# R Language

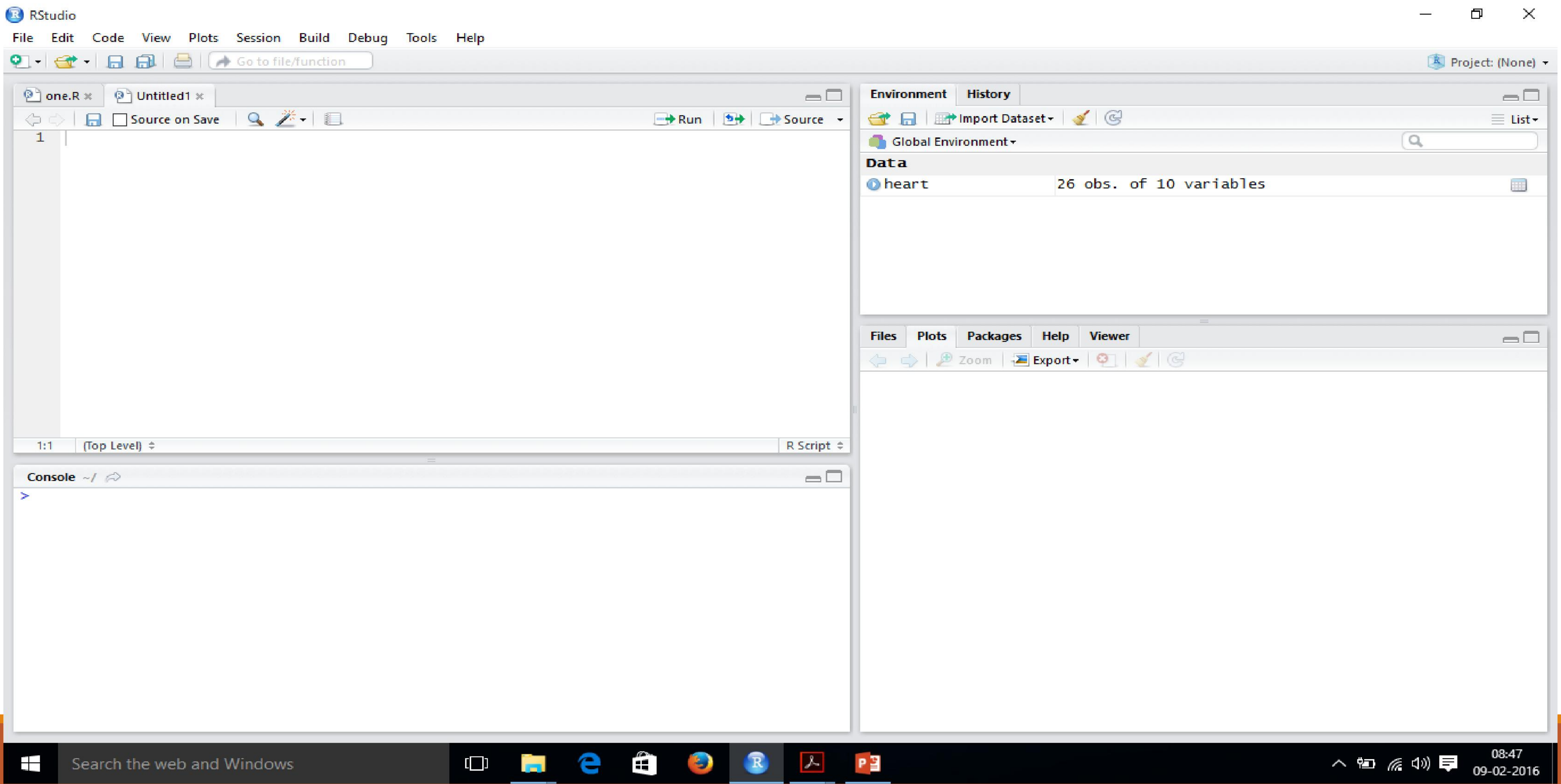
VIT UNIVERSITY

CHENNAI CAMPUS

## Introduction

- **The R system for statistical computing is an environment for data analysis and graphics.**
- **The root of R is the S language.**
- **It is developed by John Chambers and colleagues at Bell Laboratories(1960).**
- **Installation of R**
- **R has a command line interface**
- **R is a programming Language and Rstudio (IDE) is an interface for R**

