Homework-1

Farhan Alam

19th January 2025

Question 1

You are a data scientist working for a smartphone manufacturer. The company claims that the battery life of their latest model follows a normal distribution with a mean (μ) of 20 hours and a standard deviation (σ) of 2 hours. However, you suspect that the actual battery life might differ due to manufacturing variability. To investigate this, you decide to simulate and analyze battery life data.

1. **Simulation and Maximum Likelihood Estimation (MLE):**

   Write a function to simulate n_1 smartphone battery life measurements based on the claimed distribution ($\mu = 20$, $\sigma = 2$). Use Maximum Likelihood Estimation (MLE) to estimate the mean and standard deviation of the battery life from your simulated data.

   Plot a histogram of the simulated data and overlay the normal Probability Density Function (PDF) using your estimated parameters.

   Experiment with different values of n_1 (e.g., 10, 100, 1000) and n_2 (e.g., 100, 1000) and observe how the estimates change.

2. **Repeated Simulation and Estimation Analysis:**

   Repeat the simulation n_2 times (e.g., 1000 trials) and plot a histogram of the estimated means and standard deviations. Mark the true values ($\mu = 20$, $\sigma = 2$) on the same plot.

   Experiment with different values of n_1 (e.g., 10, 100, 1000) and n_2 (e.g., 100, 1000) and observe how the estimates change.

Are the estimators for μ and σ biased or unbiased? Discuss your observations and provide suggestions based on your findings.

Introduction

For the question, we have been given a gaussian distribution with ($\mu = 20$, $\sigma = 2$) . now we generate samples of size n1 from the distribution n2 and then compare the obtained mean with actual mean i. is 20 and obtained median with acutal median i.e 2.

Data

Generated during the code.

Methodology

The following python code was used for the question-:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

def simulate_and_mle(n1, mu=20, sigma=2):
    data = np.random.normal(loc=mu, scale=sigma, size=n1)
    mu_hat = np.mean(data)
    sigma_hat = np.std(data, ddof=0)
    plt.figure(figsize=(8, 5))
    count, bins, _ = plt.hist(data, bins=15, density=True,
        alpha=0.6, color='skyblue', edgecolor='black', label=
        'Simulated Data')
    x = np.linspace(min(bins), max(bins), 100)
    plt.plot(x, norm.pdf(x, mu_hat, sigma_hat), 'r-', label=
        f'Fitted Normal PDF\n$\mu$={mu_hat:.2f}, $\sigma$={
        sigma_hat:.2f}')
    plt.title(f'Simulation of Battery Life (n1 = {n1})')
    plt.xlabel('Battery Life (hours)')
    plt.ylabel('Density')
    plt.legend()
    plt.show()

    return mu_hat, sigma_hat
simulate_and_mle(10)
simulate_and_mle(100)
simulate_and_mle(1000)
```

The code for part 2 is as follows:-

```
def repeated_simulation(n1, n2, true_mu=20, true_sigma=2):
    estimated_means = []
    estimated_sigmas = []

    for _ in range(n2):
        data = np.random.normal(loc=true_mu, scale=
            true_sigma, size=n1)
        mle_mu = np.mean(data)
        mle_sigma = np.std(data, ddof=0)  # MLE uses ddof=0
        estimated_means.append(mle_mu)
```

```python
        estimated_sigmas.append(mle_sigma)

    # Plot histogram of estimated means
    plt.hist(estimated_means, bins=15, color='lightgreen', edgecolor='black', alpha=0.7)
    plt.axvline(true_mu, color='red', linestyle='dashed', linewidth=2, label=f'True     = {true_mu}')
    plt.title(f"Histogram of Estimated Means (n1={n1}, n2={n2})")
    plt.xlabel("Estimated Mean")
    plt.ylabel("Frequency")
    plt.legend()
    plt.show()

    # Plot histogram of estimated sigmas
    plt.hist(estimated_sigmas, bins=15, color='orange', edgecolor='black', alpha=0.7)
    plt.axvline(true_sigma, color='red', linestyle='dashed', linewidth=2, label=f'True     = {true_sigma}')
    plt.title(f"Histogram of Estimated Standard Deviations (n1={n1}, n2={n2})")
    plt.xlabel("Estimated Sigma")
    plt.ylabel("Frequency")
    plt.legend()
    plt.show()

    # Return for further analysis if needed
    return np.mean(estimated_means), np.mean(estimated_sigmas)

# Example usage:
repeated_simulation(n1=100, n2=1000)
```

0.1 Step-by-Step Explanation

1. **Data Simulation:**

   - The function generates n_1 random samples from a normal distribution with a true mean $\mu = 20$ hours and standard deviation $\sigma = 2$ hours.
   - `np.random.normal()` is used to simulate the battery life data.

2. **MLE Estimation of Parameters:**

   - The sample mean $\hat{\mu}$ is calculated as:

$$\hat{\mu} = \frac{1}{n_1} \sum_{i=1}^{n_1} x_i$$

- The sample standard deviation (MLE) $\hat{\sigma}$ is computed with `ddof=0`:

$$\hat{\sigma} = \sqrt{\frac{1}{n_1} \sum_{i=1}^{n_1} (x_i - \hat{\mu})^2}$$

- Setting `ddof=0` ensures population standard deviation is calculated, which aligns with MLE.

3. **Histogram Plotting:**
   - A histogram of the simulated battery life data is plotted with 15 bins.
   - `density=True` ensures the histogram is normalized.
4. **Overlaying the Fitted Normal PDF:**
   - The estimated normal distribution is plotted over the histogram.
   - The probability density function (PDF) is calculated using the estimated parameters $\hat{\mu}$ and $\hat{\sigma}$.
5. **Display and Return Values:**
   - The plot is displayed with appropriate labels and legend.
   - The estimated mean and standard deviation are returned for further analysis.

beginitemize

`n1`: Number of samples drawn in each simulation.

`n2`: Number of repeated simulations.

`true_mu`: The true mean of the normal distribution.

`true_sigma`: The true standard deviation of the normal distribution.

0.2 Simulation Process

For each of the n_2 iterations, the following steps are performed:

1. Generate n_1 random samples from the normal distribution $N(\mu, \sigma^2)$ using:

$$\text{data} \sim \mathcal{N}(\text{true\_mu}, \text{true\_sigma}^2)$$

2. Compute the MLE of the mean, which is simply the sample mean:

$$\hat{\mu}_{\text{MLE}} = \frac{1}{n_1} \sum_{i=1}^{n_1} x_i$$

3. Compute the MLE of the standard deviation, which is the square root of the sample variance using $ddof = 0$ (population standard deviation):

$$\hat{\sigma}_{\text{MLE}} = \sqrt{\frac{1}{n_1}\sum_{i=1}^{n_1}(x_i - \hat{\mu}_{\text{MLE}})^2}$$

4. Store $\hat{\mu}_{\text{MLE}}$ and $\hat{\sigma}_{\text{MLE}}$ for analysis.

0.3 Visualization

After running n_2 simulations:

- A histogram of the estimated means $\hat{\mu}_{\text{MLE}}$ is plotted to visualize the distribution of estimates around the true mean.
- A histogram of the estimated standard deviations $\hat{\sigma}_{\text{MLE}}$ is plotted to observe the variability in the estimation of σ.

The red dashed line in each plot represents the true parameter value used for simulation.

0.4 Output

Finally, the function returns the average of the estimated means and standard deviations over all simulations:

$$\text{Mean of } \hat{\mu}_{\text{MLE}} = \frac{1}{n_2}\sum_{j=1}^{n_2}\hat{\mu}_{\text{MLE}}^{(j)}$$

$$\text{Mean of } \hat{\sigma}_{\text{MLE}} = \frac{1}{n_2}\sum_{j=1}^{n_2}\hat{\sigma}_{\text{MLE}}^{(j)}$$

0.5 Example Usage

