Muestra y población, conceptos en R

Sesión 6

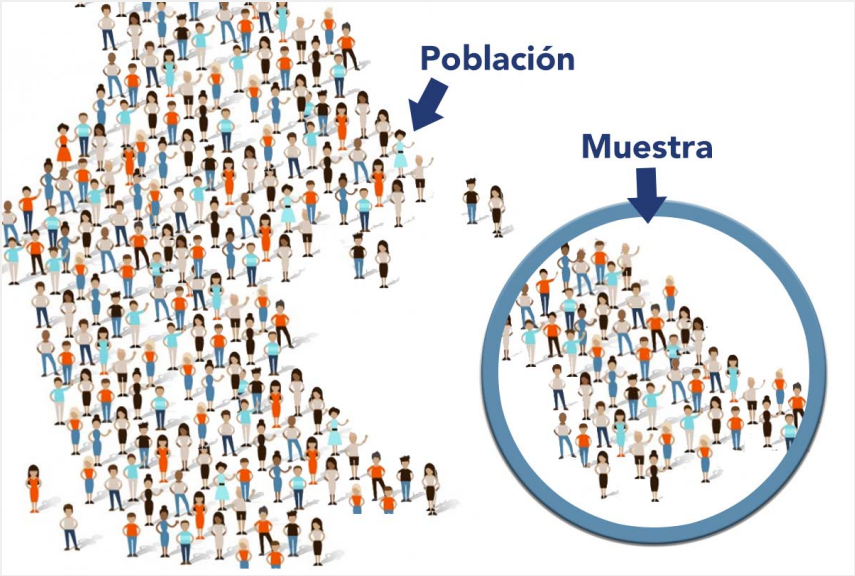
Natalie Julian - www.nataliejulian.com

Estadística UC y Data Scientist en Zippedi Inc.

Quiero conocer a mi población...

Muchas veces, queremos obtener inferencias sobre una población en particular. Sin embargo, muy rara vez es posible obtener información de todas las unidades que componen dicha población. Aquella fracción de nuestra población a la que podemos estudiar la denominamos **muestra**.

Muestras: Un Zoom en mi población



Ejemplo

El archivo `Born` contiene las respuestas de una encuesta sobre natalidad realizada el año 1988. La encuesta fue respondida por mujeres y se posee diversa información, el análisis que más interesa realizar es sobre la cantidad de hijs *children* de las encuestadas y cómo se distribuye.

Preliminares

Cargamos los datos en R y podemos ver cuántos registros se poseen:

```
library(rio)
born<-import(file.choose())

dim(born)
[1] 4361   27
```

Es decir, se poseen 4361 registros y 27 características medidas a cada registro.

Una muestra de la población

Suponga que nuestra población de interés son todas las mujeres encuestadas, es decir, las 4361 mujeres.

Pero por presupuesto solamente pudimos estudiar a $n = 10$ mujeres elegidas de forma aleatoria de entre las participantes de la encuesta. Podemos obtener nuestra muestra aleatoria de la siguiente forma:

```
set.seed(2020) #Semila de aleatoriedad
(muestra<-sample(1:nrow(born), 10, replace = FALSE))
 [1] 1436 1260  728  273 1188  170  945 3128 2248 2602
```

Semilla de aleatoriedad: Se fija una semilla para generar los mismos valores "aleatorios".

Sólo tenemos información de la muestra

Si queremos estudiar la natalidad tendríamos que utilizar los datos que tenemos, es decir, sólo los de nuestra muestra:

```
summary(born$children[muestra]) #Muestra
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     0.0     0.5     0.7     1.0     2.0
```

