Universidad Carlos III de Madrid

Técnicas de remuestreo , Grado en Estadística y Empresa

Inferencia estadística basada en métodos de remuestreo

Marcos Álvarez Martín
Fabio Scielzo Ortiz

Índice

1 Variables aleatorias i.i.d.

$\mathcal{X}_1, ..., \mathcal{X}_n$ son variables aleatorias mutuamente independientes e idénticamente distribuidas *(i.i.d.)* $\Leftrightarrow$

- $\mathcal{X}_1, ..., \mathcal{X}_n$ son mutuamente independientes , es decir:

$$P(\mathcal{X}_1 = x_1, ..., \mathcal{X}_n = x_n) \ = \ \prod_{i=1}^{n} P(\mathcal{X}_i = x_i)$$

  Lo que implica que también son independientes dos a dos , es decir, $\mathcal{X}_i \perp \mathcal{X}_j$, $\forall i \neq j$

- $\mathcal{X}_1, ..., \mathcal{X}_n$ tienen la misma distribución de probabilidad, es decir, $\mathcal{X}_i \sim F(\cdot)$, $\forall i = 1, ..., n$

Donde $F(\cdot)$ es una distribucion de probabilidad con parametros no especificados.

Usaremos la siguiente notación:

$$(\mathcal{X}_1, ..., \mathcal{X}_n) \underset{i.i.d.}{\sim} F(\cdot) \quad \Leftrightarrow \left\{ \begin{array}{l} \mathcal{X}_1, ..., \mathcal{X}_n \text{ son mutuamente independientes} \\ \mathcal{X}_i \sim F(\cdot) \ , \ \forall i = 1, ..., n \end{array} \right.$$

2 Muestra Aleatoria Simple

Sea $\mathcal{X}$ una v.a. tal que $\mathcal{X} \sim F(\cdot)$

$\mathcal{X}_1, ..., \mathcal{X}_n$ es una muestra aleatoria simple de tamaño n de $\mathcal{X}$ $\Leftrightarrow$ $(\mathcal{X}_1, ..., \mathcal{X}_n) \underset{i.i.d.}{\sim} F(\cdot)$

Observación:

Una m.a.s. de una v.a. $\mathcal{X}$ es un vector de v.a.'s mutuamente independientes y que se distribuyen probabilisticamente igual que la v.a. $\mathcal{X}$

3 Muestra de Observaciones

Sea $\mathcal{X}$ una v.a. tal que $\mathcal{X} \sim F(\cdot)$

$X = (x_1, ..., x_n)$ es una muestra de n observaciones de la v.a. $\mathcal{X}$ $\Leftrightarrow$ $x_i \in Im(\mathcal{X})$, $\forall i \in \{1, ..., n\}$ $\Leftrightarrow$ x_i es una realización de la v.a. $\mathcal{X}$, $\forall i \in \{1, ..., n\}$

Donde:

$Im(\mathcal{X})$ es la imagen de $\mathcal{X}$, es decir, su campo de variación.

Observaciones:

- Una muestra de observaciones de una v.a. es un vector de números, no son v.a.'s.
- Si $X = (x_1, ..., x_n)$ es una muestra de n observaciones de $\mathcal{X} \sim F(\cdot)$, entonces x_i es una observacion que ha sido generada por la distribución de probabilidad $F(\cdot)$, es decir, x_i puede verse como un numero aleatorio generado en base a la distribución de probabilidad $F(\cdot)$
- Si $X = (x_1, ..., x_n)$ es una muestra de n observaciones de $\mathcal{X} \sim F(\cdot)$, entonces:
  - $P(\mathcal{X} = x_i)$ es la probabilidad de observar x_i al extraer una muestra de observaciones de $\mathcal{X}$
- Si $(\mathcal{X}_1, ..., \mathcal{X}_n)$ es una muestra aleatoria simple (m.a.s.) de tamaño n de $\mathcal{X} \sim F(\cdot)$, entonces:
  - $P(\mathcal{X}_1 = x_1, ..., \mathcal{X}_n = x_n)$ es la probabilidad de obtener como valores $(x_1, ..., x_n)$ al extraer una muestra de observaciones de $\mathcal{X}$

4 Estadístico

T es un estadístico de una m.a.s $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de la v.a. $\mathcal{X} \sim F(\theta) \quad \Leftrightarrow \quad T$ es una función de la m.a.s que no depende del parámetro θ

Por tanto:

- $T(\mathcal{X}_1, ..., \mathcal{X}_n)$ es un estadístico.

Observaciones:

- $T(\mathcal{X}_1, ..., \mathcal{X}_n)$ es una v.a. al ser una función de v.a.'s
- Dada una muestra de observaciones $(x_1, ..., x_n)$ de la v.a. $\mathcal{X} \sim D(\theta)$
  - $T(x_1, ..., x_n)$ es una observación de la v.a. $T(\mathcal{X}_1, ..., \mathcal{X}_n)$
- Dadas B muestras de observaciones $(x_1^1, ..., x_n^1)$, ..., $(x_1^B, ..., x_n^B)$ de la v.a. $\mathcal{X} \sim D(\theta)$
  - $T(x_1^1, ..., x_n^1)$, ..., $T(x_1^B, ..., x_n^B)$ es una muestra de observaciones de la v.a. $T(\mathcal{X}_1, ..., \mathcal{X}_n)$

4.1 Ejemplos de estadisticos

Sea $\mathcal{X}$ una v.a. tal que $\mathcal{X} \sim D(\theta)$, y sea $(\mathcal{X}_1, ..., \mathcal{X}_n)$ una m.a.s. de $\mathcal{X}$

- **Media muestral**

$$T(\mathcal{X}_1, ..., \mathcal{X}_n) = \overline{\mathcal{X}} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{X}_i$$

- **Varianza muestral**

$$T(\mathcal{X}_1, ..., \mathcal{X}_n) = S^2 = \frac{1}{n} \sum_{i=1}^{n} (\mathcal{X}_i - \overline{\mathcal{X}})^2$$

- **Cuasi-Varianza muestral**

$$T(\mathcal{X}_1, ..., \mathcal{X}_n) = S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (\mathcal{X}_i - \overline{X})^2$$

5 Estimador Puntual

Dada una v.a. $\mathcal{X} \sim D(\theta)$ y una m.a.s. $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$,

Un estimador puntual para el parametro θ es un estadistico $\widehat{\theta}(\mathcal{X}_1, ..., \mathcal{X}_n)$ que se propone para estimar θ

6 Estimación Puntual

Dada una v.a. $\mathcal{X} \sim D(\theta)$, una m.a.s. $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$ y un estadistico $\widehat{\theta}(\mathcal{X}_1, ..., \mathcal{X}_n)$

Si $X = (x_1, ..., x_n)$ es una muestra de observaciones de $\mathcal{X}$, entonces:

- $\widehat{\theta}(X) = \widehat{\theta}(x_1, ..., x_n)$ es una estimación puntual del parametro θ

Observaciones:

Un estimador puntual es una v.a. y una estimación puntual un número.

7 Propiedades básicas de los estimadores

Dada una v.a. $\mathcal{X} \sim D(\theta)$, una m.a.s. $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$ y un estimador $\widehat{\theta}(\mathcal{X}_1, ..., \mathcal{X}_n)$

7.1 Sesgo

El sesgo del estimador $\widehat{\theta}$ se define como:

$$Sesgo(\widehat{\theta}) = E\left[\widehat{\theta}\right] - \theta$$

7.2 Varianza

La varianza del estimador $\widehat{\theta}$ se define como:

$$Var(\widehat{\theta}) = E\left[\left(\widehat{\theta} - E[\widehat{\theta}]\right)^2\right]$$

El error estandar (desviación típica) del estimador $\widehat{\theta}$ se define como:

$$s.e.(\widehat{\theta}) = \sqrt{Var(\widehat{\theta})}$$

7.3 Error Cuadratico Medio

El error cuadratico medio del estimador $\widehat{\theta}$ se define como:

$$ECM(\widehat{\theta}) = E\left[\ (\widehat{\theta} - \theta)^2\ \right]$$

Propiedades

- $ECM(\widehat{\theta}) = Var(\hat{\theta}) + Sesgo(\widehat{\theta})^2$

8 Estimación del sesgo y varianza por Jacknife

Tenemos una v.a. $\mathcal{X} \sim D(\theta)$, una m.a.s. $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$ y un estimador $\widehat{\theta}(\mathcal{X}_1, ..., \mathcal{X}_n)$ del parámetro θ

Además tenemos una muestra de observaciones $X = (x_1, ..., x_n)$ de la variable de interés $\mathcal{X}$, por lo que tenemos la estimación $\widehat{\theta}(X) = \widehat{\theta}(x_1, ..., x_n)$ del parámetro θ

Se define $X_{(r)}$ como la muestra que contiene todos los valores de X excepto x_r

Es decir:

$$X_{(r)} = (\, x_1 \,,\, x_2 \,, .., \, x_{r-1} \,,\, x_{r+1} \,, .., \, x_n \,)$$

para $r = 1, ..., n$

Se define la replica r-esima del estimador $\widehat{\theta}$ como:

$$\widehat{\theta}_{(r)} = \widehat{\theta}(X_{(r)}) = \widehat{\theta}(\, x_1 \,,\, x_2 \,, .., \, x_{r-1} \,,\, x_{r+1} \,, .., \, x_n \,)$$

8.1 Estimación Jacknife del sesgo

La **estimación Jacknife** del **sesgo** del estimador $\widehat{\theta}$ se define como:

$$\widehat{Sesgo}(\widehat{\theta})_{Jack} \; = \; (n-1) \cdot \left(\frac{1}{n} \cdot \sum_{r=1}^{n} \widehat{\theta}_{(r)} \; - \; \widehat{\theta}(X) \right)$$

Donde:

$$\widehat{\theta}(X) \; = \; \widehat{\theta}(x_1, ..., x_n)$$

$$\widehat{\theta}_{(r)} = \widehat{\theta}(X_{(r)}) = \widehat{\theta}(\, x_1 \,,\, x_2 \,, .., \, x_{r-1} \,,\, x_{r+1} \,, .., \, x_n \,)$$

8.2 Estimación Jacknife de la varianza

La **estimación Jacknife** de la **varianza** del estimador $\widehat{\theta}$ se define como:

$$\widehat{Var}(\widehat{\theta})_{Jack} \;=\; \frac{n-1}{n} \cdot \sum_{r=1}^{n} \left(\widehat{\theta}_{(r)} \;-\; \frac{1}{n} \sum_{r=1}^{n} \widehat{\theta}_{(r)} \right)^2$$

Donde:

$$\widehat{\theta}_{(r)} = \widehat{\theta}(X_{(r)}) = \widehat{\theta}(\,x_1\,,\,x_2\,,..,\,x_{r-1}\,,\,x_{r+1}\,,..,\,x_n\,)$$

La **estimación Jacknife** del **error estandar** del estimador $\widehat{\theta}$ se define como:

$$\widehat{s.e.}(\widehat{\theta})_{Jack} \;=\; \sqrt{\widehat{Var}(\widehat{\theta})_{Jack}}$$

Observación:

El Jacknife funciona bien cuando el estimador es suave (smooth).

Un estimador es suave cuando ante pequeños cambios en la muestra de datos genera pequeños cambios en el estimador.

Ejemplo de estimador suave es el estimador plug-in de la media poblacional, es decir la media muestral.

