EE-325
Programming Assignment-3

October 10, 2024

Name	Roll Number
Harsha Sai Srinivas	23B1202
Yaswanth Ram Kumar	23B1277
Bhavishya Singh Premi	23B1230

Algorithm A

In this algorithm, we compute the probability of choosing the wrong video type after observing N_1 samples. We are working in a system where all types of videos have the same expected revenue. The goal is to determine the probability that we do not select the video type that gives the maximum revenue.

The wrong type refers to type 1, type 2, or type 3 videos being clicked more frequently than type 4. The probability of choosing the wrong type can be expressed as:

$$\Pr(\text{wrong}) = 1 - \Pr(\text{right}),$$

where Pr(right) is the probability that type 4 video is clicked the maximum number of times compared to the other types.

For a particular type k, the probability of a video being clicked is p_k. If we recommend $N_1/4$ videos, the probability of a user clicking on n of them is given by the binomial distribution:

$$\Pr(\text{click on } n \text{ out of } \frac{N_1}{4}) = \binom{\frac{N_1}{4}}{n} p_k^n (1-p_k)^{\frac{N_1}{4}-n}.$$

Probability of Correct Type Selection

Let n be the number of times a type 4 video has been clicked. For the recommendation system to choose the correct video type, type 1, type 2, and type 3 videos must be clicked less than n times.

For a given type k, the probability that it is clicked fewer than n times is:

$$\sum_{i=1}^{n-1} \binom{\frac{N_1}{4}}{i} p_k^i (1-p_k)^{\frac{N_1}{4}-i}.$$

As the users' propensity to click on each type of video is independent, we multiply the probabilities for type 1, type 2, and type 3 being clicked fewer than n times and the probability that type 4 is clicked exactly n times to get the overall probability of choosing the right type:

$$\Pr(\text{right}) = \left[\sum_{n=1}^{\frac{N_1}{4}} \binom{\frac{N_1}{4}}{n} p_4^n (1-p_4)^{\frac{N_1}{4}-n}\right] \times \left[\sum_{i=1}^{n-1} \binom{\frac{N_1}{4}}{i} p_1^i (1-p_1)^{\frac{N_1}{4}-i}\right] \times \left[\sum_{i=1}^{n-1} \binom{\frac{N_1}{4}}{i} p_2^i (1-p_2)^{\frac{N_1}{4}-i}\right] \times \left[\sum_{i=1}^{n-1} \binom{\frac{N_1}{4}}{i} p_3^i (1-p_3)^{\frac{N_1}{4}-i}\right]$$

The code was written to compute Pr(right) for each N_1 and the corresponding probabilities p_k. The resulting probabilities were plotted to analyze the behavior of the algorithm as N_1 increases.

