上海立信会计金融学院
SHANGHAI LIXIN UNIVERSITY OF ACCOUNTING AND FINANCE

《Python金融数据分析》

Hong Cheng（程宏）
School of Statistics and Mathematics
Shanghai LiXin University of Accounting and Finance

April 2022

Sampling and Inference

In financial analysis, we always **infer the real mean return of stocks, or equity funds, based on the historical data of a couple years**.

This situation is in line with a core part of statistics - **Statistical Inference** - which we also base on sample data to infer the population of a target variable.In this module, you are going to understand the basic concept of statistical inference such as **population**, **samples and random sampling**.

In the second part of the module, we shall **estimate the range of mean return of a stock using a concept called** **confidence interval**, after we understand the distribution of sample mean.

We will also testify the claim of investment return using another statistical concept - **hypothesis testing**.

Learning Objectives

- Compare the properties of population and sample
- Illustrate the difference between two kinds of sampling with examples
- Explain the use of unbiased estimator (n-1;ddof=1 in python) when calculating sample variance
- Describe the distribution of sample mean and variance of normal distributed population
- Use the central limit theorem to explain the distribution of sample mean of aribitary population
- Explain the implication of Confidence Interval in estimating average stock return
- Identify the basic concept of Confidence Interval in estimating population mean
- Outline the steps involved in performing hypothesis testing for validating assertion about population
- Apply the steps involved in hypothesis testing in testifying the claims of investment return
- Recognize p-value as an alternative quatitative tool in performing two tail test - a part of hypothesis testing

Module Introduction

In this topic, we will explain **statistical inference**, which is a core part of statistics.

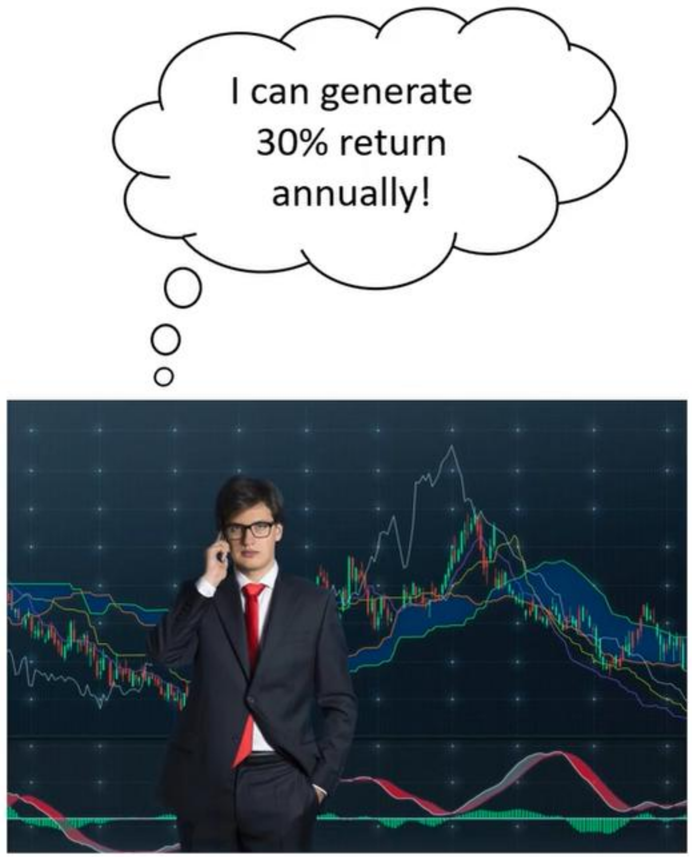
In financial analysis, we're concerned about characteristics of some targets called a population.

1.5% -4.558%
-8.8756% 13.48%

Return on investment

For example, we want to make use of historical data of a couple years, called **sample**, to **estimate the real mean return of some private equity funds.**

Sometimes, we also want to **testify some claims**. For example, ***if the fund managers claim that their investment strategy can generate 30% yearly return, we have to validate this claim using data of the last 20 years.***



These two applications are **typical tasks of statistical inference** to infer the promptings of interesting targets.

01

Population and Sample

Introduce random sampling and samples

we are going to discuss basic concepts about **population and sample**.

