Intervalos basados en percentiles

Jeimy Sandoval

Tabla de contenidos

Intervalos Basados en Percentiles

Los intervalos basados en percentiles se basan en la idea de que, la distribución simulada, se asemeja a la distribución muestral del estadístico, y de que su desviación estándar se aproxima al error estándar de dicha distribución. Acorde a esto, se puede estimar el intervalo de confianza del estadístico utilizando los percentiles de la distribución obtenida.

Los pasos a seguir son:

1) Generar B nuevas muestras mediante *bootstrapping*.

2) Calcular el estadístico de interés en cada nueva muestra.

3) Calcular los cuantiles inferior y superior acorde al intervalo deseado.

4) Generar el intervalo de confianza como [cuantil inferior, cuantil superior].

Por ejemplo, el intervalo de confianza del 95% obtenido a partir de 1000 muestras de *bootstrapping*, es el intervalo entre el el cuantil 0.025 y el cuantil 0.975 de la distribución obtenida.

Ejemplo

Se dispone de una muestra formada por 30 observaciones de una variable aleatoria continua. Se desea calcular un intervalo de confianza del 95% para la media empleando el método de *bootstrapping* basado en percentiles.

Librerías

Las librerías utilizadas en este documento son:

```
# Tratamiento de datos
# ==============================================================================
import pandas as pd
import numpy as np
from scipy.stats import trim_mean
```

```
# Gráficos
# ==============================================================================
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
```

```
# Configuración matplotlib
# ==============================================================================
style.use('ggplot') or plt.style.use('ggplot')
```

```
# Configuración warnings
# ==============================================================================
import warnings
warnings.filterwarnings('ignore')
```

```
# Varios
# ==============================================================================
from tqdm import tqdm
```

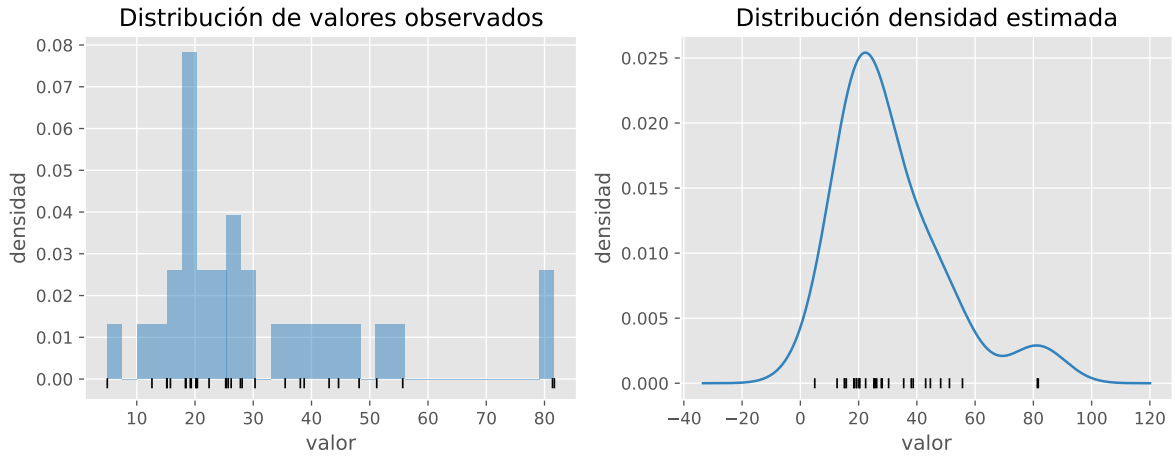
Datos

Los datos utilizados en este ejemplo se han obtenido del libro *Comparing Groups Randomization and Bootstrap Methods Using R*.

```
# Datos
# ==============================================================================
datos = np.array([
            81.372918, 25.700971, 4.942646, 43.020853, 81.690589, 51.195236,
            55.659909, 15.153155, 38.745780, 12.610385, 22.415094, 18.355721,
            38.081501, 48.171135, 18.462725, 44.642251, 25.391082, 20.410874,
            15.778187, 19.351485, 20.189991, 27.795406, 25.268600, 20.177459,
            15.196887, 26.206537, 19.190966, 35.481161, 28.094252, 30.305922
        ])
```

```
# Gráficos distribución observada
# ==============================================================================
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(10, 4))

axs[0].hist(datos, bins=30, density=True, color='#3182bd', alpha=0.5, label = 'muestra_1')
axs[0].plot(datos, np.full_like(datos, -0.001), '|k', markeredgewidth=1)
axs[0].set_title('Distribución de valores observados')
axs[0].set_xlabel('valor')
axs[0].set_ylabel('densidad')

pd.Series(datos).plot.kde(ax=axs[1],color='#3182bd')
axs[1].plot(datos, np.full_like(datos, 0), '|k', markeredgewidth=1)
axs[1].set_title('Distribución densidad estimada')
```

```
axs[1].set_xlabel('valor')
axs[1].set_ylabel('densidad')

fig.tight_layout();
```



La representación gráfica muestra evidencias de que los datos no se distribuyen de forma normal. Esto implica que, la aproximación basada en el teorema del límite central para estimar el error estándar SE=sdn√ = , deja de ser buena y con ella los intervalos paramétricos basados en la estructura [parámetro estimado±t SE][parámetro estimado±]. Una alternativa para poder calcular intervalos de confianza es emplear bootstrapping.