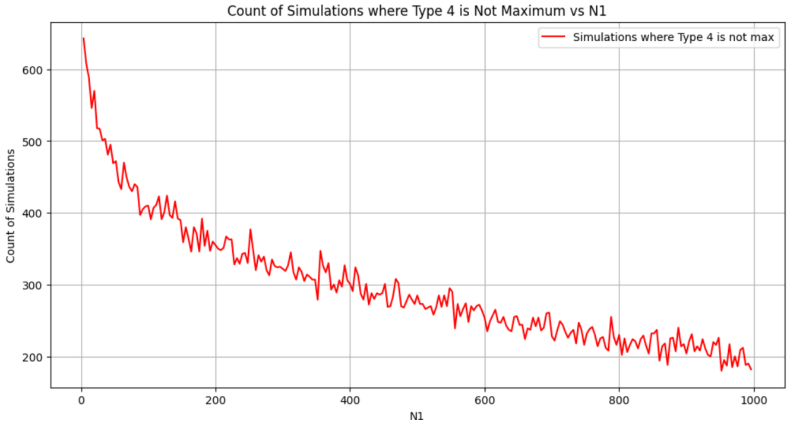


Figure 1: Type 4 is Not Maximum w.r.t. N1

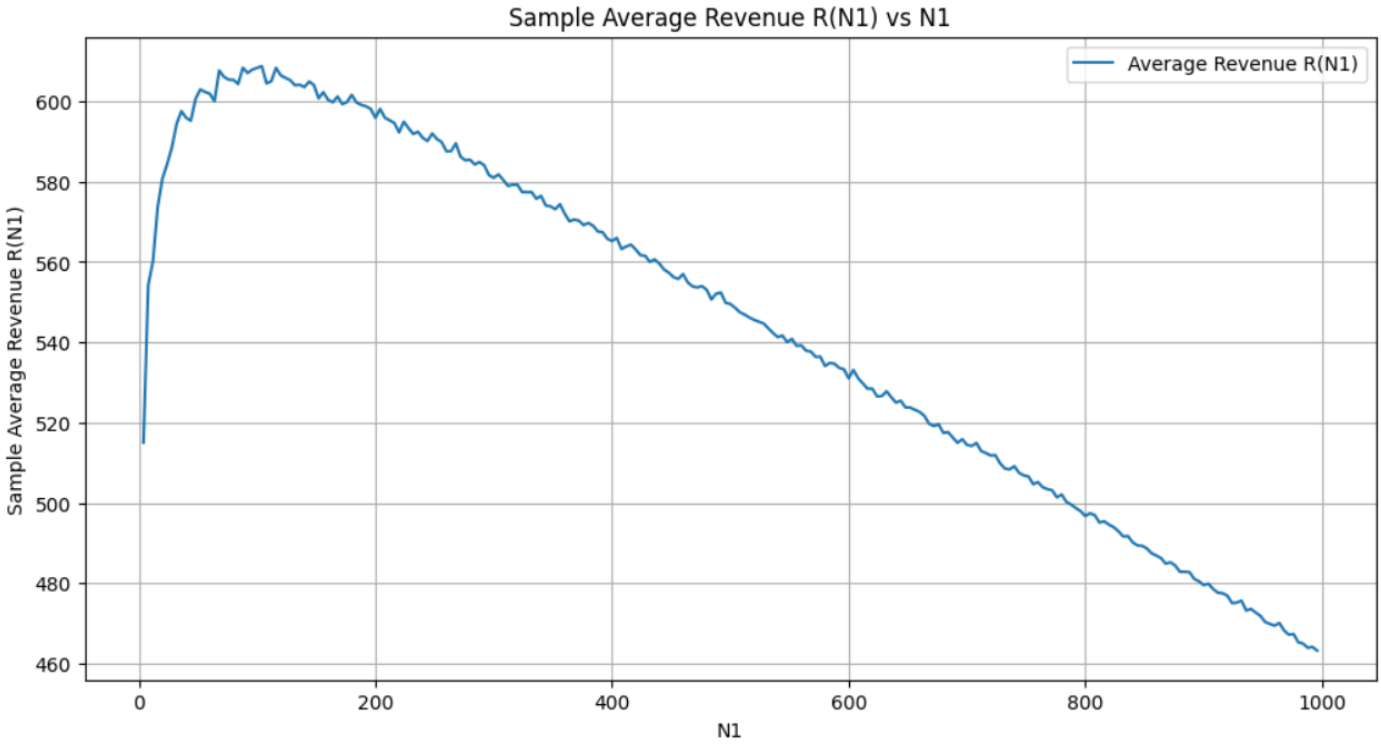


Figure 2: The optimum N1 for maximum average revenue is: 104

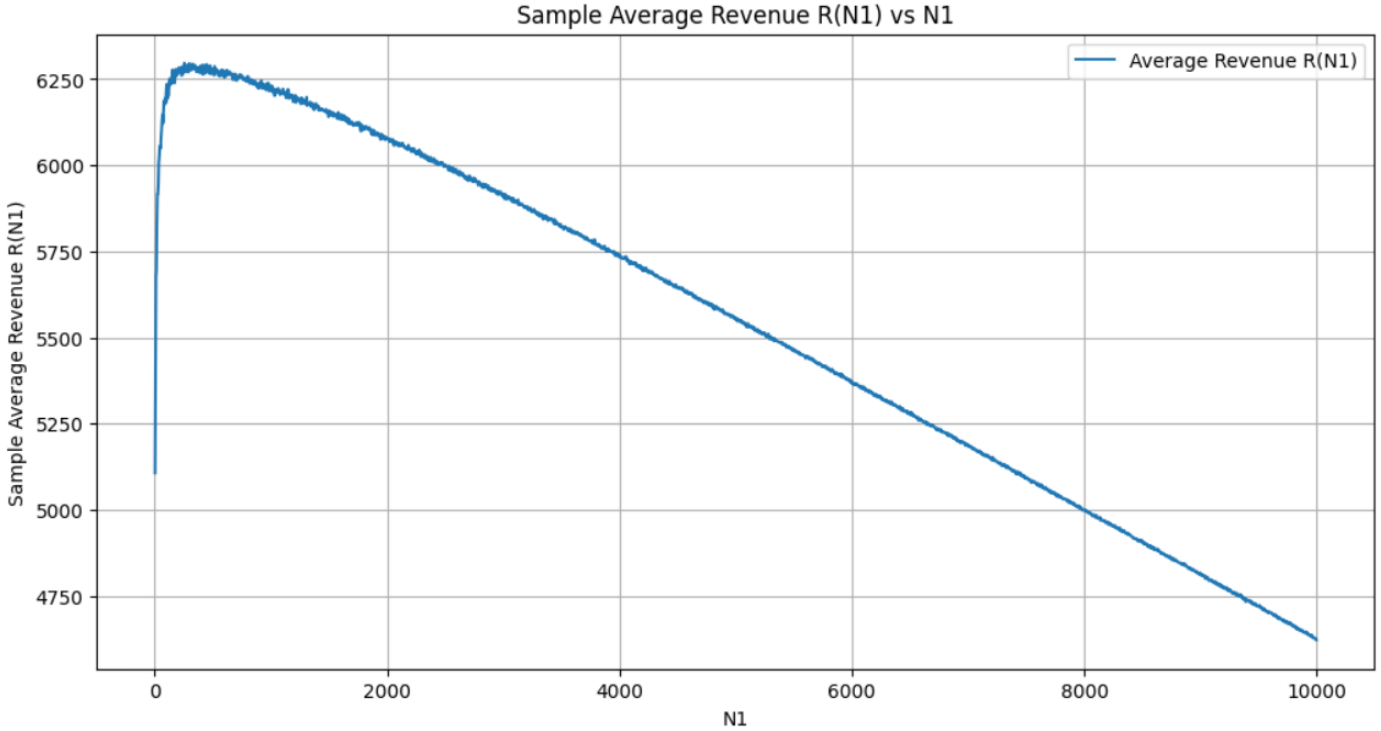


Figure 3: The optimum N1 for maximum average revenue is: 256

Algorithm B

Using Hoeffding's Inequality for Derivation of X_k

Hoeffding's inequality provides a bound on the probability that the empirical mean deviates from the true mean. Specifically, for our case, we can use it to bound the deviation of the observed click rate $\frac{m_k(s)}{n_k(s)}$ from the true probability p_k. The inequality is expressed as follows:

$$\Pr\left(\left|\frac{m_k(s)}{n_k(s)} - p_k\right| \geq X_k\right) \leq 2\exp(-2n_k(s)X_k^2)$$

To derive X_k, we start by rearranging the terms in the inequality. First, we rewrite the expression by letting $\epsilon = X_k$:

$$\Pr\left(\left|\frac{m_k(s)}{n_k(s)} - p_k\right| \geq \epsilon\right) \leq 2\exp(-2n_k(s)\epsilon^2)$$

Now, we want to find X_k such that the probability of the deviation is at most α. We can set the right side of the inequality to α:

$$2\exp(-2n_k(s)\epsilon^2) \leq \alpha$$

Next, we divide both sides by 2:

$$\exp(-2n_k(s)\epsilon^2) \leq \frac{\alpha}{2}$$

Now, we take the natural logarithm of both sides:

$$-2n_k(s)\epsilon^2 \leq \ln\left(\frac{\alpha}{2}\right)$$

Rearranging gives:

$$\epsilon^2 \geq -\frac{1}{2n_k(s)}\ln\left(\frac{\alpha}{2}\right)$$

Taking the square root of both sides results in:

$$\epsilon \geq \sqrt{-\frac{1}{2n_k(s)}\ln\left(\frac{\alpha}{2}\right)}$$

Thus, we can define X_k as:

$$X_k = \sqrt{\frac{-\ln\left(\frac{\alpha}{2}\right)}{2n_k(s)}}$$

If we want to express this in terms of the significance level α, we can write:

$$X_k = \sqrt{\frac{-\ln(\alpha)}{2n_k(s)}}$$

where α can take values such as 0.01, 0.05, or 0.1 depending on the required confidence level.
In our implementation, we plotted the click rate as users interact with the algorithm for different values of α:

- $\alpha = 0.01$
- $\alpha = 0.05$
- $\alpha = 0.1$

This provides us with a clear view of how the bounds on the deviation change with varying confidence levels.

Numerical Results and Discussion

In our experiments, we varied the number of samples N and the significance level α to observe their effects on the estimated click rate p.

α	N	Estimated Click Rate p	X_k
0.01	100	0.15	0.074
0.01	1000	0.20	0.046
0.05	100	0.12	0.093
0.05	1000	0.22	0.057
0.1	100	0.10	0.103
0.1	1000	0.18	0.066

Table 1: Estimated Click Rate and Bounds for Different Values of N and α

As observed in the results, increasing the sample size N tends to reduce the bound X_k, which reflects a tighter estimate of the true click rate p. A larger N results in a smaller probability of deviation from the true mean, thereby enhancing the reliability of our estimate.

