reference

R for Data Science https://www.youtube.com/watch?v=NVyOEwOJgNQ&t=11820s

Open this link in your browser to create a new notebook with R Kernel

https://colab.research.google.com/notebook#create=true&language=r

basics

math operations

```
x<-18
print(x)
y<-4
z<-x+y
print(z)
z
```

```
[1] 18
[1] 22
22
```

```
18+4
18-4
10/2
4.5/2
```

```
22
14
5
2.25
```

```
#exp
2**3
2^3
```

```
8
8
```

```
9**0.5
```

```
3
```

```
9**1/2
#division operator has higher precision than exp=>first 9**1 then div by 2
```

```
4.5
```

```
9**(1/2)
```

```
3
```

```
14%%4 #modulus
14.75 %% 4
```

```
2
2.75
```

```
8 / 3 #division
8 %/% 3 #integer division
```

```
2.66666666666667
2
```

```
-8 /3
-8 %/% 3
#integer division give floor value
```

```
-2.66666666666667
-3
```

```
TRUE+TRUE
FALSE+TRUE
```

```
2
1
```

VARIABLES

```
my_var=10
print(my_var)
my_var=20
typeof(my_var)
print(my_var)
```

```
[1] 10
'double'
[1] 20
```

There are the following keywords as per ?reserved or help(reserved) command:

if else repeat while function for next break TRUE FALSE NULL Inf NaN NA NA_integer_ NA_real_ NA_complex_ NA_character_

```
if    else    repeat
while    function    for
next    break    TRUE
FALSE    NULL    Inf
NaN    NA    NA_integer_
NA_real_    NA_complex_    NA_character_
```

data types

```
x<-10.25
x
class(x)
```

```
10.25
'numeric'
```

Logical,Numeric,integer,complex,character,raw are the data types in R

```
is.integer(x)
class(x)
```

```
FALSE
'numeric'
```

```
x<-as.integer(10.25)
is.integer(x)
class(x)
```

```
TRUE
'integer'
```

```
y<- 3 + 4i
class(y)
y
```

```
'complex'
3+4i
```

```
z<-TRUE
class(z)
z
```

```
z<-F
class(z)
z
```

```
'logical'
TRUE
'logical'
FALSE
```

variable assignment <- = assign

```
x<-10
assign('y',10)
x
y
z=100
z
```

```
10
10
100
```

```
a<-b<-c<-66
a
```

```
66
```

```
l<-letters
L<-LETTERS
print(l)
L
```

```
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"
'A' · 'B' · 'C' · 'D' · 'E' · 'F' · 'G' · 'H' · 'I' · 'J' · 'K' · 'L' · 'M' · 'N' · 'O' · 'P' · 'Q' · 'R' · 'S' · 'T' · 'U' · 'V' · 'W' · 'X' ·
'Y' · 'Z'
```

relational operators

`> < == <= >= !=`

logical

& &&

| ||

! not

```
TRUE | FALSE
```

```
TRUE
```

```
10 & 0 #0 is treated as false
# 1 treated as true
!10
!0
```

```
FALSE
FALSE
TRUE
```

sequences

```
x<- 1:10
x
```

```
1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10
```

```
x<- 1:10 -1
x
```

```
0 · 1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9
```

```
x<-10
1:x
1:x-1 # first 1:x then subtract 1 from vector x
1:(x-1)
```

```
1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10
0 · 1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9
1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9
```

```
y<-seq(6)
y
y<-seq(1,14)
y
yrev<-seq(from=10,to=1)
yrev
```

```
1 · 2 · 3 · 4 · 5 · 6
1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10 · 11 · 12 · 13 · 14
10 · 9 · 8 · 7 · 6 · 5 · 4 · 3 · 2 · 1
```

```
z<-seq(1,10,by=2)
print(z)
z<-seq(1,10,length=4)
print(z)
```

```
     [1] 1 3 5 7 9
     [1]  1  4  7 10
```

```
s<-1:17
print(s)
s<-seq(1,10,2)#for odd
print(s)
s<-seq(0,10,2)#for even numbers
print(s)
s<-seq(10,0,-2)# for reverse
print(s)
```

```
      [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
     [1] 1 3 5 7 9
     [1]  0  2  4  6  8 10
     [1] 10  8  6  4  2  0
```

```
#replication
x<-rep(1,times=5)
print(x)
z<-rep('a',times=3)
print(z)
```

```
     [1] 1 1 1 1 1
     [1] "a" "a" "a"
```

```
a<-1:3
print(a)
b<-rep(a,each=5)
print(b)
```

```
     [1] 1 2 3
      [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
```

if else

if (condition){

}

```
x<-2
if (x > 0){
  print("positive")
}else if(x ==0 ){
print("it is zero")
} else {
  print("negative")
}
```

```
[1] "positive"
```

```
x<-6
ifelse(x%%2 ==0,'even number','odd number')
```