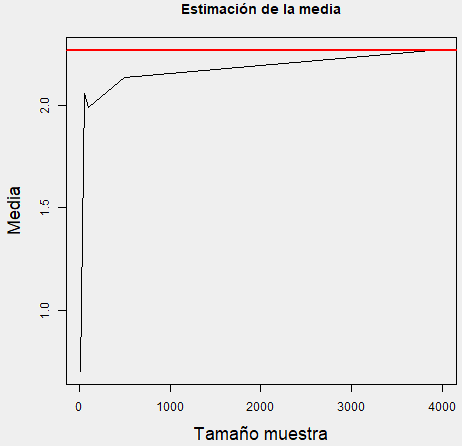
¿Pero qué pasa si comparamos con la población?

```
summary(born$children[muestra]) #Muestra
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     0.0     0.5     0.7     1.0     2.0

 summary(born$children) #Poblacion
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     0.0     2.0     2.3     4.0    13.0
```

¡Las estadíticas no coinciden! La media de la cantidad de hijs en la muestra es de 0.7, pero en la población es de 2.3! **¿Teorías?**

¿Y si tomamos muestras más grandes?

```
x<-c(10, 50, 100, 500, 4000)
y<-c(mean(born$children[muestra10]),mean(born$children[muestra50]),
mean(born$children[muestra100]),mean(born$children[muestra500]),
mean(born$children[muestra4000]))

plot(x,y, type="l", xlab="Tamaño muestra",ylab="Media", cex.lab=1.5, main="Estimación de la media") #Gráfico
abline(h=mean(born$children), col="red", lwd=2) #Añade línea horizontal
```



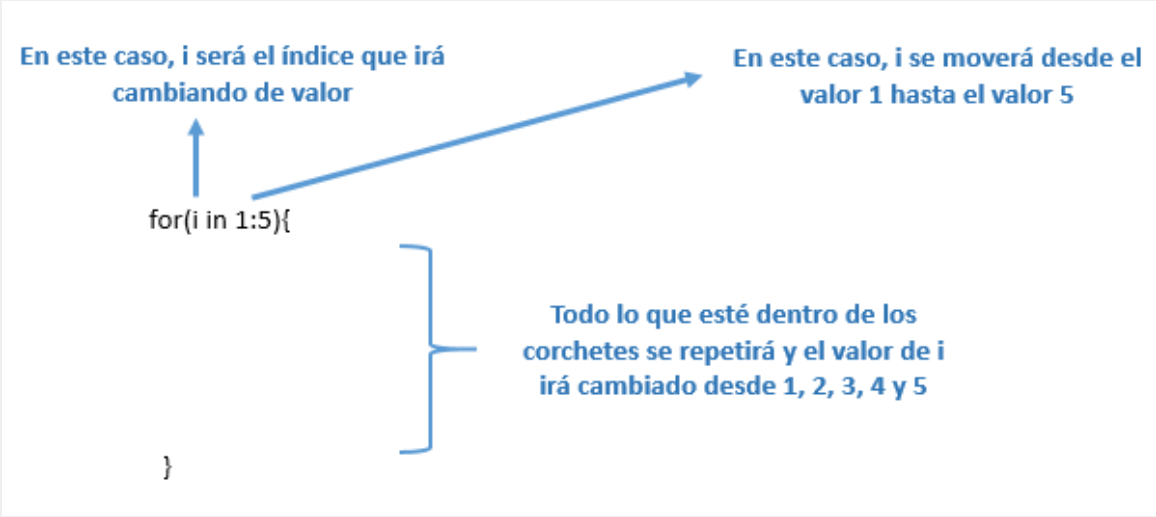
Automatizando un poco el proceso...

Podemos crear un *loop* o serie donde se repita un mismo procedimiento pero vaya cambiando el número de la iteración.

Automatizando un poco el proceso...