We may be interested in some properties about a certain group, which we call target population, that cannot be observed completely.

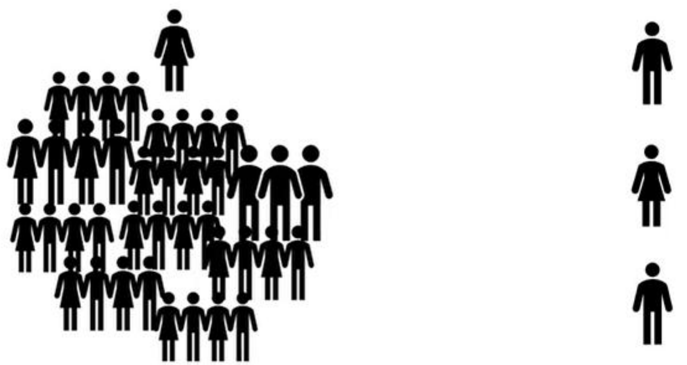
Since **we** **cannot get information for every individual** **of these populations**, we have to take sample, which is a part of target population.

Sample

- Small group of members of a population selected to represent the population

- Must be randomly selected

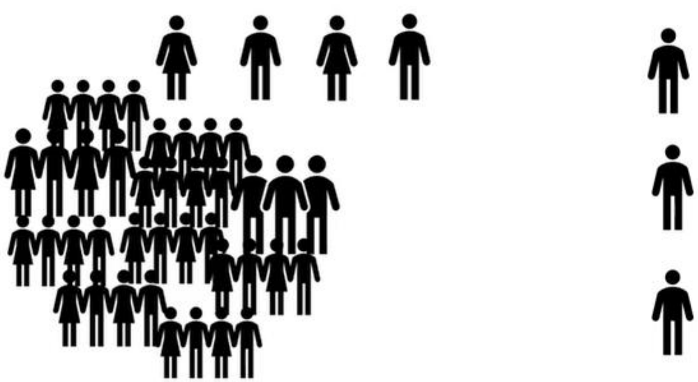
Two kinds of sampling



Sampling without replacement

- A population individual can be selected only one time
- Without putting it back to population

If the population size is very large, this method is a factor because it can generate a random sample with different individuals.



Sampling with replacement

- A randomly selected individual will be put back before the next one being selected
- A population element can be selected more than one time

When the population is small, this method can make sure that everyone in the population has the same chance being selected.

In this example, we first create a blank DataFrame and we create a new column in this DataFrame.

```
In [1]  data = pd.DataFrame()
        data['Population'] = [47, 48, 85, 20, 19, 13, 72, 16, 50, 60]

In [2]  a_sample_without_replacement = data['Population'].sample(5,replace=False)

In [3]  a_sample_with_replacement = data['Population'].sample(5, replace=True)
```

True ⟶ With replacement

Here are the results of two samples. You can see that for sampling without replacement, which is on the left-hand side, there are no duplicates individuals found.

As we can see, there are no duplicate index numbers either. **However**, on the right-hand side, which is the sampling with replacement, it is possible to see the same individual is drawn for multiple times.

```
In [4]  a_sample_without_replacement

Out [4] 8   50
        3   20
        4   19
        2   85
        1   48
        Name: Population, dtype: int64
```

```
In [5]  a_sample_with_replacement

Out [5] 0    47
        8    50
        9    60
        9    60
        9    60
        Name: Population, dtype: int64
```

Besides population sample, there is **another pair of concepts**, **parameter** and **statistics**, of which are very important in categorizing population and sample.

Parameters

- Mean
- Variance
- Standard deviation

Statistics

- Sample mean
- Sample variance
- Sample standard deviation

Population

In [6]
```
print('Population mean is ', data['Population'].mean())
print('Population variance is ', data['Population'].var(ddof=0))
print('Population standard deviation is ', data['Population'].std(ddof=0))
print('Population size is ', data['Population'].shape[0])
```

Out [6]
```
Population mean is 43.0
Population variance is 571.8
Population standard deviation is 23.9123399106
Population size is 10
```

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$$

In calculating var and std, we set the keyword **ddof as 0**. **Which means the denominator of the population variance is N, which is the number of the population.**

