Import Read And Export Data

Pavan Kumar
Senior Project Engineer
Big data Analytics Team
C-DAC KP

Reading and Getting Data into R

- Most often, you will have to deal with large sets of data which are in the form of CSV or TSV formats.

- To perform analysis on such files, you have to import/get that data into R console.

- **Commands to be discussed**
  - c() : Used to combine or concatenate data
  - scan() : Used to read large datasets and retrieve data from CSV files.
  - read.csv(), read.table(), write.csv(), write.table() : Used to read and write from csv files and tables respectively

Reading And Combining Numerical Data

Using the c() command

- The c() command is used to concatenate or combine two or more values.
- **Syntax**

```
### sampleitem1, sampleitem2, sampleitem3 are combined
c(sampleitem1, sampleitem2, sampleitem3)

## putting all combined values into new object
CombinedResult<-c(sampleitem1, sampleitem2, sampleitem3)
```

- **Reading and Combining Numerical Data**

```
### Entering the numeric values using the c() command
Result = c(678,876,566,655,74,456,6543,56,45,675,7467,567,868)

### To print the result
Result
```

USING THE C() COMMAND

- Executing on R
- Here, we have passed numerical values within the parentheses of c() command with comma separation.
- The values are stored in the new object called "Result" and to print the values on the R, we are entering the name of the object

```
> ### Entering the numeric values using the c() command
> c(678,876,566,655,74,456,6543,56,45,675,7467,567,868)
 [1]  678  876  566  655   74  456 6543   56   45  675 7467  567  868
> ## putting all combined values into new object
> Result<-c(678,876,566,655,74,456,6543,56,45,675,7467,567,868)
> ###To print the result
> Result
 [1]  678  876  566  655   74  456 6543   56   45  675 7467  567  868
> 
```

USING THE C() COMMAND

- Incorporating existing data objects with the new values.

```
> Result
 [1]  678  876  566  655   74  456 6543   56   45  675 7467  567  868  768  789  667
> Result1
[1]  111 1111 1111 1111
> ResultFull<-c(123,123,123,Result, Result1)
> ResultFull
 [1]  123  123  123  678  876  566  655   74  456 6543   56   45  675 7467  567  868  768  789  667  111 1111 1111 1111
> 
```

- Here, we are adding some values (123,123,123) to the existing values that are stored in objects Result and Result1

Reading And Combining Text Data

Using The c() Command

- The text data is entered using quotes.
- There is no difference between the single and double quotes as R converts all the quotes to double quotes.
- You can use either single or double or combination of quotes as shown in the syntax.
- **Syntax**

```
c('sampleitem1', 'sampleitem2', 'sampleitem3')
c("sampleitem1", "sampleitem2", "sampleitem3")
c("sampleitem1", 'sampleitem2', 'sampleitem3')
```

Combining And Reading Text Data

- Reading test data on R console

```
> empnames<-c("Smith","kate","Johanathan","Reddy","James","Alan","John",
+ "Ricky","Shaun","Charles","Andrew","Micheal")
> empnames
 [1] "Smith"      "kate"       "Johanathan" "Reddy"      "James"      "Alan"       "John"       "Ricky"      "Shaun"
[10] "Charles"    "Andrew"     "Micheal"
> 
```

- Adding more data to the existing data

```
> empnames<-c("Smith","kate","Johanathan","Reddy","James","Alan","John",
+ "Ricky","Shaun","Charles","Andrew","Micheal")
> empnames
 [1] "Smith"      "kate"       "Johanathan" "Reddy"      "James"      "Alan"       "John"       "Ricky"      "Shaun"
[10] "Charles"    "Andrew"     "Micheal"
> ##Adding more names to existing data
> newempnames<-c(empnames, "Pavan","Ram","Tom")
> newempnames
 [1] "Smith"      "kate"       "Johanathan" "Reddy"      "James"      "Alan"       "John"       "Ricky"      "Shaun"
[10] "Charles"    "Andrew"     "Micheal"    "Pavan"      "Ram"        "Tom"
> 
```

Reading Numeric And Text In R

- When text and numbers are combined, the entire data object becomes a text variable and the numbers are also converted to text.
- Reading both text and numeric data in R

```
> combine
 [1] "678"      "876"      "566"      "655"      "74"       "456"      "6543"     "56"       "45"
[10] "675"      "7467"     "567"      "868"      "768"      "789"      "667"      "34"       "5"
[19] "6"        "6"        "7"        "Smith"    "kate"     "Johanathan" "Reddy"  "James"    "Alan"
[28] "John"     "Ricky"    "Shaun"    "Charles"  "Andrew"   "Micheal"  "Pavan"    "Ram"      "Tom"
> |
```

- Note: Here, numeric data is shown in the double quotes like that of text data.

Using The scan() Command

- The c() command is used only for reading and combining of small data. But this can be tedious when lot of typing is involved.
- In c() command, all the values are separated by , (comma) to make a data object.
- The same can be done with out using commas through the scan() command.

