Assignment 4

> **2024017201 컴퓨터소프트웨어학부 최선웅**
> **12034 수치해석**

- main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define NRANSI
#include "nrutil.h"

#define select nrselect
#include "nr.h"
#undef select

#define NBIN 100

float ran1(long *idum);
float gasdev(long *idum);

float uniform_random(float a, float b, long *idum) {
    return a + (b - a) * ran1(idum);
}

float gaussian_random(float m, float s, long *idum) {
    return m + s * gasdev(idum);
}

void write_samples(const char *filename, float *samples, int n) {
    FILE *fp = fopen(filename, "w");
    if (fp == NULL) {
        fprintf(stderr, "Error: Cannot open file %s\n", filename);
        return;
    }
```

```
    for (int i = 0; i < n; i++) {
        fprintf(fp, "%.6f\n", samples[i]);
    }
    fclose(fp);
    printf("Samples written to %s\n", filename);
}

void generate_samples(int nsamples, float a, float b, float m, float s, long *s
eed) {
    float *uniform_samples = (float *)malloc(nsamples * sizeof(float));
    float *gaussian_samples = (float *)malloc(nsamples * sizeof(float));

    if (uniform_samples == NULL || gaussian_samples == NULL) {
        fprintf(stderr, "Error: Memory allocation failed\n");
        return;
    }

    for (int i = 0; i < nsamples; i++) {
        uniform_samples[i] = uniform_random(a, b, seed);
    }
    for (int i = 0; i < nsamples; i++) {
        gaussian_samples[i] = gaussian_random(m, s, seed);
    }

    char uniform_filename[50];
    char gaussian_filename[50];
    sprintf(uniform_filename, "uniform_%d.txt", nsamples);
    sprintf(gaussian_filename, "gaussian_%d.txt", nsamples);

    write_samples(uniform_filename, uniform_samples, nsamples);
    write_samples(gaussian_filename, gaussian_samples, nsamples);

    float uniform_mean = 0.0, gaussian_mean = 0.0;
    for (int i = 0; i < nsamples; i++) {
        uniform_mean += uniform_samples[i];
        gaussian_mean += gaussian_samples[i];
    }
    uniform_mean /= nsamples;
```

```
    gaussian_mean /= nsamples;

    float uniform_var = 0.0, gaussian_var = 0.0;
    for (int i = 0; i < nsamples; i++) {
        float diff_u = uniform_samples[i] - uniform_mean;
        float diff_g = gaussian_samples[i] - gaussian_mean;
        uniform_var += diff_u * diff_u;
        gaussian_var += diff_g * diff_g;
    }
    uniform_var /= nsamples;
    gaussian_var /= nsamples;

    printf("\n[N = %d]\n", nsamples);
    printf("Uniform distribution [%.1f, %.1f]:\n", a, b);
    printf("  Mean: %.4f (Expected: %.4f)\n", uniform_mean, (a + b) / 2.0);
    printf("  Variance: %.4f (Expected: %.4f)\n", uniform_var, (b - a) * (b - a)
/ 12.0);
    printf("Gaussian distribution (m=%.1f, s=%.1f):\n", m, s);
    printf("  Mean: %.4f (Expected: %.4f)\n", gaussian_mean, m);
    printf("  Variance: %.4f (Expected: %.4f)\n", gaussian_var, s * s);
    printf("\n");

    free(uniform_samples);
    free(gaussian_samples);
}

int main(void) {
    float a = -3.0;
    float b = 4.0;
    float m = 0.5;
    float s = 1.5;
    long seed = -12345;

    printf("==============================================
===========\n");
    printf("Random Number Generation - Homework #4\n");
    printf("==============================================
===========\n");
```

