Taller 1

Danilo Ibáñez Rojas

11 de abril de 2018

Mas información en: ***taiv.io/N***
soumraked.github.io/Soumrak

Introducción al software estadístico R

En el siguiente archivo se dará una introducción básica a “R”, esto en base a ejemplos y definiciones que se detallarán en cada punto.

1.Opción de ayuda

La ayuda es usada en este lenguaje con el propósito de orientar al programador en la función de algún comando, esto lo realiza dando una breve descripción de dicho comando, los argunmentos que puede utilizar con su respectiva definición y ejemplos que pueden ser utiles para orientarse en este tema.
Para poder usar la opción de ayuda se debe escribir en la consola “help(comando)” o “?comando” sin comillas, esto abrirá una página web con los detalles del comando.

Ejemplos con el uso de ambas formas para acceder a la ayuda.

```
help(hist)
?hist

help(scan)
?scan

help(hist)
?hist
```

2.Introducción a R

2.1 Aritmética básica

Al igual que una calculadora, R puede resolver distintas operaciones artiméticas mediante la unión de números y ciertos simbolos claves.

Simbolo	Descripción	Ejemplos
+	Suma	`1+2+3`
-	Resta	`3-2`
*	Multiplicación	`3*4`
/	División entera	`4/3`

Simbolo	Descripción	Ejemplos
^	Potencia	`4^3`
sqrt()	Raiz cuadrada	`sqrt(8)`
abs()	Valor absoluto	`abs(-6)`
pi	Valor de pi = 3.14…	`pi/2`
sin() cos() tan()	Funciones trigonométricas	`sin(0)  cos(1)  tan(1)`
asin() acos() atan()	Funciones trigonométricas inversas	`asin(0)  acos(1)  atan(1)`
exp() log()	Exponencial y logaritmo	`exp(1)  log(1)`

2.2 Asignación de valores

Asignar un valor es simplemente guardar el valor en alguna variable para después poder usarlo.
Para poder asignar se debe ocupar simbolos “<-” o “=” dejando en la izquierda el nombre de la variable y a la izquierda el valor o expresión.

```
x <- 10
y <- sqrt(4)
z = (x+y)/pi
print(z) #Muestra el valor de z en pantalla
```

```
## [1] 3.819719
```

2.3 Valores lógicos

Los valores lógicos ayudan al momento de comparar números o expresiones lógicas devolviendo valores binarios `TRUE` o `FALSE`.

Operador lógico	Descripción	Ejemplos
<	Simbolo de menor que	`10 < 20`
>	Simbolo de mayor que	`10 > 20`
<=	Simbolo de menor o igual que	`10 <= 10`
>=	Simbolo de mayor o igual que	`10 >= 10`
==	Verifica la igualdad entre expresiones	`10 == 20`
!	Es la negación de una expresión lógica	`!(10 < 20)`
&	Es la conjunción lógica, conocida como “y”	`TRUE & FALSE`

Operador lógico	Descripción	Ejemplos
|	Es la disjunción lógica, conocida como “o”	`TRUE | FALSE`

2.4 Vectores

Un vector es un tupla de `n` números reales los cuales comforman las componentes del vector.

2.4.1 Creación de un vector

Para crear un vector con datos que ya se saben se usa la siguiente sintaxis `vector <-c(1,2,3)` , si los datos se quieren introducir uno a uno sin saber la cantidad de estos se ocupa `vector <- scan()` , pero si el largo es conocido de podría usar `vector <-scan(nmax= n)` donde `n` seria la cantidad de elementos del vector, por último, si se tienen datos en un archivo de texto estos se pueden importar automáticamente usando `vector <- scan("nombreDelArchivo.txt)"` siempre y cuando el archivo de texto esté en la misma carpeta de trabajo de R.
Ejemplo:

```
x <-c(1,2,3,4,5)
y <- scan()
z <- scan(nmax=2)
archivo <- scan("datos.txt")
```

2.4.2 Secuencias númericas

Un vector puede crearse mediante una secuencia `a:b` donde `a` y `b` son los números extremos que tendrá el vector, el orden de este (ascendente o descendente) dependerá del orden en el que se escribieron los números, si `a` es menor que `b` el orden será ascendente, en caso contrario descedente.

```
x <- 1:10
print(x)
```

