Richard Pallangyo
DATA 557
Simulation Project 1:

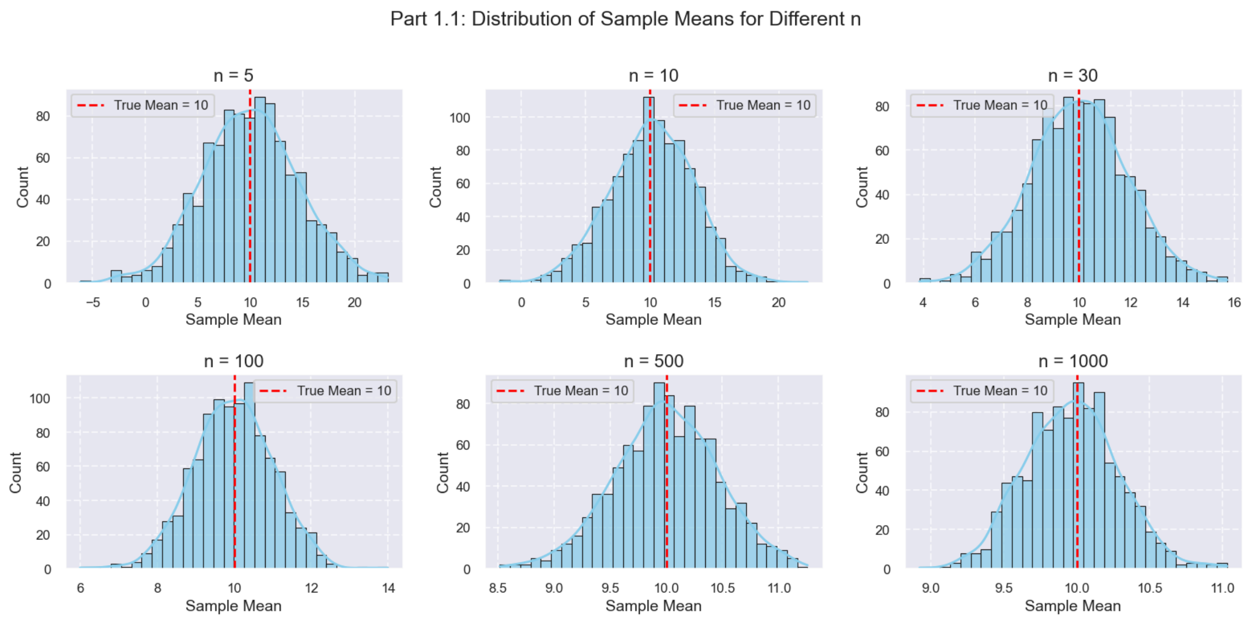
Robustness of One-Sample Estimation Procedures Under Varying Conditions

The goal of this simulation study is to investigate how estimation and hypothesis testing are impacted by sample size and the underlying population distribution. We will use a mean (mu) value of 10 and a standard deviation (sigma) value of 10. This study will comprise two parts.

1. Normal Population with varying n. We will investigate the behavior of one-sample estimation (mean, standard deviation, z-scores, t-scores) under a normal population for varying sample sizes, n (5, 10, 30, 100, 500, 1000).
2. Different Populations with varying n. We will explore the robustness of these estimation and testing procedures when sampling from non-normal distributions (Gamma, Exponential) compared to Normal for smaller sample sizes (10, 30, 500).

PART 1: Normal Population with Varying n

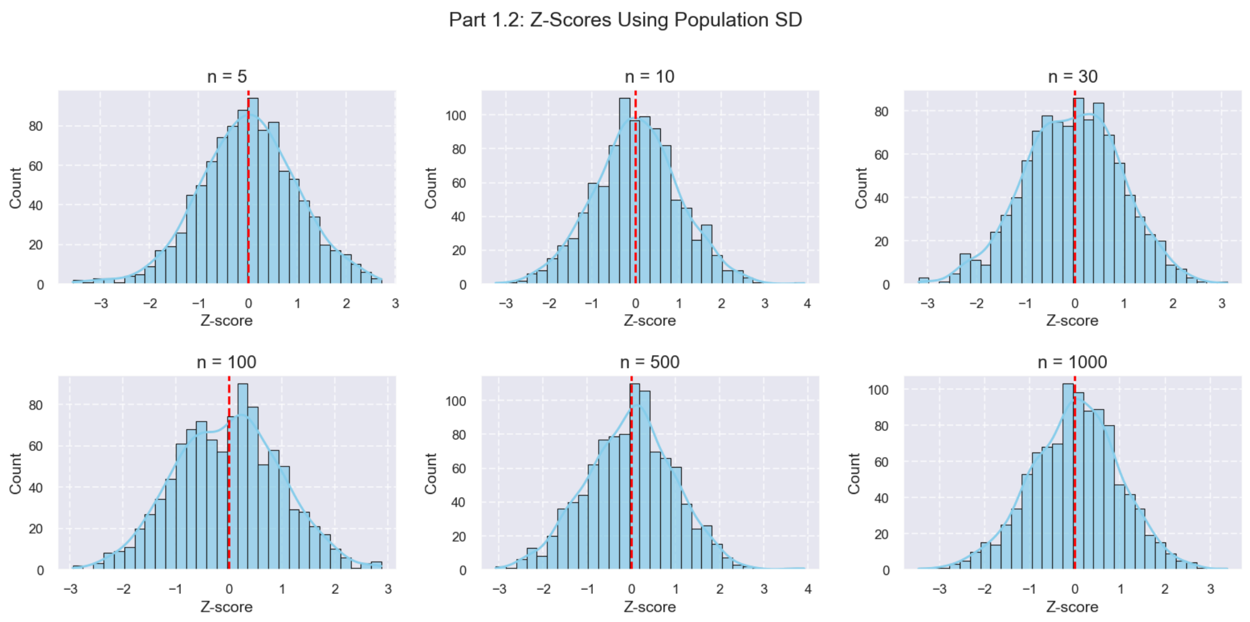
1. Distribution of Sample Means (mu = 10).



