Datos cuantitativos agrupados

Ramon Ceballos

1/2/2021

AGRUPAR DATOS EN R

Al agrupar los datos, lo que hacemos es convertir nuestra variable cuantitativa en un factor cuyos niveles son las clases en que ha sido dividida e identificamos cada dato con su clase.

A la hora de etiquetar los niveles, podemos elegir 3 codificaciones:

- Los intervalos.
- Las marcas de clase (el punto medio de cada intervalo).
- El número de orden de cada intervalo.

1. Agrupación en intervalos. Función cut()

Esta función es la básica en R para agrupar un vector de datos numéricos y codificar sus valores con clases a las que pertenecen.

Su sintaxis básica es:

```
cut(x, breaks=..., labels=..., right=...)
```

1.1. Parámetros de la función cut()

- **`x`** es el vector numérico, nuestra variable cuantitativa
- **`breaks`** puede ser un vector numérico formado por los extremos de los intervalos en los que queremos agrupar nuestros datos y que habremos calculado previamente. También puede ser un número k, en cuyo caso R agrupa los datos en k clases. Para este caso, R divide el intervalo comprendido entre los valores mínimo y máximo de x en k intervalos y, a continuación, desplaza ligeramente el extremo inferior del primer intervalo a la izquierda y el extremo del último, a la derecha.
- **`labels`** es un vector con las etiquetas de los intervalos. Su valor por defecto es utilizar la etiqueta de los mismos intervalos. Si especificamos `labels = FALSE`, obtendremos los intervalos etiquetados por medio de los números naturales correlativos, empezando por 1. Para utilizar como etiqueta las marcas de clase o cualquier otra codificación, hay que entrarlo como valor de este parámetro.
- **`right`** es un parámetro que igualadao a `FALSE` hace que los intervalos que consideremos sean cerrados por la izquierda y abiertos por la derecha. Este no es su valor por defecto.
- **`include.lowest`** igualdo a `TRUE` combinado con `right = FALSE` hace que el último intervalo sea cerrado. Puede sernos útil en algunos casos.

1.2. Resultado de cut()

En cualquier caso, el resultado de la función **cut** es una lista con los elementos del vector original codificados con las etiquetas de las clases a las que pertenecen. Bien puede ser un factor o un vector.

Ejemplo de la función cut()

```
#Cargo dataset de iris y lo guardo
irisdf = iris

#Selecciono una columna (variable) del dataset
petals = iris$Petal.Length

#Especifico nº divisiones para breaks
#Aplico la función cut()
#cerrado a la izda y abierto a la dcha
#Cada dato queda catalagada para cada marca de clase
irisdf$div1 = cut(petals, breaks = 5, right = FALSE)

#Especifico nº divisiones para breaks
#Aplico otro cut() empleando para break la regla de la raíz (13 intervalos)
irisdf$div2 = cut(petals, breaks = ceiling(sqrt(length(petals))), right = FALSE)

#Especifico un vector con los extremos de los intervalos en breaks
irisdf$div3 = cut(petals, breaks = c(1,2,3,4,5,6,7), right = FALSE)

#Igual que irisdf$div1 pero las etiquetas(labels) es FALSE
#Da nºs del 1 al 5 (un nº entero por intervalo)
irisdf$div4 = cut(petals, breaks = 5, right = FALSE, labels = FALSE)

#Le doy nombres a cada intervalo
irisdf$div5 = cut(petals, breaks = 5, right = FALSE,
    labels = c("Peq", "Norm", "Gran", "XGran", "Gigan"))

#El dataset con las variables anteriores añadidas como columnas
str(irisdf)
```

```
## 'data.frame':    150 obs. of  10 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ div1        : Factor w/ 5 levels "[0.994,2.18)",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ div2        : Factor w/ 13 levels "[0.994,1.45)",..: 1 1 1 2 1 2 1 2 1 2 ...
##  $ div3        : Factor w/ 6 levels "[1,2)","[2,3)",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ div4        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ div5        : Factor w/ 5 levels "Peq","Norm","Gran",..: 1 1 1 1 1 1 1 1 1 1 ...
```

2. Frecuencias. Función hist()

Una primera consideración es tratar las clases obtenidas en el paso anterior como los niveles de una variable ordinal y calcular sus frecuencias.

- La frecuencia absoluta de una clase será el número de datos originales que pertenecen a dicha clase.
- La frecuencia absoluta acumulada de una clase será el número de datos que pertenecen a dicha clase o alguna de las anteriores (se suman o heredan).

2.1. Tabla de frecuencias

Normalmente, las frecuencias de un conjunto de datos agrupados se suele representar de la siguiente forma:

Intervalos	X_j	n_j	N_j	f_j	F_j
$[L_1, L_2)$	X_1	n_1	N_1	f_1	F_1
$[L_2, L_3)$	X_2	n_2	N_2	f_2	F_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$[L_k, L_{k+1})$	X_k	n_k	N_k	f_k	F_k

En esta tabla cada columna representa:

- *Intervalos*. Es la etiqueta de cada uno de los intervalos.
- X_j. Es la marca de clase de cada intervalo.
- n_j. Es la frecuencia absoluta.
- N_j. Es la frecuencia absoluta acumulada.
- f_j. Es la frecuencia relativa.
- F_j. Es la frecuencia relativa acumulada.

