Estadística con R

Mariano Rico

19 03 2021

Esto es un cuaderno R Markdown. Cuando ejecutas código dentro del cuaderno, el resultado (números, gráficas, tablas, etc.) aparece tras el código.

Números aleatorios y semillas

Vamos a generar números aleatorios. Usaremos la función `runif` (r de "random" y unif de uniforme), porque usa la distribución uniforme (valores equiprobables).

```
runif(6, 0, 1)
```

```
## [1] 0.36152786 0.08204078 0.13319050 0.83245503 0.41966410 0.82545237
```

Este comando saca 6 valores entre 0 y 1. Si quiero que la próxima vez que ejecute un programa, genere los mismos valores, tendré que asignar una semilla con `set.seed(entero)`.

```
set.seed(666)
runif(6, 0, 1)
```

```
## [1] 0.7743685 0.1972242 0.9780138 0.2013274 0.3612444 0.7426119
```

```
runif(6, 0, 1)
```

```
## [1] 0.97872844 0.49811371 0.01331584 0.25994613 0.77589308 0.01637905
```

```
set.seed(666)
runif(6, 0, 1)
```

```
## [1] 0.7743685 0.1972242 0.9780138 0.2013274 0.3612444 0.7426119
```

```
runif(6, 0, 1)
```

```
## [1] 0.97872844 0.49811371 0.01331584 0.25994613 0.77589308 0.01637905
```

Como hemos vuelto a poner la misma semilla, la secuecia de números aleatorios vuelve a ser la misma de antes.

Generando números de una normal

Para generar números que siguen una distribución normal, usaremos `rnorm`:

```r
rnorm(6, 0, 1)
```

```
## [1] -1.30618526 -0.80251957 -1.79224083 -0.04203245  2.15004262 -1.77023084
```

Esto ha generado 6 números aleatorios de una distribución normal de media 0 y sd (standard deviation, desviación estándar) 1.

El caso del precio del zumo de naranja en Madrid

Generamos 6 precios a partir de una distribución normal de media 2 y sd 1.

```r
rnorm(6, 2, 1)
```

```
## [1] 2.8646536 0.2798441 2.1341257 1.9241734 2.8583005 2.3449003
```

Podemos calcular el valor medio (la media) de una "muestra" de 6 elementos como esos con

```r
mean(rnorm(6, 2, 1))
```

```
## [1] 1.880812
```

Podemos hacer tantas muestras de 6 elementos como queramos con `replicate`.