Observations:

- Center: As n increases, the sample means concentrate around \mu = 10.
- Spread: The spread of the sample mean distribution decreases with larger n, illustrating the reduction in standard error.
- Shape: Since the population is normal, as we would expect. The distribution of sample means is approximately normal for all n, but it becomes more sharply peaked around 10 as n grows.

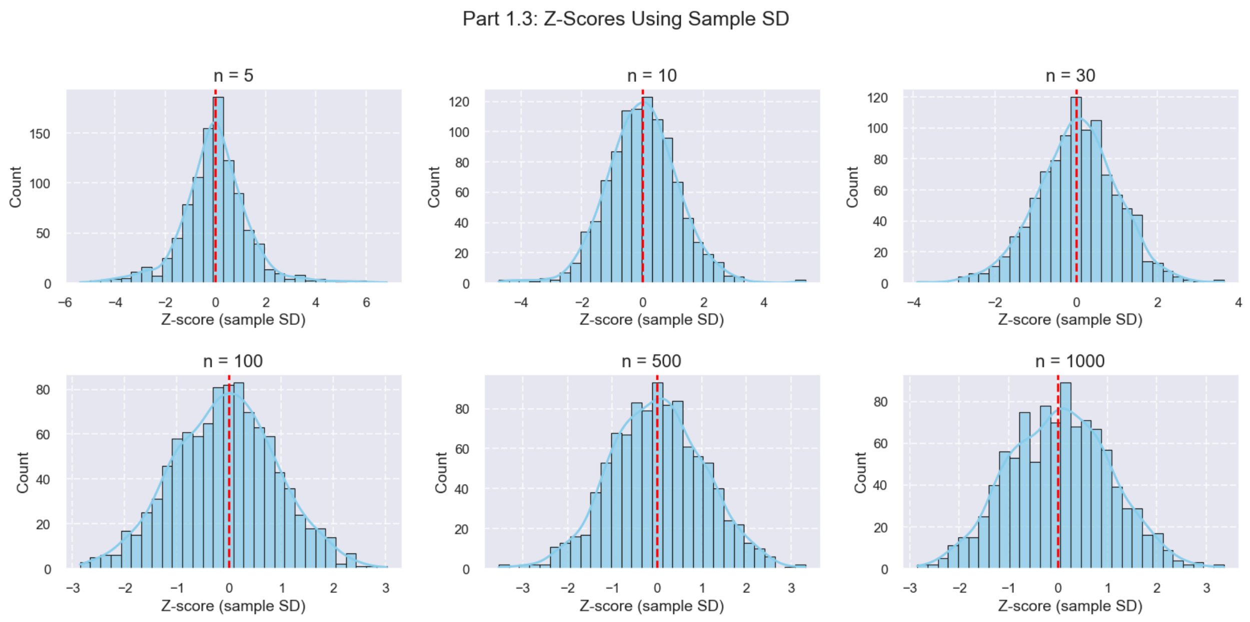
2. Z-scores Using the Population Standard Deviation (sigma = 10)



Observations:

- The standardized values are centered near 0 and approach a standard normal distribution.
- As sample size n increases, variability in these Z-scores decreases (reflecting smaller standard errors).

