INTRODUCTION TO R

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

WHAT IS R?

- The R statistical programming language is a free open source package based on the S language (developed by Bell Labs).
- R was created by Ross Ihaka and Robert Gentleman at the university of Auckland, New Zealand
- The language is very powerful for writing programs.
- Many statistical functions are already built in.
- Contributed packages expand the functionality to cutting edge research.

BASIC FEATURES OF R

- R is for data analysis and data visualization tool.
  - Visualization in the form of charts, plots and graphs
- It is supported with number of graphical, statistical techniques.
- There are several GUI editors of R language, out of which RGui and Rstudio are commonly used.
- Common characteristics of R
  - Effective and powerful data handling
  - Arrays and Matrices related operations
  - Graphical representations of the analysis

BASIC FEATURES OF R – STATISTICAL FEATURES

- R provides various statistical and graphical techniques, such as
  - Linear and non-linear modeling,
  - Classical statistical tests,
  - Time-series analysis,
  - Classification, Clustering etc.

- R has various predefined packages. User can also install packages.

- R can generate static graphs. To generate dynamic and interactive graphics, user has to install additional packages

BASIC FEATURES OF R - PROGRAMMING FEATURES

- R supports following
  - Basic Math operations
  - Vector Operations
  - Matrix Operations
  - Some other data structures like data frames and lists.

- It can be used with other programming languages such as Python, Perl, Ruby, Julia and on Hadoop & Spark

BASIC FEATURES OF R - PACKAGES

- A Package is a collection of functions and datasets.
- To access the contents of package you have to first install (if it is not in-built) and load it.
- R provides 2 types of packages
  - Standard Packages (in-built) part of R source code
  - Contributed Packages (user-defined)
- CRAN (Comprehensive R Archive Network) – Collection on R packages.
- These packages widely used in Finance, Genetics, HPC, Machine Learning, Medical Imaging, Social Sciences and Spatial Statistics