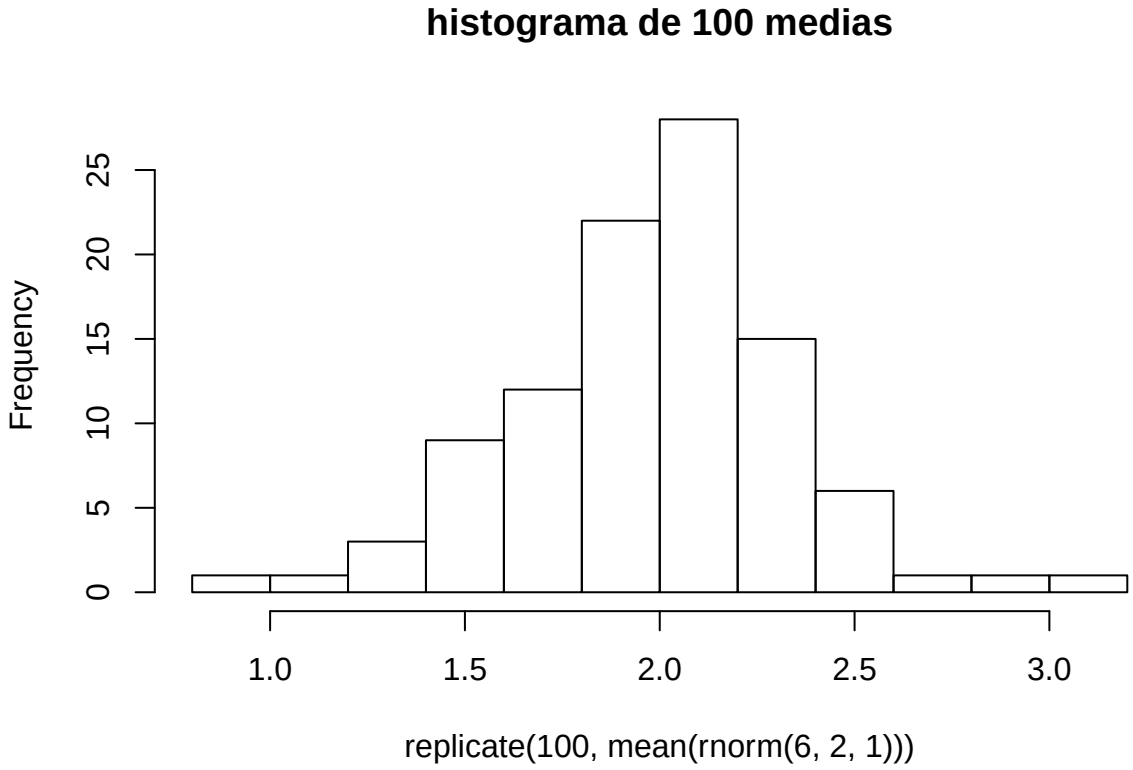
```r
replicate(100, mean(rnorm(6, 2, 1)))
```

```
##   [1] 0.8753644 2.2353774 2.2371558 2.0905670 2.1818414 1.9446008 1.8320137
##   [8] 2.2841929 1.3120666 1.9307803 2.2704014 1.7654860 1.6166722 2.3770050
##  [15] 1.4967075 1.3927091 2.5076108 2.0743534 1.1459163 2.5184078 2.3363672
##  [22] 1.2265743 2.5272391 2.1545171 2.1006188 1.5859256 1.7442877 1.8566639
##  [29] 1.7975727 2.8036079 2.8998893 2.1233903 1.6851609 2.3878657 2.2423384
##  [36] 1.5380950 1.5382618 2.1034196 1.9706647 1.9336285 1.6912853 0.9508627
##  [43] 2.5149773 1.2661285 1.4286055 1.5117136 1.6075237 2.1698490 1.9581477
##  [50] 1.0763669 2.5176864 2.4600711 1.5947416 2.8150199 1.7018612 2.0441706
##  [57] 2.0105200 2.1090720 1.7431730 1.4088885 1.7952565 2.0844812 2.2365794
##  [64] 1.9420660 2.2309571 2.4679729 2.1554970 2.1119480 1.4856972 2.0150080
##  [71] 2.2779606 1.6193314 2.6465285 1.9961196 1.5303928 1.5620395 2.2087418
##  [78] 2.2790842 2.0401410 2.6932398 2.1689997 1.6997711 1.8271775 1.8894654
##  [85] 2.4870661 1.8472894 1.4231288 2.6431628 1.5018913 2.1215007 2.1807883
##  [92] 2.1054757 2.0154856 2.7850252 1.6290772 1.8336814 2.1453800 2.5996297
##  [99] 1.6541515 2.1208859
```

En este caso hemos tomado 100 muestras de 6 bares elegidos aleatoriamente (de una población infinita), y de cada muestra mostramos su media. Observa que hemos obtenido un vector de valores.

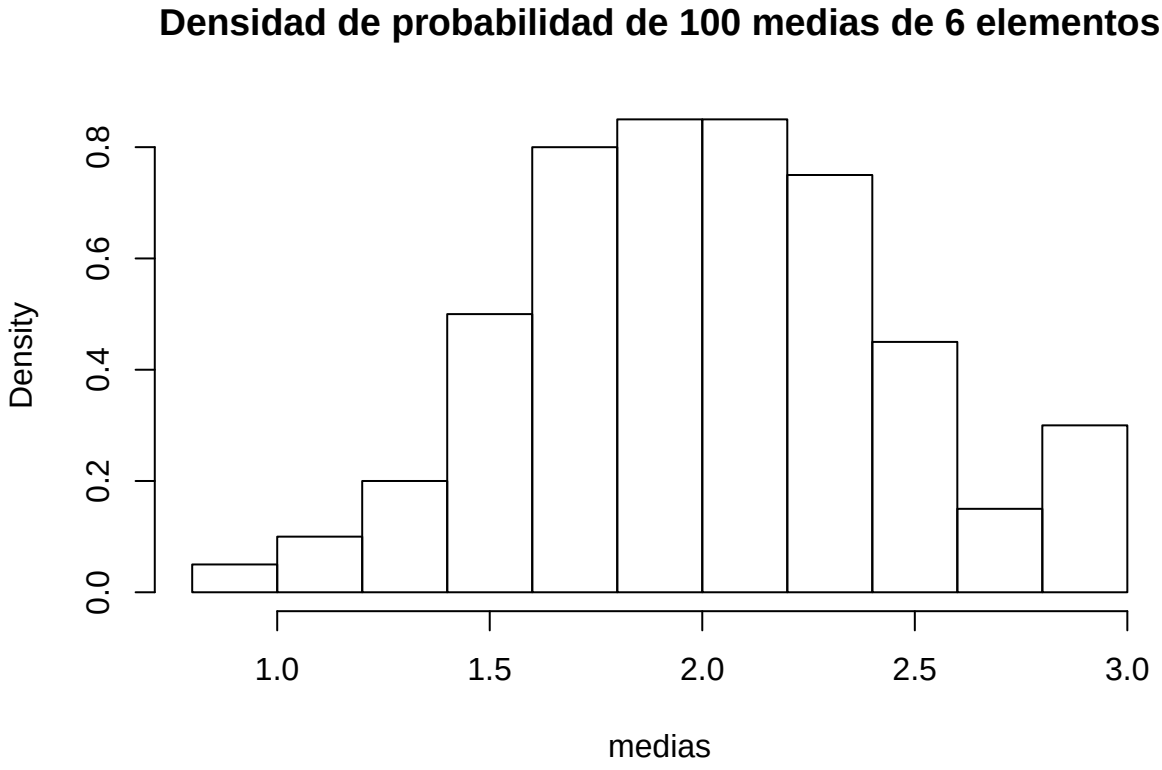
Cualquier vector de valores se puede mostrar como un histograma, con la función `hist`.

```r
hist(replicate(100, mean(rnorm(6, 2, 1))), main = "histograma de 100 medias")
```



Pero también podemos sacar su densidad de probabilidad con el parámetro `freq = FALSE` en la función `hist`.

```
medias <- replicate(100, mean(rnorm(6, 2, 1)))
hist(medias,
     freq = FALSE,
     main = "Densidad de probabilidad de 100 medias de 6 elementos")
```



La desviación típica es:

```
sd(medias)
```

```
## [1] 0.43718
```

Lo podemos redondear a dos decimales con:

```
round(sd(medias), digits = 2)
```

```
## [1] 0.44
```

Podemos ver cómo converge la media de las muestras usando la función `cummean` (cum de "cummulative" (acumulativo) y mean de "medias"). Esta función no está en el "R base", sino en la librería `dplyr`. Se entiende bien con un ejemplo sencillo:

```
library(dplyr)
x <- c(1, 3, 5, 2, 2)
cummean(x)
```

```
## [1] 1.00 2.00 3.00 2.75 2.60
```