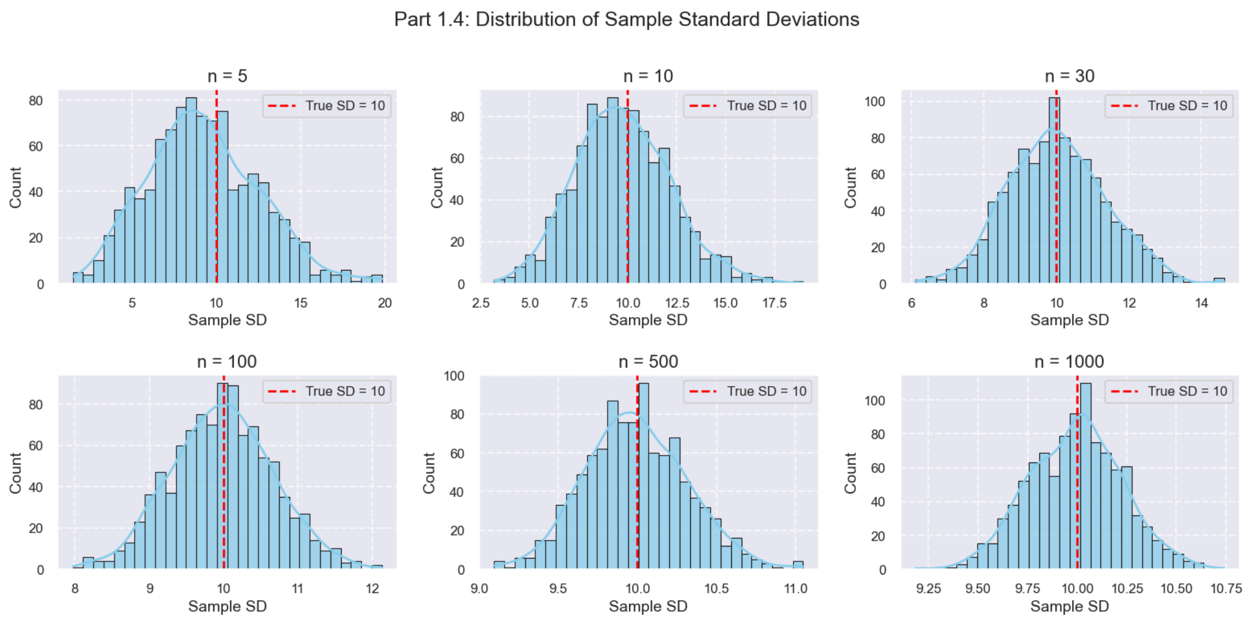
3. Z-scores Using the Sample Standard Deviation (s)



Observations:

- For large n, the distribution approximates the standard normal.
- For small n, the variability is more significant, and the distribution often appears slightly heavier-tailed.

4. Distribution of Sample Standard Deviations

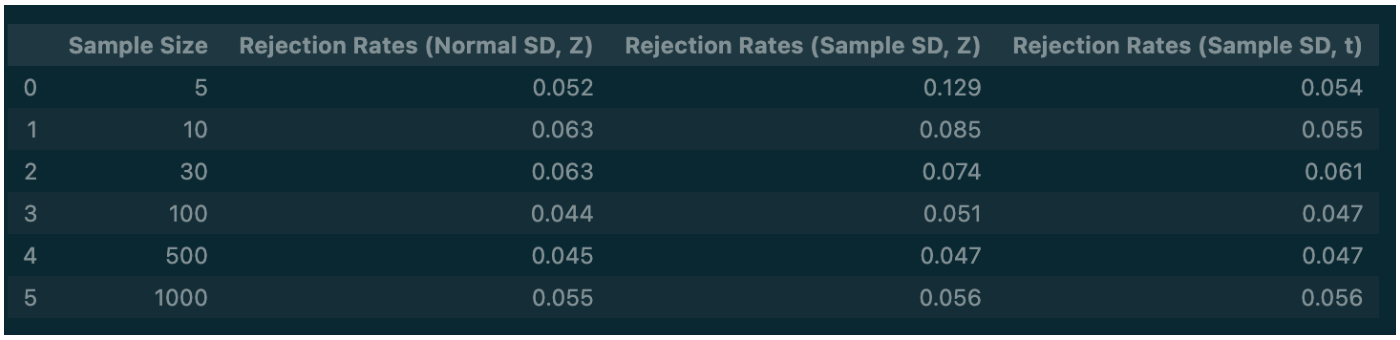


Observations:

- The sample standard deviations center around 10 but have a wider spread when n is small.
- As sample size n increases, the distribution of s narrows around sigma = 10.

5. Hypothesis Testing at α = 0.05

Rejection Rates (Proportion of 1000 samples that reject $H_0 = 10$, at α=0.05):



	Sample Size	Rejection Rates (Normal SD, Z)	Rejection Rates (Sample SD, Z)	Rejection Rates (Sample SD, t)
0	5	0.052	0.129	0.054
1	10	0.063	0.085	0.055
2	30	0.063	0.074	0.061
3	100	0.044	0.051	0.047
4	500	0.045	0.047	0.047
5	1000	0.055	0.056	0.056

Observations:

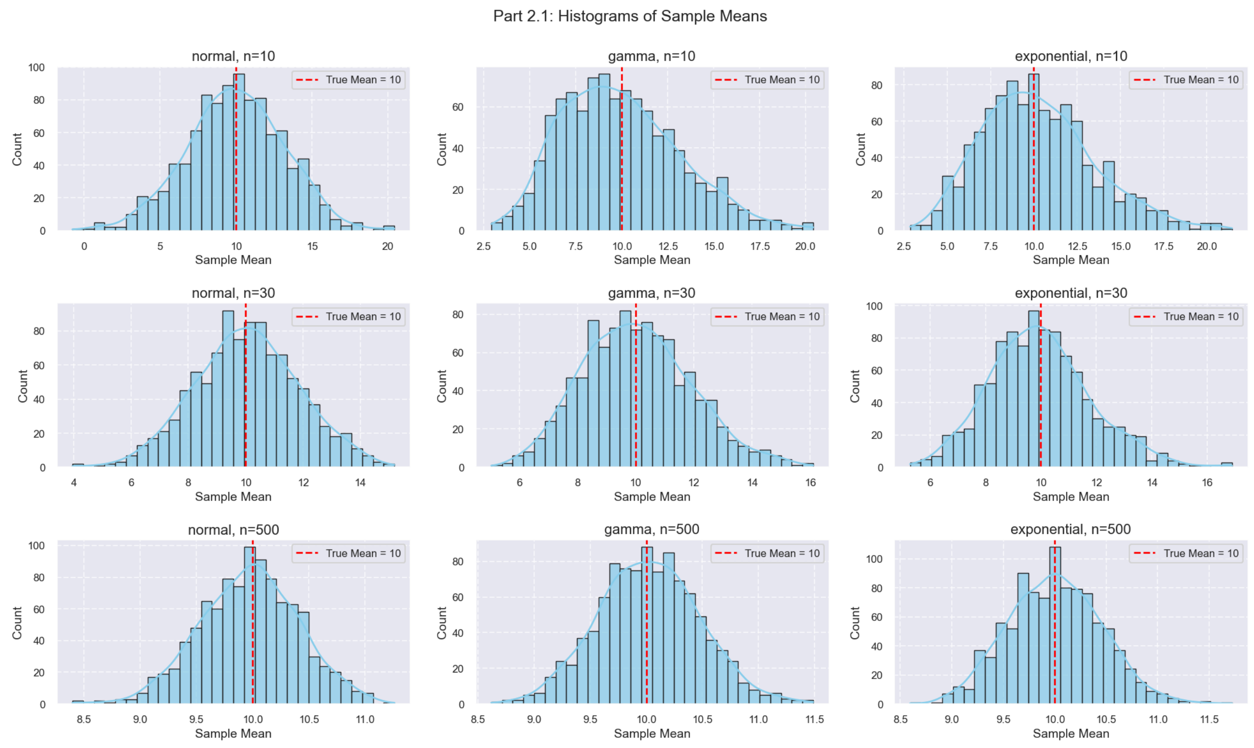
- The rejection rate is close to 5% regardless of the sample size
- However, if we use the Z critical method, the rejection rate is very high (inflated rates) for small sample sizes.
- If we use the t-critical method, the rejection rates are closer to 5%.
- As the sample size increases, the rejection rates approach 5%.

Part 1 Summary:

- Sample Means converge to the true mean with decreasing variance as n grows.
- Z-scores (with known \sigma) match the standard normal distribution well; with sample SD, small-sample variability is higher but improves with larger n.
- When performing Hypothesis Testing, the t-distribution with sample SD is more accurate for controlling Type I error even at smaller n.
- Hypothesis Testing using the standard critical value with sample SD can inflate or deflate rejection rates at small n.

Part 2: Different Populations with varying n.

