Data Science Notes

Contributor: Kirtika

[SGGSCC DU]

Computer Science Notes

21st Jan '19

Page No.
Date

Vectors

Creating vectors

1) using (c) function

```
> x <- c(1,7,6,9)
> x
[1] 1 7 6 9
```

class() & typeof() are same funct^n

```
>typeof(x)
[1] Numeric
```

eg `>y <- c(1,2, TRUE, "a")` & (# not error.

'R' interpreter

precedance = logical → numeric → character

highest precedance.

```
> y
[1] "1" "2" "TRUE" "a"
```

2) Using infix function :

restrict^n = works only on continuous values.

i.e. gap b/w data members = ±1

if we want i/p values then using 'c' funct^n

```
> v <- c(1,2,3, .. , 10)
   =
> v <- 1:10
```

means start with '1' and '10'

':' means '1' gap b/w members.

~~# gap cannot be greater than 1~~

eg `> y <- 2:-2` (:) y concatenates two diff. stat.

```
> y <- 2:-2
> y
```

but output of only 'a' is shown.

```
O/P [1] 2 1 0 -1 -2
```

step size is still '1'

Page No.
Date

eg > z ← ~~3.4~~ 3.4 : 10.2 ; z
[1] 3.4 4.4 5.4 6.4 7.4 8.4 9.4

3) using sequence funct^n (seq()) length.out divides into (to − from / length.out − 1)

Step size > 1

Syntax : seq(from = a, to = b, [by = step-value, length.out = l])
default. stepsize = 1 ; by → stepindex ; length.out → length of vector

eg eg > x ← seq(1,7); x
[1] 1 2 3 ... 7

x ← 1:7
or
x ← c(1,2,3,4,5,6,7)

eg > x ← seq(1, 10, by = 2); x
[1] 1 3 5 7 9

eg > x ← seq(1, 10, length.out = 5); x
[1] 1 2 3 4 5
→ x will have length of 5 (breaks into 5 equal parts & assign it to x)

eg > x ← seq(1, 10, length.out = 10)
→ x have length of 10
[1] 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0

Note : If we specify all 4 parameters then seq() doesn't work as length.out breaks the values accordingly so stepsize() is not needed.

4) using rep()

Syntax : > f ← rep(0, 5)
value → 0 ; 5 → how many times value is repeated.
> f
[1] 0 0 0 0 0

```
> f1 <- rep(1:3, 4); f1
[1]  1 2 3  1 2 3  1 2 3  1 2 3

> length(f1)                      # find no. of vectors
[1]  12

> h <- rep(4:6, 1:3); h
          4,5,6   1 2 3
[1]  4  5 5  6 6 6
```

* **recycling**

eg: h1 <- rep(1:10, 1:5)

10 → 5 (×2) — if factor of 5

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

if not factor

eg: h1 <- rep(1:10, 1:4)

warning is shown

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

e → > x <- rep(1:2, each=2); x

↳ each individual element is repeated 2 times

```
O/P: [1]  1 1  2 2
```

→ times

```
> y <- rep(1:2, 2); y           # rep(1:2, times=2)
[1]  1  2  1  2
```

Date

⇒ Indexing a vector (same as indexing array many lang)

a[index]

① ~~doesn't does~~ starts with '1'

② specify multiple index value.

eg> a[c(1,2)] # returns two elements(corr. to 1st & 2nd index of 'a' vector

or> a[1]
> a[2]

③ Index values can be '-ve'

eg > a ← c(1,2,3)
> a[-1]
[1] 2 3

[1] [2] [3]
| 1 | 2 | 3 |

all elements of vector should be returned except at index '1'.

possible in 'R'

```
for (i=1; i>5; i++)
{ ~~if (i%2~~ cout << a[i%2==0];
}
```

~~Que~~

* ① eg > d ← seq(1, 5, by=0.5)
> a[3]
> d[5:7]
> d > 2.8 # loops for every value in d for 2.8
> d[d>2.8]

O/P: d=(1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0)
[1] 2.0
[1] 3.0 3.5 4.0
[1] FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE
[1] 3.0 3.5 4.0 4.5 5.0
> length(d[d>2.8])

Page No.
Date

(2) t ← c(0, 1, 2, 3, 4, 5, 6)
x ← t[c(2, 3, 6)]
print(x)
v ← t[c(TRUE, FALSE, FALSE)]
print(v)
x ← t[c(-2, -3)] print(x)

Solution
[1] 1 2 5
[1] TRUE FALSE FALSE 0
[1] 0 3 4 5 6

v ← t[c(TRUE, false, false, true)]; v.

eg. OIP [1] 0 3.

R-Studio

Write your script	Variables in work environm.
x ← c(0,1,2)	x.
Console	Current packages, libraries, ...
OIP of ~~comm~~ script	

> ls() # listing of all current variables (object) in environment
> rm(x) # 'x' is deleted from ~~m~~ env.
> rm[c(x, y)] # x, y deleted.
> rm(x)
> rm(y)

* ~~>~~ seq(1, 10, length.out = 5); d.
[1] 1.00 3.25 5.50 7.75 10.00
length.out includes 1 & 10 and after including these 2

Date

3 more elements are to be included in a way that all those are equally divided.

