Statistics Project 2

Gullapalli Madhava Asrith Murthy AI23BTECH11007
Panshul Jindal AI23BTECH11018
Sreenadan Babu AI23BTECH11010

April 13, 2025

Contents

Chapter 1

Introduction

This document outlines the statistical procedures followed by us in this project:

1. Parameter estimation for the Gamma distribution using Method of Moments (MoM) and Maximum Likelihood Estimation (MLE)
2. Construction of 95% confidence intervals for the variance of a normal distribution when both mean and variance are unknown
3. Construction of 95% confidence intervals for the difference in means of two independent normal distributions with unknown mean and variance
4. Hypothesis testing for Bernoulli distributed data

The project code is availaible on `https://github.com/madhava-asrith/Statistics-Project-2`.

Chapter 2

Gamma Distribution Parameter Estimation

2.1 The Gamma Distribution

The Gamma distribution is a two-parameter continuous probability distribution with probability density function (PDF):

$$f(x;\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \quad x > 0, \alpha > 0, \beta > 0 \tag{2.1}$$

where α is the shape parameter, β is the rate parameter, and $\Gamma(\alpha)$ is the gamma function defined as:

$$\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t} dt \tag{2.2}$$

Alternative parameterization using scale parameter $\theta = 1/\beta$:

$$f(x;\alpha,\theta) = \frac{1}{\Gamma(\alpha)\theta^\alpha} x^{\alpha-1} e^{-x/\theta}, \quad x > 0, \alpha > 0, \theta > 0 \tag{2.3}$$

In this project, we work with the Shape parameter α and the Scale parameter θ.

Key properties of the Gamma distribution:

$$\text{Mean} = \frac{\alpha}{\beta} = \alpha\theta \tag{2.4}$$

$$\text{Variance} = \frac{\alpha}{\beta^2} = \alpha\theta^2 \tag{2.5}$$

2.1.1 Method of Moments (MoM) Estimation

Given a random sample $\{x_1, x_2, \ldots, x_n\}$ from a Gamma distribution, the Method of Moments estimates the parameters by equating sample moments with population moments.

Algorithm 1 Method of Moments Estimation for Gamma Parameters

1: Calculate the sample mean: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
2: Calculate the sample variance: $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$
3: Set up the system of equations:

$$\bar{x} = \frac{\alpha}{\beta} = \alpha\theta \tag{2.6}$$

$$s^2 = \frac{\alpha}{\beta^2} = \alpha\theta^2 \tag{2.7}$$

4: Solve for α and β (or θ):

$$\hat{\alpha}_{MoM} = \frac{\bar{x}^2}{s^2} \tag{2.8}$$

$$\hat{\beta}_{MoM} = \frac{\bar{x}}{s^2} \tag{2.9}$$

$$\hat{\theta}_{MoM} = \frac{s^2}{\bar{x}} \tag{2.10}$$

2.1.2 Maximum Likelihood Estimation (MLE)

For a random sample $\{x_1, x_2, \ldots, x_n\}$ from a Gamma distribution, the likelihood function is:

$$L(\alpha, \beta) = \prod_{i=1}^{n} f(x_i; \alpha, \beta) = \prod_{i=1}^{n} \frac{\beta^\alpha}{\Gamma(\alpha)} x_i^{\alpha-1} e^{-\beta x_i} \tag{2.11}$$

Taking the natural logarithm yields the log-likelihood function:

$$\ell(\alpha, \beta) = n\alpha\ln(\beta) - n\ln(\Gamma(\alpha)) + (\alpha - 1)\sum_{i=1}^{n}\ln(x_i) - \beta\sum_{i=1}^{n} x_i \tag{2.12}$$

Algorithm 2 Gamma Parameter Estimation via MLE

1: Initialize with MoM estimates:

$$\hat{\alpha}_0 = \frac{\bar{x}^2}{s^2}$$
$$\hat{\beta}_0 = \frac{\bar{x}}{s^2}$$

2: Define log-likelihood function:

$$\ell(\alpha, \beta) = n\alpha\ln\beta - n\ln\Gamma(\alpha) + (\alpha - 1)\sum\ln x_i - \beta\sum x_i$$

3: Solve numerically using:

$$\psi(\alpha) = \ln\left(\frac{\alpha}{\bar{x}}\right) + \overline{\ln x} - \ln\bar{x}$$

where ψ is the digamma function

4: Iterate until convergence using Newton-Raphson:

$$\alpha_{k+1} = \alpha_k - \frac{\ln\alpha_k - \psi(\alpha_k) - \ln\bar{x} + \overline{\ln x}}{\frac{1}{\alpha_k} - \psi'(\alpha_k)}$$

5: Final estimates:

$$\hat{\beta}_{\text{MLE}} = \frac{\hat{\alpha}_{\text{MLE}}}{\bar{x}}$$

Since there is no closed-form expression for ML Estimate of α, we initialize it to some value and continuously update it iteratively, till it converges.

2.2 Implementation

2.2.1 Dataset

We use the *Annual Rainfall* from the Rainfall in India dataset obtained from Kaggle, which includes annual rainfall data for various regions over the last 125 years.

2.2.2 Objective

Our goal is to use the Annual rainfall data, model it as Gamma Distribution and estimate its parameters using the Method of Moments and Maximum Likelihood.

2.2.3 Code Implementation

2.2.3.1 Method of Moments

```
def gamma_moments(data):
    sample = data.sample(1000)
    mean = np.mean(sample)
    variance = np.var(sample, ddof=1)
    alpha = mean ** 2 / variance
    beta = mean / alpha
    return alpha, beta

# calculate the parameters using method of moments
annual_shape_moments, annual_scale_moments = gamma_moments(annual_rainfall)
print(f"Method of Moments - Shape: {annual_shape_moments}, Scale: {
    annual_scale_moments}")
```

Output:

```
Method of Moments - Shape: 2.455735431715778, Scale: 574.8269059316884
```

2.2.3.2 Maximum Likelihood

```
sample = annual_rainfall.sample(1000)

n = len(sample)
mean = np.mean(sample)
log_mean = np.mean(np.log(sample))

sample_var = np.var(sample, ddof=1)
alpha_init = mean ** 2 / sample_var

def newton_raphson_alpha(mean, log_mean, alpha_init, tol=1e-6, max_iter=100):
    alpha = alpha_init
    for _ in range(max_iter):
        f = np.log(alpha) - digamma(alpha) - np.log(mean) + log_mean
        f_prime = 1/alpha - polygamma(1, alpha)
        alpha_new = alpha - f / f_prime
        if abs(alpha_new - alpha) < tol:
            break
        alpha = alpha_new
    return alpha
alpha = newton_raphson_alpha(mean, log_mean, alpha_init)
beta = mean / alpha

print(f"Estimated parameters for annual rainfall: shape={alpha}, scale= {beta}"
    )
```

Output:

```
Estimated parameters for annual rainfall: shape = 2.9047511379638, scale = 523.4061492398275
```

2.2.3.3 Plotting the Gamma model using Statsmodels library

We use the previously estimated Shape parameter(α_{ML}) and Scale parameter(θ_{ML}) to plot the estimated Gamma distribution.

```
annual_rainfall = df['ANNUAL']
annual_rainfall = annual_rainfall[annual_rainfall > 0]  # remove zero values

# plot the histogram of the data
plt.hist(annual_rainfall, bins=30, density=True, alpha=0.6, color='g')

# plot the fitted gamma distribution
x = np.linspace(0, annual_rainfall.max(), 100)
pdf = stats.gamma.pdf(x, alpha, loc=loc, scale=beta)
plt.plot(x, pdf, 'r-', lw=2)
plt.title('Gamma Distribution Fit to Annual Rainfall')
plt.xlabel('Rainfall (mm)')
plt.ylabel('Density')
plt.show()
```