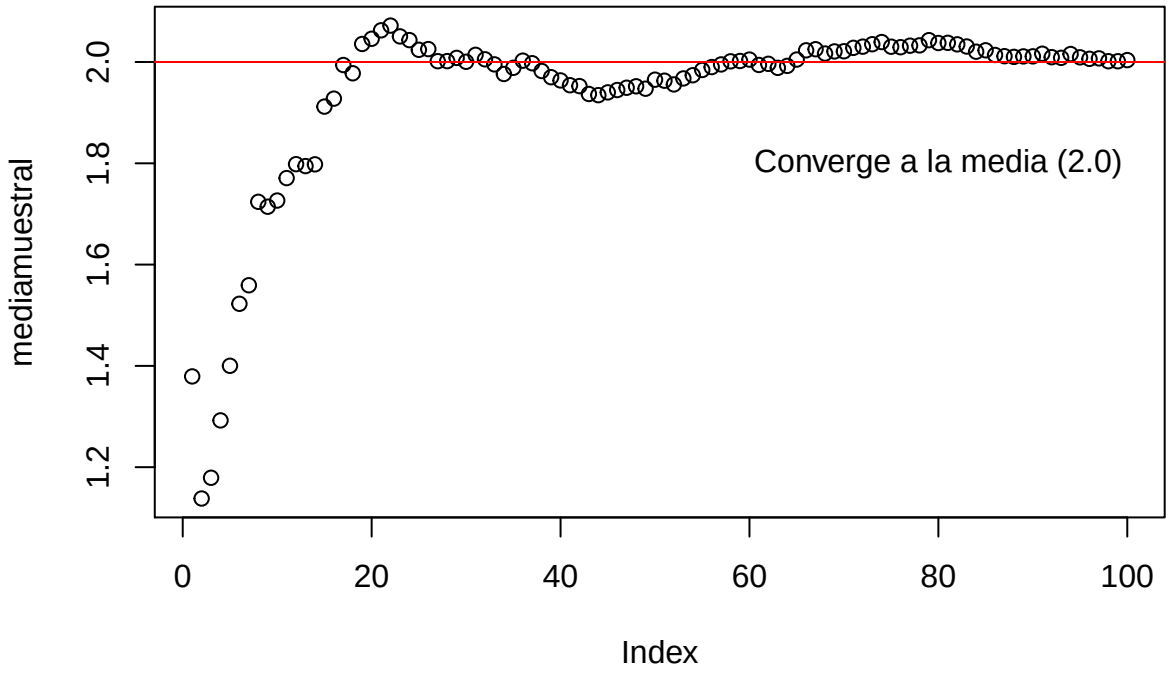
Nota: si no quieres usar la función `cummean` puedes usar `cumsum` (la suma acumulada), pero tienes que dividir cada valor por la posición en el vector.

```r
x <- c(1, 3, 5, 2, 2)
cumsum(x)/1:length(x)
```

```
## [1] 1.00 2.00 3.00 2.75 2.60
```

La función **cummean** devuelve un vector del mismo tamaño, en el que cada elemento es la media de los anteriores. Es, por tanto, la media acumulada. En nuestro caso podemos dibujar en una gráfica esa media acumulada de forma sencilla:

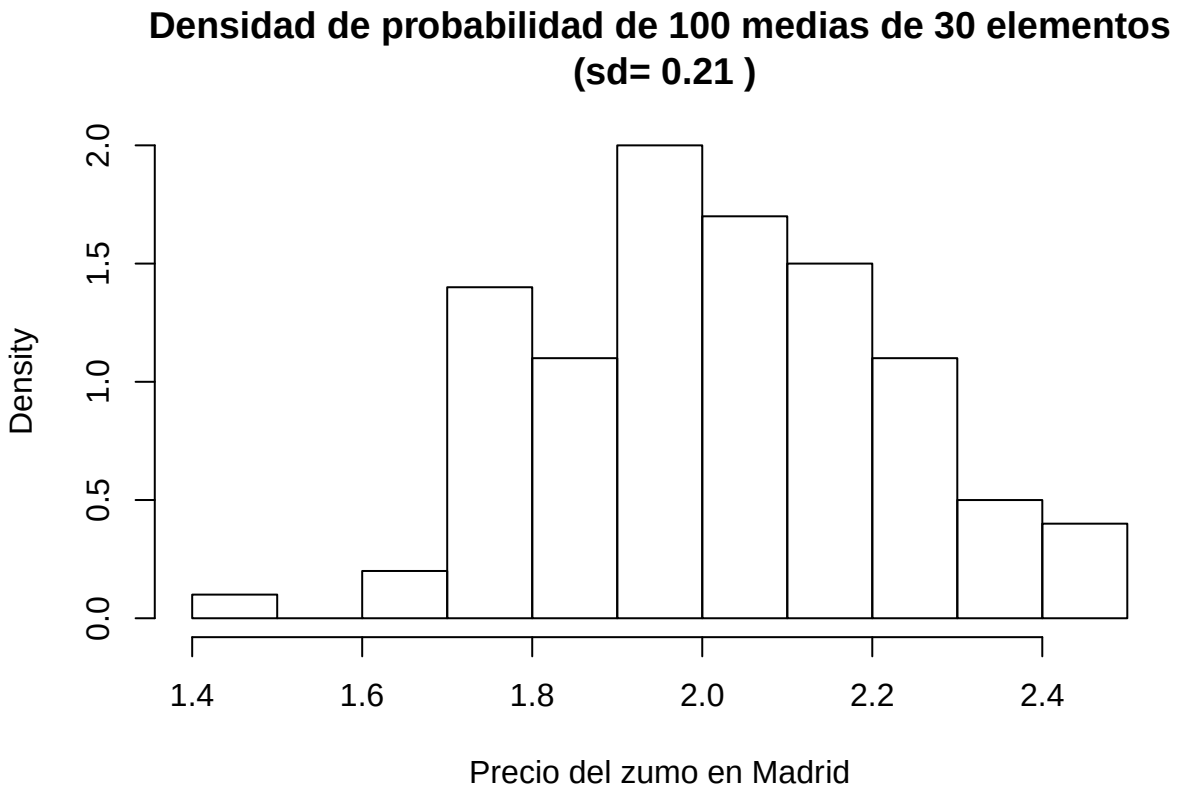
```r
library(dplyr)
set.seed(335)
mediamuestral <- cummean(replicate(100, mean(rnorm(6, 2, 1))))
plot(mediamuestral)
abline(h=2, col="red")
text(80, 1.8, "Converge a la media (2.0)")
```