Sample

In [5]
```
a_sample = data['Population'].sample(10, replace=True)
```

In [6]
```
print('Sample mean is ', a_sample.mean())
print('Sample variance is ', a_sample.var(ddof=1))
print('Sample standard deviation is ', a_sample.std(ddof=1))
print('Sample size is ', a_sample.shape[0])
```

Out [6]
```
Sample mean is 35.0
Sample variance is 415.333333
Sample standard deviation is 20.3797284902
Sample size is 10
```

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

In [6]

```
sample_length = 500
sample_variance_collection0=[data['Population'].sample(50,
                                            replace=True).var(ddof=0)
                                         for i in range(sample_length)]
sample_variance_collection1=[data['Population'].sample(50,
                                            replace=True).var(ddof=1)
                                         for i in range(sample_length)]
```

In [7]

```
print('Population variance is ', data['Population'].var(ddof=0))
print('Average of sample variance with n is',
   pd.DataFrame(sample_variance_collection0)[0].mean())
print('Average of sample variance with n-1 is',
   pd.DataFrame(sample_variance_collection1)[0].mean())
```

Out [7]

```
Population variance is 571.8
Average of sample variance with n is 570.311255510204
Average of sample variance with n-1 is 571.4541638378797
```

Degrees of Freedom

The number of values in calculation that are free to variate

$$n = 3, \bar{x} = 100$$

$$x_1 \qquad x_2 \qquad x_3$$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

90 80 Not Free!

$$\frac{90 + 80 + x_3}{3} = 100$$

$$\boxed{x_3 = 130}$$

Lab1： Population and Sample

Instructions

- You are going to realize the difference between population and sample. In particular, we illustrate Sample with or without replacement using python. You will compare between biased and unbiased estimator.

Population and Sample

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
# Create a Population DataFrame with 10 data

data = pd.DataFrame()
data['Population'] = [47, 48, 85, 20, 19, 13, 72, 16, 50, 60]
```

You may get different results from sampling.

In [3]:
```
# Draw sample with replacement, size=5 from Population

a_sample_with_replacement = data['Population'].sample(5, replace=True)
print(a_sample_with_replacement)
```

```
0    47
2    85
6    72
2    85
0    47
Name: Population, dtype: int64
```

In [4]:
```
# Draw sample without replacement, size=5 from Population

a_sample_without_replacement = data['Population'].sample(5, replace=False)
print(a_sample_without_replacement)
```

```
8    50
2    85
4    19
6    72
1    48
Name: Population, dtype: int64
```

Parameters and Statistics

In [5]:
```
# Calculate mean and variance
population_mean = None
population_var = None
print('Population mean is ', population_mean)
print('Population variance is', population_var)
```

```
Population mean is  43.0
Population variance is 571.8
```

Expected Output: Population mean is 43.0 Population variance is 571.8

You may get different result from sampling.

In [7]:
```
# Calculate sample mean and sample standard deviation, size =10
# You will get different mean and varince every time when you excecute the below code

a_sample = data['Population'].sample(10, replace=True)
sample_mean = a_sample.mean()
sample_var = a_sample.var()
print('Sample mean is ', sample_mean)
print('Sample variance is', sample_var)
```

```
Sample mean is  48.4
Sample variance is 712.488888889
```

Average of an unbiased estimator

In [8]:
```
sample_length = 500
sample_variance_collection=[data['Population'].sample(10, replace=True).var(ddof=1) for i in range(sample_length)]
```

In [ ]:
```
```

Population and Sample.ipynb在Github中下载

https://github.com/cloudy-sfu/QUN-Data-Analysis-in-Finance/tree/main/Labs

Jupyternote Book课堂练习
十五分钟

02

Variation of Sample

Understand Distribution of Sample Mean

we will talk about **the variation of sample mean and the distribution of the sample mean**. Knowing the variation and its rule is important to have us correctly evaluate the estimation, and validate assertions about population based on the samples. **For example**, you have historical data of 100 days. You can compute sample mean, and variance of stock return. **<u>Based on the statistics, can we show inference to the parameters? How close are the statistics to population parameters?</u>** By observing the stock data of 100 days, can we make a claim that this stock is in a upward trend? That is, mean for return is positive. All these rely on our understanding of sample mean distribution.

Sampling from normal distribution

In [1]

