Introduction to R – Data Importing

ANALYTIXLABS

1. GETTING SESSION INFORMATION:

```
sessionInfo()

#FOR COMMENTING r statement, USE "#" before statement
#SHORT CUT: ctrl+Shift+c
#Link- https://support.rstudio.com/hc/en-us/articles/200711853-Keyboard-Shortcuts
```

2. GETTING HELP:

a.View locally installed documentation:

```
help.start()
```

b.Help can also give you advise on syntax while Developing your own functionalities and remember quoting!

```
help("function")
```

c.Use example() to run the example code:

```
help(names)
example(names)
example(plot)
```

d.Help for packages:

```
library(help = MASS)
help(package = MASS)
```

e.Help for datasets:

```
data()
data(package = "MASS")
data(package = .packages(all.available = TRUE))
```

3.INSTALLING & MANAGING PACKAGES:

a.categorical view of packages:

```
browseURL("http://cran.r-project.org/web/views/")
```

b.Availabe packages by Name:

```
browseURL("http://cran.stat.ucla.edu/web/packages/available_packages_by_name"
)
```

c.Latest updates on existing & new packages:

```
browseURL("http://crantastic.org")
```

d.See current packages #brings editor with list of installed packages:

```
library()
```

e.Shows packages that are currently loaded:

```
search()
```

f.To install packages # download packages from CRAN and install in R:

```
install.packages("ggplot2")
```

g.Make package available ; often used for loading in scripts:

```
library("ggplot2")
```

h.Prefered for loading in functions; may be better:

```
require("ggplot2")
```

i.Brings up documentation in editor window:

```
library(help="ggplot2")
```

j.Check for pacakge updates; do it regularly:

```
update.packages("ggplot2")
```

k.Unload/Remove packages # by default, all loaded packges are unloaded when R quits:

```
detach("package:ggplot2", unload=TRUE)
```

l.To permanantly remove it:

```
remove.packages("ggplot2")
```

4. Listing out functions and structure in packages:

```
#Loading packages
require(sqldf)
#List out all the functions in the sqldf package
ls("package:sqldf")
#List out all the functions with structure in the sqldf package
lsf.str("package:sqldf")
```

5.Basics about R:

```
x<-10+20                    # Simple Math
10+20 -> x                  # "<-" Assignment Operator
x                           # Print the x value
y = 1:10                    # Assign values in a sequence
z <- c(11,12,13,14,15)          # Assigning values

print("AnalytixLabs - My First Program") # Prints data in console
```

6. DATA TYPES:

a. Numeric:

```
1.7                          # Decimal mark is a dot
7.8e3                        # One can use scientific notation
-1/0                         # Infinity is marked as inf or -inf

4/6799                       # Formatting numbers
format(4/6799, sci = TRUE) # Formatting numbers
format(4/6799, digits = 2) # Formatting numbers

round(pi, 2)                  # Rounding
```

b.Character:

```
A<- "AnalytixLabs"           # strings should be enclosed with ' or "
# Concatenation of strings
paste("AnalytixLabs", 'is #1 data science capability buidling company in Indi
a')
```

c.Factor:

```
f <- factor(c("girl", "boy", "boy", "girl", "boy"))   # Categorical variables
```

d.Special:

```
1/0                     # Inf, -Inf
-1/0                    # Inf, -Inf
```

```
log(-1)                 # Not a number
NA                      # NA - Missing values
NULL                    # NULL
```

7.Logical Operators:

```
2 == 5       #is 2 equal to 5?

(2 != 2)     # ! means negation
!(2 == 2)    # ! means negation

(2 == 2) & (5 == 5)     # & - logical conjunction
(2 == 2) | (5 != 5)     # | - logical alternative
```

8. Dynamic Evaluation using paste function:

```
a<-10 ; b<-30
MyText<-paste("sum(",a,",",b,")")
cat("The value after the evaluation of the above expression is: ",eval(parse(
text=MyText)))
?parse()
```

9. Dates in R:

```
# Dates are represented as the number of days since 1970-01-01, with negative
values for earlier dates. Use as.Date( ) to convert strings to dates
mydates <- c("2007-06-22", "2004-02-13")
mydates1 <- c("22/Jan/2015", "10/Oct/2015")
mydates2<- as.Date(mydates1, "%d/%b/%Y")
str(mydates1)
format(mydates2, "%A, %d-%B-%Y")

# number of days between today and mydates2
days <- Sys.Date() - mydates2

Sys.Date() # returns today's date.
date()     # returns the current date and time.

# Formats/informats Example.
# print today's date
today <- Sys.Date()
unclass(today)
format(today, format="%B %d %Y")
format(today, format="%b %d %y")
```