Intervalo de confianza basado en percentiles.

Mediante *bootstrapping*, se simula la variabilidad esperada en el estadístico, en este caso la media, debido únicamente al muestreo aleatorio

```
def calcular_estadistico(x):
    '''
    Función para calcular el estadístico de interés.

    Parameters
    ----------
    x : numpy array
         valores de la muestra.

    Returns
    -------
```

```
    estadístico: float
        valor del estadístico.
    '''
    estadistico = np.mean(x)

    return(estadistico)
```

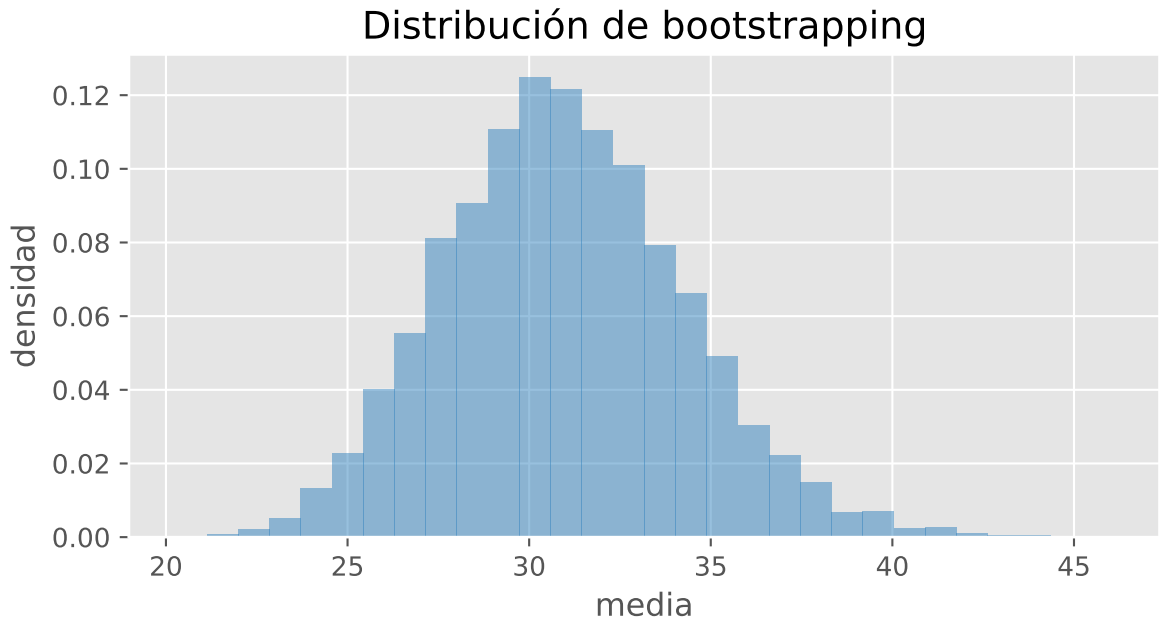
```
def bootstraping(x, fun_estadistico, n_iteraciones=9999):
    '''
    Función para calcular el valor del estadístico en múltiples muestras generadas
    mediante muestreo repetido con reposición (bootstrapping).

    Parameters
    ----------
    x : numpy array
         valores de la muestra.

    fun_estadistico : function
        función que recibe como argumento una muestra y devuelve el valor
        del estadístico.

    n_iteraciones : int
        número iteraciones (default `9999`).

    Returns
    -------
    distribuciones: numpy array
        valor del estadístico en cada muestra de bootstrapping.
    '''

    n = len(x)
    dist_boot = np.full(shape=n_iteraciones, fill_value=np.nan)

    for i in tqdm(range(n_iteraciones)):
        resample = np.random.choice(x, size=n, replace=True)
        dist_boot[i] = fun_estadistico(resample)

    return dist_boot
```

```
ist_boot = bootstraping(
                x = datos,
                fun_estadistico = calcular_estadistico,
```

```
                n_iteraciones    = 9999
            )
```

```
  0%|          | 0/9999 [00:00<?, ?it/s] 27%|         | 2735/9999 [00:00<00:00, 23785.02it/s]
```

```
# Distribución de bootstrapping
# ==============================================================================
fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(7,3.3))
ax.hist(ist_boot, bins=30, density=True, color='#3182bd', alpha=0.5)
ax.set_title('Distribución de bootstrapping')
ax.set_xlabel('media')
ax.set_ylabel('densidad');
```



