towards data science

Sign in Get started

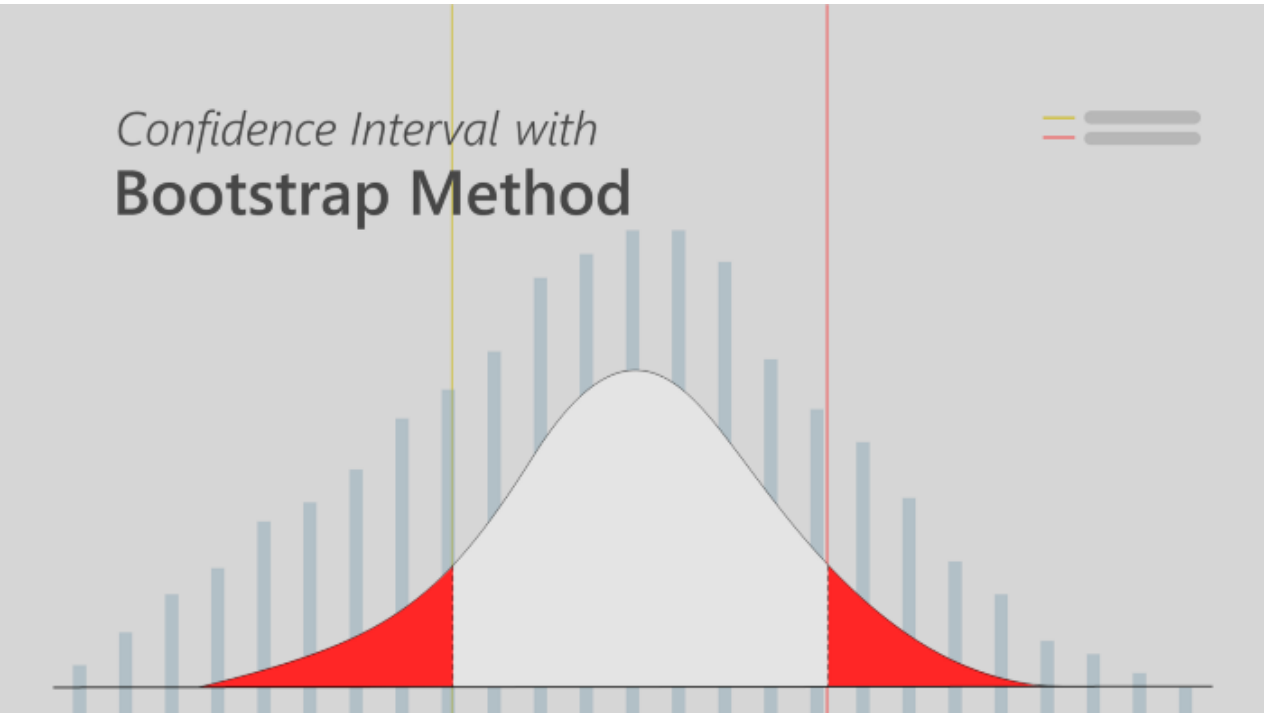
Follow 615K Followers · Editors' Picks Features Deep Dives Gr

This is your **last** free member-only story this month. Sign up for Medium and get an extra one

Calculating Confidence Intervals with Bootstrapping

How can we calculate the confidence interval with bootstrapping?

Barış Hasdemir Jul 12, 2020 · 7 min read ★



Story banner, Image by author

Hi everyone,

In this article, I will attempt to explain how we can find a confidence interval by using Bootstrap Method. ***Statistics*** and ***Python*** knowledge are needed for better understanding.

Before diving into the method, let's remember some statistical concepts.

Variance: It is obtained by the sum of squared distances between a data point and the mean for each data point divided by the number of data points.

$$\sigma^2 = \frac{\sum_{i=1}^{n} \left(x_i - \bar{X}\right)^2}{n}$$

$$s = \frac{}{n-1}$$

Sample variance

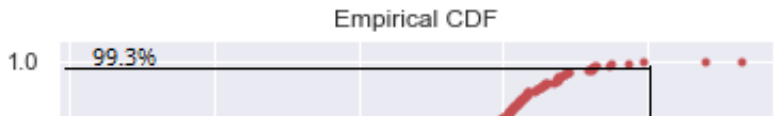
Standard Deviation: It is a measurement that shows us how our data points spread out from the mean. It is obtained by taking the square root of the variance

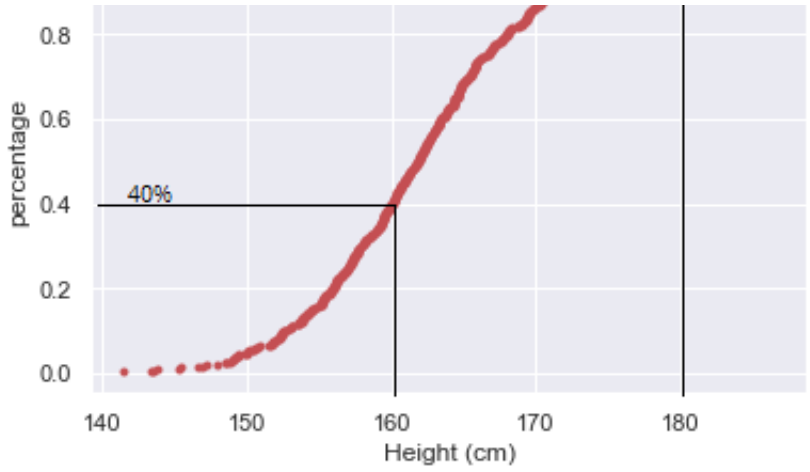
$$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n-1}}$$

Sample standard deviation

Cumulative Distribution Function: It can be used on any kind of variable X(discrete, continuous, etc.). It shows us the probability distribution of a variable. Therefore allowing us to interpret the probability of a value less than or equal to x from a given probability distribution

Empirical Cumulative Distribution Function: Also known as Empirical Distribution Function. The only difference between CDF and ECDF is, while the former shows us the hypothetical distribution of any given population, the latter is based on our observed data.





For example, how can we interpret the ECDF of the data shown on the chart above? We can say that 40% of heights are less than or equal to 160cm. Likewise, the percentage of people with heights of less than or equal to 180 cm is 99.3%

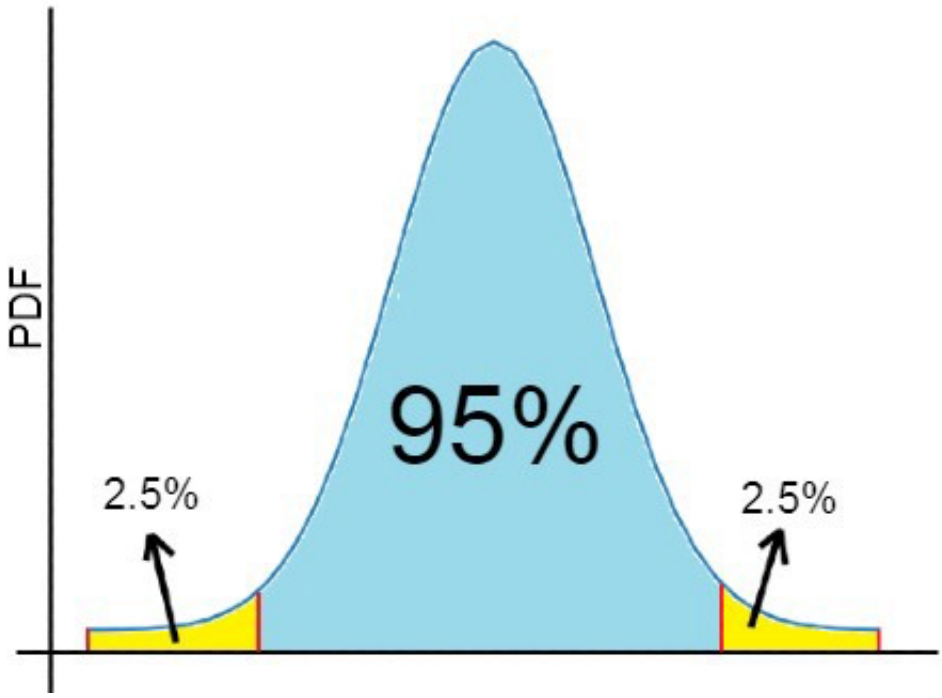
Probability Density Function: It shows us the distribution of continuous variables. The area under the curve gives us the probability so that the area must always be equal to 1

Normal Distribution: Also known as *Gaussian Distribution*. It is the most important probability distribution function in statistics which is bell-shaped and symmetric.

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Normal (Gaussian) Distribution

Confidence Interval: It is the range in which the values likely to exist in the population. It is estimated from the original sample and usually defined as 95% confidence but it may differ. You can consider the figure below which indicates a 95% confidence interval. The lower and upper limits of confidence interval defined by the values corresponding to the first and last 2.5th percentiles.



95% Confidence Interval, Image by author

What is Bootstrap Method?

Bootstrap Method is a resampling method that is commonly used in Data Science. It has been introduced by Bradley Efron in 1979. Mainly, it consists of the resampling our original sample with

replacement (***Bootstrap Sample***) and generating ***Bootstrap replicates*** by using *Summary Statistics*.

Confidence Interval of people heights

In this article, we are going to work with one of the datasets in ***Kaggle***. It is *Weight-Height* data sets. It contains height (in inches) and weight (in pounds) information of 10.000 people separated by gender.

> *If you would like to see the whole code, you can* find the *IPython notebook* via this ***link***.

We are going to use only heights of 500 randomly selected people and compute a 95% confidence interval by using Bootstrap Method

Let's start with importing the libraries that we will need.

```
# Import necessary libraries
import pandas as pd
import numpy as np
from scipy import stats
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline

# Enable Jupyter Notebook's intellisense
%config IPCompleter.greedy=True
```

```
%config InlineBackend.figure_format = 'retina'
```

The first five rows of the DataFrame like following

```
# Import the iris dataset
data = pd.read_csv('weight-height.csv')

# Display first 5 rows
display(data.head())
```

bs.py hosted with ♥ by GitHub view raw

	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801

Apparently, heights are in *inches*, let's convert heights from inches to *centimeters* and store in a new column *Height(cm)*.

```
# Convert inches to centimeters
data["Height(cm)"] = data["Height"]*2.54

# Get summary statistics of Heights in centimeters
display(data['Height(cm)'].describe())
```

Bootstrapping2.py hosted with ♥ by GitHub view raw

```
count    10000.000000
mean       168.573602
std          9.772721
min        137.828359
25%        161.304276
50%        168.447898
75%        175.702625
max        200.656806
Name: Height(cm), dtype: float64
```

As we can see above, the maximum and minimum height in the data set are 137.8 cm and 200.6 cm respectively.

We can use pandas.DataFrame's ***sample*** method to select 500 randomly selected heights. After that, we will print the summary statistics.

```
# Extract 500 random heights
heights = data['Height(cm)'].sample(500).reset_index(drop=True)

# Display Summary Statistics of heights in cm
display(heights.describe())
```

Bootstrapping3.py hosted with ♥ by GitHub view raw

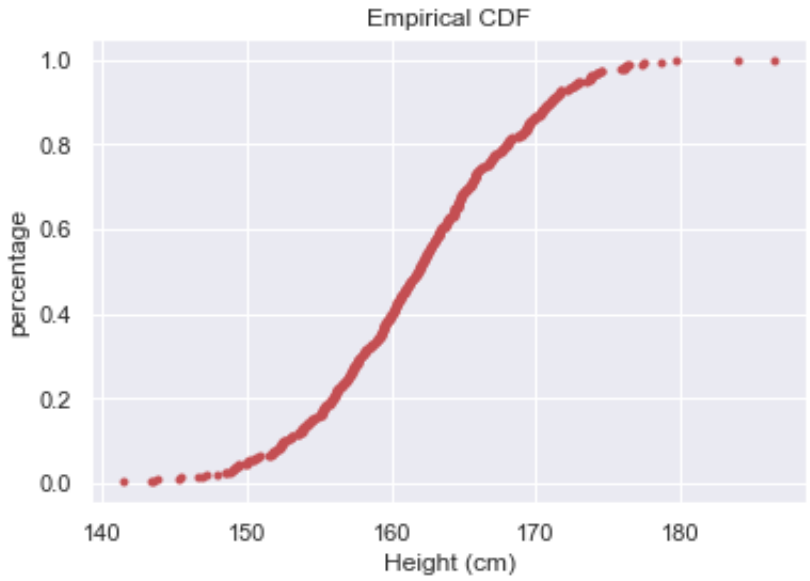
```
count    500.000000
mean     168.824438
std        9.824858
min      145.181405
25%      161.829489
50%      168.665300
75%      175.833922
max      198.363503
Name: Height(cm), dtype: float64
```

According to the output, our sample has 145 cm as minimum height and 198 cm as the maximum height.

Let's look at how ECDF and PDF look like?

```
# Create a function to get x, y for of ecdf
def get_ecdf(data):
    """Returns x,y for ecdf"""
    # Get lenght of the data into n
    n = len(data)

    # We need to sort the data
    x = np.sort(data)

    # the function will show us cumulative percentages of corresponding da
    y = np.arange(1,n+1)/n

    return x,y

# Create a function to plot ecdf
def plot_ecdf(data,labelx,labely,title,color):
    """Plot ecdf"""
    # Call get_ecdf function and assign the returning values
    x, y = get_ecdf(data)

    plt.plot(x,y,marker='.',linestyle='none',c=color)
    plt.xlabel(labelx)
    plt.ylabel(labely)
    plt.title(title)

# Plotting Empirical CDF
plot_ecdf(heights,"Height (cm)","percentage","Empirical CDF","r")
plt.show()
```

ecdf.py hosted with ❤ by GitHub view raw

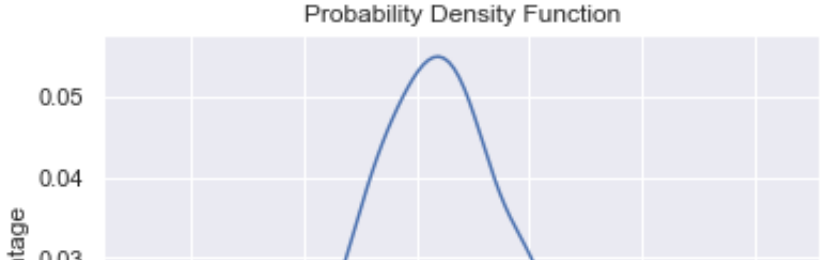


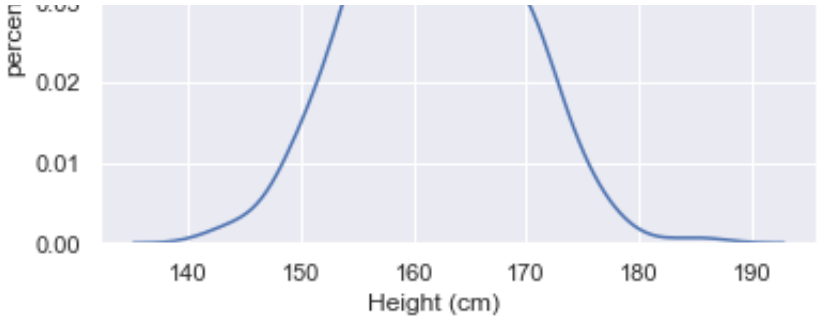
ECDF, Image by author

Empirical CDF demonstrates that 50% of people in our sample have 162 cm or less height.

What about PDF?

```
# Plotting PDF
sns.distplot(heights,hist=False)
plt.xlabel("Height(cm)")
plt.ylabel("PDF")
plt.title("Probability Density Function")
plt.show()
```



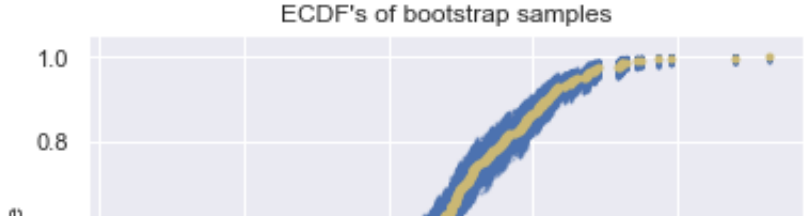


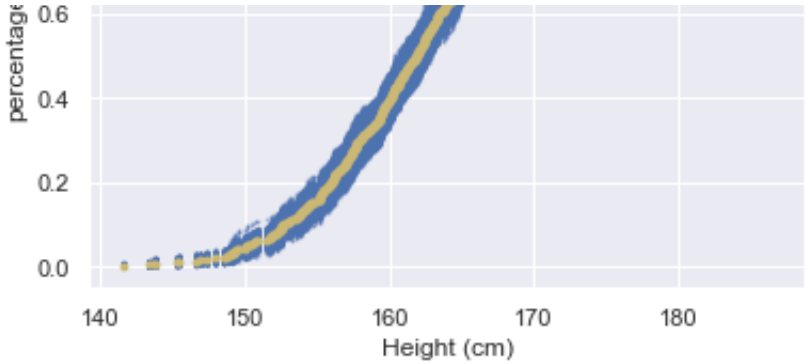
PDF, Image by author

PDF shows us the heights' distribution is too close to the normal distribution. Do not forget that the ***area under the curve*** *gives us the probability.*

Now, take a moment to think. We have only 500 observations in our sample, but there are billions of people in the world who we cannot measure their heights. Therefore, our sample does not give inference to the population. If we did the same measurements for different samples again and again, what would be the mean of heights?

For instance, assume that we did the same measurements with the same number of people (500) for 1000 times and plot the ECDF for each in a way that overlays the first observation's ECDF. It will look like the following.





ECDF, Image by author

As we can see above, we got different heights, but we can easily detect that the points are spreading in a specific range. That's the confidence interval that we want to learn

You may say that it is impossible to repeat the experiment so many times, you are not wrong. The exact reason why we use the Bootstrap Method. It helps us to simulate the same experiment thousands or even billions of times.

How?

In fact, the Bootstrap Method is quite straightforward and easy to understand. First, it generates bootstrap samples from our original sample by randomly choosing among the original sample. After that, it applies a summary statistics such as variation, standard deviation, mean, and so forth to get replicates. We will use 'mean' to generate our bootstrap replicates.