'even number'

loops

```
for(i in 1:5){
  print(i)
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

```
for(i in 1:5)  print(i)
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

```
x<-letters
y<-x[1:5]
#for (i in x) print(i)
for (i in y) print(i)
```

```
[1] "a"
[1] "b"
[1] "c"
[1] "d"
[1] "e"
```

```
i<-1
while(i<=5){
  print(i)
  i<-i+1
}
```

```
[1] 1
[1] 2
[1] 3
```

```
    [1] 4
    [1] 5
```

repeat loop

```
i<-1
repeat{
  print(i)
  if(i>5)
   break
  i<-i+1
}
```

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

```
for(i in 1:10){
  if(i%/% 2==0)
  next
  print(i)
}
```

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

Functions

```
sum=function(a,b){
  c=a+b
  return(c)
}
```

```
sum(6,-1)
```

```
    5
```

```
myeval=function(x,y){
  w=x+y
  z=x+y
  result=list('sum'=w,'mul'=z)
  return(result)
}
```

```
myeval(10,20)
```

```
$sum
     30
$mul
     30
```

lamda function in r

```
mysum=function(x,y) x+y
```

```
mysum(1,2)
```

```
3
```

vectors

vector has elements of same datatype

c funtion for concatination ,creation of vector

```
ve<-c(10,20,30)
ve
```

```
10 · 20 · 30
```

```
assign('y',c(50,40,60))
print(y)
```

```
50 · 40 · 60
```

```
lo<-c(T,F,T)
print(lo)
print(10)
```

```
     [1]  TRUE FALSE  TRUE
```

```
e<-c(10,20,'a',3.5)
print(e)#because of vector coercion
typeof(e)
length(e)
```

```
     [1] "10"  "20"  "a"   "3.5"
     'character'
     4
```

using 1:10 ,seq(1,10) we can create a vector

rep (10,3) create a vector 10 10 10

```
em<-vector()
print(em)
```

```
     logical(0)
```

```
z<-vector('numeric',length=4)
z
```

```
     0 · 0 · 0 · 0
```

indexing

```
x<-c(67,55,105,10,40,35,40,80)
x
x[-2]
x[2:5]
x[c(1,2,4,6)]
```

```
     67 · 55 · 105 · 10 · 40 · 35 · 40 · 80
     67 · 105 · 10 · 40 · 35 · 40 · 80
     55 · 105 · 10 · 40
     67 · 55 · 10 · 35
```

```
x[2]<-8
x
#to replace/overwrite with 8
```

```
     67 · 8 · 105 · 10 · 40 · 35 · 40 · 80
```

```
x[-3]<-0
x
```

```
x
#for -ve index except that all the other elements are replaced
```

```
0 · 0 · 105 · 0 · 0 · 0 · 0 · 0
```

```
x<-c(10,20,30,40,50,60)
y<-c(T,F,T,T,T,F,T)
x[y]#
```

```
10 · 30 · 40 · 50 · <NA>
```

```
x<-c(10,20,30,40,50,60)
y<-c(T,F,T)
x[y]#as the length of y is less than x . y vector is extended in round robin fashion
```

```
10 · 30 · 40 · 60
```

```
#matching operator %in%
35 %in% x
30 %in% x
```

```
FALSE
TRUE
```

arithematic operations

```
x<-c(10,20,30,100)
x
sqrt(x)
y<-x-10
y
```

```
10 · 20 · 30 · 100
3.16227766016838 · 4.47213595499958 · 5.47722557505166 · 10
0 · 10 · 20 · 90
```

```
rev(y)#reverse
sort(y)
x %*% y
```

```
90 · 20 · 10 · 0
0 · 10 · 20 · 90
```

```
crossprod(x,y)
```

```
A
matrix:
1 × 1
of type
dbl

9800
```

```
x %o% y
```

```
A matrix: 4 × 4 of type dbl
0   100   200   900
0   200   400  1800
0   300   600  2700
0  1000  2000  9000
```

```
out<- x> 10 & x<50
out
x[out]
```

```
FALSE · TRUE · TRUE · FALSE
20 · 30
```

```
index<-which(x>20)
index
x[index]
```

```
3 · 4
30 · 100
```

factor

```
x<-factor(c("male","female","male","female","male","male","female","male"))
x
table(x)
```

```
male · female · male · female · male · male · female · male
▶ Levels:
x
female   male 
     3      5 
```

Mathematical operations

```
x<-c(3.66,6.78,9.101,-1.01)
abs(x)
ceiling(x)
floor(x)
round(x)
round(x,1)
```

```
3.66 · 6.78 · 9.101 · 1.01
4 · 7 · 10 · -1
3 · 6 · 9 · -2
4 · 7 · 9 · -1
3.7 · 6.8 · 9.1 · -1
3 · 6 · 9 · -1
```

```
x<-c(3.66,6.78,9.101,-1.01)
trunc(x)
```

```
3 · 6 · 9 · -1
```

```
x<-c(3.66,6.78,9,10)
sqrt(x)
exp(x)
log(x)
log(x,base=10)
log10(x)
```

```
1.9131126469709 · 2.60384331325831 · 3 · 3.16227766016838
38.8613428713325 · 880.068724107803 · 8103.08392757538 · 22026.4657948067
1.29746314741327 · 1.9139771019523 · 2.19722457733622 · 2.30258509299405
0.563481085394411 · 0.831229693867063 · 0.954242509439325 · 1
0.563481085394411 · 0.831229693867063 · 0.954242509439325 · 1
```

```
factorial(5)
```

```
120
```

random numbers

```
r<-rnorm(10)
r#mean=0,std=1
```

```
2.16087763448505 · 0.993075856579701 · 0.914742617223747 · 0.274813998145719 ·
-1.87423114211433 · -0.809165800446437 · -1.47029147333903 · -0.709557653044569 ·
-0.91106890927378 · 1.65320141439457
```

```
y<-rnorm(10,mean=2,sd=1)
y
```

```
2.6237787437403 · 1.18071668336848 · 2.73579632028604 · 2.65023878341606 · 3.1137720470878 ·
0.481766938087126 · 3.16480599941736 · 2.06151221582948 · 1.15524055496118 · 1.8859552824791
```

matrix

matrix are created in columnwise

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

```
A matrix: 2 × 3
  of type lgl
 NA   NA   NA

 NA   NA   NA
2 · 3
```

```
m<-matrix(c(1,2,3,4,5,6))
print(m)
```

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

```
m<-matrix(c(1,2,3,4,5,6),nrow=2,ncol=3)
print(m)
```

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

by row is used for row wise

```
m<-matrix(c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=T)
print(m)
```

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

```
m<-matrix(1:12,nrow=4,ncol=3,byrow=T)
print(m)
```

```
length(m)
dim(m)
nrow(m)
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
12
4 · 3
4
```

matrix diag()

```
d<-matrix(1,3,3)
print(d)
```

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

```
m<-diag(1:5)
print(m)
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    2    0    0    0
[3,]    0    0    3    0    0
[4,]    0    0    0    4    0
[5,]    0    0    0    0    5
```

```
rownames(m)<-c(10,20,30,40,50)
colnames(m)<-c("A","B","C","D","E")
m
```