Podemos crear un *loop* o serie donde se repita un mismo procedimiento pero vaya cambiando el número de la iteración. Esto se realiza con *for* con la siguiente estructura:

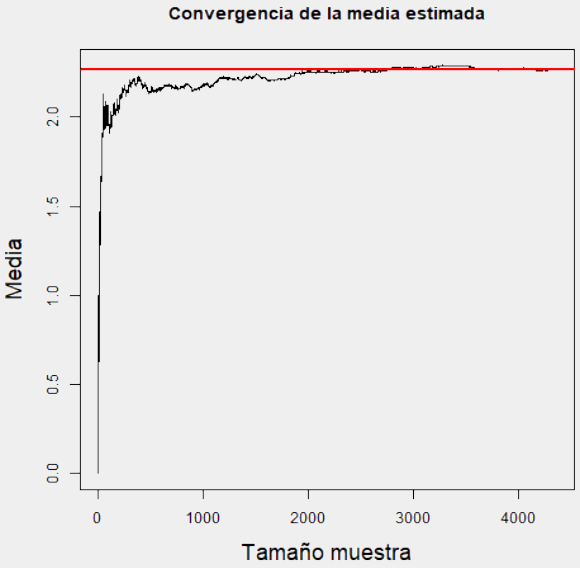
Creando loops



Uso de for

```
m<-nrow(born)  #Cantidad de filas
medias<-rep(0, m) #Vector que se llenará en cada iteración

for(i in 1:m){
  set.seed(2020)
  muestra<-sample(1:nrow(born), i, replace = FALSE)
  medias[i]<-mean(born$children[muestra])
}

plot(medias, type="l", ylab="Media", xlab="Tamaño muestra", cex.lab=1.5,
main="Convergencia de la media estimada")
abline(h=mean(born$children), col="red", lwd=2)
```

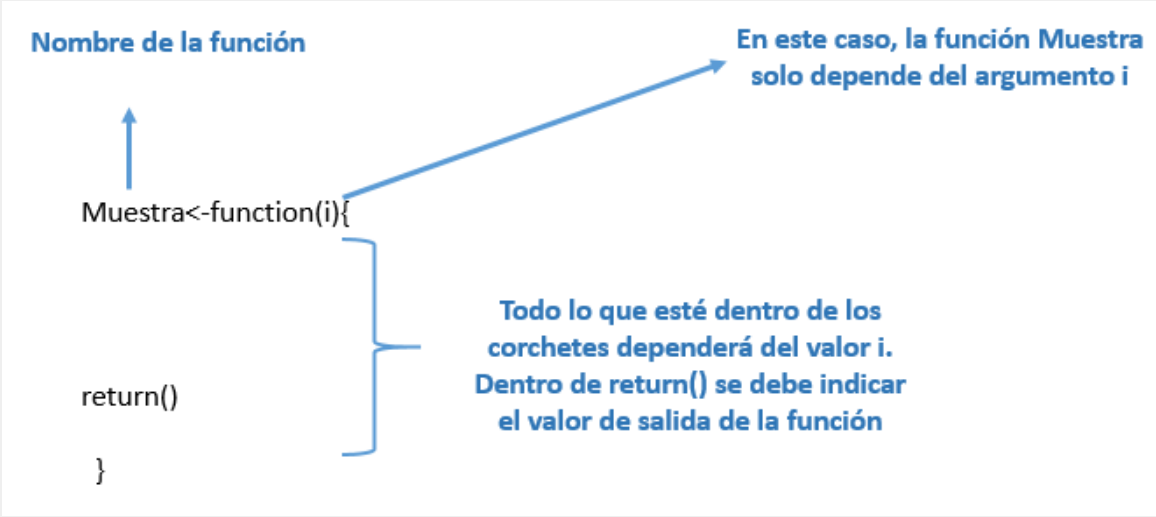
Resultados


Convergencia de la media estimada
Media
0.0
0.5
1.0
1.5
2.0
0
1000
2000
3000
4000
Tamaño muestra

Una forma más óptima:

```
muestra<-function(i){
  set.seed(2020)
  muestra<-sample(1:nrow(born), i, replace=FALSE)
  return(mean(born$children[muestra]))
}

medias<-sapply(1:nrow(born), FUN=muestra)
```

Funciones


Nombre de la función
En este caso, la función Muestra solo depende del argumento i
Muestra<-function(i){
Todo lo que esté dentro de los corchetes dependerá del valor i. Dentro de return() se debe indicar el valor de salida de la función
return()
}