10. DATA OBJECTS IN R:

a. Vectors:

```
# Vectors # set of objects of the same type, one dimensional ----------------
-------------------------------------------------

my_vector <- c(-1, 2, 6, 6.7, 2, 0.45, 2, 4)  # Numeric vector

mode(my_vector)
typeof(my_vector)
class(my_vector)
is.numeric(my_vector)

x5 <- c(5, 4, 2, 3, 1,6,8,9)   #Numeric Vector
x7 <- c(3, 4, 2, 3)
x8 <- c("HA", "DL","MH","AP")  # Character Vector

x5+x7                          # sum of vectors
data <- c(x5,x7)               # concatenation of vectors

(my_vector2 <- 2 * my_vector)  # Computations with numeric vectors
my_vector * c(2, 3)
my_vector + c(2, 3, 4)

# Character vector
(my_char_vector <- c(4, 2, "Hello, world!"))

mode(my_char_vector)
typeof(my_char_vector)
class(my_char_vector)
is.numeric(my_char_vector)

# Mathematical operations
sqrt(my_vector)        # NaN (not a number) produced as a resulting of squart
ing negative values
log(my_vector)         # etc.
sum(my_vector)         # sum
prod(my_vector)        # product
cumsum(my_vector)      # cumulates (sums up) all the values
cumprod(my_vector)     # multiplies up all the values

# Useful functions related to vectors
length(my_vector)     # vector's length
unique(my_vector)     # distinct values of the vector
head(my_vector, 3)    # displays first 3 elements of my_vector
tail(my_vector, 2)    # displays last 2 elements of my_vector
sort(my_vector)       # sorts my_vector
order(my_vector)      # returns the order of my_vector
```

```
rev(my_vector)        # reverts the order of elements in my_vector
which.max(my_vector)  # return position of the max value
which.min(my_vector)  # return position of the min value
rank(my_vector)       # rank elements

# Indexing
my_vector[2]
my_vector[-2]
my_vector[1:3]
my_vector[-(1:7)]
my_vector[c(2, 5)]
my_vector[my_vector > 3]
which(my_vector > 3)
my_vector[my_vector > 1 & my_vector < 4]
my_vector[my_vector %in% 1:3]
match(my_vector, 1:3)
my_vector[c(rank(my_vector)) == 2]
my_vector[order(my_vector)]


# Combining two vectors(Appending)
cbind(my_vector, my_char_vector)     #APPENDING COLUMNS - MATCH MERGING
rbind(my_vector, my_char_vector)     #APPENDING ROWS

# Sequences
4:7
seq(0, 1, length.out = 16)
seq(1, 9, by = 2)
seq(1, 8, by = 2)
seq(9, 1, by = -2)
seq(17)

all(seq(17) == 1:17)

rep(1:4, 2)
rep(1:4, each = 2)
rep(1:4, c(2, 1, 2, 1))
rep(1:4, each = 2, len = 10)
rep(1:4, each = 2, times = 3)
```

b. Matrices:

```
# Defined with a matrix() function with 3 arguments: vector of values, number
of rows, number of columns

(my_matrix <- matrix(c(1, 2, 3, 4), 2, 2))
(my_matrix <- matrix(c(1, 2, 3, 4), 4, 1))
(my_matrix <- matrix(c(1, 2, 3, 4), 2, 2, byrow=TRUE))
```

```
my_matrix[1, 1]
my_matrix[, 1]
my_matrix[1, ]

dim(my_matrix)                # Dimesnion of matrix
det(my_matrix)                # Determinant
eigen(my_matrix)              # Eigenvalues/Eigenvectors
(t(my_matrix) %*% my_matrix)  # Transpose and multiply

rowSums(my_matrix)            # Sum of rows
colSums(my_matrix)            # Sum of colums
rowMeans(my_matrix)           # Avgs of rows
colMeans(my_matrix)           # Avgs of colums
```

c. Arrays:

```
# Defined with a array() function with 3 arguments: vector of values, dimensi
ons, dimension names

dim_names <- list(dimension1 = c("A", "B", "C", "D"),
                  dimension2 = c("a", "b", "c"),
                  dimension3 = c("1", "2"))
(my_array <- array(data = 1:24, dim = c(4,3,2), dimnames = dim_names))

dim(my_array)

# Indexing
my_array[3, 2, 1]
my_array[3, 2, ]
my_array[3, , ]
```

d. List:

```
# Set of objects that can have different types # Defined with the list() func
tion
(my_list <- list(name = c("Analytix", "Labs"), age = 3, club = "Alabs", matri
x = my_matrix))
mode(my_list)
typeof(my_list)
class(my_list)

# Indexing
dim(list)
my_list$name
my_list[1]
my_list[[1]]
my_list[[1]][2]
names(my_list)      #lists all names of the list
```

