SESION 6

**Sesión 6 R objects and attributes**

**Contenido:**

R. **Actividad 6**

Aplicaciones en R utilizando los datos de esta sesión.

Fundamentos de programación en R

Sesión 1. Objetos

Julián López Hernández

Contenido

1. Objetos
2. Números y vectores
3. Matrices
4. Arrays o arreglos
5. Data frame o marco de datos
6. Listas

1. Objetos

R trabaja con objetos que tienen un **nombre** y un **contenido**, sin embargo, también tienen atributos que especifican el tipo de datos presentados por el objeto.

Todo objeto tiene dos atributos: el tipo y la longitud

El **tipo** se refiere a la clase básica de los elementos del objeto: numérico, caracter, complejo o lógico.

La **longitud** es simplemente el número de elementos en el objeto

x<-1

class(x)

## [1] "numeric"

mode(x)

## [1] "numeric"

length(x)

## [1] 1

A<-"facultad"

valor<-TRUE

l<-1i

mode(A); mode(valor); mode(l)

## [1] "character"

## [1] "logical"

## [1] "complex"

class(A); class(valor); class(l)

## [1] "character"

## [1] "logical"

## [1] "complex"

Cuando los datos no están disponibles, se presentan como **NA (Not available)**

## objeto Tipos Varios\_objetos

## 1 vector num, char, comp, log FALSE

## 2 factor num o char FALSE

## 3 arrays num, char, comp, log FALSE

## 4 matriz num, char, comp, log FALSE

## 5 data.frame num, char, comp, log TRUE

## 6 ts num, char, comp, log TRUE

## 7 lista num, char, comp, log TRUE

Los objetos pertenecen a una **clase**, lo cual es fundamental en R que utiliza el paradigma de **programación orientada a objetos**.

La clase es una propiedad asignada a un objeto.

2.1. Vectores numéricos

La estrucutura de datos mas simple es el vector, que es una colección ordenada de números.

La asignación de elementos a una variable requiere una concatenación de elementos, que pueden ser otros vectores.

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

x

## [1] 1 2 3 4 5

y<-c(4,x,8)

y

## [1] 4 1 2 3 4 5 8

Aritmética vectorial

Es posible realizar operaciones aritméticas con nuestros vectores, cuando estos son numéricos. Los operadores mas comunes son:

* Operadores aritméticos habituales: +,-,\*,/,^
* También tenemos log,exp,sqrt,sin, cos,tan
* Operadores estadisticos: max, min, sum, prod, mean,var,sd

La operación se realiza sobre el objeto

z<-5:15

sum(z)

## [1] 110

mean(z)

## [1] 10

max(z) ; min(z)

## [1] 15

## [1] 5

Secuencias en vectores

Para construir un vector que contenga una secuencia numérica es posible usar dos procedimientos:

1. Usar from:to
2. Usar la función seq(from, to,by)

\*R es un lenguaje posicional, por lo que no siempre hay que hacer explícito el argumento de la función.

s<-2\*1:15

s

## [1] 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30

seq(from=2,to=30,by=2)

## [1] 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30

seq(2,30,2)

## [1] 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30

La función rep() es útil para repetir datos

Los argumentos de esta función son:

* x: vector con los elementos a repetir.
* times: número de veces que el vector x se debe repetir.
* length.out: longitud deseada para el vector resultante.
* each: número de veces que cada elemento de x se debe repetir.

rep(x=1:4, times=c(2,2,2,2))

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

rep(x=1:4, times=c(2,1,2,1))

## [1] 1 1 2 3 3 4

rep(x=1:4, each=2)

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

2.2. Vectores lógicos

Los vectores lógicos están compuestos por datos *booleanos* True and False.

x<-c(1,4:8,10:5,7)

x

## [1] 1 4 5 6 7 8 10 9 8 7 6 5 7

x>5

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

## [13] TRUE

x==5

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

## [13] FALSE

Operadores lógicos

Los operadores lógicos se presentan en la siguiente tabla:

## Simbolo Expresa

## [1,] "<" "menor"

## [2,] "<=" "menor o igual"

## [3,] ">" "mayor"

## [4,] "==" "igual"

## [5,] "!=" "diferente"

## [6,] "&" "intersección"

## [7,] "|" "unión"

## [8,] "!" "negación"

Como se indicó mas arriba, cuando falta un elemento se denomina presenta con **NA**. La función is.na() crea un vector lógico de tamaño x cuyo elementos serán True si el dato está perdido y FALSE en caso contrario. El comando complete.cases() opera de forma análoga.

\*Puedo transfomar un vector lógico en un vector numérico

z<-c(1:4,NA,6:3,NA)

z

## [1] 1 2 3 4 NA 6 5 4 3 NA

x<-is.na(z)

x

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

complete.cases(z)

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

También poedo coercionar un vector lógico para que se vuelva numérico

x

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

as.numeric(x)

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

2.3. Vectores de texto

Puedo armar cadenas de caracteres con el comando paste. Puede usarse para concatenar vectores

nombres<-c("Andrés","Pedro","Ana")

apellido<-c("López","Ortega","Perez")

paste(nombres,apellido)

## [1] "Andrés López" "Pedro Ortega" "Ana Perez"

paste(nombres,1:3,sep = ":")

## [1] "Andrés:1" "Pedro:2" "Ana:3"

Filtrar elementos de un vector

Para extraer un elemento almacenado dentro un vector se usan los corchetes []

edad <- c(15, 19, 13, NA, 20,NA)

edad[3] *# filtrar elemento 3*

## [1] 13

edad[c(2, 5)] *# filtrar elementos 2 y 5*

## [1] 19 20

edad[-3] *# filtrar todo, menos el elemento 3*

## [1] 15 19 NA 20 NA

edad[-c(2, 5)] *# filtrar todo, menos los elementos 2 y 5*

## [1] 15 13 NA NA

edad[!is.na(edad)] *#filtra todo sin datos NA*

## [1] 15 19 13 20

Ejercicios

1. Cosntruya un vector de los números pares mayores a 20 y menores a 60
2. Cosntruya un vector de 10 elementos entre 20 y 60, ¿qué intervalo debe haber entre cada elemento?
3. El comando LETTERS arroja un vector con las letras del alfabéto. Indique cuántos elementos tiene y qué tipo de objeto es.
4. Construya un vector con la primeras 15 letras MAYÚSCULAS usando la función LETTERS.
5. Calcule el valor máximo, el mínimo, la suma de los elementos, el producto de los elementos, la media y la desviación estandar del siguiente vecor numérico: [5,3,2,4,2.3,4.2,7,1.2,2,2,4,3.5,4]

4. Matrices

Las matrices son arreglos rectangulares de filas y columnas con información numérica, alfanumérica o lógica.

Para construir una matriz se usa la función matrix( ).

mimatriz <- matrix(data=1:20, nrow=4, ncol=5, byrow=FALSE)

mimatriz

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

## [1,] 1 5 9 13 17

## [2,] 2 6 10 14 18

## [3,] 3 7 11 15 19

## [4,] 4 8 12 16 20

Filtros en matrices

Filtros en matrices

mimatriz[3, 4] *# Filtros de elementos*

## [1] 15

mimatriz[, 5] *# Filtro de columna*

## [1] 17 18 19 20

mimatriz[, -c(2, 4)] *# Filtro quitando columnas 2 y 4*

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

## [1,] 1 9 17

## [2,] 2 10 18

## [3,] 3 11 19

## [4,] 4 12 20

Las funciones cbind() y rbind() permiten cosntruir matrices a partir de vectores

rbind() une vectores por filas (row)

idalumno<-1:6

calificacion<-c(90,92,89,69,70,90)

matriz1<-rbind(idalumno,calificacion)

Por su parte, cbind une vectores por columnas (col)

alumno7<-c(7,80,6)

alumno8<-c(8,90,6)

matriz2<-cbind(alumno7,alumno8)

matriz2

## alumno7 alumno8

## [1,] 7 8

## [2,] 80 90

## [3,] 6 6

5. Arrays o arreglos

Un arreglo es una matriz de varias dimensiones con información numérica, alfanumérica o lógica. Para construir una arreglo se usa la función array( ).

miarray <- array(data=letters[1:16], dim=c(2, 4, 2))

miarray

## , , 1

##

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

## [1,] "a" "c" "e" "g"

## [2,] "b" "d" "f" "h"

##

## , , 2

##

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

## [1,] "i" "k" "m" "o"

## [2,] "j" "l" "n" "p"

Convirtamos un vector en un array con la función dim()

calif<-rnorm(24,90,5) *# Vector aleatorio*

calif

## [1] 91.38687 92.39696 86.60451 89.36359 86.37195 91.36911 100.08788

## [8] 93.11993 97.98271 85.51254 90.34297 91.61320 90.48587 82.89165

## [15] 87.73135 89.54534 88.27671 97.96818 86.23784 90.45381 83.62365

## [22] 88.12265 91.58578 84.28599

dim(calif) *# No tiene dimensiones*

## NULL

dim(calif)<-c(6,4) *#Le doy dimensiones*

calif

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

## [1,] 91.38687 100.08788 90.48587 86.23784

## [2,] 92.39696 93.11993 82.89165 90.45381

## [3,] 86.60451 97.98271 87.73135 83.62365

## [4,] 89.36359 85.51254 89.54534 88.12265

## [5,] 86.37195 90.34297 88.27671 91.58578

## [6,] 91.36911 91.61320 97.96818 84.28599

Ejercicios

1. Construya una matriz de 10×10 con los primeros 100 números positivos pares.
2. Construya una matriz identidad de dimension 3×3. Recuerde que una matriz identidad tiene sólo unos en la diagonal principal y los demás elementos son cero.
3. ¿Cuál es el error al correr el siguiente código? ¿A qué se debe?

