyData)
```

Figure 2 shows the content of the editor.

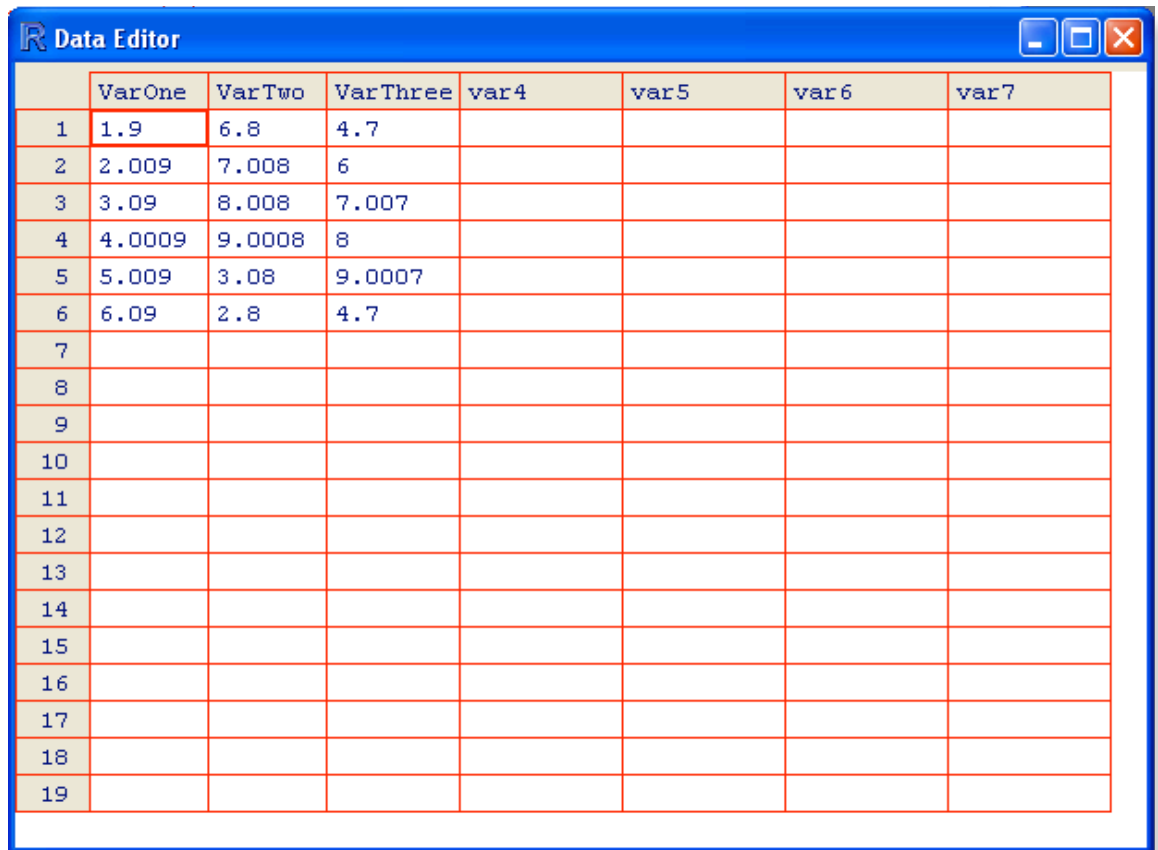



Figure 2. Screen display of the fix (MyData) command

Note that you can revise both the column names as well as the content by typing over them. Close the editor by pressing the  button.

To write out a dataframe into a CSV format, you will need to use the write command. Type the command:

```
?write.csv
```

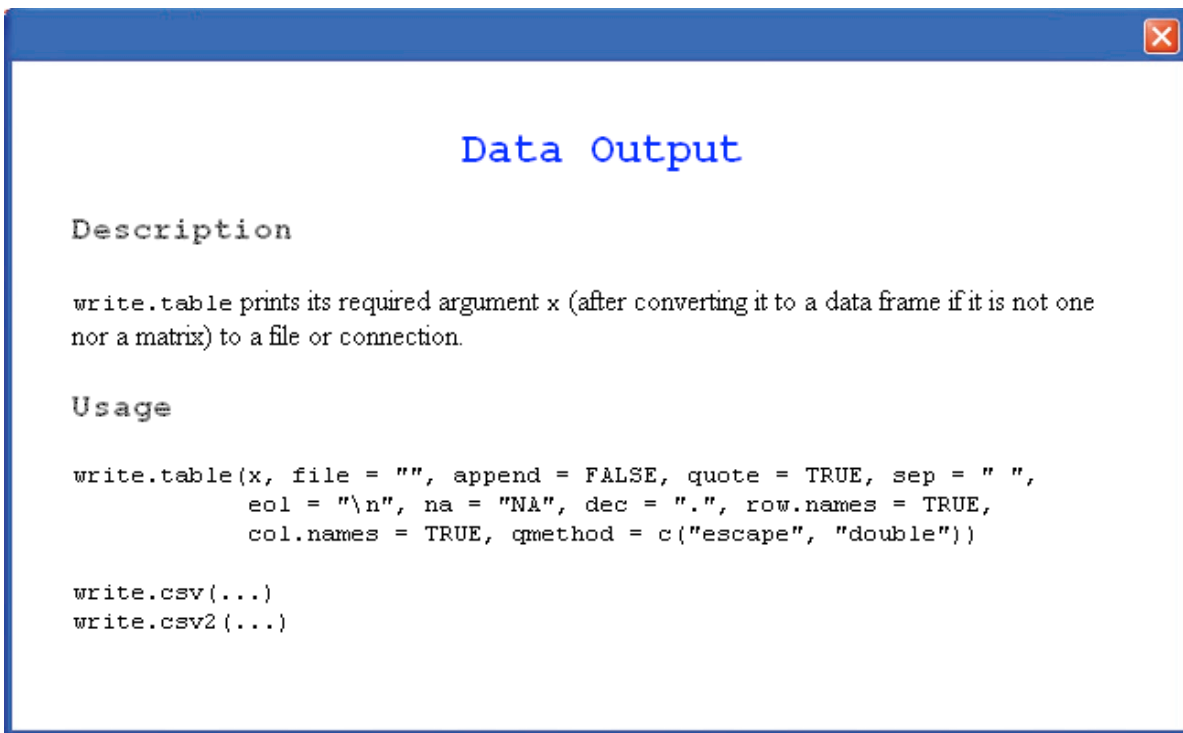


Figure 3. ?write.csv results

Figure 3 displays the result of this command. If you use the command