```
repeated_simulation(n1=100, n2=1000)
```

This runs the simulation with 100 samples per iteration, repeated 1000 times. It provides insight into the bias and variability of the MLE estimators for the normal distribution parameters.

Results and Observation

The outputs obtained were as follows:-

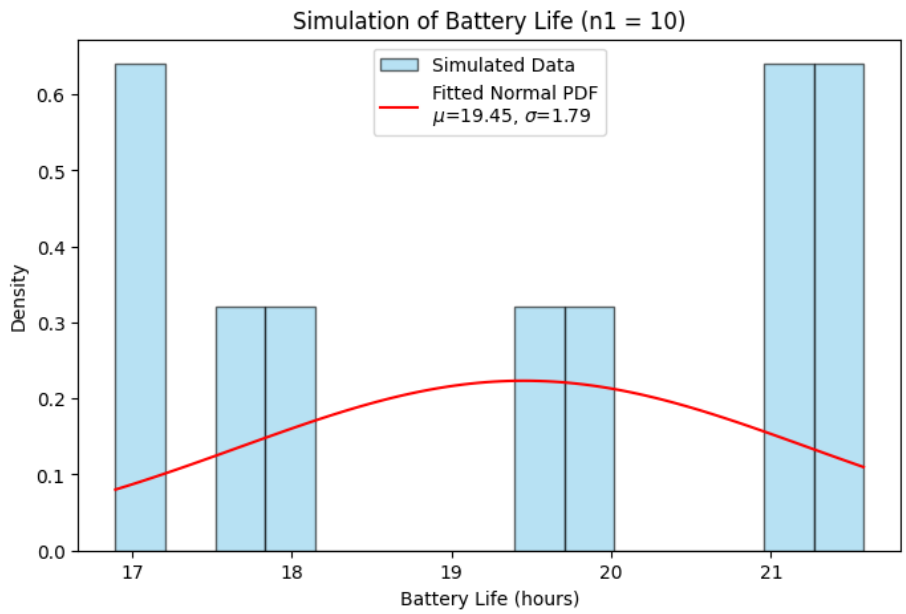


Figure 1:

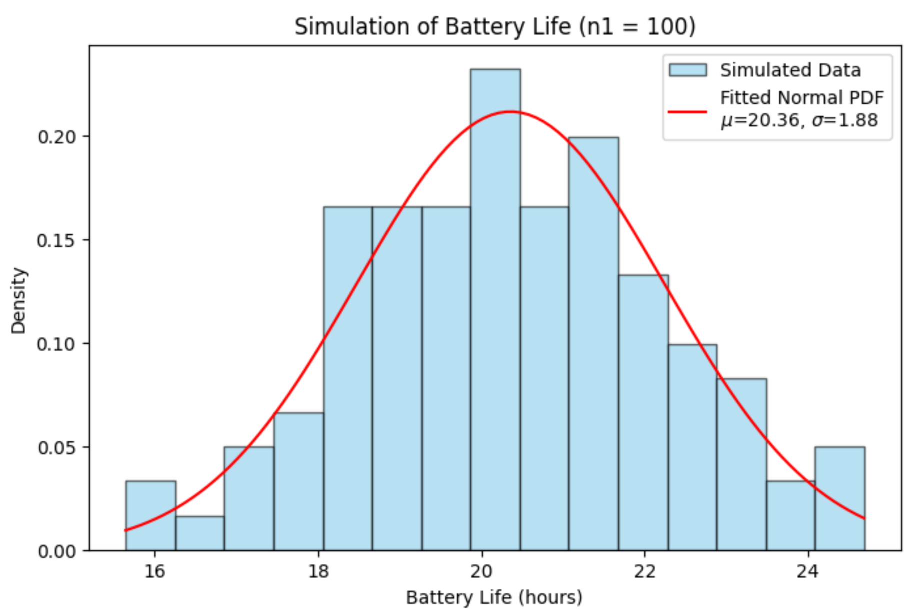


Figure 2:

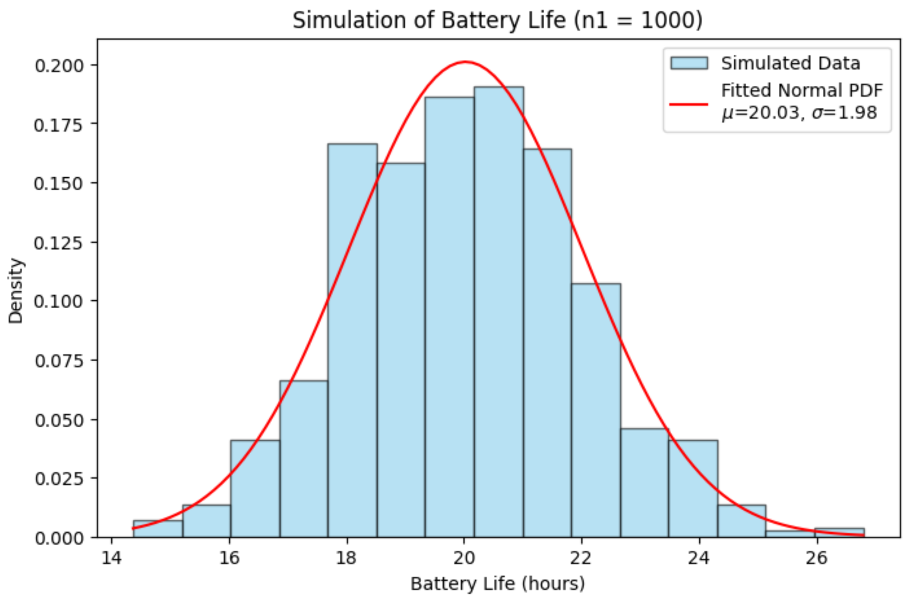

Simulation of Battery Life (n1 = 1000)
Simulated Data
Fitted Normal PDF
μ=20.03, σ=1.98
Density
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
14
16
18
20
22
24
26
Battery Life (hours)

Figure 3:

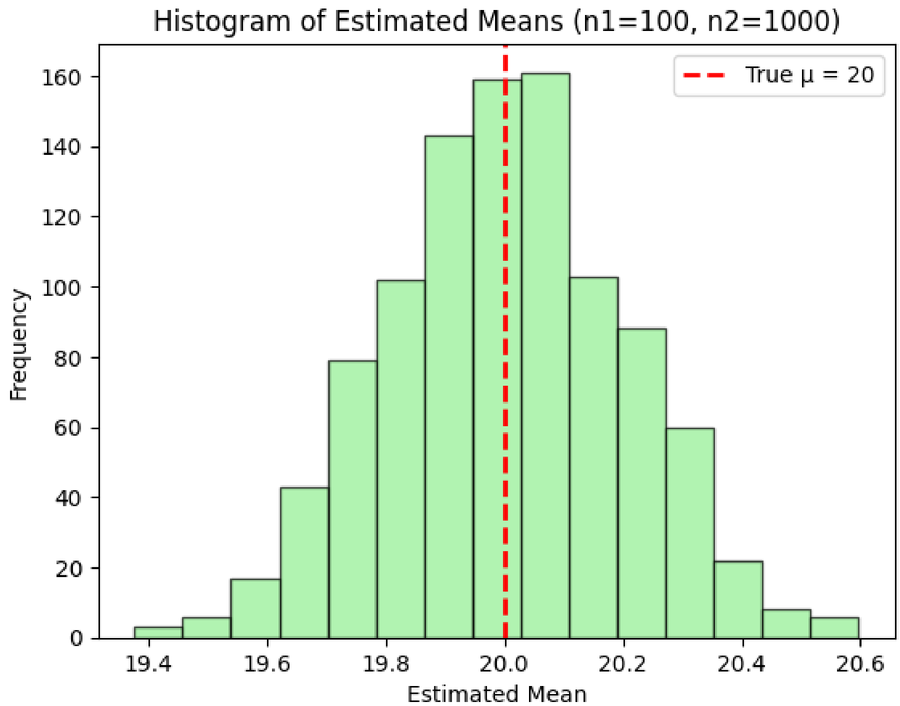

Histogram of Estimated Means (n1=100, n2=1000)
True μ = 20
Frequency
Estimated Mean
0
20
40
60
80
100
120
140
160
19.4
19.6
19.8
20.0
20.2
20.4
20.6

Figure 4:

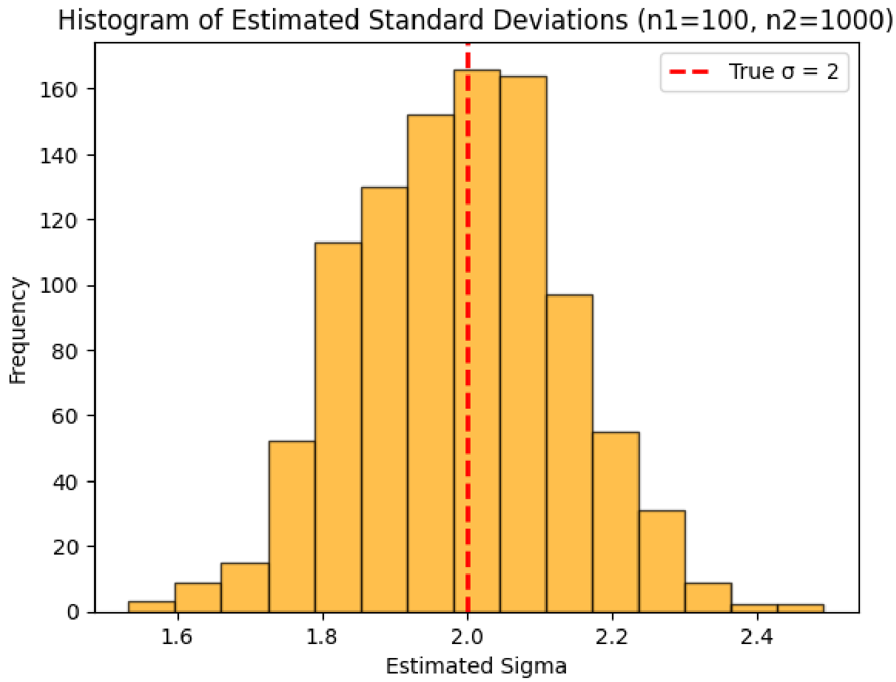


Figure 5:

- Running the function with different values of n_1 (10, 100, 1000) demonstrates how increasing the sample size improves the accuracy of the estimates.
- As n_1 increases, the estimated $\hat{\mu}$ and $\hat{\sigma}$ become closer to the true values ($\mu = 20$, $\sigma = 2$).
- The histogram becomes smoother, and the fitted normal PDF aligns better with the data for larger n_1.

1 Question 2

You are working on a project to measure the temperature of a chemical reaction using a sensor. The true temperature (X) follows a normal distribution with a mean (μ) of 50°C and a standard deviation (σ) of 5°C. However, the sensor introduces some random noise (η) due to calibration issues, where η is uniformly distributed between -1°C and 1°C. The measured temperature is given by $Y = X + \eta$.

1. **Simulation and MLE Estimation:**

Simulate n_1 temperature measurements (Y) by adding the sensor noise to the true temperature (X). Use Maximum Likelihood Estimation (MLE) to estimate the mean and standard deviation of the true temperature (X) from the noisy measurements (Y).

Plot a histogram of the noisy measurements and overlay the normal Probability Density Function (PDF) using your estimated parameters.

2. **Repeated Simulation and Analysis:**

   Repeat this experiment n_2 times and plot a histogram of the estimated means and standard deviations. Mark the true values ($\mu = 50$, $\sigma = 5$) on the same plot.

   Compare your results with part (a). How does the sensor noise affect your ability to estimate the true temperature? Are the estimators still unbiased? Discuss your findings.

2 Introduction

We need to introduce bias to our random vairbale with help of a uniform distribution

3 Data

