Taller N1 de inferencia

Diego Alonso Perez Rojas

12 de abril de 2018

R Markdown

ayuda en linea.

Ayuda en linea para saber como utilizar funciones se utiliza usando el signo “?” seguido por el comando o funcion a consultar

```
?print
```

```
## starting httpd help server ... done
```

Una funcion que se redirige al navegador web que cumple la misma funcion que la mencionada anteriormente.

```
#help.start("lm")
```

Esta funcion sirve para conocer todas las palabras que tienen el argumento entregado se utliza ocupando la palabra reservada “apropos()”

colocando adentro lo que se quiera buscar.

```
apropos("redi")
```

```
## [1] "makepredictcall" "napredict"       "predict"         "predict.glm"
## [5] "predict.lm"
```

Introduccion a R

q() este comando es para guardar el area de trabajo

```
#q()
```

Conocer la caperta en la que se encuentra el archivo

```
getwd()
```

```
## [1] "C:/Users/dapr/Desktop/inferencia"
```

cambiar la carpeta de trabajo

```
#setwd("C:/Users/dapr/Desktop/inferencia1")
```

guardar temporalmente todo el espacio de trabajo

```
#save.image(file = "1er taller.RData")
```

Mostrar lo que esta guardado en la memoria, en el caso de no haber archivo coloca “character(0)”

```
ls()
```

```
## character(0)
```

Para borrar una funcion o alguna asignacion se utiliza el comando rm(nombre del objeto), tambien aplica para multiple variables separadas por una “,”

```
f<- 1:5    # se asigna a  la variable numeros del 1 al 5
f          # imprime los valores
```

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

```
rm(f)      # se llama la funcion
#f          # se demuestra que ya no existe, por lo tanto ocasionara un error por no existir
```

cuando se quiera comenzar un nueva sesion utlizar rm(list= ls()), asi se podra tener un espacio limpio en el cual todo se encuentra disponible

```
x<- 5:1            #se define la variable con una sucesion de numeros
x                  #impre los valores
```

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

```
rm(list = ls())    #borra toda la memoria
#x                  #comprobar que no existe nada
```

Para poder volver a recuperar el area de trabajo

```
#load("1er taller.RData")
```

En R se pueden ocupar todas las bases aritmeticas como los ejemplos que se muestran a continuacion

```
1+2+3
```

```
## [1] 6
```

```
2+3*5
```

```
## [1] 17
```

```
3/2+1
```

```
## [1] 2.5
```

```
2+(3*4)
```

```
## [1] 14
```

```
(2+3)*5
```

```
## [1] 25
```

```
3**3
```

```
## [1] 27
```

```
sqrt(4)   # raiz cuadrada
```

```
## [1] 2
```

```
pi        # numero pi
```

```
## [1] 3.141593
```

```
sin(pi)   # funcion seno
```

```
## [1] 1.224606e-16
```

```
cos(pi)   #funcion coseno
```

```
## [1] -1
```

Nombre	operacion
sqrt	raiz cuadrada
abs	valor absoluto
sin	funcion trigonometrica
asin	funcion trigonometrica inversa
cos	funcion trigonometrica
acos	funcion trigonometrica inversa
tan	funcion trigonometrica
atan	funcion trigonometrica inversa
exp	exponencial
log	logaritmo natural

Todas estas funciones se pueden mezclar

```
sqrt(cos(sin(45*pi/180)))+1
```

```
## [1] 1.87192
```

R existen una multiple variedad de funciones matematicas, ademas, se pueden crear nuevas

Asignacion de valores

Se pueden asignar valores con “<-” o “=”

```
p=2
x<-sqrt(4)       # se asigna una  raiz a la variable x
x+p
```

```
## [1] 4
```

La variable puede ser tratada como un numero.

```
x*3
```

```
## [1] 6
```

```
y= log(x)
y
```

```
## [1] 0.6931472
```

```
z= x+y        # operacion de variable como numeros
z
```

```
## [1] 2.693147
```

