Analisis del Binomio de Newton

Julio Cesar Contreras Huerta

septiembre 04, 2021

Funciones Scan y Fix para generar vectores o modificarlos

Ejercicio práctico para ver si sort y rev se aplica en cualquier tiempo

```
VecPrueb <- scan(dec = ",") # Enter al acabar, por si escribo los números con 3,4 == 3.4
VecPrueb1 <- VecPrueb
VecPrueb2 <- VecPrueb

rev(sort(VecPrueb1))
sort(rev(VecPrueb1))
```

Apuntes Subvectores y filtros

```
x = seq(3, 50, by = 3.5)
x[length(x)-1]
```

```
[1] 45
```

```
x[length(x)-2]
```

```
[1] 41.5
```

```
x[-3]
```

```
 [1]  3.0  6.5 13.5 17.0 20.5 24.0 27.5 31.0 34.5 38.0 41.5 45.0 48.5
```

```
x[4:8]
```

```
[1] 13.5 17.0 20.5 24.0 27.5
```

```
x[8:4]
```

```
[1] 27.5 24.0 20.5 17.0 13.5
```

```r
x[seq(2, length(x), by = 2)] # Posicion par
```

```
[1]  6.5 13.5 20.5 27.5 34.5 41.5 48.5
```

```r
x[x%%2==0]# tambien par
```

```
[1] 10 24 38
```

```r
x[x%%2==1] # posicion impar
```

```
[1]  3 17 31 45
```

```r
x[-seq(2, length(x), by = 2)] # eliminar la posicion par
```

```
[1]  3 10 17 24 31 38 45
```

```r
x[(length(x)-3):length(x)] # Los ultimos 4
```

```
[1] 38.0 41.5 45.0 48.5
```

```r
x[x>=10]
```

```
 [1] 10.0 13.5 17.0 20.5 24.0 27.5 31.0 34.5 38.0 41.5 45.0 48.5
```

```r
x[!x<10] # igual al anterior script
```

```
 [1] 10.0 13.5 17.0 20.5 24.0 27.5 31.0 34.5 38.0 41.5 45.0 48.5
```

```r
which(x>4) # posicion
```

```
 [1]  2  3  4  5  6  7  8  9 10 11 12 13 14
```

```r
x[which(x>4)] # me da los valores de posicion
```

```
 [1]  6.5 10.0 13.5 17.0 20.5 24.0 27.5 31.0 34.5 38.0 41.5 45.0 48.5
```

```r
x = c(1,1,3,6,9,2,10,10)
which.min(x) # solo da la primera posicion y es igual a
```

```
[1] 1
```

```r
which(x == min(x))
```

```
[1] 1 2
```

```
x[which.max(x)]
```

```
[1] 10
```

```
x = c() # NULL
x = NULL
y = NULL

vect <- c(x,2,3,y,6)
vect # [1] 2 3 6 valores del vector, no considera la posicion del NULL, eso lo diferencia de los valore
```

```
[1] 2 3 6
```

Factores

```
# Desde un vector partimos para los factores
vect2 <- c(1,23,5,6,9,14, 14, 15, 15, 8, 1, 1)
vect2.factor <- as.factor(vect2)
vect2.factor
```

```
##  [1] 1  23 5  6  9  14 14 15 15 8  1  1
## Levels: 1 5 6 8 9 14 15 23
```

```
vect2.factorr <- factor(vect2)
vect2.factorr # Hasta ahora son lo mismo
```

```
##  [1] 1  23 5  6  9  14 14 15 15 8  1  1
## Levels: 1 5 6 8 9 14 15 23
```

```
vect3 <- c("H", "M", "M", "H", "M", "H", "H")
vect3.factorr <- factor(vect3, levels = c("H", "M", "B")) # Esto no se puede hacer con as.factor

vect4.factorr <- factor(vect3, levels = c("H", "M", "B"), labels = c("Hombre", "Mujer", "Bisexual")) #
levels(vect4.factorr ) <- c("Men", "Women", "Bi")

# Podemos agregar orden en los niveles = levels: comando ordered
fac = factor(c(1,1,1,2,2,3,2,4,1,3,3,4,2,3,4,4),
       levels = c(1,2,3,4), labels = c("Sus","Apr","Not","Exc")) # sin orden
fac
```

```
##  [1] Sus Sus Sus Apr Apr Not Apr Exc Sus Not Not Exc Apr Not Exc Exc
## Levels: Sus Apr Not Exc
```

```
facOrd = ordered(c(1,1,1,2,2,3,2,4,1,3,3,4,2,3,4,4),
       levels = c(1,2,3,4), labels = c("Sus","Apr","Not","Exc")) # con orden
facOrd
```

```
##  [1] Sus Sus Sus Apr Apr Not Apr Exc Sus Not Not Exc Apr Not Exc Exc
## Levels: Sus < Apr < Not < Exc
```