Generated during code

4 Methodology

The code used is as follows:-

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
def simulate_temperature_mle(n1, true_mu=50, true_sigma=5):
    X = np.random.normal(loc=true_mu, scale=true_sigma, size
        =n1)
    noise = np.random.uniform(low=-1, high=1, size=n1)
    Y = X + noise
    mu_hat = np.mean(Y)
    sigma_hat = np.std(Y, ddof=0)
    plt.figure(figsize=(8, 5))
    count, bins, _ = plt.hist(Y, bins=15, density=True,
        alpha=0.6, color='lightgreen', edgecolor='black',
        label='Noisy Measurements')
    x = np.linspace(min(bins), max(bins), 100)
```

```
    plt.plot(x, norm.pdf(x, mu_hat, sigma_hat), 'r-', label=f'Fitted Normal PDF\n$\\mu$={mu_hat:.2f}, $\\sigma$={sigma_hat:.2f}')
    plt.title(f'Temperature Measurement Simulation (n1 = {n1})')
    plt.xlabel('Measured Temperature ( C )')
    plt.ylabel('Density')
    plt.legend()
    plt.show()

    return mu_hat, sigma_hat
for n1 in [10, 100, 1000]:
    simulate_temperature_mle(n1)
def repeat_simulation(n1, n2, true_mu=50, true_sigma=5):
    mu_estimates = []
    sigma_estimates = []

    for _ in range(n2):
        mu_hat, sigma_hat = simulate_temperature_mle_single(n1, true_mu, true_sigma)
        mu_estimates.append(mu_hat)
        sigma_estimates.append(sigma_hat)

    plt.figure(figsize=(8, 5))
    plt.hist(mu_estimates, bins=15, color='skyblue', edgecolor='black', alpha=0.7)
    plt.axvline(x=true_mu, color='red', linestyle='--', label=f'True    = {true_mu}')
    plt.title(f'Histogram of Estimated Means (n1={n1}, n2={n2})')
    plt.xlabel('Estimated Mean ( C )')
    plt.ylabel('Frequency')
    plt.legend()
    plt.show()

    plt.figure(figsize=(8, 5))
    plt.hist(sigma_estimates, bins=15, color='orange', edgecolor='black', alpha=0.7)
    plt.axvline(x=true_sigma, color='red', linestyle='--', label=f'True    = {true_sigma}')
    plt.title(f'Histogram of Estimated Std Deviations (n1={n1}, n2={n2})')
    plt.xlabel('Estimated Std Deviation ( C )')
    plt.ylabel('Frequency')
    plt.legend()
    plt.show()

    print(f"Mean of estimated    : {np.mean(mu_estimates):.3f}")
    print(f"Mean of estimated    : {np.mean(sigma_estimates)
```

```
        :.3f}")

def simulate_temperature_mle_single(n1, true_mu=50,
    true_sigma=5):
    X = np.random.normal(loc=true_mu, scale=true_sigma, size
        =n1)
    noise = np.random.uniform(low=-1, high=1, size=n1)
    Y = X + noise
    mu_hat = np.mean(Y)
    sigma_hat = np.std(Y, ddof=0)
    return mu_hat, sigma_hat


n2 = 1000
repeat_simulation(n1=10, n2=n2)
repeat_simulation(n1=100, n2=n2)
repeat_simulation(n1=1000, n2=n2)
```

