DEPARTAMENTO DE CIENCIAS DE LA ATMÓSFERA Y LOS OCÉANOS

SEMINARIO DE COMPUTACIÓN

Lenguaje de Programación R

2do Cuatrimestre 2023 - Moira E. Doyle

ARREGLOS, FUNCIONES INTERNAS Y PROGRAMACIÓN

- **Arreglos numéricos**
- **Funciones internas básicas**
- Indexación múltiple e indexación lineal
- Indexación lógica
- Arreglos estructurados
- Estructura Recursiva: List – Data.Frame - Función
- Optimización de códigos y buenas costumbres
- Principales tipos de error y sus mensajes

vectores

ARREGLOS NUMERICOS

VECTORES

VECTORES - CREACIÓN
C ASSIGN

```
x=3          #En esta expresión R ha creado implícitamente  un vector de longitud 1, y le
             #asigna el valor 3
x
[1] 3        #El '[1]' que precede al valor, indica que se trata del primer elemento y único,
             #en este caso, del vector que se muestra.
```

Creamos un vector con 9 elementos, p.e.,

```
a = c(1,4, 6, 4, 3,5)      # la función c combina los valores indicados en un Vector
```

En lugar del "=" podemos utilizar "<-"

```
b<- c(1,4, 6, 4, 3,5)
```

o bien "->"

```
c(1,4, 6, 4, 3,5) -> c
```

son todas formas de escribir lo mismo

O la función ***assign***

```
assign("w", c(1,4, 6, 4, 3,5) )      #En esta función el nombre de la variable va entre ""
 print (a)
[1] 1 4 6 4 3 5
 print (b)
[1] 1 4 6 4 3 5
 print (c)
[1] 1 4 6 4 3 5
 print (w)
[1] 1 4 6 4 3 5
```

Las cuatro formas son equivalentes, crean el mismo vector

VECTORES - CREACIÓN (cont)

vector

```
v <- vector("integer", 0)      #El tipo de vector va escrito entre "", el numero indica la
                               # cantidad de elementos que quiero que tenga el vector
v                              # Un vector de enteros sin elementos
 integer(0)

w <- vector("numeric", 3)
w                              # Un vector de tres ceros
 [1] 0 0 0

u <- vector("logical", 5)
u                              # Un vector de 5 FALSE. Porque dice FALSO y no VERDADERO?
 [1] FALSE FALSE FALSE FALSE FALSE
```

vector nulo vacío

vector con tres lugares pero que valen cero

vector pero con valores logicos. Tienen dos resultados V o F pero R los traduce matemáticamente a 0 y 1

0 es F
1 es V

El operador ':' permite generar un vector entero a partir de una secuencia creciente o decreciente de enteros, cuyos extremos se indican:

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

v1<-40:13              #secuencia decreciente de números enteros
print(v1);class(v1)    #dos comandos en un renglón

[1] 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23
[19] 22 21 20 19 18 17 16 15 14 13
[1] "integer"
```

aca el valor 22 esta guardado en la posición 19

Lo escribió en dos renglones. El **[19]** indica el índice del primer elemento en ese renglón

VECTORES - CREACIÓN (cont)
OTRA FORMA

```
v<-numeric()          #crea un vector vacio
class(v)              vector numérico vacío
[1] "numeric"

v[3] <- 17            #transforma v en un vector de longitud 3
```
puedo asignar valores a las distintas posiciones
asigno el valor 17 en la posición 3 del vector v

```
v
[1] NA NA 17          #no asigne valores a las 2 primeras componentes, y pone NA
```
porque no asigné nada a las dos primeras posiciones, solo le asigne a la tercera posición. NA porque hay datos ausentes

```
v<- numeric(3)        #como queda definido v ahora?

v[8] <- 213           # que ocurre cuando le cambio la dimension ahora?
v                     # v tiene ahora 8 elementos con espacios vacios: NA
[1] 0 0 0 NA NA NA NA 213
```

CUIDADOsi uso mismo nombre de variable que