A su vez tambien R nos ofrece la posibilad de ejecutar operaciones logicas.

```
x<-10  # asignamos la variable
x<10   # pregunta si x es menor que 10
```

```
## [1] FALSE
```

```
4==7
```

```
## [1] FALSE
```

```
4!=7
```

```
## [1] TRUE
```

Asi como se puede guardar operaciones en varible, tambien se puede guardar las variables logicas

```
tf= x==1
tf
```

```
## [1] FALSE
```

Gestion de vectores

Para utilizar los vectores en “r” se pueden realizar de dos maneras, la primera es definirla en una variable con la funcion c()

```
x<- c(1,2,3,4)
x
```

```
## [1] 1 2 3 4
```

La segunda forma es realizarla a traves de una entrada de datos con la funcion scan()

```
d<- scan()
d
```

```
## numeric(0)
```

Para poder obtener los datos de un archivo de texto se introduce el scan() con un nuevo comando que es “redcell” y hacerlo de la siguiente manera

```
r<- redcell<- scan("data1.txt")
r
```

```
## [1]  2  3  3  5  4 55 44 55
```

Para realizar secuencias de datos se define en la variable a guardar y posteriormente se utiliza el comando “:” entre los dos numeros y se obtienen los datos entre ese rango puede ser; mayor, menor o menor, mayor

```
f<- 15:10
f
```

```
## [1] 15 14 13 12 11 10
```

```
f<- 10:15
f
```

```
## [1] 10 11 12 13 14 15
```

Una opcion alternativa es tambien utilizar el comando seq(from=el numero partidar . to= numero finalizado)

```
f<- seq(from = 1, to=5)
f
```

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

Tambien existe la opcion de crear vectores que contengan numeros repetidos con el comando “rep(numero a repetir, veces a repetir)” el cual tambien se puede utilizar con el comando de crear vectores como demuestra mas arriba

```
rep(1,5)
```

```
## [1] 1 1 1 1 1
```

```
a<- c(rep(2,3),4,5,rep(12,2))
a
```

```
## [1]  2  2  2  4  5 12 12
```

Por otro lado a estos vectores tambien se le pueden aplicar arismetica igual que los escalares.Se aplica aritmetica a cada numero del vector sin afectar su orden

```
x<-6:8
x
```

```
## [1] 6 7 8
```

```
x*2
```

```
## [1] 12 14 16
```

```
x*x
```

```
## [1] 36 49 64
```

Asimismo se le puede aplicar operaciones logicas

```
x<4
```

```
## [1] FALSE FALSE FALSE
```

Otra cara es poder extraer elementos de un vector solo utilizando la variable en la que se encuentra y colocar la posicion atraves de un “[]”

```
x[2]
```

```
## [1] 7
```

Al mismo tiempo tambien se puede extraer sub cojuntos con la funcion de crear vectores

```
x[c(1,3)]
```

```
## [1] 6 8
```

Tambien esta funcion se puede aplicar funciones de rango, crear vectires a traves de rango dentro de ello, ademas se puede sacar el mismo elemento varias veces, si el elemento no existe el programa lo coloco como “NA”

```
x[c(11,2,3,3,3,3,3)]
```

```
## [1] NA  7  8  8  8  8  8
```

Para eliminar datos solo se coloca la posicion en negativo y adios numero

```
x<- c(1,3,5,4,8,9,6,4,5)
x
```

```
## [1] 1 3 5 4 8 9 6 4 5
```

```
x[-4]
```

```
## [1] 1 3 5 8 9 6 4 5
```

otras funciones que sirven de complementos son las siguientes

funcion “length(x)” entrega el largo del vector

```
length(x)
```

```
## [1] 9
```

Obtener el max y minimo

```
max(x)
```

```
## [1] 9
```

```
min(x)
```

```
## [1] 1
```

Obtener la suma del vector

```
sum(x)
```

```
## [1] 45
```

Obtener el producto

```
prod(x)
```

```
## [1] 518400
```

Ordenar los datos de manera lineal y con el list() borra los elementos repetidos, ademas de registrar las posiciones

```
sort(x)
```

```
## [1] 1 3 4 4 5 5 6 8 9
```

```
sort.list(x)
```

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

Elementos de programacion en r

Para crear funciones en r se debe realizar de la sguiente manera

```
suma<- function(a,b,c){
  z=a+b+c
  return(z)
}
suma(4,5,1)
```

```
## [1] 10
```

Crear el promedio de manera que ocupe varias funciones aprendidas anteriormente

```
x                                # numeros a promediar
```

```
## [1] 1 3 5 4 8 9 6 4 5
```

```
promedio <- function(x){
  y<-0
  for (i  in 1: length(x)) {
    y<- y+ x[i]
  }
  y<- y/length(x)
  return(y)
}
promedio(x)          #imprime el promedio
```

```
## [1] 5
```

Una forma mas corta es realizarlo directo

```
prom<- function(x){
  sum(x)/length(x)
}
prom(x)
```

```
## [1] 5
```

La opcion que ofrece R, ademas simplifica codigo y linea

```
mean(x)
```

```
## [1] 5
```

Obetener el minimo de un vector

```
minimo <- function(x){
  d = sort(x)
  return (d[1])
  }
minimo(x)
```

```
## [1] 1
```

Calcular la varianza para n elementos de un vector

```
varianza <- function(x){
  dif=x-mean(x)
  v1=sum (dif**2)/length(x)
  return(v1)
}

varianza(x)
```

```
## [1] 5.333333
```

Coeficiente de variacion

```
coefvar<- function(x){
  dt= sqrt(varianza(x))
  cofv=dt/promedio(x)
  return (cofv)
}
coefvar(x)
```

```
## [1] 0.4618802
```

Mediana

```
mediana <- function(x){
  li=sort(x)
  if(length(li)%%2==0){
    mi=length(x)/2
    mi2=(length(x)+1)/2
    med= (mi+mi2)/2
    return(x[med])
  }else{
    mi= (length(x)/2)+0.5
    return(x[mi])
  }
}
mediana(x)
```

```
## [1] 8
```

MATRICES

Para utlizar matrices basta definir una variable y aplicar la funcion matrix() la cual adentro lleva datos, numero de filas, numeros de columna. Teniendo como advertencia que la cantidad de datos sea multiples de las filas y columnas para realizar una matriz cuadrada y poder obtener algunos datos

```
x<- c(1,3,5,4,8,9,6,4,5)
matris <-matrix(x,2,3 )
```

```
## Warning in matrix(x, 2, 3): la longitud de los datos [9] no es un
## submúltiplo o múltiplo del número de filas [2] en la matriz
```

```
matris
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    8
## [2,]    3    4    9
```

Tambien existe la posibilidad de importar directamente con scan() indicando como estan separadas

```
x2 <- scan("data2.txt", sep=",")
mx<- matrix(x2, nrow=6, ncol=6 , byrow = T)
mx
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1   24   32   36   33    2
## [2,]   16   44   34   33    3   20
## [3,]   31   43   32    4   23   35
## [4,]   37   35    5   27   40   40
## [5,]   31    6   19   43   32   37
## [6,]    1   24   32   36   33    2
```

A su vez se puede extraer datos segun se indique

```
mx[1,4]
```

```
## [1] 36
```

Si se quiere saber toda la fila o columna por si sola

```
mx[,1]
```

```
## [1]  1 16 31 37 31  1
```

```
mx[1,]
```

```
## [1]  1 24 32 36 33  2
```

Por otro lado tambien se puede crear la matris con solo indicar fila o columna y se ingresan los datos de manera ordenada

```
mx2<- matrix(1:30, nrow=6 , ncol = 6)
mx2
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    7   13   19   25    1
## [2,]    2    8   14   20   26    2
## [3,]    3    9   15   21   27    3
## [4,]    4   10   16   22   28    4
## [5,]    5   11   17   23   29    5
## [6,]    6   12   18   24   30    6
```

Extraer una submatriz se una matriz madre

```
mx2[c(1,4), c(3,4)]
```

```
##      [,1] [,2]
## [1,]   13   19
## [2,]   16   22
```

Otras funciones utiles

```
dim(mx)  # dimension de la matriz mx
```

```
## [1] 6 6
```

```
nrow(mx) # num de filas
```

```
## [1] 6
```

```
ncol(mx) # numeros de columnas
```

```
## [1] 6
```

Algunos ejercicios con matrices

```
mx+mx2        # Suma de matrices
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    2   31   45   55   58    3
## [2,]   18   52   48   53   29   22
## [3,]   34   52   47   25   50   38
## [4,]   41   45   21   49   68   44
## [5,]   36   17   36   66   61   42
## [6,]    7   36   50   60   63    8
```

```
mx-mx2        # Resta de matrices
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0   17   19   17    8    1
## [2,]   14   36   20   13  -23   18
## [3,]   28   34   17  -17   -4   32
## [4,]   33   25  -11    5   12   36
## [5,]   26   -5    2   20    3   32
## [6,]   -5   12   14   12    3   -4
```

```
mx*mx2        # Un escalar por una matriz
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1  168  416  684  825    2
## [2,]   32  352  476  660   78   40
## [3,]   93  387  480   84  621  105
## [4,]  148  350   80  594 1120  160
## [5,]  155   66  323  989  928  185
## [6,]    6  288  576  864  990   12
```

```
mx%*%mx2      # Multiplicacion de matrices cuadradas
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]  466 1234 2002 2770 3538  466
## [2,]  473 1373 2273 3173 4073  473
## [3,]  554 1562 2570 3578 4586  554
## [4,]  670 1774 2878 3982 5086  670
## [5,]  654 1662 2670 3678 4686  654
## [6,]  466 1234 2002 2770 3538  466
```

```
t(mx)         # Traspuesta de una matriz
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1   16   31   37   31    1
## [2,]   24   44   43   35    6   24
## [3,]   32   34   32    5   19   32
## [4,]   36   33    4   27   43   36
## [5,]   33    3   23   40   32   33
## [6,]    2   20   35   40   37    2
```

```
diag(mx)      # Diagonal de una matriz
```

```
## [1]  1 44 32 27 32  2
```

```
det(mx)       # Determinante de una matriz
```

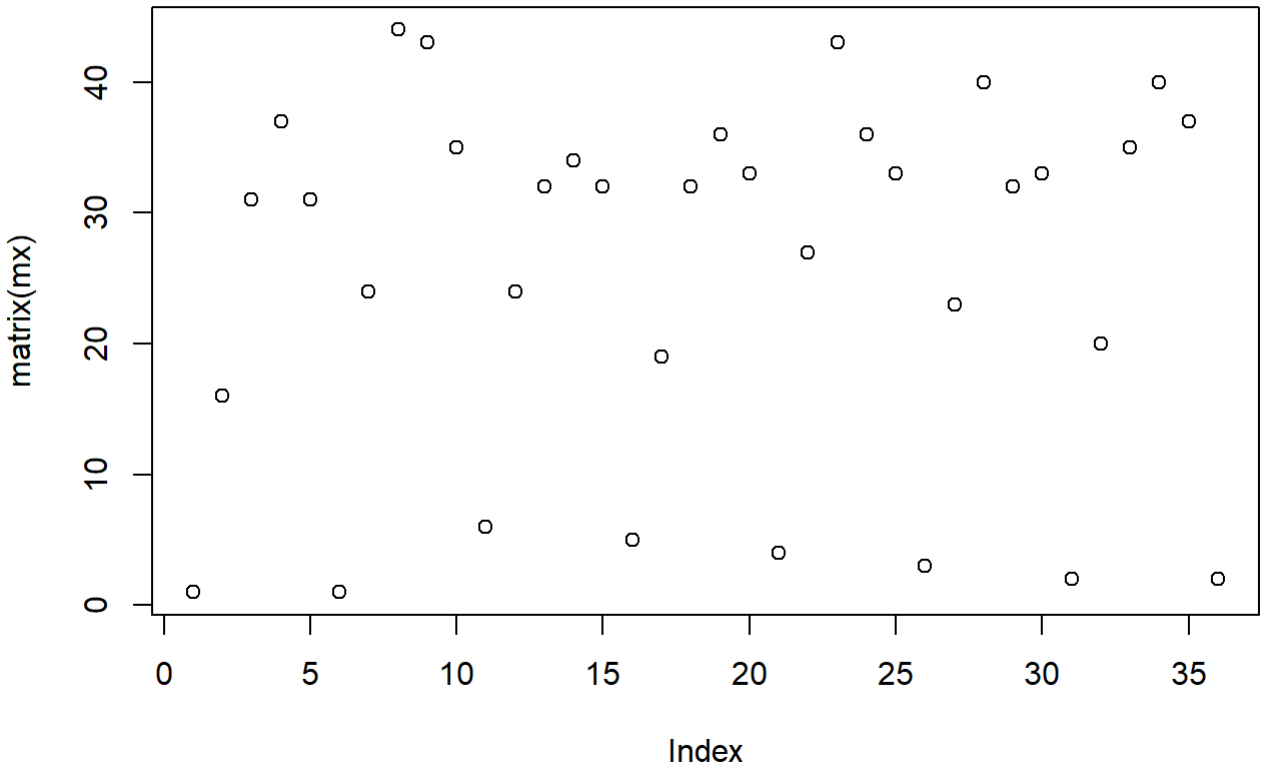
```
## [1] 0
```

Funciones extras con matrices

```
row.names(mx)=c("x","y","z","w","t","s") # cambiar nombres a las filas
colnames(mx)=c("a","b","c","d","e","f")  # cambiar nombre a las columnas
mx                                       # mostrar los cambios hechos
```

```
##    a  b  c  d  e  f
## x  1 24 32 36 33  2
## y 16 44 34 33  3 20
## z 31 43 32  4 23 35
## w 37 35  5 27 40 40
## t 31  6 19 43 32 37
## s  1 24 32 36 33  2
```

```
plot(matrix(mx))                        # construir grafico con una matriz
```



Data frames

Son un objeto parecido a la matriz pero cada columna es una variable y una fila es una unidad estadistica para leer desde un archivo de texto se utiliza la funcion read.table() y se procede igual que anteriormente

```
datos <- read.table("data3.txt")
datos
```

```
##      V1 V2   V3
## 1   8.2 70 10.3
## 2   8.6 65 10.3
## 3   8.8 63 10.2
## 4  10.5 72 16.4
## 5  10.7 81 18.8
## 6  10.8 83 19.7
## 7  11.0 66 15.6
## 8  11.0 75 18.2
## 9  11.1 80 22.6
## 10 11.2 75 19.9
## 11 11.3 79 24.2
## 12 11.4 76 21.0
## 13 11.4 76 21.4
## 14 11.7 69 21.3
## 15 12.0 75 19.1
## 16 12.9 74 22.2
## 17 12.9 85 33.8
## 18 13.3 86 27.4
## 19 13.7 71 25.7
## 20 13.8 64 24.9
## 21 14.0 78 34.5
## 22 14.2 80 31.7
## 23 14.5 74 36.3
## 24 16.0 72 38.3
## 25 16.3 77 42.6
## 26 17.3 81 55.4
## 27 17.5 82 55.7
## 28 17.9 80 58.3
## 29 18.0 80 51.5
## 30 18.0 80 51.0
## 31 20.6 87 77.0
```

Como se pudo ver tiene la misma forma que una matriz pero al analizar la estructura es algo mas complejo

```
str(datos)
```

```
## 'data.frame':    31 obs. of  3 variables:
##  $ V1: num  8.2 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
##  $ V2: int  70 65 63 72 81 83 66 75 80 75 ...
##  $ V3: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

Como las matrices se puede consultar los nombres(estos se crean automaticamente) o modificarlos

```
names(datos)
```

```
## [1] "V1" "V2" "V3"
```

```
names(datos)<-c("diametro", "altezza","volumen")
datos
```

```
##    diametro altezza volumen
## 1       8.2      70    10.3
## 2       8.6      65    10.3
## 3       8.8      63    10.2
## 4      10.5      72    16.4
## 5      10.7      81    18.8
## 6      10.8      83    19.7
## 7      11.0      66    15.6
## 8      11.0      75    18.2
## 9      11.1      80    22.6
## 10     11.2      75    19.9
## 11     11.3      79    24.2
## 12     11.4      76    21.0
## 13     11.4      76    21.4
## 14     11.7      69    21.3
## 15     12.0      75    19.1
## 16     12.9      74    22.2
## 17     12.9      85    33.8
## 18     13.3      86    27.4
## 19     13.7      71    25.7
## 20     13.8      64    24.9
## 21     14.0      78    34.5
## 22     14.2      80    31.7
## 23     14.5      74    36.3
## 24     16.0      72    38.3
## 25     16.3      77    42.6
## 26     17.3      81    55.4
## 27     17.5      82    55.7
## 28     17.9      80    58.3
## 29     18.0      80    51.5
## 30     18.0      80    51.0
## 31     20.6      87    77.0
```

Al igual que las matrices se puede llamar por columnas o filas

```
datos$diametro
```

```
##  [1]  8.2  8.6  8.8 10.5 10.7 10.8 11.0 11.0 11.1 11.2 11.3 11.4 11.4 11.7
## [15] 12.0 12.9 12.9 13.3 13.7 13.8 14.0 14.2 14.5 16.0 16.3 17.3 17.5 17.9
## [29] 18.0 18.0 20.6
```

Por otro lado se puede ocupar la funcion attach() para refererirse a una parte de los datos y poder trabajar en ella si tener que llamar a los datos todas las veces que se requieran

```
attach(datos)
mean(volumen)     #se llama ala funcion promedio para la demostracion
```

```
## [1] 30.17097
```

Hacer un reporte con lo visto anteriormente en clases

```
x<- c(1,3,5,4,4,5,9,6,4,5)     # Conjuntos de datos a analizar
mat=matrix(rep(0,17), nrow=17, ncol=1)   #Crear la matriz
rownames(mat)=c("limite superior","limite inferior","moda","mediana","media","media recortada al
 5%", "media armonica","media geometrica","rango","cuartil 1","cuartil 3","rango intercuartil",
"varianza","desviacion estandar", "coeficiente de variacion","curtosis", "coeficiente de asimetr
ia")
colnames(mat)=c("datos")        # dandonle nombres a filas y columnas de la matris

reporte<- function(c){
  limsup<- function(c){
    b=sort(c)
    d=length(c)
    e=b[d]
    return (e)
  }
  mat[1,1]=limsup(x)

  liminfer<- function(c){
    d=sort(c)
    e=d[1]
    return (e)
  }
  mat[2,1]<-liminfer(x)

  moda= function(c){  #moda solo en el caso que exista una sola
    moda1= c[0]
    suma=0
    suma_max=0
    for(i in c){
      if (c[i] == c[i+1]){
          suma= suma + 1
      }else {
        if (suma >= suma_max) {
            suma_max = suma
            suma = 1
            moda1 = c[i]
          }
      }
        i=i+1
    }
    if (moda1 == 0) {
        print("No hay moda")
    }else{
      return( moda1)
    }
  }
  mat[3,1]=moda(x)

  mediana <- function(x){
    li=sort(x)
    if(length(li)%%2==0){
      mi=length(x)/2
      mi2=(length(x)+1)/2
```

```
    med= (mi+mi2)/2
    return(x[med])
  }else{
    mi= (length(x)/2)+0.5
    return(x[mi])
  }
}
mat[4,1]=mediana(x)

promedio <- function(x){
  y<-0
  for (i  in 1: length(x)) {  #Este for corresponde a la sumatoria
    y<- y+ x[i]
  }
  y<- y/length(x)
  return(y)
}
mat[5,1]=promedio(x)

mediarec<- function(c){
  x= 0.1*(length(c))
  d=sort(c)
  aux=length(c)
  aux1= aux-x
  y<-0
  for (i  in x:aux1 ) {
    y<- y+ d[i]
  }
  y<- y/length(d)
  return(y)
}

mat[6,1]= mediarec(x)

mediaarmonica<- function(c){
  y=0
  for (i  in 1: length(c)) {
    y<- (1/y)+ (1/x[i])
  }
  y<- length(c)/y
  return(y)
}
mat[7,1]=mediaarmonica(x)

mediageometrica<-function(c){
  y=1
  for (i  in 1: length(c)) {
    y<- y * (x[i])
  }
  y<- y**(1/(length(c)))
  return(y)
}
mat[8,1]=mediageometrica(x)
```

```
rango<-function(c){
  a=sort(c)
  b=a[1]
  e=length(c)
  d=a[e]
  x=d-b
  return(x)
}
mat[9,1]=rango(c)

cuartil1<-function(c){
  s=sort(c)
  n=length(c)
  d=round(n*0.25)
  g=s[d]
  return(g)
}
mat[10,1]=cuartil1(x)

cuartil3<-function(c){
  s=sort(c)
  n=length(c)
  d=round(n*0.75)
  g=s[d]
  return(g)
}
mat[11,1]=cuartil3(x)

rangointercuartil<- function(c){
  d=cuartil3(c)
  c=cuartil1(c)
  e=d-c
  return(e)
}
mat[12,1]=rangointercuartil(x)

varianza<- function(x){
  dif=x-mean(x)
  v1=sum (dif**2)/length(x)
  return(v1)
}
mat[13,1]=varianza(x)

mat[14,1]=sqrt(varianza(x))

coefvar<- function(x){
  dt= sqrt(varianza(x))
  cofv=dt/promedio(x)
  return (cofv*100)
}
mat[15,1]=coefvar(x)

curtosis<- function(c){
  aux= promedio(c)
```

```
    aux1=0
    for (i in 1:length(c)) {
      aux1=aux1+(c[i]-aux)**4
    }
    sx<- (sqrt(varianza(c)))**4
    cur<- aux1/sx
    return(cur)
  }
  mat[16,1]=curtosis(x)

  coefasimetria<- function(c){
      aux= promedio(c)
      aux1=0
      for (i in 1:length(c)) {
        aux1=aux1+(c[i]-aux)**3
      }
      sx<- (sqrt(varianza(c)))**3
      cur<- aux1/sx
      return(cur)
  }
  mat[17,1]=coefasimetria(x)
  mat
}

reporte(x)
```

```
##                              datos
## limite superior           9.000000
## limite inferior           1.000000
## moda                      4.000000
## mediana                   4.000000
## media                     4.600000
## media recortada al 5%     3.700000
## media armonica           11.444660
## media geometrica          4.085644
## rango                     8.000000
## cuartil 1                 3.000000
## cuartil 3                 5.000000
## rango intercuartil        2.000000
## varianza                  3.840000
## desviacion estandar       1.959592
## coeficiente de variacion 42.599822
## curtosis                 37.545573
## coeficiente de asimetria  4.879843
```