Ejemplo de estimador no suave es el estimador plug-in de la mediana poblacional, es decir la mediana muestral

8.3 Estimación Jacknife de un parametro con corrección de sesgo

Tenemos una v.a. $\mathcal{X} \sim D(\theta)$, una m.a.s. $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$ y un estimador $\widehat{\theta}(\mathcal{X}_1, ..., \mathcal{X}_n)$ del parametro θ

Además tenemos una muestra de observaciones $X = (x_1, ..., x_n)$ de la variable de interés $\mathcal{X}$, por lo que tenemos la estimación $\widehat{\theta}(X) = \widehat{\theta}(x_1, ..., x_n)$ del parámetro θ

La **estimación Jacknife con sesgo corregido** del parametro θ se define como:

$$\widehat{\theta}_{Jack} \; = \; \widehat{\theta}(X) - \widehat{Sesgo}(\widehat{\theta})_{Jack} \; = \; \widehat{\theta}(X) - (n-1) \cdot \left(\frac{1}{n} \cdot \sum_{r=1}^{n} \widehat{\theta}_{(r)} - \widehat{\theta} \right) \; = \; n \cdot \widehat{\theta}(X) - (n-1) \cdot \frac{1}{n} \cdot \sum_{r=1}^{n} \widehat{\theta}_{(r)}$$

Donde:

$$\widehat{\theta} \; = \; \widehat{\theta}(X) \; = \; \widehat{\theta}(x_1, ..., x_n)$$

$$\widehat{\theta}_{(r)} = \widehat{\theta}(X_{(r)}) = \widehat{\theta}(\, x_1 \, , \, x_2 \, , .., \, x_{r-1} \, , \, x_{r+1} \, , .., \, x_n \,)$$

8.4 Jacknife en Python

```
def Jacknife(Variable , estimator_function, q=0.75):

#######################################################

    def Jacknife_sample(X , r):

            X_sample_r = np.delete(X, r)

            return(X_sample_r)


    if estimator_function == np.quantile :  estimation =
    ↪  estimator_function(Variable , q=q)

    else :  estimation = estimator_function(Variable)


    replicas_estimador = []

    for r in range(0, len(Variable)):

        if estimator_function == np.quantile :  Jack_estimation =
        ↪  estimator_function( Jacknife_sample(Variable, r) , q=q )

        else :  Jack_estimation = estimator_function(
        ↪  Jacknife_sample(Variable, r) )

        replicas_estimador.append( Jack_estimation )


    n = len(Variable)

    sesgo = (n-1) * ( np.mean( replicas_estimador ) - estimation )

    estimacion_sesgo_corregido = estimation - sesgo

    standard_error = np.sqrt( ((n-1)/n) * sum( (replicas_estimador -
↪   np.mean( replicas_estimador ))**2 ) )


#######################################################

    return(sesgo, estimacion_sesgo_corregido, standard_error)
```

```
import numpy as np
```

```
np.random.seed(123)

X = np.random.normal(loc=10, scale=15, size=50)
```

Jacknife para la mediana:

```
np.median(X)
```

```
8.23916733155832
```

```
sesgo, estimacion_sesgo_corregido, standard_error = Jacknife(X ,
↪  estimator_function=np.median , q=0.75)
```

```
sesgo
```

```
8.704148513061227e-14
```

```
estimacion_sesgo_corregido
```

```
8.239167331558233
```

```
standard_error
```

```
2.381386940718188
```

Jacknife para la media:

```
np.mean(X)
```

```
sesgo, estimacion_sesgo_corregido, standard_error = Jacknife(X ,
↪  estimator_function=np.mean)
```

```
sesgo
```

```
-8.704148513061227e-14
```

```
estimacion_sesgo_corregido
```

```
10.199071616256864
```

```
standard_error
```

```
2.5491917443460235
```

Jacknife para la desviación típica:

```
np.std(X)
```

```
sesgo, estimacion_sesgo_corregido, standard_error = Jacknife(X ,
↪   estimator_function=np.std)
```

```
sesgo
```

```
-0.26116994703598806
```

```
estimacion_sesgo_corregido
```

```
18.105512157458175
```

```
standard_error
```

```
1.6795955569730596
```

Jacknife para los cuantiless:

```
np.quantile(X , q=0.75)
```

```
23.835596841535178
```

```
sesgo, estimacion_sesgo_corregido, standard_error = Jacknife(X ,
↪   np.quantile, q=0.75)
```

```
sesgo
```

```
-0.14962568429344003
```

```
estimacion_sesgo_corregido
```

```
23.98522252582862
```

```
standard_error
```

```
0.9375849844484524
```

Jacknife para la curtosis:

```
import scipy

from scipy.stats import  kurtosis
```

```
kurtosis(X)
```

```
-0.37420768292897266
```

```
sesgo, estimacion_sesgo_corregido, standard_error = Jacknife(X ,
↪  estimator_function=kurtosis)
```

```
sesgo
```

```
-0.036248944427488833
```

```
estimacion_sesgo_corregido
```

```
-0.33795873850148384
```

```
standard_error
```

```
0.3609496287814513
```

Jacknife para la asimetria:

```
from scipy.stats import  skew
```

```
skew(X)
```

```
0.025587358812510053
```

```
sesgo, estimacion_sesgo_corregido, standard_error = Jacknife(X ,
↪  estimator_function=skew)
```

```
sesgo
```

```
0.021610720628010085
```

```
estimacion_sesgo_corregido
```

```
0.0039766381844999685
```

```
standard_error
```

```
0.27115376266443825
```

9 Estimación del sesgo , varianza y error cuadratico medio de un estimador por Bootstrap

Tenemos una v.a. $\mathcal{X} \sim D(\theta)$, una m.a.s. $\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$ y un estimador $\widehat{\theta}(\mathcal{X}_1, ..., \mathcal{X}_n)$ del parámetro θ

Además tenemos una muestra de observaciones $X = (x_1, ..., x_n)$ de la variable de interés $\mathcal{X}$, por lo que tenemos la estimación $\widehat{\theta}(X) = \widehat{\theta}(x_1, ..., x_n)$ del parametro θ

Una **muestra bootstrap** de $X = (x_1, ..., x_n)$ se define como una **muestra aleatoria con reemplazamiento** de tamaño n de X

Tenemos B muestras bootstrap de X :

$$X_{(1)}, X_{(2)}, ..., X_{(B)}$$

Se define la replica bootstap b-esima del estimador $\widehat{\theta}$ como:

$$\widehat{\theta}_{(b)} = \widehat{\theta}(X_{(b)})$$

9.1 Estimación bootstrap del sesgo de un estimador

La **estimación bootstrap** del **sesgo** del estimador $\widehat{\theta}$ se define como:

$$\widehat{Sesgo}(\widehat{\theta})_{Boot} \;=\; \frac{1}{B} \cdot \sum_{b=1}^{B} \left(\widehat{\theta}_{(b)} - \widehat{\theta}(X)\right) \;=\; \frac{1}{B} \cdot \sum_{b=1}^{B} \widehat{\theta}_{(b)} \;-\; \widehat{\theta}(X)$$

Donde:

$$\widehat{\theta}_{(b)} = \widehat{\theta}(X_{(b)})$$

$$\widehat{\theta}(X) \;=\; \widehat{\theta}(x_1, ..., x_n)$$

Observación:

La estimación bootstrap del sesgo del estimador $\widehat{\theta}$ es la media del vector de replicas bootstrap $(\widehat{\theta}_{(1)}, \widehat{\theta}_{(2)}, ..., \widehat{\theta}_{(B)})$ menos la estimación $\widehat{\theta}(X)$

9.2 Estimación bootstrap de la varianza de un estimador

La **estimación Bootstrap** de la **varianza** del estimador $\widehat{\theta}$ se define como:

$$\widehat{Var}(\widehat{\theta})_{Boot} \ = \ \frac{1}{B-1} \cdot \sum_{b=1}^{B} \left(\widehat{\theta}_{(b)} \ - \ \frac{1}{B} \cdot \sum_{b=1}^{B} \widehat{\theta}_{(b)} \right)^2$$

Donde:

$$\widehat{\theta}_{(b)} = \widehat{\theta}(X_{(b)})$$

La **estimación bootstrap** de la **desviación típica** del estimador $\widehat{\theta}$ se define como:

$$\widehat{s.e.}(\widehat{\theta})_{Boot} \ = \ \sqrt{\widehat{Var}(\widehat{\theta})_{Boot}}$$

Observación:

La estimación bootstrap de la varianza del estimador $\widehat{\theta}$ es la cuasi-varianza del vector de replicas bootstrap $(\widehat{\theta}_{(1)}, \widehat{\theta}_{(2)}, ..., \widehat{\theta}_{(B)})$

9.3 Estimación bootstrap del error cuadratico medio de un estimador

La **estimación Bootstrap** del **error cuadrático medio** del estimador $\widehat{\theta}$ se define como:

$$\widehat{ECM}(\widehat{\theta})_{Boot} \ = \ \frac{1}{B} \cdot \sum_{b=1}^{B} \left(\widehat{\theta}_{(b)} - \widehat{\theta}(X) \right)^2$$

Donde:

$$\widehat{\theta}_{(b)} = \widehat{\theta}(X_{(b)})$$

$$\widehat{\theta}(X) \ = \ \widehat{\theta}(x_1, ..., x_n)$$

9.4 Estimación bootstrap de un parametro con corrección de sesgo

La **estimación Bootstrap con sesgo corregido** del parametro θ se define como:

$$\widehat{\theta}_{Boot} \;=\; \widehat{\theta}(X) \,-\, \widehat{Sesgo}(\widehat{\theta})_{Boot} \;=\; \widehat{\theta}(X) \,-\, \frac{1}{B} \cdot \sum_{b=1}^{B} \left(\widehat{\theta}_{(b)} - \widehat{\theta}(X) \right)$$

Donde:

$$\widehat{\theta}_{(b)} = \widehat{\theta}(X_{(b)})$$

$$\widehat{\theta}(X) \;=\; \widehat{\theta}(x_1, ..., x_n)$$

9.5 Número de muestras bootstrap posibles

9.6 Bootstrap en Python

```python
def Bootstrap(Variable , B, estimator_function, q=0.75, random_seed=123 ):

#######################################################

    np.random.seed(random_seed)

#######################################################


    def Bootstrap_sample(X):

            from sklearn.utils import resample

            sample = resample( X, n_samples=len(X))

            return sample


    if estimator_function == np.quantile : estimation =
    ↪  estimator_function(Variable , q)

    else:  estimation = estimator_function(Variable)


    replicas_estimador = []

    for b in range(0, B):

        if estimator_function == np.quantile : estimation_Boot =
        ↪  estimator_function(Bootstrap_sample(Variable) , q=q)

        else:  estimation_Boot =
        ↪  estimator_function(Bootstrap_sample(Variable))

        replicas_estimador.append( estimation_Boot )


    sesgo = np.mean( replicas_estimador ) - estimation

    estimacion_sesgo_corregido = estimation - sesgo

    standard_error = np.std( replicas_estimador )


#######################################################

    return sesgo , estimacion_sesgo_corregido , standard_error,
    ↪  replicas_estimador
```

Bootstrap para la mediana:

```
np.median(X)
```

```
8.23916733155832
```

```
sesgo , estimacion_sesgo_corregido , standard_error, replicas_estimador =
↪ Bootstrap(X , B=20000, estimator_function=np.median)
```

```
sesgo
```

```
0.4499295142913571
```

```
estimacion_sesgo_corregido
```

```
7.789237817266963
```

```
standard_error
```

```
3.7580290713770914
```

Bootstrap para la media:

```
np.mean(X)
```

```
sesgo , estimacion_sesgo_corregido , standard_error, replicas_estimador =
↪ Bootstrap(X , B=20000, estimator_function=np.mean)
```

```
sesgo
```

```
0.020446171871411995
```

```
estimacion_sesgo_corregido
```

```
10.178625444385357
```

```
standard_error
```

```
2.5121071650105433
```

Bootstrap para la desviación típica:

```
np.std(X)
```

```
17.844342210422187
```

```
sesgo , estimacion_sesgo_corregido , standard_error, replicas_estimador =
↪  Bootstrap(X , B=20000, estimator_function=np.std)
```

```
replicas_estimador[0:10]
```

```
[17.290069166352062,
 17.734700491052042,
 18.091905056508867,
 19.296946848609828,
 16.371718686759674,
 17.889036386608268,
 16.84496188781967,
 18.60024334635148,
 20.188328416078278,
 19.38523325913237]
```

```
sesgo
```

```
-0.24900263565758252
```

```
estimacion_sesgo_corregido
```

```
18.09334484607977
```

```
standard_error
```

```
1.6154115156505968
```

Bootstrap para los cuantiles:

```
np.quantile(X, q=0.75)
```

```
23.835596841535178
```

```
sesgo , estimacion_sesgo_corregido , standard_error, replicas_estimador =
↪ Bootstrap(X , B=20000, estimator_function=np.quantile, q=0.75)
```

```
sesgo
```

```
-1.006032620023607
```

```
estimacion_sesgo_corregido
```

```
24.841629461558785
```

```
standard_error
```

```
3.4665427821179793
```