```
    printf("Parameters:\n");
    printf("  Uniform distribution: [%.1f, %.1f]\n", a, b);
    printf("  Gaussian distribution: mean=%.1f, std=%.1f\n", m, s);
    printf("  Histogram bins: %d\n", NBIN);
    printf("================================================
===========\n\n");

    int sample_sizes[] = {100, 1000, 10000, 100000};
    int num_sizes = 4;

    for (int i = 0; i < num_sizes; i++) {
        generate_samples(sample_sizes[i], a, b, m, s, &seed);
    }

    printf("================================================
===========\n");
    printf("All samples generated successfully!\n");
    printf("Run 'python histogram.py' to visualize the histograms.\n");
    printf("================================================
===========\n");

    return 0;
}
```

해당 코드는 Uniform distribution과 Gaussian distribution의 난수를 여러 크기로 생성하여 파일로 저장하고, 각각의 평균과 분산을 계산하여 출력한다.

- `histogram.py`

```
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path

def load_samples(filename):
    try:
        samples = np.loadtxt(filename)
        return samples
```

```
    except FileNotFoundError:
        print(f"Error: File {filename} not found")
        return None

def plot_histogram(ax, samples, bins, title, color, expected_range=None):
    if samples is None or len(samples) == 0:
        return
    mean = np.mean(samples)
    std = np.std(samples, ddof=1)
    counts, bin_edges, patches = ax.hist(samples, bins=bins, density=True,
                                         alpha=0.7, color=color, edgecolor='black', linewidt
h=0.5)
    ax.axvline(mean, color='red', linestyle='--', linewidth=2, label=f'Mean:
{mean:.4f}')
    ax.set_title(f'{title}\nN = {len(samples)}', fontsize=12, fontweight='bold')
    ax.set_xlabel('Value', fontsize=10)
    ax.set_ylabel('Density', fontsize=10)
    ax.legend(loc='upper right', fontsize=8)
    ax.grid(True, alpha=0.3)
    stats_text = f'Mean: {mean:.4f}\nStd: {std:.4f}'
    ax.text(0.02, 0.98, stats_text, transform=ax.transAxes,
            fontsize=9, verticalalignment='top',
            bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.5))

def create_uniform_histograms(sample_sizes, a=-3, b=4, bins=100):
    fig, axes = plt.subplots(2, 2, figsize=(14, 10))
    fig.suptitle(f'Uniform Distribution Histograms [a={a}, b={b}]',
                 fontsize=16, fontweight='bold')
    axes = axes.flatten()
    for i, n in enumerate(sample_sizes):
        filename = f'uniform_{n}.txt'
        samples = load_samples(filename)
        if samples is not None:
            plot_histogram(axes[i], samples, bins, 'Uniform Distribution', 'skyblu
e', (a, b))
    plt.tight_layout()
    plt.savefig('uniform_histograms.png', dpi=300, bbox_inches='tight')
    print("Saved: uniform_histograms.png")
```

```
    plt.show()

def create_gaussian_histograms(sample_sizes, m=0.5, s=1.5, bins=100):
    fig, axes = plt.subplots(2, 2, figsize=(14, 10))
    fig.suptitle(f'Gaussian Distribution Histograms [mean={m}, std={s}]',
                 fontsize=16, fontweight='bold')
    axes = axes.flatten()
    for i, n in enumerate(sample_sizes):
        filename = f'gaussian_{n}.txt'
        samples = load_samples(filename)
        if samples is not None:
            plot_histogram(axes[i], samples, bins, 'Gaussian Distribution', 'lightc
oral')
            x = np.linspace(samples.min(), samples.max(), 1000)
            theoretical = (1 / (s * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x - m) / s)
** 2)
            axes[i].plot(x, theoretical, 'b-', linewidth=2, label=f'Theoretical N
({m},{s}²)')
            axes[i].legend(loc='upper right', fontsize=8)
    plt.tight_layout()
    plt.savefig('gaussian_histograms.png', dpi=300, bbox_inches='tight')
    print("Saved: gaussian_histograms.png")
    plt.show()

