Vectores

Martin Malo

29/6/2021

```
rep(1997,10)
```

```
##  [1] 1997 1997 1997 1997 1997 1997 1997 1997 1997 1997
```

```
vec <- c(16,0,1,20,1,7,88,5,1,9)
fix(vec)
```

Progresiones y secuencias

```
seq(4, 35, length.out = 7)
```

```
## [1]  4.000000  9.166667 14.333333 19.500000 24.666667 29.833333 35.000000
```

Calcula 7 numeros que puedan satisfacer una secuencia del 4 al 35

```
seq(4, length.out = 7, by = 3)
```

```
## [1]  4  7 10 13 16 19 22
```

Muestra 7 valores que comiencen en 4 aumenten de 3 en 3

```
1:20
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
seq(2, length.out = 20, by = 2)
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
```

```
print(seq(17, 98, length.out = 30), 4)
```

```
##  [1] 17.00 19.79 22.59 25.38 28.17 30.97 33.76 36.55 39.34 42.14 44.93 47.72
## [13] 50.52 53.31 56.10 58.90 61.69 64.48 67.28 70.07 72.86 75.66 78.45 81.24
## [25] 84.03 86.83 89.62 92.41 95.21 98.00
```

```r
x <- c(rep(pi, 5), 5:10, seq(1, 5, length.out = 8))
print(c(1, 33, x, seq(1, length.out = 14, by = 30)), 4)
```

```
##  [1]   1.000  33.000   3.142   3.142   3.142   3.142   3.142   5.000   6.000
## [10]   7.000   8.000   9.000  10.000   1.000   1.571   2.143   2.714   3.286
## [19]   3.857   4.429   5.000   1.000  31.000  61.000  91.000 121.000 151.000
## [28] 181.000 211.000 241.000 271.000 301.000 331.000 361.000 391.000
```

Funciones y orden de vectores

```r
x <- 1:10
sqrt(x)
```

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

```r
sapply(x, FUN = function(elemento){sqrt(elemento)})
```

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

sapply ejecuta funciones propias a vectores. En este ejemplo sapply esta haciendo lo mismo que sqrt pero es util cuando R no tiene cargadas funciones que apliquen a vectores automaticamente o para aplicar funciones propias

```r
cuadrado <- function(x){x^2}
v <- c(1:6)
sapply(v, FUN = cuadrado)

mean(v)
prod(v)
cumsum(v)
cummax(v)
cummin(v)
cumprod(v)

sort(v)
rev(v)

x <- c(4,3,2,6,5,7,8,9,1)
rev(sort(x))
sort(v, decreasing = TRUE)
```

Si yo quisiera elevar al cuadrado el vector v, no podria realizar $cuadrado^v$. Pero si aplico sapply si se me es posible indicando que vector voy a usar y cual es la funcion.

`prod()` va a multiplicar todos los valores del vector `cumsum()` lo que hace es calcular un nuevo vector de un vector dado su acumulado

`sort()` ordena el vector por default de orden ascendente numerico y alfabetico si son palabras

`rev()` invierte el orden del vector dado

`rev(sort())` va a causar que se cree un orden descendente

```
seq(1, 20, length.out = 10)
```

```
##  [1]  1.000000  3.111111  5.222222  7.333333  9.444444 11.555556 13.666667
##  [8] 15.777778 17.888889 20.000000
```

```
diff(seq(1, 20, length.out = 33))
```

```
##  [1] 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375
## [10] 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375
## [19] 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375 0.59375
## [28] 0.59375 0.59375 0.59375 0.59375 0.59375
```

`diff()` crea un vector de las distancias de valor a valor de un vector o secuencias o pregresion dada