2.2. Función hist()

El cálculo de las frecuencias con R podemos hacerlo mediante las funciones **`table`, `prop.table` y `cumsum`**.

También podemos utilizar la función **`hist`**, que internamente genera una list cuya componente **`count`** es el vector de frecuencias absolutas de las clases. Por consiguiente, para calcular estas frecuencias, podemos utilizar la sintaxis:

```
hist(x, breaks=..., right=FALSE, plot=FALSE)$count
```

Conviene igualar el parámetro **`breaks`** al vector de los extremos del intervalo debido a que **`cut`** y **`hist`** hacen uso de diferentes métodos para agrupar los datos cuando se especifica solamente el número k de clases.

El resultado de **`hist`** incluye la componente **`mids`** que contiene el vector de puntos medios de los intervalos, es decir, nuestras marcas de clase.

2.3. Tabla de frecuencias con R

Podemos automatizar el cálculo de la ya tan mencionada tabla de frecuencias, utilizando las dos funciones que mostramos a continuación.

PRIMERA FUNCIÓN

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; y p, la precisión de los datos (p = 1 si la precisión son unidades, p = 0.1 si la precisión son décimas de unidad...).

```r
#Primera función
#L refiere a los extremos de los intervalos de las clases a generar
#xcut implementa la división de los datos según L con un cut()
#intervals son los niveles que tiene el aagrupamiento
#mc son las marcas de clase de cada intervalo
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
  }
```

SEGUNDA FUNCIÓN

Por su parte, 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; y V , un valor lógico, que ha de ser `TRUE` si queremos que el último intervalo sea cerrado, y `FALSE` en caso contrario.

```r
#Segunda función
#xcut implementa la división de los datos según L conocido
#intervals son los niveles que tiene el aagrupamiento
#mc son las marcas de clase de cada intervalo
TablaFrecs.L = function(x,L,V){
  x_cut = cut(x, breaks=L, right=FALSE, include.lowest=V)
  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
  }
```

Ejemplo de funciones para las tablas de frecuencias La tabla de frecuencias de la longitud de los pétalos de *Iris* es:

```r
#petals fue definido en el ejemplo de cut()

TablaFrecs(petals, k = 6, A = 1, p = 0.1)
```

```
##     intervals   mc Fr.abs Fr.cum.abs Fr.rel Fr.cum.rel
## 1 [0.95,1.95) 1.45     50         50 0.3333     0.3333
## 2 [1.95,2.95) 2.45      0         50 0.0000     0.3333
## 3 [2.95,3.95) 3.45     11         61 0.0733     0.4066
## 4 [3.95,4.95) 4.45     43        104 0.2867     0.6933
## 5 [4.95,5.95) 5.45     35        139 0.2333     0.9266
```

```
## 6 [5.95,6.95) 6.45     11        150 0.0733     0.9999
```

```
TablaFrecs.L(petals, L = c(1,3,4,5,5.5,6,6.5,7), V = FALSE)
```

```
##   intervals   mc Fr.abs Fr.cum.abs Fr.rel Fr.cum.rel
## 1     [1,3) 2.00     50         50 0.3333     0.3333
## 2     [3,4) 3.50     11         61 0.0733     0.4066
## 3     [4,5) 4.50     43        104 0.2867     0.6933
## 4   [5,5.5) 5.25     18        122 0.1200     0.8133
## 5   [5.5,6) 5.75     17        139 0.1133     0.9266
## 6   [6,6.5) 6.25      7        146 0.0467     0.9733
## 7   [6.5,7) 6.75      4        150 0.0267     1.0000
```

Ejemplo completo Vamos a considerar el conjunto de datos de `datacrab`. Para nuestro estudio, trabajaremos únicamente con la variable `width`.

En primer lugar, cargamos los datos en un data frame:

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

cw = crabs$width
```

Siguiendo con el ejemplo de las anchuras de los cangrejos, vamos a calcular sus tablas de frecuencias haciendo uso de todo lo aprendido anteriormente.

```
A = 1.3

L1 = min (cw) - 1/ 2*0.1
L = L1 + A*(0:10)
L
```

```
##  [1] 20.95 22.25 23.55 24.85 26.15 27.45 28.75 30.05 31.35 32.65 33.95
```

Lo haremos con las funciones que os hemos proporcionado:

```
TablaFrecs(cw,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
TablaFrecs.L(cw,L,FALSE)
```

```
##      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
```

Fijaos que los intervalos no terminan de ser los que hemos calculado nosotros (L), pero eso se debe a como funciona la función `cut`.