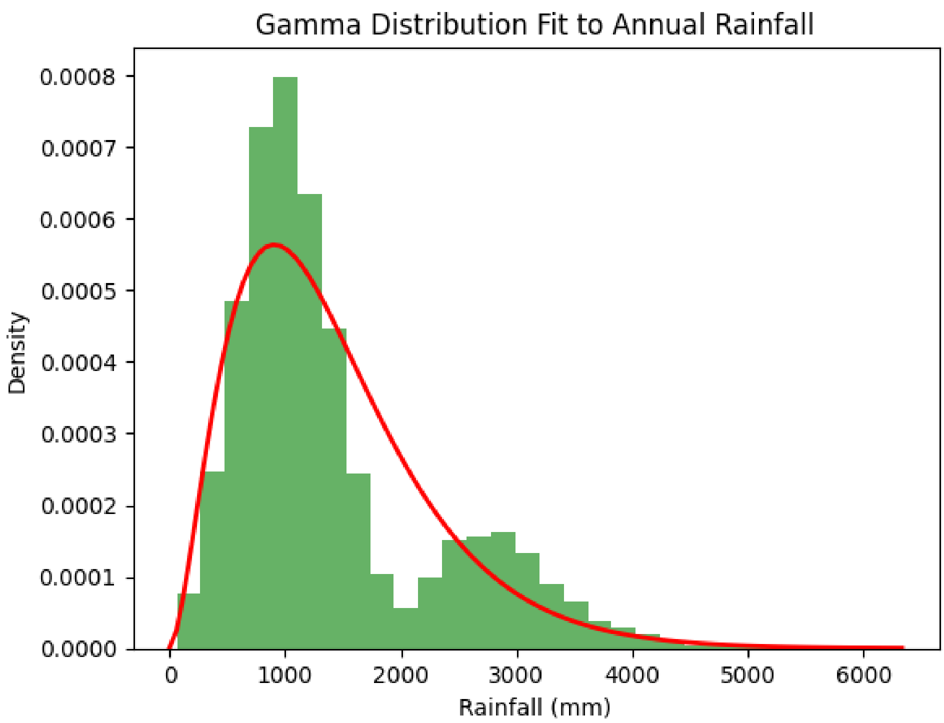

Figure 2.1: Estimated Gamma Plot

2.2.3.4 Fitting the data with Gamma distribution using Statsmodels library

```
annual_rainfall = df['ANNUAL']
annual_rainfall = annual_rainfall[annual_rainfall > 0]  # remove zero values

# fit a gamma distribution to the data
shape, loc, scale = stats.gamma.fit(annual_rainfall, floc=0)
print(f"Shape: {shape}, Location: {loc}, Scale: {scale}")

# plot the histogram of the data
plt.hist(annual_rainfall, bins=30, density=True, alpha=0.6, color='g')

# plot the fitted gamma distribution
x = np.linspace(0, annual_rainfall.max(), 100)
```

```
pdf = stats.gamma.pdf(x, shape, loc=loc, scale=scale)
plt.plot(x, pdf, 'r-', lw=2)
plt.title('Gamma Distribution Fit to Annual Rainfall')
plt.xlabel('Rainfall (mm)')
plt.ylabel('Density')
plt.show()
```

Output:

```
Shape: 2.7932780634164835, Location: 0, Scale: 505.14444595955604
```

2.2.3.5 Observation

- We can observe that the parameters estimated using the Maximum Likelihood technique are very close to the parameters obtained by using *stats.gamma.fit* from the Statsmodels library.
- This verifies the correctness of the Maximum Likelihood Iterative approach.

Chapter 3

Confidence Interval for the Variance of a Normal Distribution

3.1 Theoretical Foundation

For a random sample $\{x_1, x_2, \ldots, x_n\}$ from a normal distribution $N(\mu, \sigma^2)$ with unknown mean μ and variance σ^2, we know that:

$$\frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{n-1} \tag{3.1}$$

where $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$ is the sample variance and χ^2_{n-1} is the chi-square distribution with $n-1$ degrees of freedom.

3.2 Procedure for Constructing a 95% Confidence Interval

Algorithm 3 95% Confidence Interval for Variance (Unknown Mean)

1: Calculate the sample mean: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

2: Calculate the sample variance: $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$

3: Find the critical values from the chi-square distribution with $n-1$ degrees of freedom:

$$\chi^2_{0.975,n-1} = \text{lower critical value (left tail, 2.5\%)} \tag{3.2}$$

$$\chi^2_{0.025,n-1} = \text{upper critical value (right tail, 2.5\%)} \tag{3.3}$$

4: Construct the 95% confidence interval for σ^2:

$$\left(\frac{(n-1)s^2}{\chi^2_{0.025,n-1}}, \frac{(n-1)s^2}{\chi^2_{0.975,n-1}}\right) \tag{3.4}$$

5: Optionally, derive the 95% confidence interval for σ by taking the square root of the endpoints:

$$\left(\sqrt{\frac{(n-1)s^2}{\chi^2_{0.025,n-1}}}, \sqrt{\frac{(n-1)s^2}{\chi^2_{0.975,n-1}}}\right) \tag{3.5}$$

3.3 Implementation

3.3.1 Dataset

We use the *Annual Rainfall* from the Rainfall in India dataset dataset from Kaggle, which includes annual rainfall data for various regions over the last 125 years.