Temperature Measurement Simulation - Code Explanation

Overview

The following Python code simulates the measurement of a chemical reaction's temperature, where the true temperature X follows a normal distribution with mean $\mu = 50°C$ and standard deviation $\sigma = 5°C$. Sensor noise η, uniformly distributed between $[-1°C, 1°C]$, is added to X to obtain noisy measurements $Y = X + \eta$. The Maximum Likelihood Estimates (MLE) of μ and σ are then computed from these noisy measurements.

Part (i) - Single Simulation and Histogram

Function: `simulate_temperature_mle(n1, true_mu=50, true_sigma=5)`

- **Step 1:** Generate n_1 samples of true temperature:

$$X_i \sim \mathcal{N}(\mu, \sigma^2)$$

- **Step 2:** Generate n_1 samples of uniform sensor noise:

$$\eta_i \sim \mathcal{U}(-1, 1)$$

- **Step 3:** Compute noisy measurements:

$$Y_i = X_i + \eta_i$$

- **Step 4:** Compute MLE estimates of the mean and standard deviation from noisy measurements:

$$\hat{\mu} = \frac{1}{n_1} \sum_{i=1}^{n_1} Y_i$$

$$\hat{\sigma} = \sqrt{\frac{1}{n_1} \sum_{i=1}^{n_1} (Y_i - \hat{\mu})^2}$$

- **Step 5:** Plot a histogram of the noisy measurements and overlay the fitted normal distribution:

$$f_Y(y) = \frac{1}{\hat{\sigma}\sqrt{2\pi}} e^{-\frac{(y-\hat{\mu})^2}{2\hat{\sigma}^2}}$$

This process is repeated for sample sizes $n_1 \in \{10, 100, 1000\}$.

Part (ii) - Repeated Simulation and Bias Analysis

Function: `repeat_simulation(n1, n2, true_mu=50, true_sigma=5)`

- **Step 1:** Repeat the single simulation $n_2 = 1000$ times for each n_1.
- **Step 2:** For each repetition, store the estimated mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$.
- **Step 3:** Plot histograms of all estimated $\hat{\mu}$ values and all $\hat{\sigma}$ values.
- **Step 4:** Overlay the true values $\mu = 50$ and $\sigma = 5$ as reference lines.
- **Step 5:** Compute and display the average of the estimated means and standard deviations:

$$\text{Mean of } \hat{\mu} = \frac{1}{n_2} \sum_{i=1}^{n_2} \hat{\mu}_i$$

$$\text{Mean of } \hat{\sigma} = \frac{1}{n_2} \sum_{i=1}^{n_2} \hat{\sigma}_i$$

Helper Function - Single Run

Function: `simulate_temperature_mle_single(n1, true_mu=50, true_sigma=5)`

This helper function performs one iteration of the temperature simulation and returns the MLE estimates of μ and σ based on the noisy measurements.

Key Observations

- Repeating the simulation helps assess the variability and potential bias in the MLE estimators due to sensor noise.
- The histogram of $\hat{\mu}$ is centered around the true mean if the estimator is unbiased.
- The histogram of $\hat{\sigma}$ can change due to the added noise, indicating a potential bias in the estimation of variance. The output obtained were as follows:-

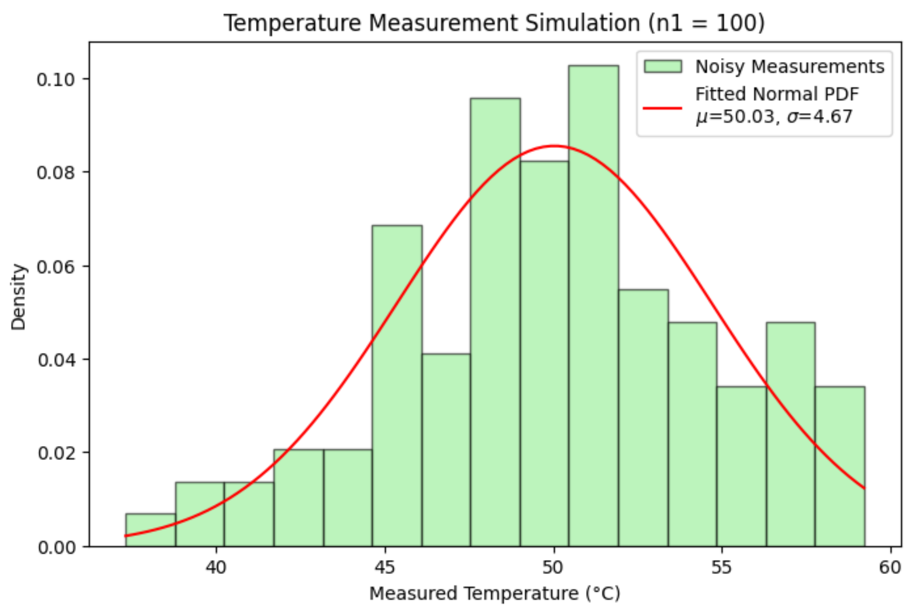


Figure 6:

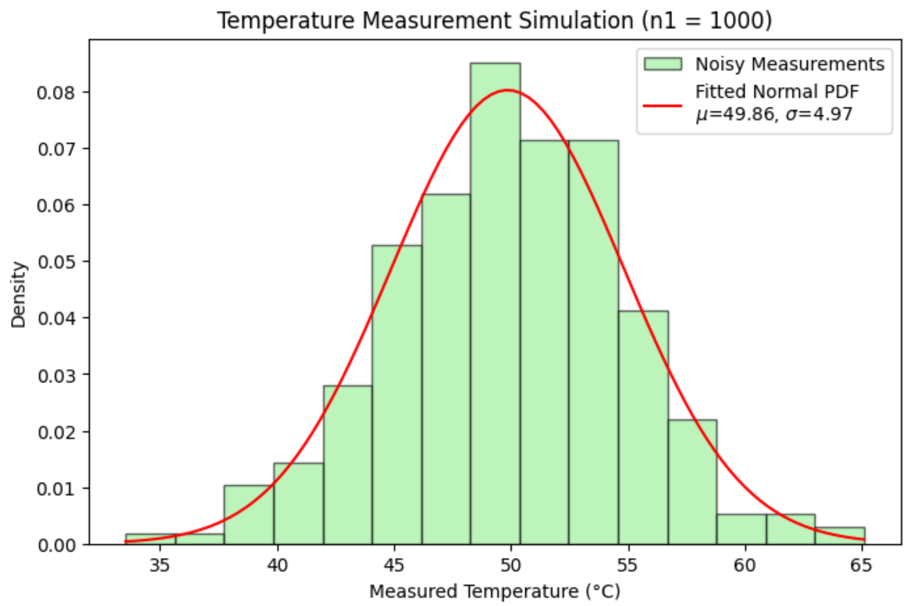


Figure 7:

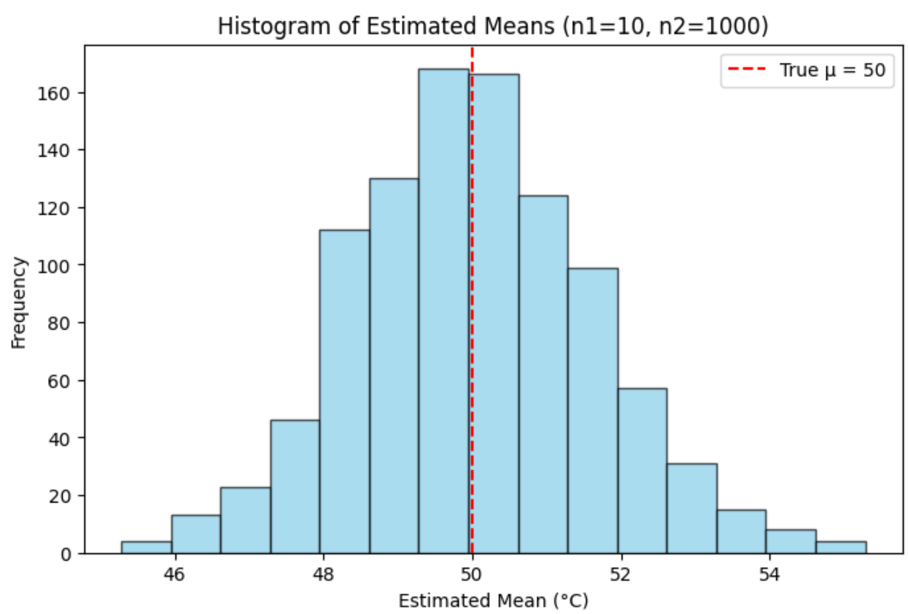


Figure 8:

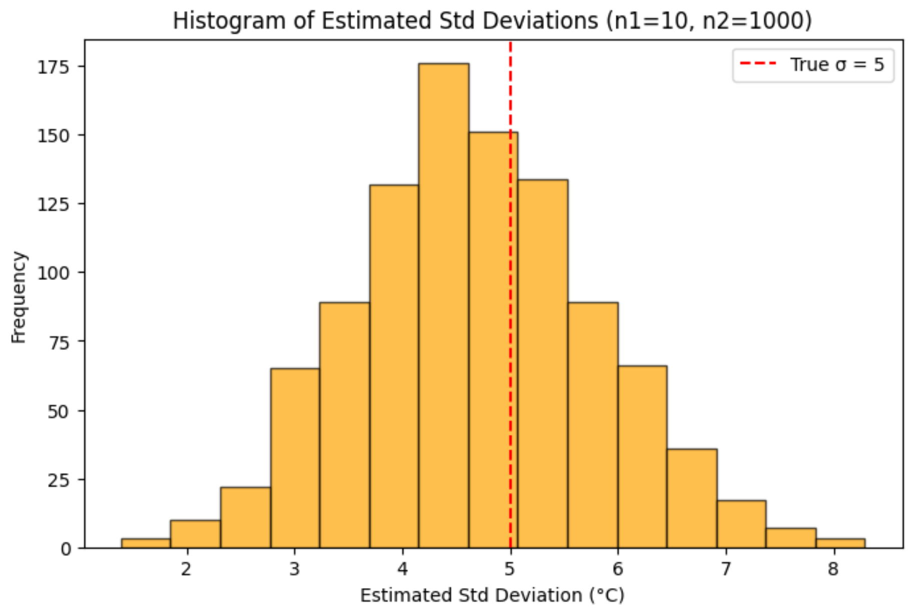


Figure 9:

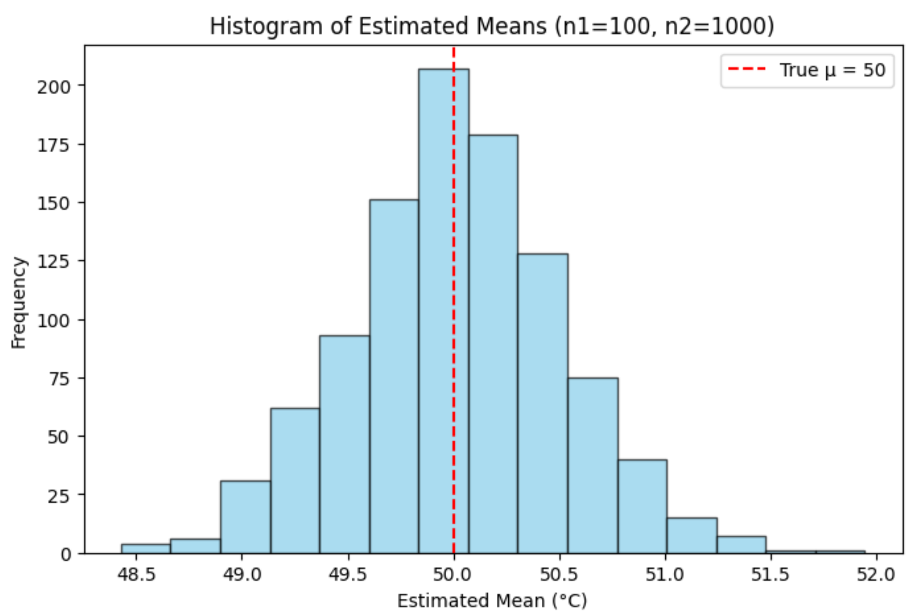


Figure 10:

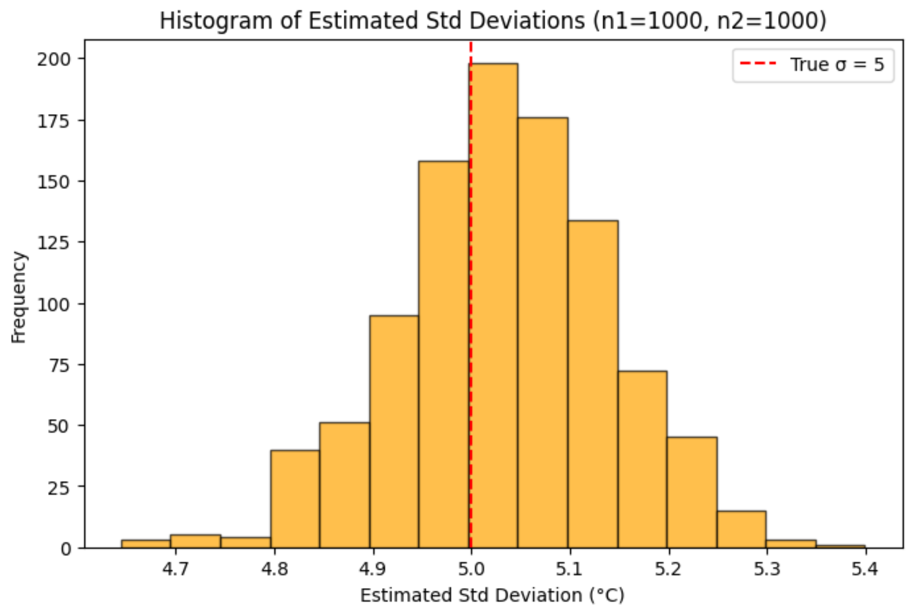


Figure 11:

5 Question 3

You are an analyst at a financial firm studying the daily returns of a high-risk stock. Unlike normal stocks, the returns of this stock follow a **t-distribution** with heavier tails, meaning extreme gains or losses are more likely. You want to estimate the mean (μ) and standard deviation (σ) of the stock returns.

(i) Simulate n_1 daily stock returns using a t-distribution with a mean (μ) of 0.1% and a standard deviation (σ) of 2%. Use **Maximum Likelihood Estimation (MLE)** to estimate the mean and standard deviation of the stock returns. Plot a histogram of the simulated returns and overlay the t-distribution Probability Density Function (PDF) using your estimated parameters.

(ii) Now, suppose the stock returns are affected by market noise (η), where η is uniformly distributed between -0.5% and 0.5%. Repeat the experiment with the noisy returns ($Y = X + \eta$) and compare your results with part (i).

How does the t-distribution and noise affect your estimates? Are the estimators biased? Discuss your observations.

6 Introduction

The questions asks us to do the same task we did in first part but with adding bias with help of uniform distribution.

Data

Data generate during program .

Methodology

The code used is as follows:-

```python
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

np.random.seed(42)  # For reproducibility

def simulate_t_distribution(n1, df, true_mu, true_sigma):
    """
    Simulate t-distributed stock returns and perform MLE
        estimation
    """
    t_samples = stats.t.rvs(df=df, size=n1)
    t_scaled = true_mu + true_sigma * t_samples / np.sqrt(df
         / (df - 2))

    mle_mu = np.mean(t_scaled)
    mle_sigma = np.std(t_scaled, ddof=0)

    # Plot histogram with t-distribution overlay
    plt.figure(figsize=(8, 5))
    plt.hist(t_scaled, bins=20, density=True, alpha=0.6,
        color='skyblue', edgecolor='black')

    x = np.linspace(min(t_scaled), max(t_scaled), 200)
    t_pdf = stats.t.pdf((x - mle_mu) / mle_sigma, df=df) /
        mle_sigma
    plt.plot(x, t_pdf, 'r-', label='Estimated t-PDF')

    plt.title(f"Simulated t-Distributed Returns (n1={n1})")
    plt.xlabel("Returns")
    plt.ylabel("Density")
    plt.legend()
    plt.show()

```

```
    return mle_mu, mle_sigma, t_scaled

def simulate_with_market_noise(t_scaled, df):
    """
    Add uniform market noise and recompute MLE
    """
    eta = np.random.uniform(-0.005, 0.005, size=len(t_scaled
        ))
    noisy_returns = t_scaled + eta

    noisy_mle_mu = np.mean(noisy_returns)
    noisy_mle_sigma = np.std(noisy_returns, ddof=0)

    # Plot histogram of noisy returns with t-distribution
        overlay
    plt.figure(figsize=(8, 5))
    plt.hist(noisy_returns, bins=20, density=True, alpha
        =0.6, color='orange', edgecolor='black')

    x = np.linspace(min(noisy_returns), max(noisy_returns),
        200)
    noisy_t_pdf = stats.t.pdf((x - noisy_mle_mu) /
        noisy_mle_sigma, df=df) / noisy_mle_sigma
    plt.plot(x, noisy_t_pdf, 'r-', label='Estimated t-PDF (
        Noisy)')

    plt.title(f"Noisy Stock Returns (n1={len(t_scaled)})")
    plt.xlabel("Noisy Returns")
    plt.ylabel("Density")
    plt.legend()
    plt.show()

    return noisy_mle_mu, noisy_mle_sigma

# Parameters
n1_list = [10, 100, 1000]
df = 8/3
true_mu = 0.001  # 0.1%
true_sigma = 0.02  # 2%

# Loop over different sample sizes
for n1 in n1_list:
    print(f"\n===== Sample Size n1 = {n1} =====")
    mle_mu, mle_sigma, t_scaled = simulate_t_distribution(n1
        , df, true_mu, true_sigma)
    print(f"Part (i) - MLE Mean: {mle_mu:.5f}, MLE Std Dev:
        {mle_sigma:.5f}")

    noisy_mle_mu, noisy_mle_sigma =
        simulate_with_market_noise(t_scaled, df)
```

```python
    print(f"Part (ii) - Noisy MLE Mean: {noisy_mle_mu:.5f},
        Noisy MLE Std Dev: {noisy_mle_sigma:.5f}")
```

Simulation of t-Distributed Stock Returns with Market Noise

Objective

The goal of this simulation is to generate stock returns modeled as a t-distribution and analyze the impact of uniform market noise on the Maximum Likelihood Estimates (MLE) of the mean and standard deviation.

