Why Learn

01 Free and Open-source Tool

02 Large Community of Users

03 Latest cutting edge technology

04 Independent Platform

05 Gateway to lucrative career

06 Has a Robust visualization library

07 Go to language for Stats. & Data Science

08 Used in almost every industry

Introduction & Basics

- What is R?
- R is a programming language developed by Ross Ihaka and Robert Gentleman in 1993.
- R possesses an extensive catalogue of statistical and graphical methods. It includes machine learning algorithm, linear regression, time series, statistical inference to name a few.
- R is entrusted many large companies also use R programming language, including Uber, Google, Airbnb, Facebook and so on.
- Data analysis with R is done in a series of steps; Programming, Transforming, Discovering, Modelling and Communicate the results.
- Program: R is a clear and accessible programming tool
- Transform: R is made up of a collection of libraries designed specifically for data science.
- Discover: Investigate the data, refine your hypothesis and analyse them.
- Model: R provides a wide array of tools to capture the right model for your data.
- Communicate: Integrate codes, graphs, and outputs to a report with R Markdown or build Shiny apps to share with the world

Introduction & Basics

- Derived from Bell Labs Language S.

- Developed at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

- R is freely available under the GNU General Public License, and precompiled binary versions are provided for various operating systems like Linux, Windows and Mac.

Features of R

- R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
- R has an effective data handling and storage facility.
- R provides a large, coherent and integrated collection of tools for data analysis.
- R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.
- R programming features include database input, exporting data, viewing data, variable labels, missing data, etc.
- R is an open source and we can use it anywhere.
- In R, anyone is welcome to provide bug fixes, code enhancements, and new packages.

How companies are using R

- **Ford** uses R to improve the design of its vehicles.
- **Twitter** uses R to monitor user experience.
- The **US National Weather Service** uses R to predict severe flooding.
- R is being used by **The New York Times** to create infographics.
- **Google** uses R to calculate the ROI of advertising campaigns.
- **Facebook** uses R to update status and its social network graph. Also, for predicting colleague interactions with R, Facebook uses it.
- **Microsoft** uses R for the Xbox matchmaking service. Also, as a statistical engine within the Azure ML framework.
- **Mozilla** it is the foundation behind the Firefox web browser, uses R to visualize Web activity.

R Data Types

- What is R Data Types?

  It can handle complex statistical operations in an easy and optimized way.

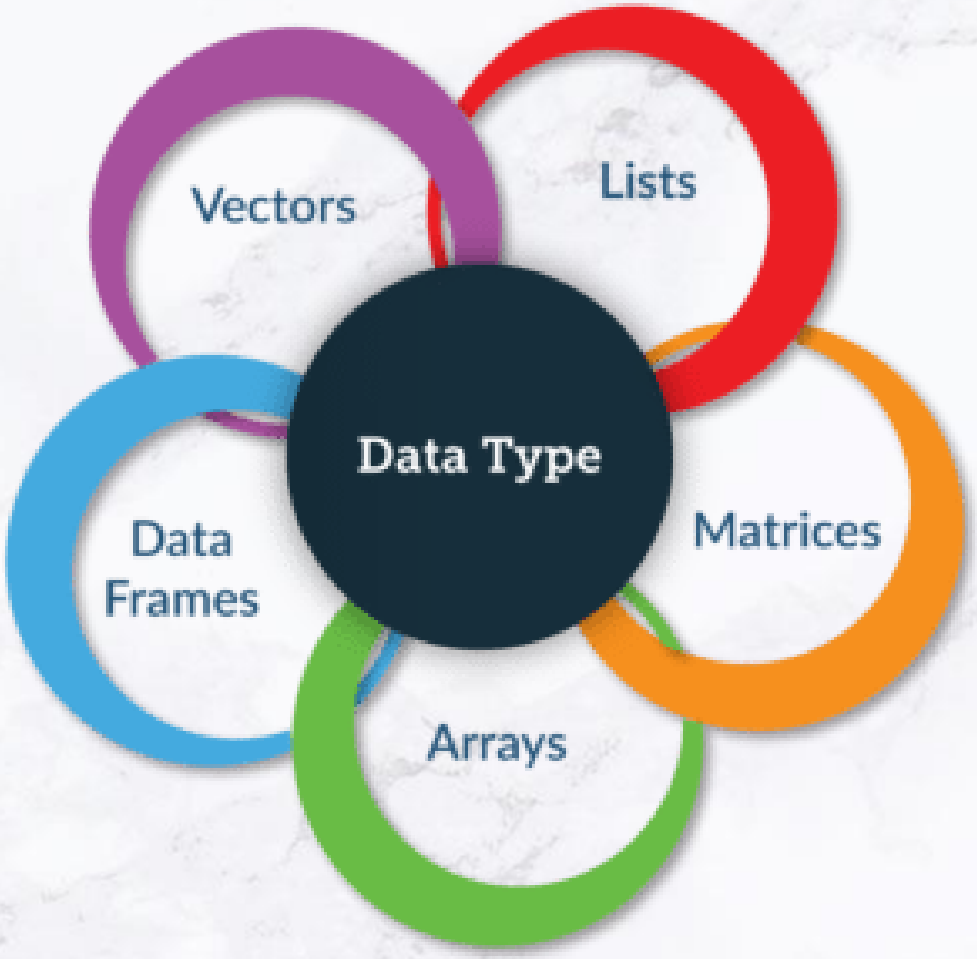
- **Vector** – A basic data structure of R containing the same type of data.