3.3.2 Objective

We model the Annual rainfall data as a Normal Distribution with unknown mean and variance. We aim to estimate the variance parameter using appropriate statistical tests.

3.3.3 Code Implementation

3.3.3.1 Calculating the Confidence Interval

```
alpha = 0.05

# take a sample
sample = np.random.choice(annual_rainfall, size=100, replace=False)
n = len(sample)

# calculate the sample variance
s2 = np.var(sample, ddof=1)

# calculate the chi-squared critical values
chi2_lower = stats.chi2.ppf(alpha / 2, n - 1)
chi2_upper = stats.chi2.ppf(1 - alpha / 2, n - 1)

# calculate the confidence interval
ci_lower = (n - 1) * s2 / chi2_upper
ci_upper = (n - 1) * s2 / chi2_lower

print(f"95% Confidence Interval for Variance: ({ci_lower}, {ci_upper})")
print(f"95% Confidence Interval for Standard Deviation: ({np.sqrt(ci_lower)}, {
    np.sqrt(ci_upper)})")
```

Output:

```
95% Confidence Interval for Variance: (638871.1969929087, 1118373.79426766)
95% Confidence Interval for Standard Deviation: (799.2941867628643, 1057.5319353417465)
```

3.3.3.2 Modelling the data using Statsmodels library

```
annual_rainfall = df['ANNUAL']
annual_rainfall = annual_rainfall[annual_rainfall > 0]  # remove zero values

# fit a normal distribution to the data
mu, std = stats.norm.fit(annual_rainfall)
print(f"Mean: {mu}, Std: {std}")

# plot the histogram of the data
plt.hist(annual_rainfall, bins=30, density=True, alpha=0.6, color='g')

# plot the fitted normal distribution
x = np.linspace(0, annual_rainfall.max(), 100)
pdf = stats.norm.pdf(x, mu, std)
plt.plot(x, pdf, 'r-', lw=2)
plt.title('Normal Distribution Fit to Annual Rainfall')
plt.xlabel('Rainfall (mm)')
plt.ylabel('Density')
plt.show()
```