# Rstudio Layout



```
1
```

Data

| | |
|---|---|
| heart | 26 obs. of 10 variables |

```
>
```

## Rstudio Layout

- **Console window:**
  - **It is situated at Bottom Left of the RStudio Layout**
  - **It is also called command window**
  - **We can write commands**
  - **All commands can be executed in this window only**

## Rstudio Layout

- **Editor window:**
  - **It is situated at Top Left of corner of the RStudio Layout**
  - **We are using this window for writing scripts → collection of commands**
  - **It is also called script window**
  - **If this window is not visible, we can get it by File → New → Rscript**
  - **Click RUN or CRTL+ENTER to send the highlighted commands to command window**

## Rstudio Layout

- **History window:**
  - **It is situated at Top Right of corner of the RStudio Layout**
  - **In this window, we can see the data and values of R in memory.**
  - **It is also called workspace window**
  - **We can view and edit the values by clicking on them**
  - **This history window shows what has been typed so far**

## Rstudio Layout

- **Help window:**
  - **It is situated at the right bottom of RStudio Layout**
  - **Here we can open files and view plots**
  - **We can install and load the packages**

# Short-cut Keys in Rstudio

- **Shortcut Keys :**
  - **Easy running of the code: CTRL+ENTER (runs highlighted lines of code)**
  - **Even easier: CTRL+ENTER+P re-run the last-run code**
  - **<tab> works for auto completion**
  - **CTRL+1 → source editior**
  - **CTRL+2 → Console**
  - **CTRL+L → clear the console**
  - **CTRL+O → Open the file**
  - **CTRL+S → save the fie**
  - **CTRL+F → find**
  - **CTRL+shift+N → opens new document**
  - **ESC → interrupt a lengthy R command**
  - **CTRL+shift+C → comment or uncomment (highlighted code)**

## Setting the Working Directory in RStudio

- **Setting the working directory:**
  - **To store working file.**
  - **create a folder and named as RdataWork.**
  - **To create working directory → setwd("path").**
    - **setwd("E:/studies/VIT/R/RdataWork") in windows environment**
    - **setwd("~/RdataWork/") in Linux environment**

## R as a Simple Calculator

- **R as a simple Calculator:**
  - **Typing in a mathematical expression and hitting enter prints out the result.**
  - **> 1 + 2**

    **[1] 3**
  - **Order of operation rules worked as expected**
  - **Mathematical functions such as the square root →sqrt**
  - **> sqrt(36)**

    **[1] 6**

## R as a Simple Calculator

- **R as a simple Calculator:**
  - **The result of mathematical expression can be assigned to an object in R. → <- Operator**
    - **var1 <- sqrt(81)**
    - **var1**
  - **Every object in R belongs to a class → type of the object it represents**
    - **class(var1)**

      **[1] "numeric"**
  - **Everything in R is an object, including functions.**
  - **ls() → list objects → prints the all the objects.**

# R as a Simple Calculator

## R as a simple Calculator:

- 5 %% 4 → Check the output
- log(2) → Check the output
- cos(pi) → Check the output
- ceiling(3.2) → Check the output ;;; round(123.456,digits=2) → check output
- 0/0 → check the out put ;;; round(-123.456,digits=-2) → round numbers to multiples of 10, 100, etc
- 1/Inf → check the output ;;; signif(-123.456,digits=4) → number of significant digits to be retained
- few maths functions are:
  - abs, sqrt, log, exp, log10, factorial
- few Trig functions are:
  - sin, cos, tan, asin, acos, atan
- Rounding functions are:
  - round, ceiling, floor, trunc, signif, zapsmall
- math quantities are:
  - Inf, -Inf, NaN, pi, exp(1), 1i

**$ floor(123.45) → check output**
**$ floor(-123.45) → Check output**
**floor(x) rounds to the nearest integer that's smaller than x**