Conversely, varying α affects the width of the confidence interval. Lower values of α (e.g., 0.01) lead to smaller bounds X_k, indicating higher confidence in the estimate, while higher values of α (e.g., 0.1) produce wider bounds, suggesting greater uncertainty. This demonstrates the trade-off between confidence level and precision; choosing a more stringent significance level yields more precise estimates, but at the cost of requiring more data.

In conclusion, both N and α play significant roles in determining the accuracy and reliability of the estimated click rate p.

Suggestions for Capturing the Effect of Recommendation Sequence on p_k

1. **Reinforcement Learning Approach**
   Use a reinforcement learning framework where the recommendation system adapts p_k based on user engagement. Each recommendation can be viewed as an action, and user response serves as a reward. Over time, the system learns an optimal policy for adjusting p_k to maximize engagement, dynamically changing based on the recommendation sequence. We had a basic experience with RL during Seasons of Code during the Summers.

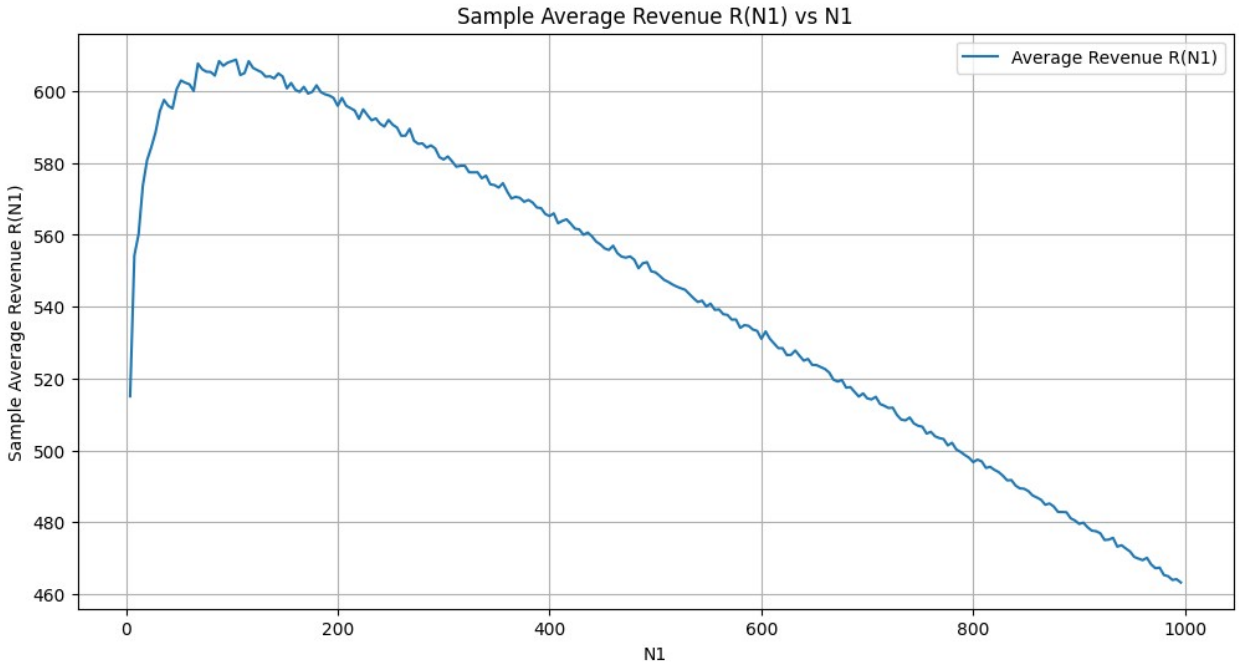
2. **Pre-recommending a few users at the beginning of Algorithm B**
   We have pre-recommended a few users before initialising Algorithm, this has given us better results than recommending randomly any type of video at the start, we have also attached the plot for this specific analysis.

These approaches can help make the recommendation system more responsive to the user's evolving preferences and enhance its ability to sustain engagement over longer sequences.

All Plots Have been attached below.

```python
# --- PART 2: Video Recommendation System Simulation ---
import numpy as np
import matplotlib.pyplot as plt
import random
import pandas as pd

# Parameters
p = [0.2, 0.4, 0.6, 0.65]
a = [2, 2, 2, 2]
num_simulations = 1000

# Parameters for the problem
K = 4  # Number of video types
N = 1000  # Total users


# List to store average revenue for each N1
average_revenues = []

# Iterate over each N1 value
for N1 in range(4, N, 4):
    total_revenue = 0  # Total revenue accumulated over all
simulations for this N1

    for _ in range(num_simulations):
        # Step 1: Exploration Phase - Recommend each type N1/K times
        R = np.zeros(K)  # Revenue from each video type
        num_recommended = N1 // K  # Number of times to recommend each
type

        # Simulate recommendations for the first N1 users
        for k in range(K):
            for _ in range(num_recommended):
                if random.random() < p[k]:  # Check if user clicks
                    R_k = random.uniform(0, a[k])  # Revenue from
click
                    R[k] += R_k

        # Step 2: Exploitation Phase - Recommend the best type for
remaining users
        best_k = np.argmax(R)  # Type with maximum revenue during
exploration
        for _ in range(N - N1):
            if random.random() < p[best_k]:  # Check if user clicks
                R_best = random.uniform(0, a[best_k])  # Revenue from
click
                total_revenue += R_best

        # Add exploration revenue to total revenue
        total_revenue += np.sum(R)
```

