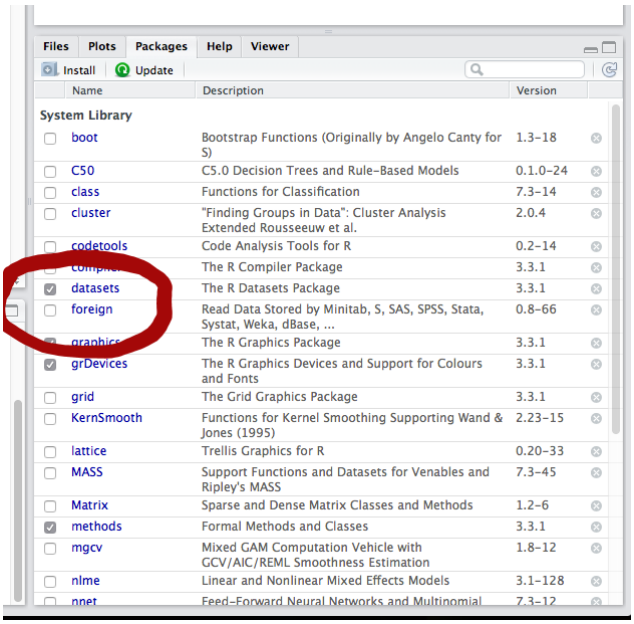
Read data into R

One of the advantages of R when compared to a sophisticated programmable calculator is that you can import data into R for analysis and manipulation. The data import functions avoid all of the keyboard entry and its potential for entry errors.

The default R datasets included in the base R distribution

The first way to import data into your R session is by using the default datasets package already included in the base R installation. If you look at the package listing in the Packages panel, you will find a package called **datasets**.



Simply check the checkbox next to the package name to load the package and gain access to the datasets. You can also click on the package name and RStudio will open a help file describing the datasets in this package.

Working through an example with one of the datasets may help you see the potential of this resource. Check the package checkbox and load the package. Now, click the package name and browse the datasets package help file. View the details on the **cars** dataset [click the dataset name to view the dataset details]. Type **cars** at the Command console prompt. R will output the contents of the cars dataset [50 pairs of values with the column headings of speed and dist]. Enter this command

plot(cars, main = "cars data - speed vs dist")

and press **ENTER**. R plots a scatter plot of the cars dataset with the speed variable as the x-axis and the dist variable as the y-axis as shown below.

While you can treat the dataset name of these package datasets as R objects, as we did in the example above, it is not advisable to do that. Usually, you will want to manipulate the data and save your results. The best way to begin using one of the datasets in this package is to assign its values to an object and then work with that object and not the dataset. Here is an example:

```
c = cars
plot(c, main = "cars data - speed vs dist")
```

This will produce the same plot as before, but now you can manipulate the data in the object **c** without modifying the default dataset.

Read an R dataset

Another type of dataset you may encounter is an R dataset. These are datasets created in R and saved with a **.RData** extension. R saves the data in a binary format. This format saves storage space but restricts usage exclusively to the R environment.

Assuming that you have an R data file named **MyData.RData** in your default file folder, you can load the file into an R object with this command:

```
x = load('MyData.RData')
```

Your data will now be in the object **x** ready for you to work with. Notice that the R data file name is enclosed in quotes [either single or double quotes are acceptable]. If you forget the quotes, R will respond with an error message. The **load( )** function is case sensitive, so be aware of the capital letters in your file name.

Read a text file

Many datasets you may encounter will be stored as text files. One reason many datasets are stored as text files is because the data can be used in any application with minimal issues. You can open, view, and edit text files using a text editor or Excel. R has built in functions to read text files into your session so they are easy to import into R. The **read.table( )** function will read most text files into your R session. The only issue with this function is that you have to tell it several pieces of information about your dataset so that it can import the data correctly.

Assume that you have a text dataset named **MyData.txt** that contains this data:

x, y, z

43, 13, 5

38, 12, 3

40, 14, 6

41, 12, 4

43, 13, 5

You can import this data into an object named **example** using this command:

example = read.table('MyData.txt', header = TRUE, sep = ",")

This function has three arguments. The first argument identifies the name of the file that you want to read into R enclosed in quotes. The second argument tells the function that your file has a header. In this example you have a header consisting of the names **x, y,** and **z**. The third argument tells the function what symbol is used to separate the data elements. In this example you have commas as the separator symbol.

This is a good place to discuss separator symbols. Most text files use some standard separators. These include a space character, or a tab character, or a comma character. Each row of data ends in a new line character. Some text files use other separator characters, but these are rare and will be addressed later. The example above uses commas. You communicate that to **read.table( )** using the **sep =** argument. You enclose the separator symbol in quotes. If the separator symbol is a space character, or a tab character, you would use a space enclosed in quotes, like this: **sep = " "**. While you can use either single or double quotes around your separator character, double quotes are clearer when you indicate a space character for your separator. The **read.table( )** function is case sensitive, so be aware of the capital letters in your file name.

Normally, the **read.table( )** function is used for space or tab separated files because R includes specialized functions for many other separator types and those specialized functions are usually used in those cases.

Read a csv file

One common type of text file you may encounter is a **comma separated** or **.csv** file. In this text file format the individual data elements are separated with commas. Additionally, each row of data is divided by line returns. While the **MyData.txt** example above is a comma separated data file, it does not conform to the accepted file name standard. Normally, coma separated text files use a **.csv** file name extension. These text files use this special file name extension because comma separated files are used so often that many applications, including R, have dedicated functions to import the data from a .csv file into the application storage and the special file name extension helps trigger these import functions.

You can import a csv file in R with the command:

ir = read.csv('Iris_Data.csv')

Your data will now be in the object **ir** ready for you to work with. Notice that, as with the load( ) and read.table( ) functions above, the R data file name is enclosed in quotes [either single or double quotes are acceptable]. If you forget the quotes, R will respond with an error message. The **read.csv( )** function is case sensitive, so be aware of the capital letters in your file name.

Other input file types

One of the advantages of R being an open source application is that users can develop functions for specialized needs and make them available for others to use. Many data file import functions exist for specialized file types. There are several variations of **read.___( )** that can read specialized file formats such as dcf, fwf [fixed-width format], and DIF. You can install R packages with functions that read Excel files, javascript JSON files, XML data, HTML files, or directly from websites. There are R packages that enable you to directly import data from files produced by proprietary analysis applications such as SPSS, SAS, Stata, Systat, or Minitab. You can even read data into R from many database systems. There are few limitations in R on your potential of data sources for your analytical needs.