BASIC FEATURES OF R – GRAPHICAL USER INTERFACE

- Some popular text editors and Integrated Development Environments (IDEs) that support R programming are
  - ConTEXT
  - Eclipse
  - Emacs (Emacs Speaks Statistics)
  - Vim editor
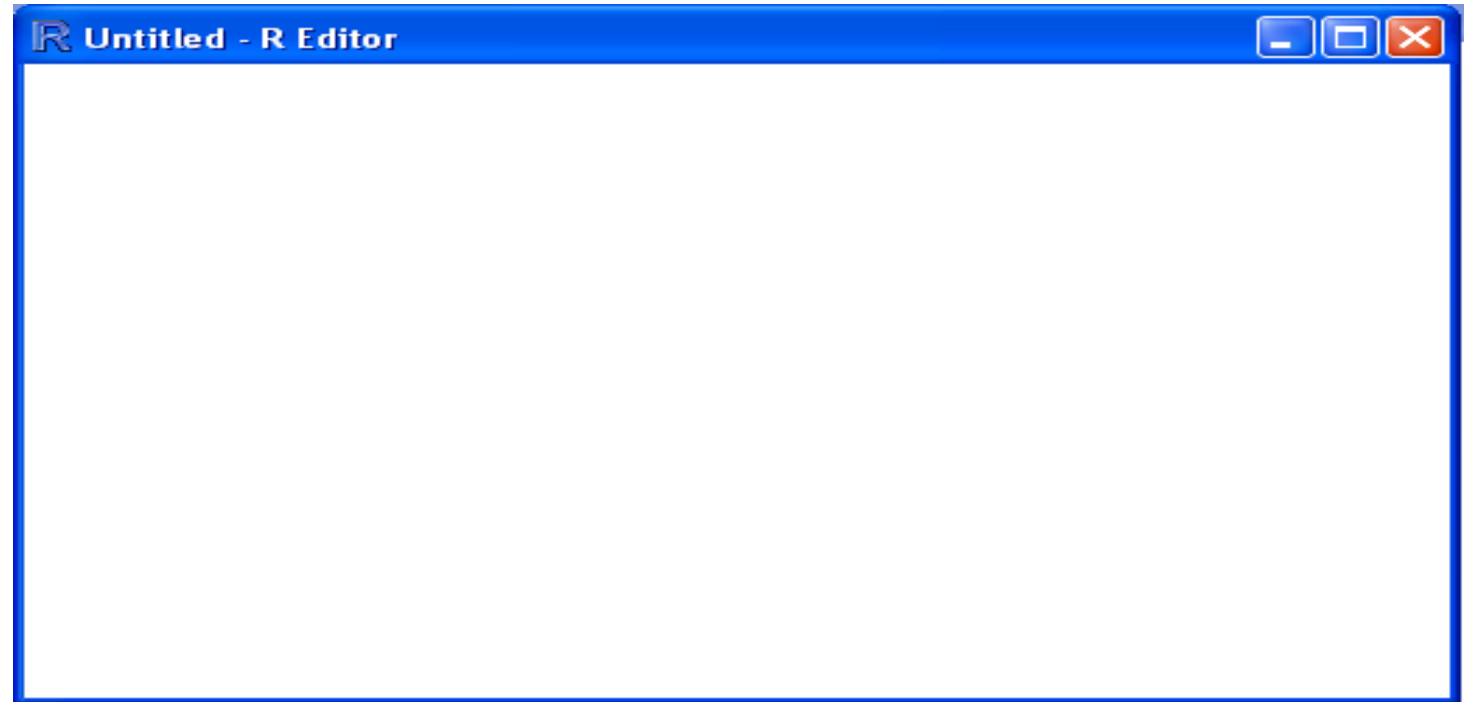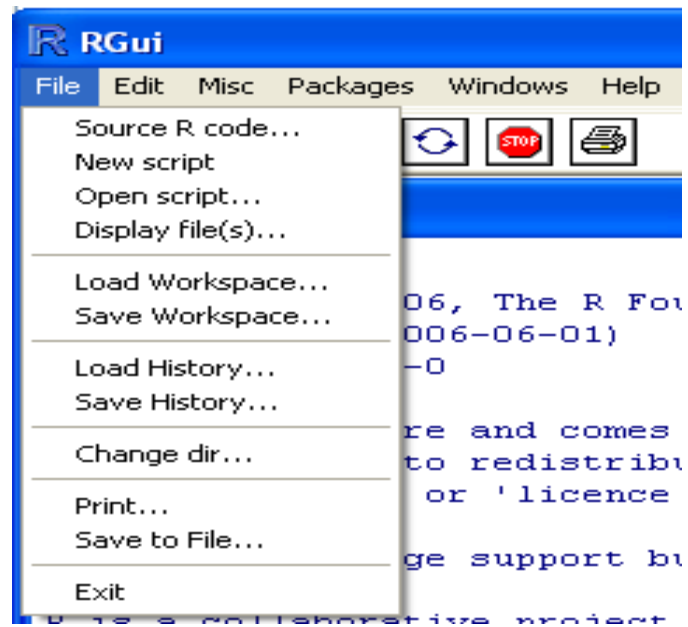 - jEdit
  - Rstudio
  - WinEdit

GETTING STARTED

- Where to get R?
- Go to www.r-project.org
- Downloads: CRAN (The Comprehensive R Archive Network)
- Set your Mirror: Any of the mirror site can be selected.

GETTING STARTED

- Opening a script.
- This gives you a script window.

GETTING STARTED

- Basic assignment and operations.
  - Arithmetic Operations:
    - +, -, *, /, ^ are the standard arithmetic operators.
  - Matrix Arithmetic.
    - * is element wise multiplication
    - %*% is matrix multiplication
  - Assignment
    - To assign a value to a variable use "<-" or "="

GETTING STARTED

- How to use help in R?
  - R has a very good built-in help system.
  - If you know which function you want help with simply use ?_____ with the function in the blank.
    - Ex: `?hist.`
  - If you don't know which function to use, then use help.search("_____").
    - Ex: `help.search("histogram")`

PACKAGES

- Packages are collections of
  - **R** functions,
  - Data and
  - compiled code in a well-defined format.
- The directory where packages are stored is called the library.
- To access and use the package, it has to be loaded first.