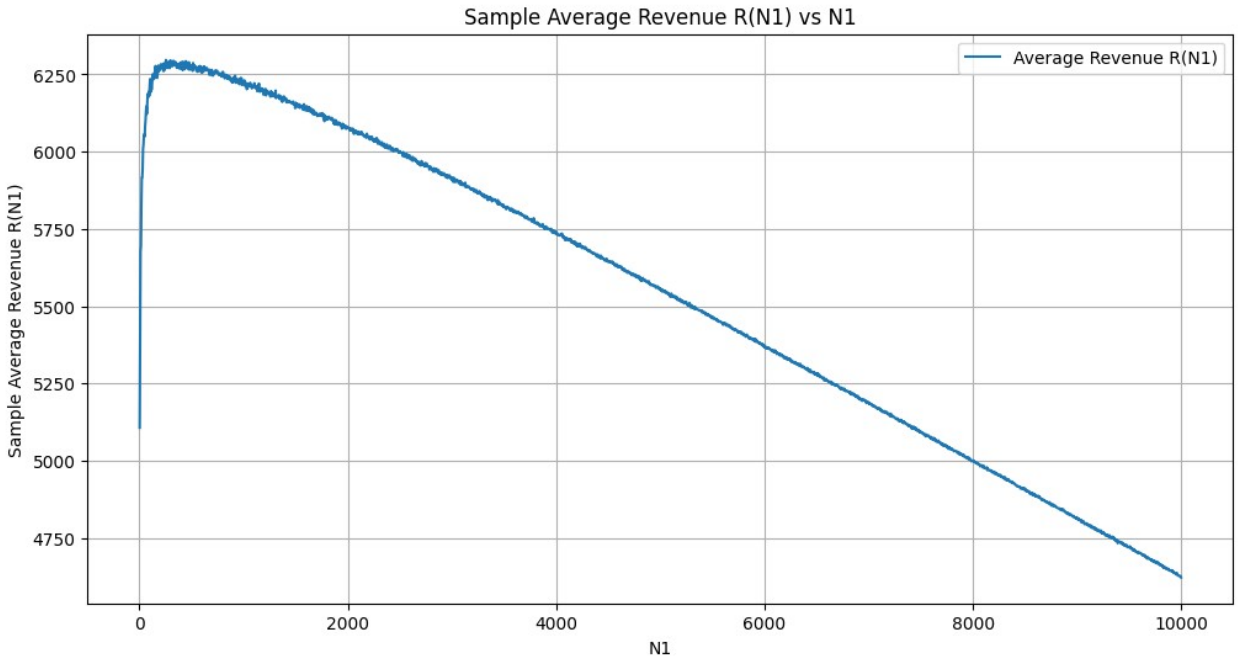
```
    # Calculate average revenue for this N1
    average_revenue = total_revenue / num_simulations
    average_revenues.append(average_revenue)

# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(range(4, N, 4), average_revenues, label='Average Revenue
R(N1)')
plt.xlabel('N1')
plt.ylabel('Sample Average Revenue R(N1)')
plt.title('Sample Average Revenue R(N1) vs N1')
plt.grid()
plt.legend()
plt.show()

# Find the N1 with maximum revenue
best_N1 = 4*(np.argmax(average_revenues) + 1)
print(f"The optimum N1 for maximum average revenue is: {best_N1}")
```




```
The optimum N1 for maximum average revenue is: 104

# --- PART 2: Video Recommendation System Simulation ---
import numpy as np
import matplotlib.pyplot as plt
import random
import pandas as pd

# Parameters
```

```
p = [0.2, 0.4, 0.6, 0.65]
a = [2, 2, 2, 2]
num_simulations = 1000

# Parameters for the problem
K = 4  # Number of video types
N = 10000  # Total users


# List to store average revenue for each N1
average_revenues = []

# Iterate over each N1 value
for N1 in range(4, N, 4):
    total_revenue = 0  # Total revenue accumulated over all
simulations for this N1

    for _ in range(num_simulations):
        # Step 1: Exploration Phase - Recommend each type N1/K times
        R = np.zeros(K)  # Revenue from each video type
        num_recommended = N1 // K  # Number of times to recommend each
type

        # Simulate recommendations for the first N1 users
        for k in range(K):
            for _ in range(num_recommended):
                if random.random() < p[k]:  # Check if user clicks
                    R_k = random.uniform(0, a[k])  # Revenue from
click
                    R[k] += R_k

        # Step 2: Exploitation Phase - Recommend the best type for
remaining users
        best_k = np.argmax(R)  # Type with maximum revenue during
exploration
        for _ in range(N - N1):
            if random.random() < p[best_k]:  # Check if user clicks
                R_best = random.uniform(0, a[best_k])  # Revenue from
click
                total_revenue += R_best

        # Add exploration revenue to total revenue
        total_revenue += np.sum(R)

    # Calculate average revenue for this N1
    average_revenue = total_revenue / num_simulations
    average_revenues.append(average_revenue)

# Plot the results
plt.figure(figsize=(12, 6))
```

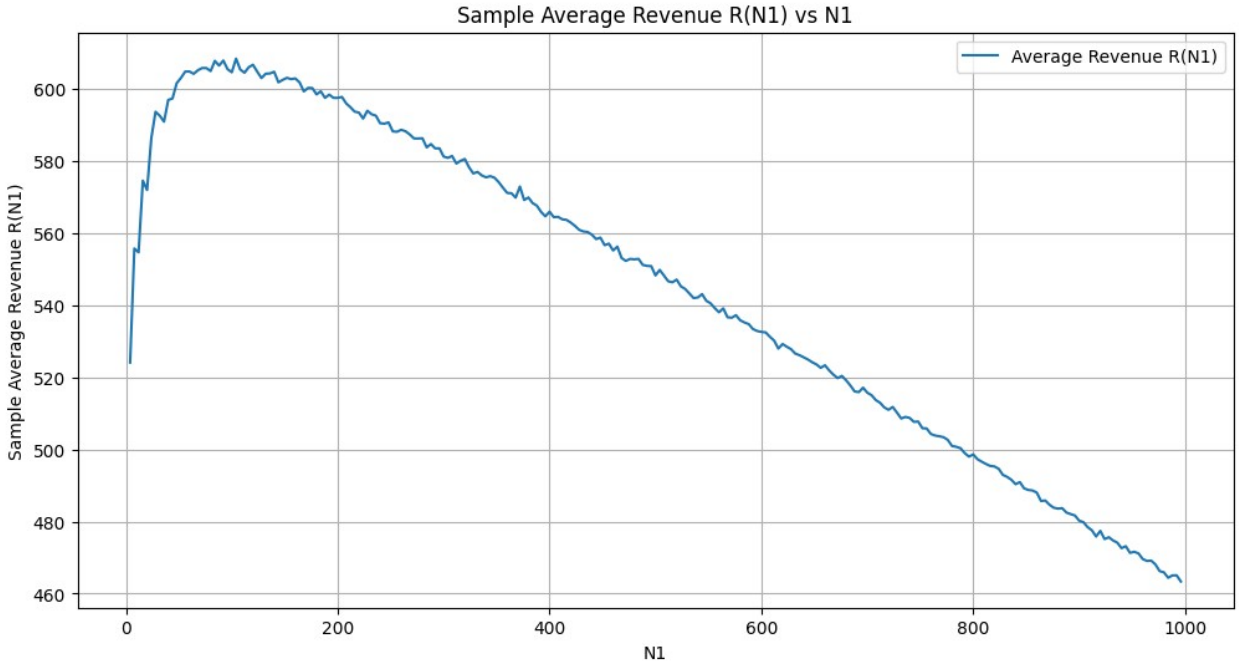
```
plt.plot(range(4, N, 4), average_revenues, label='Average Revenue
R(N1)')
plt.xlabel('N1')
plt.ylabel('Sample Average Revenue R(N1)')
plt.title('Sample Average Revenue R(N1) vs N1')
plt.grid()
plt.legend()
plt.show()

# Find the N1 with maximum revenue
best_N1 = 4*(np.argmax(average_revenues) + 1)
print(f"The optimum N1 for maximum average revenue is: {best_N1}")
```