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

```
y <- 10:1
print(y)
```

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

Lo anterior también s epuede imprementar de la siguiente forma:

```
x <- seq(from=1, to=10)
print(x)
```

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

```
y <- seq(from=10, to=1)
print(y)
```

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

Otro tipo de secuencia puede existir al escribir cierto elemento varias veces en el mismo vector, esto se puede simplificar con el comando `rep(n,times=m)`, donde `n` es el valor del vector y `m` la cantidad de veces que se repitirá.

```
  x <-c(1, rep(2,times=3), 3)
  print(x)
```

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

```
  y <-c(rep(1, times=2), 2, rep(3, times= 5))
  print(y)
```

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

A los vectores se les puede aplicar la misma aritmética básica que ocupan los escalares.

```
  x <-c(1,2,3)
  y <- x*x #Multiplica los valores de las mismas posiciones de ambos vectores
  print(x)
```

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

```
  print(y)
```

```
## [1] 1 4 9
```

También existe un vector lógico por así decirlo, el cual se obtiene al evaluar alguna regla a cierto vector.

```
  x <-c(1,2,3,4,5,6,7,8,9)
  y <- x>4 #Crea un vector indicando los valores mayores a 4
  print(x)
```

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

```
  print(y)
```

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

2.4.3 Extracción de un vector

Se pueden extraer elementos de un vector mediante su índice.

```
  vector <-c(10,20,30,40,50,60,70,80,90)
  print(vector[2]) #Extrae el valor del índice 2
```

```
## [1] 20
```

```
  print(vector[7]) #Extrae el valor del índice 7
```

```
## [1] 70
```

```
  print(vector[c(2,5,8)]) #Crea un vector con el valor de los índices 2,5 y 8
```

```
## [1] 20 50 80
```

```
  print(vector[1:4]) #Crea un vector con los numeros desde el índice 1 hasta el 4
```

```
## [1] 10 20 30 40
```

```
  print(vector[c(1:3,6:9)])#Crea un vector con los datos del 1 al 3 y del 6 al 9
```

```
## [1] 10 20 30 60 70 80 90
```

```
  print(vector[-2]) #Crea un vector con todos los datos menos el valor del índice 2
```

```
## [1] 10 30 40 50 60 70 80 90
```

```
  print(vector[c(-1,-9)])#Crea un vector con todos los datos menos los valores del índice 1 y 9
```

```
## [1] 20 30 40 50 60 70 80
```

Los vectores cuentan con ciertas funciones extras.

función	**Descripción**
length()	Muestra el largo del vector
max()	Muestra el valor máximo del vector
min()	Muestra el valor mínimo del vector
sum()	Entrega la suma de todos los elementos del vector
prod()	Entrega la multiplicación de todos los elementos del vector
sort()	Ordena los valores del vector de menor a mayor

función	Descripción
sort.list()	Entrega un vector ordenando los índices del vector

Ejemplos:

```
a <-c(7,9,4,5,1,3,8,2,6)
print(length(a))
```

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

```
print(max(a))
```

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

```
print(min(a))
```

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

```
print(sum(a))
```

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

```
print(prod(a))
```

```
## [1] 362880
```

```
print(sort(a))
```

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

```
print(sort.list(a))
```

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

2.5 Matrices

R permite utilizar matrices. Para crear una matriz se debe introducir un vector, verificando que los números llenarán primero las columna. En el vector se introducen dos comandos, “nrow” y “ncol”, los cuales significan el número de filas y columnas que tendrá nuestra matriz respectivamente.

```
matrix(vector, nrow, ncol) #Sintaxis del comando matrix
```

Un ejemplo usando matrix es:

```
  matrix(c(1,2,3,4,5,6,7,8),nrow = 2, ncol = 4)
```

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

Comandos	Descripción
dim()	Entrega la dimención de la matriz en el formato “fila columna”
nrow()	Entrega el número de filas
ncol()	Entrega el número de columnas

Se pueden extraer elementos de una matriz de dos formas, mediante los indices y mediante vectores poniendo en el primer vector las filas que ocuparé y en el segundo vector las columnas.

```
  M <- matrix(c(1:9), nrow=3, ncol =3)
  print(M)
```

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

```
  M[2,3] #Extración con coordenadas
```

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

```
  M[c(1,2,3),c(2,3)] #Extración mediante vectores
```

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

Las matrices permiten las siguientes operaciones, sea “A” y “B” dos matrices de la misma dimención y “c” un escalar.

