Introduction to R Programming - Part B

Contents

1 Basic R editing Script Editor and `edit()` **1**

2 Changing GUI options **1**

3 Embedded Datasets **1**

4 Time and Date functions **3**

5 Removing items **4**

6 Saving and Loading R Data Objects **4**

7 Quitting the R environment **5**

1 Basic R editing Script Editor and `edit()`

R has an inbuilt script editor. We will use it for this class, but there are plenty of top quality Integrated Development Environments out there. (Read up about RStudio for example.)

To start a new script, or open an existing script simply go to File on the menu bar and click the appropriate options. A new dialogue box will appear. You can write and edit code using this editor. To pass the code for compiling press the run line or selection option (The third icon on the menu). Another way to edit code is to use the `edit()` function which operates directly from the command line. To edit the code defining an object X, simply type edit(X).

2 Changing GUI options

We can change the GUI options using the GUI preferences option on the Edit menu. (Important when teaching R) A demonstration will be done in class.

3 Embedded Datasets

Several data sets intended as learning tools are automatically installed when `R` is installed. Many more are installed within packages to complement learning to use those packages. One of these is the famous "*iris*" data set, which is used in many data mining exercises.

- To see what data sets are available type `data()`.
- To load a data set, simply type in the name of the data set.
- To specify that a specific data set is to be used for analysis, use the command `attach()`.

Some data sets are very large. To just see the first few rows, we use the head() function.

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

Dimensions of a data set

We have remarked that some data sets are very large. This is perhaps a good place to consider summary information about data objects. For a simple vector a useful command to determine the length (remark: sample size) is the function `length()`.

```
> Y=5:9
> length(Y)
[1] 5
```

For more complex data sets (and data frames, which we will see later) we have several tools for assessing the size of data.

```
> dim(iris)  # dimensions of data set
[1] 150   5
> nrow(iris) # number of rows
[1] 150
> ncol(iris) # number of columns
[1] 5
```

We can also determine the row names and column names using the functions `rownames()` and `colnames()`. If there are no specific row or column names the command will just return the indices.

```
> colnames(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

The command `summary()` is one of the most useful commands in R. It is a generic function used to produce result summaries of the results of various functions. The function invokes particular methods which depend on the class of the first argument. In other words `R` picks out the most suitable type of summary for that data.

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
>
```

`summary()` is particularly useful for manipulating data from more complex data objects. Working directories You can change your working directly by using the appropriate options on the File menu. To determine the current working directory you can use the `getwd()` command:

```
> getwd()
[1] "C:/Users/Kevin/Documents"
```

To change the working directory we would use the `setwd()` command.

```
> getwd()
[1] "C:/Users/Kevin"
>
> setwd("C:/Users/Kevin/Documents")
>
> getwd()
[1] "C:/Users/Kevin/Documents"
```

4 Time and Date functions

The commands `Sys.time()` and `Sys.Date()` returns the system's idea of the current date with and without time. We can perform some simple algebraic calculations to compute time differences (i.e. to find out how long some code took to compile.

```
> X1=Sys.time()
> #Wait a few seconds
>
> X2=Sys.time()
> X2-X1
Time difference of 8.439614 secs
>
> Sys.Date()
[1] "2012-09-01"
```

Coming unstuck

If you are having trouble with a piece of code that is currently compiling all you have to do is press ESC. Just like other computing environments.

Listing all items in a workspace

To list all items in an R environment, we use the `ls()` function. This provides a list of all data objects accessible.

```
> ls()
 [1] "a"          "A"          "authors"    "b"          "books"
 [6] "C"          "D"          "ex1"        "Gerb"       "Lst"
[11] "m"          "m1"         "op"         "presidents" "r"
[16] "showSmooth" "sm"         "sm.3RS"     "sm2"        "sm3"
[21] "Trig"       "Vec1"       "x"          "X"          "x.at"
[26] "x1"         "x2"         "x3R"        "y"          "Y"
[31] "y.at"
```

5 Removing items

Sometimes it is desirable to save a subset of your workspace instead of the entire workspace. One option is to use the `rm()` function to remove unwanted objects right before exiting your R session; another possibility is to use the save function.

The save function accepts multiple arguments to specify the objects you wish to save, or, alternatively, a character vector with the names of the objects can be passed to save through the `list=` argument.

Once the objects to be saved are specified, the only other required option is the `file=option`, specifying the destination of the saved R object. Although there is no requirement to do so, it is common to use a suffix of `.rda` or `.RData` for saved R workspace files.

For example, to save the R objects x, y, and z to a file called `myData.rda` ,the following statements could be used:

```
> save(x,y,z,fil= mydata.rda)
```

6 Saving and Loading R Data Objects

In situations where a good deal of processing must be used on a raw dataset in order to prepare it for analysis, it may be prudent to save the R objects you create in their internal binary form.

One attractive feature of this scheme is that the objects created can be read by R programs running on different computer architectures than the one on which they were created, making it very easy to move your data between different computers.

Each time an R session is completed, you are prompted to save the workspace image, which is a binary file called `.RData` in the working directory.

Whenever R encounters such a file in the working directory at the beginning of a session, it automatically loads it making all your saved objects available again. So one method for saving your work is to always save your workspace image at the end of an R session. If youd like to save your workspace image at some other time during your R session, you can use the `save.image()` function, which, when called with no arguments, will also save the current workspace to a file called .RData in the working directory.

7 Quitting the R environment

As the front page text indicates all you have to do to quite the workspace is to type in `q()`. You will then be prompted to save your work.