1. After entering the scan() command and press ENTER, console will be waiting for the desired data.
2. User can type the data and DOUBLE press ENTER, your data is shown on the console

```
> scan()
1: 10
2: 20
3: 30
4: 40
5: 30
6: 40
7: 50
8: 50
9:
Read 8 items
[1] 10 20 30 40 30 40 50 50
> 
```

USING THE SCAN() COMMAND- READING

- Reading the numeric values using the scan() command.

1. After entering the empsalaries<-scan() command and press ENTER, console will be waiting for the desired data.
2. User can type the data and DOUBLE press ENTER, your data is shown on the console
3. To view the stored values, object name "empsalaries" is typed

```
> empsalaries<-scan()
1: 25000
2: 25000
3: 25000
4: 35000
5: 38000
6:
Read 5 items
> empsalaries
[1] 25000 25000 25000 35000 38000
>
```

USING THE SCAN() COMMAND- READING

- Reading the text data using scan() command
- Syntax here depicts that user is specifying that the data that has to be entered will be characters and not numbers.

```
> scan(what='character')
1: Ricky
2: Tom
3: Charles
4: Pavan
5: Alan
6: Ram
7: Harry
8: Andrew
9: Micheal
10: Samuel
11: Williams
12:
Read 11 items
 [1] "Ricky"    "Tom"      "Charles"  "Pavan"    "Alan"     "Ram"      "Harry" $
>
```

Reading The Data Of a File From Disk

- Using the scan() command, you can also read the data from files.
- The scan() command can read data in a vector or list from the console or file.
- To read a file using scan() command, add `file=`filename`` to the command as shown

```
## Reading data from the file called sample.txt
readdata<-scan(file='sample.txt')
```

- Now, the contents of `sample.txt` file is stored in `readdata` object.
- File name should be enclosed with in the quotation marks

Reading The Data Of a File From Disk

- On execution of the command, R will look for the sample.txt file in the current working directory.
- To know the current working directory and to change the directory, use following commands

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

```
> getwd()
[1] "D:/"
> setwd("D:/DBDA/")
> getwd()
[1] "D:/DBDA"
>
```

```
> dir()
 [1] "~$ links.docx"
 [3] "~$deleSyllabus.docx"
 [5] "~$gdataNoida.docx"
 [7] "~WRL0001.tmp"
 [9] "Big data and Analytics  - courses-info-from-Deity-1
[11] "Functions.pptx"
[13] "IITSyllabus.docx"
[15] "ImportReadExport.pptx"
[17] "KP-pgDBDA-feb2016-faculty-plan-v1.pdf"
[19] "Manipulating_Processing_Data.pptx"
[21] "National BigData Analytics Capacity Building Progra
[23] "Noida"
[25] "Noida.zip"
[27] "PGDBDA_Team_faculties.doc"
[29] "R links.docx"
[31] "RJosephAdler"
[33] "Ses3_3_ApacheHive_Pig.ppt"
[35] "Source Book August 2015"
[37] "SurveyPeopleBD.docx"
[39] "Teaching Guidelines of Statistical Analysis with R.
> list.files()
 [1] "~$ links.docx"
 [3] "~$deleSyllabus.docx"
 [5] "~$gdataNoida.docx"
```

Reading The Data Of a File From Disk

- Using scan() command for reading from file

```
> AAA<-scan("sample.txt")
Read 10 items
> AAA
 [1] 1 2 4 5 5 6 7 6 6 7
> 
```

- The scan() command has a option of choosing the file by browsing the file system

```
scan(file.choose())
```

Note: scan(file.choose()) function will not work in Linux OS

Using The read.csv() Command

- Reading from CSV files, read.csv() command is used.
- The command read.csv() reads entire CSV file and display the contents on the R console.
- **Syntax**

```
read.csv(file, header = TRUE, sep = ",")
```

  - **file**: to specify the file name
  - **sep**: to provide the separator
  - **header**: to specify whether or not the first row of CSV file should be set as column names. Default is TRUE

Using The read.csv() Command

- Before executing the read.csv() command, file is read and saved in appropriate format CSV/XLS or TSV format

```
> read.csv(file.choose(), sep=",",)
   year sex births
1  1880 boy 118405
2  1881 boy 108290
3  1882 boy 122034
4  1883 boy 112487
5  1884 boy 122745
6  1885 boy 115948
7  1886 boy 119046
8  1887 boy 109312
9  1888 boy 129914
10 1889 boy 119044
11 1890 boy 119704
12 1891 boy 109272
13 1892 boy 131457
14 1893 boy 121045
15 1894 boy 124902
16 1895 boy 126650
17 1896 boy 129082
18 1897 boy 121952
19 1898 boy 132116
> |
```

```
> read.table(file.choose(), sep="\t")
       V1   V2       V3          V4
1   storm wind pressure        date
2 Alberto  110     1007 2000-08-03
3    Alex   45     1009 1998-07-27
4 Allison   65     1005 1995-06-03
5     Ana   40     1013 1997-06-30
6  Arlene   50     1010 1999-06-11
7  Arthur   45     1010 1996-06-17
> |
```

Importing Data from fwf

- Reading data from FWF (fixed width format) in to a dataframe.
- To read data from fwf, we have **read.fwf()** function in R
- You use this function when your data file has columns containing spaces, or columns with no spaces to separate them.
- **Syntax :**

```
read.fwf(file, width="",col.names="")
```

- **Example**

```
read.fw("fwf.txt", widths=c(4,-10,-3,1,-2,2),col.names=c("Subject","Gender","Marks"))
```

Importing Excel Spreadsheets Into R

- From the base R, you will not able to import Excel file directly.
- Package to be installed is **xlsx**, **openxlsx** package.