Esto es resultado de la ley de los grandes números.

Podemos ver el efecto de aumentar el número de elementos de cada muestra usando 30 elementos en lugar de los 6 de antes

```r
set.seed(335)
mediamuestral <-replicate(100, mean(rnorm(30, 2, 1)))
hist(mediamuestral,
     freq = FALSE,
     main = paste('Densidad de probabilidad de 100 medias de 30 elementos\n',
                  '(sd=', round(sd(mediamuestral), digits=2), ')'
                  ),
     xlab = "Precio del zumo en Madrid"
    )
```

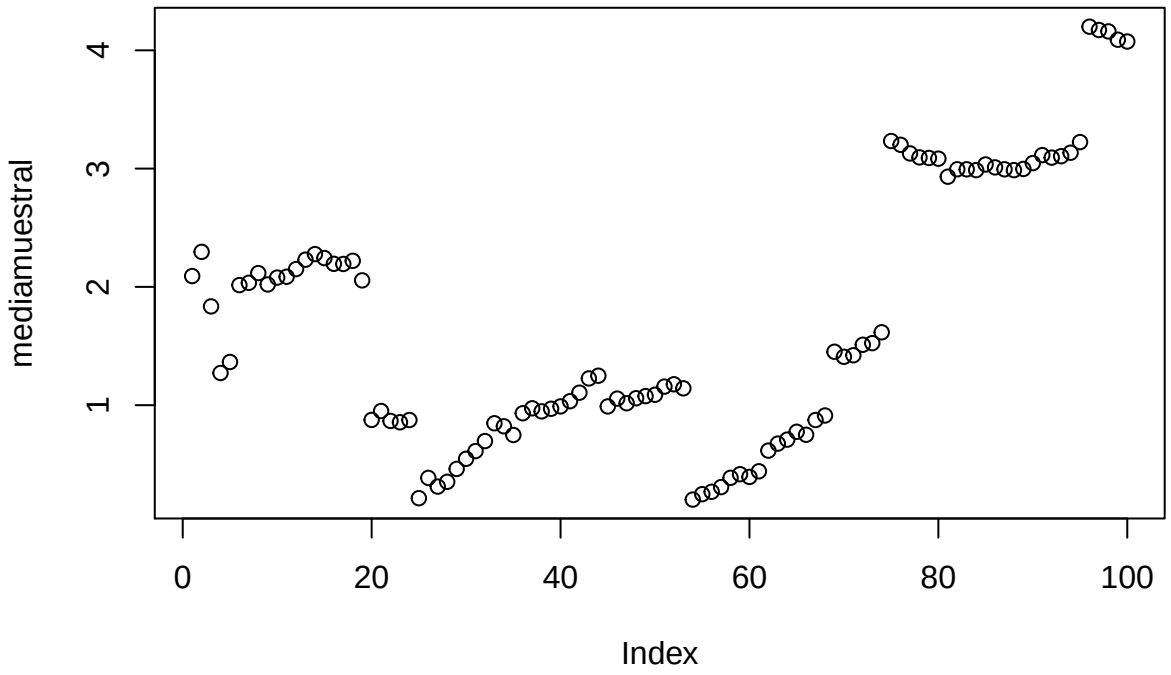


Observa que el rango sobre el que se muestra la gráfica (sd=0.21) es más estrecho que antes (sd=0.44). Este es el resultado del teorema central del límite. Bueno, y que la curva es una normal, sea cual sea la distribución de la población que se está muestrando.