Bootstrap para la asimetría:

```
skew(X)
```

```
0.025587358812510053
```

```
sesgo , estimacion_sesgo_corregido , standard_error, replicas_estimador =
↪ Bootstrap(X , B=20000, estimator_function=skew)
```

```
sesgo
```

```
0.019212954042191917
```

```
estimacion_sesgo_corregido
```

```
0.006374404770318136
```

```
standard_error
```

```
0.253427740069774
```

Bootstrap para la curtosis:

```
kurtosis(X)
```

```
-0.37420768292897266
```

```
sesgo , estimacion_sesgo_corregido , standard_error, replicas_estimador =
↪  Bootstrap(X , B=20000, estimator_function=kurtosis)
```

```
sesgo
```

```
-0.022470288214714473
```

```
estimacion_sesgo_corregido
```

```
-0.3517373947142582
```

```
standard_error
```

```
0.35103721258189574
```

10 Fundamentos del Bootstrap

10.1 La función de distribución

Dada una v.a. $\mathcal{X} \sim D(\theta)$ y una m.a.s. $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$

La **función de distribución** de la v.a. $\mathcal{X}$ es :

$$F_X(z) = P(X \leq z) \ \ , \ \ \forall z \in \mathbb{R}$$

Observación:

La función de distribución de la v.a. $\mathcal{X}$ coincide con las funciones de distribución de las v.a's $\mathcal{X}_1, ..., \mathcal{X}_n$
, porque tienen la misma distribución de probabilidad.

$$F_X(z) = F_{X_i}(z) \ \ , \ \ \forall z \in \mathbb{R} \ \ , \ \ \forall i \in \{1, ..., n\}$$

10.2 La función de distribución empírica

Dada una v.a. $\mathcal{X} \sim D(\theta)$

cmpan>

La función de distribución empírica basada en una m.a.s. $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$ se define como:

$$\widehat{F}_n(z) \ = \ \frac{1}{n} \sum_{i=1}^{n} I(\mathcal{X}_i \geq z)$$

donde:

$$I(X_i \geq z) = \begin{cases} 1 \ , & \text{si} \quad X_i \geq z \\ 0 \ , & \text{si} \quad X_i > z \end{cases}$$

para $z \in \mathbb{R}$

Observaciones:

- $\widehat{F}_n(z)$ es una v.a.

- $\widehat{F}_n(z)$ es usada como estimador de $F_X(z)$

10.2.1 Porpiedades de la función de distribución empírica como v.a.

Algunas propiedades de la distribución empírica como variable aleatoria:

- $I(X_i \geq z) \quad \sim \quad Bernoulli\,(p) \quad , \quad \text{con} \quad p = F_X(z) = P(X < z)$

- $\sum_{i=1}^{n} I(X_i \geq z) \quad \sim \quad Binomial(n,p) \quad , \quad \text{con} \quad p = F_X(z) = P(X < z)$

- $\widehat{F}_n(z) \; = \; \dfrac{1}{n} \sum_{i=1}^{n} I(X_i \geq z) \quad \sim \quad \dfrac{1}{n} \cdot Binomial(n,p)$

  donde $p \; = \; F_X(z) \; = \; P(X < z)$

- $E\left[\widehat{F}_n(z)\right] \; = \; E\left[\dfrac{1}{n} \cdot Binomial(n,p)\right] \; = \; \dfrac{1}{n} \cdot n \cdot p \; = \; p \; = \; F_X(z) = P(X < z)$

- $Var\left[\widehat{F}_n(z)\right] \; = \; Var\left[\dfrac{1}{n} \cdot Binomial(n,p)\right] \; = \; \dfrac{1}{n^2} \cdot n \cdot p \cdot (1-p) =$

  $= \; \dfrac{1}{n} \cdot F_X(z) \cdot (1 - F_X(z)) \; = \; F_X(z) = P(X < z)$

10.3 Función de distribución empírica como estimación de la funcion de distribución

Si tenemos una muestra de observaciones $X = (x_1, ..., x_n)$ de la variable de interés $\mathcal{X}$

Tenemos la siguiente **estimación** de la función de distribución de $\mathcal{X}$ a través de la funcion de distribución emprica:

$$\widehat{F}_n(z) \ = \ \frac{1}{n}\sum_{i=1}^{n} I(x_i \geq z) \ = \ \frac{\#\{\ i = 1, .., n \ / \ x_i \geq z \ \}}{n} \quad , \quad z \in \mathbb{R}$$

Propiedades de la función de distribución empírica como estimación

- $\widehat{F}_n(z) \ = \ Q(X, z)$

  Donde: $Q(X, z)$ es el cuantil de orden z de $X = (x_1, ..., x_n)$

- Si se ordena la muestra $X = (x_1, ..., x_n)$ de menor a mayor $x_{(1)} \ < \ x_{(2)} \ < ... < \ x_{(n)}$, entonces:

$$\widehat{F}_n(z) = \begin{cases} 0 \ , \quad \text{si} \quad z < x_{(1)} \\ 1/n \ , \quad \text{si} \quad z = x_{(1)} \\ 1/n \ , \quad \text{si} \quad x_{(1)} \leq z < x_{(2)} \\ 2/n \ , \quad \text{si} \quad z = x_{(2)} \\ 2/n \ , \quad \text{si} \quad x_{(2)} \leq z < x_{(3)} \\ \dots \\ (n-1)/n \ , \quad \text{si} \quad z = x_{(n-1)} \\ (n-1)/n \ , \quad \text{si} \quad x_{(n-1)} \leq z < x_{(n)} \\ 1 \ , \quad \text{si} \quad z \geq x_{(n)} \end{cases}$$

10.4 Ley debil de los grandes números

La ley debil de los grandes números afirma lo siguiente:

Dada una v.a. $\mathcal{X} \sim D(\theta)$ tal que $E[\mathcal{X}] = \mu$

Si $\widehat{F}_n(z)$ es la función de distribución empirica basada en la m.a.s. $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$, se cumple que:

$$\frac{1}{n}\sum_{i=1}^{n} \mathcal{X}_i \underset{p}{\rightarrow} E[X] = \mu$$

Observación: $E[X] = E[X_i]$, $\forall i \in \{1, ..., n\}$

Podemos aplicar la ley de los grandes números a la distribución empirica:

Como $I(\mathcal{X}_i \geq z) \sim Bernoulli(p)$, con $E\left[I(\mathcal{X}_i \geq z)\right] = p = F_X(z) = P(X \leq z)$

Aplicando la ley debil de los grandes números tenemos lo siguiente:

$$\widehat{F}_n(z) = \frac{1}{n}\sum_{i=1}^{n} I(\mathcal{X}_i \geq z) \underset{p}{\rightarrow} p = F_X(z)$$

En conclusión:

$$\widehat{F}_n(z) \underset{p}{\rightarrow} F_X(z)$$

Usando la definición de convergencia en probabilidad, se tiene que:

$$\underset{n\to\infty}{lim} P\left(|\widehat{F}_n(z) - F_X(z)| \leq \varepsilon \right) = 1 \ , \ \forall \varepsilon > 0$$

Pero se cumple un resultado más fuerte aun, el **teorema de Glivenko-Cantelli**.

10.5 Teorema de Glivenko-Cantelli

Dada una v.a. $\mathcal{X} \sim D(\theta)$ tal que $E[\mathcal{X}] = \mu$

Si $\widehat{F}_n(z)$ es la función de distribución empirica basada en la m.a.s. $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$, se cumple que:

$$sup\left\{ \left| \widehat{F}_n(z) - F_X(z) \right| \ / \ z \in \mathbb{R} \right\} \underset{p}{\rightarrow} 0$$

10.5.1 Demostración del teorema de Glivenko-Cantelli

11 Intervalos de confianza basados en bootstrap

Las desviaciones típicas o errores estándar se pueden usar para calcular intervalos de confianza aproximados para los parametros de interés.

11.1 Intervalos cuantil-bootstrap con una población

Primero vamos a fijar una vez mas el contexto en el que no estamos moviendo, puesto que es importante recordarlo:

Tenemos una v.a. $\mathcal{X} \sim D(\theta)$, una m.a.s. $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$ y un estimador $\widehat{\theta}(\mathcal{X}_1, ..., \mathcal{X}_n)$ del parámetro θ

Además tenemos una muestra de observaciones $X = (x_1, ..., x_n)$ de la variable de interés $\mathcal{X}$, por lo que tenemos la estimación $\widehat{\theta}(X) = \widehat{\theta}(x_1, ..., x_n)$ del parámetro θ

- Se obtienen B muestras bootstrap (aleatorias y con reemplazamiento) de X :

$$X_{(1)}, X_{(2)}, ..., X_{(B)}$$

- Se calcula para $b \in \{1, ..., B\}$ la replica bootstrap b-esima del estimador $\widehat{\theta}$ como:

$$\widehat{\theta}_{(b)} = \widehat{\theta}(X_{(b)})$$

Asi que se tiene:

$$\widehat{\theta}_{boot} = \left(\widehat{\theta}_{(1)}, ..., \widehat{\theta}_{(B)}\right)$$

Sea $Q(\alpha \, , \widehat{\theta}_{boot})$ el cuantil de orden α de la variable $\widehat{\theta}_{boot}$, entonces se cumple lo siguiente:

$$\frac{\# \left\{ b = 1, ..., B \ \ / \ \ \widehat{\theta}_{(b)} \ \leq \ Q(\alpha \, , \widehat{\theta}_{boot}) \right\}}{B} \ = \ \alpha$$

- El intervalo cuantil-bootstrap para el parámetro θ a un nivel $1-\alpha$ es :

$$IC(\theta)_{1-\alpha}^{boot} \ = \ \left[\ Q(\alpha/2 \ , \ \widehat{\theta}_{boot}) \ \ ; \ \ Q(1-\alpha/2 \ , \ \widehat{\theta}_{boot}) \ \right]$$

11.2 Intervalos cuantil-bootstrap con dos poblaciones

Tenemos dos v.a's $\mathcal{X}_1 \sim D_1(\theta_1)$ y $\mathcal{X}_2 \sim D_2(\theta_2)$, una m.a.s. $(\mathcal{X}_{11},...,\mathcal{X}_{n_1 1})$ de $\mathcal{X}_1$, otra m.a.s. $(\mathcal{X}_{12},...,\mathcal{X}_{n_2 2})$ de $\mathcal{X}_2$ y un par de estimadores $\widehat{\theta}_1(\mathcal{X}_{11},...,\mathcal{X}_{n_1 1})$ y $\widehat{\theta}_2(\mathcal{X}_{12},...,\mathcal{X}_{n_2 2})$ de los parámetros θ_1 y θ_2, respectivamente.

Además tenemos una muestras de observaciones $X_1 = (x_{11},...,x_{n_1 1})$ de la v.a. $\mathcal{X}_1$ y otra $X_1 = (x_{11},...,x_{n_1 1})$ de $\mathcal{X}_2$, por lo que tenemos las estimaciones $\widehat{\theta}_1(X_1) = \widehat{\theta}_1(x_{11},...,x_{n_1 1})$ de los parámetros θ_1 y θ_2 , respectivamente.

- Se obtienen B muestras bootstrap (aleatorias y con reemplazamiento) de X_1 y X_2 :

$$X_{1(1)}, X_{1(2)},...,X_{1(B)}$$

$$X_{2(1)}, X_{2(2)},...,X_{2(B)}$$

- Se calcula para $b \in \{1,...,B\}$ la replica bootstrap b-esima de los estimadores $\widehat{\theta}_1$ y $\widehat{\theta}_2$ como:

$$\widehat{\theta}_{1(b)} = \widehat{\theta}(X_{1(b)})$$

$$\widehat{\theta}_{2(b)} = \widehat{\theta}(X_{2(b)})$$

  Así que se tiene:

$$\widehat{\theta}_{1\,,\,boot} = \left(\widehat{\theta}_{1(1)},...,\widehat{\theta}_{1(B)}\right)$$

$$\widehat{\theta}_{2\,,\,boot} = \left(\widehat{\theta}_{2(1)},...,\widehat{\theta}_{2(B)}\right)$$

Sea $Q(\alpha\,,\,\widehat{\theta}_{1\,,\,boot} - \widehat{\theta}_{2\,,\,boot})$ el cuantil de orden α de la variable $\widehat{\theta}_{1\,,\,boot} - \widehat{\theta}_{2\,,\,boot}$

Por tanto, se cumple lo siguiente:

$$\frac{\#\,\left\{\, b=1,...,B \;\;/\;\; \widehat{\theta}_{1(b)} - \widehat{\theta}_{2(b)} \,\leq\, Q(\alpha\,,\,\widehat{\theta}_{1\,,\,boot} - \widehat{\theta}_{2\,,\,boot}) \,\right\}}{B} \;=\; \alpha$$

- El intervalo cuantil-bootstrap para la diferencia de parametros $\theta_1 - \theta_2$ a un nivel $1-\alpha$ es :

$$IC(\theta_1 - \theta_2)^{boot}_{1-\alpha} \;=\; \left[\; Q(\alpha/2\;,\;\widehat{\theta}_{1\,,\,boot} - \widehat{\theta}_{2\,,\,boot}) \;\;;\;\; Q(1-\alpha/2\;,\;\widehat{\theta}_{1\,,\,boot} - \widehat{\theta}_{2\,,\,boot}) \;\right]$$

Los intervalos cuantil-bootstrap pueden conducir a estimaciones del intervalo de confianza algo erráticas cuando el estimador del parametro de interés es sesgado.