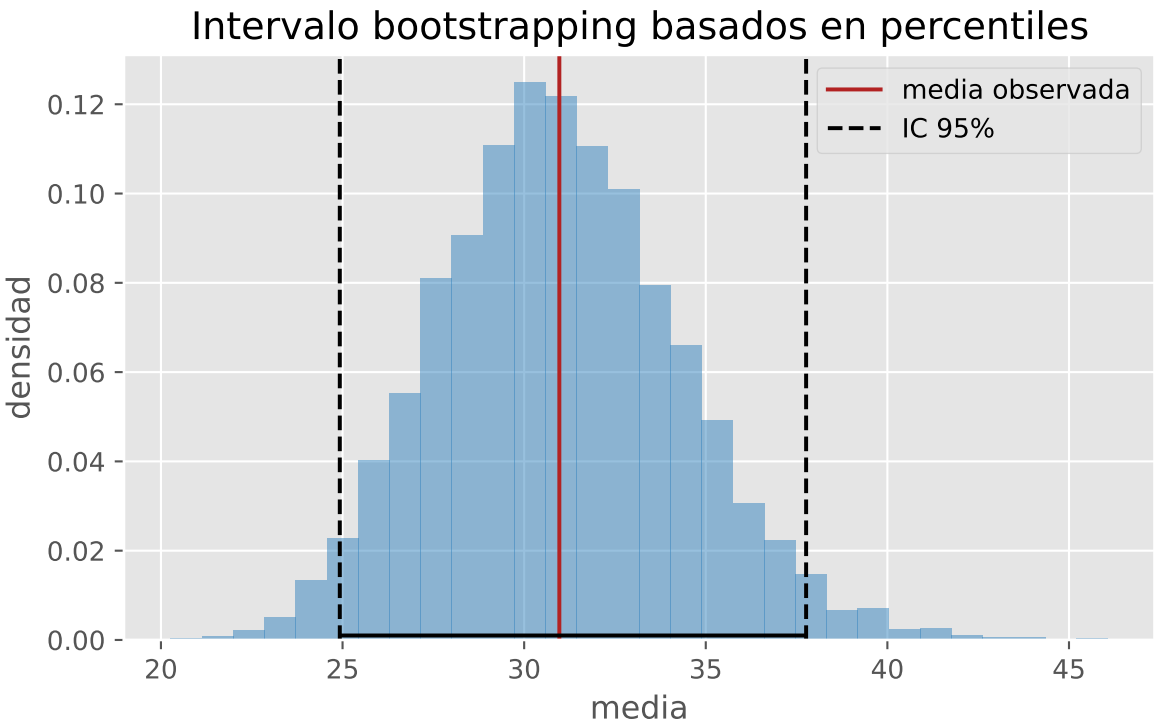
La dispersión de la distribución obtenida por *bootstrapping* es una aproximación del error estándar esperado debido a proceso de muestreo. Por esta razón, pueden emplearse sus percentiles para calcular intervalos de confianza.

```
# Intervalo IC basado en percentiles de la distribución bootstrapping
# ==============================================================================
# Un IC del 95% debe abarcar desde el cuantil 0.025 al 0.975
cuantiles = np.quantile(a = ist_boot, q = [0.025, 0.975])
print('------------------------------')
```

```
print('Intervalo basado en percentiles')
print('-------------------------------')
print(cuantiles)
```

```
-------------------------------
Intervalo basado en percentiles
-------------------------------
[24.92406793 37.76279069]
```

```
# Gráfico intervalo de confianza del 95%
# ==============================================================================
fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(7,4))
ax.hist(ist_boot, bins=30, density=True, color='#3182bd', alpha=0.5)
ax.axvline(x=datos.mean(), color='firebrick', label='media observada')
ax.axvline(x=cuantiles[0], color='black', linestyle='--', label='IC 95%')
ax.axvline(x=cuantiles[1], color='black', linestyle='--')
ax.hlines(y=0.001, xmin=cuantiles[0], xmax=cuantiles[1], color='black')
ax.set_title('Intervalo bootstrapping basados en percentiles')
ax.set_xlabel('media')
ax.set_ylabel('densidad')
ax.legend();
```


Intervalo bootstrapping basados en percentiles
media observada
IC 95%
densidad
0.12
0.10
0.08
0.06
0.04
0.02
0.00
20
25
30
35
40
45
media