Joseph Guy Evans Hilaire Jr

Bootstrap
&
Estimacion intervalo de confianza para la media

El metodo **Bootstrap** es un metodo que se apoya en la tecnica del muestreo aleatorio sin reemplazo. Con dicho metodo, se pueden estimar las propiedades de un estimador(media, varianza) mediante el muestreo sobre una distribucion aproximada.

En este trabajo, intentaremos estimar intervalos de confianza para la media de una muestra *Normal* usando el metodo Bootstrap.

Algoritmo I

Un algoritmo para estimar la media θ mediante el metodo Bootstrap es el siguiente:

1. Empezamos con una muestra observada $x_1, ...x_m$ y $\hat{\theta}$ un estimador *plug-in* de θ.
2. Para j=1,....,N generamos:

   $$x^{(j)} = \{x_{j1}, ..., x_{jm}\}$$

   donde $j_1, ..., j_m \sim U\{1, ..., m\}$
3. Luego para j=1,....,N hacemos:

   $$\hat{\theta}^{(j)} = \hat{\theta}(x^{(j)})$$

4. Ordenamos la muestra $\{\hat{\theta}^{(1)}, ..., \hat{\theta}^{(N)}\}$ de menor a mayor
5. Hacemos $l = \lceil \frac{\alpha}{2} N \rceil$ y $u = \lceil (1 - \frac{\alpha}{2}) N \rceil$
6. Luego hacemos $t = \hat{\theta}(x_1, ..., x_m)$
7. Entonces, el intervalo de confianza que buscabamos seria dado por:

   $$\left(2t - \hat{\theta}_{(u)}, 2t - \hat{\theta}_{(l)}\right)$$

A modo de **implementacion** del algoritmo anterior, supongamos que tenemos los datos $X_1, ..., X_m \sim N(0, 1)$. El intervalo de confianza para la media de esta muestra quedara como sigue:

```
import pandas as pd
import matplotlib.pyplot as plt

import random
import numpy as np
import math
import scipy.stats
```

```
alpha = 0.05
N = 1000


def intervConf_Bootstrap(x, estimador, N):
    n = int(x.size)  ## El tamano de las muestras con reemplazo
    theta = np.zeros(N)

    # Generamos las muestras Bootstrap
    for k in range(0,N):
        x_bos = np.random.choice(x,n,replace=True)  # Muestreo con reemplazo
        theta[k] = estimador(x_bos)

    # ordenamos los valores de theta
    theta.sort()
    thetaOrd = theta

    # Construimos el intervalo de confianza
    t = estimador(x)
    l = math.ceil(0.5*alpha*N )
    u = math.ceil((1-0.5*alpha)*N )
    return([2*t-thetaOrd[u] ,
            2*t-thetaOrd[l]])

# Intervalo de confianza segun su formula exacta:
def intervConf(x):
    n = x.size
    m = np.mean(x)
    s = np.std(x)

    # cuantil (1-0.5*alpha) de una T-student
    cuan = scipy.stats.t.ppf(1-0.5*alpha, n-1)

    return([m-cuan*s/math.sqrt(n), m+cuan*s/math.sqrt(n)])
```

```
n0 = 100
IC1 = np.zeros(n0)
IC2 = np.zeros(n0)

II1 = np.zeros(n0)
II2 = np.zeros(n0)

# Generamos los intervalos de confianza
# que despues graficaremos
for k in range(0,n0):
    x = np.random.normal(0,1,n0)

    IC1[k] = intervConf_Bootstrap(x,np.mean,N)[0]
    IC2[k] = intervConf_Bootstrap(x,np.mean,N)[1]

    II1[k] = intervConf(x)[0]
    II2[k] = intervConf(x)[1]
```

```python
# Graficamos los intervalos de confianza
# bajo el metodo Bootstrap
M = np.zeros(n0)

plt.figure(figsize=(10,5))  ## Determinamos las dimensiones de la grafica
plt.plot(M, color='red', label='Media')

# Graficamos los intervalos obtenidos mediante Bootstrap
plt.plot(IC1, 'o', color='royalblue', linestyle='dashed', label='Bootstrap')
plt.plot(IC2, 'o', color='royalblue', linestyle='dashed')

# Graficamos los intervalos obtenidos a partir de su formula exacta
plt.plot(II1, 'o', color='grey', linestyle='dashed', label='Metodo Exacto')
plt.plot(II2, 'o', color='grey', linestyle='dashed')


plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

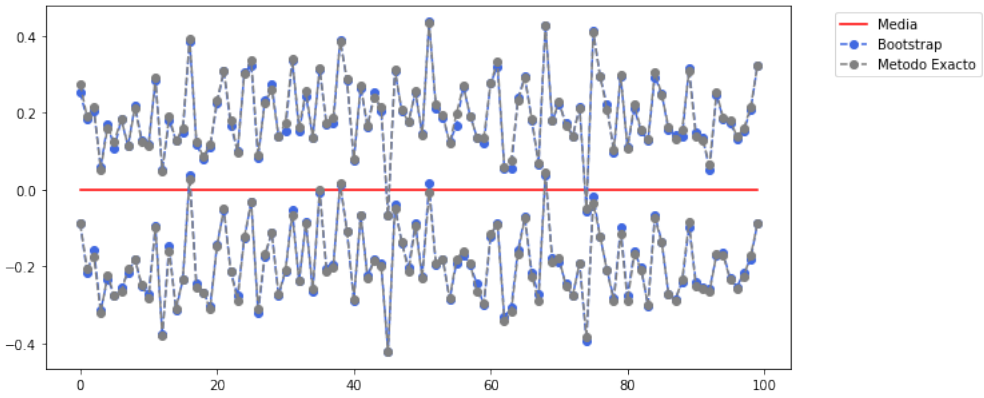

Figure 1: **Comparacion Intervalos de Confianza**

En la grafica anterior, notamos que los intervalos obtenidos bajo el metodo Bootstrap y el metodo exacto son casi parecidos. Por lo cual, podemos afirmar que el metodo Bootstrap es una buena metodologia para calcular los intervalos de confianza para los parametros de una muestra dada.

Algoritmo II

Con el algoritmo anterior, pudimos obtener una buena aproximacion para los intervalos de confianza de una muestra dada. Tambien, existe otro algoritmo de calculo de intervalos de confianza. Dicho algoritmo se traduce como sigue:

1. Empezamos con una muestra observada $x_1, ...x_m$ y $\hat{\theta}$ un estimador *plug-in* de θ.
2. Para j=1,....,N generamos:

   $$x^{(j)} = \{x_{j1}, ..., x_{jm}\}$$

   donde $j_1, ..., j_m \sim U\{1, ..., m\}$
3. Luego para j=1,....,N hacemos:

   $$\hat{\theta}^{(j)} = \hat{\theta}(x^{(j)})$$

4. Calculamos el siguiente cuantil de una distribucion Normal:

   $$\hat{z} = \Phi^{-1}\Big(\frac{\#\{j|\hat{\theta}^{(j)} \leq \hat{\theta}\}}{N}\Big)$$

5. Despues, para i=1,...,m , hacemos lo siguiente:

   $$\theta_{(i)} = \hat{\theta}_{n-1}(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$$

   en donde $\hat{\theta}_{n-1}(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$ es la evaluacion de $\hat{\theta}$ en la muestra pero sin el valor x_i, de ahi el subindice n-1.
6. Con eso calculamos:

   $$\bar{\theta} = \frac{1}{m}\sum_{i=1}^{m} \theta_{(i)}$$

7. Eso nos sirve para calcular los siguientes valores:

   $$\hat{a} = \frac{1}{6}\frac{\sum_{i=1}^{m}(\bar{\theta} - \theta_{(i)})^3}{\Big(\sum_{i=1}^{m}(\bar{\theta} - \theta_{(i)})^2\Big)^{3/2}}$$
   $$q_l = \Phi\Big(\hat{z} + \frac{\hat{z} + \Phi^{-1}(\alpha/2)}{1 - \hat{a}(\hat{z} + \Phi^{-1}(\alpha/2))}\Big)$$
   $$q_u = \Phi\Big(\hat{z} + \frac{\hat{z} + \Phi^{-1}(1-\alpha/2)}{1 - \hat{a}(\hat{z} + \Phi^{-1}(1-\alpha/2))}\Big)$$

8. Ordenamos la muestra $\{\hat{\theta}^{(1)}, ..., \hat{\theta}^{(N)}\}$ de menor a mayor
9. Obtenemos los siguientes valores: $l = \lceil q_l.N \rceil$ y $u = \lceil q_u.N \rceil$
10. Entonces, el intervalo de confianza bajo el metodo **Bias Corrected and Accelerated** seria dado por:

$$\left(\hat{\theta}_{(l)}, \hat{\theta}_{(u)}\right)$$

Ahora podemos implementar el algoritmo anterior de la siguiente manera:

```
def interv_BCa(x, estimador, N):
    n = x.size
    theta = np.zeros(N)

    for k in range(0,N):
        Xb = np.random.choice(x,n,replace=True)
        theta[k] = estimador(Xb)

    # numero de veces que es menor que el estimador(media)
    media = np.mean(x)
    c = np.where(theta <= media)
    q = len(c[0])/N

    # calculamos el cuantil correspondiente
    # a partir de una Normal
    zGorro = scipy.stats.norm.ppf(q)

    # Vector para guardar las medias a continuacion
    Ti = np.zeros(n)

    for k in range(0,n):
        # creamos una lista de los indices sin el indice k
        ind = np.delete(np.arange(n),k)
        # obtenemos la muestra sin el k-esimo dato
        xReducido = x[ind]
        # guardamos las medias
        Ti[k] = np.mean(xReducido)

