



INTRODUCTION

OBJECTIVES OF TUTORIAL

i.	Understand basic statistics and create visualization to solve data analysis problems in various areas.
ii.	Apply statistical methodologies to model practical problems for solutions.
iii.	Apply the fundamental concepts of probability to solve problems related engineering field.
iv.	Apply statistical techniques on large data using R programming.



WHAT IS R ?

- R is a language and environment for statistical computing and graphics.
- It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.
- A well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

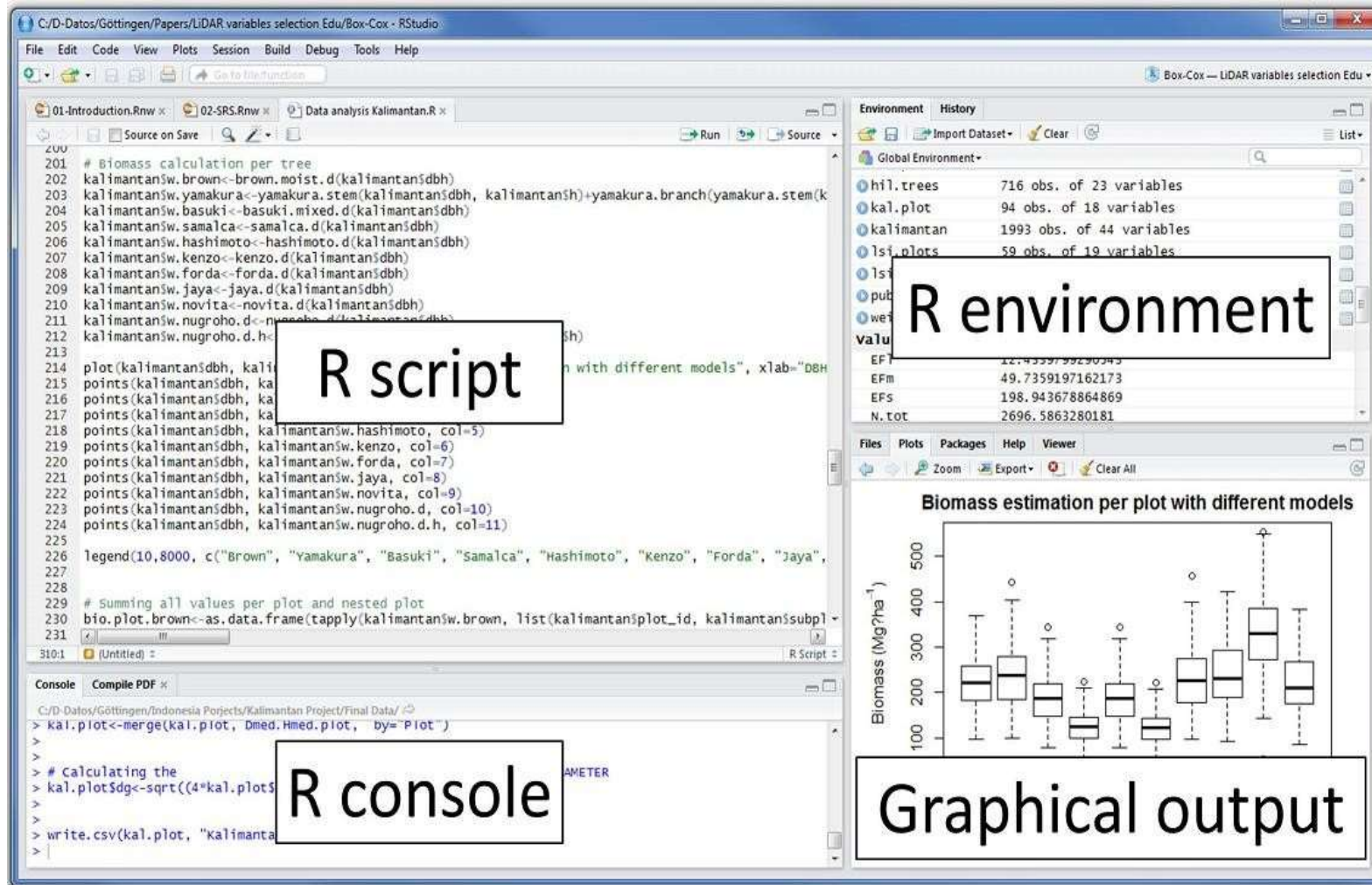


R FEATURES

- R provides a wide variety of statistical and graphical techniques.
- R is an integrated suite of software facilities for data manipulation, calculation and graphical display.
- It includes an effective data handling and storage facility.



