Importing Data into R

R version 2.15.1

January 28, 2014

R is an open source programming language focused on statistical computing. R supports many types of files as input and the following tutorial will cover some of the most popular.

Importing from text (.txt) file:

We'll begin with the simplest type of data set to import into R: the text file. When you import data into R, you're creating a data frame (a data type in R that can be easily manipulated and worked with). Every file type you import will be turned into a data frame by R. If your data file is a simple text file, this is the statement you would use:

```
mydata <- read.table('c:/datafile.txt')
```

You will now have a data frame named `mydata` in R that you can call functions on, analyze, create graphs with, or otherwise manipulate to your heart's content.

This is the most basic function call, but there are other options that are included. Say, for example, the first line of your data set contained column names. The above function call would have those column names be the first entry in your data set. To fix this and have the column names properly displayed you would use:

```
mydata <- read.table('c:/datafile.txt', header=TRUE)
```

This will take the first row of your data file and correctly interpret it as the names of the columns in your data frame.

If you have a data file where each row is separated by a character, such as '&', for instance, you can use the `sep` option in this function call.

```
mydata <- read.table('c:/datafile.txt', sep='&')
```

If you have a tab delimited data file, which are somewhat common, R has a special import function:

```
mydata <- read.delim('c:/datafile.txt')
```

However, it's worth noting that `read.delim` is just a special case of `read.table`, and tab delimited files can also be imported via `read.table` with the `sep` option set as the tab character:

```
mydata <- read.table('c:/datafile.txt', sep='\t')
```

There are many more options you can utilize with the `read.table` function if you have a data set that won't import properly or just want to format it in a different way (for example, there is a `row.names` option). If you have an interest or need of these additional options, use the following command:

```
?read.table
```

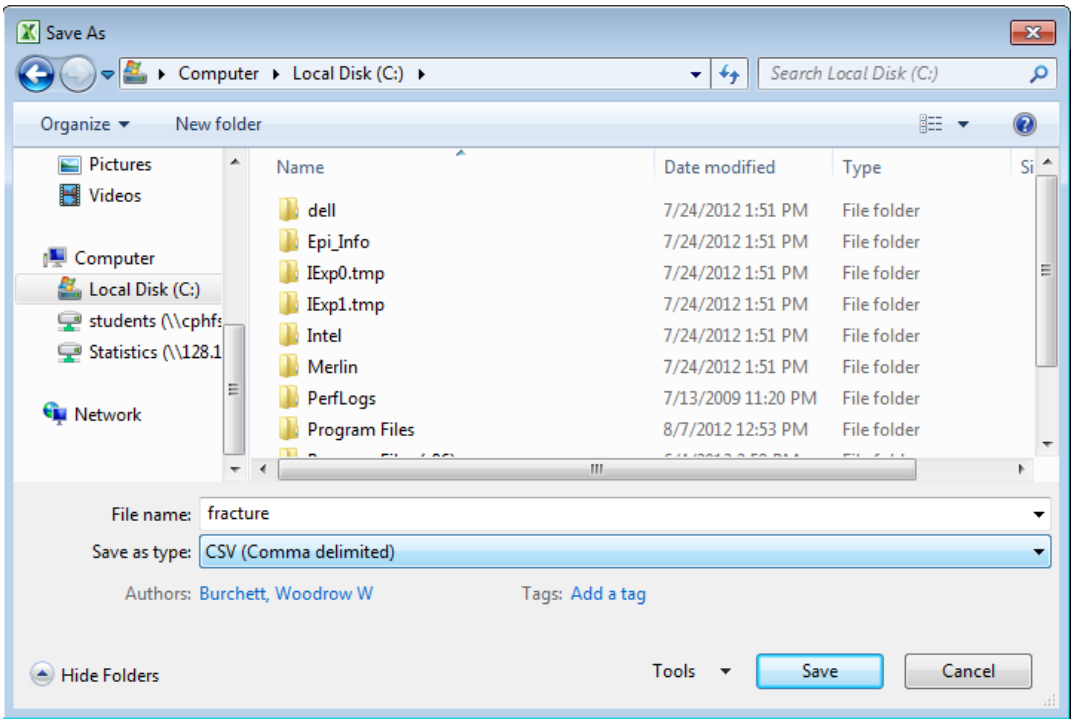
This command will bring up the help/documentation page for `read.table` and give you a detailed look at all the options available to you. Note that this syntax works for any function you want more information about in R.

TROUBLESHOOTING TIP: If you're having issues with the file path, make sure you're using '/' when specifying the file location instead of '\'. Note that when copying and pasting a file path from Windows, '\' will be used.

For example: `'C:/myFiles/R/data'` will be a valid file path while `'C:\myFiles\R\data'` will result in an error.

Importing from excel (.xls, .xlsx) file:

R has the ability to import data files from excel (.xls or .xlsx) files directly, however there are sometimes formatting issues that will lead to problems or errors with the import. Luckily there is a very quick and easy work around that excel provides: saving your excel sheet as a .csv file. I always recommend doing this, as it is much easier, faster, and will result in far less errors. The first step is to use Microsoft Excel to open the .xls or .xlsx file you with to import. Next, select 'save as' from the file menu and save the file as a CSV (comma delimited) file as shown below:



Now you simply need to use the following function call:

```
mydata <- read.csv('c:/datafile.csv')
```

Note that the default setting in this case is that `header = TRUE`, so the column names from your excel file will be automatically incorporated into your R data frame. If you do not have column names in your excel file, just set `header=FALSE`.

Importing from other Statistical Software:

R also has the ability to import data files from other statistical software packages, such as Minitab, SPSS, STATA, SAS, Octave, and more. In this tutorial we will look at SAS and SPSS.

Unlike `read.table`, `read.delim`, and `read.csv`, the functions needed to import the above file types aren't built into R, we will need to call a package that contains these functions:

```
library('foreign')
```

This package should be included in most R installations. If the above command doesn't work, you can install the package to your machine via CRAN (the Comprehensive R Archive Network) with the install.packages function:

```
install.packages('foreign')
```

After this command you will need to select a CRAN mirror to install from and then you will be able to call the library function to load the package as previously shown. Note that there are many useful R packages that can be downloaded in this way. For a full list, go to http://cran.r-project.org/.

Now that we have the package loaded, let's import a data file from SAS. Currently, SAS datasets cannot be directly imported into R in any other format. Therefore, we recommend exporting the SAS dataset to either the .xpt format or to .csv format and then importing to R. To import a SAS data set into R, first step export the data from SAS as a SAS XPORT file. After that, we can simply use the following function:

```
mydata <- read.xport('c:/datafile.xpt')
```

Importing data from SPSS is just as simple:

```
mydata <- read.spss ('c:/datafile.spss')
```

One nice feature about importing SPSS data files is that factors (or categorical variables) are automatically coerced to factors when the data set is imported into R. Also note that

`read.spss` has a few options you can use. Use `?read.spss` if you desire information about them.

The foreign package includes other functions like this for STATA (`read.dta`), Octave (`read.octave`), and others which are just as simple to use. For full details, type:

```
help(package=foreign)
```