```
A<- character()
A[5]<-12
A                     # que ocurre en este caso? Por que?
```
no voy a poder hacer cuenta porque si pido class (A) me lo toma como character.
Para hacer cuentas necesito primero convertirlo

```
m= logical()          creo un vector lógico
class(m)
m[4]=T                a la posición 4 le pone True y a las posiciones anteriores, si estan vacías les pone NA
m                     #como guardo la información?
m[5]=10               # y ahora que ocurrió?  Que paso con la T?
```

VECTORES - CREACIÓN (cont)
paste()

Como ya vimos la función ***paste()*** une todos los vectores de caracteres que se le suministran y construye una sola cadena de caracteres. Muy útil para gráficos.

También admite argumentos numéricos, que convierte inmediatamente en cadenas de caracteres.

En su forma predeterminada, en la cadena de caracteres, cada argumento original se separa del siguiente por un espacio en blanco, aunque ello puede cambiarse utilizando el argumento ***sep= "algo"***.

labs <- paste(c("X","Y"),1:10,sep="") # almacena en labs, X e Y son caracteres , asi que es un vector caracter
que forma tiene labs? Ver en la consola X1, Y2, X3, Y4. Primer caracter con 1 elem, 2 caract con 2 elem, 1 caract con 3 elem

Crear labs1 cambiando el objeto en sep= #ej. Uso el guion - entre comillas

c("X1","Y2","X3","Y4","X5","Y6","X7","Y8","X9","Y10") # genera lo mismo
pero no almacena

no lo guarda en el global environment

VECTORES - CREACIÓN (cont)

seq

```
v <- pi:6                    #Empiezo en PI y voy sumando 1 para obtener el siguiente valor
print(v); class(v)                    # El vector y su clase
[1] 3.142 4.142 5.142                #El limite es 6
[1] "numeric"
```

no termina en el valor exacto al que le dije yo xq, si sigue la secuencia, se pasa

```
zz=seq(1,30,by=3)                    #Es una secuencia de 1 a 30 con paso 3
zz
[1] 1 4 7 10 13 16 19 22 25 28
 class(zz)                    El resultado no es un vector de ENTEROS sino de REALES
[1] "numeric"

v <- seq(from = 4, by = 2, length.out = 8)    Le digo donde empieza, el paso y cuantos valores
print(v)                    # secuencia de 8 números iniciando desde 4 y de 2 en 2
 [1] 4 6 8 10 12 14 16 18
```

length.out=length

Comprobar las siguientes expresiones:

```
pi:1
seq(0,1,length=10)      # ¿qué pasa si en lugar de 10 ponemos 11?
seq(3)                  # equivale a 1:3 y a seq(1,3)
seq(1,5,by=0.5)
seq(from=1,by=0.5, length.out=5)  # que deberia cambiar para obtener el resultado anterior?
-1:1/0
```

cambia el intervalo que separa los numeros

manda la secuencia 1,2,3.
asume que empieza en 1 y que tiene que seguir 3 valores
si no le digo el inicio asume que el inicio es 1

cambia porque ahora tiene un máximo de 5 elementos
si quiero que sea lo mismo que antes tengo q poner legth=9

-Inf NaN Inf porque es -1/0, 0/0 y 1/0

VECTORES - CREACIÓN (cont)

rep

Una función relacionada es ***rep()***, que permite duplicar un objeto de formas diversas. Su forma más sencilla es

rep(x, times=n)

que produce ***n*** copias de x, una tras otra.

rep(1:4,2) toma la secuencia, la hace y despues la repite otra vez

rep(1:4,c(2,3,1,2)) # como genera las copias en este caso? los valores dentro de c me dice como va repitiendo los valores de la secuencia 1 1 2 2 2 3 4 4

rep(1:4,c(2,2)) # error, tamaños no coinciden

sequence

SEQUENCE ES DISTINTO A SEQ

sequence(c(3,2)) #genera secuencias de números. Por default empieza en 1

sequence(c(3, 2), from=2L) #que cambio? en clase no funcionó, F máximo

sequence(c(3, 2), from=2L, by=2L) #y ahora?

VECTORES – MUESTRAS

las de estadística

sample

x: variable
size: tamaño de la muestra
replace: si la muestra vuelve a la población o no

R nos permite seleccionar muestras de un vector utilizando la función sample(x,size,replace=FALSE):
Obtiene una muestra de tamaño size de x, con o sin reemplazo.