PACKAGES

- **R** comes with a standard set of packages. Others are available for download and installation. Once installed, they have to be loaded into the session to be used.

- To install or add new R packages
  - `install.packages("package_name")`

- To load the package
  - `library(package_name)`

- To see default packages on R
  - `library()`

- To see installed packages on R
  - `installed.packages()`

- You can create your own package

CRAN

- It is A Comprehensive R Archive Network, contains many packages which can be used in many domains like
  - Genetics, Bioinformatics
  - Finance
  - HPC (High Performance Computing)
  - Machine Learning
  - Medical Imaging
  - Big data

R CONSOLE

- After installing R on the Linux machine. Just type R on the command line
- After R console is opened, it shows some basic information about R, such as R version, date of release, licensing

```
172.20.1.104 - PuTTY
pavank@bio:~/rstudio-0.99.489/bin$R

R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```
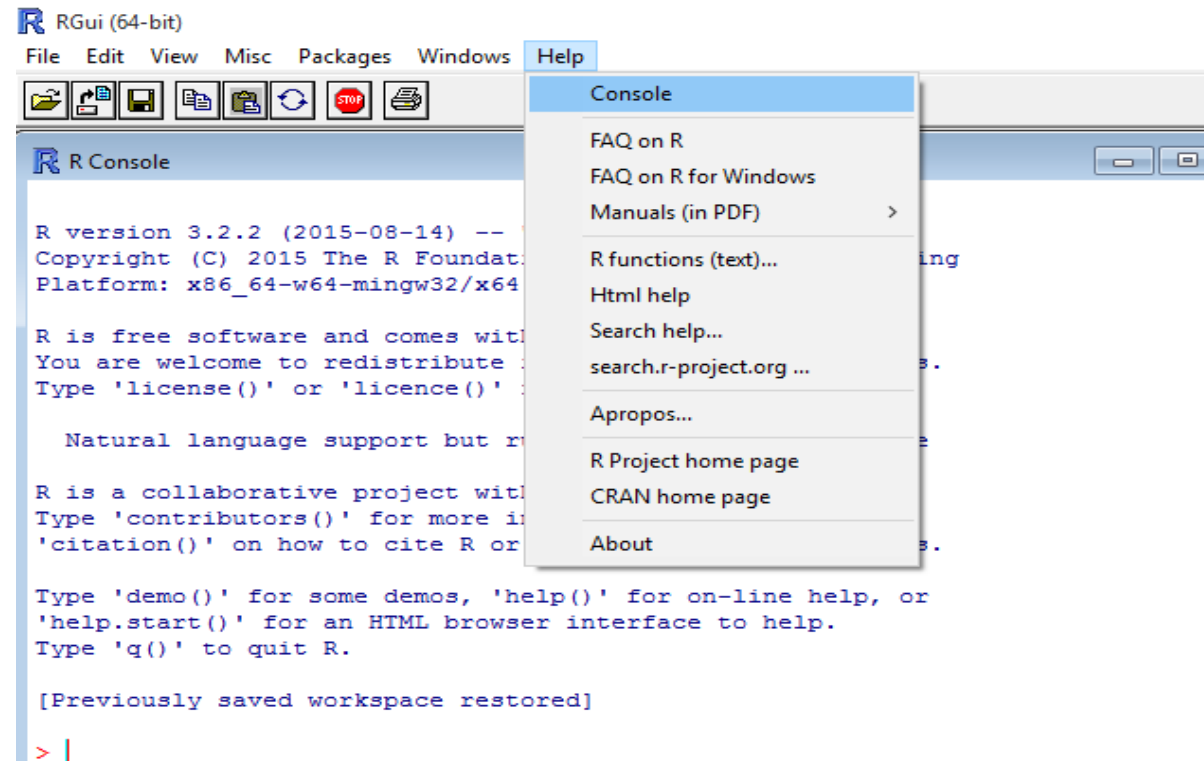
R CONSOLE

- In the previous figure, notice ">" symbol.
- This is called R prompt, which allows users to write commands and then press ENTER key to execute the command.
- To get more information about the console, go to Help->Console.

DEVELOPING A SIMPLE PROGRAM

- Sample program for printing
  - Here, we are using the `print()` function to display "Hello World" on the R console
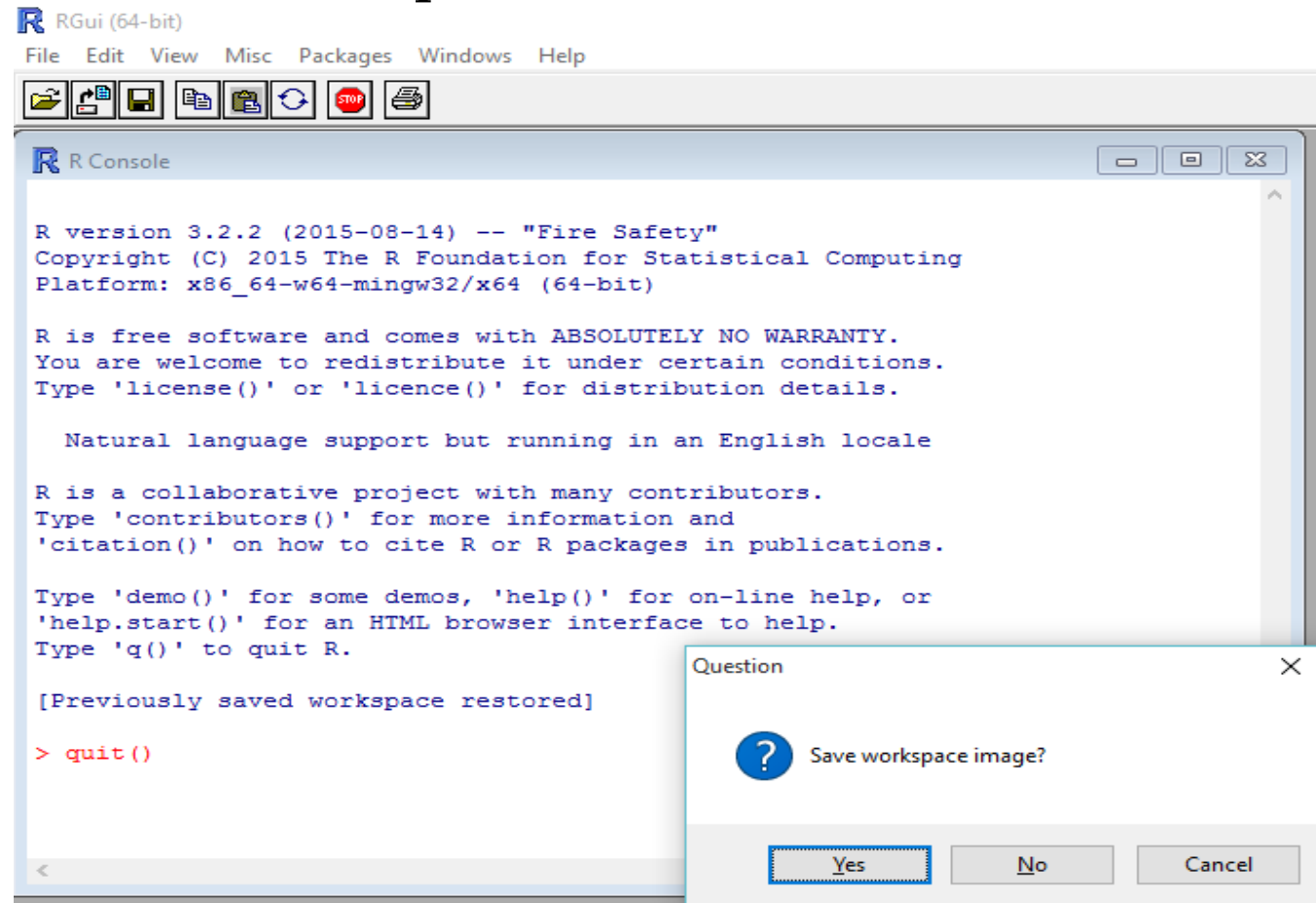
  ```
  >print("Hello World")
  [1] "Hello World"
  ```

  - Here, we are doing simple math