# Operation Symbols in R

## ❖Operation Symbols in R:

| Symbol | Meaning |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| **%%** | **Modulo (estimates remainder in a division)** |
| ^ | Exponential |

# Numbers in R

- **Numbers in R:**
  - **NAN (not a number)**
  - **NA (missing value)**
  - **Basic handling of missing values**
  - **> var2 =c(1,2,3,4,5,6,NA)**
  - **> var2**
    - **[1]  1  2  3  4  5  6 NA**
  - **> mean(var2)**
    - **[1] NA**
  - **> mean(var2,na.rm=TRUE)**
    - **[1] 3.5**

## Variable assignment in R

- **Variable assignment in R:**
  - **> var10 = 25**
    - **[1] 25**
  - **Var11 <- 29**
    - **[1] 29**

# Variable assignment in R

- **Variable assignment in R:**
  - **> var10 = 25**
  - **>var10**
    - [1] 25
  - **>var11 <- 29** **;; typeof(var11) → check result**
  - **>var11**
    - [1] 29
  - **we can create a list using c-command**
    - var22 <- c(1,2,3,4)
    - mean(var22) → check the output
    - var(var22) → check the output

## Basic Data Types in R

- **Basic Data types in R:**
  - **Numeric**
  - **Integer**
  - **Complex**
  - **Logical**
  - **Character**
  - **Vector**
  - **Matrix**
  - **List**
  - **Data Frame**

## Basic Data Types in R

- **Numeric in R:**
  - **> var12 = 25.12**
    - **>var12 → check output**
  - **is.integer() → used to check wheater a given variable object is integer or not**
    - **is.integer(var12) → check output**

**typeof(var12) → check output**

# Basic Data Types in R

- **Integer in R:**
  - **To create an integer variable in R → as.integer().**
    - **>var23 <- as.integer(999)**
      - **> var23 → check output**
      - **is.integer(var23) → check output**
      - **typeof(var23) → check output**
      - **class(var23) → check output**
      - **var34 <- as.integer(10.28)**
      - **var34 → check output**
      - **is.integer(var34) → check output**
      - **typeof(var34) → check output**

## Basic Data Types in R

- **Logical values in R: (and (&), or (|) , Negation(!)**
  - **True → 1 and False → 0**
    - **> as.integer(TRUE)**
      - **> 1**
    - **> as.integer(FALSE)**
      - **0**
  - **>x = 1**
  - **>y = 2**
  - **>z = x > y**
  - **>z → check the output**
  - **class(z) → check the output**

# Basic Data Types in R

- **complex values in R:**
  - **A complex value in R is defined via the pure imaginary value i.**
    - **> z = 1 + 2i**
      - **> z**
        - **[1] 1+2i**
    - **class(z) → check output**
    - **sqrt(-1) → check output**
    - **sqrt(as.complex(-1)) → check the output**

## Basic Data Types in R

- **Vector in R:**
  - **A vector is a sequence of data elements of the same basic type.**
  - **components → members in a vector**
  - **Examples**
    - **>c(10,20,40) → 3 members/ components**
      - **> [1] 10 20 40**
    - **>c(TRUE, FALSE, TRUE) → check output**
    - **>c(“VIT”, “Chennai”, “Campus”) → check output**
    - **>c(“VIT”, 600126) → check output**
    - **> class(c(600123,"vit")) → check output**
    - **>length(c(600123,"vit")) → check Output**

# Basic Data Types in R

## Combining Vector in R:

- >var25 = c(1,4,7)
- >var26 = c("vit", "university")
- >var27 = c(var25, var26)
- >var27
- class(var27) → check output
- class(var26) → check output
- class(var25) → check output

## Basic Data Types in R

### Combining Vector in R:

- A vector is an ordered collection of objects of the same type
- The function c(...) concatenates its arguments to form a vector
- To create a patterned vector
  - : → Sequence of integers
  - seq() → General Sequence
  - rep() → Vector of replicated elements

```
$ rr ←   - 3 : 3
$ rr → check output
$ seq(0,2,by=0.5) → check output
$  seq(0,2,len=6)
$ rep(1:5, each=2)
$ rep(1:5, times=2)
```