```
write.csv(MyData,"c:\\R\\Rcorner\\test.csv")
```

you will get a CSV file, but the first column will be one containing the row numbers. To make sure the output conforms to the same structure as what came in, when using `read.csv`, you should use the command:

```
write.csv(MyData,"c:\\R\\Rcorner\\test.csv", row.names=FALSE)
```

Now let's discuss how to use the RODBC package to get database and workbook data into R.

RODBCⁱⁱ

To get data from an Access database, you will need to use the RODBC package. The first time you use this package you will need to download it from some R package repository on the Web. The first step will be to select the Package option off of the R menu. When the dropdown list appears, choose the "Set Cran Mirror ..." option. See Figure 4 for a display of various mirrors.

CONTINUED ON PAGE 14



Figure 4. CRAN Mirror list

Choose the country or state nearest to your location and press the OK button. Next you will need to choose the “Install Packages ...” option from the dropdown list. See Figure 5, for a display of what this looks like. Look down the alphabetical list until you locate the RODBC package. Highlight this and press the OK button. If you get some message like: “Warning: unable to access index for repository ...,” you will need to choose a different CRAN repository and choose RODBC from the Install Packages again. Once you find an active repository and RODBC downloads, R will then proceed to install the package. You know that you are successful if you see the following message displayed:

package ‘RODBC’ successfully unpacked and MD5 sums checked
 Next you will need to load the package. Again choose the Package dropdown list and choose “Load Package” option. From this list, choose RODBC.

To get further insight in the overall flexibility of RODBC, you can display the help browser on the package by typing either

?RODBC

or

help(RODBC)



Figure 5. Install Packages List

To get data into R from an access database, we will use two commands. The first sets up a “channel” to link R through an ODBC link to a specified MSAccess database to R and the other command sqlFetch will fetch the data from a specific table in the database.

I have created a simple MS Access database called “db1.mdb” and it contains one table called “Example.” The contents of this table are displayed in Figure 6.

Example : Table				
	ID1	Val1	Val2	Val3
▶	1	1	9	14
	2	8	8	7
	3	323	32	94
	4	94	22	11
	5	2	2	3
	6	3232	32	22
*	(AutoNumber)	0	0	0

Figure 6. Example data table in db1.mdb database

First you need to create the R object “channel1” by using the command:

```
channel1 <- odbcConnectAccess("e:\\db1.mdb")
```

Note how I designate the drive and path structure with the “\\” instead of the “.” Also, if you are accessing a 2007 Access database you need to change the command “odbcConnectAccess” to “odbcConnectAccess2007” above.

Once the channel is setup you can store the entire block of data from the Example table into an R dataframe by the following command:

```
MyData <- sqlFetch(channel1,"Example")
```

Display MyData in R by typing “MyData” in the R command window and press enter. R will display:

```
MyData
  ID1 Val1 Val2 Val3
1   1    1    9   14
2   2    8    8    7
3   3   323   32   94
4   4   94   22   11
5   5    2    2    3
6   6  3232   32   22
```

Finally, so there are no problems with re-accessing your database later, you may want to drop the linkage

to the table and close the channel by the using the following two commands:

```
sqlDrop(channel1, "Example")
close(channel1)
```

If you have an Excel workbook that you want to extract a specific worksheet from and store into R using RODBC, you will need to use something like the following set of commands:

```
chan<-odbcConnectExcel("e:\\book1.xls", readOnly = TRUE )
MyData <- sqlFetch(chan, "Sheet1")
close(chan)
```

If you want to be more choosy on what you want to extract from an Excel workbook, I have copied the following remark from the RODBC help browser:

A ‘table’ in an Excel ‘database’ (spreadsheet) can be either a ‘named range’ (<http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q195951&>) or a worksheet: the latter have (the???)table (name the name???) of the worksheet with \$ appended (and such names may contain spaces and other characters not allowed in SQL table names). RODBC will generally allow worksheets to be referred to with or without the trailing \$, but this does need to be taken into account in SQL queries (where non-standard table names are escaped by enclosing them in square brackets).

This is just the tip of the iceberg regarding the flexibility of RODBC. Examine the RODBC help browser for further insight on how to conduct specific queries against various tables or to link to other database structures such as MySQL. However, realize that you will need a separate channel for each separate data source you want to access. ■

FOOTNOTES

¹ R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org>. Originally Michael Lapsley and from Oct 2002 B. D. Ripley (2008).

² RODBC: ODBC Database Access. R package version 1.2-3.