22nd Jan '19

```
> assign ("y", c(1,2))   # y ← c(1,2) or y=c(1,2)
> y
[1] 1 2
> x ← c(1,2,9,6,5)
> sort(x); x              # sort values of vector in acc. order
[1] 1 2 5 6 9

> sort(x, decreasing = TRUE)
[1] 9 6 5 2 1

> sum(x)                  # sum of all elements of x
> mean(x)                 # gives mean. sum(x)/length(x)

> var(x)                  # st. deviat" of x.
> y ← c(x, 0, x); y
[1] 1 2 9 6 5 0 1 2 9 6 5.

> 1/y → z; z
[1] 1  1/2  1/9  1/6  1/5

> a ← c(1,2,3,1,2,3)
> table(a)                # values of all values w.r.t its multiplicity.
a
1 2 3   → elements in vector
2 2 2   → multiplicity (freq) of those elements.
```

NaN → Not a No.
Inf ⇒ Infinity
-Inf → $-\infty$
NA → same as null

Page No.
Date

```
>unique(a)
NA
```
gives only unique vectors
NA → not available

```
>x ← c(1,2,NA,3)
>length(x)
[1] 4
```
NA is element but no value is assigned

```
>x[1] ← 3.; x
[1] 3 2 NA 3
```

```
> x[10] =
NA
```
doesn't give errors for out of bound index.

* Vectors in R are dynamic i.e., length can be ↑

```
>length(x) ← 6; x
[1] 3 2 NA 3 NA NA
```
new length of x = 6

NOTE * The vectors consisting 'NA' as elements won't work with all functions
e.g. >x ← c(1,2,NA,3)
>sum(x)
[1] NA (as NA + 3 = NA)
→ to avoid NA use

check
```
> sum(x, rm.na=TRUE)
    or
    na.rm
[1] 6
```
NA is excluded

```
> mean(x, rm.na=TRUE)
    or
    na.rm
```

Page No.
Date

> max(x)
> min(x)
} both will work but other functⁿ won't for x

Ques (1) > x=1:8 x=1 2 3 4 5 6 7 8
> x[4]=-9; x
> x[c(1,4)]
> x[x>3]
> x[x>3 & x<8]
> x[x%%2==0]
mod operator in R

Soln 1 2 3 -9 5 6 7 8
> 1 2 3 -9 5 6 7 8
> 1 -9
> 5 6 7 8
> ~~numeric(0)~~ [1] 5 6 7
> [1] 2 6 8

(2) > y<-(1:5)^2; y → (1,2,3,4,5)^2 = 1 4 9 16 25
> y[2:4]
> y[c(-2,-4)]
> y[c(1,-1)] or y[c(4,-1)] X not allowed.
> y[-1.9] # y[-1]
> y[0] # NA
> y[1.9] # 1.9 is truncated to 1 y[1]
> y+1
> y+5.1

mixing of (+ve, -ve) indexing is not allowed.

Page No.
Date

(3) > a ← seq(1,9, by=2); a
> rep(1:3, length.out=9) → b; b
> a ← a[1:4]; a.
> a + 2*1:5

2x(1,2,3,4,5) = (2,4,6,8,10)

Soln (2) [1] 1 4 9 16 25.
[1] 4 9 16
[1] 1 9 25
Error
[1] 4 9 16 25
[1] NA
[1] 1
[1] 2 5 10 17 26
[1] 6.1 9.1 14.1 21.1 30.1

(3) [1] 1 3 5 7 9 11 13 15 17 19
[1] 1 2 3 1 2 3 1 2 3
[1] 1 8
[1] 3 5 7 9 11

⇒ Functions

1) abs(x) # absolute
sqrt(x)
ceiling(x)
floor(x)
round(x, digits=n)
trunc(x)

} works on vectors, arrays, list etc.

#2018

Ques Given a vector 'f' as f ← c(0,1,1,2,3,5,8,13,21,34)
what is the output of following 'R' commands

(i) >f[1:3]
(ii) >f[-(1:3)]
(iii) >f<10
(iv) f[f<10]
(v) f[f%%2==0]

Soln

(i) 0 1 1
(ii) 2 3 5 8 13 21 34
(iii) T T T T T T T F F F
(iv) 0 1 1 2 3 5 8
(v) 0 2 8 34

29th Jan '19

Page No.
Date

Matrix → 2D object

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

default is '1' — default is "false"

[i, j]

eg: > a ← matrix(1:9, 3, 3, FALSE); a

E	[,1]	[,2]	[,3]
[1,]	1	4	7
[2,]	2	5	8
[3,]	3	6	9

> a ← matrix(1:9, 3, 3, TRUE); a

1	2	3
4	5	6
7	8	9

> c ← matrix(1:3, nrow=3, TRUE); c

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

> c ← matrix(1:5, nrow=3, ncol=3, byrow=TRUE); c

Warning: In matrix(1:5, nrow=3, ncol=3, byrow=TRUE) data length [5] is not a multiple of the no. of rows [3]

5 is not multiple of 3 so it is a warning case we write output warning

	[,1]	[,2]	[,3]
x [1,]	1	2	3
y [2,]	4	5	1
z [3,]	2	3	4

Page No.
Date

```
> row_names ← c("r1", "r2", "r3")
> col_names ← c("c1", "c2", "c3")
> p ← matrix(9:1, 3, 3, TRUE, dimnames = list(row_names, col_names)); p
```

dimnames is either a list/vector.
list() c()
list(rownames, colnames)
if not col_name then only rows will be given name.

```
    c1  c2  c3
r1   9   8   7
r2   6   5   4
r3   3   2   1

> colnames(p)
[1] "c1" "c2" "c3"
> rownames(p)
[1] "r1" "r2" "r3"
> rownames ← c("A", "B", "C")
> colnames ← c("X", "Y", "Z")
> p
  X  Y  Z
A 9  8  7
B 6  5  4
C 3  2  1

> nrow(p)        # takes matrix & returns no. of row
[1] 3
> ncol(p)        # "  "  "  "  "  " col
[1] 3
> dim(p)         # "  "  "  "  "  " dimension
[1] 3 3
```