# Basic Data Types in R

## Vector arithmetic in R:

- > var28 = c(2, 4, 6, 8)
- > var29 = c(2, 4, 6, 8)
- > var28 * var29
- > var28 + var29
- What's happens if one vector size less compared to other vector → Recycling rule
- > var30 = c(2,4)
- > var28 + var30 → check the result
- What happens if one vector is numeric and other is character → able to perform arithmetic operation these two vectors → No. why

```
y <- c(4,2,0,9,5,3,10)
sort(y)
sort(y, decreasing = TRUE)
```

# Basic Data Types in R

## Vector index in R:

- > var33 = c (“vit”, “chennai”, “vellore”, “campus”)
- > var33[1] → Check the output
- We can retrieve the values in a vector, using indices, that has to used in array bracket. Index is starting from 1.
- The result of the slice is also a vector.
- Negative Index:
  - If the index is negative, it would strip the member whose position has the same absolute value as the negative index.
  - >var33[-2] → it skip the values at 2nd position in vector
- Out of Range index:
  - If an index is out-of-range, a missing value will be reported via the symbol NA.
  - > var33[10] → check the output

# Basic Data Types in R

## Numeric index Vector in R:

- A new vector can be sliced from a given vector with a numeric index vector, which consists of member positions of the original vector to be retrieved.
  - > var33[c(3,4)] → check output
- Duplicate Indexes:
  - > var33[c(3,4,4)]
- Out of Order Indexes:
  - >var33[c(4,3,1)]
- Range – Index
  - > var33[1:3]

```
$ var28 <- c(4,7,2,10,1,0)

$ var28[ var28 > 3] → check
output
```

# Basic Data Types in R

## which() and match() in R:

- Additional functions that will return the indices of a vector
- which() → Indices of a logical vector where the condition is TRUE
- which.max() → Location of the (first) maximum element of a numeric vector
- which.min() → Location of the (first) minimum element of a numeric vector
- match() → First position of an element in a vector
  - var28 <- c(4,7,2,10,1,0)
  - var28 >= 4 → check the output
  - which(var28 >= 4) → check the output
  - which.max(var28) → check the output
  - var28[which.max(var28)] → check output
  - max(var28)

```
$ var40 <- rep(1:5, times = 5:1)
$ var40 → check output
$ match(1:5, var40)
$ match(unique(var40), var40)
```

# Basic Data Types in R

## Named Vector in R:

- **We can assign names to vector members**
  - >var33
  - > names(var33) = c(“one”, “two”, “three”, “four”)
  - >var33 → check the output.
  - >var33[“two”] → check the output

## Useful Vector Functions

| | | |
|---|---|---|
| `sum(x)` | `prod(x)` | Sum/product of the elements of x |
| `cumsum(x)` | `cumprod(x)` | Cumulative sum/product of the elements of x |
| `min(x)` | `max(x)` | Minimum/Maximum element of x |
| `mean(x)` | `median(x)` | Mean/median of x |
| `var(x)` | `sd(x)` | Variance/standard deviation of x |
| `cov(x,y)` | `cor(x,y)` | Covariance/correlation of x and y |
| `range(x)` | | Range of x |
| `quantile(x)` | | Quantiles of x for the given probabilities |
| `fivenum(x)` | | Five number summary of x |
| `length(x)` | | Number of elements in x |
| `unique(x)` | | Unique elements of x |
| `rev(x)` | | Reverse the elements of x |
| `sort(x)` | | Sort the elements of x |
| `which()` | | Indices of TRUEs in a logical vector |
| `which.max(x)` | `which.min(x)` | Index of the max/min element of x |
| `match()` | | First position of an element in a vector |
| `union(x, y)` | | Union of x and y |
| `intersect(x, y)` | | Intersection of x and y |
| `setdiff(x, y)` | | Elements of x that are not in y |
| `setequal(x, y)` | | Do x and y contain the same elements? |