Output:

```
Mean: 1411.0088997555013, Std: 903.7360635097127
```

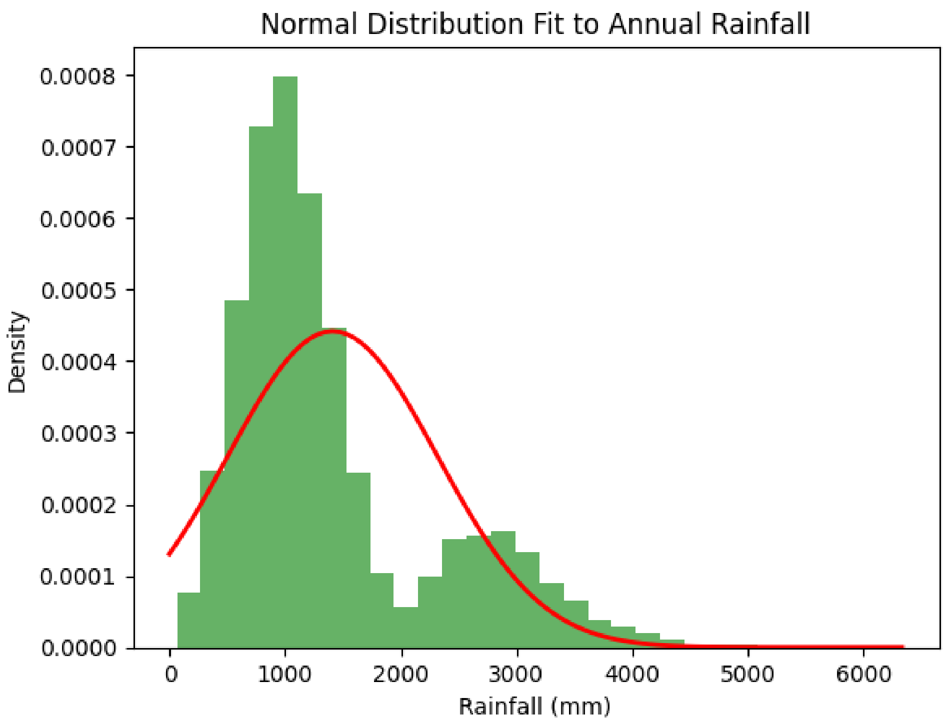

Figure 3.1: Modelled Normal distribution

Chapter 4

Confidence Interval for the Difference of Means

4.1 Theoretical Foundation

Let $X_1, X_2, \ldots, X_{n_1}$ be a random sample from $N(\mu_1, \sigma_1^2)$ and $Y_1, Y_2, \ldots, Y_{n_2}$ be a random sample from $N(\mu_2, \sigma_2^2)$, with both populations independent. We consider two cases:

4.1.1 Case 1: Equal Variances ($\sigma_1^2 = \sigma_2^2 = \sigma^2$)

The pooled variance estimator is:

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

The confidence interval for $\mu_1 - \mu_2$ is:

$$(\bar{X} - \bar{Y}) \pm t_{\alpha/2, n_1+n_2-2} \cdot s_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

4.1.2 Case 2: Unequal Variances (Welch-Satterthwaite Procedure)

When $\sigma_1^2 \neq \sigma_2^2$, we use:

$$(\bar{X} - \bar{Y}) \pm t_{\alpha/2, \nu} \cdot \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

where the degrees of freedom ν is approximated by:

$$\nu = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{(s_1^2/n_1)^2}{n_1 - 1} + \frac{(s_2^2/n_2)^2}{n_2 - 1}}$$

4.2 Procedure for Constructing a 95% Confidence Interval

Algorithm 4 95% CI for $\mu_1 - \mu_2$ (Welch's Procedure)

1: Calculate sample means:

$$\bar{x} = \frac{1}{n_1}\sum_{i=1}^{n_1} x_i, \quad \bar{y} = \frac{1}{n_2}\sum_{i=1}^{n_2} y_i$$

2: Calculate sample variances:

$$s_1^2 = \frac{1}{n_1 - 1}\sum_{i=1}^{n_1}(x_i - \bar{x})^2, \quad s_2^2 = \frac{1}{n_2 - 1}\sum_{i=1}^{n_2}(y_i - \bar{y})^2$$

3: Compute standard error:

$$SE = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

4: Calculate degrees of freedom:

$$\nu = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{(s_1^2/n_1)^2}{n_1 - 1} + \frac{(s_2^2/n_2)^2}{n_2 - 1}}$$

5: Find critical value $t_{0.025,\nu}$ from t-distribution table

6: Construct 95% CI:

$$(\bar{x} - \bar{y}) \pm t_{0.025,\nu} \cdot SE$$

4.3 Implementation

4.3.1 Dataset

We used the historical rainfall dataset from the Indian Meteorological Department, which includes annual rainfall data for various regions over the last 125 years.

4.3.2 Objective

Our goal is to compare the mean annual rainfall between **Lakshadweep** and **Andaman & Nicobar Islands** using a statistical test.

Key Assumptions:

- Rainfall data from both regions are independent, which is a reasonable assumption as the two regions are geographically distant and located on different coasts of India.
- Population variances are assumed to be unequal due to regional and climatic differences. This justifies the use of Welch's t-test.

4.3.3 Code

Listing 4.1: Two-Sample Welch's t-Test for Difference in Means

```python
from scipy.stats import t
import numpy as np

# Extracting populations
population1 = df[df['SUBDIVISION'] == 'LAKSHADWEEP']['ANNUAL'].dropna()
population2 = df[df['SUBDIVISION'] == 'ANDAMAN & NICOBAR ISLANDS']['ANNUAL'].
    dropna()

# Population statistics
mean1 = population1.mean()
mean2 = population2.mean()
```

```python
print(f"Mean rainfall in Lakshadweep: {mean1:.4f}")
print(f"Mean rainfall in Andaman & Nicobar Islands: {mean2:.4f}")
print(f"Population mean difference: {mean1 - mean2:.4f}")

# Sampling from populations
sample_size = 100
sample1 = population1.sample(n=sample_size, random_state=1)
sample2 = population2.sample(n=sample_size, random_state=1)

n1, n2 = len(sample1), len(sample2)
mean_sample1, mean_sample2 = sample1.mean(), sample2.mean()
var1, var2 = np.var(sample1, ddof=1), np.var(sample2, ddof=1)

# Welch-Satterthwaite degrees of freedom
numerator = (var1/n1 + var2/n2)**2
denominator = ((var1/n1)**2 / (n1 - 1)) + ((var2/n2)**2 / (n2 - 1))
df = numerator / denominator

# Confidence interval
std_error = np.sqrt(var1/n1 + var2/n2)
t_critical = t.ppf(0.975, df)
mean_diff = mean_sample1 - mean_sample2
ci_lower = mean_diff - t_critical * std_error
ci_upper = mean_diff + t_critical * std_error

print(f"Sample Mean Lakshadweep: {mean_sample1:.4f}")
print(f"Sample Mean Andaman: {mean_sample2:.4f}")
print(f"Sample Mean Difference: {mean_diff:.4f}")
print(f"95% Confidence Interval: ({ci_lower:.4f}, {ci_upper:.4f})")
```

