Estructuras de datos en R - Funciones y orden de vectores

Ramon Ceballos

15/1/2021

FUNCIONES DE VECTORES

Cuando queremos aplicar una función a cada uno de los elementos de un vector de datos, la función `sapply` nos ahorra tener que programar con bucles en `R`:

- `sapply(nombre_de_vector,FUN=nombre_de_función)`: para aplicar dicha función a todos los elementos del vector.
- `sqrt(x)`: calcula un nuevo vector con las raíces cuadradas de cada uno de los elementos del vector x.

```
#Guardo la secuencia en una variable
x <- 1:10
#Guardo la función para el cálculo de la raiz cuadrada
raiz_cuadrada <- function(elemento)(sqrt(elemento))
#Aplico sapply para utilizar la función en el vector
sapply(x,FUN=raiz_cuadrada)
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427
##  [9] 3.000000 3.162278
```

Ejemplo

Sacar los 100 primeros términos de una fórmula cualquiera

```
n <- 1:10
#Calcula los 10 primeros terminos de esta sucesion n
x = 2*3^(n/2)
x
```

```
##  [1]   3.464102   6.000000  10.392305  18.000000  31.176915  54.000000
##  [7]  93.530744 162.000000 280.592231 486.000000
```

Medidas estadísticas de un vector

Dado un vector de datos x podemos calcular muchas medidas estadísticas acerca del mismo:

- `length(x)`: calcula la longitud del vector x
- `max(x)`: calcula el máximo del vector x

- `min(x)`: calcula el mínimo del vector x
- `sum(x)`: calcula la suma de las entradas del vector x
- `prod(x)`: calcula el producto de las entradas del vector x
- `mean(x)`: calcula la media aritmética de las entradas del vector x
- `diff(x)`: calcula el vector formado por las diferencias sucesivas entre entradas del vector original x
- `cumsum(x)`: calcula el vector formado por las sumas acumuladas de las entradas del vector original x
  - Permite definir sucesiones descritas mediante sumatorios
  - Cada entrada de `cumsum(x)` es la suma de las entradas de x hasta su posición
- `cummax(x)`: va rellenando el vector dejando el valor más alto
- `cummin(x)`: hace lo inverso de lo anterior
- `cumprod(x)`: calcula el vector formado por los productos acumulados
- `sort(x)`: ordena el vector en orden natural de los objetos que lo forman: el orden numérico creciente, orden alfabético...
- `rev(x)`: invierte el orden de los elementos del vector x

```
cuadrado = function(x){x^2}
v = c(1,2,3,4,5,6)
sapply(v, FUN = cuadrado)
```

```
## [1]  1  4  9 16 25 36
```

```
mean(v)
```

```
## [1] 3.5
```

```
cumsum(v)
```

```
## [1]  1  3  6 10 15 21
```

Ejercicios

- Combinad las dos funciones anteriores, `sort` y `rev` para crear una función que dado un vector x os lo devuelva ordenado en orden decreciente. Razonad si aplicar primero `sort` y luego `rev` a un vector x daría en general el mismo resultado que aplicar primero `rev` y luego `sort`.

```
f_ej1 <- function (x){rev(sort(x))}
vector_ej1 <- c(23, 4, -7, 0, 99)
f_ej1(vector_ej1)
```

```
## [1] 99 23  4  0 -7
```

```
#Al aplicar primero sort() y luego rev() obtenemos el vector ordenado a la inversa
#Al aplicar primero rev() y luego sort() obtendriamos el vector ordenado de menor a mayor
#Por tanto son diferentes resultados
```

- Investigad la documentación de la función `sort` (recordad que podéis usar la sintaxis `?sort` en la consola) para leer si cambiando algún argumento de la misma podéis obtener el mismo resultado que habéis programado en el primer ejercicio.

```r
#Al poner TRUE en decreasing la dunción sort() ordena de mayor a menor
sort(vector_ej1, decreasing = TRUE)
```

```
## [1] 99 23  4  0 -7
```