def create_comparison_plot(sample_sizes):
    fig = plt.figure(figsize=(16, 10))
    for i, n in enumerate(sample_sizes):
        ax1 = plt.subplot(2, 4, i+1)
        filename = f'uniform_{n}.txt'
        samples = load_samples(filename)
        if samples is not None:
            ax1.hist(samples, bins=100, density=True, alpha=0.7,
                     color='skyblue', edgecolor='black', linewidth=0.5)
            ax1.set_title(f'Uniform (N={n})', fontsize=10, fontweight='bold')
            ax1.set_xlabel('Value', fontsize=8)
            ax1.set_ylabel('Density', fontsize=8)
            ax1.grid(True, alpha=0.3)
            ax1.axhline(1/7, color='red', linestyle='--', linewidth=2, label='Theore
```

```
tical')
            ax1.legend(fontsize=7)
        ax2 = plt.subplot(2, 4, i+5)
        filename = f'gaussian_{n}.txt'
        samples = load_samples(filename)
        if samples is not None:
            ax2.hist(samples, bins=100, density=True, alpha=0.7,
                     color='lightcoral', edgecolor='black', linewidth=0.5)
            ax2.set_title(f'Gaussian (N={n})', fontsize=10, fontweight='bold')
            ax2.set_xlabel('Value', fontsize=8)
            ax2.set_ylabel('Density', fontsize=8)
            ax2.grid(True, alpha=0.3)
            x = np.linspace(samples.min(), samples.max(), 1000)
            theoretical = (1 / (1.5 * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x - 0.5) /
1.5) ** 2)
            ax2.plot(x, theoretical, 'b-', linewidth=2, label='Theoretical')
            ax2.legend(fontsize=7)
    plt.suptitle('Convergence of Histograms with Sample Size',
                 fontsize=16, fontweight='bold')
    plt.tight_layout()
    plt.savefig('comparison_histograms.png', dpi=300, bbox_inches='tight')
    print("Saved: comparison_histograms.png")
    plt.show()

def print_statistics(sample_sizes):
    print("\n" + "="*80)
    print("STATISTICS SUMMARY")
    print("="*80)
    print("\n[UNIFORM DISTRIBUTION: a=-3, b=4]")
    print(f"{'N':<10} {'Mean':<12} {'Std Dev':<12} {'Theoretical Mean':<20}
{'Theoretical Std':<20}")
    print("-"*80)
    for n in sample_sizes:
        filename = f'uniform_{n}.txt'
        samples = load_samples(filename)
        if samples is not None:
            mean = np.mean(samples)
            std = np.std(samples, ddof=1)
```

```
            theo_mean = 0.5
            theo_std = np.sqrt((4-(-3))**2 / 12)
            print(f"{n:<10} {mean:<12.4f} {std:<12.4f} {theo_mean:<20.4f} {theo
_std:<20.4f}")
    print("\n[GAUSSIAN DISTRIBUTION: mean=0.5, std=1.5]")
    print(f"{'N':<10} {'Mean':<12} {'Std Dev':<12} {'Theoretical Mean':<20}
{'Theoretical Std':<20}")
    print("-"*80)
    for n in sample_sizes:
        filename = f'gaussian_{n}.txt'
        samples = load_samples(filename)
        if samples is not None:
            mean = np.mean(samples)
            std = np.std(samples, ddof=1)
            theo_mean = 0.5
            theo_std = 1.5
            print(f"{n:<10} {mean:<12.4f} {std:<12.4f} {theo_mean:<20.4f} {theo
_std:<20.4f}")
    print("\n" + "="*80)

def main():
    print("="*80)
    print("Random Number Generation - Histogram Visualization")
    print("="*80)
    sample_sizes = [100, 1000, 10000, 100000]
    print("\nCreating uniform distribution histograms...")
    create_uniform_histograms(sample_sizes)
    print("\nCreating Gaussian distribution histograms...")
    create_gaussian_histograms(sample_sizes)
    print("\nCreating comparison plot...")
    create_comparison_plot(sample_sizes)
    print_statistics(sample_sizes)
    print("\n" + "="*80)
    print("DISCUSSION: Shape of Histograms vs. Number of Samples")
    print("="*80)
    print("""
As the number of samples increases:
```

