Variables and datatypes in R

In this lecture

- Variables
- Basic data types
- R objects
  - Vectors
  - Lists

Variables

Variables

Variable names - Rules

- **Allowed characters are Alphanumeric, '_' and '.'**
- **Always start with alphabets**
- **No special characters like !,@,#,$,….**

Examples

Correct naming:

> b2 =7

> Manoj_GDPL = "Scientist"

> Manoj.GDPL = "Scientist"

Wrong naming :

> 2b = 7

Error: unexpected input in "2b "

Predefined constants

Constant	Symbol in R
Pi	pi
letters	a,b,c,…….x,y,z
LETTERS	A,B,,…...,X,Y,Z
Months in a year	month.name, month.abb

```
Console C:/Users/Prem/Downloads/
> pi
[1] 3.141593
> letters
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i"
[10] "j" "k" "l" "m" "n" "o" "p" "q" "r"
[19] "s" "t" "u" "v" "w" "x" "y" "z"
> LETTERS
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I"
[10] "J" "K" "L" "M" "N" "O" "P" "Q" "R"
[19] "S" "T" "U" "V" "W" "X" "Y" "Z"
> month.name
 [1] "January"   "February"  "March"
 [4] "April"     "May"       "June"
 [7] "July"      "August"    "September"
[10] "October"   "November"  "December"
> month.abb
 [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun"
 [7] "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
>
```

Data types

Basic data types

Basic data types	Values
Logical	TRUE and FALSE
Integer	Set of all integers, Z
Numeric	Set of all real numbers
Complex	Set of complex numbers
Character	“a”,”b”,”c”,….,”x”,”y”,”z”,”@”,”#”,”$”, “”,”*”, “1”,”2”,… etc..

Basic data types

TASK	ACTION	SYNTAX/EXAMPLE
Find data type of object	→ *use command "typeof()"*	→ *Syntax: typeof(object)*
Verify if object is of a certain datatype	→ *use prefix "is." before datatype as command.*	→ *Syntax: is.data_type(object) Example : is.integer()*
Coerce or convert data type of object to another	→ *use prefix "as." before datatype as command.*	→ *Syntax: as.data_type(object) Example :as.logical()*

Note : Not all coercions are possible and if attempted will return "NA" as output

Sample Codes

```
Console ~/
> typeof(1)
[1] "double"
> typeof(("22-01-2001"))
[1] "character"
```

```
Console ~/
> is.character("21-11-2001")
[1] TRUE
> is.character(as.Date("21-11-2001"))
[1] FALSE
```

```
Console ~/
> as.complex(2)
[1] 2+0i
> as.numeric("a")
[1] NA
```

Basic objects

Object	Values
Vector	Ordered collection of same data types
List	Ordered collection of objects
Data frame	Generic tabular object

Vectors

Vectors

- **Vector : an ordered collection of basic data types of given length**
- **All the elements of a vector must be of same data type**

Code

```
# Vectors Example

X = c(2.3,4.5,6.7,8.9)

print(X)
```

Console Output

```
Console ~/
> # Vectors Example
> X = c(2.3,4.5,6.7,8.9)
> print(X)
[1] 2.3 4.5 6.7 8.9
>
>
```

Lists

Lists in R: create a list

- **List : a generic object consisting of an ordered collection of objects**
- **A list could consist of a numeric vector, a logical value, a matrix, a complex vector, a character array, a function, and so on**

Code

```
# List Example : Employee details
ID = c(1,2,3,4)
emp.name =c("Man","Rag","Sha","Din")
num.emp = 4
emp.list = list(ID, emp.name,num.emp)
print(emp.list)
```

Console Output

```
Console ~/
> ID = c(1,2,3,4)
> emp.name =c("Man","Rag","Sha","Din
")
> num.emp = 4
> emp.list = list(ID, emp.name,num.e
mp)
> print(emp.list)
[[1]]
[1] 1 2 3 4

[[2]]
[1] "Man" "Rag" "Sha" "Din"

[[3]]
[1] 4
```

Accessing components (by names)

- **All the components of a list can be named**
- **These components can be accessed using the given names**

Code

```
# Continue after first 4 lines of R
# code from previous example
emp.list = list("Id"= ID,
"Names" = emp.name,
"Total staff"=num.emp)
print(emp.list$Names)
```

Console Output

```
Console ~/
> # Continue after first 4 lines
> # of R code from previous example
> emp.list = list("Id"= ID,
+        "Names" = emp.name,
+        "Total staff"=num.emp)
> print(emp.list$Names)
[1] "Man" "Rag" "Sha" "Din"
>
>
```

Accessing components (indices)

To access top level components, use double slicing operator " [[ ]]" and for lower/inner level components use "[ ]" along with "[[ ]]"

Code

```
# continuing from previous
# code
print(emp.list[[1]])
print(emp.list[[2]])
print(emp.list[[1]][1])
print(emp.list[[2]][1])
```

Console Output

```
Console ~/
> print(emp.list[1])
$Id
[1] 1 2 3 4

> print(emp.list[2])
$Names
[1] "Man" "Rag" "Sha" "Din"

> print(emp.list[[1]][1])
[1] 1
> print(emp.list[[2]][1])
[1] "Man"
>
```

Manipulating lists

A list can be modified by accessing components & replacing them

Code

```
# continuing from previous code
emp.list['Total staff']=5
emp.list[[2]][5]="Nir“
emp.list[[1]][5]=5
print(emp.list)
```

Console Output

```
Console ~/
> # continuing from previous code
> emp.list['Total staff']=5
> emp.list[[2]][5]="Nir"
> emp.list[[1]][5]=5
> print(emp.list)
$Id
[1] 1 2 3 4 5

$Names
[1] "Man" "Rag" "Sha" "Din" "Nir"

$`Total staff`
[1] 5
```

Concatenation of lists

Two lists can be concatenated using the concatenation function,

c(list1, list2)

Code

```
# continuing from previous code
emp.ages = list("ages" =
c(23,48,54,30,32))
emp.list= c(emp.list , emp.ages)
print(emp.list)
```

Console Output

```
Console ~/
> emp.ages = list("ages" = c(23,48,5
4,30,32))
> emp.list= c(emp.list , emp.ages)
> print(emp.list)
$Id
[1] 1 2 3 4 5

$Names
[1] "Man" "Rag" "Sha" "Din" "Nir"

$`Total staff`
[1] 5

$ages
[1] 23 48 54 30 32
```