Se pueden considerar una versión mejorada del intervalo cuantil-bootstrap llamada BCa, abreviatura que procede de sesgo-corregido (bias-corrected) y acelerado (accelerated).

11.3 Intervalo BCa-bootstrap

En la determinación de los intervalos BCa-bootstrap juegan un rol central dos cantidades:
$\Rightarrow \quad \hat{z}_0$ y $\hat{a}$

$\hat{z}_0$ se introduce para corregir el sesgo del estimador $\hat{\theta}$

$\hat{z}_0$ se define como :

$$\hat{z}_0 \ = \ F^{-1}_{N(0,1)} \left(\frac{\# \left\{ \ b = 1, ..., B \ \ / \ \ \widehat{\theta}_{(b)} \ \leq \ \hat{\theta}(X) \ \right\}}{B} \right)$$

Aclaremos esto un poco:

Si ρ es la proporción de replicas bootstrap del estimador $\hat{\theta}_{(1)}, ..., \hat{\theta}_{(B)}$ que son menores o iguales que la estimacion $\hat{\theta}(X)$, entonces:

$$\rho \ = \ \frac{\# \left\{ \ b = 1, ..., B \ \ / \ \ \widehat{\theta}_{(b)} \ \leq \ \hat{\theta}(X) \ \right\}}{B}$$

Por tanto:

$$\hat{z}_0 \ = \ F^{-1}_{N(0,1)}(\rho) \quad \Rightarrow \quad F_{N(0,1)}(\hat{z}_0) \ = \ P\,(\ N(0,1) \leq \hat{z}_0 \ ) \ = \ \rho$$

En conclusión:

$\hat{z}_0$ es el cuantil de orden ρ de la distribucion $N(0,1) \quad \Rightarrow \quad \hat{z}_0 \ = \ Q\,(\ \rho\ ,\ N(0,1)\ )$

La segunda cantidad, $\hat{a}$, denominada aceleración, corrige el caso en el que el error estandar del estimador del parámetro de interés $s.e.(\hat{\theta})$ no sea constante, y se define en términos de estimaciones Jacknife.

Recordemos el contexto Jacknife:

Se define $X_{(r)}$ como la muestra que contiene todos los valores de la muestra $X = (x_1, ..., x_n)$ del la variable aleatoria de interés $\mathcal{X}$ excepto el valor x_r

Es decir:

$$X_{(r)} = (\, x_1 \, , \, x_2 \, , .., \, x_{r-1} \, , \, x_{r+1} \, , .., \, x_n \,)$$

para $r = 1, ..., n$

Se define la replica r-esima del estimador $\widehat{\theta}$ como:

$$\widehat{\theta}_{(r)} = \widehat{\theta}(X_{(r)}) = \widehat{\theta}(\, x_1 \, , \, x_2 \, , .., \, x_{r-1} \, , \, x_{r+1} \, , .., \, x_n \,)$$

Teniendo todo esto en cuenta, $\hat{a}$ se define como sigue:

$$\hat{a} \;=\; \frac{\sum_{r=1}^{n} \left(\, \hat{\theta}_{(\cdot)} - \hat{\theta}_{(r)} \right)^3}{6 \cdot \left[\sum_{r=1}^{n} \left(\, \hat{\theta}_{(\cdot)} - \hat{\theta}_{(r)} \right)^2 \right]^{3/2}}$$

donde:

$$\hat{\theta}_{(\cdot)} = \frac{1}{n} \cdot \sum_{r=1}^{n} \hat{\theta}_{(r)}$$

El intervalo BCa-bootstrap de nivel $1 - \alpha$ es:

$$\left[\; Q(\alpha_1 \; , \; \hat{\theta}_{boot}) \;\; ; \;\; Q(\alpha_2 \; , \; \hat{\theta}_{boot}) \;\right]$$

donde:

$$\alpha_1 \;=\; F_{N(0,1)} \left(\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1 - \hat{a} \cdot (\hat{z}_0 + z_{1-\alpha/2})} \right)$$

$$\alpha_2 \;=\; F_{N(0,1)} \left(\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1 - \hat{a} \cdot (\hat{z}_0 + z_{\alpha/2})} \right)$$

Teniendo en cuenta que:

- z_α el valor tal que $P(N(0,1) \leq z_\alpha) = \alpha$

- $\hat{a} \ = \ \dfrac{\sum_{r=1}^{n} \left(\hat{\theta}_{(\cdot)} - \hat{\theta}_{(r)} \right)^3}{6 \cdot \left[\sum_{r=1}^{n} \left(\hat{\theta}_{(\cdot)} - \hat{\theta}_{(r)} \right)^2 \right]^{3/2}}$

- $\hat{z}_0 \ = \ Q\,(\ \rho\ ,\ N(0,1)\ )$

- $\rho \ = \ \dfrac{\#\ \left\{\ b = 1, ..., B \ \ / \ \ \widehat{\theta}_{(b)} \ \leq \ \hat{\theta}(X)\ \right\}}{B}$

Si $\hat{z}_0 = \hat{a} = 0$, entonces:

$$\alpha_1 \ = \ F_{N(0,1)}(z_{1-\alpha/2}) \ = \ \alpha/2$$

$$\alpha_2 \ = \ F_{N(0,1)}(z_{\alpha/2}) \ = \ 1 - \alpha/2$$

Por lo que en este caso particular el intervalo BCa coincide con el intervalo percentil.

El valor de $\hat{z}_0$ traslada el intervalo a la derecha o a la izquierda, y $\hat{a}$ hace que sea más ancho o más estrecho.

Con este intervalo se recomienda usar $B \geq 1000$.

11.4 Intervalos cuantil-bootstrap en Python

```
def cuantil_boot_interval(Variable1, Variable2, alpha, estimator , B,
↪  q=0.75, random_seed=123):

    from itertools import chain

######################################################################

    np.random.seed(random_seed)

######################################################################

    def Bootstrap_sample(Variable):

        from sklearn.utils import resample

        sample = resample( Variable, n_samples=len(Variable))

        return sample

######################################################################

    replicas_estimador = []


    if estimator == 'mean':

        for b in range(0, B):

            replicas_estimador.append( np.mean(
↪  Bootstrap_sample(Variable1) ) )

        estimation = np.mean(Variable1)

######################################################################
#
    if estimator == 'median':

        for b in range(0, B):

            replicas_estimador.append( np.median(
↪  Bootstrap_sample(Variable1) ) )

        estimation = np.median(Variable1)

######################################################################

    if estimator == 'std':

        for b in range(0, B):

            replicas_estimador.append( np.std( Bootstrap_sample(Variable1)
↪  ) )
```

```
        estimation = np.std(Variable1)

######################################################################

    if estimator == 'skewness':

        from scipy.stats import skew

        for b in range(0, B):

            replicas_estimador.append( skew( Bootstrap_sample(Variable1) )
↪   )

        estimation = skew(Variable1)

######################################################################

    if estimator == 'kurtosis':

        from scipy.stats import kurtosis

        for b in range(0, B):

            replicas_estimador.append( kurtosis(
↪   Bootstrap_sample(Variable1) ) )

        estimation = kurtosis(Variable1)

######################################################################

    if estimator == 'quantile':

        for b in range(0, B):

            replicas_estimador.append( np.quantile(
↪   Bootstrap_sample(Variable1) , q=q ) )

        estimation = np.quantile(Variable1 , q=q)

######################################################################

    if estimator == 'proportion': # Variable1 debe ser una variable
    ↪   categorica **binaria**.

        for b in range(0, B):

            replicas_estimador.append( np.mean(
↪   Bootstrap_sample(Variable1) ) )

        estimation = np.mean(Variable1)
```

```
#################################################################

#################################################################

    replicas_estimador_1 , replicas_estimador_2 = [] , []


    if estimator == 'mean_diff':

        for b in range(0, B):

            replicas_estimador_1.append( np.mean(
↪   Bootstrap_sample(Variable1) ) )

            replicas_estimador_2.append( np.mean(
↪   Bootstrap_sample(Variable2) ) )

        replicas_estimador_diff = np.array(replicas_estimador_1) -
↪   np.array(replicas_estimador_2)

        estimation = np.mean(Variable1) - np.mean(Variable2)

#################################################################

    if estimator == 'median_diff':

        for b in range(0, B):

            replicas_estimador_1.append( np.median(
↪   Bootstrap_sample(Variable1) ) )

            replicas_estimador_2.append( np.median(
↪   Bootstrap_sample(Variable2) ) )

        replicas_estimador_diff = np.array(replicas_estimador_1) -
↪   np.array(replicas_estimador_2)

        estimation = np.median(Variable1) - np.median(Variable2)

#################################################################

    if estimator == 'std_diff':

        for b in range(0, B):

            replicas_estimador_1.append( np.std(
↪   Bootstrap_sample(Variable1) ) )

            replicas_estimador_2.append( np.std(
↪   Bootstrap_sample(Variable2) ) )

        replicas_estimador_diff = np.array(replicas_estimador_1) -
↪   np.array(replicas_estimador_2)
```

```
        estimation = np.std(Variable1) - np.std(Variable2)

######################################################################

    if estimator == 'quantile_diff':

        for b in range(0, B):

            replicas_estimador_1.append( np.quantile(
↪   Bootstrap_sample(Variable1), q=q) )

            replicas_estimador_2.append( np.quantile(
↪   Bootstrap_sample(Variable2), q=q ) )

        replicas_estimador_diff = np.array(replicas_estimador_1) -
↪   np.array(replicas_estimador_2)

        estimation = np.quantile(Variable1, q=q) - np.quantile(Variable2,
↪   q=q)

######################################################################

    if estimator == 'skewness_diff':

        from scipy.stats import skew

        for b in range(0, B):

            replicas_estimador_1.append( skew( Bootstrap_sample(Variable1)
↪   ) )

            replicas_estimador_2.append( skew( Bootstrap_sample(Variable2)
↪   ) )

        replicas_estimador_diff = np.array(replicas_estimador_1) -
↪   np.array(replicas_estimador_2)

        estimation = skew(Variable1) - skew(Variable2)

######################################################################

    if estimator == 'kurtosis_diff':

        from scipy.stats import kurtosis

        for b in range(0, B):

            replicas_estimador_1.append( kurtosis(
↪   Bootstrap_sample(Variable1) ) )

            replicas_estimador_2.append( kurtosis(
↪   Bootstrap_sample(Variable2) ) )
```

```
        replicas_estimador_diff = np.array(replicas_estimador_1) -
↪   np.array(replicas_estimador_2)

        estimation = kurtosis(Variable1) - kurtosis(Variable2)

######################################################################

    if estimator == 'proportion_diff': # Variable1 y Variable2 deben ser
    ↪   variables categoricas **binarias**.

        for b in range(0, B):

            replicas_estimador_1.append( np.mean(
↪   Bootstrap_sample(Variable1) ) )

            replicas_estimador_2.append( np.mean(
↪   Bootstrap_sample(Variable2) ) )

        replicas_estimador_diff = np.array(replicas_estimador_1) -
↪   np.array(replicas_estimador_2)

        estimation = np.mean(Variable1) - np.mean(Variable2)

######################################################################
######################################################################

    if estimator in ['mean','median','std','quantile','kurtosis',
    'skewness','proportion']:

        L1_1 = np.quantile( replicas_estimador  , q=alpha/2)

        L2_1 = np.quantile( replicas_estimador  , q=1-alpha/2)

        interval = [L1_1 , L2_1]

######################################################################

    if estimator in ['mean_diff','median_diff','std_diff','quantile_diff',
    'kurtosis_diff','skewness_diff', 'proportion_diff']:

        L1_2 = np.quantile( replicas_estimador_diff  , q=alpha/2)

        L2_2 = np.quantile( replicas_estimador_diff  , q=1-alpha/2)

        interval = [L1_2 , L2_2]

######################################################################
######################################################################

    return interval , estimation
```