```
1. UNIFORM DISTRIBUTION:
   - N=100: Histogram shows high variability with irregular bins
   - N=1000: Shape becomes more rectangular but still noisy
   - N=10000: Clearly approaching uniform flat distribution
   - N=100000: Very smooth, almost perfectly flat rectangular shape

2. GAUSSIAN DISTRIBUTION:
   - N=100: Shows rough bell shape but with significant noise
   - N=1000: Bell shape is clearer with some irregularities
   - N=10000: Smooth bell curve closely matching theoretical curve
   - N=100000: Almost perfectly matches the theoretical Gaussian curve

3. LAW OF LARGE NUMBERS:
   - Sample mean converges to theoretical mean
   - Sample variance converges to theoretical variance
   - Histogram shape converges to theoretical probability density

4. CONVERGENCE RATE:
   - Standard error decreases as 1/sqrt(N)
   - For accurate distribution shape, N >= 10000 is recommended
   - For smooth density estimation, N >= 100000 is ideal
""")
    print("="*80)
    print("All histograms generated successfully!")
    print("="*80)

if __name__ == "__main__":
    main()
```

Uniform과 Gaussian 분포 난수들을 다양한 샘플 크기로 읽어 히스토그램과 통계 요약을 자동으로 시각화한다. 각 분포와 샘플 크기별로 PNG 이미지를 생성하고, 이론적 분포와 실제 샘플 데이터를 비교한다.

- Output

```
================================================================================
```

```
Random Number Generation - Homework #4
================================================================
Parameters:
  Uniform distribution: [-3.0, 4.0]
  Gaussian distribution: mean=0.5, std=1.5
  Histogram bins: 100
================================================================

Samples written to uniform_100.txt
Samples written to gaussian_100.txt

[N = 100]
Uniform distribution [-3.0, 4.0]:
  Mean: 0.3962 (Expected: 0.5000)
  Variance: 4.1519 (Expected: 4.0833)
Gaussian distribution (m=0.5, s=1.5):
  Mean: 0.7028 (Expected: 0.5000)
  Variance: 1.7480 (Expected: 2.2500)

Samples written to uniform_1000.txt
Samples written to gaussian_1000.txt

[N = 1000]
Uniform distribution [-3.0, 4.0]:
  Mean: 0.5069 (Expected: 0.5000)
  Variance: 4.0906 (Expected: 4.0833)
Gaussian distribution (m=0.5, s=1.5):
  Mean: 0.4342 (Expected: 0.5000)
  Variance: 2.1367 (Expected: 2.2500)

Samples written to uniform_10000.txt
Samples written to gaussian_10000.txt

[N = 10000]
Uniform distribution [-3.0, 4.0]:
  Mean: 0.5148 (Expected: 0.5000)
```

```
  Variance: 4.1217 (Expected: 4.0833)
Gaussian distribution (m=0.5, s=1.5):
  Mean: 0.5235 (Expected: 0.5000)
  Variance: 2.2189 (Expected: 2.2500)

Samples written to uniform_100000.txt
Samples written to gaussian_100000.txt

[N = 100000]
Uniform distribution [-3.0, 4.0]:
  Mean: 0.5011 (Expected: 0.5000)
  Variance: 4.0840 (Expected: 4.0833)
Gaussian distribution (m=0.5, s=1.5):
  Mean: 0.5036 (Expected: 0.5000)
  Variance: 2.2435 (Expected: 2.2500)

