Q3P1

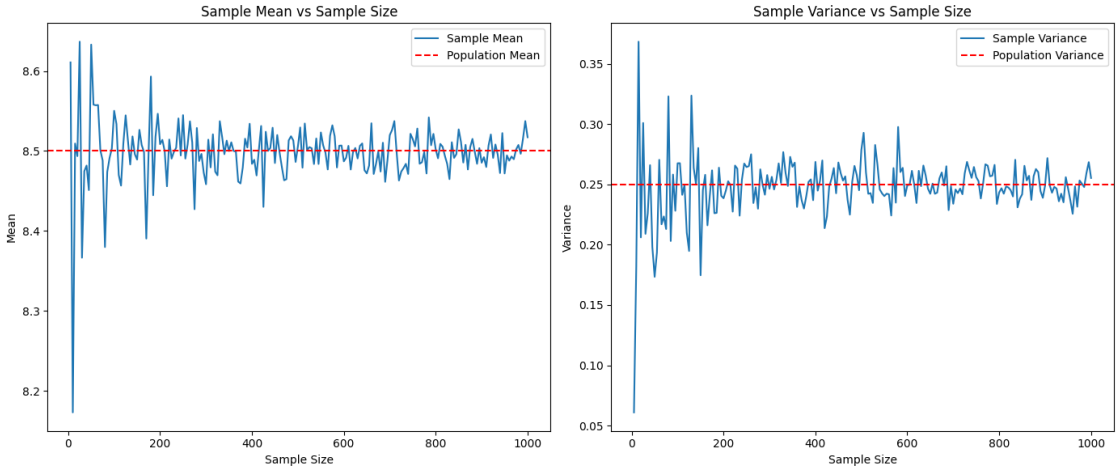
October 24, 2024

0.1 Part 1: Point Estimation; Estimating the Average Battery Life (20 points)

6. (8 points) Dynamically simulate different sample sizes n [5, 1000, step=5] and generate samples from a normal distribution with a known population mean = 8.5 and standard deviation = 0.5. Vary n and compare how the sample mean and variance behave as the sample size changes.

[1]:
```
import numpy as np

import matplotlib.pyplot as plt

# Parameters
mu = 8.5
sigma = 0.5
sample_sizes = range(5, 1001, 5)

# Lists to store the sample means and variances
sample_means = []
sample_variances = []

# Simulate samples and calculate means and variances
for n in sample_sizes:
    samples = np.random.normal(mu, sigma, n)
    sample_means.append(np.mean(samples))
    sample_variances.append(np.var(samples))

# Plotting the results
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
plt.plot(sample_sizes, sample_means, label='Sample Mean')
plt.axhline(y=mu, color='r', linestyle='--', label='Population Mean')
plt.xlabel('Sample Size')
plt.ylabel('Mean')
plt.title('Sample Mean vs Sample Size')
plt.legend()

plt.subplot(1, 2, 2)
```

```
plt.plot(sample_sizes, sample_variances, label='Sample Variance')
plt.axhline(y=sigma**2, color='r', linestyle='--', label='Population Variance')
plt.xlabel('Sample Size')
plt.ylabel('Variance')
plt.title('Sample Variance vs Sample Size')
plt.legend()

plt.tight_layout()
plt.show()
```



0.1.1 USING DYNAMIC INTERACTIVE PLOTS

[5]:
```
import ipywidgets as widgets
from ipywidgets import interact

def plot_sample_statistics(sample_size):
    samples = np.random.normal(mu, sigma, sample_size)
    sample_mean = np.mean(samples)
    sample_variance = np.var(samples)

    plt.figure(figsize=(14, 6))

    plt.subplot(1, 2, 1)
    plt.axhline(y=mu, color='r', linestyle='--', label='Population Mean')
    plt.bar(['Sample Mean'], [sample_mean], color='blue')
    plt.ylim(mu - 1, mu + 1)
    plt.ylabel('Mean')
    plt.title('Sample Mean')
    plt.legend()

    plt.subplot(1, 2, 2)
```

```
    plt.axhline(y=sigma**2, color='r', linestyle='--', label='Population␣
 ↪Variance')
    plt.bar(['Sample Variance'], [sample_variance], color='green')
    plt.ylim(0, sigma**2 + 1)
    plt.ylabel('Variance')
    plt.title('Sample Variance')
    plt.legend()

    plt.tight_layout()
    plt.show()

interact(plot_sample_statistics, sample_size=widgets.IntSlider(min=5, max=1000,␣
 ↪step=5, value=100))
```

```
interactive(children=(IntSlider(value=100, description='sample_size', max=1000,␣
 ↪min=5, step=5), Output()), _do…
```

[5]:
```
<function __main__.plot_sample_statistics(sample_size)>
```

The more you increase the sample size, the closer it gets to the true population mean and true population variance