Intervalo para la media:

```
interval , estimation = cuantil_boot_interval(Variable1=X, Variable2='no',
↪ alpha=0.05, estimator='mean' , B=20000, random_seed=123)
```

```
interval
```

```
[5.274553626813773, 15.146375204290525]
```

```
estimation
```

```
10.199071616256777
```

Comparación con el intervalo de confianza clásico frecuentista:

```
def CI_Mean(Variable , alpha=0.05):

    n = len(Variable)

    t_alpha_medios = scipy.stats.t.ppf( 1 - alpha/2 , df=n-1)

    X_mean = Variable.mean()

    X_cuasi_var = Variable.std()**2

    # std() esta definida por defecto como la cuasi-desviacion-tipica

    L1 = X_mean - t_alpha_medios * np.sqrt(X_cuasi_var/n)

    L2 = X_mean + t_alpha_medios * np.sqrt(X_cuasi_var/n)

    interval = [L1 , L2]

    return interval , X_mean
```

```
interval , X_mean = CI_Mean(X , alpha=0.05)
```

```
interval
```

```
[5.127765678327374, 15.27037755418618]
```

Intervalo para la desviación típica:

```
interval , estimation = cuantil_boot_interval(Variable1=X, Variable2='no',
↪  alpha=0.05, estimator='std' , B=20000, random_seed=123)
```

```
interval
```

```
[14.427214668666169, 20.756590926218593]
```

```
estimation
```

```
17.844342210422187
```

Comparación con el intervalo de confianza clásico frecuentista:

```
def CI_Variance(Variable , alpha=0.05):

    n = len(Variable)

    chi_alpha_medios = scipy.stats.chi2.ppf( 1 - alpha/2 , df=n-1)

    chi_1_alpha_medios = scipy.stats.chi2.ppf(alpha/2 , df=n-1)

    X_cuasi_var = Variable.std()**2

    X_var = ( (n-1)/n )*X_cuasi_var

    # std() esta definida por defecto como la cuasi-desviacion-tipica

    L1 =  (n*X_var) / chi_alpha_medios

    L2 = (n*X_var) / chi_1_alpha_medios

    interval = [L1 , L2]

    return interval , X_var
```

```
interval , X_var = CI_Variance(Variable=X , alpha=0.05)
```

```
np.sqrt(interval)
```

```
array([14.90598594, 22.23643012])
```

Intervalo para la mediana:

```
interval , estimation = cuantil_boot_interval(Variable1=X, Variable2='no',
↪ alpha=0.05, estimator='median' , B=20000, random_seed=123)
```

```
interval
```

```
[2.4028635430970975, 15.99323919612726]
```

```
estimation
```

```
8.23916733155832
```

Intervalo para el coeficiente de asimetria:

```
interval , estimation = cuantil_boot_interval(Variable1=X, Variable2='no',
↪ alpha=0.05, estimator='skewness' , B=20000, random_seed=123)
```

```
interval
```

```
[-0.43998686459545305, 0.5624134382838676]
```

```
estimation
```

```
0.025587358812510053
```

Intervalo para el coeficiente de curtosis:

```
interval , estimation = cuantil_boot_interval(Variable1=X, Variable2='no',
↪ alpha=0.05, estimator='kurtosis' , B=20000, random_seed=123)
```

```
interval
```

```
[-0.9952663071794702, 0.38601833503542765]
```

```
estimation
```

```
-0.37420768292897266
```

Intervalo para los cuantiles:

```
interval , estimation = cuantil_boot_interval(Variable1=X, Variable2='no',
↪  alpha=0.05, estimator='quantile' , B=20000, random_seed=123, q=0.75)
```

```
interval
```

```
[15.078835764997024, 28.98904388058301]
```

```
estimation
```

```
23.835596841535178
```

Intervalo para la proporción:

```
X_dummy = np.random.uniform(low=0 , high=1, size=50).round()
```

```
interval , estimation = cuantil_boot_interval(Variable1=X_dummy,
↪  Variable2='no', alpha=0.05, estimator='proportion' , B=20000,
↪  random_seed=123)
```

```
interval
```

```
[0.46, 0.74]
```

```
estimation
```

```
0.6
```

Intervalo para la diferencia de medias:

```
np.random.seed(123)

X_1 = np.random.normal(loc=10, scale=15, size=50)

X_2 = np.random.normal(loc=13, scale=15, size=100)
```

```
interval , estimation = cuantil_boot_interval(Variable1=X_1,
↪  Variable2=X_2, alpha=0.05, estimator='mean_diff' , B=20000,  q=0.75,
↪  random_seed=123)
```

```
interval
```

```
[-9.788335434742939, 1.810648408844556]
```

```
estimation
```

```
-4.001757444976697
```

Comparación con el intervalo de confianza clásico frecuentista:

```
def CI_Mean_Diference(Variable1 , Variable2 , alpha=0.05):

    X1 = Variable1
    X2 = Variable2

    n1 = len(X1)
    n2 = len(X2)

    X1_mean = X1.mean()
    X2_mean = X2.mean()

    X1_cuasi_var = X1.std()**2
    X2_cuasi_var = X2.std()**2

    X1_var = ( (n1-1)/n1 )*X1_cuasi_var
    X2_var = ( (n2-1)/n2 )*X2_cuasi_var

    v = ( X1_var/n1 + X2_var/n2 )**2 / ( (X1_var/n1)**2 / (n1-1)  +
↪   (X2_var/n2)**2 / (n2-1)  )

    t_alpha_medios = scipy.stats.chi.ppf( 1 - alpha/2 , df=v)


    L1 =  (X1_mean - X2_mean) - t_alpha_medios * np.sqrt(X1_var/n1 +
↪   X2_var/n2)

    L2 =  (X1_mean - X2_mean) + t_alpha_medios * np.sqrt(X1_var/n1 +
↪   X2_var/n2)

    interval = [L1 , L2]

    return interval , (X1_mean - X2_mean)
```

```
interval , estimation = CI_Mean_Diference(Variable1=X_1 , Variable2=X_2 ,
↪   alpha=0.05)
```

```
interval
```

```
[-35.535058586780195, 27.531543696826805]
```

```
estimation
```

```
-4.001757444976697
```

Intervalo para la diferencia de medianas:

```
interval , estimation = cuantil_boot_interval(Variable1=X_1,
↪  Variable2=X_2, alpha=0.05, estimator='median_diff' , B=20000,  q=0.75,
↪  random_seed=123)
```

```
interval
```

```
[-13.881285812364158, 3.1918214145850397]
```

```
estimation
```

```
-6.812252818435496
```

Intervalo para la diferencia de desviaciones típicas:

```
interval , estimation = cuantil_boot_interval(Variable1=X_1,
↪  Variable2=X_2, alpha=0.05, estimator='std_diff' , B=20000,  q=0.75,
↪  random_seed=123)
```

```
interval
```

```
[-1.440066276291272, 5.750492255040537]
```

```
estimation
```

```
2.297921770341434
```

Intervalo para la diferencia de cuantiles:

```
interval , estimation = cuantil_boot_interval(Variable1=X_1,
↪  Variable2=X_2, alpha=0.05, estimator='quantile_diff' , B=20000,
↪  q=0.75, random_seed=123)
```

```
interval
```

```
[-12.249938782339909, 4.819498365231816]
```

```
estimation
```

```
-1.2929002407485939
```

Intervalo para la diferencia de curtosis:

```
interval , estimation = cuantil_boot_interval(Variable1=X_1,
↪  Variable2=X_2, alpha=0.05, estimator='kurtosis_diff' , B=20000,
↪  q=0.75, random_seed=123)
```

```
interval
```

```
 [-0.504742036075702, 1.1320430654741647]
```

```
estimation
```

```
0.2817371126835462
```

Intervalo para la diferencia de asimetría:

```
interval , estimation = cuantil_boot_interval(Variable1=X_1,
↪  Variable2=X_2, alpha=0.05, estimator='skewness_diff' , B=20000,
↪  q=0.75, random_seed=123)
```

```
interval
```

```
[-0.5445430937619894, 0.6394836839729549]
```

```
estimation
```

```
0.019079758603481104
```

Intervalo para la diferencia de proporciones:

```
np.random.seed(123)

X_dummy_1 = np.random.uniform(low=0 , high=1, size=40).round()

X_dummy_2 = np.random.uniform(low=0 , high=1, size=300).round()
```

```
interval , estimation = cuantil_boot_interval(Variable1=X_dummy_1,
↪  Variable2=X_dummy_2, alpha=0.05, estimator='proportion_diff' ,
↪  B=20000,  q=0.75, random_seed=123)
```

```
interval
```

```
[-0.22499999999999998, 0.10666666666666663]
```

```
estimation
```

```
-0.06
```

11.5 Intervalos BCa en Python

12 Contrastes de hipotesis basados en bootstrap

Existen múltiples aproximaciones a los contrastes de hipótesis desde una perspectiva bootstrap. En este caso nos aproximaremos usando los intervalos cuantil-bootstrap, por simplicidad.

Vamos a diferenciar contrastes de hipótesis sobre una población y sobre dos poblaciones.

12.1 Contraste de hipótesis sobre una población

Tenemos una v.a. $\mathcal{X} \sim D(\theta)$, una m.a.s. $(\mathcal{X}_1, ..., \mathcal{X}_n)$ de $\mathcal{X}$ y un estimador $\widehat{\theta}(\mathcal{X}_1, ..., \mathcal{X}_n)$ del parámetro θ

Además tenemos una muestra de observaciones $X = (x_1, ..., x_n)$ de la variable de interés $\mathcal{X}$, por lo que tenemos la estimación $\widehat{\theta}(X) = \widehat{\theta}(x_1, ..., x_n)$ del parámetro θ.

Se quieren resolver los siguientes contrastes:

$$
\begin{array}{lll}
H_0 : \theta = \theta_0 & H_0 : \theta = \theta_0 & H_0 : \theta = \theta_0 \\
H_1 : \theta \neq \theta_0 & H_1 : \theta < \theta_0 & H_1 : \theta > \theta_0
\end{array}
$$

La regla de decisión para resolver estos contrastes basada en los intervalos cuantil-bootstrap es la siguiente:

Para un nivel de significación α, partimos del intervalo cuantil-bootstrap del parametro θ para un nivel de confianza $1 - \alpha \;\; \Rightarrow \;\; IC(\theta)^{boot}_{1-\alpha} = [L1, L2]$

- Caso $H_0 : \theta = \theta_0$ vs $H_1 : \theta \neq \theta_0$

$$\text{Rechazar } H_0 \;\; \Leftrightarrow \;\; \theta_0 \notin IC(\theta)^{boot}_{1-\alpha}$$

- Caso $H_0 : \theta = \theta_0$ vs $H_1 : \theta < \theta_0$

$$\text{Rechazar } H_0 \;\; \Leftrightarrow \;\; IC(\theta)^{boot}_{1-\alpha} << \theta_0 \;\; \Leftrightarrow \;\; L2 < \theta_0$$

- Caso $H_0 : \theta = \theta_0$ vs $H_1 : \theta > \theta_0$

$$\text{Rechazar } H_0 \;\; \Leftrightarrow \;\; IC(\theta)^{boot}_{1-\alpha} >> \theta_0 \;\; \Leftrightarrow \;\; L1 > \theta_0$$

12.2 Contraste de hipótesis sobre dos poblaciones

Tenemos dos v.a's $\mathcal{X}_1 \sim D_1(\theta_1)$ y $\mathcal{X}_2 \sim D_2(\theta_2)$, una m.a.s. $(\mathcal{X}_{11}, ..., \mathcal{X}_{n_1 1})$ de $\mathcal{X}_1$, otra m.a.s. $(\mathcal{X}_{12}, ..., \mathcal{X}_{n_2 2})$ de $\mathcal{X}_2$ y un par de estimadores $\widehat{\theta}_1(\mathcal{X}_{11}, ..., \mathcal{X}_{n_1 1})$ y $\widehat{\theta}_2(\mathcal{X}_{12}, ..., \mathcal{X}_{n_2 2})$ de los parámetros θ_1 y θ_2 , respectivamente.