# Basic Data Types in R

## Matrix in R:

- A matrix is a collection of data elements arranged in a two-dimensional rectangular layout.
- A matrix is just a two-dimensional generalization of a vector
- To create a Matrix:
  - matrix(data = NA, nrow=1, ncol=1, byrow=FALSE, dimnames=NULL)
    - data → a vector that gives data to fill the matrix;

      → if data does not have enough elements to fill the matrix, then the elements are recycled.
    - nrow → desired number of rows;; ncol → desired number of columns
    - byrow → if false( default) matrix is filled by columns, otherwise rowwize
    - dimnames → (optional) list of length 2 giving the row and column names respectively, list
    - names will be used as names for the dimensions

## Basic Data Types in R

- **Matrix in R:**
  - **Example:**
    - **> A = matrix(c(2,4,1,3,5,7),nrow=2,ncol=3,byrow=TRUE)**
    - **> A → check the output**
    - **> A[2,3] → check the output**
    - **> A[2, ] → check the output**
    - **> A[, 3] → check the output**
    - **> A[, c(1,3)] → check the output**
    - **> dimnames(A) = list( c(“r1” ,”r2”), c(“c1”,”c2”,”c3”) )**
    - **> A → check output**

**$ A <- matrix(1:4, nrow=2)**
**$ B <- matrix(1, nrow=2, ncol=2)**
**$ A → check output**
**$ B → check output**
**$ A * B → check output**
**$ A %*% B → check output**

**$ y <- 1:3**
**$ y → check output**
**$ y %*% y → check output**

# Basic Data Types in R

## ❖Matrix in R:

- ❖Example:
  - ❖> K <- t(A) → check output → transpose of A
  - ❖ > B = matrix( c(10,20,30), nrow=3, ncol = 1)
  - ❖ > cbind(K,B) → check output → cbind() used for combining matrices by columns
  - ❖ > rbind() → use to combine matrices based on row-wise
  - ❖ c(B) → destruction the matrix →form a vector

## Useful Matrix Functions

| | |
|---|---|
| `t(A)` | Transpose of `A` |
| `det(A)` | Determinate of `A` |
| `solve(A, b)` | Solves the equation `Ax=b` for `x` |
| `solve(A)` | Matrix inverse of `A` |
| `MASS::ginv(A)` | Generalized inverse of `A` (MASS package) |
| `eigen(A)` | Eigenvalues and eigenvectors of `A` |
| `chol(A)` | Choleski factorization of `A` |
| `diag(n)` | Create a `n`×`n` identity matrix |
| `diag(A)` | Returns the diagonal elements of a matrix `A` |
| `diag(x)` | Create a diagonal matrix from a vector `x` |
| `lower.tri(A),upper.tri(A)` | Matrix of logicals indicating lower/upper triangular matrix |
| `apply()` | Apply a function to the margins of a matrix |
| `rbind(...)` | Combines arguments by rows |
| `cbind(...)` | Combines arguments by columns and |
| `dim(A)` | Dimensions of `A` |
| `nrow(A), ncol(A)` | Number of rows/columns of `A` |
| `colnames(A), rownames(A)` | Get or set the column/row names of `A` |
| `dimnames(A)` | Get or set the dimension names of `A` |

# Basic Data Types in R

- **apply() in R:**
  - **The apply() function is used for applying functions to the margins of a matrix, array, or dataframes.**
  - **apply(matrix,  margin,  fun(), …)**
    - **matrix → A matrix or array or a dataframe**
    - **margin → Vector of subscripts indicating which margins to apply the function to 1 = rows , 2 = columns, c(1,2) = rows and columns**
    - **fun() → function to be applied**
- **x <- matrix(1:12, nrow=3, ncol=4) → check output**
- **apply(x, 1, sum) → row totals → check output**
- **apply(x,2, mean) → column means → check output**

## Basic Data Types in R

- **Lists in R:**
  - **A list is a generic vector containing other objects.**
    - **Example:**
      - **> n = c(2, 3, 5)**
      - **> s = c("aa", "bb", "cc", "dd", "ee")**
      - **> b = c(TRUE, FALSE, TRUE, FALSE, FALSE)**
      - **> x = list(n, s, b, 3)**
      - **x → check the output**
      - **x[2] → check the output**
      - **x[c(2,4)] → check the output**