  ```
  >2+3
  [1] 5
  ```

- Code begins with '>' symbol and output begins with [1]

QUITTING R

- You can quit an active session of R by entering q() command
- After executing the q() command, the question dialogue box appears asking whether to save the work space.

HANDLING BASIC EXPRESSIONS

- Anything that you type on R console, it executes immediately on pressing the ENTER key.
- **Basic Arithmetic in R**

```
>12+45+9-7
[1] 59
```

R executes the expression in the following order

12+45+9=66

66-7=59

Lets look at complex mathematical operation

```
>18+23/2-5/4*3.5
[1] 25.125
```

HANDLING BASIC EXPRESSIONS

- To calculate such complex mathematical expressions, R uses BODMAS (Brackets of Division Multiplication Addition Subtraction)

```
>18+22/2-4/4*3.5
[1] 25.5

>(18+22/2-4/4)*3.5
[1] 98.875
```

HANDLING BASIC EXPRESSIONS

- **Mathematical Operators in R**
  - +, -, *, () - Simple Mathematical operations
  - pi   - Stands for Pie value
  - X^Y   - X raised to Y
  - sqrt(x)   - square root of x
  - abs(x)   - Absolute value of x
  - factorial(x)   - Factorial of x
  - log(x)   - logarithm of x
  - exp(x)   - Exponent of x
  - cos(x), sin(x), tan(x)   - Trigonometric functions

DECLARING VARIABLES IN R

- Variables are symbols that are used to contain and store the values.
- Two ways to assign the values
  - Using "=" symbol

  `>MyVar=10`

  - Using "<-" symbol

  `>MyVar<-10`

  Here, `MyVar` is a object and it is assigned with the value 10.

  Any of the above mentioned can used to assign the values.

VARIABLE TYPES IN R

- **Numbers**
  - Real numbers
  - R organizes numbers in 3 formats
    - **Scalar** : Represents a single number (0 dimensional)
    - **Vector** : Represents row of numbers (1 dimensional)
    - **Matrix**: Represents the table like format (2 dimensional)
  - **Working with Vectors**
    - It consists of ordered collection of numbers or strings
    - Numerical Vector
    - String vector

VARIABLE TYPES IN R - VECTORS

- **Constructing the vector in R**
- The `c()` is used to construct the vector (Integer/Character)

- `c(10,20,20,30,40)`
  - It is a Numerical/Integer vector

- `c("Hello2", 20, "Hello4", 30)`
  - It is combination of Numerical and Character vector

- `c("Hello1", "Hello2", "Hello3")`
  - It is a Character vector

VARIABLE TYPES IN R - VECTORS

- Creating the vector using (:) operator

```
>1:15 (generates numbers form 1 to 15)
> c(1:15)
[1]   1   2   3   4   5   6   7   8   9 10 11 12 13 14 15


> sum(1:15) ## it sums the numbers from 1 to 15
[1] 120
```

VARIABLE TYPES IN R

○ **Strings (characters)**

　○A string should be specified by using quotes. Both single and double quotes will work

```
a <- "hello"      ## Assigning a string to variable a
a         ## Printing variable a
"hello"      ## Output of variable a
b <- c("hello","there")     ## Assigning two strings to variable b
b             ## Printing variable b
"hello" "there"   ## Output of variable b
b[1]         ## Printing first element of variable b
"hello"      ## Output of variable b[1]
```

VARIABLE TYPES IN R

- **Factors**
  - Another important way R can store data is in the form of factors
  - Example of Factorial data Yes/No, Male/Female, A/B/C/D

VARIABLE TYPES IN R – DATA FRAMES

- **Data Frames**
  - It is the collection of many vectors of different types, stores in single variable