Page No.
Date

→ Changing vector into matrix (2 ways)

1) 1st method

```
> x <- c(1,2,3,4,5,6); x        (Recycling allowed)
[1] 1 2 3 4 5 6
> dim(x) <- c(2,3)              <- default filled by col
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

> class(x)
[1] "matrix"
```

2) 2nd method [by default create a matrix of form nx1]

```
> y <- 1:5; y
[1] 1 2 3 4 5
> class(y)
[1] "numeric"
> m <- as.matrix(y); m
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
[5,]    5
```

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Page No.
Date

Combining vectors to form matrix

- Functions cbind() & rbind() are used to combine vectors to form a matrix.
  cbind() will combine n vectors by column
  and rbind() " " " " row

```
> x ← 1:3
> y ← 4:6

> m1 ← cbind (x, y) ; m1
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

> m2 ← rbind (x, y) ; m2
     1  2  3
     4  5  6
```

Accessing Elements of a matrix

Elements of a matrix can be accessed by using an index of the form [i, j]

```
> a
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Page No.
Date

> a[2,1]
[1] 4
> a[2,]
[1] 4 5 6

> a[,3]
[1] 3 6 9

(i = row, j = col)

> a[c(1,2) ,c(2,3)] #(multiple indexing in 1 statement)

[,1] [,2] → it has nothing to do with a ij

[1,] 2 3
[2,] 5 6

→ > a [c(3,2),] # every col

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

→ > a [c (3,2)] # by default o/p w.r.t col '1' only

[,1]
[1,] 7
[2,] 4

> a[,] or a. #all rows & all cols

> a[-1,]

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

Page No.
Date

* It is possible to index a matrix with a single vector while indexing in such a way it acts like a vector formed by stacking the cols of the matrix one after another.
The result is returned as a vector.

eg: > a [1:4] ≡ a[c(1,2,3,4)]
when vector is used for indexing then elements are stored in col wise
i.e., 1, 4, 7 2 5 8 3 6 9 → vector

> [1] 1 4 7 2

→ * Using logical vectors as index

* Here rows & cols where the value is true will be returned. Also the logical vectors will be recycled if necessary.

* Logical vectors can be mixed with integer vectors for indexing

Recycling will be done in logical indexing on its own

eg: > x

	[,1]	[,2]	[,3]
[1,]	4	8	3
[2,]	6	0	7
[3,]	1	2	9

> x [c(TRUE, FALSE, TRUE), c(TRUE, TRUE, FALSE)]
≡
x[c(1,3), c(1,2)]

Page No.
Date

```
     [,1]  [,2]
[1,]   4     8
[2,]   1     2
```

> x[c(TRUE,FALSE), c(2,3)] # Recycling of index
=
x[c(TRUE,FALSE,TRUE), c(2,3)]

```
     [,1]  [,2]
[1,]   8     3
[2,]   2     9
```

> x [c(TRUE,FALSE)]

means all elements of x are stack together as element into a vector.

so vector will be 4 6 1 8 0 2 3 7 9.

so the recycling takes place as

T F T F T F T F T.

4 6 1 8 0 2 3 7 9.

```
[1] 4 1 0 3 9
```

> x[x>5] #(store as 1-D & print elements greater than 5)

```
[1] 6 8 7 9
```

> x

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

> det(x) ## determinant is returned
[1] 10

A^{-1} = adj A / |A|

Page No.
Date

```
> solve(x)                    # inverse of a matrix
          [,1]    [,2]
[1,]     1.1     -0.3
[2,]    -0.4      0.2

> sum(x)                      # sum of all elements of mat.
[1] 20

> product(x)                  # product " " " " "
[1]

> sum[x[1,]                   # sum of 1st row all (obs)
[1] 5

> y ← diag(3); y              # used to create diag. mat.
          [,1]  [,2]  [,3]    of 3x3 order.
[1,]       1     0     0
[2,]       0     1     0
[3,]       0     0     1

> t(x)                        # transpose.
          [,1]  [,2]
[1,]       2     4
[2,]       3     11

> y ← matrix(1:4, 2, 2); y
          [,1]  [,2]
[1,]       1     3
[2,]       2     4

> x+y                         # element wise addition
> x-y                         # " " subtn.
> x/y                         # " " division
```

Page No.
Date

> x * y # (element wise ~~matrix~~ multiplicat^n

> x %.* %. y # matrix multiplicat^n.

2018
Ques: write a R script used to solve a system of linear eqn.

Soln
$$a_1x_1 + b_1x_2 = c_1$$
$$a_2x_1 + b_2x_2 = c_2$$

$$A \rightarrow \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \rightarrow B$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}^{-1} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

1→ A → matrix (.....)
2→ B → matrix (.....)
3→ y ← Solve(A)
4→ ans ← y %.* %.B # inverse of A.

OR Solve(A, B) # (Calculates inverse of A & multiplies it by B)

⇒ Reading user Input

* n ← as.integer(readline(prompt = "Enter no"))

prompt user with message & store value into 'n'

Type casting into integer.

used to read 1 line input → input is returned as "string"

either a no. or a string

* more than 1 element (no. or string)

> x ← scan() ↵
1: 3 ↵
2: 7
ctrl+Z / ctrl+D # to stop entering input.

Read 2 inputs # msg shown on console

Ques Take 2 vectors of size '2' as an input from the user and combine those vectors to form a matrix where the elements will be filled by row.

Soln

```
> a ← scan()
1: 1
2: 2
3: ctrl+Z / ctrl+D
read 2 items

> b ← scan()
1: 3
2: 4
3:
read 2 items.

