BIG DATA TOOLS FOR MANAGERS

Unit-3 : Introduction to R

by
Ankit Velani
Adjunct Faculty, Dept. of MBA,
Siddaganga Institute of Technology,Tumkur

Session-1 Overview

- Introduction to R-tool
- R & R-Studio
- Getting started with R
  - Variables
  - Comments
  - Working directory
  - Packages

Introduction to R

- R is an open-source programming language and software environment for statistical & data analysis.
- In 1993, R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand.
- **R** name derived from the first letter of two founders (**R**oss & **R**obert)

How R is useful?

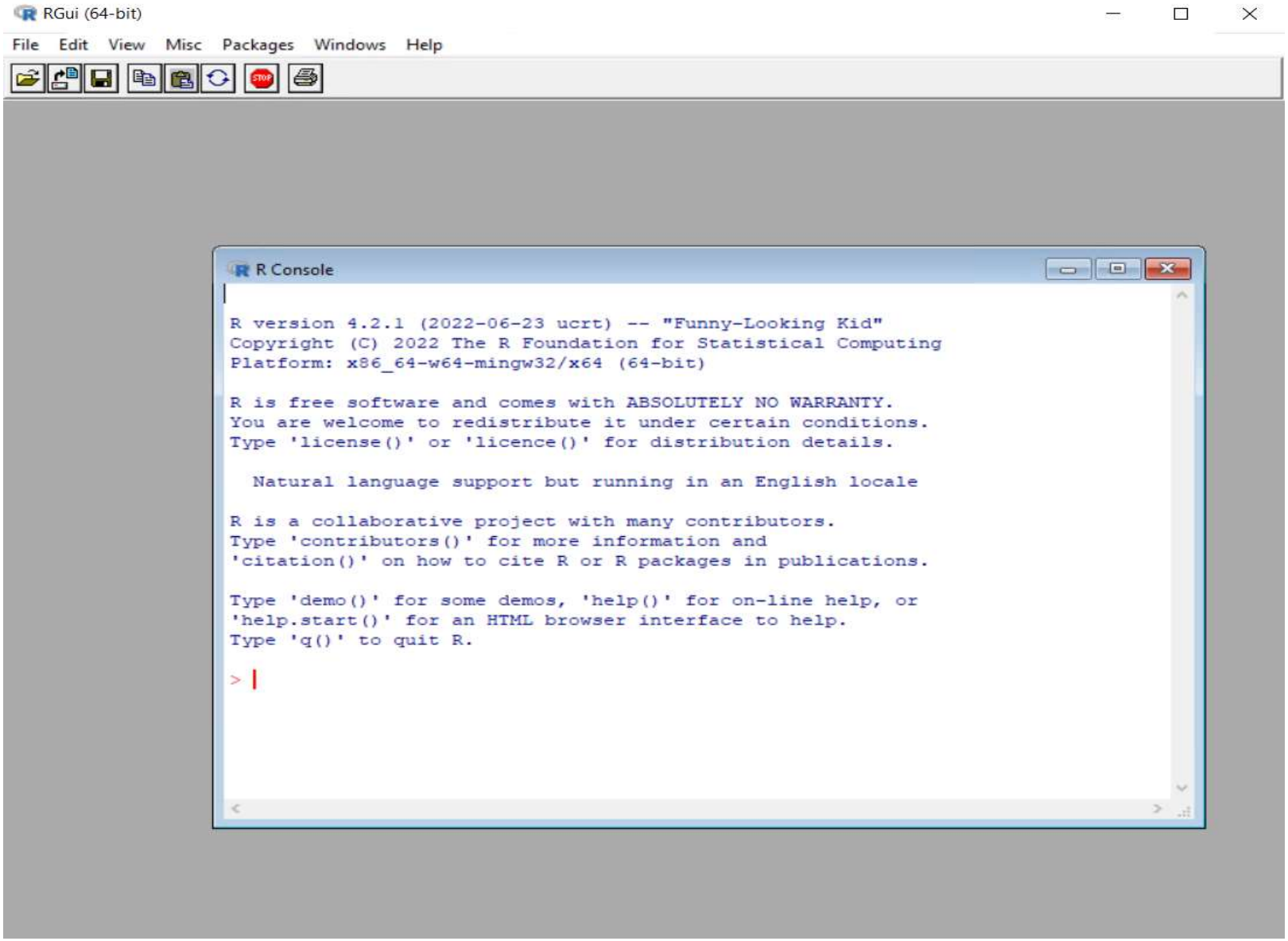
- R has built-in features and functionality for statistical analysis and data visualization.
- R can easily preprocess varieties of structure & unstructured data.
- R is available for all the operating system (Windows, Linux, macOS), and easy to move R projects from one operating system to another operating system.
- R supports Shiny App which is a web-based interactive app to develop and showcase R projects.
- R has a wide & active community, and 10,000+ packages to improve project code and productivity while working with R.