Además tenemos una muestras de observaciones $X_1 = (x_{11}, ..., x_{n_1 1})$ de la v.a. $\mathcal{X}_1$ y otra $X_1 = (x_{11}, ..., x_{n_1 1})$ de $\mathcal{X}_2$, por lo que tenemos las estimaciones $\widehat{\theta}_1(X_1) = \widehat{\theta}_1(x_{11}, ..., x_{n_1 1})$ de los parámetros θ_1 y θ_2 , respectivamente.

Se quieren resolver los siguientes contrastes:

$$
\begin{array}{lll}
H_0 : \theta_1 = \theta_2 & H_0 : \theta_1 = \theta_2 & H_0 : \theta_1 = \theta_2 \\
\\
H_1 : \theta_1 \neq \theta_2 & H_1 : \theta_1 < \theta_2 & H_1 : \theta_1 > \theta_2
\end{array}
$$

La regla de decisión para resolver estos contrastes basada en los intervalos cuantil-bootstrap es la siguiente:

Para un nivel de significación α.

Partimos del intervalo cuantil-bootstrap de la diferencia de parámetros $\theta_1 - \theta_2$ para un nivel de confianza $1 - \alpha \quad \Rightarrow \quad IC(\theta_1 - \theta_2)_{1-\alpha}^{boot} = [L1, L2]$

- Caso $H_0 : \theta_1 = \theta_2$ vs $H_1 : \theta_1 \neq \theta_2$

$$\text{Rechazar } H_0 \quad \Leftrightarrow \quad 0 \notin IC(\theta_1 - \theta_2)_{1-\alpha}^{boot}$$

- Caso $H_0 : \theta_1 = \theta_2$ vs $H_1 : \theta_1 < \theta_2$

$$\text{Rechazar } H_0 \quad \Leftrightarrow \quad IC(\theta_1 - \theta_2)_{1-\alpha}^{boot} << 0 \quad \Leftrightarrow \quad L2 \, < \, 0$$

- Caso $H_0 : \theta_1 = \theta_2$ vs $H_1 : \theta_1 > \theta_2$

$$\text{Rechazar } H_0 \quad \Leftrightarrow \quad IC(\theta_1 - \theta_2)_{1-\alpha}^{boot} >> 0 \quad \Leftrightarrow \quad L1 \, > \, 0$$

12.3 Contrastes de hipotesis bootstrap en Python

```
def bootstrap_cuantil_test(Variable1, Variable2, estimator, H1_type,
↪ theta_0,  alpha , B, random_seed, q):


    interval , estimation = cuantil_boot_interval(Variable1=Variable1,
↪   Variable2=Variable2, alpha=alpha, estimator=estimator , B=B,
↪   random_seed=random_seed, q=q)


    if estimator in
    ↪  ['mean','median','std','quantile','kurtosis','skewness','proportion']:


        if H1_type == 'greater':

            if interval[0] > theta_0 : result = 'Reject H0: theta =
            ↪  theta_0  -->  Accept H1: theta > theta_0'

            else : result = 'Not reject H0: theta = theta_0  -->  Not
            ↪  accept H1: theta > theta_0'


        if H1_type == 'less':

            if interval[1] < theta_0 : result = 'Reject H0: theta =
            ↪  theta_0  -->  Accept H1: theta < theta_0'

            else : result = 'Not reject H0: theta = theta_0  -->  Not
            ↪  accept H1: theta < theta_0'


        if H1_type == 'two.sided':

            if (interval[1] < theta_0)  | (interval[0] > theta_0) : result
            ↪  = 'Reject H0: theta = theta_0  -->  Accept H1: theta =/=
            ↪  theta_0'

            else : result = 'Not reject H0: theta = theta_0  -->  Not
            ↪  accept H1: theta =/= theta_0'



    if estimator in ['mean_diff','median_diff','std_diff','quantile_diff',
    'kurtosis_diff','skewness_diff', 'proportion_diff']:


        if H1_type == 'greater':

            if interval[0] > 0 : result = 'Reject H0: theta_1 = theta_2
            ↪  -->  Accept H1: theta_1 > theta_2'
```

```
            else : result = 'Not reject H0: theta_1 = theta_2 -->  Not
            ↪  accept H1: theta_1 > theta_2'


        if H1_type == 'less':

            if interval[1] < 0 : result = 'Reject H0: theta_1 = theta_2
            ↪  -->  Accept H1: theta_1 < theta_2'

            else : result = 'Not reject H0: theta_1 = theta_2  -->  Not
            ↪  accept H1: theta_1 < theta_2'


        if H1_type == 'two.sided':

            if (interval[1] < 0)  | (interval[0] >  0)  : result = 'Reject
            ↪  H0: theta_1 = theta_2  -->  Accept H1: theta_1 =/=
            ↪  theta_2'

            else : result = 'Not reject H0: theta_1 = theta_2  -->  Not
            ↪  accept H1: theta_1 =/= theta_2'



    return result , interval
```

```
np.random.seed(123)

X_1 = np.random.normal(loc=62, scale=25, size=150)

X_2 = np.random.normal(loc=80, scale=25, size=150)
```

Contraste para la media de una población:

```
np.mean(X_1)
```

```
63.44484985484652
```

```
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪  Variable2='no', estimator='mean', H1_type='greater', theta_0=50,
↪  alpha=0.05 , B=1000, random_seed=123, q='no')
```

```
resultado
```

```
'Reject H0: theta = theta_0  -->  Accept H1: theta > theta_0'
```

```
intervalo
```

```
[59.300517176372914, 68.0853420655662]
```

```
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪  Variable2='no', estimator='mean', H1_type='greater', theta_0=60,
↪  alpha=0.05 , B=1000, random_seed=123, q='no')
```

```
resultado
```

```
'Not reject H0: theta = theta_0  -->  Not accept H1: theta > theta_0'
```

```
bootstrap_cuantil_test(Variable1=X_1, Variable2='no', estimator='mean',
↪  H1_type='two.sided', theta_0=60,  alpha=0.05 , B=1000,
↪  random_seed=123, q='no')
```

```
resultado
```

```
'Not reject H0: theta = theta_0  -->  Not accept H1: theta =/= theta_0'
```

Contraste para la mediana de una población:

```

```

```

resultado , intervalo =
```

```
resultado
```

```
intervalo
```

Contraste para la desviación típica de una población:

```
np.std(X_1)
```

```
27.258568783722534
```

```
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪ Variable2='no', estimator='std', H1_type='less', theta_0=30,
↪ alpha=0.05 , B=1000, random_seed=123, q='no')
```

```
resultado
```

```
'Reject H0: theta = theta_0  -->  Accept H1: theta < theta_0'
```

```
intervalo
```

```
[24.53136884527364, 29.84312683035364]
```

```
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪ Variable2='no', estimator='std', H1_type='less', theta_0=26,
↪ alpha=0.05 , B=1000, random_seed=123, q='no')
```

```
resultado
```

```
'Not reject H0: theta = theta_0  -->  Not accept H1: theta < theta_0'
```

Contraste para los cuantiles de una población:

```
np.quantile(X_1, q=0.6)
```

```
70.37736516880977
```

```
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪ Variable2='no', estimator='quantile', H1_type='greater', theta_0=50,
↪ alpha=0.05 , B=1000, random_seed=123, q=0.6)
```

```
resultado
```

```
'Reject H0: theta = theta_0  -->  Accept H1: theta > theta_0'
```

```
intervalo
```

```
[65.5777835208156, 76.34514656012644]
```

Contraste para la curtosis de una población:

```
kurtosis(X_1)
```

```
-0.4799399747044939
```

```
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪  Variable2='no', estimator='kurtosis', H1_type='less', theta_0=0,
↪  alpha=0.05 , B=1000, random_seed=123, q='no')
```

```
resultado
```

```
'Reject H0: theta = theta_0  -->  Accept H1: theta < theta_0'
```

```
intervalo
```

```
[-0.8726641157248832, -0.03722884140184115]
```

Contraste para la asimetría de una población:

```
skew(X_1)
```

```
0.0030735881326157516
```

```
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪  Variable2='no', estimator='skewness', H1_type='two.sided', theta_0=0,
↪  alpha=0.05 , B=1000, random_seed=123, q='no')
```

```
resultado
```

```
'Not reject H0: theta = theta_0  -->  Not accept H1: theta =/= theta_0'
```

```
intervalo
```

```
[-0.26701282609616306, 0.27862278159471443]
```

Contraste para la media de dos poblaciones:

```
np.mean(X_1)
```

```
63.44484985484652
```

```
np.mean(X_2)
```

```
77.09585348416586
```

```
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪ Variable2=X_2, estimator='mean_diff', H1_type='greater', theta_0='no',
↪ alpha=0.05 , B=1000, random_seed=123, q='no')
```

```
resultado
```

```
'Not reject H0: theta_1 = theta_2 -->  Not accept H1: theta_1 > theta_2'
```

```
intervalo
```

```
[-19.673858833181058, -7.98554491914148]
```

```
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪ Variable2=X_2, estimator='mean_diff', H1_type='less', theta_0='no',
↪ alpha=0.05 , B=1000, random_seed=123, q='no')
```

```
resultado
```

```
'Reject H0: theta_1 = theta_2  -->  Accept H1: theta_1 < theta_2'
```

Contraste para la desviación típica de dos poblaciones:

```
np.std(X_1)
```

```
27.258568783722534
```

```
np.std(X_2)
```

```
23.76463767935987
```

```python
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪ Variable2=X_2, estimator='std_diff', H1_type='greater', theta_0='no',
↪ alpha=0.05 , B=1000, random_seed=123, q='no')
```

```python
resultado
```

```
'Not reject H0: theta_1 = theta_2 -->  Not accept H1: theta_1 > theta_2'
```

```python
intervalo
```

```
[-0.4051994934727355, 7.849597038849807]
```

```python
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪ Variable2=X_2, estimator='std_diff', H1_type='two.sided',
↪ theta_0='no',  alpha=0.05 , B=1000, random_seed=123, q='no')
```

```python
resultado
```

```
'Not reject H0: theta_1 = theta_2  -->  Not accept H1: theta_1 =/= theta_2'
```

Contraste para la mediana de dos poblaciones:

```
np.median(X_1)
```

```
62.99540261926173
```

```
np.median(X_2)
```

```
77.7284396794982
```

```
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪ Variable2=X_2, estimator='median_diff', H1_type='less', theta_0='no',
↪ alpha=0.05 , B=1000, random_seed=123, q='no')
```

```
resultado
```

```
 'Reject H0: theta_1 = theta_2  -->  Accept H1: theta_1 < theta_2'
```

```
intervalo
```

```
[-23.9186359851631, -7.0611603691117395]
```

Contraste para la curtosis de dos poblaciones:

```
kurtosis(X_1)
```

```
-0.4799399747044939
```

```
kurtosis(X_2)
```

```
 0.6321922906256292
```

```
bootstrap_cuantil_test(Variable1=X_1, Variable2=X_2,
↪  estimator='kurtosis_diff', H1_type='less', theta_0='no',  alpha=0.05 ,
↪  B=1000, random_seed=123, q='no')
```

```
resultado
```

```
'Not reject H0: theta_1 = theta_2  -->  Not accept H1: theta_1 < theta_2'
```

```
intervalo
```

```
[-2.145917163300932, 0.03602423829734017]
```

```
bootstrap_cuantil_test(Variable1=X_1, Variable2=X_2,
↪  estimator='kurtosis_diff', H1_type='less', theta_0='no',  alpha=0.1 ,
↪  B=1000, random_seed=123, q='no')
```

```
resultado
```

```
'Reject H0: theta_1 = theta_2  -->  Accept H1: theta_1 < theta_2'
```

```
intervalo
```

```
[-1.9699920673146376, -0.10952260840217383]
```

Contraste para los cuantiles de dos poblaciones:

```
np.quantile(X_1, 0.70)
```

```
79.99489543995588
```

```python
np.quantile(X_2, 0.70)
```

```
87.39568341964177
```

```python
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪  Variable2=X_2, estimator='quantile_diff', H1_type='less',
↪  theta_0='no',  alpha=0.05 , B=1000, random_seed=123, q=0.70)
```

```python
resultado
```

```
'Reject H0: theta_1 = theta_2  -->  Accept H1: theta_1 < theta_2'
```

```python
intervalo
```

```
[-17.07794746423375, -1.5905138337254294]
```

