Tablas de Datos Cualitativos

Victor Lopez

2023-01-17

Vector de elementos random

```
respuestas = factor( sample(c("si", "no"), size = 12, replace = TRUE) ) # El replace sirve para
# sacar y volver a meter lo obtenido de manera random. sample() devuelve un factor
```

Tabla de frecuencias absolutas (Unidimensionales)

```
tabla = table(respuestas) # Tabla de frecuencias de un vector o un factor
names(tabla) # Niveles de la tabla de frecuencias

# Podemos añadir names a la hora de crear el factor:
x = factor( sample(c("si", "no"), size = 12, replace = TRUE), levels = c("si", "no", "tal vez") )

# Acceder a sus columnas (Devuelven otra tabla con la parte seleccionada):
tabla[3] # Columna 3
tabla["tal vez"] # La misma columna. Si no existen, devuelven NA

# Las frecuencias las podemos tratar como si fuese un vector para operaciones:
3 * tabla

# which
names(which(tabla == n)) # Obtenemos el nivel donde la frecuencia absoluta es n
names(which(tabla == max(tabla))) # Moda
mode = function(factor){ names(which(table(factor) == max(table(factor))))} # Moda de un factor
```

Tabla de frecuencias relativas (Unidimensionales)

```
# Unidimensional
prop.table(tabla) == tabla / length(suFactor) # Se le aplica a una tabla de frecuencias
# absolutas ya creada
```

Tablas de frecuencias absolutas (bidimensionales)

```r
sexo = c("H", "M", "H", "H", "M")
respuestas = ("si", "no", "si", "no", "no")

tablaBi = tables(sexo, respuestas) # La primera sera la fila y la segunda la columna. Podriamos hacer t

# Acceder a las entradas (igual que en matrices)
tablaBi[1, 2] == tabla["H", "no"]

colSums(tablaBi)
rowSums(tablaBi)

apply(tablaBi, FUN = sum, MARGIN = 1) # Sum toma como parametro vectores (Filas)
apply(tablaBi, FUN = sum, MARGIN = 2)
apply(tablaBi, FUN = sqrt, MARGIN = c(1, 2)) # A cada elemento
```

Tablas de frecuencias relativas (bidimensionales)

```r
# Frecuencias relativas globales
prop.table(tablaBi)

# Frecuencias relativas marginales
prop.table(tablaBi, margin = 1) # Lo haremos con base al sexo
prop.table(tablaBi, margin = 1) # Lo haremos con base a las respuestas
# El margin depende de como hayamos puesto las variables al crear la table
# O tambien podriamos interpretarlo como, calcular la fr de cada uno de los niveles de la
# variable seleccionada por el margin, las cuales estan representadas o divididas por los niveles
# de la otra variable

# Funcion CrossTable() del paquete gmodels
# Hace un resumen de la fa, fr global y las dos fr marginales en tablas bidimensionales
library(gmodels)
CrossTable(sexos, respuestas, prop.chisq = FALSE)

colSums(prop.table(tablaBi))
rowSums(prop.table(tablaBi))

apply(prop.table(tablaBi), FUN = sum, MARGIN = 1) # Sum toma como parametro vectores (Filas)
apply(prop.table(tablaBi), FUN = sum, MARGIN = 2)
apply(prop.table(tablaBi), FUN = sqrt, MARGIN = c(1, 2)) # A cada elemento
```