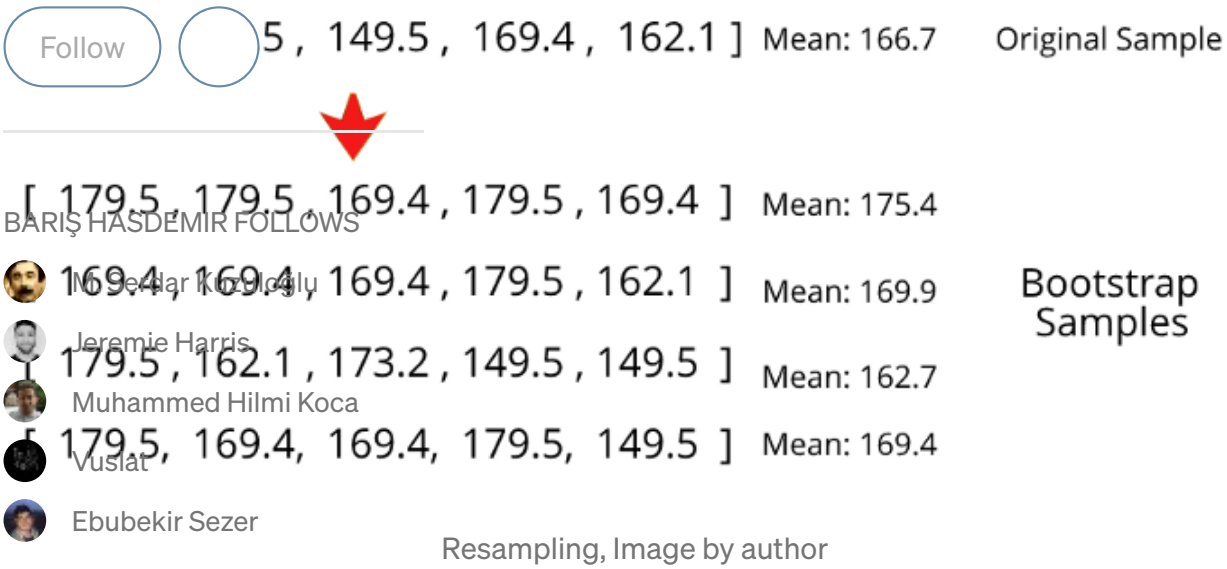
To understand the method, let's apply it to a small sample that

contains only 5 heights. We can generate our bootstrap samples like the following. Do not forget the fact that we can choose any observation more than once (resampling with replacement)

Barış Hasdemir

Data Engineer @trendyol

Follow



Resampling, Image by author

BARIŞ HASDEMIR FOLLOWS
Mevlüdiar Kızıloğlu
Jeremie Harris
Muhammed Hilmi Koca
Vuslat
Ebubekir Sezer
See all (49)

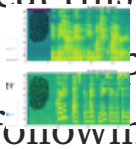
As we can see above we create 4 bootstrap samples and after that calculate their means. We will call these means our bootstrap replicates. Instead of 'mean' we could choose variance, standard deviation, median, or anything else.

10

Come back to our project. The next step, we are going to generate our bootstrap sample from our original sample and we will apply to mean to get bootstrap replicate. We will repeat this process 15.000 times (drawing) in a for loop and store the replicates in an array. To do this we can define a function like following

Related

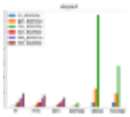

Andrew Chen on network effects and competing for growth in a blocketing market


A Speech Embedding Model for Speaker Recognition


Neutral Style Transfer
If you've ever pictured what ...

```
1  def draw_bs_replicates(data,func,size):
```

```
    """creates a bootstrap sample, computes replicates and returns replica
    # Create an empty array to store replicates
    bs_replicates = np.empty(size)

    # Create bootstrap replicates as much as size
    for i in range(size):
        # Create a bootstrap sample
        bs_sample = np.random.choice(data,size=len(data))
        # Get bootstrap replicate and append to bs_replicates
        bs_replicates[i] = func(bs_sample)

    return bs_replicates
```


Comparing Open Source BGP stacks with internet routes

After we got 15.000 replicates by calling the function, we can compare between the means of the original sample and the bootstrap replicates

```
# Draw 15000 bootstrap replicates
bs_replicates_heights = draw_bs_replicates(heights,np.mean,15000)

# Print empirical mean
print("Empirical mean: " + str(heights_mean))

# Print the mean of bootstrap replicates
print("Bootstrap replicates mean: " + str(np.mean(bs_replicates_heights)))
```