- **Lists** – Lists store collections of objects when vectors are of same type and length in a matrix.

- **Matrices** – A matrix is a rectangular array of numbers or other mathematical objects. We can do operations such as addition and multiplication on Matrix in R.

- **Arrays** – Arrays are the R data objects which can store data in more than two dimensions.

- **Data Frames** – Generated by combining together multiple vectors such that each vector becomes a separate column.



Vectors in R

- In R, Vector is a basic data structure in R that contains element of similar type.
- These data types in R can be logical, integer, double, character, complex or raw.
- In R using the function typeof() / class() one can check the data type of vector. The function length() determines the number of elements in a vector.

Create a Vector :

```
N1 <- c('white', 'pink', 'blue')

N2 <- c(15L,20L,25L)

N3 <- c(15,20,25)

N4 <- c(TRUE,FALSE,FALSE,TRUE)

N5 <- c(2+3i, 4+6i, 8+9i)

N6 <- charToRaw("Hello")


print(N1),  length(N1),
typeof(N1),  class(N1)
```

Vectors in R

- Creating a sequence

```
v <- 5:13
print(v)
```

- If the final element specified does not belong to the sequence then it is discarded.

```
v <- 3.8:11.4
print(v)
```

- Create vector with elements from 5 to 9 incrementing by 0.4.

```
print(seq(5, 9, by = 0.4))
```

Typecasting

If atomic variables aren’t similar datatype they are type casted to highest precedence of collective elements.

```
s <-c('apple','red’,5,TRUE,
2+3i)
print(s)
class(s)          > character
```

List in R

- Lists are the R objects which contain elements of different types like - numbers, strings, vectors and another list inside it.

- A list can also contain a matrix or a function as its elements. List is created using list() function.

- Type of List is List itself.

- Creating a List:

```
list_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)
print(list_data)
class(list_data)
```

List in R

- Naming a List

```
list_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8),nrow = 2), list("green",12.3))
```

- Give names to the elements in the list.

```
names(list_data) <- c("1st Quarter", "A_Matrix", "A Inner list")
print(list_data)
```

- Accessing List Elements:

Access the first element of the list.

```
print(list_data[1])
```

Access the third element.

```
print(list_data[3])
```

Access the list element using the name of the element.

```
print(list_data$A_Matrix)
```

Matrices in R

- Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout. They contain elements of the same atomic types.

- The basic syntax for creating a matrix in R is – matrix(data, nrow, ncol, byrow, dimnames)

- Creating a Matrix Elements are arranged sequentially by row.

```
M <- matrix(c(3:14), nrow = 4, byrow = TRUE)
print(M)
```

Elements are arranged sequentially by column.

```
N <- matrix(c(3:14), nrow = 4, byrow = FALSE)
print(N)
```

Matrices in R

- Defining Row & Column names

```
rownames = c("row1", "row2", "row3", "row4")
colnames = c("col1", "col2", "col3")

P <- matrix(c(3:14), nrow = 4, byrow = TRUE,
dimnames = list(rownames, colnames))

print(P)
```

- Accessing Elements of Matrix Access the element at 3rd column and 1st row.

```
print(P[1,3])
```

- Access the element at 2nd column and 4th row.

```
print(P[4,2])
```

- Access only the 2nd row.

```
print(P[2,])
```

- Access only the 3rd column.

```
print(P[,3])
```

Array in R

- Arrays are the R data objects which can store data in more than two dimensions.
- For example - If we create an array of dimension (2, 3, 4) then it creates 4 rectangular matrices each with 2 rows and 3 columns.
- An array is created using the **array()** function. It takes vectors as input and uses the values in the **dim** parameter to create an array.