> x ← rbind(a, b)
```


4th Feb'19

Page No.
Date

Control structures in R

```
if(condition){
==
}
```

```
if(condn){
==
}else{
==
}
```

```
if(condn){
==
}else{
==
}
```

```
if(condn){
==
}else if(condn){
==
}else{
==
}
```

→ switch (exp, case1, case2, ...) → NO DEFAULT CASE

No multiple statements in 1 case.
No fall through since no specification of break.
If case1 is matched then execute case1.

→ Loops in R

```
for(value in vector){
==
}
```

eg.
```
for (i in 1:3){
  print i
}
```

Page No.
Date

→ while (test exp) {
≡
}

→ repeat {
≡
if (cond) {
break
}
}

same as do while.

do {
≡
while (=)
}

→ Break and next

```
x=1:10
for(i in x){
  if(i==2){
    break
  }
  print(i)
}
```

[1] 1

```
x=1:4
for (i in x){
  if (i==2) {
    next    // same as continue
  }
  print(i)
}
```

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

Q. Write a 'R' prog. to take a matrix as an input from user and print it's sum. Also take the dimensions of the matrix as an input.

```
r <- readline
r <- as.integer(readline(prompt="Enter rows"))
c <- as.integer(readline(prompt="Enter col"))
for (i in
```

length(m) ← r*c

m ← vector()
dim(m) = c(r,c)
print(m)
for(i in r){
  for(j in c){
    m[i,j] ← as.integer() // since no elements in vector.
  }
}
sum(m)

// dim.of m => r x c
eg >c=1:4
>dim(c)=c(2,2)
[,1] [,2] [,3]
[1,] NA NA ....
[2,]
[3,]

// or rowSums(m)
colSums(m)

II

a ← vector()
length(a) ← 10
print(a)
for(i in a){
  a[i] ← ....
}

// [1] NA NA ....
// input vectors.

III

a, b vector
↳ c (vector) ⟶ d dim

Q write a prog. to input a matrix A and prog that $A \cdot A^{-1} = I$.

Q write a prog. to find sum of main diag. of a matrix.

Q Write a prog. to modify the main diag. elements of a 3x3 matrix given as

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

by the values entered by the user.

diag(a) ← c(x,y,z)

Page No.
Date

4) WAP to find ~~whether~~ whether a no. is prime or not.

5) WAP to find factorial of no.

6) " " print fibonacci series upto 'n'.

Armstrong no. ⇒ digits sum = square.

12th feb '19

Page No.
Date

```
f <- function (n){
    s <- 0
    m <- n
    while (n > 0) {
        r = n %% 10                  # returns modulus
        s = s + (r * r * r)
        n = n %/% 10                 # integer div
    }
print (m)
print (s)
if (m == s){
    print ("a")
}else{
    print ("b")
}
}
n <- as.integer (
```

* by default whatever is the last variable used in the funct" (last statement) is returned ~~that~~ irrespective of 'return' statement

O/P: for n =

Page No.
Date

Arrays in R

3 x 4 x 2
2 arrays of 3 x 4.

Syntax:

array(data, dim = c(r, c, m), dimnames = list(rn, cn, mn))

(data: type can be anything; r: row; c: col; m: matrix)

eg: a ← array(1:24, dim = c(3,4,2))
∴ 3 x 4 x 2 = 24

eg
u1 ← 1:12
u2 ← 13:24
b ← array(c(u1, u2), dim = c(3,4,2)) # may have used cbind(), rbind()

1st array →

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

→ 2nd array

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

Matrices using Array()

> i ← array(c(1:3,3:1), dim = c(3,2))
> row_names ← c("r1", "r2", "r3")
> col_names ← c("c1", "c2", "c3")
> row_names ← c("m1", "m2")
> b ← array(1:24, dim = c(3,4,2), dimnames = list(row_names, col_names, m_names))

name(matrix)
attribute ( )
mode ( )
} checks.

Page No.
Date

* IsNotMatrix( ) → tells if anything is matrix or array

a[1,2,1] # 4
↳ 1st row & 2nd col of 1st matrix.

a[,2,1] # 456
a[, ,] # entire array.

a[c(1), c(2,3), 2] # 13 16
1st row, 2nd & 3rd col, ↳ 2nd mat

a[c(TRUE, FALSE), c(2,3), 2] # 13, 16, 15, 19.
Recycling so 1st & 3rd row

a[-1, , 1] X.
a[-1, ,]

functions available,
sum(a)
nrow(a)
ncol(a)
] check?

a(, 2, 1)

~~if x=~~

⇒ Index Matrices.
using another array as index to an array

x = [1 4 7; 2 5 8; 3 6 9]

i = [2 1; 1 2; 3 3]

x[i] = ~~8~~ 5 4 9
i[x] // not possible

Negative indexes are not allowed!

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Page No.
Date

→ Lists Heterogeneous 1-D combinat^n of data.

~~Vector~~ v ← c(1,2,c(1,2)) ≠ v ← c(1,2,1,2) ✗

* Vector homogeneous & list heterogeneous
* list can have nested list but nested vectors are not available in R.

