Vectores

Victor Lopez

2023-01-11

Tipos de datos en Vectores

Todos los datos deben ser del mismo tipo. De lo contrario los convertira automaticamente, siendo el ultimo el mas fuerte

- logical: TRUE o FALSE
- integer: enteros
- numeric : reales
- complex
- character

Definir un Vector

```
c() # Normal. Podemos pasarle otros vectores y combinara sus elementos

a:b # vector de a hasta b de numeros consecutivos

scan(what = "character") # Escribiendo sus datos en consola. Parametro what
#opcional que leera los datos con ese tipo

rep(a, n) # Vector con dato o vector repetido

seq(a, b, by = d) # Vector de a hasta b con diferencia d

seq(a, b, length.out = n) # Vector de a hasta b dividida en n partes
```

Metodos de vectores

```
fix(x) # Modificar un vector


sapply(vector, FUN = function(element){sqrt(element)}) # Aplica funcion a los elementos de un
#vector. Nos sirve para cuando queremos aplicarle funciones complejas, ya que las funciones que
#vienen por defecto en R, no necesitan de sapply(). Por ejemplo:
```

```r
vectorResultante = vector^2/(vector^2+1) # Devuelve un vector al que se le aplico esas
#operaciones. Se pueden sumar, restar, multiplicar, dividir y potenciar vectores siempre y
#cuando tengan las misma longitud

length(v)

max(v)

min(v)

sum(v)

prod(v)

mean(v)

diff() # Diferencias sucesivas entre los elementos. Al final quedaran n-elementos - 1

cumsum(v) # Sumas acumuladas. Nos sirve para ver el proceso de una sumatoria

sort(v, decreasing = TRUE) # decreasing opcional

rev(v) # Reverse
```

Ejercicio

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} \cdot a^{n-k} \cdot b^k = \binom{n}{0} \cdot a^n \cdot b^0 + \cdots \binom{n}{n} \cdot a^0 \cdot b^n$$

```r
binomioNewton = function(a, b, n){
  cumsum( choose(n, (0:n))*a^(n-(0:n))*b^(0:n) )[n + 1]
}

binomioNewton(2, 1, 2)
```

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

Subvectores

```r
vector[i]
vector[length(vector)]

vector[a:b] # Subvector de a hasta b. Ambos incluidos

vector[-i] # i puede ser otro vector o un numero. Devuelve el vector excepto esos elementos

v[v != 19 & v > 5] # Devuelve un subvector que cumpla con esa condicion. Si para ninguno se
#cumple la condicion, devolvera integer(0), numeric(0), character(0), etc. dependiendo del tipo
```

```r
#de dato que sea el vector

which(v > 4) # Devuelve un subvector con las posiciones donde se cumpla esa condicion

which.min(v) # Devuelve la primera posicion en la que el vector toma su valor minimo

which.max(v)

which(v == min(v))

which(v == max(v))

v[which(v > 4)]

c() # Vector vacio = NULL
```

Los valores NA

```r
v = rep(1, 5)

v[8] = 4 # Al asignarlo a una posicion lejana de las que tiene, rellenara los demas con NA

# Cuando un vector tiene variables NA, al hacer operaciones con el, nos daran NA. Para ello
#usamos un parametro

sum(v, na.rm = TRUE)

# NA no es tipo de dato, por lo tanto no se puede hacer comparaciones con el

is.na(v) # Devuelve el vector con TRUE y FALSE, donde TRUE son NA

v[which(is.na(v))]

# Esto es lo que se suele hacer para trabajar con NA en estadistica descriptiva
v[is.na()] = mean(y, na.rm = TRUE)


# Hay funciones que no tienen el parametro na.rm. Por lo que se hace lo siguiente
cumsum(v[!is.na()])

v_clean = na.omit(v) # Devuelve el vector sin valores NA. Pero tambien devuelve otros atributos
#que no queremos ver y para eso usamos:
attr(v_clean, "na.action")
```