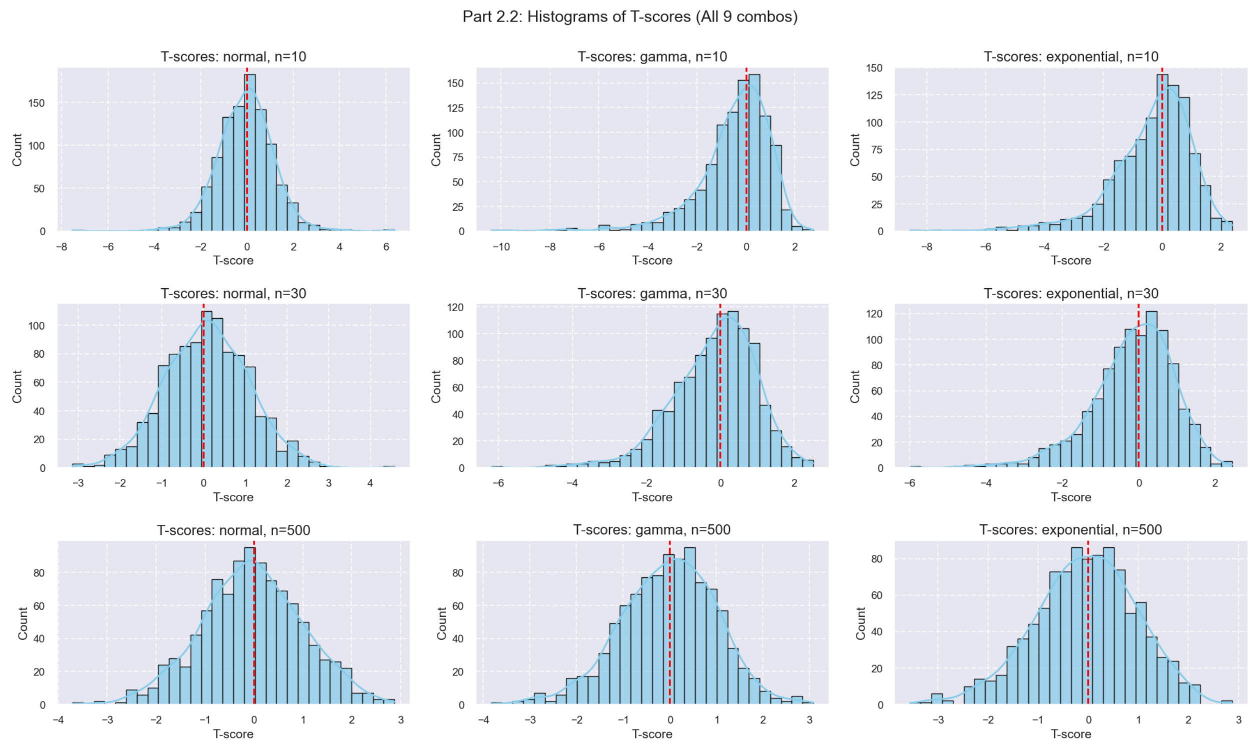
1. Distribution of Sample Means



Observations:

- Normal: Sample means are symmetric around 10 for all n.
- Gamma: Positive skew is more pronounced for small n but diminishes with larger n.
- Exponential: Even stronger skew at small n; the distribution of \overline{X} gradually looks more normal with increasing n; as discussed in the course, this asserts the Central Limit Theorem.

2. Z-scores (Using Sample SD) or T-scores

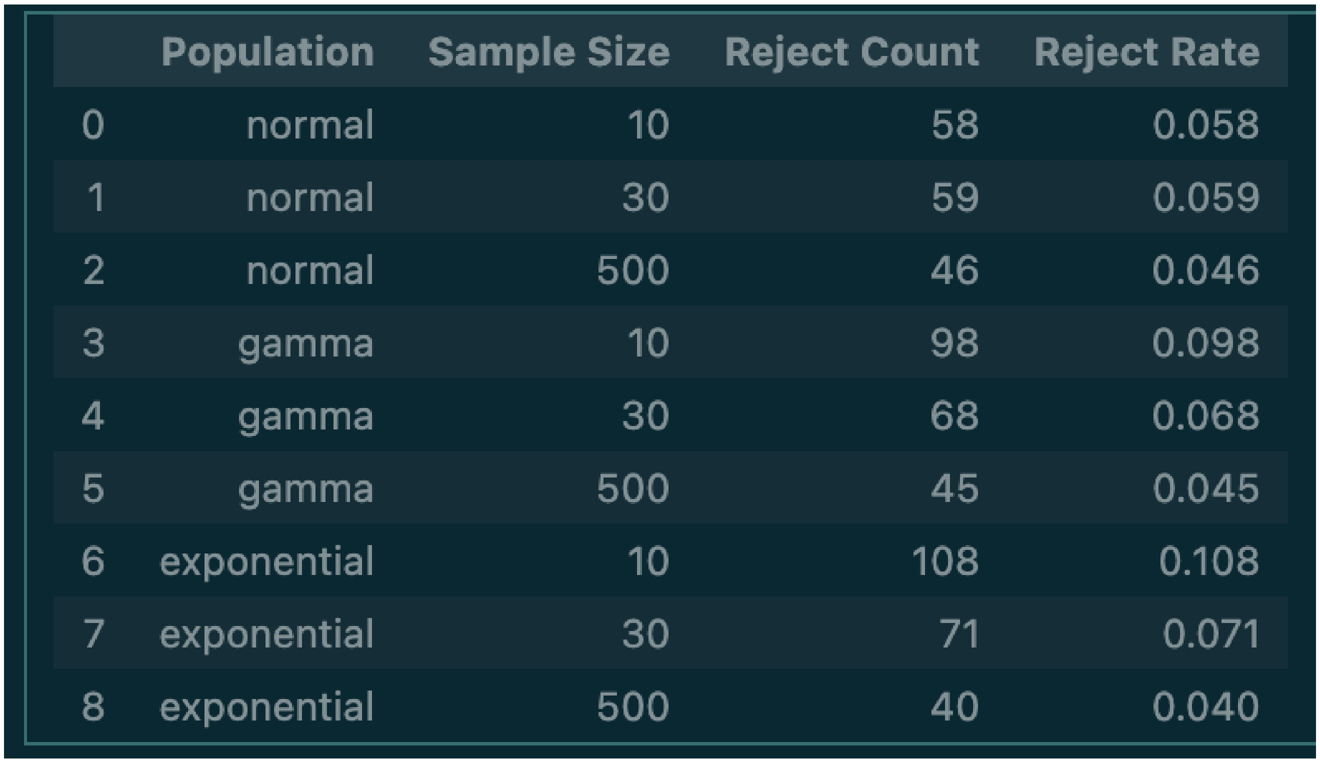


Observations:

- Normal population: T-scores approximate a t-distribution well and look close to standard normal for moderate/large n.
- Gamma and exponential populations: For small n, the shape can deviate from the expected distribution (heavier tails/skew), but as n increases, the distribution of T-scores becomes more symmetric and closer to normal/t-distribution due to the Central Limit Theorem.

3. Hypothesis Testing via One-Sample T-Test

Rejection Rates (Proportion of the 1000 samples that reject $H_0 = 10$, at α=0.05)



	Population	Sample Size	Reject Count	Reject Rate
0	normal	10	58	0.058
1	normal	30	59	0.059
2	normal	500	46	0.046
3	gamma	10	98	0.098
4	gamma	30	68	0.068
5	gamma	500	45	0.045
6	exponential	10	108	0.108
7	exponential	30	71	0.071
8	exponential	500	40	0.040

.

Observations:

- For the normal population, the rejection rate is approx 5% at all sample sizes.
- The rejection rate for gamma and exponential distributions deviates from 5% for smaller sample sizes. As the plots above show, this is due to skewness or heavy tails. For larger n, the rate moves closer to 5%, demonstrating the Central Limit Theorem.

Part 2 Summary:

- Non-Normal Distributions: Small sample size estimation and inference can be biased or show excessive variability for gamma and exponential distributions. As we observed in the plots, they tend to have more significant skewness and heavier tails.
- The adherence to the Central Limit Theorem (CLT): As n increases significantly, for instance, when n=500, the sample means tend toward normality regardless of the shape of the population, and t-tests maintain their Type I error rate closer to the nominal level.

Key Takeaways from the Simulation Study:

1. **Sample Size Effects**: Larger sample sizes decrease the variability of the sample mean and result in more accurate estimates of standard deviation. This improves the reliability of Z and T statistics and ensures they align more closely with their expected distributions.

2. **Distribution Shape**: Both Z and T tests perform well with normally distributed data. However, the Type I error rate can be misleading with skewed distributions, such as Gamma or Exponential, particularly in small sample sizes. The t-test usually remains effective, depending on the extent of skewness in the data and the sample size.
3. **Testing Methods**: When the population standard deviation is known, tests effectively reject the null hypothesis about 5% of the time under the null hypothesis. Employing the sample standard deviation with Z tests can slightly increase false rejections in small samples. On the other hand, using the t-distribution with n-1 degrees of freedom helps keep the rejection rate closer to the expected 5%, making the t-test a more suitable option when the standard deviation is unknown.