Tablas de frecuencias absolutas y relativas (Multidimensionales)

```r
# FRECUENCIAS ABSOLUTAS

ans = sample( c("Si", "No"), size = 100, replace = TRUE )
```

```r
sex = sample( c("H", "M"), size = 100, replace = TRUE )
place = sample( c("San Francisco", "Barcelona", "Valencia", "Cobija"), size = 100, replace = TRUE)

tablaMulti = table(sex, ans, place) # Fila: sex, columma: ans, tablas organizadas con base al
# place

# ftable()
# Al paser 3 variables a table, si en vez de usar table, usamos ftable(), lo que hara no es
# dividir en tablas la tercer variable, lo que hara es poner el sex y ans como filas y el place
# como columnas
ftable(sex, ans, place)
```

```
##         place Barcelona Cobija San Francisco Valencia
## sex ans
## H   No                5      6             4        7
##     Si                7      7             5        4
## M   No                4      7            11        7
##     Si                4      8             6        8
```

```r
ftable(sex, ans, place, col.vars = c("sex", "ans")) # Podemos elegir las columnas, las demas
```

```
##               sex  H     M
##               ans No Si No Si
## place
## Barcelona          5  7  4  4
## Cobija             6  7  7  8
## San Francisco      4  5 11  6
## Valencia           7  4  7  8
```

```r
# seran filas. En el orden que la pongamos, sera de arriba hacia abajo

# Filtrar
tablaMulti["M", "Si", "San Francisco"]
```

```
## [1] 6
```

```r
tablaMulti[ , "Si", "Valencia"]
```

```
## H M
## 4 8
```

```r
tablaMulti[ , "No", ]
```

```
##    place
## sex Barcelona Cobija San Francisco Valencia
##   H         5      6             4        7
##   M         4      7            11        7
```

```r
tablaMulti["M", , "Cobija"]
```

```
## No Si
##  7  8
```

```r
# FRECUENCIAS RELATIVAS

prop.table(tablaMulti) # Frecuencias globales
```

```
## , , place = Barcelona
##
##    ans
## sex   No   Si
##   H 0.05 0.07
##   M 0.04 0.04
##
## , , place = Cobija
##
##    ans
## sex   No   Si
##   H 0.06 0.07
##   M 0.07 0.08
##
## , , place = San Francisco
##
##    ans
## sex   No   Si
##   H 0.04 0.05
##   M 0.11 0.06
##
## , , place = Valencia
##
##    ans
## sex   No   Si
##   H 0.07 0.04
##   M 0.07 0.08
```

```r
prop.table(tablaMulti, margin = 3) # Calcularemos las frecuencias relativas de los datos dentro
```

```
## , , place = Barcelona
##
##    ans
## sex        No        Si
##   H 0.2500000 0.3500000
##   M 0.2000000 0.2000000
##
## , , place = Cobija
##
##    ans
## sex        No        Si
##   H 0.2142857 0.2500000
##   M 0.2500000 0.2857143
```

```
##
## , , place = San Francisco
##
##    ans
## sex        No        Si
##   H 0.1538462 0.1923077
##   M 0.4230769 0.2307692
##
## , , place = Valencia
##
##    ans
## sex        No        Si
##   H 0.2692308 0.1538462
##   M 0.2692308 0.3076923
```

```
# de cada place

prop.table(tablaMulti, margin = 1) # Calcularemos las frecuencias relativas de los datos para
```

```
## , , place = Barcelona
##
##    ans
## sex         No         Si
##   H 0.11111111 0.15555556
##   M 0.07272727 0.07272727
##
## , , place = Cobija
##
##    ans
## sex         No         Si
##   H 0.13333333 0.15555556
##   M 0.12727273 0.14545455
##
## , , place = San Francisco
##
##    ans
## sex         No         Si
##   H 0.08888889 0.11111111
##   M 0.20000000 0.10909091
##
## , , place = Valencia
##
##    ans
## sex         No         Si
##   H 0.15555556 0.08888889
##   M 0.12727273 0.14545455
```

```
# cada sexo, incluyendo todos los datos de todos los paises

prop.table(tablaMulti, margin = c(1, 3)) # Lo mismo que lo anterior, pero para cada pais
```

```
## , , place = Barcelona
##
```

```
##    ans
## sex        No        Si
##   H 0.4166667 0.5833333
##   M 0.5000000 0.5000000
##
## , , place = Cobija
##
##    ans
## sex        No        Si
##   H 0.4615385 0.5384615
##   M 0.4666667 0.5333333
##
## , , place = San Francisco
##
##    ans
## sex        No        Si
##   H 0.4444444 0.5555556
##   M 0.6470588 0.3529412
##
## , , place = Valencia
##
##    ans
## sex        No        Si
##   H 0.6363636 0.3636364
##   M 0.4666667 0.5333333
```

```r
# individualmente

# Podriamos hacer el mismo uso de ftable():
ftable(prop.table(tablaMulti, margin = 1))
```

```
##         place  Barcelona     Cobija San Francisco   Valencia
## sex ans
## H   No        0.11111111 0.13333333    0.08888889 0.15555556
##     Si        0.15555556 0.15555556    0.11111111 0.08888889
## M   No        0.07272727 0.12727273    0.20000000 0.12727273
##     Si        0.07272727 0.14545455    0.10909091 0.14545455
```

Tabla de fa HairEyeColor

```r
HairEyeColor # Por defecto viene ordenada como table(Hair, Eye, Sex)
sum(HairEyeColor) # Total de datos

# Permutar una tabla ya creada:
aperm(HairEyeColor, perm = c("Sex", "Hair", "Eye"))

library(kableExtra)
kable(HairEyeColor) # Muestra la tabla como si fuera un dataframe

library(xtable)
```

```r
xtable(table(sex, ans)) # Imprime una tabla con LaTeX, para poder presentarla correctamente
# tendriamos que utilizar el parametro results = "asis" al chunk
```

Tablas con dataframes

```r
Beb_Energ = read.table("../../data/EnergyDrink")

# Para un df con variables cualitativas, podemos hacer un resumen de sus frecuencias absolutas:
summary(Beb_Energ)
```

```
##    estudio              bebe              sexo
##  Length:122         Length:122         Length:122
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```r
# Tablas de cada variable
apply(Beb_Energ, MARGIN = 2, FUN = table ) # Se convierte en una lista con las tablas de cada v
```

```
## $estudio
##
## Industriales  Informatica        Mates   Telematica
##           37           53           16           16
##
## $bebe
##
## No Si
## 97 25
##
## $sexo
##
## Hombre  Mujer
##     83     39
```

```r
# Lo anterior equivaldria a:
table(Beb_Energ$sexo)
```

```
##
## Hombre  Mujer
##     83     39
```

```r
# Si hacemos un table() del dataframe, devuelve la table con las variables como parametros en el
# mismo orden en que estan en el df

# No necesariamente debemos seleccionar todas las columnas, asi que:

table(Beb_Energ[c(1, 3)]) # Solo seleccionamos estudio y sexo
```

```
##               sexo
## estudio        Hombre Mujer
##   Industriales     25    12
##   Informatica      37    16
##   Mates             9     7
##   Telematica       12     4
```

```
# Es lo mismo con:
ftable(Beb_Energ) # Sin necesidad de convertirlo a table() o prop.table()
```

```
##                   sexo Hombre Mujer
## estudio      bebe
## Industriales No            19    10
##              Si             6     2
## Informatica  No            30    11
##              Si             7     5
## Mates        No             8     6
##              Si             1     1
## Telematica   No            10     3
##              Si             2     1
```