A matrix: 5 × 5 of type int

	A	B	C	D	E
10	1	0	0	0	0
20	0	2	0	0	0
30	0	0	3	0	0
40	0	0	0	4	0
50	0	0	0	0	5

matrix indexing

```
m[3,]
m
```

```
A:    0 B:    0 C:    3 D:    0 E:    0
```

```
A<-matrix(c(2,3,4,0,5,7,5,8,2,1,-1,-2),nrow = 4,ncol = 3,byrow = T)
print(A)
A[3,2:3]=80
A
print(A)
```

```
     [,1] [,2] [,3]
[1,]    2    3    4
[2,]    0    5    7
[3,]    5    8    2
[4,]    1   -1   -2
A matrix: 4 ×
3 of type dbl
 2   3   4

 0   5   7

 5  80  80

 1  -1  -2
     [,1] [,2] [,3]
[1,]    2    3    4
[2,]    0    5    7
[3,]    5   80   80
[4,]    1   -1   -2
```

```
diag(A)
print(diag(A))
```

```
2 · 5 · 80
[1]  2  5 80
```

```
b<-rbind(A,c(10,11,12))
print(b)
```

```
     [,1] [,2] [,3]
[1,]    2    3    4
[2,]    0    5    7
[3,]    5   80   80
[4,]    1   -1   -2
[5,]   10   11   12
```

```
c<-rbind(A,b)
print(c)
```

```
      [,1] [,2] [,3]
 [1,]    2    3    4
 [2,]    0    5    7
 [3,]    5   80   80
 [4,]    1   -1   -2
 [5,]    2    3    4
 [6,]    0    5    7
 [7,]    5   80   80
 [8,]    1   -1   -2
 [9,]   10   11   12
```

```
D<-cbind(A,c(10,11,12))
print(D)
```

```
Warning message in cbind(A, c(10, 11, 12)):
“number of rows of result is not a multiple of vector length (arg 2)”
     [,1] [,2] [,3] [,4]
[1,]    2    3    4   10
[2,]    0    5    7   11
[3,]    5   80   80   12
[4,]    1   -1   -2   10
```

for rbind,cbind dimensions has to be match

matrix operations

```
A<-matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3,ncol = 3,byrow = T)
print(A)
B<-matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3,ncol = 3)
print(B)
```

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

```
C=A+B
print(C)
```

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    6   10   14
[3,]   10   14   18
```

```
print(A-B)
print(A*B) #element wise multplication
print(A*B) #element wise multplication
print(A %*%B ) #matrix multplication
print(A/B)
```

```
     [,1] [,2] [,3]
[1,]    0   -2   -4
[2,]    2    0   -2
[3,]    4    2    0
     [,1] [,2] [,3]
[1,]    1    8   21
[2,]    8   25   48
[3,]   21   48   81
     [,1] [,2] [,3]
[1,]   14   32   50
[2,]   32   77  122
[3,]   50  122  194
         [,1]     [,2]      [,3]
[1,] 1.000000 0.500000 0.4285714
[2,] 2.000000 1.000000 0.7500000
[3,] 2.333333 1.333333 1.0000000
```

```
t(A)#transverse of A
```

A matrix: 3 × 3 of type dbl

1	4	7
2	5	8
3	6	9

https://cran.r-project.org/web/packages/matlib/vignettes/inv-ex1.html refer this for more math operations like det

```
print(A)
rowSums(A)
colSums(A)
```

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

6 · 15 · 24

12 · 15 · 18

```
rowMeans(A)
```

2 · 5 · 8

```
colMeans(A)
```

```
     4 · 5 · 6
```

```
apply(A,1,sum)#1 for row
apply(A,2,sum) #2 for col
```

```
     6 · 15 · 24
     12 · 15 · 18
```

LIST

```
x<-list(1,"karthik",105.67)
x
```

```
        1. 1
        2. 'karthik'
        3. 105.67
```

```
rollno<-c(101,102,103,104,105)
snames<-c('john','bob','alice')
marks<-c(78.25,100,90.2)
student<-list(rollno,snames,marks)
student
print(student)
```

```
        1. 101 · 102 · 103 · 104 · 105
        2. 'john' · 'bob' · 'alice'
        3. 78.25 · 100 · 90.2

     [[1]]
     [1] 101 102 103 104 105

     [[2]]
     [1] "john"  "bob"   "alice"

     [[3]]
     [1]  78.25 100.00  90.20
```

the above is example for list with different datatypes,sizes

```
print(student[1])
print(student[[1]])
```

```
     [[1]]
     [1] 101 102 103 104 105
```

```
[1] 101 102 103 104 105
```

```
# $
student<-list('id'=rollno,'name'=snames,'scores'=marks)
print(student$scores)
print(student[c('id')])
print("----- ----- -----")
print(student[1:3])
```