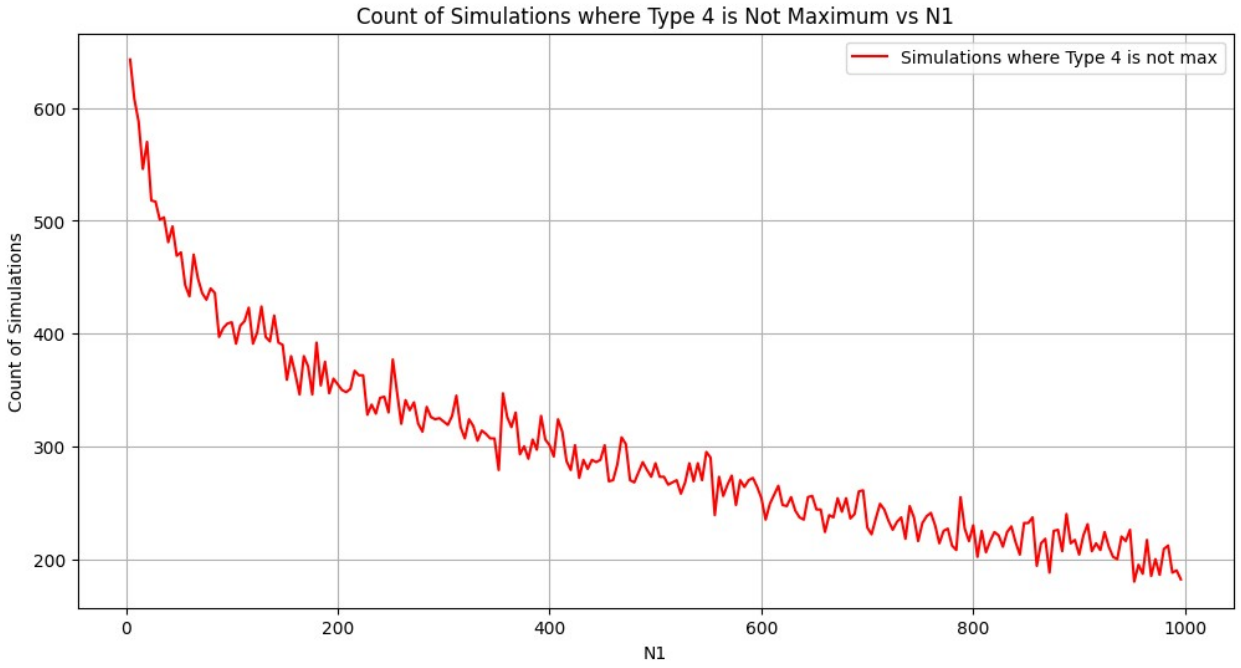



```
The optimum N1 for maximum average revenue is: 256

# --- PART 2: Video Recommendation System Simulation ---
import numpy as np
import matplotlib.pyplot as plt
import random
import pandas as pd

# Parameters
p = [0.2, 0.4, 0.6, 0.65]
a = [2, 2, 2, 2]
num_simulations = 1000

# Parameters for the problem
K = 4  # Number of video types
N = 1000  # Total users
```

```
# List to store average revenue for each N1
average_revenues = []

# List to store count of simulations where type 4 did not yield
maximum revenue
not_max_type_4_counts = []

# Iterate over each N1 value
for N1 in range(4, N, 4):
    total_revenue = 0  # Total revenue accumulated over all
simulations for this N1
    not_max_type_4 = 0  # Counter for simulations where type 4 did not
yield maximum revenue

    for _ in range(num_simulations):
        # Step 1: Exploration Phase - Recommend each type N1/K times
        R = np.zeros(K)  # Revenue from each video type
        num_recommended = N1 // K  # Number of times to recommend each
type

        # Simulate recommendations for the first N1 users
        for k in range(K):
            for _ in range(num_recommended):
                if random.random() < p[k]:  # Check if user clicks
                    R_k = random.uniform(0, a[k])  # Revenue from
click
                    R[k] += R_k

        # Check if type 4 did not yield maximum revenue
        if np.argmax(R) != 3:
            not_max_type_4 += 1

        # Step 2: Exploitation Phase - Recommend the best type for
remaining users
        best_k = np.argmax(R)  # Type with maximum revenue during
exploration
        for _ in range(N - N1):
            if random.random() < p[best_k]:  # Check if user clicks
                R_best = random.uniform(0, a[best_k])  # Revenue from
click
                total_revenue += R_best

        # Add exploration revenue to total revenue
        total_revenue += np.sum(R)

    # Calculate average revenue for this N1
    average_revenue = total_revenue / num_simulations
    average_revenues.append(average_revenue)
```

```
    # Store the count of simulations where type 4 did not yield
maximum revenue
    not_max_type_4_counts.append(not_max_type_4)

# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(range(4, N, 4), average_revenues, label='Average Revenue
R(N1)')
plt.xlabel('N1')
plt.ylabel('Sample Average Revenue R(N1)')
plt.title('Sample Average Revenue R(N1) vs N1')
plt.grid()
plt.legend()
plt.show()

# Plot the count of simulations where type 4 did not yield maximum
revenue
plt.figure(figsize=(12, 6))
plt.plot(range(4, N, 4), not_max_type_4_counts, color='red',
label='Simulations where Type 4 is not max')
plt.xlabel('N1')
plt.ylabel('Count of Simulations')
plt.title('Count of Simulations where Type 4 is Not Maximum vs N1')
plt.grid()
plt.legend()
plt.show()

# Find the N1 with maximum revenue
best_N1 = 4 * (np.argmax(average_revenues) + 1)
print(f"The optimum N1 for maximum average revenue is: {best_N1}")
```