- **Reading Excel Spreadsheets into R From The Clipboard**
  - Functions used in R are **read.table(file=clipboard)**

- **You can conver Excel file to CSV file and import in R using read.csv()**

IMPORTING JSON FILES INTO R

- Package used for importing json files in to R is rjson.
- Library need to load is jsonlite
- Function used is **fromJSON**
- Three procedures under fromJSON() :
  simplifyVector, simplifyDataFrame and simplifyMatrix

install.packages("rjson")
library(jsonlite)

JSON structure	Example JSON data	Simplifies to R class	Argument in fromJSON
Array of primitives	`["Amsterdam", "Rotterdam", "Utrecht", "Den Haag"]`	Atomic Vector	simplifyVector
Array of objects	`[{"name":"Erik", "age":43}, {"name":"Anna", "age":32}]`	Data Frame	simplifyDataFrame
Array of arrays	`[ [1, 2, 3], [4, 5, 6] ]`	Matrix	simplifyMatrix

IMPORTING JSON FILES INTO R

- Simple commands

```
> json <- '["Mario", "Peach", null, "Bowser"]'
> fromJSON(json)
[1] "Mario"  "Peach"  NA       "Bowser"
> json <-
+ '[
+   {"Name" : "Mario", "Age" : 32, "Occupation" : "Plumber"}
+   {"Name" : "Peach", "Age" : 21, "Occupation" : "Princess"
+   {},
+   {"Name" : "Bowser", "Occupation" : "Koopa"}
+ ]'
> mydf <- fromJSON(json)
> mydf
    Name Age Occupation
1  Mario  32    Plumber
2  Peach  21   Princess
3   <NA>  NA       <NA>
4 Bowser  NA      Koopa
>
```

```
> toJSON(mydf)
[{"Name":"Mario","Age":32,"Occupation":"Plu
> toJSON(mydf, pretty=TRUE)
[
  {
    "Name": "Mario",
    "Age": 32,
    "Occupation": "Plumber"
  },
  {
    "Name": "Peach",
    "Age": 21,
    "Occupation": "Princess"
  },
  {},
  {
    "Name": "Bowser",
    "Occupation": "Koopa"
  }
]
>
```

Importing Data From Databases Into R

- Packages used for importing from various databases
  - MonetDB.R
  - Rmongodb
  - RMySQL,
  - Mongolite
  - Rmongo
  - RODBC
  - Roracle
  - RPostgreSQL
  - RSQLite
  - RJDBC

Importing Data From MYSQL Into R

- Packages and library needed

```
install.packages("RMySQL")
library(RMySQL)
```

- MySQL Connection

```
con = dbConnect(MySQL(),user="training", password="training123",
dbname="trainingDB", host="localhost")
```

- Retrieving data

```
myQuery <- "select pclass, survived, avg(age) from titanic where survived=1 group by
pclass;“
dbGetQuery(con, myQuery)
```

http://www.unomaha.edu/mahbubulmajumder/data-science/fall-2014/lectures/20-database-mysql/20-database-mysql.html#/1

```
  pclass survived avg(age)
1    1st        1 36.83379
2    2nd        1 24.85870
3    3rd        1 21.54517
```

Importing Large Data Sets Into R

- Package used in data.table
- Function is fread()
- Example :

```
library(data.table)
data <- fread("textfile.txt")
```

Exporting Data From R

- After undergoing any computations in R, the data now needs to be used in reports or various other sources.
- Therefore, you need to extract data from R.
- To export data from R, we use write.csv() and write.table() functions.

USING THE WRITE.TABLE() AND WRITE.CSV() COMMAND

- The write.table() command is used to write the data stored in a vector to a file.
- The data is saved using the delimiters such as spaces or tabs as shown.
- Here, in the below screenshots, we are saving the 'births' object to the BIRTHS.csv and BIRTHS.txt in the D drive local system

```
> head(births)
  year sex births
1 1880 boy 118405
2 1881 boy 108290
3 1882 boy 122034
4 1883 boy 112487
5 1884 boy 122745
6 1885 boy 115948
> dim(births)
[1] 260   3
> write.csv(births, "D:/DBDA/BIRTHS.csv")
>
```

```
> head(births)
  year sex births
1 1880 boy 118405
2 1881 boy 108290
3 1882 boy 122034
4 1883 boy 112487
5 1884 boy 122745
6 1885 boy 115948
> dim(births)
[1] 260   3
> write.table(births, "D:/DBDA/BIRTHS.txt",
+ sep="\t")
>
```

THANK YOU!!!!!