Appendix: Code

Below is the Python script for running the simulation. To access the complete code along with the notebook, please check out this GitHub repository: **Simulation Project.**

```python
# Importing the libraries

import pandas as pd

import numpy as np

import scipy.stats as stats

import matplotlib.pyplot as plt

import seaborn as sns


# Set up the plotting configuration and styles

sns.set_theme(

   style="darkgrid",

   palette="pastel",

   context="talk",

   font_scale=1.2,

   rc={

       "figure.figsize": (12, 8),

       "axes.labelsize": 14,

       "axes.titlesize": 16,

       "legend.fontsize": 12,

       "xtick.labelsize": 12,

       "ytick.labelsize": 12,

       "lines.linewidth": 2,

       "grid.linestyle": "--",

       "grid.alpha": 0.7

   }

)


# Setup seed for reproducibility

np.random.seed(5571)
```

```python
# Define sample sizes
sample_sizes_part1 = [5, 10, 30, 100, 500, 1000]
sample_sizes_part2 = [10, 30, 500]

# Population parameters
mu = 10
sigma = 10

# Number of simulations
simulations = 1000

# Function to simulate population data
def simulation_population_data(population_type, sample_size):
    """
    Generate a random sample from the specified population type.
    population_type can be 'normal', 'gamma', or 'exponential'.
    """
    if population_type == 'normal':
        return np.random.normal(mu, sigma, sample_size)
    elif population_type == 'gamma':
        # For a Gamma distribution with mean = mu and sd = sigma,
        # shape = (mu^2 / sigma^2), scale = (sigma^2 / mu)
        shape = (mu**2) / (sigma**2)
        scale = (sigma**2) / mu
        return np.random.gamma(shape, scale, sample_size)
    else:  # 'exponential'
        # Exponential with mean = mu => lambda = 1/mu
        return np.random.exponential(mu, sample_size)

# PART 1: Normal Population with Varying n
# 1. Distribution of Sample Means
```

```
fig, axs = plt.subplots(2, 3, figsize=(16, 8))  # 2 rows, 3 columns

axs = axs.flatten()  # Flatten to easily index subplots


for i, n in enumerate(sample_sizes_part1):

   sample_means = []

   for _ in range(simulations):

       sample_data = simulation_population_data('normal', n)

       sample_means.append(np.mean(sample_data))


   # Plot histogram of sample means

   sns.histplot(sample_means, bins=30, kde=True, color='skyblue', edgecolor='k',

                alpha=0.7, ax=axs[i])

   axs[i].axvline(mu, color='red', linestyle='--', label='True Mean = 10')

   axs[i].set_title(f'n = {n}')

   axs[i].set_xlabel('Sample Mean')

   axs[i].set_ylabel('Count')

   axs[i].legend()


fig.suptitle('Part 1.1: Distribution of Sample Means for Different n', fontsize=18)

plt.tight_layout()

plt.show()


# 2. Z-scores Using the Population Standard Deviation (sigma = 10)


# %%

fig, axs = plt.subplots(2, 3, figsize=(16, 8))

axs = axs.flatten()


for i, n in enumerate(sample_sizes_part1):

   z_vals = []

   for _ in range(simulations):

       sample_data = simulation_population_data('normal', n)
```

```python
        x_bar = np.mean(sample_data)
        z = (x_bar - mu) / (sigma / np.sqrt(n))
        z_vals.append(z)

    sns.histplot(z_vals, bins=30, kde=True, color='skyblue', edgecolor='k',
                 alpha=0.7, ax=axs[i])
    axs[i].axvline(0, color='red', linestyle='--')
    axs[i].set_title(f'n = {n}')
    axs[i].set_xlabel('Z-score')
    axs[i].set_ylabel('Count')


fig.suptitle('Part 1.2: Z-Scores Using Population SD', fontsize=18)
plt.tight_layout()
plt.show()

# 3. Z-scores Using the Sample Standard Deviation (s)
# %%
fig, axs = plt.subplots(2, 3, figsize=(16, 8))
axs = axs.flatten()

for i, n in enumerate(sample_sizes_part1):
    z_sample_vals = []
    for _ in range(simulations):
        sample_data = simulation_population_data('normal', n)
        x_bar = np.mean(sample_data)
        s = np.std(sample_data, ddof=1)
        z_samp = (x_bar - mu) / (s / np.sqrt(n))
        z_sample_vals.append(z_samp)

    sns.histplot(z_sample_vals, bins=30, kde=True, color='skyblue', edgecolor='k',
                 alpha=0.7, ax=axs[i])
    axs[i].axvline(0, color='red', linestyle='--')

    axs[i].set_title(f'n = {n}')

    axs[i].set_xlabel('Z-score (sample SD)')

    axs[i].set_ylabel('Count')



fig.suptitle('Part 1.3: Z-Scores Using Sample SD', fontsize=18)

plt.tight_layout()

plt.show()


# %% [markdown]

# Observations:

# - For large n, the distribution approximates the standard normal.

# - For small n, the variability is more significant, and the distribution often
appears slightly heavier-tailed.


# %% [markdown]

# 4. Distribution of Sample Standard Deviations


# %%

fig, axs = plt.subplots(2, 3, figsize=(16, 8))

axs = axs.flatten()


for i, n in enumerate(sample_sizes_part1):

    s_vals = []

    for _ in range(simulations):

        sample_data = simulation_population_data('normal', n)

        s = np.std(sample_data, ddof=1)

        s_vals.append(s)


    sns.histplot(s_vals, bins=30, kde=True, color='skyblue', edgecolor='k',

                 alpha=0.7, ax=axs[i])
