Class 4

R – Data Structures

Data Structures

Programmers need to create variables to store different kinds of data.

Reserved memory location (Variables)

Data structures helps the computer for efficient memory handling and data analysis.

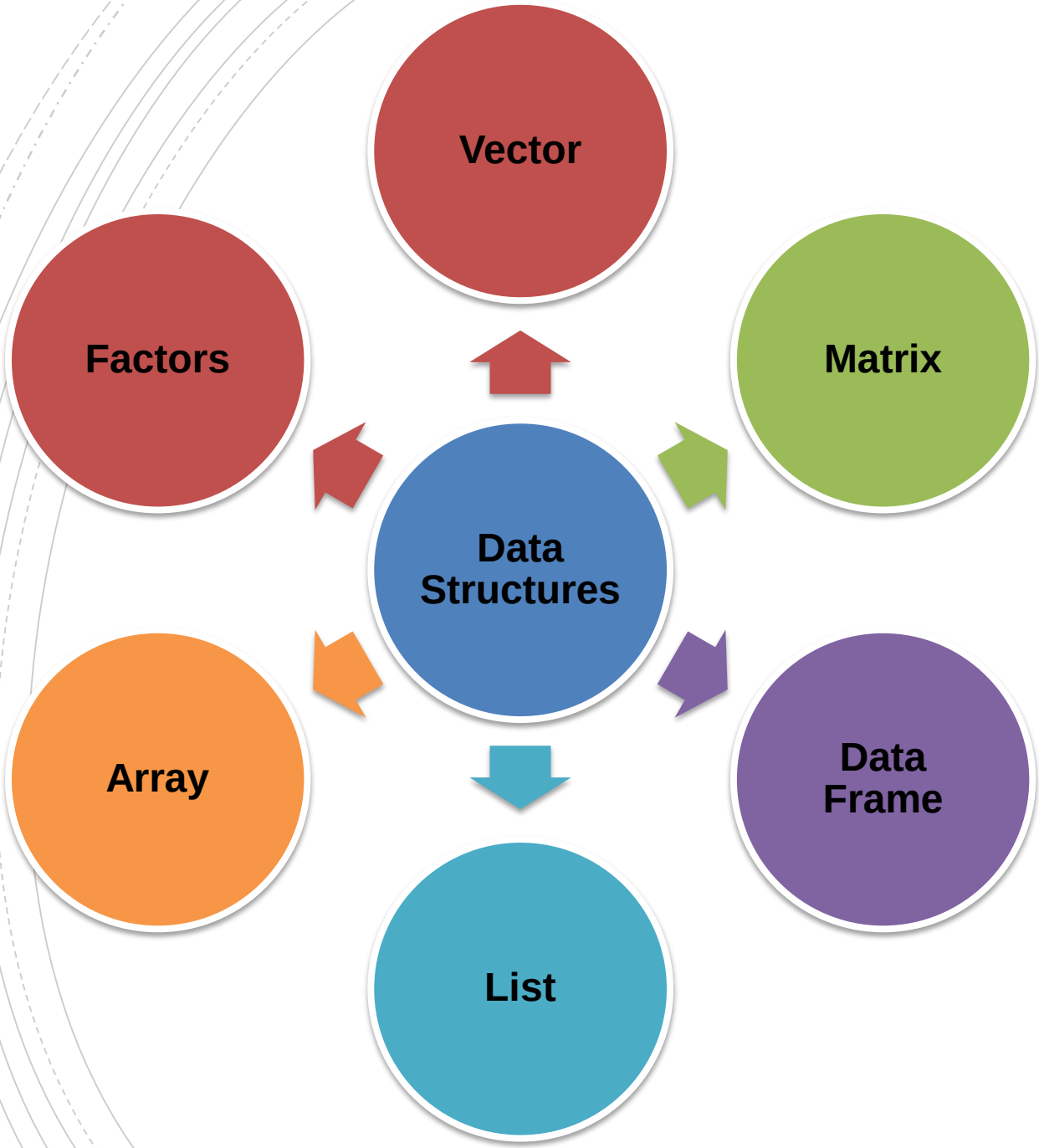
No need to declare data types in R, whereas other programming languages like C and Java need data type while creating variables.

Data Structures

Data structures are the objects.

