APPM2720 Week5 Lecture 5

Reading and writing data in R

Reading data into R can range from being very easy to being a project on its own. This lecture covers a few different approaches.

Data that is already in R format

This is easy. If the data set is part of a package then use the library function to load the package and the data function to load the specific data set.

```
library(dataWorkshop)
data( AudiA4)
```

If the data is already in R format in a file then use the load function.

```
load("BoulderTemperature.rda")
```

Note that this function needs quotes around the filename while the data command does not.

Saving a dataset in R format

This is very useful after you worked to read in a complicated data file oor have spent some time modifying a dataset into a new form. Use the `save` function and it is good to use an extension such as `.rda` to indicate it is a binary R data file.

```
newDataSet<-  (BoulderJuneTemperature - 32) * (5/9)
save( newDataSet, file="newDS.rda")
```

Now the next time you open R `load("newDS.rda")` will load this into your workspace.

See also write.table function to write out a data frame in text format.

Reading in a simple data set in text format.

In the Week5 folder is the text file `First1000Primes.txt` This is just a text file and has the primes in 100 rows of 10 numbers each. For this very simple format just use `scan`.

```
primeData<- scan("First1000Primes.txt")
```

The scan function is both simple but can also be the "nuclear option" if a really complex data set needs to be read into R and then subsequently cleaned up using other functions.

Reading a data frame

The file `WorldBankData.txt` is a text file that can be used to read in the data frame `WorldBankCO2` from the dataWorkshop package. The first three lines of this file are:

```
 "GDP.cap" "Pop.mid" "Pop.urb" "CO2.cap" "Pop"
"Algeria" 1785.071723 60.68126966 59.04 164168.334 30032758
"Angola" 656.4225592 50.40936817 48.84 6582.411 13500820
```

The read.table function is designed to handle the row and column names for the data set.

```
testData<- read.table( "WorldBankData.txt")
```

Avoiding factors

When character data is encountered by **read.table** is converted to a factor object. To avoid this use the option `stringsAsFactor=FALSE`

What does a factor look like? Below is an example that creates and manipulates a factor data object.

```
 someData<- c("red", "blue", "green","blue",
             "red","scarlet","green", "blue", "red")

 testF<- as.factor( someData)
 print( testF)

 [1] red     blue    green   blue    red     scarlet
 [7] green   blue    red
 Levels: blue green red scarlet

 # extracting the parts of this
 testID<- as.numeric( testF) # the integer IDs for each data values
 print( testID)
 [1] 3 1 2 1 3 4 2 1 3

 testLevel<- levels( testF) # character tags
 print( testLevel)
 [1] "blue"    "green"   "red"     "scarlet"

# to reproduce just the character version
 testLevel[testID]
```

Reformatting to a data frame or matrix

Suppose the data set is supposed to have three columns but the file `data.txt` looks like:

```
1 2  3   5 5   5
4 4 6.5 7 8.5 20
```

(i.e. two rows to each line of the file).

You can read this with just

```
D1<- scan("data.txt")
```

and then reformat as a matrix: `D2<- matrix( D1, ncol=3, byrow=TRUE)`

To convert to a data frame

```
D3<- as.data.frame( D2)
```

Other ways to read data

Here are some tips to read in other formats:

- Use `read.cvs` for data that is separated by commas. This is handy if the data is written out or saved in csv format from an Xcel spreadsheet
- Use `read.fwf` to read in data where you want to control *exactly* how one counts spaces and interprets the numbers and characters. Usually this is a last resort if free format reading is not possible.
- There are many R packages and functions to read data directly from the web and to extract tables from web pages. See for example the package XML.
- The nuclear option: Read in the file where the entire line is a single character string. Then chop up each string into the informaion. Coerce the parts that are numbers from the character string using **as.numeric**. In this effort the function **substr** is useful for grabbing just part of a string and setting `sep="\n", what="a"` will read an entire line as a character string.

Tips on reading complicated data

- Keep all your steps in an R script
- Sometimes it is easiest to just edit the few places where the reading break down. I this case keep the old line as a comment (e.g. with **#**) so it is documented.
- Break up the reading into simpler steps.
- One of the most common problems is expecting a single character string but it a name with a space. E.g. **North Carolina** as a state name when reading in a data frame.
- Try avoid "hard coding" in details that might change or are used more than once. For example, the file name could be set as character and the number of lines another variable. Instead of `tempD<- scan("dataF.txt", skip=5)` try

```
fileName<- "dataF.txt"
lines2Skip<- 5
tempD<- scan(fileName, skip=lines2Skip)
```