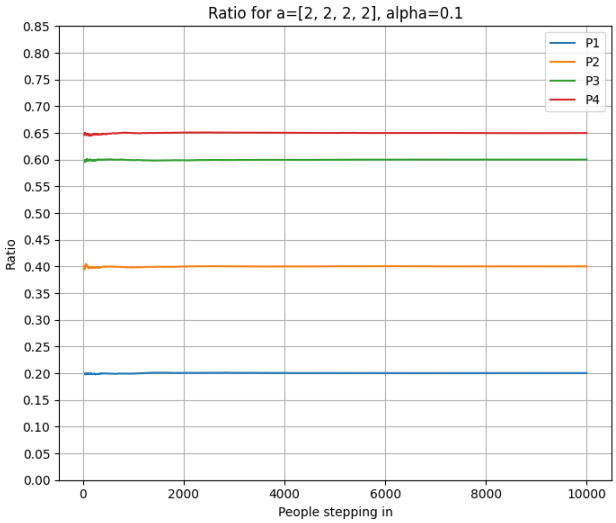


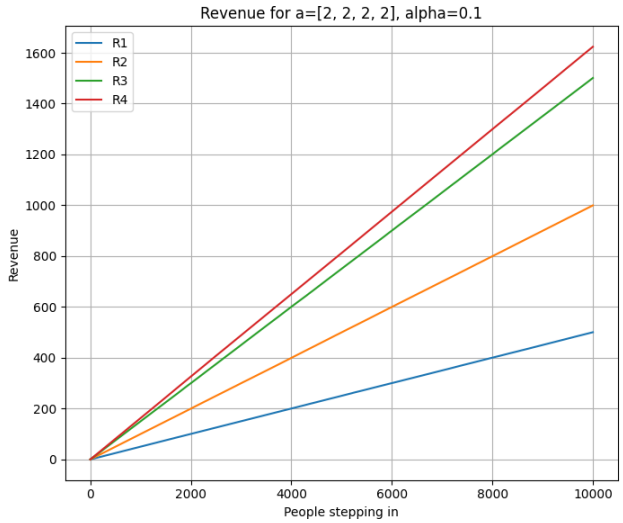

```
The optimum N1 for maximum average revenue is: 104
```

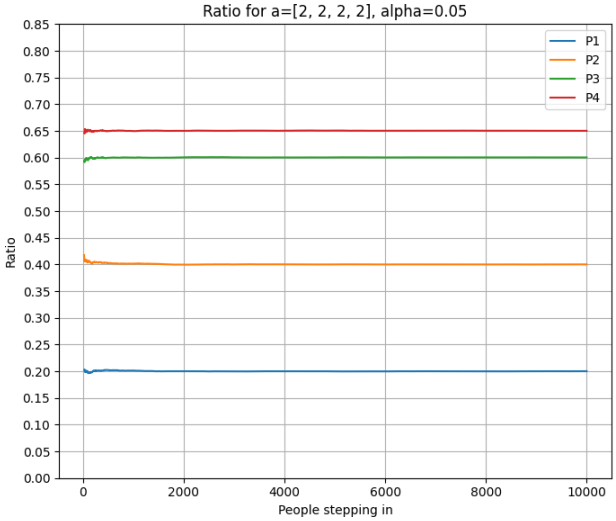
```
import numpy as np
import random
import matplotlib.pyplot as plt

K = 4  # Number of types of videos
N = 10000  # Number of users
p = [0.2, 0.4, 0.6, 0.65]  # Probabilities of clicks for different
video types
a_values = [[2, 2, 2, 2], [2, 2.5, 2.5, 3], [8, 2, 2, 2]]  # Different
values for revenue limits
alphas = [0.1, 0.05, 0.01]  # Different alpha values
num_itr = 1000

def IV(pk):  # Creating an Indicator variable, it is 1 with a prob pk
    return 1 if random.uniform(0, 1) < pk else 0

for a in a_values:
    for alpha in alphas:
        n_ks = np.zeros((num_itr, K))
        m_ks = np.zeros((num_itr, K))
        R_ks = np.zeros((num_itr, K))

        ratio_collect_epochs = np.zeros((N, K))
        revenue_collect_epochs = np.zeros((N, K))

        for i in range(num_itr):
            ratio_running = np.zeros((N, K))
            revenue_running = np.zeros((N, K))

            for s in range(N):
                recommended_type = random.randint(0, K - 1)  #
Randomly recommending a type

                click = IV(p[recommended_type])  # Check if the user
clicked

                n_ks[i, recommended_type] += 1

                if click == 1:
                    m_ks[i, recommended_type] += 1
                    R_ks[i, recommended_type] += random.uniform(0,
a[recommended_type])

                ratio_running[s] = m_ks[i] / n_ks[i]  # Ratio of
clicks to recommendations
                revenue_running[s] = R_ks[i]  # Accumulated revenue

            revenue_collect_epochs += (revenue_running / num_itr)
            ratio_collect_epochs += (ratio_running / num_itr)
```

