Proving Central Limit Theorem through Uniform, Binomial, and Poisson distribution

Theorem Let W_1, W_2,... be an infinite sequence of independent random variables, each with the same distribution. Suppose that the mean μ and the variance σ^2 of $f_w(w)$ are both finite. For any numbers a and b,

$$\lim_{n \to \infty} P(a \leq \frac{W1 + \ldots + Wn - n\mu}{\sqrt{n}\sigma} \leq b) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-\frac{z^2}{2}} dz$$

The Central Limit Theorem is often stated in terms of the average of W_1,W_2, …, and W_n rather than their sum. Since

$$E\left[\frac{1}{n}(W_1 + \ \ldots \ + W_n)\right] = E(\overline{W}) = \mu \text{ and } Var\left[\frac{1}{n}(W_1 + \ \ldots \ + W_n)\right] = \frac{\sigma^2}{n}$$

Because of the Central Limit Theorem, we know that averages of any set of random variables, when suitably scaled, have distributions that can be approximated by a standard normal curve. We will show that the Central limit theorem works by looking at samples from Uniform, Binomial with different values of p, and Poisson distribution with different lambdas.

Example We set the sample size to 50. We use uniform distribution to store random values ranging from 0 to 50 in an array. We use binomial distribution to store random values from either 0 or 1 with an even success rate, i.e p = 0.5, in an array. We use poisson distribution to store random values, where lambda equals 25, in an array. See Appendix 1.

```
sample_size = 50

# Uniform distribution
uniform_distribution = np.random.uniform(low=0, high=50, size = sample_size)

# Binomial distribution
binomial_distribution = np.random.binomial(n=1, p=0.5, size=sample_size)

# Poisson distribution
poisson_distribution = np.random.poisson(lam=25, size=sample_size)
```

We iterate through each array, and combine one value from each distribution into a set. We convert this into a 2D array. See Appendix B.

```
subsets = []

# Loop through each distribution and combine one sample from each into a set
for i in range(sample_size):
    subset = [uniform_distribution[i], binomial_distribution[i], poisson_distribution[i]]
    subsets.append(subset)

# Convert subsets to a NumPy array for 1. Plotting 2. Mean Calculation
subsets_array = np.array(subsets)
```

We iterate through the 2D array, and calculate the mean of the subset. See Appendix C.

```
subset_means = np.mean(subsets_array, axis=1)
subset_means
```

Figure 1 shows the density plot for sample size of 50, compared with the normal distribution curve.

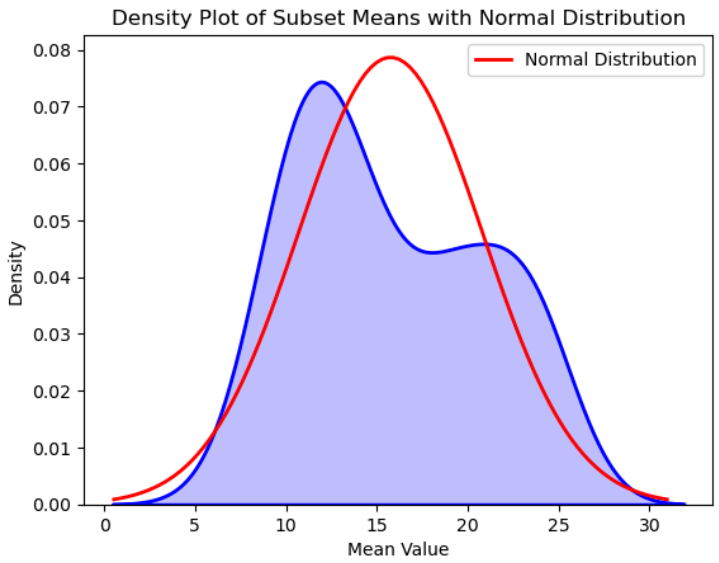


Figure 1

Comment As expected, the most common values for the mean value round off in the range of 10 to 15, and the frequencies of averages more extreme in either direction fall off rapidly. The superimposed red curve is the normal distribution, as given by the Central Limit Theorem, of the

sample averages as the number of observations, n, in each of the averages goes to ∞. How closely an observed distribution of sample means agrees with that theoretical distribution can be determined if the mean and variance of the mean value can be calculated. See Appendix D.

```
# Calculate theoretical mean and variance
theoretical_mean = (np.mean(uniform_distribution) + np.mean(binomial_distribution) + np.mean(poisson_distribution)) / 3
theoretical_variance = (np.var(uniform_distribution) + np.var(binomial_distribution) + np.var(poisson_distribution)) / 9

# Calculate the emperical mean and variance
empirical_mean = np.mean(subset_means)
empirical_variance = np.var(subset_means)

# Storing results in a dictionary
results = {
    "Theoretical Mean": theoretical_mean,
    "Empirical Mean": empirical_mean,
    "Theoretical Variance": theoretical_variance,
    "Empirical Variance": empirical_variance
}
```