4.3.4 Results

Population Statistics:

- Mean rainfall in Lakshadweep: $\mu_1 = 1590.8864$
- Mean rainfall in Andaman & Nicobar Islands: $\mu_2 = 2927.4394$
- Population mean difference: $\mu_1 - \mu_2 = -1336.5530$

Sample Statistics:

- Sample mean rainfall in Lakshadweep: $\bar{x}_1 = 1590.2980$
- Sample mean rainfall in Andaman & Nicobar Islands: $\bar{x}_2 = 2924.9880$
- Sample mean difference: $\bar{x}_1 - \bar{x}_2 = -1334.6900$

95% Confidence Interval for the Difference in Means:

$$(-1432.0621, -1237.3179)$$

Interpretation: Since the confidence interval does not contain zero and is entirely negative, we can conclude that the average annual rainfall in Lakshadweep is significantly lower than that in Andaman & Nicobar Islands at the 95% confidence level.

4.3.5 Experiment: Understanding the Confidence Interval

4.3.5.1 Objective

The goal of this experiment is to empirically validate the interpretation of confidence intervals. Specifically, we explore how often a confidence interval captures the true population parameter across repeated sampling. We evaluate this behavior across different confidence levels and sample sizes.

4.3.5.2 Methodology

We performed repeated sampling from the rainfall data of two independent regions: Lakshadweep and the Andaman & Nicobar Islands. For each sample:

- We compute the sample mean difference in annual rainfall.
- Construct a confidence interval using Welch's t-test (assuming unequal variances).
- Check whether the true population mean difference lies within the computed interval.

This process is repeated for different confidence levels **(75%, 80%, 90%, 95%, 97.5%)** and sample sizes **(30, 50, 100)**, with **1000 iterations** per configuration.

4.3.5.3 Code

Listing 4.2: Effect of Sample Size and Confidence Level

```python
alphas = [0.75, 0.80, 0.90, 0.95, 0.975,1]
n_samples = 1000
sample_sizes = [30, 50, 100]

for sample_size in sample_sizes:
    print(f'Sample Size {sample_size}')
    for alpha in alphas:
        count = 0
        for i in range(n_samples):
            sample1 = population1.sample(n=sample_size)
            sample2 = population2.sample(n=sample_size)

            sample_mean1 = sample1.mean()
            sample_mean2 = sample2.mean()
            n1, n2 = len(sample1), len(sample2)

            var1, var2 = np.var(sample1, ddof=1), np.var(sample2, ddof=1)

            # Welch-Satterthwaite approximation
            numerator = ((var1/n1 + var2/n2))**2
            denominator = ((var1/n1)**2/(n1 - 1)) + ((var2/n2)**2/(n2 - 1))
            degrees_of_freedom = numerator / denominator

            std_error = np.sqrt(var1/n1 + var2/n2)
            t_critical = t.ppf(alpha, degrees_of_freedom)

            mean_diff = sample_mean1 - sample_mean2
            ci_lower = mean_diff - t_critical * std_error
            ci_upper = mean_diff + t_critical * std_error

            if ci_lower < actual_population_diff < ci_upper:
                count += 1

        print(f'Confidence Level: {alpha*100:.1f}%, Containment Rate: {count/
            n_samples*100:.2f}%')
```

4.3.5.4 Results and Observations

- **Sample Size: 30**
  - 75% Confidence → 58.00% CIs contained the true mean difference
  - 80% Confidence → 67.30%
  - 90% Confidence → 86.40%
  - 95% Confidence → 94.70%