Contraste para la asimetría de dos poblaciones:

```python
skew(X_1)
```

```
0.0030735881326157516
```

```python
skew(X_2)
```

```
-0.026878668133587646
```

```python
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪  Variable2=X_2, estimator='skewness_diff', H1_type='two.sided',
↪  theta_0='no',  alpha=0.05 , B=1000, random_seed=123, q='no')
```

```python
resultado
```

```
'Not reject H0: theta_1 = theta_2  -->  Not accept H1: theta_1 =/= theta_2'
```

```python
intervalo
```

```
[-0.5568062631492594, 0.6399990949904947]
```

```python
resultado , intervalo = bootstrap_cuantil_test(Variable1=X_1,
↪  Variable2=X_2, estimator='skewness_diff', H1_type='greater',
↪  theta_0='no',  alpha=0.05 , B=1000, random_seed=123, q='no')
```

```python
resultado
```

```
'Not reject H0: theta_1 = theta_2 -->  Not accept H1: theta_1 > theta_2'
```

13 Bootstrap en Regresión Lineal

13.1 Botstrap en Regresión Lineal basado en residuos

Tenemos un modelo de regresión lineal:

$$y_i = x_i^t \cdot \beta + \varepsilon_i \quad , \quad \forall\, i \in \{1, ..., n\}$$

donde $x_i \in \mathbb{R}^{p+1}$, $\varepsilon_i \in \mathbb{R}$, $y_i \in \mathbb{R}$

El modelo de regresión lineal estimado por mínimos cuadrados ordinarios es:

$$\hat{y}_i = x_i^t \cdot \widehat{\beta} \quad , \quad \forall\, i \in \{1, ..., n\}$$

Donde:

$$\widehat{\beta} = (X \cdot X^t)^{-1} \cdot X^t \cdot y$$

Recordemos que en el modelo de regresión lineal los residuos estimados del modelo son:

$$\widehat{\varepsilon} = (\widehat{\varepsilon}_1, ..., \widehat{\varepsilon}_n)^t$$

donde:

$$\widehat{\varepsilon}_i \;=\; y_i - x_i^t \cdot \widehat{\beta} \;=\; y_i - \hat{y}_i \quad , \quad \forall\, i \in \{1, ..., n\}$$

Se toman B muestras *bootstrap* (aleatorias y con reemplazamiento) del vector de residuos estimados $\hat{\varepsilon}$ del modelo:

$$\widehat{\varepsilon}_{(1)} \, , ..., \, \widehat{\varepsilon}_{(B)}$$

Se generan B replicas *bootstrap* de las respuestas del siguiente modo:

$$Y_{(b)} \;=\; X \cdot \widehat{\beta} + \varepsilon_{(b)} \quad , \quad \forall\; b \in \{1, ..., B\}$$

Para cada $b \in \{1, ..., B\}$

Se entrenan el modelo de regresion lineal M con la muestra $\left(X, Y_{(b)}\right)$ de los predictores y la respuesta $\Rightarrow$ $\widehat{M}_{(b)}$

Se obtienen así B modelos de regresión lineal entrenados $\widehat{M}_{(1)} \, , ..., \, \widehat{M}_{(B)}$, que son las replicas bootstrap del modelo inicial (el entrenado con los datos iniciales).

Podemos usar estos modelos para obtener intervalos de confianza bootstrap de los coeficientes betas y de otros parámetros como el coeficiente de determinación (R-cuadrado).

13.1.1 Intervalo de confianza bootstrap para los coeficientes betas

Para cada modelo $M_{(b)}$ se tiene la estimación $\widehat{\beta}_{(b)}$ del vector de coeficientes betas, y con ello se tiene la estimación $\widehat{\beta}_{j(b)}$ del coeficiente β_j , para cada predictor.

Así que para cada estimador $\widehat{\beta}_j$ se tiene un vector de replicas bootstrap: $\widehat{\beta}_{j\,,boot} = \left(\widehat{\beta}_{j(1)}\,,\widehat{\beta}_{j(2)}\,,...,\widehat{\beta}_{j(B)}\right) \quad , \quad \forall\, j \in \{0,1,...,p\}$

Se puede usar la filosofía de los intervalos cuantil-bootstrap para obtener el intervalos bootstrap para los coeficientes betas del modelo:

$$IC(\beta_j)^{boot}_{1-\alpha} \;=\; \left[\; Q\left(\; \alpha/2 \;,\; \widehat{\beta}_{j\,,boot} \;\right) \;,\; Q\left(\; 1-\alpha/2 \;,\; \widehat{\beta}_{j\,,boot} \;\right) \;\right]$$

13.1.2 Intervalo de confianza bootstrap para los coeficientes betas

Para cada modelo $M_{(b)}$ se tiene la estimación $R^2_{adj\,(b)}$ del coeficiente de determinación ajustado.

Así que para el estimador $R^2_{adj\,,boot}$ se tiene un vector de replicas bootstrap: $R^2_{adj\,,boot} = \left(R^2_{adj\,(1)}\,,R^2_{adj\,(2)}\,,...,R^2_{adj\,(B)}\right) \quad , \quad \forall\, j \in \{0,1,...,p\}$

Se puede usar la filosofía de los intervalos cuantil-bootstrap para obtener el intervalos bootstrap para los coeficientes betas del modelo:

$$IC(R^2_{adj})^{boot}_{1-\alpha} \;=\; \left[\; Q\left(\; \alpha/2 \;,\; R^2_{adj\,,boot} \;\right) \;,\; Q\left(\; 1-\alpha/2 \;,\; R^2_{adj\,,boot} \;\right) \;\right]$$

13.2 Botstrap en Regresión Lineal basado en pares

En este caso se obtienen B muestras bootstrap (aleatorias y con reemplazamiento) de las observaciones de los predictores y la respuesta $(X,Y) = (\ (x_i, y_i)\ /\ i \in \{1,...,n\}\ )$

Con ello se obtienen las siguientes B muestras:

$$(X,Y)_{(1)}\ ,...,\ (X,Y)_{(B)}$$

Para cada $b \in \{1,...,B\}$

Se entrena el modelo de regresión lineal M con la muestra $(X,Y)_{(b)}$ de los predictores y la respueta $\Rightarrow$ $\widehat{M}_{(b)}$

Se obtienen así B modelos de regresión lineal entrenados $\widehat{M}_{(1)}, ..., \widehat{M}_{(B)}$, que son las replicas bootstrap del modelo inicial (el entrenado con los datos iniciales).

Podemos usar estos modelos para obtener intervalos de confianza bootstrap de los coeficientes betas y de otros parámetros como el coeficiente de determinación (R-cuadrado).

13.2.1 Intervalo de confianza bootstrap para los coeficientes betas

Para cada modelo $M_{(b)}$ se tiene la estimación $\widehat{\beta}_{(b)}$ del vector de coeficientes betas, y con ello se tiene la estimación $\widehat{\beta}_{j(b)}$ del coeficiente β_j , para cada predictor.

Así que para cada estimador $\widehat{\beta}_j$ se tiene un vector de replicas bootstrap: $\widehat{\beta}_{j\,,boot} = \left(\widehat{\beta}_{j(1)}\,,\widehat{\beta}_{j(2)}\,, ..., \widehat{\beta}_{j(B)}\right)$, $\forall\, j \in \{0,1,...,p\}$

Se puede usar la filosofía de los intervalos cuantil-bootstrap para obtener el intervalos bootstrap para los coeficientes betas del modelo:

$$IC(\beta_j)^{boot}_{1-\alpha} \ = \ \left[\ Q\left(\ \alpha/2\ ,\ \widehat{\beta}_{j\,,boot}\ \right)\ ,\ Q\left(\ 1-\alpha/2\ ,\ \widehat{\beta}_{j\,,boot}\ \right)\ \right]$$

13.2.2 Intervalo de confianza bootstrap para los coeficientes betas

Para cada modelo $M_{(b)}$ se tiene la estimación $R^2_{adj\,(b)}$ del coeficiente de determinación ajustado.

Así que para el estimador $R^2_{adj\,,boot}$ se tiene un vector de replicas bootstrap: $R^2_{adj\,,boot} = \left(R^2_{adj\,(1)}\,, R^2_{adj\,(2)}\,, ..., R^2_{adj\,(B)}\right)$, $\forall\, j \in \{0,1,...,p\}$

Se puede usar la filosofía de los intervalos cuantil-bootstrap para obtener el intervalos bootstrap para los coeficientes betas del modelo:

$$IC(R^2_{adj})^{boot}_{1-\alpha} \ = \ \left[\ Q\left(\ \alpha/2\ ,\ R^2_{adj\,,boot}\ \right)\ ,\ Q\left(\ 1-\alpha/2\ ,\ R^2_{adj\,,boot}\ \right)\ \right]$$

13.3 Regresión lineal bootstrap en Python

```
import pandas as pd
```

```
Data = pd.read_csv('House_Price_Regression.csv')

Data = Data.loc[:, ['latitude', 'longitude', 'price',
↪  'size_in_m_2','no_of_bedrooms', 'no_of_bathrooms', 'quality_recode']]

Data.head()
```

```
    latitude  longitude    price  size_in_m_2  no_of_bedrooms
0  25.113208  55.138932  2700000   100.242337               1
1  25.106809  55.151201  2850000   146.972546               2
2  25.063302  55.137728  1150000   181.253753               3
3  25.227295  55.341761  2850000   187.664060               2
4  25.114275  55.139764  1729200    47.101821               0

   no_of_bathrooms  quality_recode
0                2             2.0
1                2             2.0
2                5             2.0
3                3             1.0
4                1             2.0
```

```
Data['quality_recode'] = Data['quality_recode'].astype('category')
```

```
import sklearn

from sklearn.linear_model import LinearRegression
```

```
X = Data[['size_in_m_2', 'longitude', 'latitude', 'no_of_bedrooms',
↪  'no_of_bathrooms', 'quality_recode']]
Y = Data['price']
```

```
X.head()
```

```
   size_in_m_2  longitude   latitude  no_of_bedrooms  no_of_bathrooms
0   100.242337  55.138932  25.113208               1                2
1   146.972546  55.151201  25.106809               2                2
2   181.253753  55.137728  25.063302               3                5
3   187.664060  55.341761  25.227295               2                3
4    47.101821  55.139764  25.114275               0                1
```

```
  quality_recode
0            2.0
1            2.0
2            2.0
3            1.0
4            2.0
```

```
Y.head()
```

```
0    2700000
1    2850000
2    1150000
3    2850000
4    1729200
Name: price, dtype: int64
```

```
def varcharProcessing(X, varchar_process = "dummy_dropfirst"):

    dtypes = X.dtypes

    if varchar_process == "drop":
        X = X.drop(columns = dtypes[dtypes == np.object].index.tolist())

    elif varchar_process == "dummy":
        X = pd.get_dummies(X,drop_first=False)

    elif varchar_process == "dummy_dropfirst":
        X = pd.get_dummies(X,drop_first=True)

    else:
        X = pd.get_dummies(X,drop_first=True)

    X["intercept"] = 1
    cols = X.columns.tolist()
    cols = cols[-1:] + cols[:-1]
    X = X[cols]

    return X
```

```
X = varcharProcessing(X, varchar_process = "dummy_dropfirst")

X.head()
```

```
   intercept  size_in_m_2  longitude   latitude  no_of_bedrooms
0          1   100.242337  55.138932  25.113208               1
1          1   146.972546  55.151201  25.106809               2
2          1   181.253753  55.137728  25.063302               3
```

```
3          1   187.664060  55.341761  25.227295              2
4          1    47.101821  55.139764  25.114275              0

   no_of_bathrooms  quality_recode_1.0  quality_recode_2.0  quality_recode_3.0
0                2                   0                   1                   0
1                2                   0                   1                   0
2                5                   0                   1                   0
3                3                   1                   0                   0
4                1                   0                   1                   0
```

```python
def boot_interval_linear_regression(X, Y, method, parameter, j, B, alpha)
↪ :

    model_b_list , beta_hat_j_boot, R_2_adj_boot = [] , [] , []

    n = len(X)

    p = X.shape[1] - 1  # Si X no contiene el intercept --> p =
↪ X.shape[1]

    model = LinearRegression().fit(X, Y)

    residuals = Y - model.predict(X)

    def Bootstrap_sample(X):

        from sklearn.utils import resample

        sample = resample( X, n_samples=len(X))

        return sample


    beta_hat = np.concatenate( ( np.array([model.intercept_]) ,
↪ model.coef_[1:len(X)]) )

########################################################################

    if method == 'residuals':

        if parameter == 'beta' :

            for b in range(0, B):

                residuals_b = Bootstrap_sample(residuals)

                Y_b = X.to_numpy() @ beta_hat + residuals_b

                model_b = LinearRegression().fit(X, Y_b)

                beta_hat_j_boot.append( model_b.coef_[j] )
```

```
            L1 = np.quantile( beta_hat_j_boot  , q=alpha/2)

            L2 = np.quantile( beta_hat_j_boot  , q=1-alpha/2)

            interval = [L1,L2]

#######################################################################

        elif parameter == 'adj_R2' :

            for b in range(0, B):

                residuals_b = Bootstrap_sample(residuals)

                Y_b = X.to_numpy() @ beta_hat + residuals_b

                model_b = LinearRegression().fit(X, Y_b)

                R_2_adj_boot.append( 1 - (1 - model_b.score(X , Y_b))*(
↪   (n-1) / (n-p-1) ) )


            L1 = np.quantile( R_2_adj_boot  , q=alpha/2)

            L2 = np.quantile( R_2_adj_boot  , q=1-alpha/2)

            interval = [L1,L2]

#######################################################################

    if method == 'pairs' :

        X_Y = pd.concat([X,Y], axis=1)

        if parameter == 'beta' :

            for b in range(0, B):

                X_Y_b = Bootstrap_sample(X_Y)

                X = X_Y_b.iloc[: , 0:(X_Y_b.shape[1]-1)]

                Y = X_Y_b.iloc[: , X_Y_b.shape[1]-1]

                model_b = LinearRegression().fit(X, Y)

                beta_hat_j_boot.append( model_b.coef_[j] )


            L1 = np.quantile( beta_hat_j_boot  , q=alpha/2)

            L2 = np.quantile( beta_hat_j_boot  , q=1-alpha/2)
```

```
            interval = [L1,L2]
########################################################################

        elif parameter == 'adj_R2' :

            for b in range(0, B):

                X_Y_b = Bootstrap_sample(X_Y)

                X = X_Y_b.iloc[: , 0:(X_Y_b.shape[1]-1)]

                Y = X_Y_b.iloc[: , X_Y_b.shape[1]-1]

                model_b = LinearRegression().fit(X, Y)

                R_2_adj_boot.append( 1 - (1 - model_b.score(X , Y))*(
↪   (n-1) / (n-p-1) ) )


            L1 = np.quantile( R_2_adj_boot  , q=alpha/2)

            L2 = np.quantile( R_2_adj_boot  , q=1-alpha/2)

            interval = [L1,L2]

########################################################################

    return interval
```

