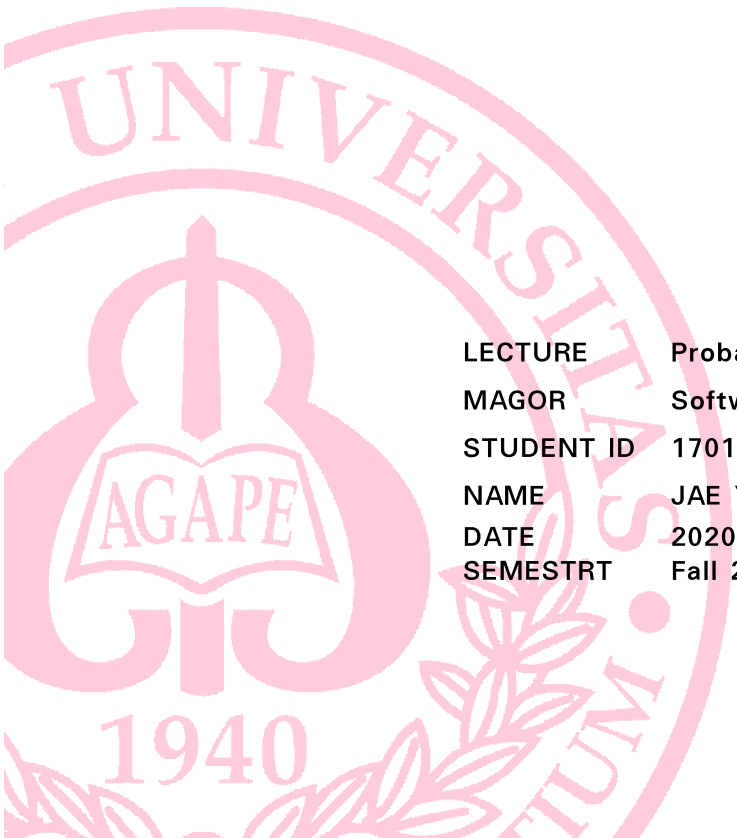
# Demonstrate the Central Limit Theorem

세종대학교




| | |
|---|---|
| LECTURE | Probability & Statistics Programming |
| MAGOR | Software |
| STUDENT ID | 17011658 |
| NAME | JAE YEON, YANG |
| DATE | 2020.06.19 |
| SEMESTRT | Fall 2020 |

## □ Contents

# 1. INTRODUCTION

## 1.1. Project Title

Demonstrate the Central Limit Theorem (CLT) in Python

## 1.2. Objective

Reflecting the knowledge of sampling distribution using Python programming.

## 1.3. Tasks Recommended

(1) A gray-scale image is a two-dimensional array of numbers, each of which represents the corresponding pixel intensity. You can obtain this array of numbers (i.e. image read) using various python packages.

(2) Consider the supplied "lena_gray.gif" gray-scale image as the population. Based on the population, you need to implement the following tasks.

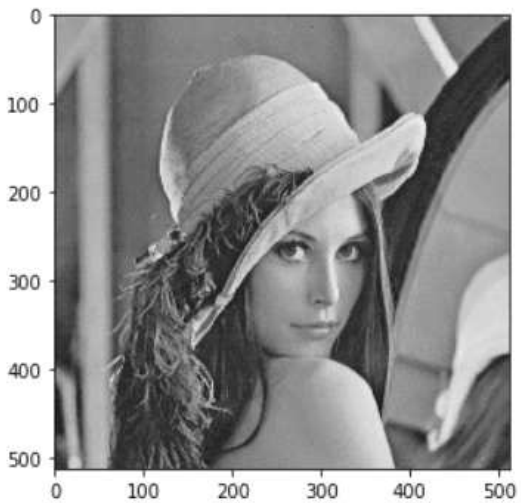
# 2. TASK1

## 2.1. Setting

Before I carried out the tasks, I import the numpy, pyplot, stats, skimage libraries.
And I've read 'lena_gray.gif' the way I learned at WEEK10.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sp
from skimage import io

image = io.imread(fname='lena_gray.gif')
io.imshow(image)
```

```
<matplotlib.image.AxesImage at 0x2375222c7c8>
```



## 2.2. Task1

**TASK1**

- Find out the population size $n$, population mean ($\mu$), population variance ($\sigma$^2), population range, minimum number, maximum number, population mode, and population median.