- 97.5% Confidence → 97.90%
- 100% Confidence → 100%

- **Sample Size: 50**
  - 75% Confidence → 63.80%
  - 80% Confidence → 77.80%
  - 90% Confidence → 92.00%
  - 95% Confidence → 97.40%
  - 97.5% Confidence → 99.30%
  - 100% Confidence → 100%
- **Sample Size: 100**
  - 75% Confidence → 99.90%
  - 80% Confidence → 100.00%
  - 90% Confidence → 100.00%
  - 95% Confidence → 100.00%
  - 97.5% Confidence → 100.00%
  - 100% Confidence → 100%

4.3.5.5 Analysis

- For smaller sample sizes (30 and 50), the empirical containment rate closely aligns with the theoretical confidence level.
- As the sample size increases, confidence intervals become narrower and more reliable. For large sample sizes, containment rates tend to slightly exceed the theoretical confidence level, especially in this dataset where the population difference is large and stable.
- For Confidence Level 100 , We get entire range (-inf,inf) as confidence interval and thus containment rate is always 100

4.3.5.6 Conclusions

- **Effect of Sample Size:** Increasing sample size improves the precision and reliability of confidence intervals.
- **Interpretation of Confidence Level:** A confidence level (e.g., 95%) indicates the proportion of confidence intervals, over repeated sampling, expected to capture the true population parameter.

Empirical Validation: This experiment confirms the frequentist interpretation of confidence intervals and demonstrates the convergence of empirical containment to the nominal confidence level

Chapter 5

Hypothesis Testing of probability of success of Bernoulli Distributed Data

5.1 Procedure for Hypothesis Testing

Algorithm 5 Approximate α Significance Level Test for Bernoulli Distributed Data (Using Sample Mean)

$$\text{For testing } H_0 : p \leq \frac{1}{2} \text{ versus } H_1 : p > \frac{1}{2} \text{ with Bernoulli data:} \tag{5.1}$$
$$\text{1. Let } X \sim \text{Binomial}(n, p) \text{ be the number of successes in } n \text{ trials} \tag{5.2}$$
$$\text{and let } \bar{X} = \frac{X}{n} \text{ be the sample mean (proportion of successes)} \tag{5.3}$$
$$\text{2. For large } n\text{, by the Central Limit Theorem:} \tag{5.4}$$
$$\bar{X} \dot{\sim} N\left(p, \frac{p(1-p)}{n}\right) \tag{5.5}$$
$$\text{3. Under } H_0\text{, the least favorable configuration is } p = \frac{1}{2} \tag{5.6}$$
$$\text{4. The test statistic is: } Z = \frac{\bar{X} - \frac{1}{2}}{\sqrt{\frac{1}{4n}}} \tag{5.7}$$
$$\text{5. Rejection rule: Reject } H_0 \text{ if } Z \geq z_\alpha \text{ or equivalently if } \bar{X} \geq \frac{1}{2} + z_\alpha \sqrt{\frac{1}{4n}} \tag{5.8}$$
$$\tag{5.9}$$

5.2 Implementation for Significance Level 0.05

We perform a one-sided hypothesis test:

$$H_0 : p \leq \frac{1}{2} \quad \text{vs.} \quad H_1 : p > \frac{1}{2}$$

We reject the null hypothesis H_0 if the value of the test statistic $Z > z_{0.05} = 1.96$, corresponding to a significance level of $\alpha = 0.05$.

5.2.1 Dataset

We used the publicly available data set from Kaggle:
https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data?resource=download

In this dataset, the diagnosis of cancer being **benign (B)** or **malignant (M)** is treated as a Bernoulli random variable with unknown parameter p. Since each observation is assumed independent, this can be modeled as a sequence of i.i.d. Bernoulli trials.