Intervalo de confianza para los coeficientes beta del modelo de regresión lineal:

```
boot_interval_linear_regression(X=X, Y=Y, method='residuals',
↪   parameter='beta', j=1, B=500, alpha=0.05)
```

```
[34457.35430762553, 37169.50119078313]
```

```
boot_interval_linear_regression(X=X, Y=Y, method='pairs',
↪   parameter='beta', j=1, B=500, alpha=0.05)
```

```
[29306.688429798087, 40389.149018568656]
```

```
boot_interval_linear_regression(X=X, Y=Y, method='residuals',
↪   parameter='beta', j=2, B=500, alpha=0.05)
```

```
[-2949083.96469503, -297017.35523746273]
```

```
boot_interval_linear_regression(X=X, Y=Y, method='pairs',
↪ parameter='beta', j=2, B=500, alpha=0.05)
```

```
[-2906726.4973525056, -394604.81858878786]
```

Calculamos los intervalos de confianza para los coeficientes betas del modelo de regresión lineal con la libería `statmodels`

```
import statsmodels.api as sm
```

```
model_SM = sm.OLS(Y , X).fit()
```

```
model_SM.conf_int(alpha=0.05)
```

```
                               0             1
intercept           -1.204625e+08 -2.999117e+06
size_in_m_2          3.424446e+04  3.708364e+04
longitude           -3.031978e+06 -3.223444e+05
latitude             4.583358e+06  7.646506e+06
no_of_bedrooms      -9.991120e+05 -6.742539e+05
no_of_bathrooms     -1.910590e+05  7.681737e+04
quality_recode_1.0  -6.448764e+05 -3.634981e+04
quality_recode_2.0  -4.884944e+05  8.730549e+04
quality_recode_3.0  -5.140369e+05  3.903336e+05
```

Intervalo de confianza para el coeficiente de determinación ajustado:

```
boot_interval_linear_regression(X=X, Y=Y, method='residuals',
↪ parameter='adj_R2', j='none', B=500, alpha=0.05)
```

```
[0.6536518705306413, 0.7417060490432771]
```

```
boot_interval_linear_regression(X=X, Y=Y, method='pairs',
↪ parameter='adj_R2', j='none', B=500, alpha=0.05)
```

```
[0.6248263646856418, 0.7629699519613915]
```

```
model_SM.rsquared_adj
```

```
0.6965926130210288
```

14 Estimación bootstrap de la varianza de las predicciones de un modelo de aprendizaje supervisado

Consideraremos que una estimacion de la varianza de las predicciones de un modelo de regresion (variable respuesta cuantitativa) M es:

$$\frac{1}{h}\sum_{i=1}^{h}\widehat{Var}(\hat{y}_i)$$

Cálculo de $\widehat{Var}(\hat{y}_i)$ por remuestreo (en algunos modelos no habra expresiones cerradas para este estimacion, por eso veo interesante un procedimiento general que no dependa del modelo usado):

Tenemos una muestra inicial de predictores y de la respuesta $(X,Y)=(X_1,...,X_p,Y)$ con n filas (observaciones)

Tomamos B muestras bootstrap (muestras aleatorias con reemplazamiento) de (X,Y):

$$(X,Y)_1,...,(X,Y)_B$$

Entrenamos el modelo M con cada una de las B muestras bootstrap , asi obtenemos B modelos entrenados diferentes $M_1,..,.M_B$

Notese que el modelo M_r ha sido entrenado con la muestra train de observaciones $(X,Y)_r$

Con cada uno de los B modelos entrenados $M_1,..,.M_B$ obtener la prediccion de test de la respuesta, es decir $\widehat{Y}^{test}$, usando una misma muestra fija de test de los predictores $(X_1^{test},...,X_p^{test})$, asi se obtienen B vectores de predcciones de la respuesta $(\widehat{Y}_1^{test},...,\widehat{Y}_B^{test})$ y con ellos se obtienen B predicciones de la respuesta para la i-esima observacion de test de los predictores $x_i^{test}=(x_{i1}^{test},...,x_{ip}^{test})^t$, esto es, se obtiene $\hat{y}_i^{boot}=(\hat{y}_{i1}^{test},...,\hat{y}_{iB}^{test})$, para $i=1,...,h$

Notese que:

$\widehat{Y}_r^{test}$ es el vector con las predicciones de la respuesta que el modelo entrenado M_r genera usando la muestra test de observaciones de los predictores $(X_1^{test},...,X_p^{test})$, , para $r=1,...,B$

$\hat{y}_{ir}^{test}$ es la prediccion de la variable respuesta que el modelo entrenado M_r genera usando la observacion test de los predictores $x_i^{test}=(x_{i1}^{test},...,x_{ip}^{test})^t$, para $r=1,...,B$

$\hat{y}_i^{boot}=(\hat{y}_{i1}^{test},...,\hat{y}_{iB}^{test})$ es la prediccion de la respuesta para la observacion de test x_i^{test} hecha por los distintos modelos $M_1,...,M_B$ entrenados.

Estimamos $Var(\hat{y}_i)$ como la varianza de $\hat{y}_i^{boot}=(\hat{y}_{i1}^{test},...,\hat{y}_{iB}^{test})$

$$\widehat{Var}(\hat{y}_i)=\frac{1}{B}\sum_{r=1}^{B}\left(y_{ir}^{test}-\overline{\hat{y}_i^{boot}}\right)^2$$

Repetimos el proceso con cada $i=1,...,h$, donde h es el tamaño de la muestra de test.

Asi obtenderemos: $\widehat{Var}(\hat{y}_1),\widehat{Var}(\hat{y}_2),...,\widehat{Var}(\hat{y}_h)$

Promediamos y obtenimos asi una estimacion de la varianza de las predcciones del modelo:

$$\frac{1}{h}\sum_{i=1}^{h}\widehat{Var}(\hat{y}_i)$$

15 Estimación bootstrap del sesgo de las predicciones de un modelo de aprendizaje supervisado

Consideraremos que una estimacion del sesgo de las predicciones de un modelo de regresion (variable respuesta cuantitativa) M es:

$$\frac{1}{h}\sum_{i=1}^{h}\widehat{Sesgo}(\hat{y}_i)$$

Cálculo de $\widehat{Sesgo}(\hat{y}_i)$ por remuestreo (en algunos modelos no habra expresiones cerradas para este estimacion, por eso veo interesante un procedimiento general que no dependa del modelo usado):

Tenemos una muestra inicial de predictores y de la respuesta $(X,Y) = (X_1,...,X_p,Y)$ con n filas (observaciones)

Tomamos B muestras bootstrap (muestras aleatorias con reemplazamiento) de (X,Y):

$$(X,Y)_1,...,(X,Y)_B$$

Entrenamos el modelo M con cada una de las B muestras bootstrap , asi obtenemos B modelos entrenados diferentes $M_1,..,.M_B$

Notese que el modelo M_r ha sido entrenado con la muestra train de observaciones $(X,Y)_r$

Con cada uno de los B modelos entrenados $M_1,..,.M_B$ obtener la prediccion de test de la respuesta, es decir $\widehat{Y}^{test}$, usando una misma muestra fija de test de los predictores $(X_1^{test},...,X_p^{test})$, asi se obtienen B vectores de predcciones de la respuesta $(\widehat{Y}_1^{test},...,\widehat{Y}_B^{test})$ y con ellos se obtienen B predicciones de la respuesta para la i-esima observacion de test de los predictores $x_i^{test} = (x_{i1}^{test},...,x_{ip}^{test})^t$, esto es, se obtiene $\hat{y}_i^{boot} = (\hat{y}_{i1}^{test},...,\hat{y}_{iB}^{test})$, para $i = 1,...,h$

Notese que:

$\widehat{Y}_r^{test}$ es el vector con las predicciones de la respuesta que el modelo entrenado M_r genera usando la muestra test de observaciones de los predictores $(X_1^{test},...,X_p^{test})$, , para $r = 1,...,B$

$\hat{y}_{ir}^{test}$ es la prediccion de la variable respuesta que el modelo entrenado M_r genera usando la observacion test de los predictores $x_i^{test} = (x_{i1}^{test},...,x_{ip}^{test})^t$, para $r = 1,...,B$

$\hat{y}_i^{boot} = (\hat{y}_{i1}^{test},...,\hat{y}_{iB}^{test})$ es la prediccion de la respuesta para la observacion de test x_i^{test} hecha por los distintos modelos $M_1,...,M_B$ entrenados.

Notese que sabemos que y_i^{test} es el verdadero valor de la respuesta en la muestra de test para la observacion x_i^{test}

Estimamos $Sesgo(\hat{y}_i)$ como la diferencia entre la media de $\hat{y}_i^{boot} = (\hat{y}_{i1}^{test},...,\hat{y}_{iB}^{test})$ y el verdadero valor y_i^{test} de la respuesta en la muestra de test para la observacion de test x_i^{test}

$$\widehat{Sesgo}(\hat{y}_i) = \left(\frac{1}{h}\sum_{r=1}^{B}\hat{y}_{ir}^{test}\right) - y_i^{test}$$

Repetimos el proceso con cada $i = 1,...,h$, donde h es el tamaño de la muestra de test.

Asi obtenderemos: $\widehat{Sesgo}(\hat{y}_1),\widehat{Sesgo}(\hat{y}_2),...,\widehat{Sesgo}(\hat{y}_h)$

Promediamos y obtenimos asi una estimacion de la varianza de las predcciones del modelo:

$$\frac{1}{h}\sum_{i=1}^{h}\widehat{Sesgo}(\hat{y}_i)$$

16 Bibliografía