```
> a<-c(1,2,3,4)
> b<-c(2,4,6,8)
> levels <- factor(c("A","B","A","B"))
> bubba<-data.frame(a, b, levels)
> bubba
  a b levels
1 1 2      A
2 2 4      B
3 3 6      A
4 4 8      B
```

CALLING FUNCTIONS IN R

- Many predefined functions are there in R.
- To invoke, user has to type their names
- For example

```
> sum(10,20,30)
1] 60
> rep("Hello",3)
[1] "Hello" "Hello" "Hello"
> sqrt(100)
[1] 10
> substr("example",2,4)
[1] "xam"
```

```
> sum(10,20,30)
[1] 60
> ## Replicate Function
> rep("Hello", 3)
[1] "Hello" "Hello" "Hello"
> rep("1", 3)
[1] "1" "1" "1"
> ## Squaring the Number
> sqrt(100)
[1] 10
> ## Substring
> substr("example", 2,4)
[1] "xam"
>
```

CREATING AND USING OBJECTS

- R uses objects to store the results of a computation

```
> myobj<-25+12/2-16+(7*pi/2)
> myobj
[1] 25.99557
```

Assigns a mathematical expression to an object called **myobj**

Invokes the **myobj** object

- R is case sensitive – that is, it treats data15 and Data15 as completely different objects.

CREATING AND USING OBJECTS

- An object can be assigned a set of numbers, as for example:

```
> x12 <- c(10,6,8)
> x12
[1] 10  6  8
```

```
> x12<-c(10,12,14)
> x12*2
[1]  20 24 28
> |
```

- Operations can then be performed on the whole set of numbers.
  - For example, for the object **x12** created above, check the results of the following:

```
> x12 * 10
[1] 100  60  80
```

READING DATASETS

- Using the `c()` command:
  - `c()` function is used to combine or concatenate two or more values. Here example shown is concatenating 2 numerical vectors.
  - Syntax for the `c()` command

```
> a<-c(1:100)
> b<-c(1:100)
> d<-c(a,b)
> d
  [1]    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18
 [19]   19   20   21   22   23   24   25   26   27   28   29   30   31   32   33   34   35   36
 [37]   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51   52   53   54
 [55]   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69   70   71   72
 [73]   73   74   75   76   77   78   79   80   81   82   83   84   85   86   87   88   89   90
 [91]   91   92   93   94   95   96   97   98   99  100    1    2    3    4    5    6    7    8
[109]    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26
[127]   27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44
[145]   45   46   47   48   49   50   51   52   53   54   55   56   57   58   59   60   61   62
[163]   63   64   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80
[181]   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98
[199]   99  100
```

HANDLING DATA IN R WORKSPACE

○ Here are some functions that allows us to handle data in R workspace

- `ls()` function: Used to view all the created variables in current active space

```
> ls()
 [1] "a"   "b"   "bubba"   "fun"   "levels"   "msg"
 [7] "myobj"   "n"   "x12"   "yourname"
```

HANDLING DATA IN R WORKSPACE

- The `rm()` function :rm() function is used to remove the variables that are not required anymore in a session

```
> ls()
[1] "a"      "b"      "bubba"     "fun"      "levels"     "msg"
[7] "myobj"      "n"              "x12"          "yourname"


> rm(a)


> ls()
[1] "b"      "bubba"     "fun"  "levels"  "msg"    "myobj"      "n"
[8] "x12"         "yourname"
```

HANDLING DATA IN R WORKSPACE

- **getwd() function:** Function used to display the current working directory of the user

```
> getwd()
[1] "/home/bioinfo/pavank/rstudio-0.99.489/bin"
```

- **save() function:** Function used to save the objects created in the active session.