Operación	Descripción
A+B	Suma de matrices
A-B	Resta de matrices
c*A	Multiplicación de un escalar a una matriz

Operación	Descripción
A%*%B	Multiplicación de matrices
t(A)	Matriz transpuesta
diag(A)	Entrega el vector de la diagonal de la matriz
det(A)	Entrega el determinante de una matriz

Ejemplos:

```
M1 <- matrix(c(1:9), nrow=3, ncol=3)
M2 <- matrix(c(9:1), nrow=3, ncol=3)
print(M1)
```

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

```
print(M2)
```

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

```
print(M1+M2)
```

```
##      [,1] [,2] [,3]
## [1,]   10   10   10
## [2,]   10   10   10
## [3,]   10   10   10
```

```
print(M1-M2)
```

```
##      [,1] [,2] [,3]
## [1,]   -8   -2    4
## [2,]   -6    0    6
## [3,]   -4    2    8
```

```
print(5*M1)
```

```
##      [,1] [,2] [,3]
## [1,]    5   20   35
## [2,]   10   25   40
## [3,]   15   30   45
```

```
  print(M1%*%M2)
```

```
##      [,1] [,2] [,3]
## [1,]   90   54   18
## [2,]  114   69   24
## [3,]  138   84   30
```

```
  print(t(M1))
```

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

```
  print(diag(M1))
```

```
## [1] 1 5 9
```

```
  print(det(M1))
```

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

2.6 Data frame

Un data-frame es un objeto similar a una matriz, sin embargo, al contrario que en las matrices aquí pueden existir distintos tipos de datos en su interior. Estos tienen una categoria, cada fila es una unidad estadística y cada columna es una variable, estas últimas pueden ser numéricas o categoriales.
Se puede leer un data-frame desde un archivo de texto usando la función `read.table()`

```
    nuevo <- read.table("Archivo.dat")
    nuevo <- read.table("Arvhivo.dat",col.names = c("nombre1", "nombre2")) #Incluye los nombre q
ueridos al cargar el archivo al data-frame
```

Se pueden usar ciertas funciones sobre un data-frame.

Función	Descripción
dim()	Entrega la dimención del data-frame
str()	Muestra la estructura interna del objeto

Función	Descripción
names()	Entrega los nombres de las variables, si no fuerons introducidos quedan por defecto como V1,V2,..,Vn
names()<-c()	Introduce los nombres de variables al data frame en forma de vector

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

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

```
str(datos)
```

```
## 'data.frame':    4 obs. of  2 variables:
##  $ V1: num  1.5 7.8 8.23 6.9
##  $ V2: num  2.5 3.6 7.9 9.23
```

```
names(datos)
```

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

```
names(datos)<-c("Columna 1", "Columna 2")
names(datos)
```

```
## [1] "Columna 1" "Columna 2"
```

```
datos<-read.table("datos.txt", col.names = c("Columna 1", "Columna 2"))
datos
```

```
##   Columna.1 Columna.2
## 1      1.50      2.50
## 2      7.80      3.60
## 3      8.23      7.90
## 4      6.90      9.23
```

Se pueden ver los datos de variables de dos formas:

```
datos<-read.table("datos.txt", col.names = c("Columna1", "Columna2"))
datos
```

```
##   Columna1 Columna2
## 1     1.50     2.50
## 2     7.80     3.60
## 3     8.23     7.90
## 4     6.90     9.23
```

```
   datos$Columna1
```

```
## [1] 1.50 7.80 8.23 6.90
```

```
   datos[,2]
```

```
## [1] 2.50 3.60 7.90 9.23
```

Existe la función `attach()` la cual comunica a R que se realizaran operaciones referentes al data-frame.

```
   datos<-read.table("datos.txt", col.names = c("Columna1", "Columna2"))
   attach(datos)
   Columna1
```

```
## [1] 1.50 7.80 8.23 6.90
```

```
   Columna2
```

```
## [1] 2.50 3.60 7.90 9.23
```