# Basic Data Types in R

- **Lists in R:**
  - **Member Reference:**
    - **Example:**
      - **> x[ [2] ] → check the output**
      - **> x[ [2]] = “ra”**
      - **> x[ [2]] → check the output**
      - **> s → check the output**

# Basic Data Types in R

- **Lists in R:**
  - **Named List Member:**
    - **We can assign names to list members, and reference them by names instead of numeric indexes.**
      - **> v = list(bob=c(2, 3, 5), john=c("aa", "bb"))**
      - **> v → check the output**
      - **> v[“bob”]**
      - **> v[c(“bob”, “john”)]**
      - **> v[["bob"]] → check the output**
      - **> v$bob → check the output**

## Basic Data Types in R

- **attach() and detach() in R:**
  - **We can attach a list to the R search path and access its members without explicitly mentioning the list.**
  - **It should to be detached for clean up.**
    - **example:**
      - **> attach(v)**
      - **> bob → check output**
      - **detach(v)**

## Basic Data Types in R

- **Data frame in R:**
  - **A data frame is used for storing data tables.**
  - **It is a list of vectors of equal length.**
    - **Example:**

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc")
> b = c(TRUE, FALSE, TRUE)
> df = data.frame(n, s, b)      # df is a data frame
```

## Basic Data Types in R

- **Data frame in R:**
  - **Build – in Data Frames:**
    - **Example: → > mtcars**
      - **> mtcars → check the output**
        - **Header, data row or instances, cell or variable**
      - **mtcars[1,2] → give the cell value of first row and 2nd column.**
      - **> mtcars["Mazda RX4", "cyl"] → check output**
      - **> nrow(mtcars) → check output**
      - **> ncol(mtcars) → check the output**
      - **> help(“mtcars”)**
      - **> head(mtcars) → check output**
      - **> tail(mtcars) → check output**
      - **>dim(mtcars) → check output**

# Basic Data Types in R

- **Data frame column vector in R:**
  - **uses [[]] → data frame column**
    - **Example:**
      - **>mtcars[[6]]**
      - **> mtcars[[“am”]]**
      - **>mtcars$am**
      - **> mtcars[, “am”]**

## Basic Data Types in R

- **Data frame column slice in R:**
  - **>mtcars[1]**
  - **> mtcars[“mpg”]**
  - **> mtcars[c(“mpg”, “hp”]**

## Basic Data Types in R

- **Data frame Row slice in R:**
  - **Example:**
    - **>mtcars[24,]**
    - **> mtcars[c(3,24)] → to retrieve more than one row data**

## Data Import in R

- **Data import in R:**
  - **Code – 1:**
    - **> library(XLConnect)**
    - **> wk = loadWorkbook("mydata.xls")**
    - **> df = readWorksheet(wk, sheet="Sheet1")**

# Data Import in R

- **Data import in R:**
  - **Import of xls sheets**
    - **Code – 3 :**
      - **> library(xlsx)**
      - **> mydata <- read.xlsx("c:/myexcel.xlsx", 1)**
      - **> mydata <- read.xlsx("c:/myexcel.xlsx", sheetName = "mysheet")**

## Data Import in R

- **Data import in R:**
  - **Import of table file:**
    - **Code -1:**
      - **> mydata = read.table(“~~path/mydata.txt")**
      - **> mydata**
    - **Code – 2:**
      - **>  mydata = read.table(file.choose(),header=TRUE/FALSE)**
      - **> mydata**
      - **>help(read.table) → see the output**

## Data Import in R

- **Data import in R:**
  - **Import of CSV file:**
    - **Code -1:**
      - **> mydata = read.table(“~~path/mydata.csv")**
      - **> mydata**
    - **Code – 2:**
      - **> mydata = read.table(file.choose(),header=TRUE/FALSE, sep=“,” , row.names=“id”)**
      - **> mydata**