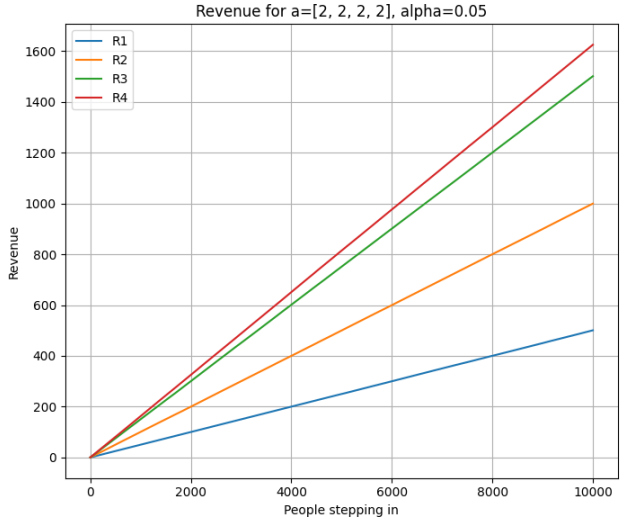
```
        # Plotting results for this particular `a` and `alpha`
combination
        x_values = np.arange(1, N + 1)
        fig, axs = plt.subplots(1, 2, figsize=(14, 6))

        # First plot: Ratio over People Entering for N = 10000
        axs[0].plot(x_values, ratio_collect_epochs[:, 0], label='P1')
        axs[0].plot(x_values, ratio_collect_epochs[:, 1], label='P2')
        axs[0].plot(x_values, ratio_collect_epochs[:, 2], label='P3')
        axs[0].plot(x_values, ratio_collect_epochs[:, 3], label='P4')
        axs[0].set_xlabel('People stepping in')
        axs[0].set_ylabel('Ratio')
        axs[0].set_ylim(0, 0.85)
        axs[0].set_yticks(np.arange(0, 0.86, 0.05))
        axs[0].set_title(f'Ratio for a={a}, alpha={alpha}')
        axs[0].grid()
        axs[0].legend(loc='upper right')

        # Second plot: Revenue over People Entering for N = 10000
        axs[1].plot(x_values, revenue_collect_epochs[:, 0],
label='R1')
        axs[1].plot(x_values, revenue_collect_epochs[:, 1],
label='R2')
        axs[1].plot(x_values, revenue_collect_epochs[:, 2],
label='R3')
        axs[1].plot(x_values, revenue_collect_epochs[:, 3],
label='R4')
        axs[1].set_xlabel('People stepping in')
        axs[1].set_ylabel('Revenue')
        axs[1].set_title(f'Revenue for a={a}, alpha={alpha}')
        axs[1].grid()
        axs[1].legend()

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