```
x<-c(1:10)   podría poner x<- 1:10

sample(x,3)      # toma cualquier muestra con 3 elementos   toma 3 elementos cualquiera de esa poblacion del 1 al 10

sample(x)        # permutaciones de los elementos de x   los 10 numeros de la secuencia pero cambiados de lugar

y<-sample(5:15,5)      # probar que a cada uno le da algo diferente
                       en la variable y me guarda 5 valores random de la secuencia del 5 al 15

PROBAR VARIAS VECES CADA SENTENCIA
```

posicion es distinto al valor guardado dentro de la posición

VECTORES DE INDICES

indice me identifica la posición dentro del vector

R nos permite seleccionar subconjuntos de elementos de un vector añadiendo al nombre del vector un vector índice entre corchetes [].

El vector índice puede ser de tres tipos distintos:

- ✓ Vector lógico
- ✓ Vector de nombres
- ✓ Vector de números enteros

vector lógico: en este caso el vector índice debe ser de la misma longitud que el vector del cual queremos seleccionar elementos. Los valores correspondientes a TRUE en el vector índice son seleccionados, los correspondientes a FALSE omitidos.

me devuelve los valores en las posiciones TRUE y nada en las FALSE

```
x<-c(0,NA,1,2)
y<-x[!is.na(x)]              #¿qué estoy asignando a y?
(x+1)[!is.na(x) & x>0] -> z  # ¿en z que se guarda?
```

posiciones de x que cumplan con el requisito se guardan en y

el ! significa negación

guarda en z a 2 y 3

vector de nombres: esta posibilidad se aplica cuando el objeto tiene el atributo names que identifica sus componentes. La función ***names*** permite añadir etiquetas (o nombres) a un vector numérico

VECTORES DE INDICES
names

R permite dar nombre y acceder por ese nombre a los elementos individuales de un vector

```
frutas <- c(15, 100, 2, 30)      #cantidades de distintas frutas
frutas
[1] 15 100 2 30

names(frutas) <- c("naranja", "pera", "manzana", "durazno")   #asigno nombres a las cantidades
 frutas                          #despliego el vector
naranja    pera manzana durazno
     15     100       2      30

frutas <- c(naranja = 15, pera = 100, manzana = 2, durazno = 30)  #Es equivalente a lo anterior

 frutas[2];frutas["pera"]    #puedo acceder a un elemento por el indice o el nombre
pera
 100
pera
 100
```

aca con comillas

acá sin comillas pero es lo mismo

puedo nombrar la posición del vector con
el índice [numero] o el nombre ["nombre"]

VECTORES DE INDICES

vector de números enteros: en caso de ser positivos, los correspondientes elementos indicados en el índice son concatenados, mientras que si son negativos los valores especificados son omitidos. No hace falta que sea de la misma longitud que el vector de donde seleccionamos.

```
x<-letters[c(13,15,9,18,1)]          # Escriban su nombre!!!
x
[1] "m" "o" "i" "r" "a"
paste(letters[c(13,15,9,18,1)], sep="")
paste(letters[c(13,15,9,18,1)],collapse="")      #collapse es una forma adicional de unir los caracteres.

x[]
x[-(4:5)]                    # ¿qué devuelve ahora? ¿qué le pasó a x?
```

me saca los valores de esas posiciones

letters

Los números en la tabla indican la posición que ocupa cada letra en el vector *letters*

A	1	H	8	O	15	V	22
B	2	I	9	P	16	W	23
C	3	J	10	Q	17	X	24
D	4	K	11	R	18	Y	25
E	5	L	12	S	19	Z	26
F	6	M	13	T	20		
G	7	N	14	U	21		

VECTORES – MODIFICACIONES which

Se puede construir un vector a partir de dos o más vectores ya existentes