```
p_width = image.shape[0]
p_height = image.shape[1]
print("population size : %d * %d = %d" %(p_width, p_height, p_width*p_height))

re_img = image.reshape(-1) # Two-dimensional array to one dimension

p_mean = np.mean(re_img)
print("population mean :", p_mean)

p_var = np.var(re_img)
print("population variance :", p_var)

p_min = np.min(re_img)
print("minimum number :", p_min)

p_max = np.max(re_img)
print("maximum number :", p_max)
print("population range : %d ~ %d" %(p_min, p_max))

p_mode = sp.mode(re_img)[0][0]
print("population mode :", p_mode)

p_median = np.median(image)
print("population median :", p_median)
```

```
population size : 512 * 512 = 262144
population mean : 124.05046081542969
population variance : 2289.9760151074734
minimum number : 25
maximum number : 245
population range : 25 ~ 245
population mode : 154
population median : 129.0
```

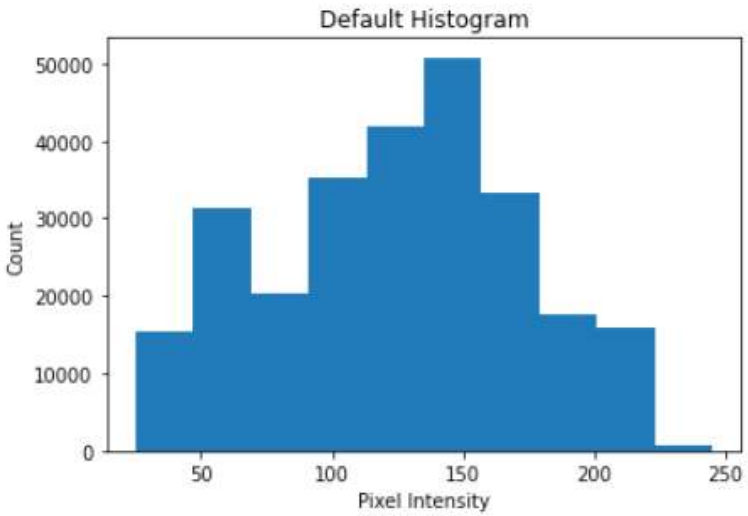
# 3. TASK2

## 3.1. Task2

**TASK2**

- Find out the histogram of the population. Comment on the population distribution.

```
# to draw histogram
re_img = np.sort(re_img) # sort array

plt.title("Default Histogram")
plt.xlabel("Pixel Intensity")
plt.ylabel("Count")
plt.hist(re_img)
plt.show()
```




## 3.2. Comment

This is'n a normal distribution.

# 4. TASK3

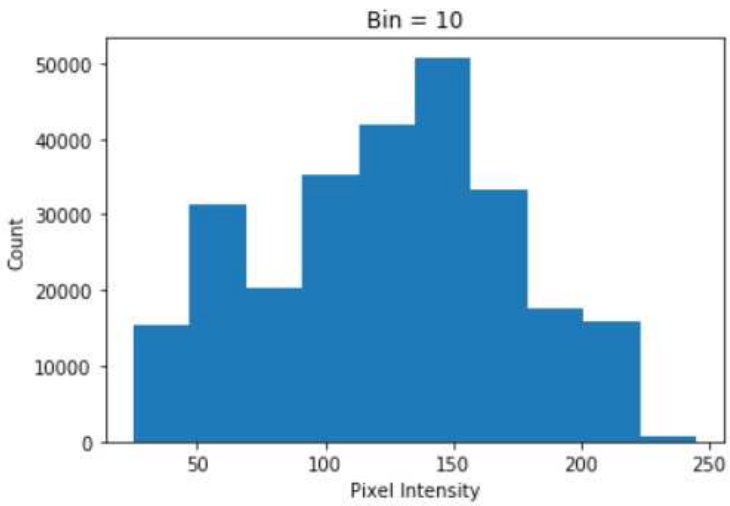
## 4.1. Task3

**TASK3**

- Investigate the histogram by changing the number of bins to 10, 100, and 1,000. Provide your observations.

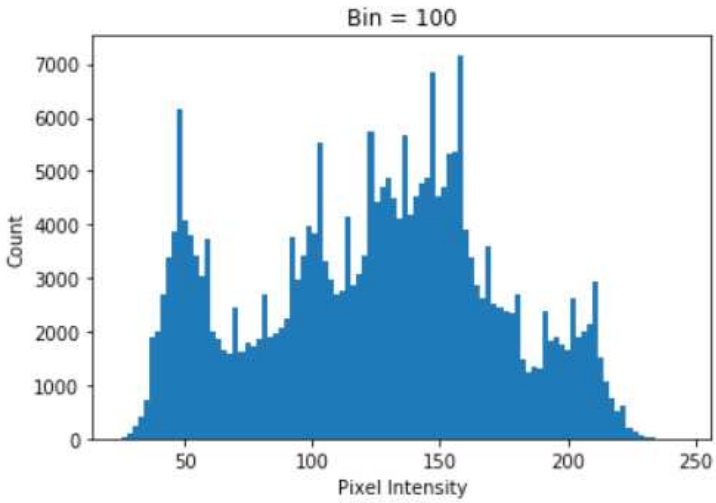
## 4.2. Bin = 10