==================================================
=====
All samples generated successfully!
Run 'python histogram.py' to visualize the histograms.
==================================================
=====
python3 histogram.py
==================================================
==========================
Random Number Generation - Histogram Visualization
==================================================
==========================
\nCreating uniform distribution histograms...
Saved: uniform_histograms.png
\nCreating Gaussian distribution histograms...
Saved: gaussian_histograms.png
\nCreating comparison plot...
Saved: comparison_histograms.png
\n==================================================
==========================
STATISTICS SUMMARY
==================================================
```

```
============================
\n[UNIFORM DISTRIBUTION: a=-3, b=4]
N         Mean        Std Dev      Theoretical Mean     Theoretical Std
-------------------------------------------------------------------------
---
100       0.3962      2.0479       0.5000               2.0207
1000      0.5069      2.0235       0.5000               2.0207
10000     0.5148      2.0303       0.5000               2.0207
100000    0.5011      2.0209       0.5000               2.0207
\n[GAUSSIAN DISTRIBUTION: mean=0.5, std=1.5]
N         Mean        Std Dev      Theoretical Mean     Theoretical Std
-------------------------------------------------------------------------
---
100       0.7028      1.3288       0.5000               1.5000
1000      0.4342      1.4625       0.5000               1.5000
10000     0.5235      1.4897       0.5000               1.5000
100000    0.5036      1.4979       0.5000               1.5000
\n=========================================================
============================
\n=========================================================
============================
DISCUSSION: Shape of Histograms vs. Number of Samples
=========================================================
============================
```

As the number of samples increases:

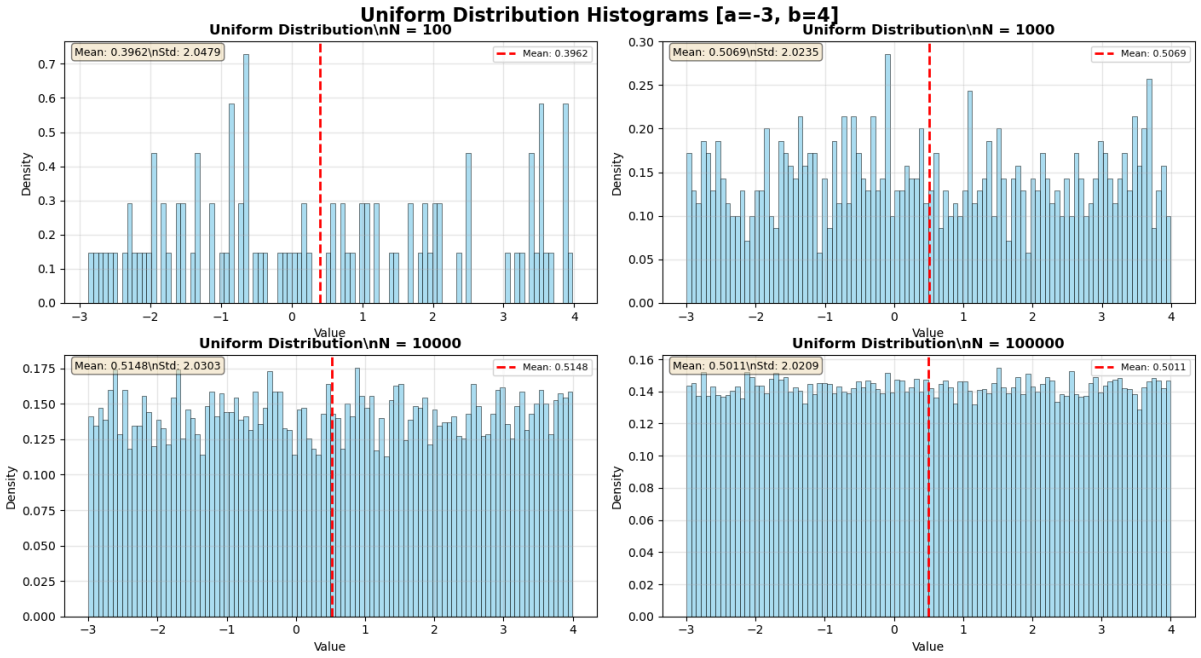
1. UNIFORM DISTRIBUTION:
   - N=100: Histogram shows high variability with irregular bins
   - N=1000: Shape becomes more rectangular but still noisy
   - N=10000: Clearly approaching uniform flat distribution
   - N=100000: Very smooth, almost perfectly flat rectangular shape

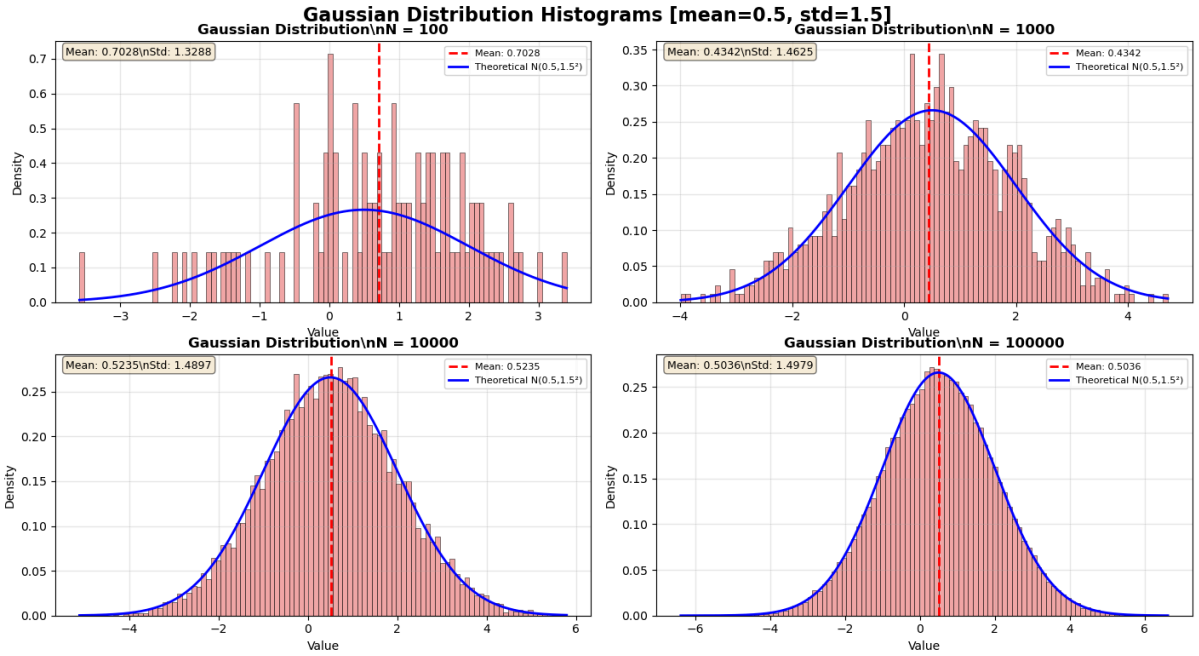
2. GAUSSIAN DISTRIBUTION:
   - N=100: Shows rough bell shape but with significant noise
   - N=1000: Bell shape is clearer with some irregularities