```
> save(x12, file="x12.rda")
```

  - It will save in the current working directory with the name "x12.rda"
  - You can also save entire working image `save.image()`

HANDLING DATA IN R WORKSPACE

○ load() function : Function used to retrieve the saved data

```
yourname<-"mary"
> ls()
[1] "b" "fun" "levels" "msg" "myobj" "n" "x12"   "yourname"
> save(yourname, file="yourname.rda")
> rm(yourname)
> ls()
[1] "b" "fun" "levels" "msg" "myobj" "n" "x12"
> load("yourname.rda")
> ls()
[1] "b" "fun" "levels" "msg" "myobj" "n" "x12" "yourname"
```

EXECUTING R SCRIPTS

- Creating and Executing R script on Windows:
  - Open Notepad, and write R commands
  - Save it has "filename.R"
  - From the Rgui, file->Open script. It opens a window for browsing the Rscript
  - Click Open

EXECUTING R SCRIPTS

- Creating and Executing R script on Linux:
- R script is the series of commands written and saved in .R extension
- To run a script "`/home/bioinfo/pavank/R/use1.R`"
- You may either use:
  - From R Shell

    `source("/home/bioinfo/pavank/R/use1.R")` ##
  - On the Linux Shell

    `R CMD BATCH /home/bioinfo/pavank/R/use1.R` (OR)

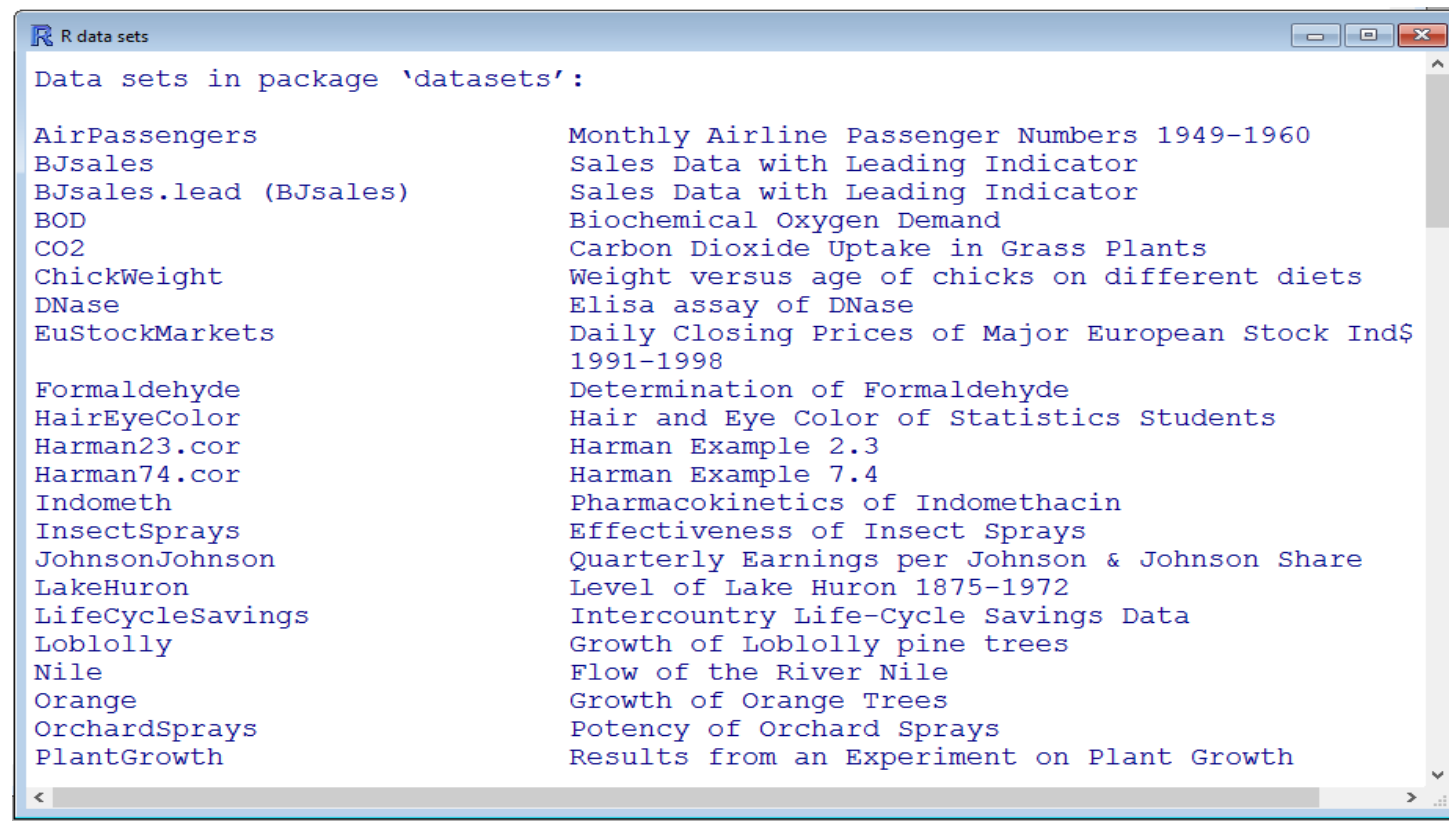   `Rscript use1.R`

ACCESSING HELP AND DOCUMENTATION IN R

- Function used to get help pages of the in-built functions are `help()` and `example()`

  > `help(ls)` or `?ls()`

  > `example(ls)` – It shows the examples of ls function

- Sample datasets : R has many in-built datasets

  > `data()`

USING BUILT-IN DATASETS IN R

○ There many built-in data sets which can be viewed by `data()` command. The output is shown

>`data()`   ##Generates the list of built-in datasets