```
> list_1 ← list("Red", "Green", c(21,31,11), TRUE, 51.23)
> list_1
[[1]]                          # 1st element of list
  [1] "Red"                    # 1 O/P is Red.
[[2]]
  [1] "Green"
[[3]]
  [1] 21 32 11
```

```
>v
[1] 1,2,.... 20
[21] 21,22,.... 40
```
→ screen size can have 20 elements in 1 row

[1] → element corresponding to the 1st element in row.

[ ] → used to select element of i^{th} component
[[ ]] → " " " components of list.

```
> list_1[1]
[[1]]
  [1] "Red"

> list_1[[1]]
[1] "Red"
```

Page No.
Date

→ Adding Elements to a list

'null' instead of 'NA'

~~> list_1[[0]] = c(1,1)~~

```
list_1[[0]] = c(1,1)
  [1] 0
> list_1
[[1]]
  :
  :
  :
[[7]]
    NULL
[[8]]
  [1]  1  1
```

→ Naming Elements in List

Two way

```
> list_data ← list(c("A","b"), matrix(c(3,9,2),
                   nrow=2), list("green",12.3))
> names(list_data) ← c("1st Quater", "matrix", "List")
> list_data
$`1st Quarter`
[1] "A"  "B"

$Matrix
  [,1]

$List
$List[[1]]
[1] "green"
```

Page No.
Date

$'list' [[2]]
[1] 12.3

$'list' [[3]]
[1]

2nd way

x ← list(day="Tuesday", lecture="DS", date=c(13,2,2018))
>x
$day
[1] "Tuesday"
$lecture
[1] "DS"

$date
[1] 13 2 2018

>x$day
[1] "Tuesday"
>x$date
[1] 13 2 2018
>x$date[1]
13

⇒ Merging lists

>is.list(x) # checks if x is or not a list
[1] TRUE
>y=c(1)
>y=as.list(y)
>new_list=c(x,y) # changes vector into list
x elements comes first & then y comes.

5 + 10 : 15

Page No.
Date

c() is also used to combine multiple list.

* no arithmetic operators on list (like +, -, *, %, %

> list1 + list2 #Error
> v1 ← unlist(list1)
> v2 ← unlist(list2)
> v1 + v2

11th March in AI reg.

11th March'19

Factors (Data Science)

factor(data_vector) or factor(data_vector, level_vector)

```
> data <- c("east", "west", "east", "north")
> is.factor(data)
[1] FALSE
> f1 <- factor(data); f1
[1] east west east north
Levels: east north west
```

levels are created alphabetically.

or

```
> f1 <- factor(data, levels = c("north", "east", "west"))
> class(data)
numeric
> class(f1)
factor
```

functions

1) > levels(f1) : returns level corr. to factor f1.
   O/P : "east" "north" "west"

> nlevels(f1) : returns no. of levels.
[1] 3

Factors

- ordered (sorting is w.r.t to levels) — Mandatory to specify levels.
- unordered (by default)

if unordered.
> max(f1) or min(f1)
↓
gives max level of f1
[1] min is not meaning full for factors.

> is.ordered(f1) # tells if ordered or not
[1] FALSE

> f2 ← factor(data, levels = c("east", "north", "west"), ordered = TRUE)
no order

> f2
[1]
levels: east < north < west
> max(f2)
[1] west

> str(f1)
Factor w/3 levels # structure of factor f1
east → 1
west → 3
east → 2
north
west
1 3 1 2 3
↓ ↓
level of 1st element, level of 2nd elem.

> f1[3]
east
> f1[c(2,4)]
west north
> f1[c(TRUE, False)]
east east west
Recycling.

> f1[-1]

> length(f1)
[1] 5
> f1[6] ← "south"
valid in vector but not here as only 3 levels are available so we can't insert any value w.r.t new level

[1] invalid factor level, NA generated.

```
> f1
[1] ....  NA
```

⇒ Adding levels to existing factors.

```
> levels(f1) ← c(levels(f1), "south")
```

old levels: east north west.

```
> nlevels(f1)
[1] 4

> f1[6] ← "south" ; f1
[1] .....  south
```

⇒ Major applicat" of factors (Interpretat" of dataframes w.r.t visualizat")

⇒ tapply(): this funct" is used to run a predefine function or a user defined funct" on a factor.

factor.

empid	region	income
	e	20k
	s	30k
	n	
	w	

income w.r.t each regio

acc. to factors level mean is calculated & retu

```
> region ← c("e","w","e","n","s","e","n","s")
> r1 ← factor(region)
> income ← c(20,40,60,80,30,30,60,40)
> mean_income ← tapply(income, r1, mean)
```

mean can be user defi funct" al

funct" to run

data is dependent on what factor

values of data

> mean_income

e n s w → avg value for each level.

Q. Differentiate b/w ordered / unordered factors?

Ques >items <- data.frame(color = c('red', 'blue', 'blue', 'red', 'green', 'green', 'yellow', 'blue', 'green', 'yellow', 'red', 'blue', 'blue', 'green', 'red', 'yellow'),
size = c(5, 10, 11, 6, 15, 16, 20, 9, 13, 18, 7, 14, 8, 13, 6, 18))

items → | colors (factors) | size |

> tapply(items$size, items$color, sum)

blue	green	red	yellow
52	57	24	56

> tapply(items$size, items$color, mean)

27th March '19

Ch-10

> data()
> data(iris)
> install.packages.....
> str(iris)
> summary(iris)
> head(iris)
> head(iris, n=10)
> tail(iris)

list of whatever datasets are inbuilt
for particular dataset in current environment
name of dataset
structure of dataset
1st 5 records/rows
10 " "
last 5 rows.

In practical it runs 6 but in theory 5 rows.

viva

airquality, pressure

> max(pressure $temperature)
[1] 360.

Quartile, decile, percentile
4 10 100

> quantile(pressure $temperature)

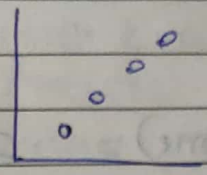
> summary gives min, 1st quartile, median, mean, 3rd quar. max.
summary(pressure $temperature)

> nrow(iris)
> ncol(iris)
> dim(iris)

no. of rows
" " cols

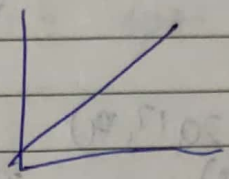
Page No.
Date

→ Graphical (Ch-10)
> x ← 1:10
> plot(x)

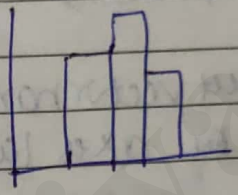


scattered plot grap

line graph



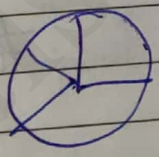
~~Bar plo~~ Bar plot



Boxplot

Histogram

piechart



1) Scattered plot → (intervals can't be specified)
plot(x, [y, main, xlab, ylab, xlim, ylim, axes])
xlab - "xlabel" x → mandatory para
ylab → y "
xlim → x limits ⇒ xlim=c(0,50)
ylim → y limits
axes = "true" → to draw axes (default)
main ⇒ "title of graph"

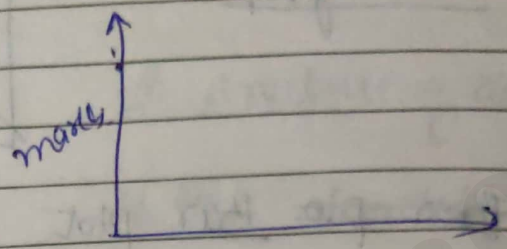
speed
xlabel

Date

```
> x <- 1:10
> plot(x)

> plot(x, y)
> plot(dataframe)        #(graph b/w 1st 2 col by default)

> test <- 1:5
> marks <- c(23,21,29,17,19)
> plot(test, marks)
```



by default vector name is set as axes label

```
> test
> marks
> df = dataframe(test, marks)
> df
> plot(df)               #always 1st 2 cols

> plot(df$c, df$d)       # col c & col d are plotted
```

} plot using data frames.

```
> data(mtcars)           #load current env. with mtcars dataset
> #head(mtcars)
> plot(mtcars[,1], mtcars[,2])     (1st col & 2nd col)
   or
   plot(mtcars$mpg, mtcars$cyl)
```

read.csv ("path of file")
read.csv ("filename")

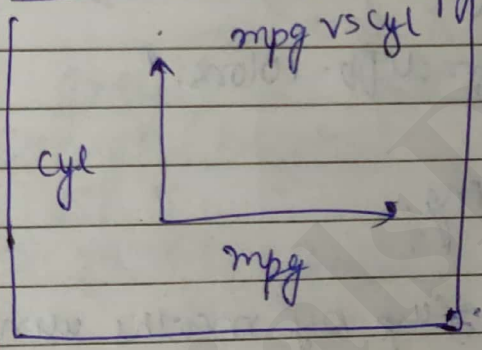
Page No.
Date

If want to use > plot (mpg, cyl)
R will look for mpg but dataset is not known so error.
but > attach (mtcars)
this attaches the mtcars dataset to the search path.

```
>attach (mtcars)
> plot (mpg, cyl)
```

→ adding titles (labels)

```
>plot(mtcars[,1], mtcars[,2], main="mpg vs cyl",
      xlab="mpg", ylab="cyl")
```



Note :> plot(x,y) is same as plot (y~x)

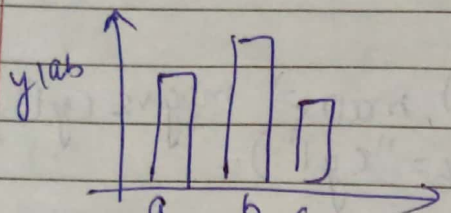
```
df<-read.csv("C:\\Users\\ved\\Desktop\\data.csv")
> str(df)
> nrow(df)
> ncol(df)
> dim(df)
```

Page No.
Date

2) Barcharts

barplot (H, xlab, ylab, main, names.arg, col)

H → vector or matrix.
xlab, ylab → label of x & y-axis.
main → title
names.arg = is a vector of names appearing under each bar
col → is used to give colors to bars in graph.



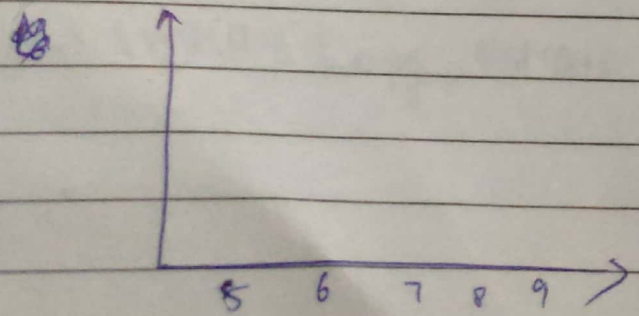
names.arg = c("a", b, c)
col = c("red", - -)
↑
for diff. colors.

→ ~~color~~ (col) has recycling.

Ques plot barchart of mean temp by months using airquality.