The observed proportion of the same averages rounding to the theoretical mean was 0.6. This shows that about 60% of data fall within one standard deviation of the mean, which is very close to the theoretical limit. Hence, the result supports the central limit theorem.

```
# Calculate the range for one standard deviation around the theoretical mean
lower_bound = theoretical_mean - np.sqrt(theoretical_variance)
upper_bound = theoretical_mean + np.sqrt(theoretical_variance)

# Calculate the observed proportion of subset means within one standard deviation of the theoretical mean
observed_proportion_within_1_std = np.sum((subset_means >= lower_bound) & (subset_means <= upper_bound)) / len(subset_means)

# Store results
observed_proportion_within_1_std
```

```
0.6
```

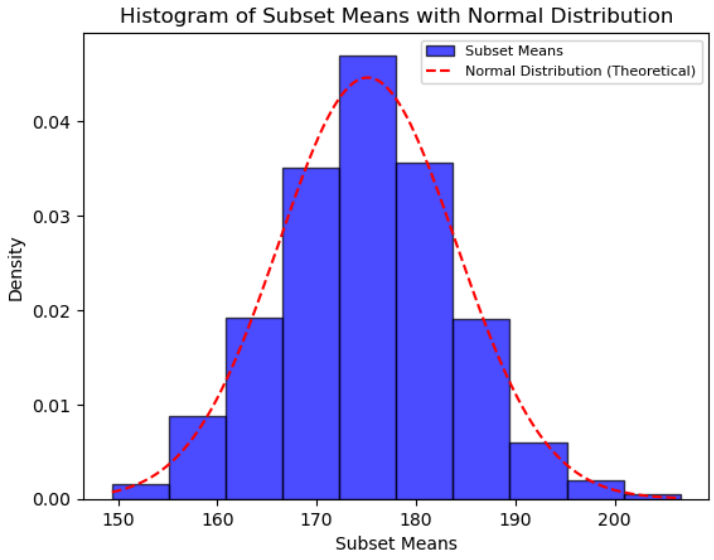
Further Evaluation While the example above proves that central limit theorem works by looking at samples from different types of distributions, a further implementation of increasing the sample size and lambda of Poisson distribution allows an accurate capture of density-scaled histogram.

```
sample_size = 1000

# Uniform distribution
uniform_distribution = np.random.uniform(low=0, high=50, size = sample_size)

# Binomial distribution
binomial_distribution = np.random.binomial(n=1, p=0.5, size=sample_size)

# Poisson distribution
poisson_distribution = np.random.poisson(lam=500, size=sample_size)
```

We increase the sample size to 1000, and lambda to 500. We follow the same method shown above to solve for the theoretical mean and variance, along with a superimposed curve of the normal distribution.

```
# Calculate the theoretical mean and variance
theoretical_mean = (np.mean(uniform_distribution) + np.mean(binomial_distribution) + np.mean(poisson_distribution)) / 3
theoretical_variance = (np.var(uniform_distribution) + np.var(binomial_distribution) + np.var(poisson_distribution)) / 9

# Plot the histogram of subset means
plt.hist(subset_means, bins=10, density=True, color='blue', edgecolor='black', alpha=0.7, label='Subset Means')

# Generate x values for the normal distribution
x = np.linspace(min(subset_means), max(subset_means), 1000)

# Calculate the corresponding y values for the normal distribution
y = norm.pdf(x, loc=theoretical_mean, scale=np.sqrt(theoretical_variance))

# Plot the normal distribution curve
plt.plot(x, y, 'r--', label='Normal Distribution (Theoretical)')

# Add labels, title, and legend
plt.title('Histogram of Subset Means with Normal Distribution')
plt.xlabel('Subset Means')
plt.ylabel('Density')
plt.legend(loc='upper right', prop={'size': 8})

# Show the plot
plt.show()
```

Figure 2 is the density-scaled histogram constructed for one such set of 1000 samples. As expected, the most common values for the subset mean round off to either 170, 175, or 180, and the frequencies of averages more extreme in either direction fall off rapidly.



Comment The result is explained when observing the type of distributions used. For uniform distribution, the random values spread across a fixed range of 0 to 50. The binomial distribution consisting of binary values 0 and 1 adds low variance. The Poisson distribution, where the lambda value is 500, dominates the overall distribution's mean and variance. Note that subset mean was calculated for each subset by the sum of each value of uniform, binomial, and poisson value. Neglecting the effect of uniform and binomial distribution, on average, the subset mean is heavily influenced by the poisson value divided by 3, which aligns closely with 170, 175, and 180.

Conclusion The most remarkable feature of the Central Limit Theorem is its generality – it holds for every sequence of independent random variables subject only to the weak restriction that all the Wi's have the same finite mean and the same finite variance. Very few random variables fail to meet those conditions. Hence, it does not matter what the shape of the distribution is – as this investigation shows, when averaging many independent samples, the distribution of the sample mean will approach a normal distribution.