Part (i) - Simulating t-Distributed Returns and MLE Estimation

The true stock returns R follow a scaled and shifted t-distribution:

$$R = \mu_{\text{true}} + \sigma_{\text{true}} \cdot \frac{T}{\sqrt{\frac{\nu}{\nu-2}}}$$

where:

- $T \sim t(\nu)$ is a t-distributed random variable with degrees of freedom $\nu = \frac{8}{3}$,
- $\mu_{\text{true}} = 0.001$ (mean return 0.1%),
- $\sigma_{\text{true}} = 0.02$ (standard deviation 2%).

MLE Estimation:

$$\hat{\mu} = \frac{1}{n_1} \sum_{i=1}^{n_1} R_i$$

$$\hat{\sigma} = \sqrt{\frac{1}{n_1} \sum_{i=1}^{n_1} (R_i - \hat{\mu})^2}$$

A histogram of the simulated returns is plotted with the t-distribution probability density function (PDF) overlayed using the estimated parameters.

Part (ii) - Adding Uniform Market Noise and Re-estimation

Uniform market noise $\eta \sim \mathcal{U}(-0.005, 0.005)$ is added to each simulated return:

$$R_{\text{noisy}} = R + \eta$$

MLE Re-estimation for noisy returns:

$$\hat{\mu}_{\text{noisy}} = \frac{1}{n_1} \sum_{i=1}^{n_1} R_{\text{noisy},i}$$

$$\hat{\sigma}_{\text{noisy}} = \sqrt{\frac{1}{n_1}\sum_{i=1}^{n_1}(R_{\text{noisy},i} - \hat{\mu}_{\text{noisy}})^2}$$

The noisy returns are plotted similarly, overlaying the estimated t-distribution PDF.

Key Observations

- As n_1 increases, the MLE estimates of mean and standard deviation become more accurate and stable.
- The addition of uniform market noise has a negligible effect on the mean estimator due to its zero mean.
- However, the standard deviation estimator is slightly biased upwards due to the added noise variance.

The output obtained is as follows:-

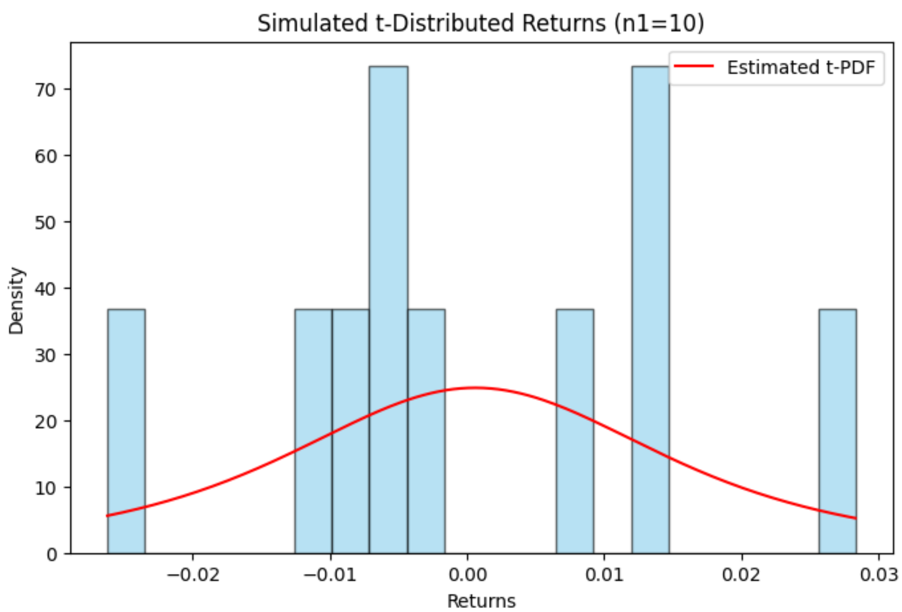


Figure 12:

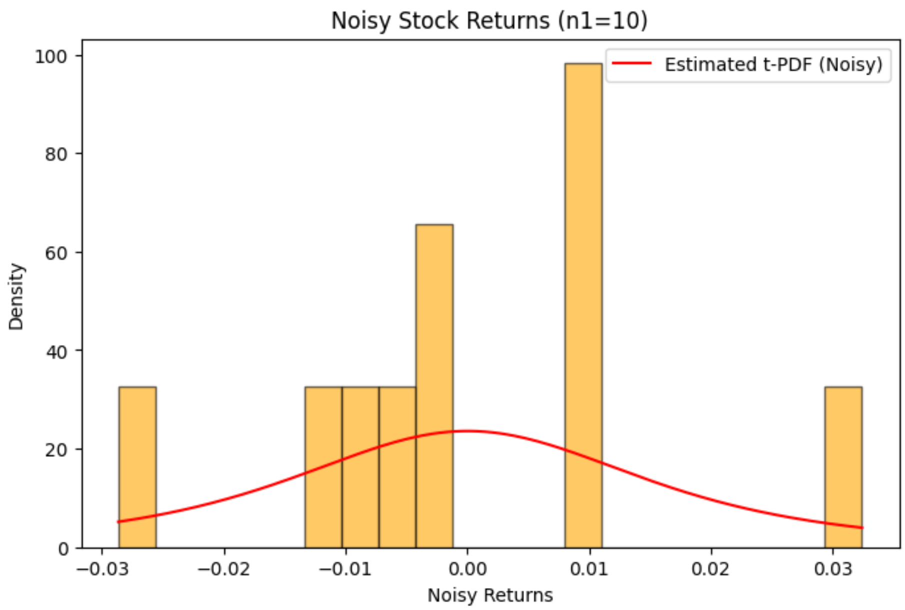


Figure 13:

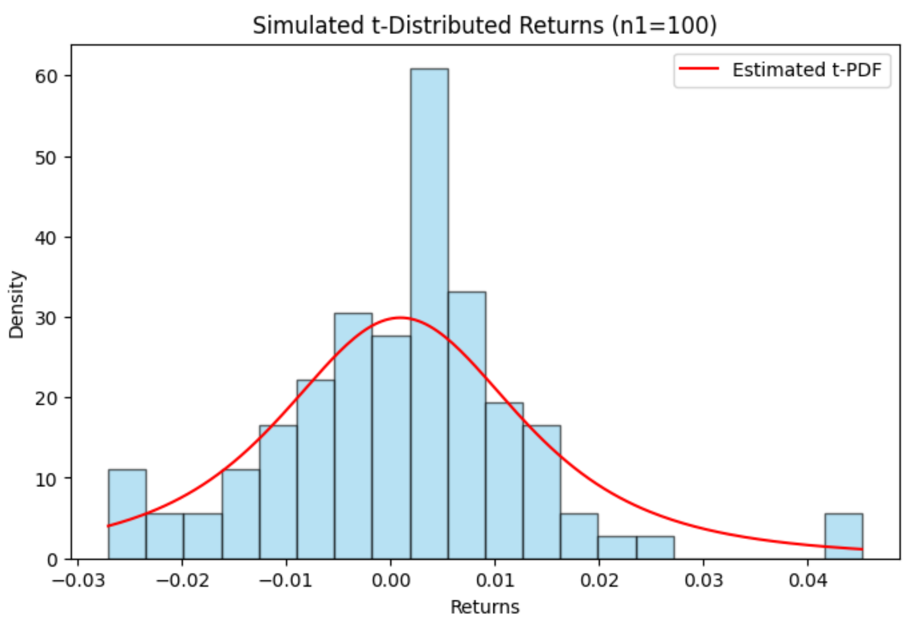


Figure 14:

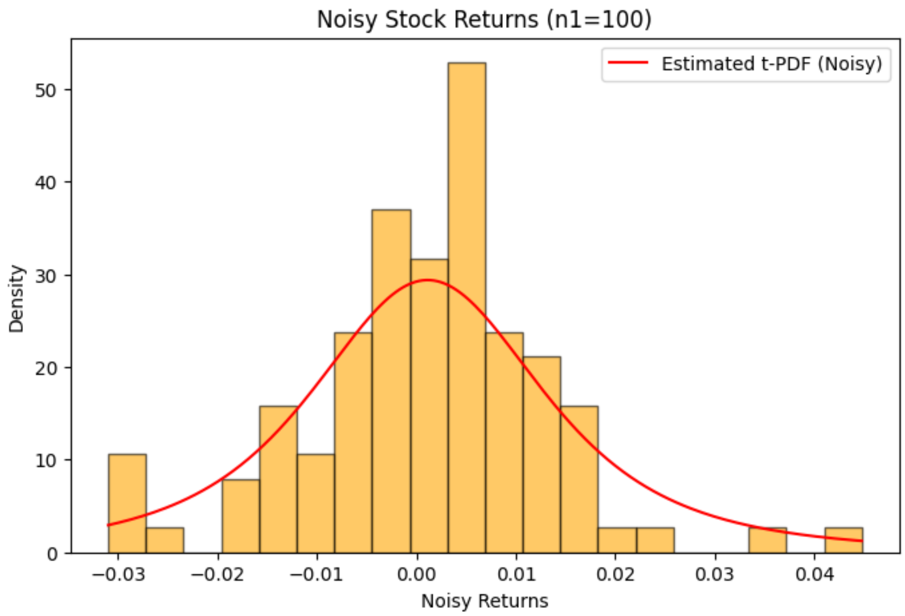


Figure 15:

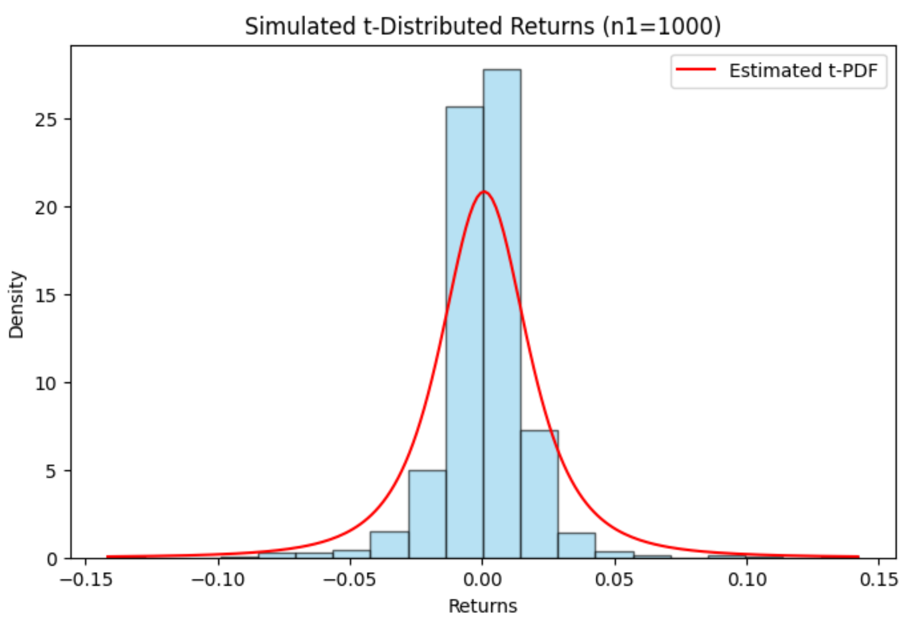
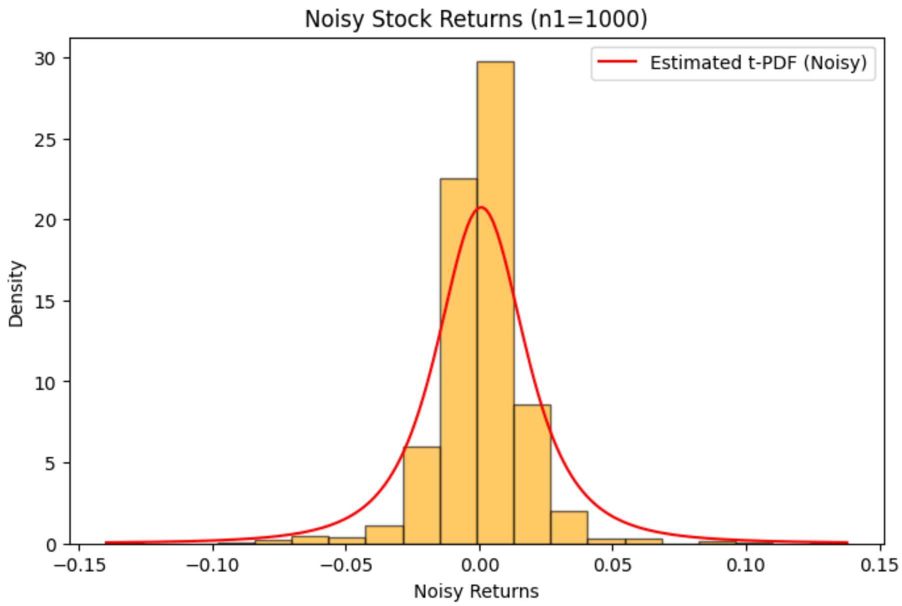


Figure 16:



Noisy Stock Returns (n1=1000)
Estimated t-PDF (Noisy)
Density
30
25
20
15
10
5
0
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
Noisy Returns

Figure 17: