**ASSIGNMENT OPERATOR**

x<-1, 1 is assigned to x.

x<-1:50

x<-"hello"

**OBJECTS**

* 5 types character, numeric, integer, complex, logical (T/F)
* Vector
* List can comprises of different type of datatypes.
* 1L for number in integer
* Attributes
* name, dimnames
* dimensions (matrices, arrays)
* class
* length
* Other user-defined attributes /metadata
* A function attributes () is used to modify or set attribute for an R objects

**R DATA TYPES: VECTORS AND LISTS**

* By using c (concatenation)

x<- c (0.5, 0.6) ##numeric

x<- c (TRUE, FALSE) ##logical

x<- c (T, F) ##logical

x<- c ("a","b") ## character

x<- 1:15 ##integer

x<- c (1+4i, 2+7i) ##complex

* By using vector function

x<- vector("numeric”, length=5)

* Mixed object type vectors

x<- c ("a", 0.5) ## character- converted into least memory type variable

x<- c (T, 5) ## True will converted into 1

* Explicit conversion by using as.\_\_(x)

x<-1:5

as.logical(x)

illogical conversion results into NA

**Lists**

x<- list ("a", 1, T)

**Matrix**

> m<- matrix (nrow=2, ncol=3)

> m

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

[1,] NA NA NA

[2,] NA NA NA

> attributes(m)

$dim

[1] 2 3

> dim(m)

[1] 2 3

* matrix are filled column-wise

> m<-matrix(1:6,nrow=2,ncol=3)

> m

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

[1,] 1 3 5

[2,] 2 4 6

* transforming a vector into matrix

> m<-1:10

> m

[1] 1 2 3 4 5 6 7 8 9 10

> dim(m)<-c(2,5)

> m

[,1] [,2] [,3] [,4] [,5]

[1,] 1 3 5 7 9

[2,] 2 4 6 8 10

* by using cbind and rbind

> x<-1:3

> y<-10:12

> x

[1] 1 2 3

> y

[1] 10 11 12

> rbind(x,y)

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

x 1 2 3

y 10 11 12

> cbind(x,y)

x y

[1,] 1 10

[2,] 2 11

[3,] 3 12

**Factors**

> x<- factor(c ("yes", "yes", "no", "yes", "no"))

> x

[1] yes yes no yes no

Levels: no yes

* To change levels, levels attribute can be used

> x<- factor (c ("yes", "yes", "no", "yes”, "no"), levels=c("yes", "no"))

> x

[1] yes yes no yes no

Levels: yes no

> table (x)

x

yes no

3 2

> unclass(x)

[1] 1 1 2 1 2

attr(,"levels")

[1] "yes" "no"

**MISSING VALUES**

* NA, NaN
* function is.na(x) and is.nan(x) can be used to find missing values

**DATA FRAMES**

* Use to represent the tables
* Each column of table is a list and has same length
* Attribute of row.names can be used to name the row.

> x<-data.frame(id=1:4, header = c(T,T,F,T))

> x

id header

1 1 TRUE

2 2 TRUE

3 3 FALSE

4 4 TRUE

> nrow(x)

[1] 4

> ncol(x)

[1] 2

**NAME FUNCTION**

* naming a variable

> x<-1:3

> x

[1] 1 2 3

> names(x)<-c("ho","ha","hi")

> x

ho ha hi

1 2 3

* naming a list

> y<-list(ho= c("a","b"),ha=1:3)

> y

$ho

[1] "a" "b"

$ha

[1] 1 2 3

* naming a matrix

> m=matrix (1:6,nrow=2,ncol=3)

> m

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

[1,] 1 3 5

[2,] 2 4 6

> dimnames(m)<-list(c("a","b"),c("l","m","n"))

> m

l m n

a 1 3 5

b 2 4 6

**READING DATA INTO R**

* read.table, read.csv for reading tabular data, reads text file and return dataframe in R
* readLines reading any type of file into character datatype in R
* source for reading r codes
* dget reading r code
* load loading in saved workspace (into console)
* unserialize for rading single R objects in binary form

**Analogas function for writing in R**

* write.table, writeLines, dump, dput, save, serialize

**read.table**

attributes:->

* file (name of the file)
* header (logical indicating if the table has a header)
* sep (a string indicating how the data are separated)
* colClasses (a character vector indicating the class of each column)
* comment.char (a character string indicating the comment character)
* nrows (the no of rows )
* skip (no fo lines to be skipped from the begining)
* stringsAsFactors (whwther character variables to be coded as factors)
* For reading large tables use nrows to read desired no of rows
* data are stored in RAM

**use of dget() and dput()**

> x<-data.frame(a=1,b="a")

> x

a b

1 1 a

> dput(x,"x.R")

> y<-dget("x.R")

> y

a b

1 1 a

**Example showing us of dump**

> x<-"good"

> y<-list(a=1,b="c")

> y

$a

[1] 1

$b

[1] "c"

> x

[1] "good"

> dump(c("x","y"), file = "y.R")

> rm(x,y)

> x

Error: object 'x' not found

> y

Error: object 'y' not found

> source("y.R")

> x

[1] "good"

> y

$a

[1] 1

$b

[1] "c"

**Interface to outside world**

* file
* gzfile , for compressed file with gz algorithm
* bzfile , for compressed file with bz2 algorithm
* url , for connecting to the url