```
Fstsample = pd.DataFrame(np.random.normal(10, 5, size=30))
print('sample mean is ', Fstsample[0].mean())
print('sample SD is ', Fstsample[0].std(ddof=1))
```

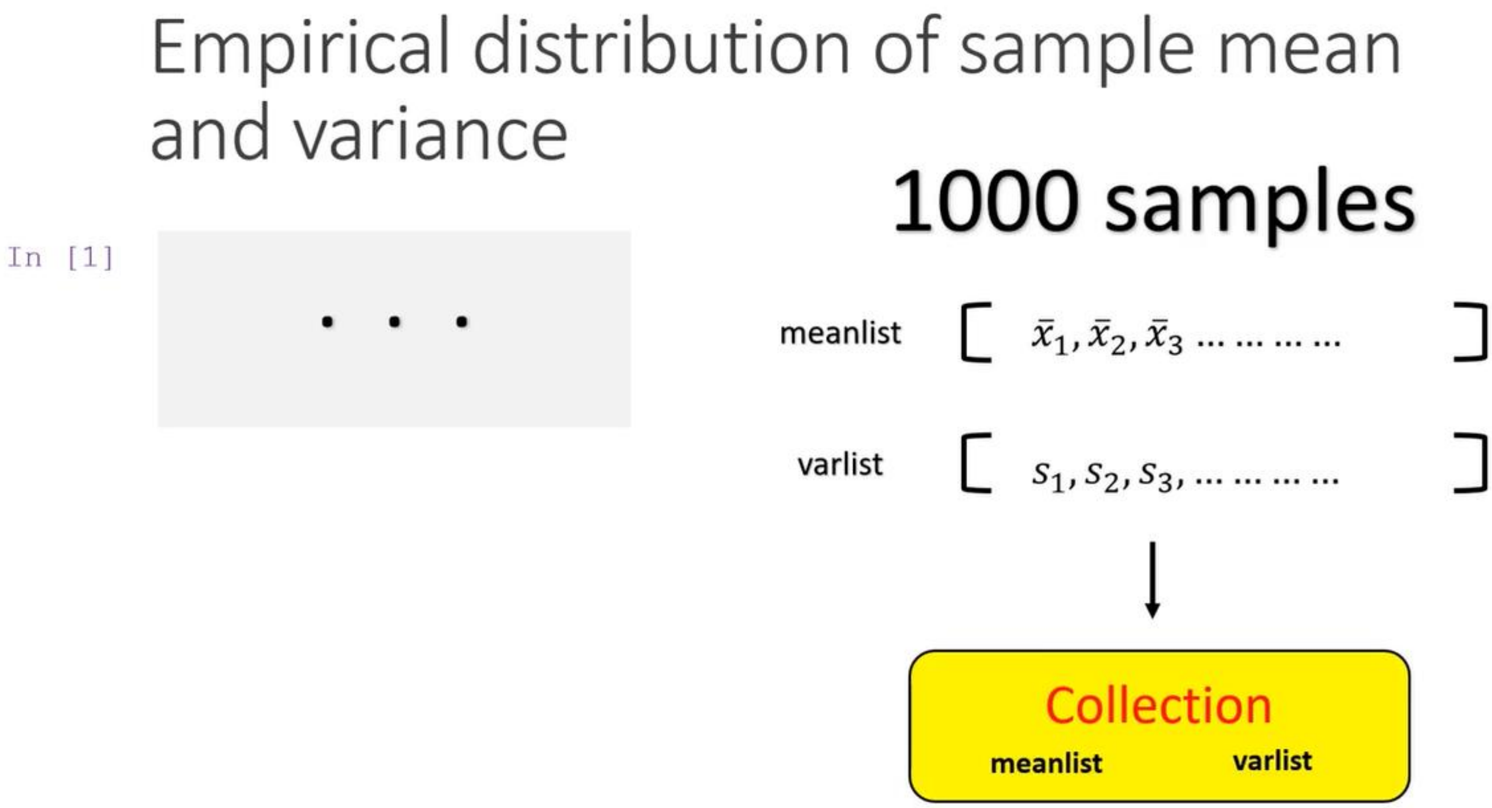
Out [1]