2.7 Elementos de programación en R

Para crear una función en R se debe seguir la siguiente estructura

```
nombreDeLaFuncion <- function(parametros,separdos,por,comas){
#Expresiones
}
```

Funciones estadisticas implementadas en R

Mediana:

```
mediana <- function(vector){
  vector <- sort(vector)
  largo = length(vector)
  if ((largo%%2) == 0)
    return ((vector[largo/2] + vector[(largo/2)+1])/2)
  return (vector[(largo+1)/2])
}
```

Media aritmética:

```
media <- function(vector){
  vector <- sort(vector)
  largo = length(vector)
  vectorSuma = sum(vector)
  return (vectorSuma/largo)
}
```

Moda:

```
moda <- function(vector){
  vector = sort(vector)
  vVal <- c()
  vRep <- c()
  for( i in vector){
    if(!(is.element(i,vVal))){
      vVal <- c(vVal,i)
    }
  }
  for(i in vVal){
    rep <- 0
    for(j in vector){
      if(i == j){
        rep <- rep + 1
      }
    }
    vRep <- c(vRep,rep)
  }
  modas <- c()
  repMax <- max(vRep)
  for(i in 1:length(vVal)){
    if(vRep[i] == repMax){
      modas <- c(modas,vVal[i])
    }
  }
  return(modas)
}
```

Percentil:

```
percentil<- function(vector,val){
  valor <- val/100
  if(valor==0 ){
    return(min(vector))
  }
  else if(valor == 1){
    return(max(vector))
  }
  vector <-sort(vector)
  n <- length(vector)
  i <- valor*(n+1)
  e <- floor(i)
  d <- (i-e)
  p <-(1-d)*vector[e]
  if(d>0){
    p <- p + (d*vector[e+1])
  }
  return(p)
}
```

Cuartil 1:

```
  cuartil1<- function(vector){
    return(percentil(vector,25))
  }
```

Cuartil 2:

```
  cuartil2<- function(vector){
    return(percentil(vector,50))
  }
```

Cuartil 3:

```
  cuartil3<- function(vector){
    return(percentil(vector,75))
  }
```

Rango:

```
rango<-function(vector){
  rmin <- vector[1]
  rmax <- vector[(length(vector))]
  return(rmax - rmin)
}
```

Rango intercuartil:

```
  rangoInter<-function(vector){
  rmin <- cuartil1(vector)
  rmax <- cuartil3(vector)
  return(rmax - rmin)
}
```

Varianza muestral:

```
  varianza<-function(vector){
    me = media(vector)
    var <- 0
    for(x in vector){
      var <- var + (x-me)^2
    }
    return((1/(length(vector)-1))*var)
  }
```

Desviacion estandar:

```
  desviacionEstandar<-function(vector){
    return(sqrt(varianza(vector)))
  }
```

Desviación media:

```
  desviacionMedia<-function(vector){
    me <- media(vector)
    dem <- 0
    for(x in vector){
      dem <- dem + abs(x-me)
    }
    return((1/length(vector))*dem)
  }
```

Coeficiente de variación:

```
  coeficienteVariacion <- function(vector){
    return((desviacionEstandar(vector)/media(vector))*100)
  }
```

Coeficiente de asimetria:

```
  coeficienteAsimetria <- function(vector){
    s <- desviacionEstandar(vector)
    me <- media(vector)
    cs <- 0
    for(x in vector){
      cs <- cs + (x -me)^3
    }
    return(cs/(s^3))
  }
```

Coeficiente de curtosis:

```
coeficienteCurtosis <- function(vector){
  s <- desviacionEstandar(vector)
  me <- media(vector)
  cs <- 0
  for(x in vector){
    cs <- cs + (x -me)^4
  }
  return(cs/(s^4))
}
```

Puntuación z :

```
zInferior <-function(vector){
  vector <- sort(vector)
  s <- desviacionEstandar(vector)
  me <- media(vector)
  zI <- (me-3*s)
  for(i in vector){
    if(i>=zI){
      return(i)
    }
  }
}
zSuperior <-function(vector){
  vector <- sort(vector)
  s <- desviacionEstandar(vector)
  me <- media(vector)
  zs <- (me+3*s)
  resp <- vector[1]
  for(i in vector){
    if(i>=zs){
      return(resp)
    }
    resp <- i
  }
  return(resp)
}
puntuacionZ<-function(vector){
  vector <- sort(vector)
  zi <- zInferior(vector)
  zs <- zSuperior(vector)
  pz <- c()
  for(i in vector){
    if((i>=zi)&&(i<=zs)){
      pz <-c(pz,i)
    }
  }
  return(pz)
}
```