Object – bundle of variables, functions, or methods

R has many objects.


Vector
Factors
Matrix
Data Structures
Array
Data Frame
List

Vector

- Basic data structure in R
- Collection of elements (Numeric, Integer, Character, logical)
- Two parts in vector

    Atomic Vector

    List

```
> vec <- c(1,2,3,4,5)
> vec
[1] 1 2 3 4 5
> class(vec)
[1] "numeric"
```

Vector

Sequence elements that shares the same data type.

c() used for creating vector in R

One dimensional array

Vector Creation

: operator

```
> a <- 3:10
```

seq function:

```
>a <- seq(3,10)
```

Atomic vector

- 4 types of atomic vector
- Very important in the data science

Numeric Vector

Integer Vector

Character Vector

Logical Vector

Numeric vector

- Decimal values are numeric vector
- Numeric elements stored in the numeric vectors

Ex.

```
num1 <- 45

num1_vec <- c(10,30,50,3.1,-30)

class(num1)

class(num1_vec)
```

Integer vector

- Non-fraction values are integer data
- as.integer function used to converting numeric to integer or appending L with the number

Ex.

```
num1 <- as.integer(5)

num1_vec <- c(10,30,50,3.1,-30)

num1_vec1 <- as.integer(num1_vec)

class(num1)

class(num1_vec1)
```

Character vector

- Writing character inside the “” or ‘’ quotes
- as.character() used for converting number into character

Ex.

```
char1 <- c(“ATCG”,”ATCTAG”)

num1_vec <- c(10,30,50,3.1,-30)

char_vec1 <- as.character(num1_vec)

class(char1)

class(char_vec1)
```

Logical Vector vector

- Having only 2 values (TRUE or FALSE)
- Vector contains Boolean values are known as logical vector

Ex.

```
logi1 <- c(TRUE,FALSE)

class(logi1)

a <- "ATGC"

b <- "ATGC"

c <- "TGCT"

check_seq <- c(a==b,a==c)

class(check_seq)
```

Accessing vector elements

- Indexing with Integer vector
- Specify the integer values in [ ] brackets.

Ex.

```
char1 <- c("ATCG","ATCTAG","ATGCTAG")

char1[3]
```

Accessing vector elements

- Indexing with character vector
- Specify the integer values in [ ] brackets.

Ex.

```
> myseq <- c("AUG"="Start Codon","UAG"="Stop Codon","UGA"="Stop Codon","UAA"="Stop Codon")

> myseq["AUG"]
```

Applications of vector

- ML for Principal Component Analysis
- Vector format data supplied as a input layer of neural network.
- Used to develop SVM algorithms
- Used in image recognition and text processing

List

- Special type of vector
- Generic vector containing other data types.
- Mixture of data types
- Can be any type of R objects
- list() or as.list() function helps to create list object

```
> lis <- list("Hi",3,TRUE,24L,4+3i)
> lis
[[1]]
[1] "Hi"

[[2]]
[1] 3

[[3]]
[1] TRUE

[[4]]
[1] 24

[[5]]
[1] 4+3i

> class(lis)
[1] "list"
>
```

Example

```
>logi1 <- c(TRUE,FALSE)

>vec1 <- c(1,2,2,45,6)

>char1 <- c("ATGC","ATGCTA")

>list1 <- list(logi1,vec1,char1)

>list1


>list2 <-
     list(c(TRUE,FALSE),c(1,2,3,4,5,6),c("ATGC","ATGATC"))

>list2
```

```
> logi1 <- c(TRUE,FALSE)
> vec1 <- c(1,2,2,45,6)
> char1 <- c("ATGC","ATGCTA")
> list1 <- list(logi1,vec1,char1)
> list1
[[1]]
[1]  TRUE FALSE

[[2]]
[1]  1  2  2 45  6

[[3]]
[1] "ATGC"   "ATGCTA"

> list2 <- list(c(TRUE,FALSE),c(1,2,3,4,5,6),c("ATGC","ATGATC"))
> list2
[[1]]
[1]  TRUE FALSE

[[2]]
[1] 1 2 3 4 5 6

[[3]]
[1] "ATGC"   "ATGATC"
```

Naming the list elements

```
> list2 <- list(c(TRUE,FALSE),c(1,2,3,4,5,6),c("ATGC","ATGATC"))
> names(list2) <- c("Logical_Vector","Numeric_Vector","Character_Vector")
> list2
$Logical_Vector
[1]  TRUE FALSE

$Numeric_Vector
[1] 1 2 3 4 5 6

$Character_Vector
[1] "ATGC"   "ATGATC"
```

Accessing list elements

```
> #Accessing list elements
> list2 <- list(c(TRUE,FALSE),c(1,2,3,4,5,6),c("ATGC","ATGATC"))
> list2[1]
[[1]]
[1]  TRUE FALSE
```

```
> list2 <- list(c(TRUE,FALSE),c(1,2,3,4,5,6),c("ATGC","ATGATC"))
> names(list2) <- c("Logical_Vector","Numeric_Vector","Character_Vector")
> list2["Logical_Vector"]
$Logical_Vector
[1]  TRUE FALSE

> list2$Logical_Vector
[1]  TRUE FALSE
```

Matrix

- Arranged in two-dimensional form
- Data should must be in same data type
- Collection of numbers in fixed number of rows and columns
- matrix() function helps to create matrix object.

```
> mat <- matrix(c(1,4,3,1,2,3),nrow = 2, ncol = 3)
> mat
     [,1] [,2] [,3]
[1,]    1    3    2
[2,]    4    1    3
```

Matrix

Created using vector

Matrix can contain only character or only logical values (Not much use)

User can perform addition, subtraction, multiplication and division operation

Matrix - Syntax

```
matrix(data, nrow, ncol, byrow, dimnames)
```

data – data elements in the matrix

nrow – number of rows to be created

ncol – number of columns to be created

byrow – elements are arranged by row

dimnames – names assigned to the row and column

Matrix - Example

```
> #Example Matrix
> my_matrix <- matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3, ncol = 3)
> my_matrix
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> my_matrix <- matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3, ncol = 3,byrow = T)
> my_matrix
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Matrix - Example

```
> my_matrix <- matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3, ncol = 3,byrow = T,
dimnames = list(c("Row1","Row2","Row3"),c("Col1","Col2","Col3")))
> my_matrix
     Col1 Col2 Col3
Row1    1    2    3
Row2    4    5    6
Row3    7    8    9
```

Matrix – Accessing Elements

```
> #Accessing the elements from the matrix
> #Access the element from 1st row and second column
> my_matrix[1,2]
[1] 2
> my_matrix["Row1","Col2"]
[1] 2
> my_matrix["Row1",]
Col1 Col2 Col3
   1    2    3
> my_matrix[,"Col1"]
Row1 Row2 Row3
   1    4    7
```

Matrix – Update element

```
> my_matrix
     Col1 Col2 Col3
Row1    1    2    3
Row2    4    5    6
Row3    7    8    9
> #Update Matrix Element
> my_matrix[3,3] <- 25
> my_matrix
     Col1 Col2 Col3
Row1    1    2    3
Row2    4    5    6
Row3    7    8   25
> my_matrix[my_matrix==1] <- 10
> my_matrix
     Col1 Col2 Col3
Row1   10    2    3
Row2    4    5    6
Row3    7    8   25
```

Matrix – Add Column/Row

```
> my_matrix
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> my_matrix1 <- rbind(my_matrix,c(11,12,13))
> my_matrix1
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   11   12   13
> my_matrix1 <- cbind(my_matrix1,c(14,15,16,18))
> my_matrix1
     [,1] [,2] [,3] [,4]
[1,]    1    2    3   14
[2,]    4    5    6   15
[3,]    7    8    9   16
[4,]   11   12   13   18
```

Matrix – Transpose matrix

```
> my_matrix
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> #Transpose Matrix
> t(my_matrix)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Matrix Operation - Addition

```
> mat1 <- matrix(c(1,1,1,1),nrow = 2,ncol = 2)
> mat2 <- matrix(c(2,2,2,2),nrow = 2,ncol = 2)
> mat1
     [,1] [,2]
[1,]    1    1
[2,]    1    1
> mat2
     [,1] [,2]
[1,]    2    2
[2,]    2    2
> #Matrix Addition
> mat_add <- mat1 + mat2
> mat_add
     [,1] [,2]
[1,]    3    3
[2,]    3    3
```