```
u <- c(3, 4, 5)
v <- c(5, 4, 3)
w <- c(u, v)              # La concatenación de u y v
print(w)
 [1] 3 4 5 5 4 3
```

Acceso a los elementos individuales de un vector

```
 v <- c(8, 7, -3, 2, 182)
 v[5]
[1] 182

 print(v[1]);print(v[3])    #pido el valor del primer y tercer elemento
[1] 8
[1] -3

v[4]+v[2]                   #sumo el segundo y cuarto elemento
[1] 9
```

Modificación de los elementos de un vector

```
v[1] <- v[2] - v[5]      #guardo en el elemento 1 de v el resultado de la resta de los elementos
v                        # 2 y 5
[1] -175  7 -3  2 182      # el elemento 1 antes era 8 ahora  es -175 que surge de (7-182)
```

La función which() genera un vector numérico con las posiciones seleccionadas.

```
which(v<5)                  # que posiciones cumplen con la condición
which(v!=5)
```

devuelve posiciones que cumplen con la condición

para ver posiciones del vector que cumplen con determinada condición

no te lo guarda como variable, sino que solo queda en la consola, si quiero guardar las variables tengo que asignar las posiciones a una variable

VECTORES
OPERACIONES SENCILLAS

MIEMBRO A MIEMBRO

```
v <- c(2, 3) - c(5, 1)    # RESTA   Resulta en un vector de longitud 2
```
resta miembro a miembro y lo guarda en v

```
v
 [1] -3 2

v <- c(2, 3, 4) * c(2, 1, 3) #PRODUCTO M.A.M     Resulta en un vector de longitud 3
v
[1] 4 3 12
```
si hago producto de vectores con distinta cantidad de elementos,el producto no lo hace.
tiene que haber = cantidad de elementos pero con múltiplos funciona

```
v <- c(2, 3, 4)^(3:1)          # Eleva a potencias 3,2,1
v
[1] 8 9 4
```
si hago potencia de vectores con distinta cantidad de elementos,el producto no lo hace.
tiene que haber = cantidad de elementos pero con múltiplos funciona

```
v <- c(9, 8, 31)                    # Calculo de la RAIZ CUADRADA de cada elemento
sqrt(v)
[1] 3.000 2.828 5.568

angulos <- c(30, 45, 60) * (pi/180)     #Uso de la funcion SENO
angulos                                          # En radianes
[1] 0.5236 0.7854 1.0472
senos <- sin(angulos)
senos
[1] 0.5000 0.7071 0.8660
```

VECTORES (cont)
warn order

¿qué pasa cuando los vectores operandos no son de la misma longitud?

```
v <- c(4, 5, 6, 7, 8, 9, 10) * c(1, 2)
```

Warning: longitud de objeto mayor no es múltiplo de la longitud de uno menor

```
v
 [1] 4 10 6 14 8 18 10
```

Completa la operación **reciclando** los elementos del operador de menor longitud

La operacion anterior es equivalente a:

```
v <- c(4, 5, 6, 7, 8, 9, 10) * c(1, 2, 1, 2, 1, 2, 1)
v
 [1] 4 10 6 14 8 18 10

options(warn = -1)        #este comando elimina el mensaje de advertencia que apareció antes
v <- c(4, 5, 6, 7, 8, 9, 10) * c(1, 2)
v
 [1] 4 10 6 14 8 18 10
```

sort: Ordena un vector en orden ascendente o descendente

sort(x, decreasing = FALSE, ...)