Barreras de Tukey:

```
bInteriorInferior<-function(vector){
  return(cuartil1(vector)-1.5*rangoInter(vector))
}
bInteriorSuperior<-function(vector){
  return(cuartil3(vector)+1.5*rangoInter(vector))
}
bExteriorInferior<-function(vector){
  return(cuartil1(vector)-3*rangoInter(vector))
}
bExteriorSuperior<-function(vector){
  return(cuartil3(vector)+3*rangoInter(vector))
}
```

Extra, redondear vector:

```
roundV<-function(vector){
  v <-c()
  for(i in vector){
    v <-c(v,round(i,digits=2))
  }
  return(v)
}
```

Explorar:

```
    explorar <- function(nombreDatos,vector){
      print(paste("Resumen de",nombreDatos))
      print("Datos recopilados:")
      print(vector)

      n <- length(moda(vector))
      mTendeCen <- c("Minimo","Maximo","Media", "Mediana", "Moda")
      cuartiles<-c("Cuartil 1","Cuartil 2","Cuartil 3")
      rangos<-c("Rango","Rango intercuartil")
      varianza<-c("Varianza muestral","Desviacion estandar","Desviacion media","Coeficiente de v
ariacion")
      datosAtipicos<-c("Coeficiente de asimetria","Coeficiente de curtosis","Puntuacion z inferi
or","Puntuacion z superior")
      tukey<-c("Barrera interior inferior Tukey","Barrera interior superior Tukey","Barrera exte
rior inferior Tukey","Barrera exterior superior Tukey")
      names <-c(mTendeCen,cuartiles,rangos,varianza,datosAtipicos,tukey)

      aux <- paste(moda(vector),collapse = " ")
      mTendeCenVal<-c(min(vector),max(vector),media(vector),mediana(vector),aux)
      cuartilesVal<-c(cuartil1(vector),cuartil2(vector),cuartil3(vector))
      rangosVal<-c(rango(vector),rangoInter(vector))
      varianzaVal<-c(varianza(vector),desviacionEstandar(vector),desviacionMedia(vector), coefic
ienteVariacion(vector))
      datosAtipicosVal<-c(coeficienteAsimetria(vector),coeficienteCurtosis(vector),zInferior(vec
tor),zSuperior(vector))
      tukeyVal<-c(bInteriorInferior(vector),bInteriorSuperior(vector),bExteriorInferior(vector),
bExteriorSuperior(vector))
      valores <-c(mTendeCenVal,cuartilesVal,rangosVal,roundV(varianzaVal),roundV(datosAtipicosVa
l),tukeyVal)

      tabla <- matrix(valores,nrow=22,byrow=T)
      dimnames(tabla)<-list(names,c("Valores"))
      tabla<-data.frame(tabla)
      print(tabla)

    }
    explorar("Completos que come un estudiante a la semana",c(1,2,3,4,5,1,2,3))
```

```
## [1] "Resumen de Completos que come un estudiante a la semana"
## [1] "Datos recopilados:"
## [1] 1 2 3 4 5 1 2 3
##                                 Valores
## Minimo                                1
## Maximo                                5
## Media                             2.625
## Mediana                             2.5
## Moda                              1 2 3
## Cuartil 1                          1.25
## Cuartil 2                           2.5
## Cuartil 3                          3.75
## Rango                                 2
## Rango intercuartil                  2.5
## Varianza muestral                  1.98
## Desviacion estandar                1.41
## Desviacion media                   1.12
## Coeficiente de variacion          53.63
## Coeficiente de asimetria           2.52
## Coeficiente de curtosis           12.65
## Puntuacion z inferior                 1
## Puntuacion z superior                 5
## Barrera interior inferior Tukey    -2.5
## Barrera interior superior Tukey     7.5
## Barrera exterior inferior Tukey   -6.25
## Barrera exterior superior Tukey   11.25
```