edad <- c(15, 19, 13, NA, 20) deporte <- c(TRUE, TRUE, NA, FALSE, TRUE) comic\_fav <- c(NA, ‘Superman’, ‘Batman’, NA, ‘Batman’) matrix(edad, deporte, comic\_fav)

1. Corrija el error y filtre las filas que tengan información completa.

4. Data frame o marco de datos

Los **Data frame** son uno de los objetos más utilizados porque permite agrupar vectores con información de diferente tipo (numérica, alfanumérica o lógica) en un mismo objeto, la única restricción es que los vectores deben tener la misma longitud.

Para crear un marco de datos se usa la función data.frame( )

edad <- c(15, 19, 13, NA, 20)

deporte <- c(TRUE, TRUE, NA, FALSE, TRUE)

comic\_fav <- c(NA, 'Superman', 'Batman', NA, 'Batman')

mimarco <- data.frame(edad, deporte, comic\_fav)

mimarco

## edad deporte comic\_fav

## 1 15 TRUE <NA>

## 2 19 TRUE Superman

## 3 13 NA Batman

## 4 NA FALSE <NA>

## 5 20 TRUE Batman

4. Data frame o marco de datos

miDf01 <- data.frame(

numbers = c(1, 2, 3, 4),

logicals = c(TRUE, TRUE, FALSE, TRUE),

characters = c("a", "b", "c", "d")

)

print(miDf01)

## numbers logicals characters

## 1 1 TRUE a

## 2 2 TRUE b

## 3 3 FALSE c

## 4 4 TRUE d

Filtros en data frame

Para filtrar en marcos de datos se puede usar el operador $ (filtra un vector), corchetes simples [] (como df) o corchetes dobles [[]]

mimarco$deporte *#se filtra la variable deporte*

## [1] TRUE TRUE NA FALSE TRUE

mimarco[, 2]

## [1] TRUE TRUE NA FALSE TRUE

mimarco[["deporte"]]

## [1] TRUE TRUE NA FALSE TRUE

Para extraer un data frame con algunas variables

mimarco[c("deporte", "edad")]

## deporte edad

## 1 TRUE 15

## 2 TRUE 19

## 3 NA 13

## 4 FALSE NA

## 5 TRUE 20

Función subset

Para extraer partes de un marco de datos se puede utilizar la función subset(x, subset, select)

subset(mimarco, subset=deporte == TRUE) *# Quienes practican deporte*

## edad deporte comic\_fav

## 1 15 TRUE <NA>

## 2 19 TRUE Superman

## 5 20 TRUE Batman

subset(mimarco, subset=edad >= 17) *#Mayores de 17 años*

## edad deporte comic\_fav

## 2 19 TRUE Superman

## 5 20 TRUE Batman

subset(mimarco, subset=edad < 20, select=c('deporte', 'comic\_fav')) *#deporte y comic de los menores de 20 años*

## deporte comic\_fav

## 1 TRUE <NA>

## 2 TRUE Superman

## 3 NA Batman

5. Listas

Las listas son otro tipo de objeto muy usado para almacenar objetos de diferente tipo.

La instrucción para crear una lista es list( ).

mivector <- runif(n=5)

matriz2 <- matrix(data=1:10, ncol=5)

milista <- list(E1=mivector, E2=matriz2, E3=mimarco)

milista

## $E1

## [1] 0.5424767 0.6806203 0.2351745 0.3509040 0.7102877

##

## $E2

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

## [1,] 1 3 5 7 9

## [2,] 2 4 6 8 10

##

## $E3

## edad deporte comic\_fav

## 1 15 TRUE <NA>

## 2 19 TRUE Superman

## 3 13 NA Batman

## 4 NA FALSE <NA>

## 5 20 TRUE Batman

milista <- list(E1=mivector, E2=matriz2, E3=mimarco)

milista[[3]][,2]

## [1] TRUE TRUE NA FALSE TRUE

milista$E3[3]

## comic\_fav

## 1 <NA>

## 2 Superman

## 3 Batman

## 4 <NA>

## 5 Batman

Ejercicios

1. Cosntruya un data frame con 10 observaciones y las siguientes variables: nombre, edad, nivel de escolaridad.