```
sample mean is 9.565159745164573
sample SD is 4.776758039061961
```

```
sample mean is 8.3450293121344
sample SD is 3.2311234323411
```

```
sample mean is 9.21312468950434
sample SD is 2.48950394532341
```

To see that, in this code, **we generate 1,000 samples from the same population**. We got mean and variance for each sample and saved in a DataFrame collection.



Meanlist is a name of a list to same sample means of 1,000 samples. Varlist is a name of list to save sample variances of 1,000 samples.

Empirical distribution of sample mean and variance

In [1]
```
meanlist = []
varlist = []
for t in range(1000):
    sample = pd.DataFrame(np.random.normal(10, 5, size=30))
    meanlist.append(sample[0].mean())
    varlist.append(sample[0].var(ddof=1))
```

上海立信会计金融学院
SHANGHAI LIXIN UNIVERSITY OF ACCOUNTING AND FINANCE

Finally, we build an empty DataFrame called collection, **the same meanlist and varlist in different columns of this DataFrame.**

Empirical distribution of sample mean and variance

In [1]
```
meanlist = []
varlist = []
for t in range(1000):
    sample = pd.DataFrame(np.random.normal(10, 5, size=30))
    meanlist.append(sample[0].mean())
    varlist.append(sample[0].var(ddof=1))
```

In [2]
```
collection = pd.DataFrame()
collection['meanlist'] = meanlist
collection['varlist'] = varlist
```

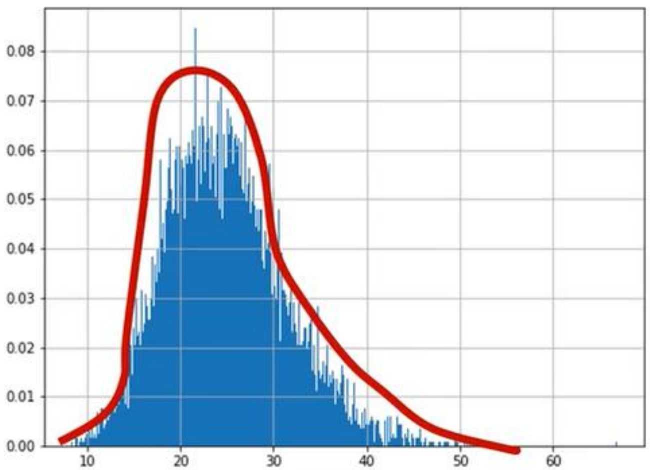
We can draw a histogram for the collection of sample means. It looks symmetric and like a normal distribution. The histogram of sample variance is not normal as you can see it is right-skewed.

Empirical distribution of sample variance

In [2]

```
collection['varlist'].hist(bins=500, normed=1)
```

Out [2]



normed: 是否将得到的直方图向量归一化

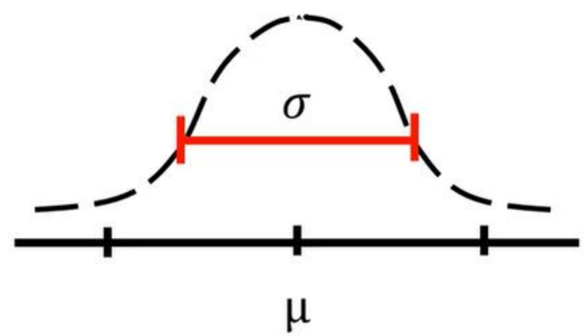
We can guess, in fact, we can mathematically prove that the sample mean has a normal distribution,**then the sample mean is also normal, with mean equal to Mu and variance equal to sigma square, divided by sample size N.**

Why variance of the sample mean is smaller than variance of a population?

Intutionally, the sample mean is the average of N individuals of population, and hence the variation of sample mean is smaller than the variation of individuals in population.

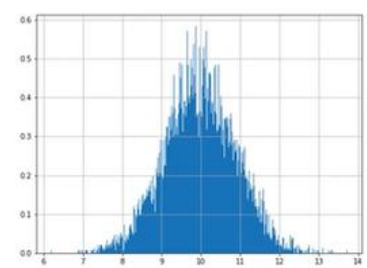
Why variance of sample mean is smaller than variance of population?

Population is normal $N(\mu, \sigma^2)$



$$\Downarrow$$

Sample mean is also normal $N(\mu, \frac{\sigma^2}{n})$



Here is a demonstration using Python. Then **blue histogram is for population**, **the red one is for the sample mean**.

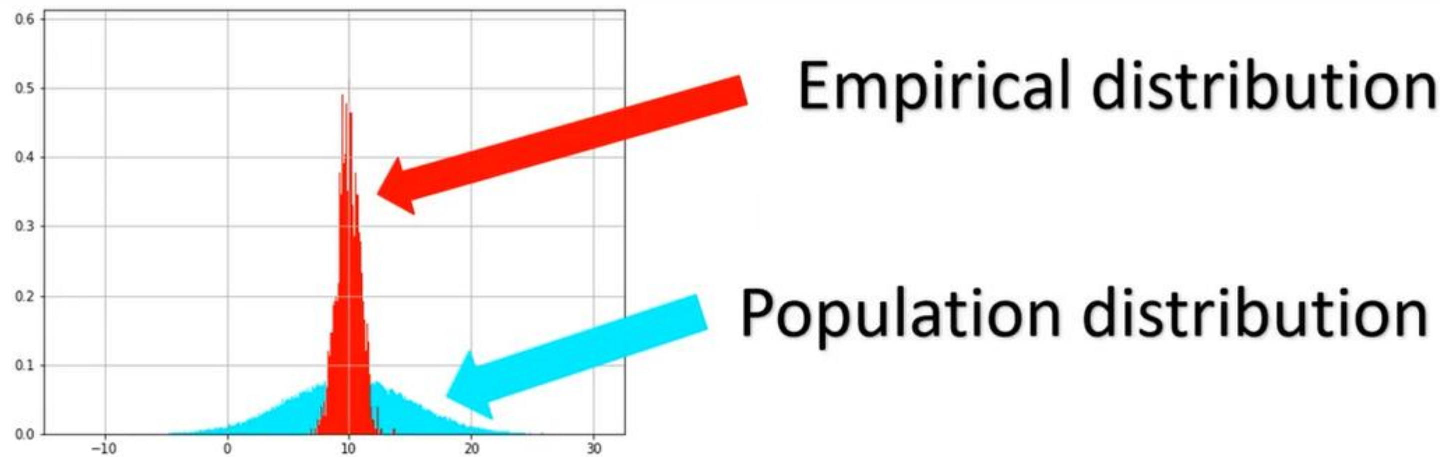
Demo in python – Population vs Empirical

In [4]

```
pop=pd.DataFrame(normal(10,5,size=100000))  # approximate to population

pop[0].hist(bins=500,color='cyan', normed=1)
collection['meanlist'].hist(bins=500,normed=1,color='red')
```

Out [4]



What if the population is not normal?

Central limit theorem

If the sample size is larger enough, the distribution of sample mean is approximately normal with $N\left(\mu, \frac{\sigma^2}{n}\right)$.

Hence, we can conclude this way, **even if the population is not normal, the sample is approximately normal if the sample size is large enough**.

Here's an example about distribution of sample mean when the population is not normal.

Sampling from general distribution

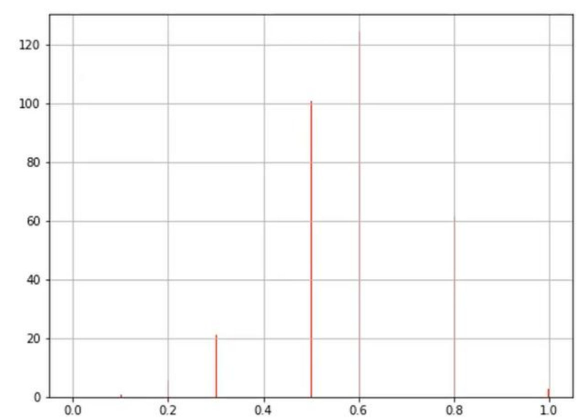
In [5]
```
samplemeanlist = []
apop =  pd.Dataframe([1, 0, 1, 0, 1])
for t in range(100000):
    sample = apop[0].sample(10, replace=True)  # small sample size
    samplemeanlist.append(sample.mean())

acollec = pd.DataFrame()
acollec['meanlist'] = samplemeanlist
```

Sampling from general distribution

In [6]
```
acollec['meanlist'].hist(bins=500, color='red', normed=1)
```

Out [6]



<u>You can see that in this histogram, for sample means, it does not look like a normal distribution.</u>

But, if you generate 100,000 samples with **large sample size 2,000**, the distribution of sample mean now looks like a normal distribution.

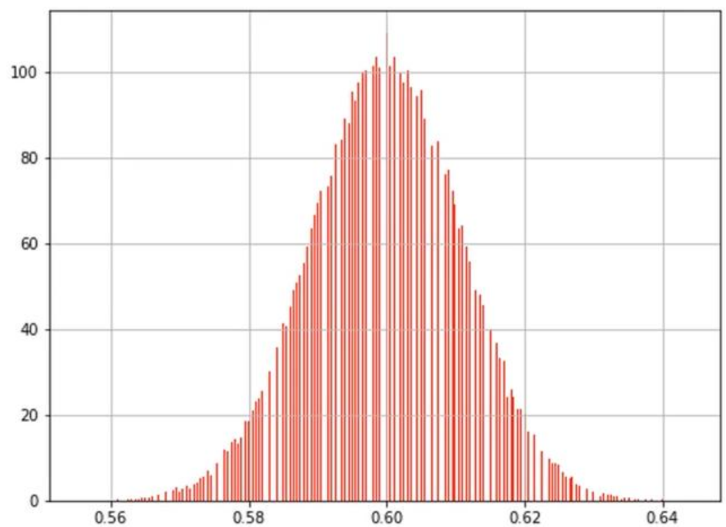
Sampling from general distribution

In [7]
```
samplemeanlist = []
for t in range(100000):
    sample = apop[0].sample(2000, replace=True)  # large sample size
    samplemeanlist.append(sample.mean())