- Creating an array

Create two vectors of different length

```
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)
```

Use vectors as input to the array.

```
result <- array(c(vector1,
vector2),dim = c(3,3,2))
print(result)
```

Variables

- A variable provides us with named storage that our programs can manipulate.

- A variable in R can store an atomic vector, group of atomic vectors or a combination of many R-Objects.

- A valid variable name consists of letters, numbers and the dot or underline characters. The variable name starts with a letter or the dot not followed by a number.

Variable Name	Validity	Reason
var_name2.	Valid	Has letters, numbers, dot and underscore.
var_name%	Invalid	Has the character '%'. Only dot(.) and underscore allowed.
2var_name	Invalid	Starts with a number
.var_name , var.name	Valid	Can start with a dot(.) but the dot(.)should not be followed by a number.
.2var_name	Invalid	The starting dot is followed by a number making it invalid.

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:
  1. Arithmetic Operators
  2. Relational Operators
  3. Logical Operators
  4. Assignment Operators
  5. Miscellaneous Operators

Arithmetic Operators

Operator	Description	Example
+	Adds two vectors	v <- c(2,5.5,6) t <- c(8, 3, 4) print(v+t)
-	Subtracts second vector from the first	print(v-t)
*	Multiplies both vectors	print(v*t)
/	Divide the first vector with the second	print(v/t)
%%	Give the remainder of the first vector with the second	print(v%%t)
%/%	The result of division of first vector with second (quotient)	print(v%/%t)
^	The first vector raised to the exponent of second vector	print(v^t)

Relational Operators

Operator	Description	Example
>	Checks if each element of the first vector is greater than the corresponding element of the second vector.	v <- c(2,5.5,6,9) t <- c(8,2.5,14,9) print(v>t)
<	Checks if each element of the first vector is less than the corresponding element of the second vector.	print(v < t)
==	Checks if each element of the first vector is equal to the corresponding element of the second vector.	print(v == t)
<=	Checks if each element of the first vector is less than or equal to the corresponding element of the second vector.	print(v<=t)
>=	Checks if each element of the first vector is greater than or equal to the corresponding element of the second vector.	print(v>=t)
!=	Checks if each element of the first vector is unequal to the corresponding element of the second vector.	print(v!=t)

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 a output TRUE if both the elements are TRUE.	v <- c(3,1,TRUE,2+3i) t<-c(4,1,FALSE,2+3i) print(v&t)
|	It is called Element-wise Logical OR operator. It combines each element of the first vector with the corresponding element of the second vector and gives a output TRUE if one the elements is TRUE.	v <- c(3,0,TRUE,2+2i) t <- c(4,0,FALSE,2+3i) print(v|t)
!	It is called Logical NOT operator. Takes each element of the vector and gives the opposite logical value.	v <- c(3,0,TRUE,2+2i) print(!v)

Operator	Description	Example
&&	Called Logical AND operator. Takes first element of both the vectors and gives the TRUE only if both are TRUE.	v <- c(3,0,TRUE,2+2i) t <- c(1,3,TRUE,2+3i) print(v&&t)
||	Called Logical OR operator. Takes first element of both the vectors and gives the TRUE if one of them is TRUE.	v <- c(0,0,TRUE,2+2i) t <- c(0,3,TRUE,2+3i) print(v||t)

Miscellaneous Operators

Operator	Description	Example
:	Colon operator. It creates the series of numbers in sequence for a vector.	v <- 2:8 print(v)
%in%	This operator is used to identify if an element belongs to a vector.	v1 <- 8 v2 <- 12 t <- 1:10 print(v1 %in% t) print(v2 %in% t)
%*%	This operator is used to multiply a matrix with its transpose.	M = matrix(c(2,6,5,1,10,4), nrow = 2,ncol = 3,byrow = TRUE) t = M %*% t(M) print(t)