**file**

> str(file)

function (description = "", open = "", blocking = TRUE, encoding = getOption("encoding"),

raw = FALSE, method = getOption("url.method", "default"))

* discription is the name of the file
* open = r,w,a,rb,wb,ab in last three binary mode is used

**using file**

> dir()

[1] "r\_example1.csv" "r\_lecture\_note.docx" "r\_lecture\_note.txt"

[4] "x.R" "y.R"

> connec<-file("r\_example1.csv","r")

> y<-read.csv(connec)

> y

id name marks

1 1 e 95

2 2 d 99

3 3 f 96

4 4 b 100

> close(connec)

**Reading lines of a text file**

> connec<-gzfile("example2.gz")

> x<-readLines(connec,10)

> x

[1] "a" "s" "d" "f" "f" "g" "h" "h" "j" "1"

> y<-readLines(connec)

> y

[1] "a" "s" "d" "f" "f" "g" "h" "h" "j" "1" "2" "3" "4" "5" "6" "7"

[17] "78" "9"

> close(connec)

**Reading text from web**

> x<-url("https://en.wikipedia.org/wiki/Parliament\_of\_India","r")

> y<-readLines(x,5)

> y

[1] "<!DOCTYPE html>"

[2] "<html class=\"client-nojs\" lang=\"en\" dir=\"ltr\">"

[3] "<head>"

[4] "<meta charset=\"UTF-8\"/>"

[5] "<title>Parliament of India - Wikipedia</title>"

**Subsetting**

* Can be done by using [],[[ ]],$

> x<-c("a","b","s","d")

**numeric indexing**

> x[1]

[1] "a"

> x[2]

[1] "b"

> x[1:3]

[1] "a" "b" "s"

**logical indexing**

> x[x>"b"]

[1] "s" "d"

> logicalvector<-x>"a"

> logicalvector

[1] FALSE TRUE TRUE TRUE

> x[logicalvector]

[1] "b" "s" "d"

**subsetting a list**

> ##subsetting a list

> x<-list(foo=1:4,bar=0.6)

> x[1]

$foo

[1] 1 2 3 4

> x[2]

$bar

[1] 0.6

> x[[1]]

[1] 1 2 3 4

> x[[2]]

[1] 0.6

> x$foo

[1] 1 2 3 4

> x$bar

[1] 0.6

> x[foo]

Error: object 'foo' not found

> x["foo"]

$foo

[1] 1 2 3 4

> x[c(1,2)]

$foo

[1] 1 2 3 4

$bar

[1] 0.6

> name<-"foo"

> x[name]

$foo

[1] 1 2 3 4

> x$name

NULL

**Subsetting a matrix**

> m

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

[1,] 1 4 7 10

[2,] 2 5 8 11

[3,] 3 6 9 12

> m[1,2]

[1] 4

> m[ ,2]

[1] 4 5 6

> m[1,]

[1] 1 4 7 10

> m[1,2,drop = FALSE]

[,1]

[1,] 4

> m[1, ,drop =FALSE]

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

[1,] 1 4 7 10

**PARTIAL MATCHING**

> ## PARTIAL MATCHING

> x<-list(deepak=1:10,godfather=5:12,geek=7)

> x$g

NULL

> x$d

[1] 1 2 3 4 5 6 7 8 9 10

> x$ge

[1] 7

> x$go

[1] 5 6 7 8 9 10 11 12

> x[["d"]]

NULL

> x[["d",exact=FALSE]]

[1] 1 2 3 4 5 6 7 8 9 10

**REMOVING NA VALUES FROM DATA**

by using is.na(x)

> x<-c(1,2,NA,3,4,NA,5)

> x

[1] 1 2 NA 3 4 NA 5

> bad<-is.na(x)

> x[!bad]

[1] 1 2 3 4 5

> bad

[1] FALSE FALSE TRUE FALSE FALSE TRUE FALSE

> x[!is.na(x)]

[1] 1 2 3 4 5

> y<-c("a","d","f",NA,"h","i")

> good<-complete.cases(x,y)

Error in complete.cases(x, y) : not all arguments have the same length

> y<-c(y,"b")

> y

[1] "a" "d" "f" NA "h" "i" "b"

> good<-complete.cases(x,y)

> good

[1] TRUE TRUE FALSE FALSE TRUE FALSE TRUE

> x[good]

[1] 1 2 4 5

> y[good]

[1] "a" "d" "h" "b"

**VECTORIZED OPERATION**

> ##VECTORIZED OPERATION

> x<-1:4

> y<-6:9

> x+y

[1] 7 9 11 13

> x\*y

[1] 6 14 24 36

> x/y

[1] 0.1666667 0.2857143 0.3750000 0.4444444

> x.3

Error: object 'x.3' not found

> x>3

[1] FALSE FALSE FALSE TRUE

> m1<-matrix(1:4,nrow=2,ncol=2)

> m2<-matrix(5:8,nrow=2,ncol=2)

> m1+m2

[,1] [,2]

[1,] 6 10

[2,] 8 12

> m2-m1

[,1] [,2]

[1,] 4 4

[2,] 4 4

> m1\*m2

[,1] [,2]

[1,] 5 21

[2,] 12 32

> m1/m2

[,1] [,2]

[1,] 0.2000000 0.4285714

[2,] 0.3333333 0.5000000

> m1%\*%m2

[,1] [,2]

[1,] 23 31

[2,] 34 46