acollec = pd.DataFrame()
acollec['meanlist'] = samplemeanlist
```

In [8]
```
acollec['meanlist'].hist(bins=500, color='red', normed=1)
```

Out [8]



上海立信会计金融学院
SHANGHAI LIXIN UNIVERSITY OF ACCOUNTING AND FINANCE

Lab2：Variation of Sample

Instructions

- **What is the distribution of sample mean for normal distributed or arbitrary distributed data?** In this Jupyter Notebook, you are going to visualize using python.

- From the Jupyter Notebook, **you realize that the average stock return is indeed approximately close to a normal distribution**, owing to the central limited theorem.

Variation of Sample

In [8]:
```
import pandas as pd
import numpy as np
from scipy.stats import norm
%matplotlib inline
```

In [4]:
```
# Sample mean and SD keep changing, but always within a certain range
Fstsample = pd.DataFrame(np.random.normal(10, 5, size=30))
print('sample mean is ', Fstsample[0].mean())
print('sample SD is ', Fstsample[0].std(ddof=1))
```

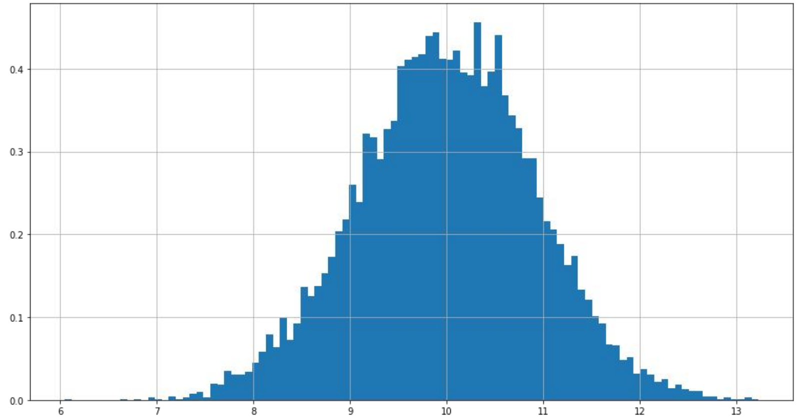
```
sample mean is  10.14251947495814
sample SD is  4.114798060251902
```

Empirical Distribution of mean

In [10]:
```
meanlist = []
for t in range(10000):
    sample = pd.DataFrame(np.random.normal(10, 5, size=30))
    meanlist.append(sample[0].mean())
```

In [11]:
```
collection = pd.DataFrame()
collection['meanlist'] = meanlist
```

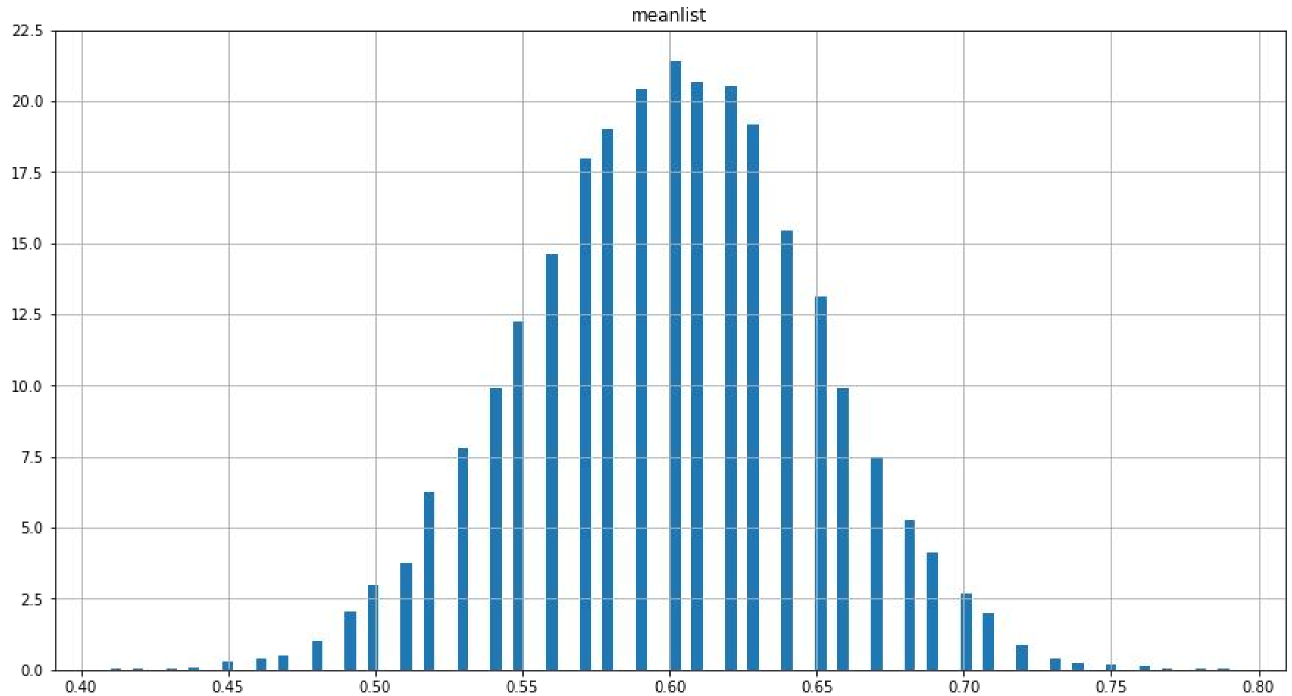
In [12]:
```
collection['meanlist'].hist(bins=100, normed=1,figsize=(15,8))
```



Sampling from arbitrary distribution

In [16]:
```
# See what central limit theorem tells you... the sample size is larger enough,
# the distribution of sample mean is approximately normal
# apop is not normal, but try to change the sample size from 100 to a larger number. The distribution of sample mean of apop
# becomes normal.
sample_size = 100
samplemeanlist = []
apop =  pd.DataFrame([1, 0, 1, 0, 1])
for t in range(10000):
    sample = apop[0].sample(sample_size, replace=True)  # small sample size
    samplemeanlist.append(sample.mean())

acollec = pd.DataFrame()
acollec['meanlist'] = samplemeanlist
acollec.hist(bins=100, normed=1,figsize=(15,8))
```



Variation of Sample.ipynb在Github中下载

https://github.com/cloudy-sfu/QUN-Data-Analysis-in-Finance/tree/main/Labs

Jupyternote Book课堂练习

十五分钟

Thank You