```
> data(airquality)
> head(" ")
```

	Month	Temp	day
1	5	30	1
2	6	27	2
3	7		
4	8		
5	9		
6	5		
	6		
	⋮		



Page No.
Date

tapply → apply funct^n on complete dataset
tapply (vector, factor to divide, mean)

heights ← tapply (airquality $ Temp, airquality $ Month, mean)
vector factor funct^n to be applied.

>barplot (heights, main="mean temp by month",
names.arg = c("May", "June", "July", "Aug", "Sep"),
ylab="Temp (deg. F)")

Que Shade bars acc. to rank.
shorter → light, taller → darker.
→ range is [0-1]
gray() is used → to find color value

→ >length(heights)
[1] 5

→ > rank(heights) # give ranking (lowest to highest)
5 6 7 8 9
1 3 4 5 2
(lowest value) (highest value)

→ > relhts ← rank(heights)/length(heights)
> grays ← gray(1-relhts)
color value acc. to rank.
>barplot(heights, col=gray, ...)

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

2018 (5 marks)

Page No.
Date

Q Write a code in R to plot a graph to depicts the relat" b/w temp ~~b/w~~ in C° (on x-axis) and in F (on y-axis) using the formula

$$T(F) = T(C) * \frac{9}{5} + 32$$

Label the axis and give title as Celsius Vs Farh. to graph.

Soln generate code to take values for °C & use formula for 'F' or vicevesa.

Ques Store the following data in a dataframe & perform the following.

A	B
alpha	100
Beta	120
gamma	80
Delta	110

(i) Display the contents of dataframe.
(ii) Draw barplot and name the bars according to the values of col A.

28th March 19

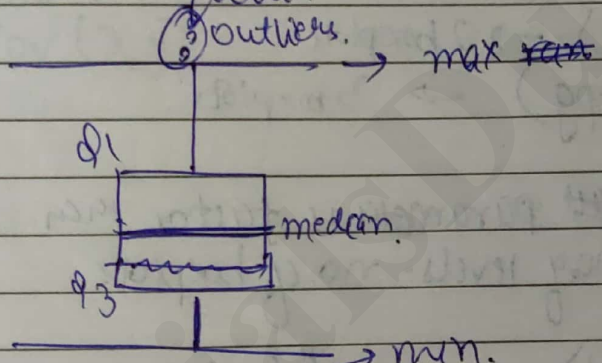
Page No.
Date

⇒ Plotting a line from x & y points.

> plot (x, y, type="l")
because for plot & not scattered.

> plot (df, type="l")

⇒ Boxplot
→ mean
→ Standard deviat" $\sum (x-\bar{x})^2$
→ max, min
→ outliers ✗ remove
→ quartile.



Range of data

boxplot (x, data, notch, varwidth, names, main)

x → vector or a formula (from where we want to plot data)
data → dataframe (factor) (categorical data)
notch → logical value. set TRUE to draw a notch.
varwidth: logical value. Set as true to draw width of box proportional to sample space.
names:
main:

Page No.
Date

eg.

Sid	marks	test
1	30	1
2	20	2
3	25	3
1	20	2
1	30	3

↳ factor.

boxplot(df$marks, df$test)

input <- mtcars[, c('mpg', 'cyl')]
↳ new dataframe with 1st & 2nd col

boxplot(mpg, cyl) → 2 boxplots
** boxplot(cyl, mpg) → 3 boxplots.
↳ If 1st parameter is factor then no. of levels = no. of boxplots.

table(input$cyl) # freq. of cyl is given
o/p 4 6 8
11 7 14
11 + 7 + 14 = 32 → no. of rows

⇒ Histogram → depict freq. corr. to something.

hist(v, main, xlab, xlim, ylim, breaks, col, border)
v – vector whose freq. is to be plotted
main –
breaks – width of each bar.
col → colors for bars
border → used to set border color for each bar
border = c('black', ...)

Page No.
Date

Histogram is ~~not~~ reversed for more than 1 attribute.

⇒ names(df) # name of every col.

* Boxplot variable 'final' w.r.t grade
means final pehle & grade baadmein

Sect^n 2.6 → to be covered. (Result of foll. funct^n with diff data structures.)

→ mean(x)
→ median(x)
→ sd(x) # standard deviat^n
→ var(x)
→ cor(x,y) # correlat^n
→ cov(x,y) # covariance

Unit 5
Section 2.6:

Ques: i) data given → write syntax for variance, etc.
2) mean, etc used; asking for o/p.

Ques1:
```
> x ← c(0, 1, 1, 2, 3, 5, 8, 13, 21, 34)
> mean(x)
> median(x)
> var(x)
> sd(x)  #Ans
```
— 1)

i) $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$ n: length of x

ii) median $\begin{cases} \text{odd} & \frac{\left[\frac{n+1}{2}\right] + \frac{n}{2}}{2} \\ \text{even} & \frac{n}{2} \end{cases}$

iii) var → $\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$

iv) sd → $\sqrt{\text{var}(x)}$

* In case of NA if m. NA not used then ans is NA; if used, (NA) term not calculated in total.

mean(x) → 8.8

median → 4

iii) $Var(x) = \frac{1}{10-1} \sum_{i=1}^{10} (x_i - \bar{x})^2$

$= \frac{1}{9}[(0-8.8)^2 + (1-8.8)^2 + (1-8.8)^2 + (2-8.8)^2 + (3-8.8)^2 + (5-8.8)^2 + (8-8.8)^2 + (13-8.8)^2 + (21-8.8)^2 + (34-8.8)^2]$

$= 121.73.$

$sd = 11.03$ Ans

Ques: Consider the vector x defined as follows and right the o/p of the following questions:

x ← seq(1, 10, by=2)

x = 1, 3, 5, 7, 9

i) find mean x & median x

iii) > x[6] ← 9
> x[7] ← 4
> x[c(8,9,10)] ← 2
> print(length(x))
~~final~~
~~consider~~ print length of ~~of~~ x & the final mode of x.
↳ has no predefined function.

iii) find st. dev. & variance of x.

iv) x[12] ← NA
→ now, find mean(x)
→ Also state the effect of the introduction of the NA value on this vector x.
→ Rectify the syntax so as to cal. the actual mean of vector x.

Soln: i) mean - 5
median - 5

ii) 1, 3, 5, 7, 9, 2, 4, 2, 2, 2.
~~length(x)~~ → 10
finding mode of (x)
where, x = c(1, 3, 5, 7, 9, 2, 4, 2, 2, 2)

(2) ans ← y[which.max(tabulate(match(x,y)))]

(1) y ← unique(x); y: o/p: [1] 1 2 3 4 5 7 9

unique(x): This func returns a row vector that contains the sorted set of unique values of its argument.

* some versions of R return a set of unique values w/o sorting it.

match(x,y): This func returns the first occurence of the first argument in the second arg.

tabulate(): takes the integer valued vector and counts the no. of times each integer occurs in it.

iii) var → 1/9 (← [which.(.)] : → returns the index of the logical ~~func~~ object when it is true.

var → 6.67

sd → 2.58

iv) mean(x) → NA

mean can't be cal.

mean(x, m. NA = True)

1, 3, 5, 7, 9, 2, 4, 2, 2, 2, NA, NA

length → 12

* unique(x): * match: 1 2 3 4 5 6 7 6 6 6

1 3 5 7 9 2 4

(1) (2) (3) (4) [5] [6] [7]

tabulate(x)

≈ tabulate(1,2,3,4,5,6,7,6,6,6)

O/P: 1 1 1 1 1 4 1

↓

frequency of each no.

max(tabulate(i →)) → (4) ✓

ie, > ans ← y[which.4].

v)

CORRELATION & covariance

> y ← rep(1+n, each=2)

i) find correlation (x,y) and Covariance (x,y)

vi) remove NA values from x & create a vector x1 from x with values < 4.

vii) y[x<=1:6]

now find cor(x1,y1) & cov(x1,y1)

$$cov(x,y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

$$cor(x,y) = \frac{cov(x,y)}{var(x) \cdot var(y)}$$

where, x & y have to be vectors of same dimensions.

Do!!: v) x → 1 3 5 7 9 2 4 2 2 2 NA NA

y ← 2 2 4 4 6 6 8 8 10 10 33 55 33 33 33 NA NA NA NA

i) has a mismatch in dimensions.

vi)
```
> x1 ← x[1:12] < 4
> x1 ← x1[!is.na]
```

