2. Confindence Intervals with Bootstrap_boot_function_pandas_data_frame_new_naming

November 30, 2020

0.1 2. Confidence Intervals with the Bootstrap

In the second lecture we learned that the bootstrap can be used to create asymptotically valid confidence intervals for the parameter θ by

$$C_n = \Big(q_{boot}(\frac{\alpha}{2}), q_{boot}(1 - \frac{\alpha}{2})\Big),$$

where $q_{boot}(\beta)$ is the β-sample quantile of the bootstrapped estimators $\hat{\theta}^{(1)}, ..., \hat{\theta}^{(B)}$.

In this exercise we are interested in constructing confidence intervals for the population median, i.e., $\theta = \text{median}(X)$, based on an i.i.d. sample $X_1, \ldots, X_n$.

1. Implementation of Bootstrapped Confidence Intervals

a) Write down the idea behind the bootstrap in your own words.

b) Implement the bootstrapped confidence intervals as defined above in a function `bootstrap_ci`. In your implementation use the following arguments:

- `X`: A *pandas DataFrame* containing the sample $X_1, \ldots, X_n$.
- `alpha`: Significance level
- `B`: Number of Bootstrap repetitions

The function should return a bootstrapped (1-α)-confidence interval as a `numpy.ndarray`.

2. Simulation Study for Bootstrapped Confidence Intervals Set up a simulation study that illustrates the asymptotic validity of the bootstrap-based confidence intervals. Implement the following setting in your simulation: Let $X_i, \ldots, X_n \sim N(\mu, \sigma^2)$ with $\mu = 10$ and $\sigma^2 = 5$.

a) Generate the data set according to this setting with a sample of size $n = 200$ and demonstrate that your implementation of `bootstrap_ci` provides a $(1 - \alpha)$-confidence interval. Set $\alpha = 0.1$ and $B = 500$. Does the confidence interval cover the true median? (Set np.random.seed(1234))

b) Repeat your calculation in part a) 100 times and count in how many cases your confidence interval covers the true median. Does the confidence interval maintain the coverage probability $1 - \alpha$?

c) Run your simulation from part b) for different sample sizes, i.e., for $n = 40$ and $n = 100$. Summarize your findings on the coverage of the confidence intervals. How does the average length of the confidence intervals change if n is increased by a factor of 2.5, and 5, respectively? (*Hint*: Calculate the average length as the mean of the length of the confidence intervals over all repetitions)

[131]:
```
#a explain idea behind bootstrap
```

Bootstrapping is a statistic method for resempling data. The main aspect of bootstrapping is to repeat statistics based on one sample. It is used when f. e. if the theoretical distribution of the statistics is not known. The sample function is calculated repeatedly on the basis of the sub-samples drawn and the distribution properties of a sample are examined on the basis of these results.

The following code shows the implementation with the difinition of the bootstrap under certain arguments like x as the data frame that includes the observation, B as the number of times for a bootstrap to repeat, alpha as the likelihood that the true parameter lies outside the confidence interval.

0.1.1 Set up Data

[132]:
```
#a) set up sample with seed

import numpy as np
import scipy.stats as stats
import pandas as pd

def generate_data(N):                      #data set size "N", not to be confused␣
 ↪with sample size "n"
    mu = 10
    sigma = 5
    data = np.random.normal(mu, sigma, N)
    data = pd.DataFrame(data = data)
    return data
```

0.1.2 Implementation of Bootstrapped Confidence Intervals

[133]:
```
#a) set up boostrap function

import math
import statistics as st
import pandas as pd

B=500
alpha = 0.1

def boostrap_ci(X, alpha, B):                      #sample drawn from data set need␣
 ↪to be incoprorate into boostrap_ci
    sample_medians = []
    for s in range(B):
```

```
        sample = X.sample(n=len(X), replace=True)              #draw sample␣
↪from generated data 500 times
        sample_median = np.median(sample)                    #find median of that␣
↪sample
        sample_medians.append(sample_median)                 #include that median␣
↪to the median list
    var_boot_sample = np.var(sample_medians)*1/(500-1)     #find variance of␣
↪median list
    std_boot_sample = math.sqrt(var_boot_sample)           #convert variance to␣
↪standart deviation
    quantile_left = np.quantile(sample_medians, q = alpha/2)              ␣
↪#calculate left quantile
    quantile_right = np.quantile(sample_medians, q = 1-alpha/2)           ␣
↪#calculate right quantile
    confidence_interval2 = (quantile_left, quantile_right)                ␣
↪#calcualte confidence interval
    return confidence_interval2
```

[112]:
```
np.random.seed(1234)
X = generate_data(200)
boostrap_ci(X, alpha, B)
```

[112]: (9.590264740866674, 11.076342904847216)

[113]:
```
data_median = np.median(generate_data(200))
```

[114]:
```
data_median
```

[114]: 9.982856837523084

0.1.3 Simulation Study for Bootstrapped Confidence Intervals

[115]:
```
#b) set up sample without seed to repeat 100 times

def repeat_boostrap_ci(n):
    #set up repeat
    k = 100              #times to repeat
    intervals =[]
    for i in range(k):
        X = generate_data(n)                                    #a new sample generated␣
↪100 times
        confidence_interval2 = boostrap_ci(X, alpha, B)
        intervals.append(confidence_interval2)

    return intervals
```

[116]:
```python
intervals_200 = repeat_boostrap_ci(200)
```

[117]:
```python
def covers_median(intervals):
    #check if confidence interval covers true median
    count = 0
    for i in intervals:
        if i[0] < data_median and i[1] > data_median:
             count +=1
    print("it covers the true median", count, "percent of the time")
    return count
```

[118]:
```python
coverage_200 = covers_median(intervals_200)
coverage_200
```

```
it covers the true median 93 percent of the time
```

[118]:
```
93
```

the coverage probability has decreased from (1-alpha) 90% to *see above*

[119]:
```python
#c) run simulation for different samples sizes and determine average length of␣
 ↪ci

def mean_ci_length(intervals):
    #calculate average length of confidence interval
    for i in intervals:
        all_ci_lengths = []
        ci_length = i[1]-i[0]
        all_ci_lengths.append(ci_length)

    mean_ci_length = st.mean(all_ci_lengths)
    print("mean interval length is", mean_ci_length)
    return mean_ci_length
```

[120]:
```python
mean_200 = mean_ci_length(intervals_200)
mean_200
```

```
mean interval length is 1.3839659853014261
```

[120]:
```
1.3839659853014261
```

[121]:
```python
#c simulation for different samples sizes
```

[122]:
```python
#40

intervals_40 = repeat_boostrap_ci(40)
coverage_40 = covers_median(intervals_40)
mean_40 = mean_ci_length(intervals_40)
```

```
it covers the true median 88 percent of the time
mean interval length is 4.600910797240228
```

[123]:
```
#100 (increase by a factor of 2.5)

intervals_100 = repeat_boostrap_ci(100)
coverage_100 = covers_median(intervals_100)
mean_100 = mean_ci_length(intervals_100)
```

```
it covers the true median 92 percent of the time
mean interval length is 1.8833411249316665
```

[127]:
```
#simulation summary

print("Sample size:", 40, 100, 200)
print("Coverage:", coverage_40, coverage_100, coverage_200)
print("average interval length:", round(mean_40,2), round(mean_100, 2),␣
 ↪round(mean_200, 2))
```

```
Sample size: 40 100 200
Coverage: 88 92 93
average interval length: 4.6 1.88 1.38
```

[ ]:
```

```