Appendix A

```
uniform_distribution
```

```
array([ 9.82157842,  4.70865076, 37.77118334, 46.27322241, 34.75836709,
        8.74946087,  8.57754405, 49.55266355, 42.50345643,  9.9208241 ,
       22.82025445, 48.21926413,  9.61650458, 38.04857022,  2.63191061,
       12.31526472, 13.10322063, 22.62968331,  5.0541366 , 16.03499252,
       12.45517533,  2.45041006,  7.19928921, 28.27681513, 39.21444078,
       40.8092056 , 19.82224806, 29.99883095, 45.64115915, 42.78889116,
        3.48864183, 23.52702769,  8.3494679 , 13.84558711, 32.52439379,
        3.50061429,  8.05569871, 20.29494401, 49.91908664,  2.16880454,
       27.45786578,  7.76959706, 15.92809195, 15.19379361, 25.68975563,
       18.9933624 , 44.84264323, 10.21549603, 30.91836007,  7.16809382])
```

```
binomial_distribution
```

```
array([1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1,
       1, 0, 0, 0, 1, 1])
```

```
poisson_distribution
```

```
array([21, 30, 26, 26, 20, 26, 22, 22, 29, 28, 21, 24, 24, 27, 27, 32, 22,
       16, 22, 35, 20, 25, 20, 28, 27, 28, 24, 28, 28, 22, 19, 32, 21, 24,
       23, 26, 26, 19, 20, 29, 29, 29, 24, 28, 26, 26, 19, 27, 27, 33])
```

Appendix B

```
subsets_array
```

```
array([[ 9.82157842,  1.        , 21.        ],
       [ 4.70865076,  1.        , 30.        ],
       [37.77118334,  1.        , 26.        ],
       [46.27322241,  0.        , 26.        ],
       [34.75836709,  0.        , 20.        ],
       [ 8.74946087,  0.        , 26.        ],
       [ 8.57754405,  0.        , 22.        ],
       [49.55266355,  1.        , 22.        ],
       [42.50345643,  0.        , 29.        ],
       [ 9.9208241 ,  1.        , 28.        ],
       [22.82025445,  0.        , 21.        ],
       [48.21926413,  0.        , 24.        ],
       [ 9.61650458,  1.        , 24.        ],
       [38.04857022,  0.        , 27.        ],
       [ 2.63191061,  0.        , 27.        ],
       [12.31526472,  1.        , 32.        ],
       [13.10322063,  1.        , 22.        ],
       [22.62968331,  0.        , 16.        ],
       [ 5.0541366 ,  1.        , 22.        ],
       [16.03499252,  0.        , 35.        ],
       [12.45517533,  0.        , 20.        ],
       [ 2.45041006,  0.        , 25.        ],
       [ 7.19928921,  1.        , 20.        ],
       [28.27681513,  0.        , 28.        ],
       [39.21444078,  1.        , 27.        ],
       [40.8092056 ,  1.        , 28.        ],
       [19.82224806,  1.        , 24.        ],
       [29.99883095,  0.        , 28.        ],
       [45.64115915,  1.        , 28.        ],
       [42.78889116,  1.        , 22.        ],
       [ 3.48864183,  0.        , 19.        ],
       [23.52702769,  1.        , 32.        ],
       [ 8.3494679 ,  1.        , 21.        ],
       [13.84558711,  0.        , 24.        ],
       [32.52439379,  1.        , 23.        ],
       [ 3.50061429,  0.        , 26.        ],
       [ 8.05569871,  0.        , 26.        ],
       [20.29494401,  0.        , 19.        ],
       [49.91908664,  0.        , 20.        ],
       [ 2.16880454,  1.        , 29.        ],
       [27.45786578,  1.        , 29.        ],
       [ 7.76959706,  0.        , 29.        ],
       [15.92809195,  1.        , 24.        ],
       [15.19379361,  1.        , 28.        ],
       [25.68975563,  1.        , 26.        ],
       [18.9933624 ,  0.        , 26.        ],
       [44.84264323,  0.        , 19.        ],
       [10.21549603,  0.        , 27.        ],
       [30.91836007,  1.        , 27.        ],
       [ 7.16809382,  1.        , 33.        ]])
```

Appendix C

```
subset_means
```

```
array([10.60719281, 11.90288359, 21.59039445, 24.09107414, 18.25278903,
       11.58315362, 10.19251468, 24.18422118, 23.83448548, 12.97360803,
       14.60675148, 24.07308804, 11.53883486, 21.68285674,  9.87730354,
       15.10508824, 12.03440688, 12.8765611 ,  9.35137887, 17.01166417,
       10.81839178,  9.15013669,  9.39976307, 18.75893838, 22.40481359,
       23.2697352 , 14.94074935, 19.33294365, 24.88038638, 21.92963039,
        7.49621394, 18.84234256, 10.1164893 , 12.6151957 , 18.8414646 ,
        9.8335381 , 11.35189957, 13.09831467, 23.30636221, 10.72293485,
       19.15262193, 12.25653235, 13.64269732, 14.73126454, 17.56325188,
       14.99778747, 21.28088108, 12.40516534, 19.63945336, 13.72269794])
```

Appendix D

```
results
```

```
{'Theoretical Mean': 15.757456962016107,
 'Empirical Mean': 15.757456962016109,
 'Theoretical Variance': 26.86248303146769,
 'Empirical Variance': 25.72966297544664}
```