   - N=10000: Smooth bell curve closely matching theoretical curve
   - N=100000: Almost perfectly matches the theoretical Gaussian curve

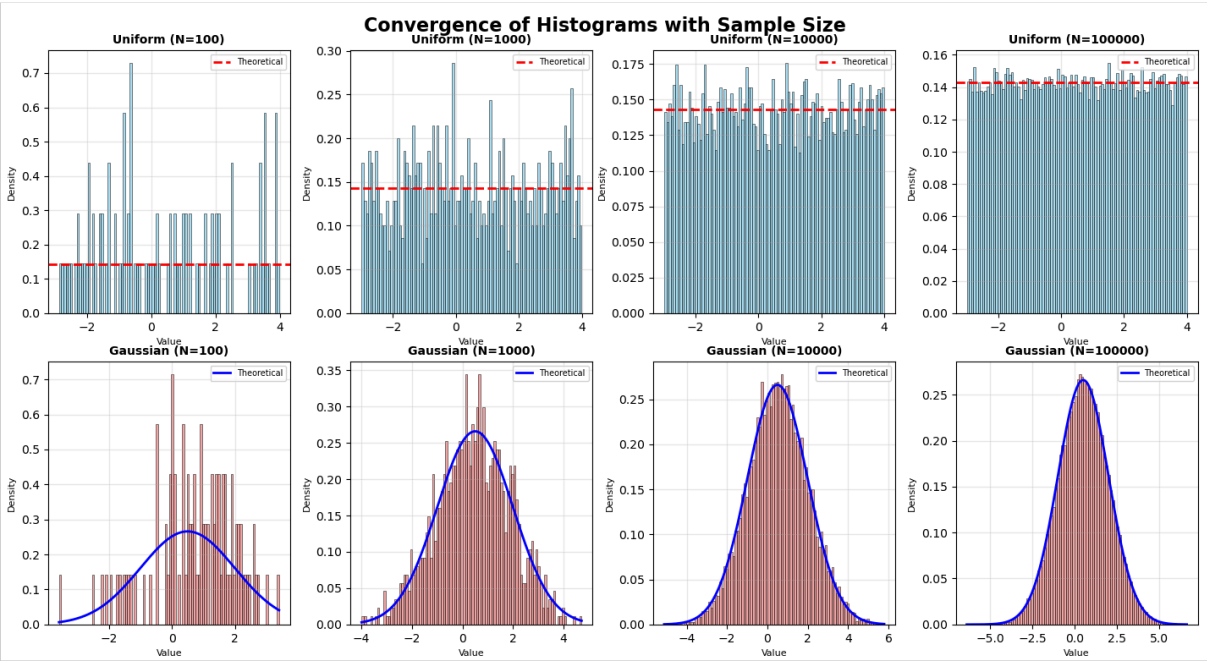
3. LAW OF LARGE NUMBERS:
   - Sample mean converges to theoretical mean
   - Sample variance converges to theoretical variance
   - Histogram shape converges to theoretical probability density

4. CONVERGENCE RATE:
   - Standard error decreases as 1/sqrt(N)
   - For accurate distribution shape, N >= 10000 is recommended
   - For smooth density estimation, N >= 100000 is ideal

출력 결과이다. 샘플 개수가 많을수록 히스토그램 모양은 이론적 분포에 점점 더 가까워지며 표본 평균과 표준편차도 샘플이 많아질수록 기댓값과 거의 일치하게 수렴한다.




Gaussian Distribution Histograms [mean=0.5, std=1.5]
Gaussian Distribution\nN = 100
Mean: 0.7028\nStd: 1.3288
Mean: 0.7028
Theoretical N(0.5,1.5²)
Gaussian Distribution\nN = 1000
Mean: 0.4342\nStd: 1.4625
Mean: 0.4342
Theoretical N(0.5,1.5²)
Gaussian Distribution\nN = 10000
Mean: 0.5235\nStd: 1.4897
Mean: 0.5235
Theoretical N(0.5,1.5²)
Gaussian Distribution\nN = 100000
Mean: 0.5036\nStd: 1.4979
Mean: 0.5036
Theoretical N(0.5,1.5²)
Density
Value


Convergence of Histograms with Sample Size
Uniform (N=100)
Uniform (N=1000)
Uniform (N=10000)
Uniform (N=100000)
Gaussian (N=100)
Gaussian (N=1000)
Gaussian (N=10000)
Gaussian (N=100000)
Theoretical
Density
Value

히스토그램은 위와 같다.