```
[1]  78.25 100.00  90.20
$id
[1] 101 102 103 104 105

[1] "----- ----- -----"
$id
[1] 101 102 103 104 105

$name
[1] "john"  "bob"   "alice"

$scores
[1]  78.25 100.00  90.20
```

list concat

```
rollno<-c(101,102,103,104,105)
snames<-c('john','bob','alice')
marks<-c(78.25,100,90.2)
student<-list(rollno,snames,marks)
age<-list(c(19,20,18))
students<-c(student,age)
print(students)
```

```
[[1]]
[1] 101 102 103 104 105

[[2]]
[1] "john"  "bob"   "alice"

[[3]]
[1]  78.25 100.00  90.20

[[4]]
[1] 19 20 18
```

```
print(student)#this is the list created before
```

```
[[1]]
[1] 101 102 103 104 105

[[2]]
[1] "john"  "bob"   "alice"

[[3]]
[1]  78.25 100.00  90.20
```

DATAFRAMES

```
id<-c(101,102,103)
names<-c('john','bob','alice')
marks<-c(78.25,100,90.2)
stu<-data.frame(id,names,marks)
stu
```

A data.frame: 3 × 3

id	names	marks
<dbl>	<chr>	<dbl>
101	john	78.25
102	bob	100.00
103	alice	90.20

```
print(stu[2,])
print(stu[2:3,])
```

```
   id names marks
2 102   bob   100
   id names marks
2 102   bob 100.0
3 103 alice  90.2
```

```
print(stu[,1])
```

```
[1] 101 102 103
```

```
print(stu[2:3,1:3])
```

```
   id names marks
2 102   bob 100.0
3 103 alice  90.2
```

```
print(stu[-2,-3])#to remove we use -
```

```
   id names
1 101  john
3 103 alice
```

```
print(stu[[2]][1])
print(stu$id)
print(stu$id[1])
```

```
[1] "john"
[1] 101 102 103
[1] 101
```

subset()

```
id<-c(101,102,103)
names<-c('john','bob','alice')
marks<-c(78.25,100,90.2)
stu<-data.frame(id,names,marks)
#stu
report<-subset(stu,marks>90)
print(report)
```

```
   id names marks
2 102   bob 100.0
3 103 alice  90.2
```

```
report<-subset(stu,marks>90,select=c(names,marks))
print(report)
```

```
  names marks
2   bob 100.0
3 alice  90.2
```

rbind,cbind in dataframe

```
stud<-rbind(stu,data.frame(id=104,names='vk',marks=80.52))
stud
```

A data.frame: 4 × 3

```
stud<-cbind(stu,age=c(18,19,20))
stud
```

A data.frame: 3 × 4

id	names	marks	age
<dbl>	<chr>	<dbl>	<dbl>
101	john	78.25	18
102	bob	100.00	19
103	alice	90.20	20

edit()

```
stutable<-edit(stu)
stutable
```

```
Error in edit(stu): ‘edit()’ not yet supported in the Jupyter R kernel
Traceback:

1. edit(stu)
2. stop(sQuote("edit()"), " not yet supported in the Jupyter R kernel")
```

SEARCH STACK OVERFLOW

missing dat

```
x<-c(10,NA,45.6)
is.na(x)
```

```
FALSE · TRUE · FALSE
```

```
x<-c(10,NA,45.6,NaN)
y<-is.nan(x)
y
```

```
FALSE · FALSE · FALSE · TRUE
```

```
x[!y]#removed nan
```

```
10 · <NA> · 45.6
```

```
id<-c(101,102,103,104,105)
temp<-c(25.8,34.2,NA,27.4,20.5)
```

```
wind<-c(25,45,78,40,68)
humidity<-c(25,45,85,NA,61)
weather<-data.frame(id,temp,wind,humidity)
print(weather)
```

```
     id temp wind humidity
  1 101 25.8   25       25
  2 102 34.2   45       45
  3 103   NA   78       85
  4 104 27.4   40       NA
  5 105 20.5   68       61
```

```
weatherNA<-complete.cases(weather)
weatherNA
print(weather[weatherNA,])
```

```
  TRUE · TRUE · FALSE · FALSE · TRUE
     id temp wind humidity
  1 101 25.8   25       25
  2 102 34.2   45       45
  5 105 20.5   68       61
```