```
> x ← x[ ! is.na(x) ]   // removes NA values.
> x
 [1] 1 3 5 7 9 2 4 2 2 2.

> x1 ← x [x<4]
> x1
 [1] 1 3 2 2 2
```

vii) > y1 ← c(1,2,3,4,5,6)

> cov(x1, y1)
> cor(x1, y1)

Values:

$\bar{x} = 2 \; ; \bar{y} = 3.5$

$$cov(x,y) = \frac{(1-2)(1-3.5)}{5} + \frac{(3-2)(2-3.5)}{5}$$

$$+ \frac{(2-2)(3-3.5)}{5} + (0+0+0$$

$$= 0.5 + (-0.3)$$

$$= 0.2.$$

$$var(x) = \frac{1}{5}\left[(1-2)^2 + (3-2)^2 + 0^2\right]$$

$$= \frac{1}{5}\left[1^2 + 1^2\right] = \frac{2}{5} = 0.4$$

$$var(y) = \frac{1}{5}\left[(1-3.5)^2 + (2-3.5)^2 + (3-3.5)^2 + (4-3.5)^2 + (5-3.5)^2 + (6-3.5)^2\right]$$

$$= 3.5$$

$$cor = \frac{0.2}{0.4 \times 3.5} = 0.1493. = 0.169$$

Dataframes:

1) mean and standard deviation of a df works by column and these functions will only work for the numerical cols of a dfname.

eg: %df

small	medium	large
2	5	9
3	6	10
4	7	8

1) %mean(df)

small — mean value

medium — m.v.

large — m.v.

2) %ed(df)

small

medium

large

) median function doesn't apply directly for a df; needs to be applied individually.

1) median (df$small)
median (df$medium)
median (df$large)

or using tapply() or lapply()
[to apply through the df
→ med by mode also.

2) variance & covariance returns a covariance matrix for the dataframe.

> var(df)
> cov(df)

o/p:

	small	medium	large
small	smallvar small	mediumvars	largevars
medium		mvar	
large			lvar

→ covariance matrix

4) cor(df) - same.

Pg 22 - 2.6: apply family functions.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well