Funciones utiles para datos agrupados

Victor Lopez

2023-01-28

Datos agrupados de ejemplo

```
crabs = read.table("../../data/datacrab.txt", header = TRUE)
cw = cut(crabs$width,
    breaks =  c(20.95, 22.5, 23.55, 24.85, 26.15, 27.45, 28.75, 30.05, 31.35, 32.65, 33.95),
    right = FALSE)
```

Funcion cut

Agrupa vectores en clases

```
cut(datos,
    breaks = c(12, 14, 16, 18, 20),
    labels = c("Doce", "Catorce", "Dieciseis", "Diesiocho", "Veinte"),
    right = FALSE,
    include.lowest = TRUE)

# breaks puede ser un vector formado por los extremos de los intervalos o puede ser un numero k
# de clases. Para este ultimo caso, R divide el intervalo comprendido entre los valores minimos
# y maximos de x en k intervalos y, a continuacion, desplaza ligeramente el extremo inferior del
# primer intervalos a la izquierda y el ultimo a la derecha. Es algo extraño que hace R, lo cual
# es mejor recomendado poner el vector de los extremos.

# labels = FALSE, pone como labels 1, 2, 3, 4... hasta n cantidad de intervalos a los intervalos

# right = FALSE hace que los intervalos sean cerrados por la izquierda y abiertos por la derecha

# include.lowest sirve para indicar si el ultimo intervalo sera cerrado o no
```

Tablas de frecuencias con datos agrupados

Podemos hacerlo con table, prop.table y cumsum. Tambien podemos hacerlo con la funcion hist() de la siguiente manera:

```
crabs = read.table("../../data/datacrab.txt", header = TRUE)

hist(crabs$width, breaks = 3, right = FALSE, plot = FALSE)$counts
```

```
## [1]  46 118   9
```

```
# Devuelve un vector con las fecuencias absolutas de cada intervalos
# Aunque conviene mas igualar el parametro breaks al vector de los extremos del intervalo debido
# que cut y hist hacen uso de diferentes metodos para agrupar los datos cuando se especifica
# solamente el numero k de clases

hist(crabs$width, breaks = 3, right = FALSE, plot = FALSE)$counts
```

```
## [1]  46 118   9
```

```
# Devuelve las marcas de clase
```

Dos funciones para calcular tablas de frecuencias

La primera sirve en el caso en que vayamos a tomar todas las clases de la misma amplitud. Sus parámetros son: x, el vector con los datos cuantitativos; k, el número de clases; A, su amplitud; p, la precisión de los datos (p = 1 si la precisión son unidades, p = 0.1 si la precisión son décimas de unidad...).

```
# La primera sirve en el caso en que vayamos a tomar todas las clases de la misma amplitud. Sus
# parámetros son:
# x, el vector con los datos cuantitativos;
# k, el número de clases;
# A, su amplitud;
# p, la precisión de los datos (p = 1 si la precisión son unidades, p = 0.1 si la precisión son
# décimas de unidad...).

TablaFrecs = function(x,k,A,p){
  L = min(x)-p/2+A*(0:k)
  x_cut = cut(x, breaks = L, right=FALSE)
  intervals = levels(x_cut)
  mc = (L[1]+L[2])/2+A*(0:(k-1))
  Fr.abs = as.vector(table(x_cut))
  Fr.rel = round(Fr.abs/length(x),4)
  Fr.cum.abs = cumsum(Fr.abs)
  Fr.cum.rel = cumsum(Fr.rel)
  tabla = data.frame(intervals, mc, Fr.abs, Fr.cum.abs, Fr.rel, Fr.cum.rel)
  tabla
}

TablaFrecs(crabs$width, 10, 1.3, 0.1)
```

```
##      intervals   mc Fr.abs Fr.cum.abs Fr.rel Fr.cum.rel
## 1  [20.9,22.2) 21.6      2          2 0.0116     0.0116
## 2  [22.2,23.6) 22.9     14         16 0.0809     0.0925
## 3  [23.6,24.9) 24.2     27         43 0.1561     0.2486
## 4  [24.9,26.1) 25.5     44         87 0.2543     0.5029
## 5  [26.1,27.4) 26.8     34        121 0.1965     0.6994
## 6  [27.4,28.8) 28.1     31        152 0.1792     0.8786
## 7    [28.8,30) 29.4     15        167 0.0867     0.9653
## 8    [30,31.4) 30.7      3        170 0.0173     0.9826
```

```
## 9  [31.4,32.6) 32.0      2        172 0.0116     0.9942
## 10   [32.6,34) 33.3      1        173 0.0058     1.0000
```

```r
# La segunda es para cuando conocemos los extremos de las clases. Sus parámetros son:
# x, el vector con los datos cuantitativos;
# L, el vector de extremos de clases;
# V, un valor lógico, que ha de ser TRUE si queremos que el último intervalo sea cerrado, y
# FALSE en caso contrario.

TablaFrecs.L = function(x,L,V){
  x_cut = cut(x, breaks=L, right=FALSE, include.lowest=V, diag.lab = 5)
  intervals = levels(x_cut)
  mc = (L[1:(length(L)-1)]+L[2:length(L)])/2
  Fr.abs = as.vector(table(x_cut))
  Fr.rel = round(Fr.abs/length(x),4)
  Fr.cum.abs = cumsum(Fr.abs)
  Fr.cum.rel = cumsum(Fr.rel)
  tabla = data.frame(intervals, mc, Fr.abs, Fr.cum.abs, Fr.rel, Fr.cum.rel)
  tabla
}

TablaFrecs.L(crabs$width, c(20.95, 22.25, 23.55, 24.85, 26.15, 27.45, 28.75, 30.05, 31.35, 32.65, 33.95)
```

```
##      intervals   mc Fr.abs Fr.cum.abs Fr.rel Fr.cum.rel
## 1  [20.9,22.2) 21.6      2          2 0.0116     0.0116
## 2  [22.2,23.6) 22.9     14         16 0.0809     0.0925
## 3  [23.6,24.9) 24.2     27         43 0.1561     0.2486
## 4  [24.9,26.1) 25.5     44         87 0.2543     0.5029
## 5  [26.1,27.4) 26.8     34        121 0.1965     0.6994
## 6  [27.4,28.8) 28.1     31        152 0.1792     0.8786
## 7  [28.8,30.1) 29.4     15        167 0.0867     0.9653
## 8  [30.1,31.4) 30.7      3        170 0.0173     0.9826
## 9  [31.4,32.6) 32.0      2        172 0.0116     0.9942
## 10   [32.6,34) 33.3      1        173 0.0058     1.0000
```

```r
# Si te das cuenta, la funcion cut, parece que pierde precision a la hora de generar los levels,
# pero no es asi, los intervalos en si siguen estando precisos, pero las etiquetas al cortar los
# datos con cut() se truncan
```