Matrix Operation - Subtraction

```
> mat1 <- matrix(c(1,1,1,1),nrow = 2,ncol = 2)
> mat2 <- matrix(c(2,2,2,2),nrow = 2,ncol = 2)
> mat1
     [,1] [,2]
[1,]    1    1
[2,]    1    1
> mat2
     [,1] [,2]
[1,]    2    2
[2,]    2    2
> #Matrix Subtraction
> mat_sub <- mat1 - mat2
> mat_sub
     [,1] [,2]
[1,]   -1   -1
[2,]   -1   -1
```

Matrix Operation - Multiplication

```
> mat1 <- matrix(c(1,1,1,1),nrow = 2,ncol = 2)
> mat2 <- matrix(c(2,2,2,2),nrow = 2,ncol = 2)
> mat1
     [,1] [,2]
[1,]    1    1
[2,]    1    1
> mat2
     [,1] [,2]
[1,]    2    2
[2,]    2    2
> #Matrix Multiplication
> mat_mul <- mat1 * mat2
> mat_mul
     [,1] [,2]
[1,]    2    2
[2,]    2    2
```

Matrix Operation - Division

```
> mat1 <- matrix(c(1,1,1,1),nrow = 2,ncol = 2)
> mat2 <- matrix(c(2,2,2,2),nrow = 2,ncol = 2)
> mat1
     [,1] [,2]
[1,]    1    1
[2,]    1    1
> mat2
     [,1] [,2]
[1,]    2    2
[2,]    2    2
> #Matrix Division
> mat_div <- mat1/mat2
> mat_div
     [,1] [,2]
[1,]  0.5  0.5
[2,]  0.5  0.5
```

Matrix - Applications

- Used in geology plotting
- Survey plotting
- Best element for robotic movements
- Calculating efficiency of goods and products
- 3D to 2D Projection
- Electric circuit analysis

Array

- Storing data in more than 2 dimensions.
- Collection of similar data type with contiguous memory allocation
- Data stored in matrix form (rows and columns)
- array() function help to create array object
- It takes vector as input and array will be created based on the dim parameter.

Arrays

```
> arr1 <- c(1,4,3,4)
> arr <- array(arr1,dim=c(2,2))
> arr
     [,1] [,2]
[1,]    1    3
[2,]    4    4
> 
```

Array - Syntax

```
array(data, dim=(row_size, col_size, matrices, dimnames))
```

data – data elements in the array

matrices – multi dimensional matrix

row_size – number of row elements

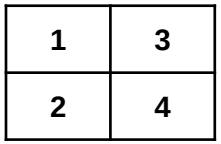
col_size – number of columns elements

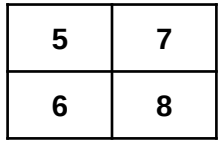
byrow – elements are arranged by row

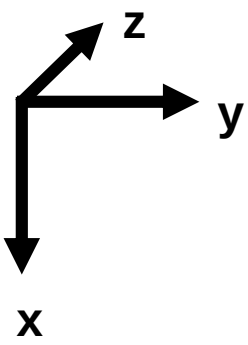
dimnames – names assigned to the row and column

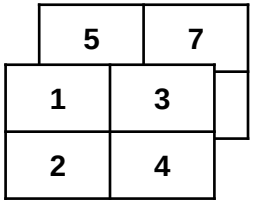
Array – Example

```
> #Array Creation
> my_array <- array(1:8,dim = c(2,2,2))
> my_array
, , 1

     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2

     [,1] [,2]
[1,]    5    7
[2,]    6    8
```


1
3
2
4


5
7
6
8


z
y
x


5
7
1
3
2
4

Array – Accessing element

```
> #Array Creation
> my_array <- array(1:8,dim = c(2,2,2))
> my_array
, , 1

     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2

     [,1] [,2]
[1,]    5    7
[2,]    6    8

> #Access the Elements
> my_array[1,1,2]
[1] 5
```

Data frame

- Two dimensional array like structure
- Tabular data
- Each column name will be considered as variable
- Rows names will be unique
- Column names will be non-empty
- data.frame() helps to create data frame in R

Factors

- factor() function helps to create the factor object.
- Useful for statistical modeling
- It can store both strings and integers