R Tool Software

- R Interpreter/software, which is available for all the OS
- R documents and packages are available on Comprehensive R Archive Network(CRAN)
- Official website :

http://www.r-project.org

R Interpreter



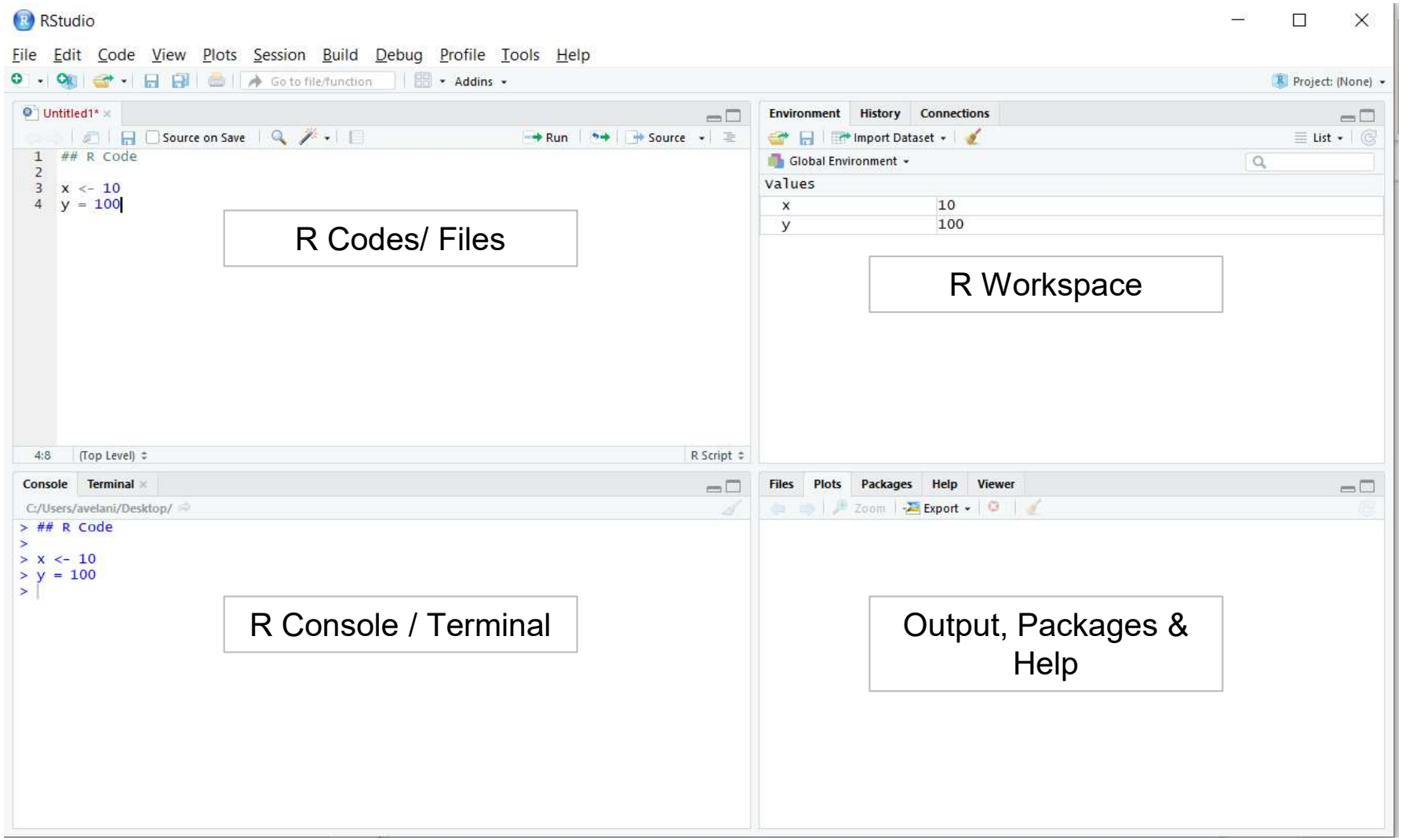
RStudio

- RStudio is development environments for R projects.

- RStudio has a four main window panel
  - Code
  - Workspace
  - R Terminal
  - Output

- Official website :
  https://www.rstudio.com/products/rstudio/download/

RStudio


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
1 ## R Code
2
3 x <- 10
4 y = 100
4:8
(Top Level)
R Script
R Codes/ Files
Environment
History
Connections
Import Dataset
List
Global Environment
Values
x 10
y 100
R Workspace
Console
Terminal
C:/Users/avelani/Desktop/
> ## R Code
>
> x <- 10
> y = 100
>
R Console / Terminal
Files
Plots
Packages
Help
Viewer
Zoom
Export
Output, Packages & Help

Variables

- Variables are container for storing data value in memory.
- In R, Variable gets created as soon as it gets assign with some values to it.
- R supports both ←(left assignment) and = (equal to) operator for assigning the value
- Most prefer assignment operator is ←(left assignment)

Variables

Rules to create variable name:

- Variable name must start with a letter and followed by letters, numbers, and underscore
- Variable name cannot start with number or underscore
- Variable names are case sensitive so ie. name, Name and NAME are three different variables
- Reserved words can not be used as variable name(TRUE, FALSE, NULL, if…)

Variables

Example :

x ← 10
y = 100

a ← b ← c ← “hello”

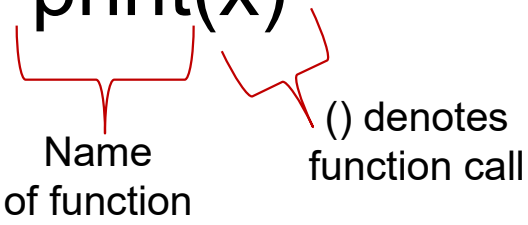
R also allow to assign same value to multiple variable together.

Print

R does have print function to get the values of variable.

```
print(x)
print(y)

x ← 50  # Storing 50 value in memory and address is X
print(x)
```



Comments

- R allows to annotate codes with Comments
- Comments line starts with hash mark(#), and anything that comes thereafter will be ignored by R interpreter/tool.
- Comments are highly recommended to keep notes, complex logic details and other details for documentation.

Ex.

```
# This is comment line
10 + 10 #addition of two number
```

Working directory

- An Active R Session always has a working directory associated with it, unless you explicitly specify a file path when saving or importing data.
- Working directory denotes by default R read/write data from working directory.

getwd() : get a current working directory name

getwd functions gives folder path displayed with forward slashes(/) to path to specific location

setwd("path-to-folder") : change current working directory

Needs to provide full path to specific folder with forward slashes(/)

Working directory

getwd()
o/p:
"C:/Users/home/Documents“

setwd(“D:/class/R/”)

R Packages

- R tools comes with lots of built in commands, function for numeric calculations for common statistical analysis, plotting visualization.

- R packages are collection of R functions code and sample data that stored under a directory which called “library” in the R environment.

- R package easily available on CRAN network & over the internet.

R Packages

- Install & download from Internet/ CRAN network
  install.packages(“package-name”)

ie. install.packages(“car”)
install.packages(“caret”)
install.packages(“MASS”)

R Packages

- Install & download from Internet/ CRAN network
  install.packages("package-name")

ie. install.packages("car")
install.packages("caret")
install.packages("MASS")

- Load/import package in R code
  library("package-name")

Ie.

library("car")
library("MASS")

Recap

- R is an open source programming language for statistical analysis.
- To download R from official website (http://www.r-project.org)
- R documentation & packages are available on CRAN (Comprehensive R Archive Network)
- R Studio is development environment to develop & work with R projects.
- R Studio has 4 window panel (Code, Workspace, Terminal and Output)
- Comments can be used to annotate R code and it start with hash(#)
- getwd() functions to get the current working directory.
- setwd() function to change or modify working directory.

Session-2 : R data types

- Basic data types
- Advanced data structures
  - Vectors
  - Lists
  - Matrices
  - Data Frames

Basic Data types

- There are several basic data types in R and those frequent occurrences in the routine of R calculation.
- While writing R programming, need to store the data in a variable and this data might be of different types like Integer, Decimal, String, Complex…etc
- Data types are helpful to the programmer for understanding the types of data that developer is handling and manipulating.

Basic Data types

Data types	Values
Logical	TRUE or FALSE
Integer	Set of all the integer numbers
Numeric	Set of all the real numbers
Complex	Set of complex numbers (a+bi)
Characters	The sequence of characters enclosed within single or double quotes

Code : R data types

Logical

```
x ← TRUE
print(x)    #print function helps to display the value of x variable
class(x)    #class function helps to get the name/class of data type
```

Numeric

By default, all the numbers are numeric types and if we create numbers with the suffix L then it becomes an integer type of data.

```
x ← 70.15
print(x)
class(x)
```

Code : R data types

Integer

```
x ← 101L
print(x)
class(x)
```

Complex

```
x ←6 + 4i
print(x)
class(x)
```

Code : R data types

Integer

```
x ← 101L
print(x)
class(x)
```

Complex

```
x ←6 + 4i
print(x)
class(x)
```

Characters

```
x ← “Hello World”
y ← ‘Y’

print(x)
print(y)

class(x)
class(y)
```

Vector

- Vector is the essential building block for handling multiple items in R.
- It's a list of items that are of the same type.
- Combine function **c()** used to combine multiple values of same type.
- ie.

fruits ← c("Apple", "Banana", "Orange")

num ← c(1,2,3,4,5,6)

num ← 1:100 #Integer values in a sequence

dec ← 1.5 : 6.5 #Numeric values in a sequence

Vector

- Length function which helps to find out how many items a vector has.

length(fruits)

length(num)

Vector

- Combining two vectors

```
num_all <- c(num, dec)
print(num_all)

alpha = c("A", "B", "C", "D")
fruits = c("Apple", "Banana", "Orange")

data = c(alpha, fruits, num_all)
print(data)
```

Vector

- **Sequence**

common useful functions to create continuous number generation.

seq(from=, to=, by=)
seq(from=, to=, length.out=)

Example:
seq(from=1, to=10, by=2)
seq(from=1, to=10, length.out=20)

Vector

- **Repeat**

common useful functions to repeat the certain values in vector.

rep(x=, each=, times=)

Example:

rep(x=c(1,2,3), each=2)

o/p: 112233

rep(x=c(1,2), each=2, times=4)

O/p :1122 1122 1122 1122

Vector

- **Repeat**

common useful functions to repeat the certain values in vector.

rep(x=, each=, times=)

- **times** provide no of times entire vector elements to repeat
- **each** provide no of times each vector elements to gets repeat.

Example:

rep(x=c(1,2,3), each=2)

o/p: 112233

rep(x=c(1,2), each=2, times=4)

O/p :1122 1122 1122 1122

Vector

- **Sort**

Sort function used to sort vector elements in increasing or decreasing order.

sort(x=, decreasing=)

x = is vector

decreasing is TRUE/FALSE

Example: Sort element in increasing order

sort(x= c(2.5, -1, -10, 3.44), decreasing=FALSE)

Vector

- **Accessing Vector Elements**

Index to be used to access the vector elements, index starts from 1 to length of vector

Syntax:

vector_name[index]

Example :

num <- c(“A”, “B”, “C”, “D”, “E”)

num[1] #1st Element

num[5] #5th Element

Data Frames

- A data frame is R's most natural way of presenting two-dimensional dataset with collection of observation with one or more variables.
- Data frame is one of the most important and frequently used in R for data analysis.

- **data.frame()** function helps to create data frame in R

Data Frames

- Creation of data frame

```
my_data <- data.frame(
        name   = c("A","B","C","D"),
        age    = c(40,45,70,60),
        gender = c("F","M","F","M")
)
```

name, age, gender are the vectors

Output:

Name	Age	gender
A	40	F
B	45	M
C	70	F
D	60	M

Data Frames

- Accessing elements
  data-frame_object[row_range , col_range]

Example:

my_data[1,] **#first row with all the columns**

my_data[1,3] **# first row with only 3rd columns**

my_data[1, 2:3] **# first row with 2 & 3rd columns**

my_data[2:3, 1] **# 2 & 3rd row with 1st columns**

Data Frames

- Accessing elements with Variable Name

  data-frame_object**$variable_name**

  my_data$name
  my_date$age

- Accessing elements with condition on row index

  data-frame_object**[condition,]**

  my_data[my_data$gender==“M”,]
  my_data[my_data$gender==“F”, 2]