5.2.2 Code

Listing 5.1: One-Sample Z-Test for Bernoulli Distribution

```
import numpy as np
import pandas as pd

db = pd.read_csv("data.csv")
db_trunc = db['diagnosis']

# Population statistics
res = db_trunc.value_counts()
B, M = res.values
print("# Benign:", B)
print("# Malignant:", M)

p = B / (B + M)
print("Population Probability of being Benign:", p)

# Sample-based test
n = 30
sample = db_trunc.sample(n=n)
sample = sample.transform(lambda x: 1 if x == 'B' else 0)
X_mean = sample.mean()
Z = (np.sqrt(n) * (X_mean - 0.5)) / 0.5
print("Test Statistic Z:", Z)
```

5.2.3 Results

Population Statistics:

- Number of Benign cases: 357
- Number of Malignant cases: 212
- Population Probability of Benign: $\hat{p} = 0.6274$

Hypothesis Testing Result:
For $n = 30$, the test statistic was computed to be:

$$Z = 1.826 < 1.96$$

Hence, we **fail to reject** H_0 at significance level 0.05.

5.2.4 Effect of Sample Size

5.2.4.1 Experiment

We varied the sample size and computed the test statistic for each, checking whether H_0 is rejected.

Listing 5.2: Effect of Sample Size on Hypothesis Test

```
for n in range(1, 300, 5):
    sample = db_trunc.sample(n=n)
    sample = sample.transform(lambda x: 1 if x == 'B' else 0)
    X_mean = sample.mean()
```

```python
    Z = (np.sqrt(n) * (X_mean - 0.5)) / 0.5
    decision = "Reject H0" if Z > 1.96 else "Fail to Reject H0"
    print(f"n={n} | Z={Z:.2f} | {decision}")
```

5.2.4.2 Observations

Initially, with small sample sizes, the decision fluctuates due to high variance:

```
n 1 value 1.00   Failed to Reject H0
n 6 value 0.82   Failed to Reject H0
n 11 value 2.11   Reject H0
n 16 value 1.00   Failed to Reject H0
n 21 value 1.09   Failed to Reject H0
n 26 value 3.14   Reject H0
n 31 value 0.90   Failed to Reject H0
n 36 value 1.00   Failed to Reject H0
n 41 value 0.47   Failed to Reject H0
n 46 value 0.59   Failed to Reject H0
n 51 value 3.50   Reject H0
n 56 value 2.14   Reject H0
n 61 value 1.66   Failed to Reject H0
n 66 value 2.22   Reject H0
n 71 value 2.25   Reject H0
n 76 value 1.61   Failed to Reject H0
n 81 value 2.78   Reject H0
n 86 value 1.08   Failed to Reject H0
n 91 value 2.20   Reject H0
n 96 value 3.88   Reject H0
n 101 value 3.68   Reject H0
n 106 value 4.47   Reject H0
n 111 value 2.75   Reject H0
n 116 value 2.79   Reject H0
n 121 value 1.73   Failed to Reject H0
n 126 value 0.89   Failed to Reject H0
n 131 value 3.41   Reject H0
n 136 value 2.40   Reject H0
n 141 value 3.62   Reject H0
n 146 value 4.47   Reject H0
n 151 value 2.69   Reject H0
n 156 value 3.84   Reject H0
n 161 value 2.76   Reject H0
n 166 value 3.42   Reject H0
n 171 value 4.05   Reject H0
n 176 value 3.02   Reject H0
n 181 value 2.16   Reject H0
n 186 value 3.37   Reject H0
n 191 value 3.26   Reject H0
n 196 value 3.14   Reject H0
n 201 value 5.71   Reject H0
n 206 value 3.48   Reject H0
n 211 value 2.96   Reject H0
n 216 value 4.22   Reject H0
n 221 value 4.24   Reject H0
n 226 value 4.66   Reject H0
n 231 value 3.22   Reject H0
n 236 value 3.12   Reject H0
n 241 value 3.80   Reject H0
n 246 value 3.06   Reject H0
```

```
n 251 value 3.85   Reject H0
n 256 value 4.38   Reject H0
n 261 value 4.02   Reject H0
n 266 value 3.80   Reject H0
n 271 value 4.31   Reject H0
n 276 value 4.45   Reject H0
n 281 value 5.31   Reject H0
n 286 value 3.67   Reject H0
n 291 value 4.98   Reject H0
n 296 value 4.30   Reject H0
```

We observe that after a certain sample size, the test consistently rejects H_0. This threshold depends on the underlying population probability and the randomness in sampling.

5.2.4.3 Explanation

Since the empirical population probability of a benign diagnosis is $\hat{p} \approx 0.6274$, which is greater than 0.5, larger sample sizes lead to more consistent estimates of the true mean. As the sample mean increasingly reflects this bias, the test statistic tends to exceed the critical value 1.96, leading to consistent rejection of H_0.