```
plt.title("Bin = 10")
plt.xlabel("Pixel Intensity")
plt.ylabel("Count")
plt.hist(re_img, bins=10)
plt.show()
```



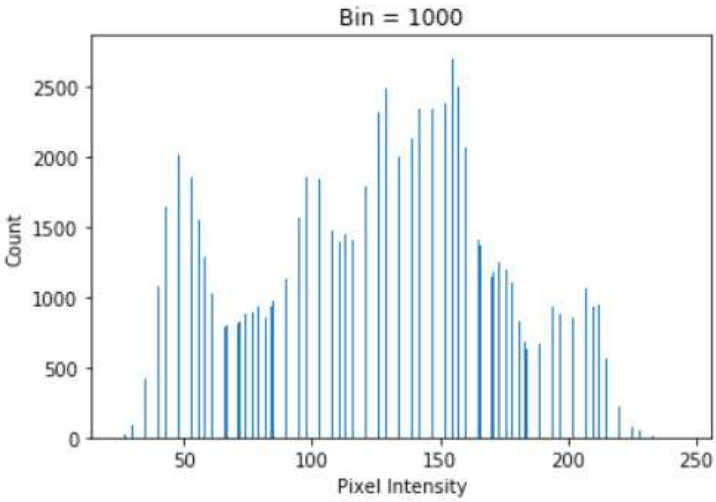
### 4.3.

```
plt.title("Bin = 100")
plt.xlabel("Pixel Intensity")
plt.ylabel("Count")
plt.hist(re_img, bins=100)
plt.show()
```




### 4.4.

```
plt.title("Bin = 1000")
plt.xlabel("Pixel Intensity")
plt.ylabel("Count")
plt.hist(re_img, bins=1000)
plt.show()
```




### 4.5. Comment

Bin means a section of the histogram. As the number of bins changes, the shape of the histogram changes, but it is still not a normal distribution.

# 5. TASK4

## 5.1. Define

**TASK4**

- Demonstrate the central limit theorem (i.e., the distribution of the sampling mean will approach towards the normal distribution with the mean $\mu$ and variance $\sigma$^2/$n$ as the sample size increases). Recommended sample sizes are 5, 10, 20, 30, 50, 100. In addition to any content that you think appropriate for this demonstration, you will include various graphical representations such as the respective histogram for each sample size.

```
def getMeanFromPopulation (population, size):
  arr = []
  for i in range(size):
    idx = np.random.randint(0, 512*512)
    arr.append(population[idx])

  mean = np.mean(arr)
  return mean
```

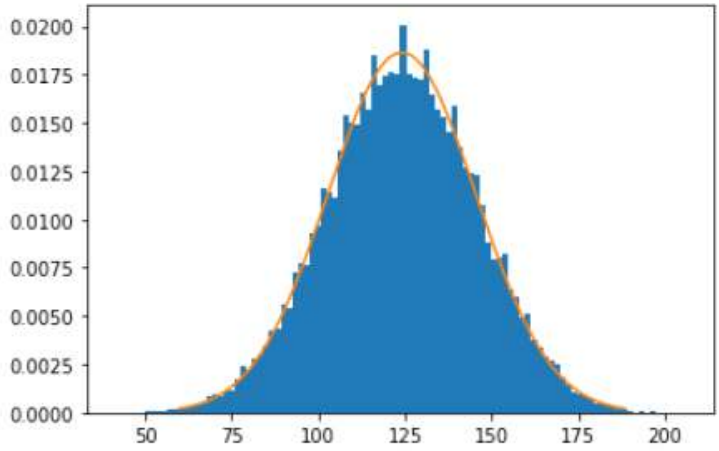
```
def getArrMean (iter, population, size):
  arr = []
  for i in range(iter):
    mean = getMeanFromPopulation(population, size)
    arr.append(mean)
  return arr
```