e. Data Frame:

```
# table with the same type within a column and different types between column
s # defined with a data.frame() function
my_df <- data.frame(id = c(1, 2, 3),
                    name = c("Analytics", "Labs", "alabs"),
                    Goals = c(50, 49, 25))
dim(my_df)
str(my_df)
summary(my_df)
```

f. Built in data objects(datasets):

```
?datasets                        # Using R's built in data sets

library(help=datasets)
library(datasets)

data(mtcars)                    # Loading mtcars data set
cars <-mtcars                   # Save the data into workspace
detach(package:datasets)        # To remove the datasets package
```

g. Basic functions of data frame:

```
# Viewing data set
mtcars                          # Total data set in console
View(mtcars)                    # Viewing dataset in spreadsheet
head(mtcars,10)                 # Viewing top-10 observations (default: top-6)
tail(mtcars)                    # Viewing bottom 10 observations
str(mtcars)                     # Viewing data dictionary
names(mtcars)                   # Viewing column names

v1 = mtcars$disp
newvar <-mtcars$disp + mtcars$hp
v1 <- mtcars$mpg                # Assigning single variable from mtcars data to
v1
v2 <- mtcars$cyl
v3 <- mtcars$disp
v4 <- mtcars$hp

mtcars1<-rbind(v1,v2,v3,v4) # Combined as rows #Horizontal joins
mtcars2<-cbind(v1,v2,v3,v4) # Combined as columns # Vertical joins
```

11. IMPORTING DATA:

a. Import from flat file:

```
data.txt <- read.table("C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation
Training\\1. R Foundation\\Data Sets\\Data_txt.txt",sep = "\t",header = TRUE)
```

```
#data.txt <- read.table("Data_txt.txt",sep = "\t",header = TRUE)

data_delim.csv <- read.delim("C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation Training\\1. R Foundation\\Data Sets\\Data_txt_delim.txt",sep = "@",header = TRUE)
#Data.User <- read.table(choose.files(),header=TRUE)
```

b. Reading CSV File:

```
data_table.csv <- read.table("C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation Training\\1. R Foundation\\Data Sets\\Data_csv.csv",sep = ",",header = TRUE)

data_csv.csv <- read.csv("C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation Training\\1. R Foundation\\Data Sets\\Data_csv.csv",header = TRUE)

data_delim.csv <- read.delim("C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation Training\\1. R Foundation\\Data Sets\\Data_csv.csv",sep = ",",header = TRUE)
```

c. Reading EXCEL File:

```
#install.packages("XLConnect")
require(XLConnect)
data.Xlsx1 <- readWorksheet(loadWorkbook("C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation Training\\1. R Foundation\\Data Sets\\Data_Xlsx.xlsx"),sheet=1)
View(data.Xlsx1)

#install.packages("xlsx")
require(xlsx)
data.XlsX2 <- read.xlsx(file="C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation Training\\1. R Foundation\\Data Sets\\Data_Xlsx.xlsx", sheetIndex=1)
```

d. Reading SAS File:

```
library(sas7bdat)
data.Sas <- read.sas7bdat(file="C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation Training\\1. R Foundation\\Data Sets\\ccu.sas7bdat", debug=FALSE)
View(data.Sas)
```

e. Reading WEB File:

```
fpe <- read.table("http://data.princeton.edu/wws509/datasets/effort.dat")
```

f. Reading other types of statistical data files:

```
require(foreign)
```

```
# SPSS files:
dat.spss <- read.spss("C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation
Training\\1. R Foundation\\Data Sets\\hsb2.sav", to.data.frame=TRUE)

# Stata files:
dat.dta <- read.dta("C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation Tr
aining\\1. R Foundation\\Data Sets\\hsb2.dta")
```

g. Reading Semi Structured data(.xml files)::

```
# Load the package required to read XML files.
library("XML")

# Also load the other required package.
library("methods")

# Give the input file name to the function.
result <- xmlParse("emp.xml")
#result <- xmlTreeParse("emp.xml", useInternalNodes = TRUE)
#sapply(getNodeSet(result, "//variable"), xmlValue)

# Print the result.
print(result)

# Exract the root node form the xml file.
rootnode <- xmlRoot(result)

# Find number of nodes in the root.
rootsize <- xmlSize(rootnode)

# Print the result.
print(rootsize)

# Exract the root node form the xml file.
rootnode <- xmlRoot(result)

# Print the result.
print(rootnode[1])
```