```
R data sets

Data sets in package 'datasets':

AirPassengers            Monthly Airline Passenger Numbers 1949-1960
BJsales                  Sales Data with Leading Indicator
BJsales.lead (BJsales)   Sales Data with Leading Indicator
BOD                      Biochemical Oxygen Demand
CO2                      Carbon Dioxide Uptake in Grass Plants
ChickWeight              Weight versus age of chicks on different diets
DNase                    Elisa assay of DNase
EuStockMarkets           Daily Closing Prices of Major European Stock Ind$
                         1991-1998
Formaldehyde             Determination of Formaldehyde
HairEyeColor             Hair and Eye Color of Statistics Students
Harman23.cor             Harman Example 2.3
Harman74.cor             Harman Example 7.4
Indometh                 Pharmacokinetics of Indomethacin
InsectSprays             Effectiveness of Insect Sprays
JohnsonJohnson           Quarterly Earnings per Johnson & Johnson Share
LakeHuron                Level of Lake Huron 1875-1972
LifeCycleSavings         Intercountry Life-Cycle Savings Data
Loblolly                 Growth of Loblolly pine trees
Nile                     Flow of the River Nile
Orange                   Growth of Orange Trees
OrchardSprays            Potency of Orchard Sprays
PlantGrowth              Results from an Experiment on Plant Growth
```

USING BUILT-IN DATASETS IN R

○ There is a command for viewing all the data sets that are user-built or contributed packages.

```
data(package=(all.available=TRUE))
```

```
Data sets in package 'boot':

acme                          Monthly Excess Returns
aids                          Delay in AIDS Reporting in England and Wales
aircondit                     Failures of Air-conditioning Equipment
aircondit7                    Failures of Air-conditioning Equipment
amis                          Car Speeding and Warning Signs
aml                           Remission Times for Acute Myelogenous Leukaemia


Data sets in package 'cluster':

agriculture                   European Union Agricultural Workforces
animals                       Attributes of Animals
chorSub                       Subset of C-horizon of Kola Data
flower                        Flower Characteristics
plantTraits                   Plant Species Traits Data
pluton                        Isotopic Composition Plutonium Batches
```

THANK YOU !!!