```
a=c(3,7,6,1,8,9,2)
sort(a)              #Por default es ASCENDENTE
[1] 1 2 3 6 7 8 9

sort(a,decreasing=TRUE)
[1] 9 8 7 6 3 2 1
```

NA porque no hubo observación, no lo pusieron por algun motivo, etc

se usa -999 o -777. En pp se puede usar negativo chiquito onda -9 pero en Temp no

En algunos casos las componentes de un objeto pueden no ser completamente conocidas.
Cuando un elemento o valor es ***not available*** (no disponible, faltante) le asignamos el valor especial

NA

En general una ***operación*** con elementos **NA** resulta **NA**

Mediante una opción de la función, podamos omitir o tratar los datos faltantes de forma especial: para que no considere en la operación al dato ausente

na.rm=FALSE o TRUE

FALSE: NO elimina los NA, que da como resultado NA cuando existe al menos un dato faltante.

TRUE: la operación se efectúa con los datos válidos.

EJEMPLO

1) Asignar NA a la variable ***x***

x<-NA

calcular

x+1

Que obtengo como resultado?

Es también un NA.

VECTORES (cont)
sum prod

sum(x) es la suma de todos los elementos del vector x

```
 a
[1] 3 7 6 1 8 9 2
 sum(a)
[1] 36
 b=c(6,5,3,2,NA,8)
 sum(b)
[1] NA              #al tener un NA da como resultado ese valor
 sum(b,na.rm = TRUE)
[1] 24              #no incluye el NA en la suma
```

prod(x) es el producto de todos los elementos del vector x

```
prod(b)
[1] NA
prod(b na.rm = TRUE)
[1] 1440
```

VECTORES (cont)
mean

mean(x) es el promedio de todos los elementos del vector x

Considerando nuestras variables ***a*** y ***b*** definidas para la suma, que resultado obtengo al calcular la media

mean(a)
mean(b)

Es posible mejorar el resultado del promedio de ***b*** para obtener un resultado numérico?

Cual es la instrucción que debo utilizar?

puedo usar na.rm=TRUE para que me pueda calcular el promedio excluyendo el valor NA que teníoa

EJERCITEMOS

para generar vectores con posiciones que tengan NAes por ejemplo hacer una secuencia y a deter

Asignar al vector ***y*** dos NA y dos números.

y <- c(x,3,5,x) donde estan las x es como si estuviera NA

Calcular la suma y el producto teniendo en cuenta los NA's.

sum(y) ; prod (y)

Calcular la suma y el producto SIN tener en cuenta los NA's.

sum(y, na.rm=TRUE)
prod(y, na.rm=TRUE)

EJERCICIOS

```
x<-c(2,5,3,7,1,8,9,4,6)
sum(x)                          # suma todos los elementos
length(x)                       # largo total
sum(x<5)                #cuantos cumplen la condición x<5, vector lógico
x[x<5]                  # quienes cumplen la condición
sum(x[x<5])              # ahora si, los sumo
length(x[x<5])
x*(x<5)                  # el valor de x solo en las posiciones TRUE
sum(x*(x<5))                  # ahora los sumo
which(x<5)                 # que posiciones cumplen con la condición
sum(which(x<5))       # sumo posiciones, NO valores
z<-numeric(10)         # genero otro vector
id<-which(x<5)    # en “id” asigno los valores que cumplen la condición
z[id] <- x[x<5]
z[which(x<5)] <- x[x<5]      # equivalente al anterior (¿mejor?)
```

EJERCICIOS PARA QUE PIENSEN

1) ¿Qué ocurre si hacemos x[1:3] cuando x tiene los datos 2; 1; 0; 3; 6; 1?

2) 1:5 # genera los valores 1 2 3 4 5
   2*1:5 # ¿qué operador "gana"?

3) ¿Pueden predecir el valor de la expresión 1:7*1:2?

4) ¿Qué pasa con esta expresión 1:8*1:2?

EJEMPLO DE RESOLUCION DE PROBLEMA

De un edificio, a una altura de 15 m, se ha lanzado con un ángulo de 50 grados, un proyectil a una velocidad de 7 m/s.
¿Cuáles serán las alturas (coordenadas y) del proyectil a cada 0.5 m de distancia horizontal desde donde se lanzó y hasta los 11 m?

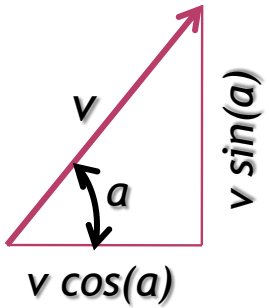
Ecuaciones:

$$x = v_{0x} * t + x_0$$
$$y = -1/2\ g*t^2 + v_{0y}*t + y_0$$

g: aceleración de la gravedad, *t*: tiempo, v_{0x} y v_{0y}: componentes de la velocidad.

La Figura muestra como obtener a partir de la velocidad inicial y el ángulo, las componentes de velocidad.

En R, los argumentos de las funciones trigonométricas deben estar dados en *radianes*

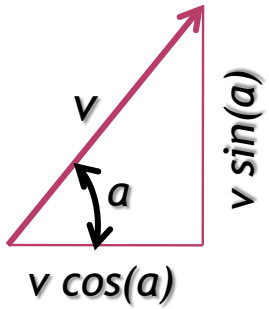


EJEMPLO (cont)

De un edificio, a una altura de 15 m, se ha lanzado con un ángulo de 50 grados, un proyectil a una velocidad de 7 m/s.
¿Cuáles serán las alturas (coordenadas y) del proyectil a cada 0.5 m de distancia horizontal desde donde se lanzó y hasta los 11 m?

Definimos en R nuestra variables iniciales:
Cuales son????