h. Reading Semi Structured data(.json files):

```
# Load the package required to read JSON files.
library("rjson")

# Give the input file name to the function.
result <- fromJSON(file="emp.json")

# Print the result.
print(result)
```

```
# Convert JSON file to a data frame.
json_data_frame <- as.data.frame(result)

print(json_data_frame)
```

i. Download data from web:

```
# Gather the html links present in the webpage.
links <- getHTMLLinks("http://www.geos.ed.ac.uk/~weather/jcmb_ws/")

# Identify only the links which point to the JCMB 2015 files.
require(stringr)
filenames <- links[str_detect(links, "JCMB_2015")]

# Store the file names as a list.
filenames_list <- as.list(filenames)

# Create a function to download the files by passing the URL and filename lis
t.
downloadcsv <- function (mainurl,filename){
        filedetails <- str_c(mainurl,filename)
        download.file(filedetails,filename)
        }

# Now apply the l_ply function and save the files into the current R working
directory.
require(plyr)
l_ply(filenames,downloadcsv,mainurl="http://www.geos.ed.ac.uk/~weather/jcmb_w
s/")
```

12. EXPORTING DATA:

a. Writing Text File:

```
write.table(fpe, file = "C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundatio
n Training\\1. R Foundation\\Data Sets\\Write_Text.txt", sep=",")
```

b. Writing CSV File:

```
write.csv(fpe, file = "C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation
Training\\1. R Foundation\\Data Sets\\Write_csv.csv")
```

c. Writting Excel File:

```
library(xlsx)
write.xlsx(fpe, file="C:\\Users\\ChandraMouli\\Desktop\\R-Hadoop Foundation T
raining\\1. R Foundation\\Data Sets\\Write_Excel.xlsx")
```

13. CONNECTING TO DATA BASE:

```
#Important packages for connecting databases #RODBC #RMySQL #RJDBC #DBI

#Important functions in RODBC Package

#Function                              Description
#odbcConnect(dsn, uid="", pwd="")   Open a connection to an ODBC database
#sqlFetch(channel, sqtable)       Read a table from an ODBC DB into a data fr
ame
#sqlQuery(channel, query)           Submit a query to an ODBC DB and return r
esults
#sqlSave(channel, mydf, tablename = sqtable, append = FALSE)    Write or upda
te (append=True) a data frame to a table in the ODBC database
#sqlDrop(channel, sqtable)        Remove a table from the ODBC database
#close(channel)                   Close the connection

# RODBC Example
# import 2 tables (Crime and Punishment) from a DBMS
# into R data frames (and call them crimedat and pundat)

library(RODBC)
myconn <-odbcConnect("ANALYTIXLABS")
?odbcConnect
Transactions <- sqlFetch(myconn, "Transactions")
Transactions1 <- sqlQuery(myconn, "select * from Transaction")
View(Transactions)
close(myconn)

# Connecting to MS-ACCESS

# Load RODBC package
library(RODBC)

# Connect to Access db
channel <- odbcConnectAccess2007("Car_data_oth_prices.accdb")

# Get data
data <- sqlQuery( channel , paste ("select *  from Car_data_oth_prices"))

# Get data
data <- sqlQuery( channel , paste ("select Manufacturer , Model,
                                    Price
                                    from Car_data_oth_prices"))
```

14. WORKSPACE & ENVIRONMENT:

```
getwd()                          # print the current working directory - cwd
setwd("C:\\Users\\ChandraMouli/Desktop/R-Hadoop Foundation Training/1. R Foun
dation/Data Sets")        # / instead of \ in windows
ls()                             # List elements of the environment

list=ls()

# view and set options for the session
help(options)                   # learn about available options
options()                       # view current option settings
options(digits=3)               # number of digits to print on output

# work with your previous commands
history()                       # display last 25 commands
history(max.show=Inf)           # display all previous commands

# save your command history
MyFileName<-paste0("MyHistoryFile","_",format(Sys.time(),format="%b_%d_%y_%H_
%M"))
savehistory(file=MyFileName) # default is ".Rhistory"

# recall(loading) your command history
loadhistory(file=MyFileName) # default is ".Rhistory"

# save the workspace to the file .RData in the cwd
save.image()

x<-10
y<-20
# save specific objects to a file
# if you don't specify the path, the cwd is assumed
save(x,y,file="myfile.RData")

# load a workspace into the current session
# if you don't specify the path, the cwd is assumed
load("myfile.RData")

rm(x)                           # remove an object from workspace
rm(a,b)                         # remove multiple object from workspace
rm(list=ls(all = TRUE))         # clear workspace

q() # quit R. You will be prompted to save the workspace.
```