El monstruo de Cauchy

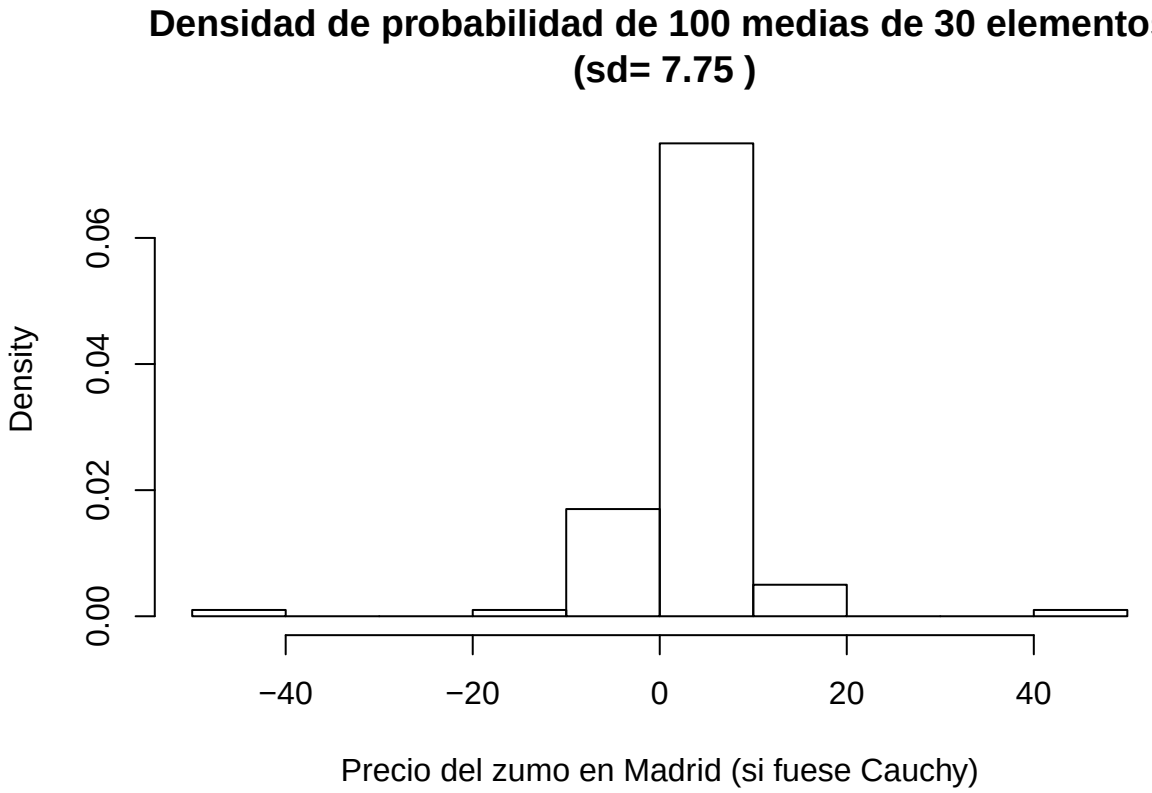
Si usamos la distribución de Cauchy, tendremos "desastres" que hacen que la media muestral no converja.

```
library(dplyr)
set.seed(328)
mediamuestral <- cummean(replicate(100, mean(rcauchy(6, location = 2, scale = 1))))
plot(mediamuestral)
```



Como curiosidad, la densidad de la media muestral anterior (población con distribución de Cauchy con location = 2 y scale = 1) es así:

```
set.seed(335)
mediamuestral <-replicate(100, mean(rcauchy(30, 2, 1)))
hist(mediamuestral,
     freq = FALSE,
     main = paste('Densidad de probabilidad de 100 medias de 30 elementos\n',
                  '(sd=', round(sd(mediamuestral), digits=2), ')'
                  ),
      xlab = "Precio del zumo en Madrid (si fuese Cauchy)"
    )
```


Densidad de probabilidad de 100 medias de 30 elementos
(sd= 7.75)
Density
0.00
0.02
0.04
0.06
−40
−20
0
20
40
Precio del zumo en Madrid (si fuese Cauchy)