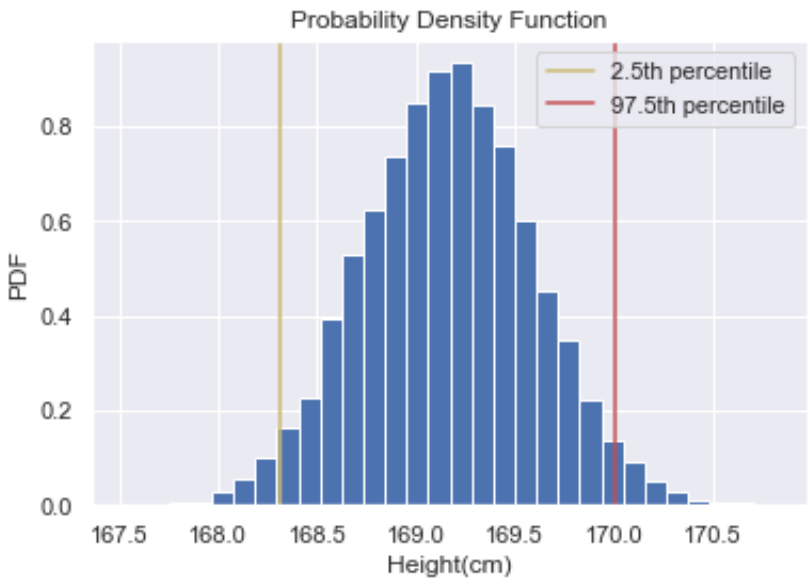
```
Empirical mean: 168.65661881557742
Bootstrap replicates mean: 168.6535488982738
```

Their means are too close.

So, what are we going to do to calculate a 95% confidence interval?

After obtaining bootstrap replicates, the rest is so simple. As we know, our lower and upper limits are the values correspond to the 2.5th and 97.5th percentiles.

```
# Plot the PDF for bootstrap replicates as histogram
plt.hist(bs_replicates_heights,bins=30,normed=True)

# Showing the related percentiles
plt.axvline(x=np.percentile(bs_replicates_heights,[2.5]), ymin=0, ymax=1,]
plt.axvline(x=np.percentile(bs_replicates_heights,[97.5]), ymin=0, ymax=1,

plt.xlabel("Height(cm)")
plt.ylabel("PDF")
plt.title("Probability Density Function")
plt.legend()
plt.show()
```



Percentiles, Image by author

We can find the boundaries with following simple Python code

```
# Get the corresponding values of 2.5th and 97.5th percentiles
conf_interval = np.percentile(bs_replicates_heights,[2.5,97.5])

# Print the interval
print("The confidence interval: ",conf_interval)
```

Bootstrapping6.py hosted with ♥ by GitHub view raw

```
The confidence interval:  [167.77904671 169.51984003]
```

Our boundaries are found at 167.7 and 169.5. Therefore, we can say that if we do the same experiment with the whole population. The mean of heights will be between 167.7 cm and 169.5 cm with 95% of chance

Summary

Let's summarize what we did. We have randomly selected 500 heights and generated bootstrap samples. We calculated the 'mean' from those samples and got bootstrap replicates of means. Ultimately we calculated a 95% confidence interval.

I wish you good luck in your data journey :)

Reference

Introduction to Bootstrapping in Statistics with an Example — Statistics By Jim

Bootstrapping is a statistical procedure that resamples a single dataset to create many simulated samples....

statisticsbyjim.com

Statistical Thinking in Python (Part 2)

When doing statistical inference, we speak the language of probability. A probability distribution that...

www.datacamp.com

An Introduction to the Bootstrap Method

An exploration about bootstrap method, the motivation, and how it works

towardsdatascience.com

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. Take a look.

Get this newsletter

About Write Help Legal