```
NUM_ITERATION = 50000 # proximitly infinte
```

```
def drawGraph (data, SAMPLE_SIZE):
  s_sd = (p_var/SAMPLE_SIZE)**0.5

  x = np.linspace(p_mean-s_sd*3, p_mean+s_sd*3, 1024)
  x = np.sort(x)
  y = sp.norm.pdf(x, loc=p_mean, scale=s_sd)


  plt.hist(data, bins=100, density=True)
  plt.plot(x, y)
  plt.show()

  s_mean = np.mean(data)
  print("Population mean = ", p_mean)
  print("Sampling mean = ", s_mean)
  print("Distance = ", p_mean - s_mean)
```

### 5.2. Sample size = 5

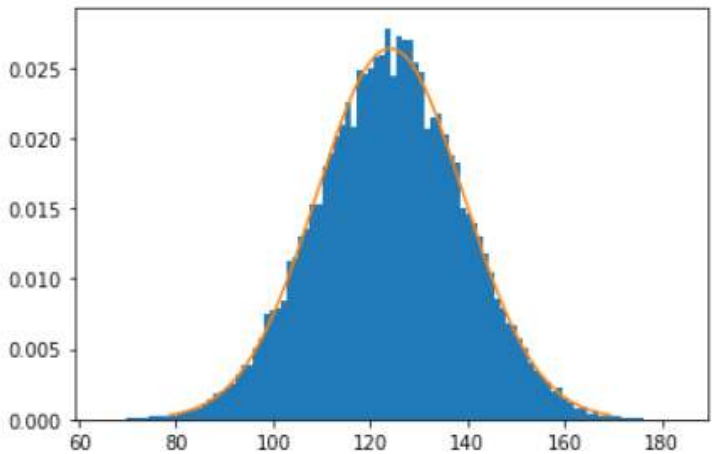
```
SAMPLE_SIZE = 5
data = getArrMean (NUM_ITERATION, re_img, SAMPLE_SIZE)

drawGraph(data, SAMPLE_SIZE)
```




```
Population mean =  124.05046081542969
Sampling mean =  124.05310399999999
Distance =  -0.002643184570302992
```

### 5.3. Sample size = 10

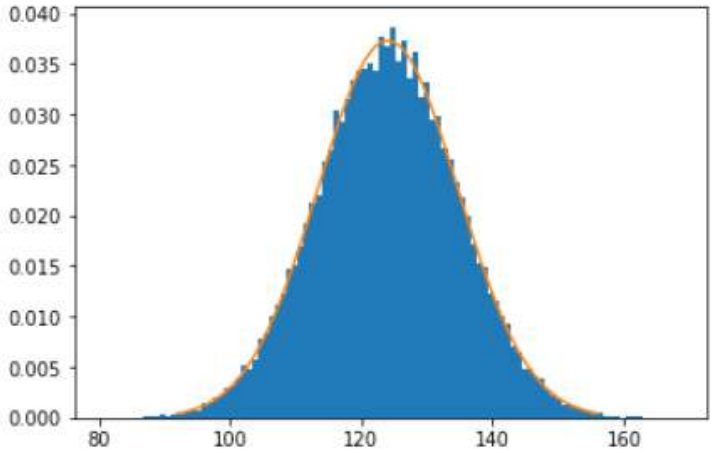
```
SAMPLE_SIZE = 10
data = getArrMean (NUM_ITERATION, re_img, SAMPLE_SIZE)

drawGraph(data, SAMPLE_SIZE)
```




```
Population mean =  124.05046081542969
Sampling mean =  124.066902
Distance =  -0.016441184570311407
```

### 5.4. Sample size = 20

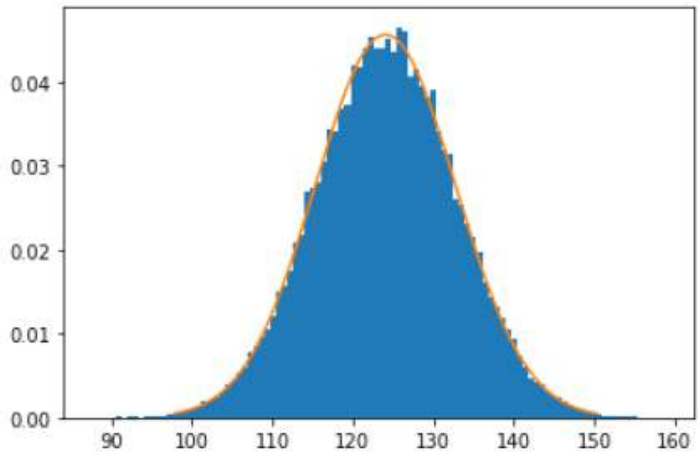
```
SAMPLE_SIZE = 20
data = getArrMean (NUM_ITERATION, re_img, SAMPLE_SIZE)