```

```python
    axs[i].axvline(sigma, color='red', linestyle='--', label='True SD = 10')
    axs[i].set_title(f'n = {n}')
    axs[i].set_xlabel('Sample SD')
    axs[i].set_ylabel('Count')
    axs[i].legend()

fig.suptitle('Part 1.4: Distribution of Sample Standard Deviations', fontsize=18)
plt.tight_layout()
plt.show()

# %% [markdown]
# 5. Hypothesis Testing at α = 0.05
# - Rejection Rates (Proportion of 1000 samples that reject H0 = 10, at α=0.05):

# %%
results_part1 = {
    'sample_size': [],
    'rejections_normal_sd_z': [],
    'rejections_sample_sd_z': [],
    'rejections_sample_sd_t': []
}

for n in sample_sizes_part1:
    rej_pop_z, rej_samp_z, rej_t = 0, 0, 0
    for _ in range(simulations):
        sample_data = simulation_population_data('normal', n)
        x_bar = np.mean(sample_data)
        s = np.std(sample_data, ddof=1)

        # (1) Z-test with known SD = 10
        z_pop = (x_bar - mu) / (sigma / np.sqrt(n))
        if abs(z_pop) > 1.96:
            rej_pop_z += 1

        # (2) Z-test with sample SD
        z_samp = (x_bar - mu) / (s / np.sqrt(n))
        if abs(z_samp) > 1.96:
            rej_samp_z += 1

        # (3) t-test with sample SD
        t_stat = z_samp  # same formula, different critical value
        t_crit = stats.t.ppf(0.975, n - 1)
        if abs(t_stat) > abs(t_crit):
            rej_t += 1

    results_part1['sample_size'].append(n)
    results_part1['rejections_normal_sd_z'].append(rej_pop_z)
    results_part1['rejections_sample_sd_z'].append(rej_samp_z)
    results_part1['rejections_sample_sd_t'].append(rej_t)

df_part1 = pd.DataFrame(results_part1)
df_part1['rejections_normal_sd_z'] = df_part1['rejections_normal_sd_z'] / simulations
df_part1['rejections_sample_sd_z'] = df_part1['rejections_sample_sd_z'] / simulations
df_part1['rejections_sample_sd_t'] = df_part1['rejections_sample_sd_t'] / simulations

df_part1.rename(columns={
    'sample_size': 'Sample Size',
    'rejections_normal_sd_z': 'Rejection Rates (Normal SD, Z)',
    'rejections_sample_sd_z': 'Rejection Rates (Sample SD, Z)',
    'rejections_sample_sd_t': 'Rejection Rates (Sample SD, t)'
}, inplace=True)

print("Rejection Rates (Proportion of 1000 samples that reject H0 at α=0.05):")
df_part1
```

```
# Part 2: Different Populations with varying n.
populations = ['normal', 'gamma', 'exponential']
# 1. Distribution of Sample Means
# Plot all histograms of sample means for different populations
fig, axs = plt.subplots(3, 3, figsize=(20, 12))
axs = axs.flatten()

i = 0
for n in sample_sizes_part2:
   for pop in populations:
       sample_means = []
       for _ in range(simulations):
           data_pop = simulation_population_data(pop, n)
           sample_means.append(np.mean(data_pop))

       sns.histplot(sample_means, bins=30, kde=True, color='skyblue', edgecolor='k',
                alpha=0.7, ax=axs[i])
       axs[i].axvline(mu, color='red', linestyle='--', label='True Mean = 10')
       axs[i].set_title(f'{pop}, n={n}')
       axs[i].set_xlabel('Sample Mean')
       axs[i].set_ylabel('Count')
       axs[i].legend()
       i += 1

fig.suptitle("Part 2.1: Histograms of Sample Means", fontsize=18)
plt.tight_layout()
plt.show()

# 2. Z-scores (Using Sample SD) or T-scores
# Plot all histograms of sample means for different populations
```

```python
fig, axs = plt.subplots(3, 3, figsize=(20, 12))
axs = axs.flatten()

i = 0
for n in sample_sizes_part2:
   for pop in populations:
       t_vals = []
       for _ in range(simulations):
           data_pop = simulation_population_data(pop, n)
           x_bar = np.mean(data_pop)
           s = np.std(data_pop, ddof=1)
           t_score = (x_bar - mu) / (s / np.sqrt(n))
           t_vals.append(t_score)

       sns.histplot(t_vals, bins=30, kde=True, color='skyblue', edgecolor='k',
                    alpha=0.7, ax=axs[i])
       axs[i].axvline(0, color='red', linestyle='--')
       axs[i].set_title(f'T-scores: {pop}, n={n}')
       axs[i].set_xlabel('T-score')
       axs[i].set_ylabel('Count')
       i += 1

fig.suptitle("Part 2.2: Histograms of T-scores (All 9 combos)", fontsize=18)
plt.tight_layout()
plt.show()

# 3. Hypothesis Testing via One-Sample T-Test
results_part2 = {
   'Population': [],
   'Sample Size': [],
   'Reject Count': [],
   'Reject Rate': []
}


for pop in populations:
   for n in sample_sizes_part2:
       reject_count = 0
       for _ in range(simulations):
           data_pop = simulation_population_data(pop, n)
           x_bar = np.mean(data_pop)
           s = np.std(data_pop, ddof=1)
           t_stat = (x_bar - mu) / (s / np.sqrt(n))
           t_crit = stats.t.ppf(0.975, n - 1)
           if abs(t_stat) > t_crit:
               reject_count += 1
       results_part2['Population'].append(pop)
       results_part2['Sample Size'].append(n)
       results_part2['Reject Count'].append(reject_count)
       results_part2['Reject Rate'].append(reject_count / simulations)


df_part2 = pd.DataFrame(results_part2)
print("Part 2.3: Rejection Rates for One-Sample t-tests")
df_part2
```