RSTUDIO SCREEN



R COMMAND PROMPT

- Type your first program

#Sample R program

- `myString <- "Hello World!"`
- `print (myString)`

OR

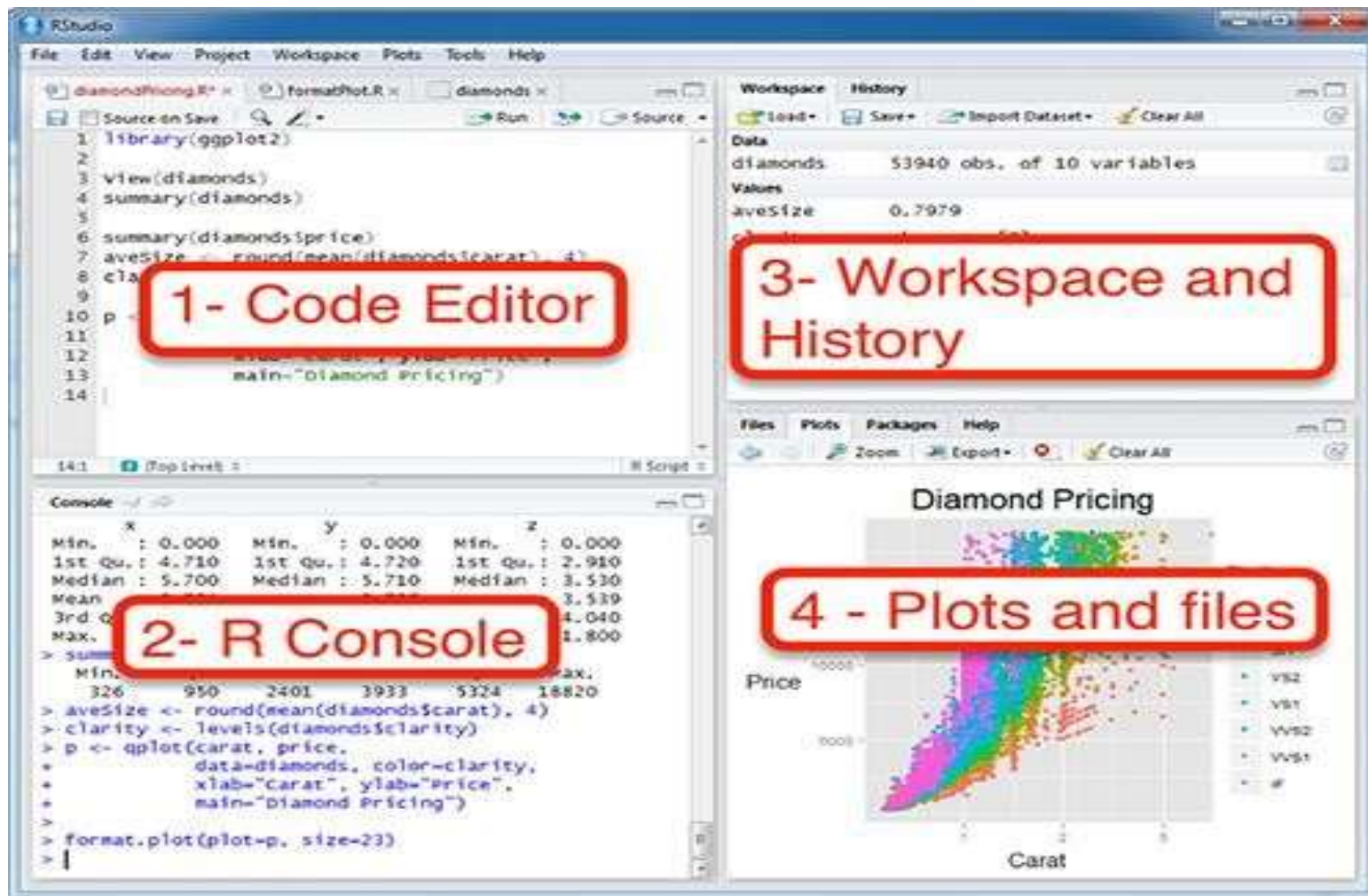
- `myString`

OR

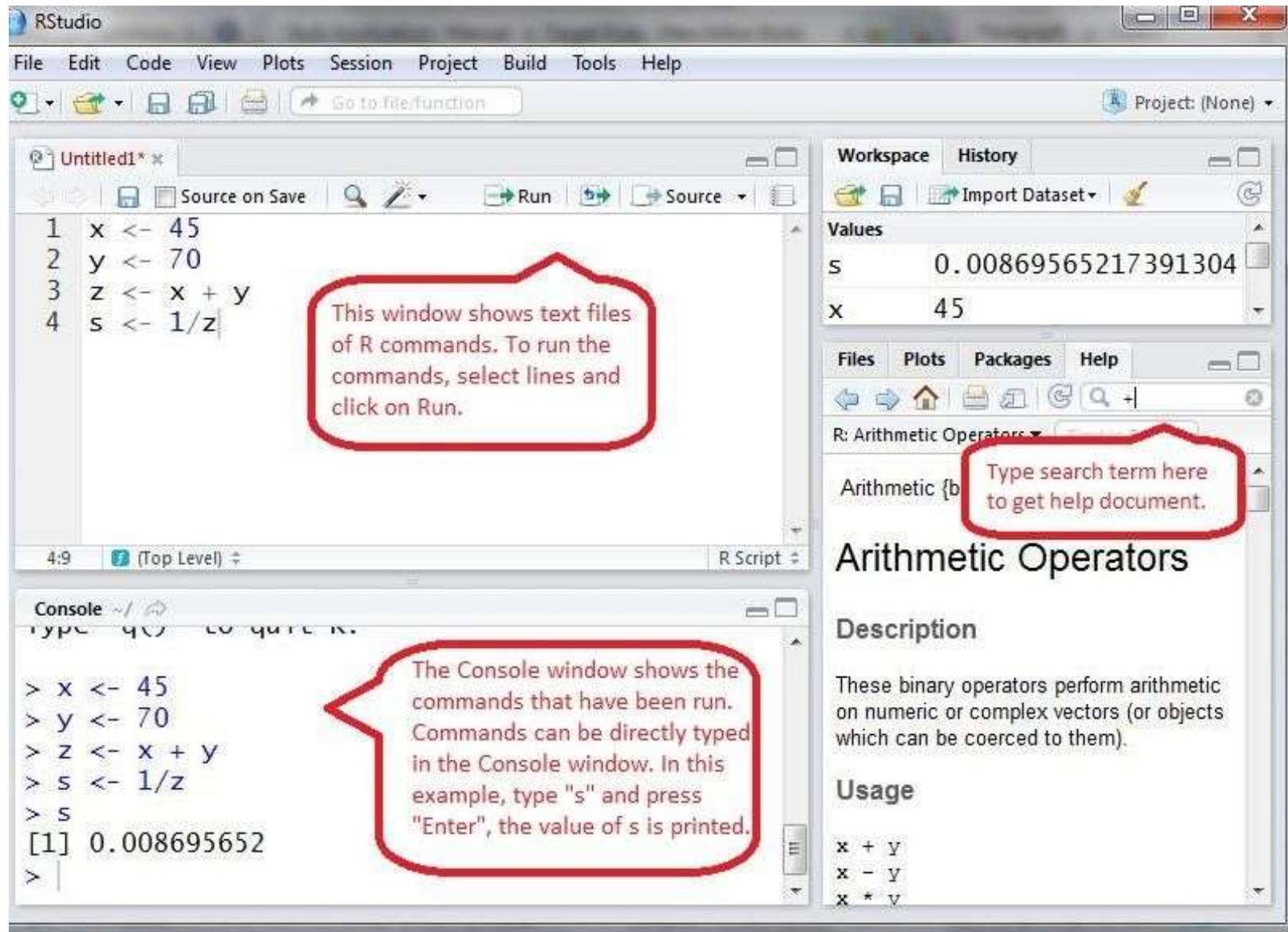
- `cat (myString)`



RSTUDIO SCREEN



RSTUDIO SCREEN



EXAMPLE

- `x = 50`
- `y = 20`
- `sum = x + y`
- `diff = x - y`
- `product = x * y`
- `div = x / y`
- `print (sum)`
- `print(diff)`
- `print(product)`
- `print(div)`



DATATYPE OF VARIABLES

- `var_x <- "Hello"`
- `cat("The class of var_x is ",class(var_x),"\n")`

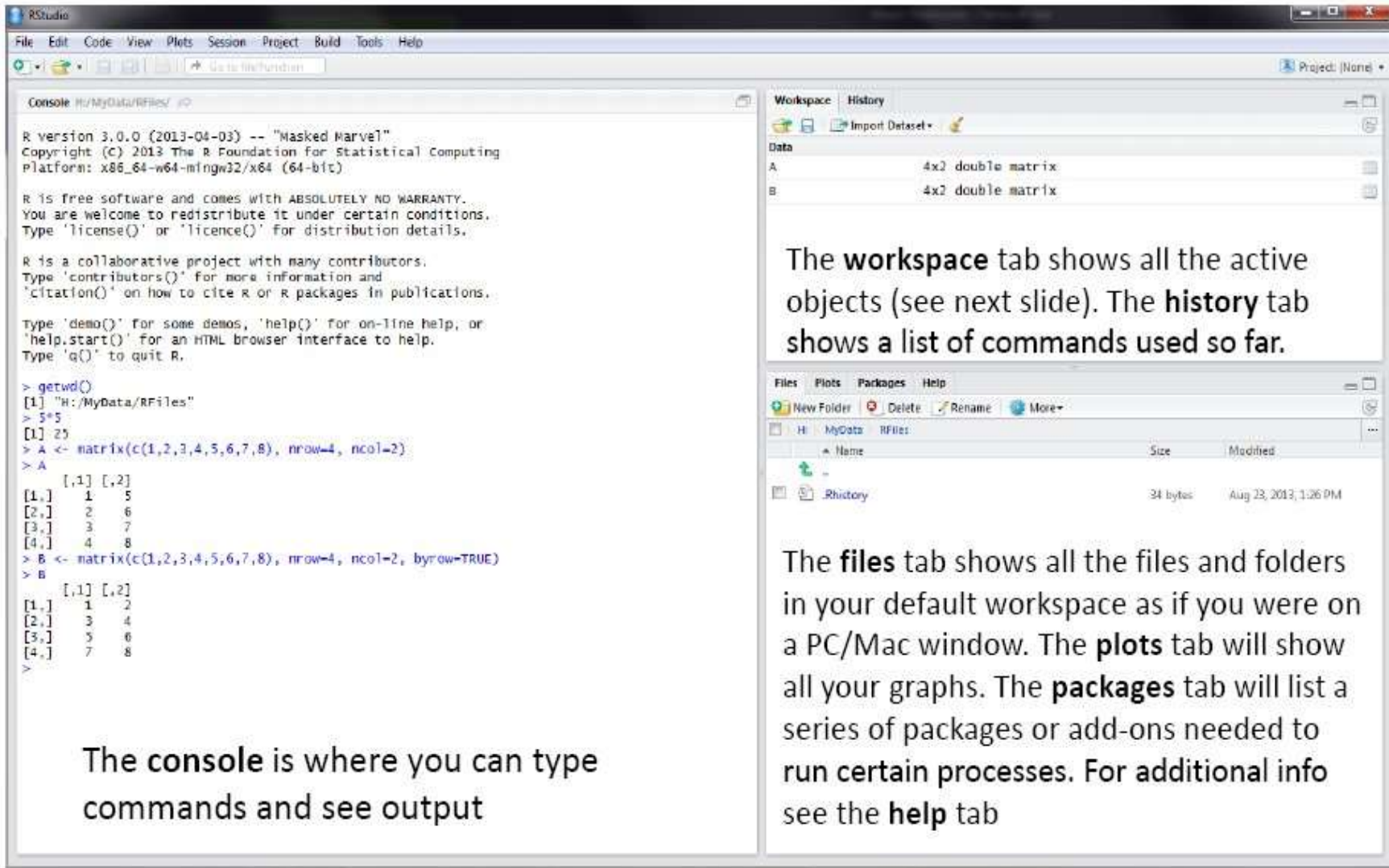
- `var_y <- 34.5`
- `cat(" Now the class of var_y is ",class(var_y),"\n")`

#Finding Variables

- `print(ls())`



RSTUDIO SCREEN



The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Project, Build, Tools, and Help. The toolbar below the menu contains icons for file operations and running code. The main interface is divided into four panes: Console, Workspace, History, and Files. The Console pane on the left shows the R version (3.0.0), copyright information, and the output of several R commands. The Workspace pane on the top right lists active objects A and B, both 4x2 double matrices. The History pane on the top right shows a list of commands used. The Files pane on the bottom right shows the file explorer for the current project, displaying a folder named 'MyData' and a file named '.Rhistory'.

Console `R/MyData/RFiles/`

```
R version 3.0.0 (2013-04-03) -- "Masked Marvel"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

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.

> getwd()
[1] "H:/MyData/RFiles"
> 5*5
[1] 25
> A <- matrix(c(1,2,3,4,5,6,7,8), nrow=4, ncol=2)
> A
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
> B <- matrix(c(1,2,3,4,5,6,7,8), nrow=4, ncol=2, byrow=TRUE)
> B
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
>
```

The **workspace** tab shows all the active objects (see next slide). The **history** tab shows a list of commands used so far.

The **files** tab shows all the files and folders in your default workspace as if you were on a PC/Mac window. The **plots** tab will show all your graphs. The **packages** tab will list a series of packages or add-ons needed to run certain processes. For additional info see the **help** tab

The **console** is where you can type commands and see output

DATA TYPES

- The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable.
- There are many types of R-objects.
- **Vectors**
- **Lists**
- **Matrices**
- **Arrays**
- **Factors**
- **Data Frames**



VECTORS

- When you want to create vector with more than one element, you should use `c()` function which means to combine the elements into a vector.

```
# Create a vector.
```

```
color <- c('red','green',"yellow")
```

```
print(color)
```

```
# Get the class of the vector.
```

```
print(class(color))
```



VARIABLES USAGE

- # Assignment using equal operator.
- `var.1 = c(0,1,2,3)`
- # Assignment using leftward operator.
- `var.2 <- c("learn","R")`
- # Assignment using rightward operator.
- `c(TRUE,1) -> var.3 print(var.1)`
- `cat ("var.1 is ", var.1 ,"\n")`
- `cat ("var.2 is ", var.2 ,"\n")`
- `cat ("var.3 is ", var.3 ,"\n")`



EXAMPLES ON VECTORS

- `a=c(1,2,3,4)`
- `b= a+5`
- `print(b)`

- `sqrt(a)`

- `a <- c(1,2,3)`
- `b <- c(10,11,12,13)`
- `a+b`

- `a = c(2,4,6,3,1,5)`
- `b = sort(a)`
- `c = sort(a,decreasing =TRUE)`



LISTS

- A list is a R-object which can contain many different types of elements inside it like vectors, functions.

Create a list.

```
list1 <- list(c(2,5,3),21.3,"sin")
```

Print the list.

```
print(list1)
```



MATRICES

- A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

Create a matrix.

```
M=matrix(c('a','a','b','c','b','a'),  
nrow=2,ncol=3,byrow = TRUE)  
print(M)
```

```
N=matrix(c('23','12','45','67','90','55'),  
nrow=3,ncol=2)  
print(M)
```



ARRAYS

- While matrices are confined to two dimensions, arrays can be of any number of dimensions.
- The array function takes a dim attribute which creates the required number of dimension.
- Example below we create an array with two elements which are 3x3 matrices each.

Create an array.

```
a <- array(c('green','yellow'),dim=c(3,3,2))  
print(a)
```



FACTORS

- Factors are the r-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels.
- The labels are always character irrespective of whether it is numeric or character or boolean etc. in the input vector. They are useful in statistical modeling.
- Factors are created using the **factor()** function.
- The **nlevels** functions gives the count of levels.



EXAMPLE ON FACTORS

Create a vector.

```
apple_colors <-  
  c('green','green','yellow','red','red','red','green')
```

Create a factor object.

```
factor_apple <- factor(apple_colors)
```

Print the factor.

```
print(factor_apple)  
print(nlevels(factor_apple))
```



DATA FRAMES

- Data frames are tabular data objects. Unlike a matrix in data frame ,each column can contain different modes of data.
- The first column can be numeric while the second column can be character and third column can be logical. It is a list of vectors of equal length.
- Data Frames are created using the **data.frame()** function.

Create the data frame.

```
BMI <- data.frame( gender = c("Male", "Male","Female"), height =  
  c(152, 171.5, 165), weight = c(81,93, 78), Age =c(42,38,26) )  
print(BMI)
```

```
dat <- data.frame(id = letters[1:10], x = 1:10, y = 11:20)  
dat
```



OPERATORS

- An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. R language is rich in built-in operators and provides following types of operators.
- **Types of Operators**
- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Miscellaneous Operators



ARITHMETIC OPERATORS

Operator	Description	Example
+	Adds two vectors	<pre>v <- c(2,5.5,6) t <- c(8, 3, 4) print(v+t) It produces following result: [1] 10.0 8.5 10.0</pre>
%%	Give the remainder of the first vector with the second	<pre>v <- c(2,5.5,6) t <- c(8, 3, 4) print(v%%t) it produces following result: [1] 2.0 2.5 2.0</pre>
%/%	The result of division of first vector with second (quotient)	<pre>v <- c(2,5.5,6) t <- c(8, 3, 4) print(v%/%t) it produces following result: [1] 0 1 1</pre>
^	The first vector raised to the exponent of second vector	<pre>v <- c(2,5.5,6) t <- c(8, 3, 4) print(v^t) it produces following result: [1] 256.000 166.375 1296.000</pre>



RELATIONAL OPERATORS

Operator	Description	Example
>	Checks if each element of the first vector is greater than the corresponding element of the second vector.	<pre>v <- c(2,5.5,6,9) t <- c(8,2.5,14,9) print(v>t)</pre> <p>It produces following result: [1] FALSE TRUE FALSE FALSE</p>
==	Checks if each element of the first vector is equal to the corresponding element of the second vector.	<pre>v <- c(2,5.5,6,9) t <- c(8,2.5,14,9) print(v==t)</pre> <p>it produces following result: [1] FALSE FALSE FALSE TRUE</p>
!=	Checks if each element of the first vector is unequal to the corresponding element of the second vector.	<pre>v <- c(2,5.5,6,9) t <- c(8,2.5,14,9) print(v!=t)</pre> <p>it produces following result: [1] TRUE TRUE TRUE FALSE</p>



LOGICAL OPERATORS

Operator	Description	Example
&	It is called Element-wise Logical AND operator. It combines each element of the first vector with the corresponding element of the second vector and gives an output TRUE if both the elements are TRUE.	<pre>v <- c(3,1,TRUE,2+3i) t <- c(4,1,FALSE,2+3i) print(v&t) it produces following result: [1] TRUE TRUE FALSE TRUE</pre>
!	It is called Logical NOT operator. Takes each element of the vector and gives the opposite logical value.	<pre>v <- c(3,0,TRUE,2+2i) print(!v) it produces following result: [1] FALSE TRUE FALSE FALSE</pre>
&&	Called Logical AND operator. Takes first element of both the vectors and gives the TRUE only if both are TRUE.	<pre>v <- c(3,0,TRUE,2+2i) t <- c(1,3,TRUE,2+3i) print(v&& t) it produces following result: [1] TRUE</pre>



ASSIGNMENT OPERATORS

Operator	Description	Example
<- or = or <<-	Called Left Assignment	<pre>v1 <- c(3,1,TRUE,2+3i) v2 <<- c(3,1,TRUE,2+3i) v3 = c(3,1,TRUE,2+3i) print(v1) print(v2) print(v3) it produces following result: [1] 3+0i 1+0i 1+0i 2+3i [1] 3+0i 1+0i 1+0i 2+3i [1] 3+0i 1+0i 1+0i 2+3i</pre>
-> or ->>	Called Right Assignment	<pre>c(3,1,TRUE,2+3i) -> v1 c(3,1,TRUE,2+3i) ->> v2 print(v1) print(v2) it produces following result: [1] 3+0i 1+0i 1+0i 2+3i [1] 3+0i 1+0i 1+0i 2+3i</pre>



MISCELLANEOUS OPERATORS

Operator	Description	Example
<code>%in%</code>	This operator is used to identify if an element belongs to a vector.	<pre>v1 <- 8 v2 <- 12 t <- 1:10 print(v1 %in% t) print(v2 %in% t)</pre> it produces following result: [1] TRUE [1] FALSE
<code>%*%</code>	This operator is used to multiply a matrix with its transpose.	<pre>M = matrix(c(2,6,5,1,10,4), nrow=2,ncol=3,byrow = TRUE) t = M %*%t(M) print(t)</pre> it produces following result: [1,] [2,] [1,] 65 82 [2,] 82 117



- <https://www.coursera.org/learn/decision-making/lecture/K57sl/the-role-of-r>
- <https://www.rstudio.com/products/rstudio/>