    # Sacamos el promedio de las Ti
    Tmean = np.mean(Ti)

    # Calculamos el valor de (a_gorro)
    a0 = sum( (Tmean-Ti)**3 )
    a1 = sum( (Tmean-Ti)**2 )
    aGorro = a0 / (6*a1**(3/2))

    # Calculamos el valor de (qL)
    q0 = zGorro + scipy.stats.norm.ppf(alpha/2)
    q1 = 1- aGorro*q0
    qL = scipy.stats.norm.cdf(zGorro + q0/q1)

    # Calculamos el valor de (qU)
    q00 = zGorro + scipy.stats.norm.ppf(1-alpha/2)
    q11 = 1- aGorro*q00
    qU = scipy.stats.norm.cdf(zGorro + q00/q11)

    # Ordenamos la muestra
    theta.sort()

    # Calculamos (l) y (u):
    l = math.ceil(qL*N)
    u = math.ceil(qU*N)

    return([theta[l],theta[u]])
```

```
BCa1 = np.zeros(n0)
BCa2 = np.zeros(n0)

# Generamos los intervalos de confianza
# con el metodo BCa :
for k in range(0,n0):
    x = np.random.normal(0,1,n0)

    BCa1[k] = interv_BCa(x,np.mean,N)[0]
    BCa2[k] = interv_BCa(x,np.mean,N)[1]

    II1[k] = intervConf(x)[0]
    II2[k] = intervConf(x)[1]
```

```
# Graficamos los intervalos de confianza
# bajo el metodo Bias Corrected and Accelerated
M = np.zeros(n0)

plt.figure(figsize=(10,5))  ## Determinamos las dimensiones de la grafica
plt.plot(M, color='red', label='Media')

# Graficamos los intervalos obtenidos mediante el metodo BCa
plt.plot(BCa1, 'o', color='royalblue', linestyle='dashed', label='metodo BCa')
plt.plot(BCa2, 'o', color='royalblue', linestyle='dashed')

# Graficamos los intervalos obtenidos a partir de su formula exacta
plt.plot(II1, 'o', color='grey', linestyle='dashed', label='Metodo Exacto')
plt.plot(II2, 'o', color='grey', linestyle='dashed')


plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

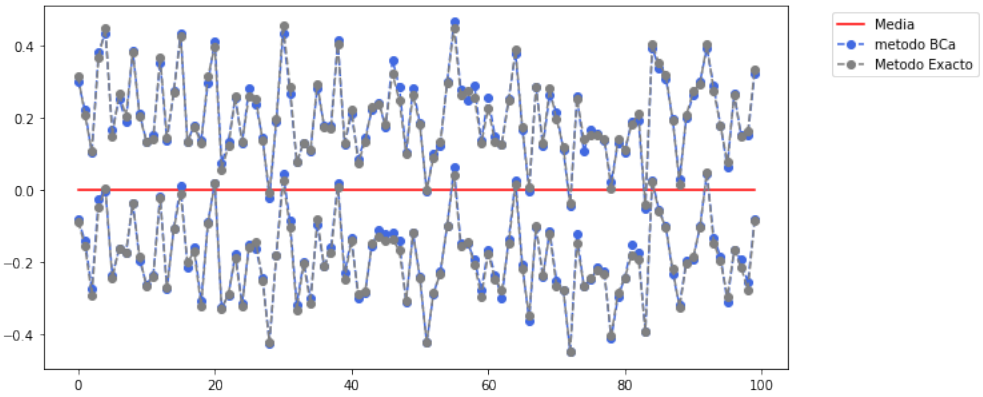

Figure 2: Comparacion Intervalos de Confianza

Con los intervalos de confianza obtenidos bajo las 2 metodologias anteriores, podemos decir que la que estimo mejores intervalos es la que logra capturar en mayor medida el valor de la media de los datos.

Para ello, procedemos a lo siguiente:

```
media = 0

num_01 = [IC1[k]<= media <= IC2[k] for k in range(len(IC1))]
num_02 = [BCa1[k]<= media <= BCa2[k] for k in range(len(BCa1))]

num_01 = np.sum(num_01)
num_02 = np.sum(num_02)

num_01, num_02
```

Obtenemos el siguiente resultado:

Metodologia	Bootstrap	BCa
Cantidad buena	94	87

Entonces podemos ver que hay 94 intervalos *Bootstrap* que logran capturar la media igual a **0** mientras que hay 87 intervalos *BCa* que capturan al valor de **0**.
Eso nos dice que en esta iteracion, el intervalo *Bootstrap* fue mejor que la otra metodologia para estimar intervalos de confianza adecuados para la media del conjunto de datos *Normales*.