```
g <- 9.81          # aceleración gravedad
x0 <- 0            # x inicial
y0 <- 15           # y inicial
vi <- 7            # velocidad inicial
alphaD <- 50       # ángulo-grados
```



Y las componentes de la velocidad son….

```
alpha <- (pi/180) * alphaD      # ángulo-radianes
vox <- vi * cos(alpha)          # componente x de velocidad inicial
voy <- vi * sin(alpha)          # componente y de velocidad inicial
vox; voy
[1] 4.499513
[1] 5.362311
```

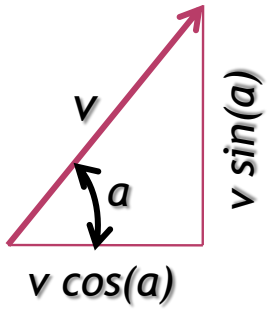
EJEMPLO (cont)

De un edificio, a una altura de 15 m, se ha lanzado con un ángulo de 50 grados, un proyectil a una velocidad de 7 m/s.
¿Cuáles serán las alturas (coordenadas y) del proyectil a cada 0.5 m de distancia horizontal desde donde se lanzó y hasta los 11 m?

Ahora obtenemos las x para las que se desea hacer el cálculo
Cuales son????

desde 0 hasta 11 de 0.5 en 0.5 ***Como lo genero???***

las.x <- seq(from = 0, to = 11, by = 0.5)



Como se relacionan x e y???

$$x = v_{0x} * t + x_0$$
$$y = -1/2\ g*t^2 + v_{0y}*t + y_0$$

*Para cada **x** obtengo el **t** y para ese tiempo obtengo la altura (**y**) a la que se encuentra el proyectil*

$$t = (x - x_0)/v_{0x}$$

```
las.t<-(las.x-x0)/vox
 las.y <- -(g/2) * las.t^2 + voy * las.t + y0
```

EJEMPLO (cont)

De un edificio, a una altura de 15 m, se ha lanzado con un ángulo de 50 grados, un proyectil a una velocidad de 7 m/s.
¿Cuáles serán las alturas (coordenadas y) del proyectil a cada 0.5 m de distancia horizontal desde donde se lanzó y hasta los 11 m?

```
# Los resultados:

las.x
[1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
[12] 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0 10.5
[23] 11.0

las.y
[1] 15.0000 15.5353 15.9495 16.2425 16.4144 16.4652 16.3948
[8] 16.2033 15.8906 15.4568 14.9019 14.2258 13.4286 12.5103
[15] 11.4708 10.3102 9.0285 7.6256 6.1015 4.4564 2.6901
[22] 0.8026 -1.2059
```