drawGraph(data, SAMPLE_SIZE)
```




```
Population mean =  124.05046081542969
Sampling mean =  124.02266399999999
Distance =  0.027796815429695698
```

### 5.5. Sample size = 30

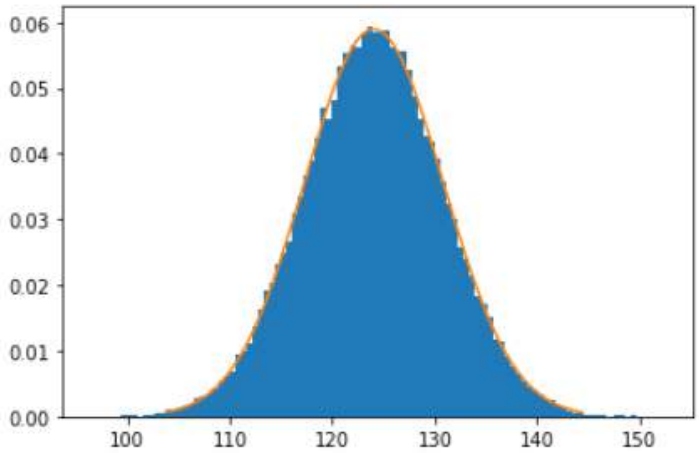
```
SAMPLE_SIZE = 30
data = getArrMean (NUM_ITERATION, re_img, SAMPLE_SIZE)

drawGraph(data, SAMPLE_SIZE)
```




```
Population mean =  124.05046081542969
Sampling mean =  124.08521666666665
Distance =  -0.034755851236965896
```

### 5.6. Sample size = 50

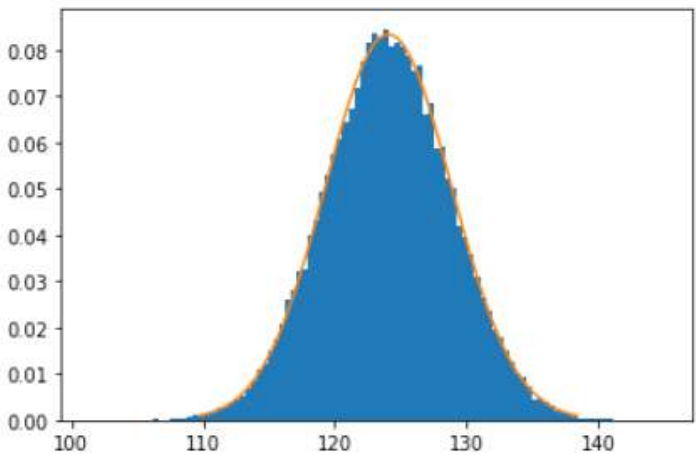
```
SAMPLE_SIZE = 50
data = getArrMean (NUM_ITERATION, re_img, SAMPLE_SIZE)

drawGraph(data, SAMPLE_SIZE)
```




```
Population mean =  124.05046081542969
Sampling mean =  124.0222348
Distance =  0.02822601542968073
```

### 5.7. Sample size = 100

```
SAMPLE_SIZE = 100
data = getArrMean (NUM_ITERATION, re_img, SAMPLE_SIZE)

drawGraph(data, SAMPLE_SIZE)
```




```
Population mean =  124.05046081542969
Sampling mean =  124.04069439999999
Distance =  0.00976641542969503
```

# 6. CLT

## 6.1. Central Limit Theorem

Central Limit Theorem(CLT) means that if the sample size n is large enough, the distribution of sample means is close to the normal distribution regardless of the population distribution shape.
A similar example is the 'Law of the Big Number'.
If you use the site below which you checked in class, you can easily understand it.
http://onlinestatbook.com/stat_sim/sampling_dist/index.html

## 6.2. Importance

Central Limit Theorem(CLT) is important because a number of important facts have been discovered and can be used to make data analysis easier.
As I saw directly in Task 4, the distribution becomes a bell shape and becomes similar to the normal distribution. I've extracted 50000 times, but if I keep extracting it, it'll be almost completely normal distribution. In addition, this normal distribution allows any type of distribution in the population.
And the mean of the sample means is similar to the mean of the population. To confirm this, I printed the population mean, sample mean, and difference for each distribution.
To sum up, we can know the mean and standard deviation almost accurately with enough samples. In addition, regardless of the distribution of the population, CLT can make it a normal distribution.
We can perform various analyses using the mean, standard deviation, and normal distribution obtained in this way.