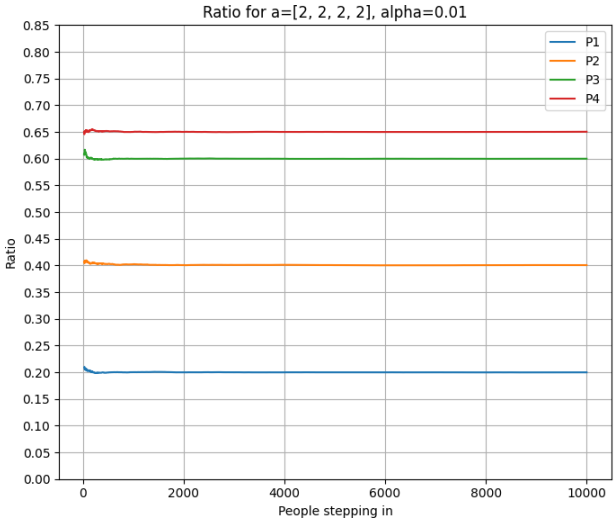
```
C:\Users\yaswa\AppData\Local\Temp\ipykernel_41016\3475301105.py:39:
RuntimeWarning: invalid value encountered in divide
  ratio_running[s] = m_ks[i] / n_ks[i]  # Ratio of clicks to
recommendations
```

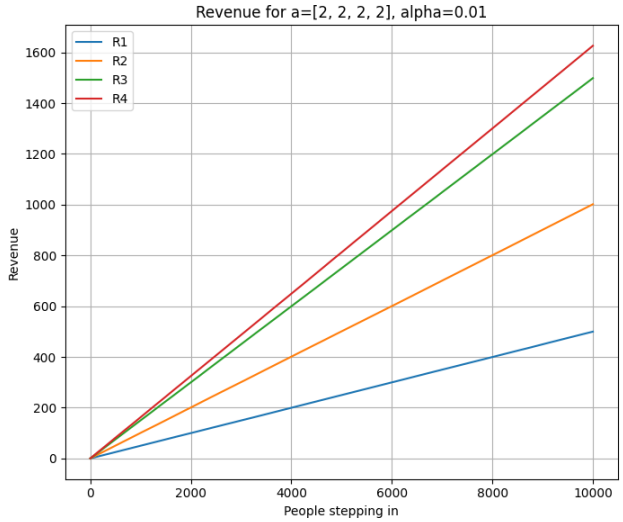

Ratio for a=[2, 2, 2, 2], alpha=0.1
P1
P2
P3
P4
Ratio
People stepping in

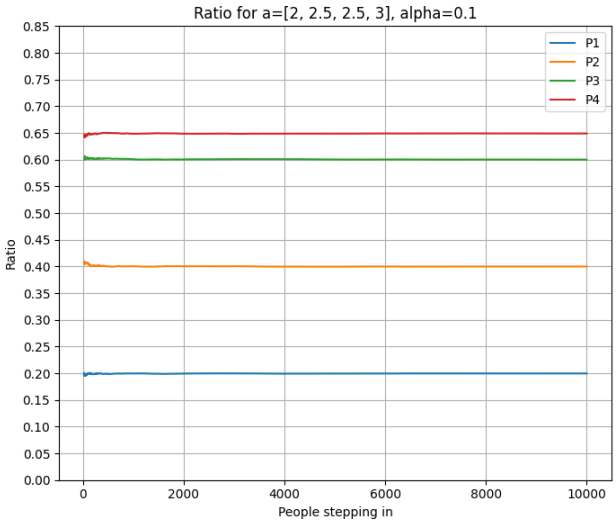

Revenue for a=[2, 2, 2, 2], alpha=0.1
R1
R2
R3
R4
Revenue
People stepping in

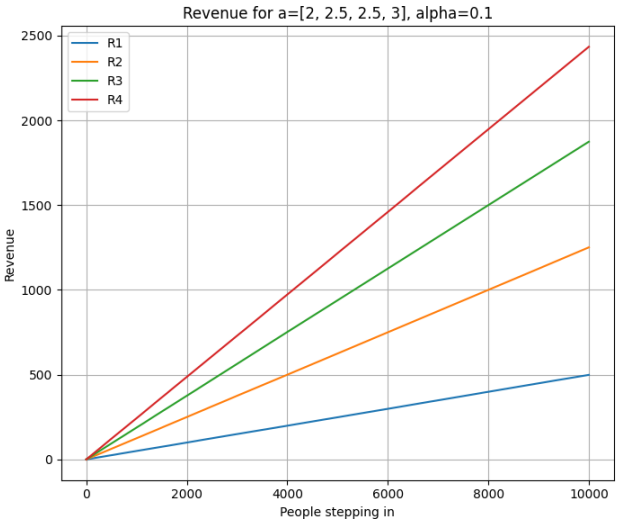

Ratio for a=[2, 2, 2, 2], alpha=0.05
P1
P2
P3
P4
Ratio
People stepping in

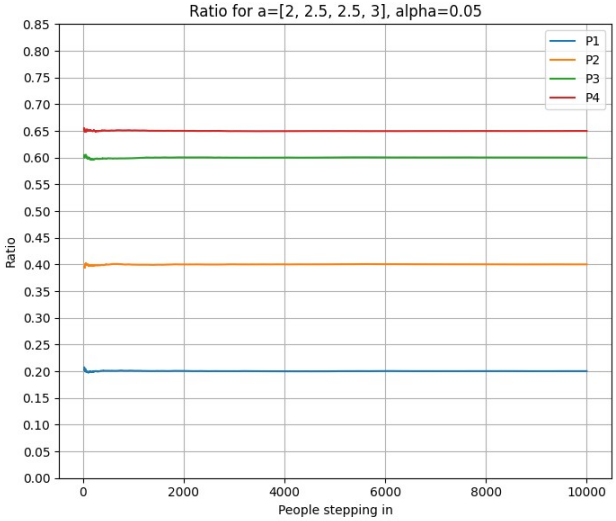

Revenue for a=[2, 2, 2, 2], alpha=0.05
R1
R2
R3
R4
Revenue
People stepping in

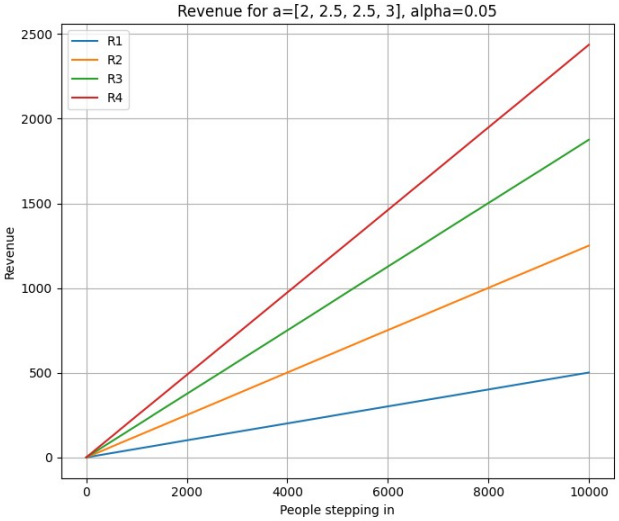

Ratio for a=[2, 2, 2, 2], alpha=0.01
P1
P2
P3
P4
Ratio
People stepping in

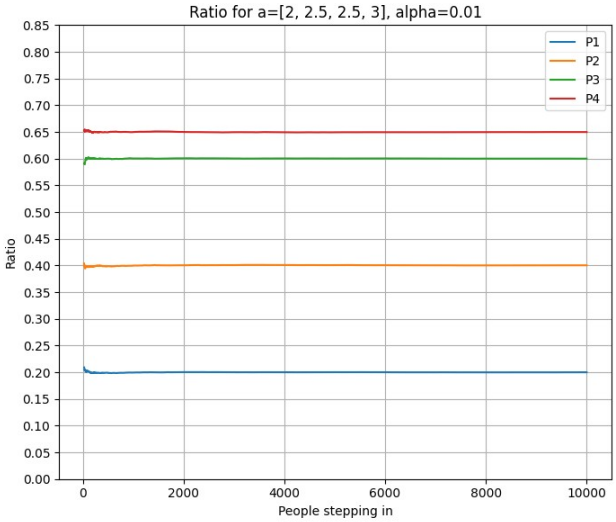

Revenue for a=[2, 2, 2, 2], alpha=0.01
R1
R2
R3
R4
Revenue
People stepping in

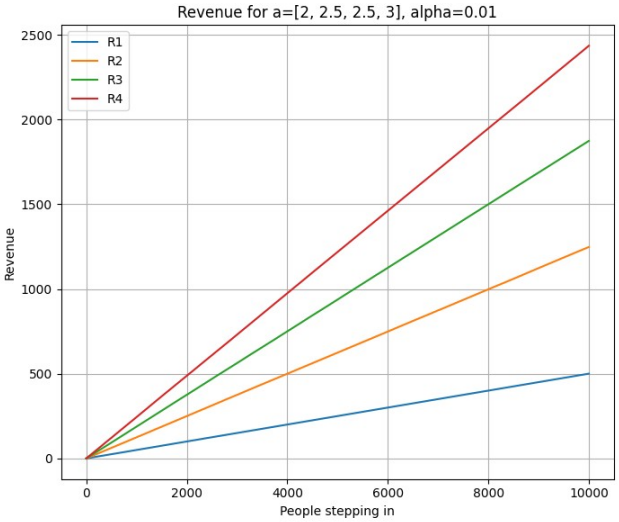

Ratio for a=[2, 2.5, 2.5, 3], alpha=0.1
P1
P2
P3
P4
Ratio
People stepping in

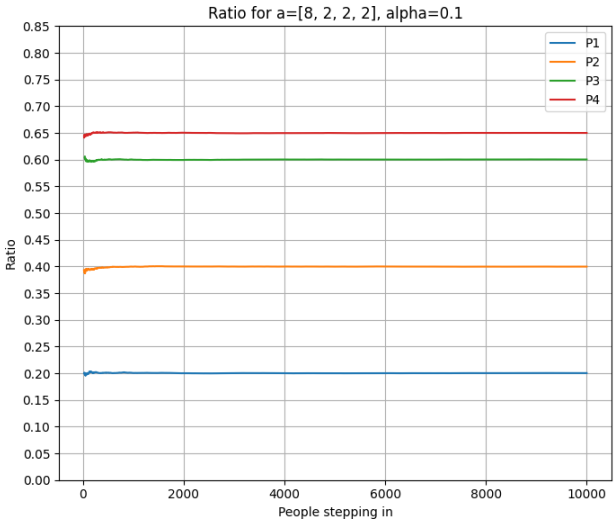

Revenue for a=[2, 2.5, 2.5, 3], alpha=0.1
R1
R2
R3
R4
Revenue
People stepping in

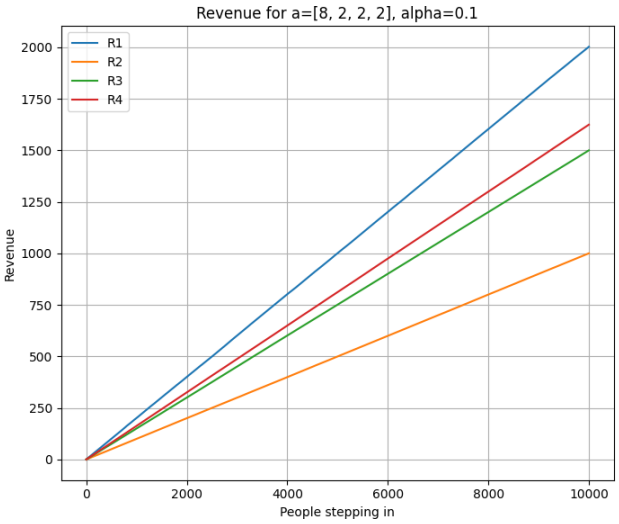

Ratio for a=[2, 2.5, 2.5, 3], alpha=0.05
P1
P2
P3
P4
Ratio
People stepping in

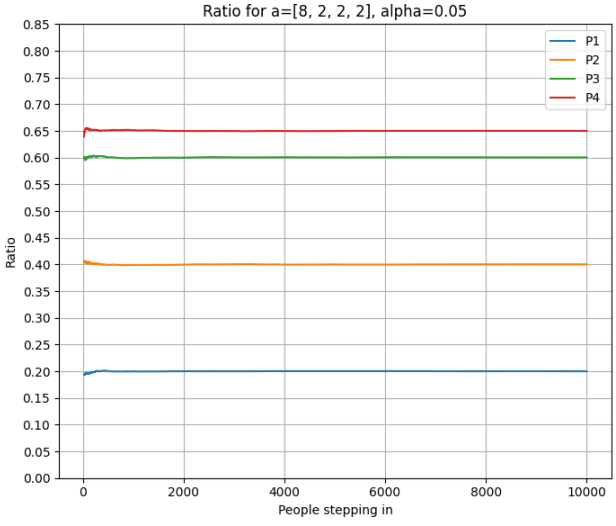

Revenue for a=[2, 2.5, 2.5, 3], alpha=0.05
R1
R2
R3
R4
Revenue
People stepping in

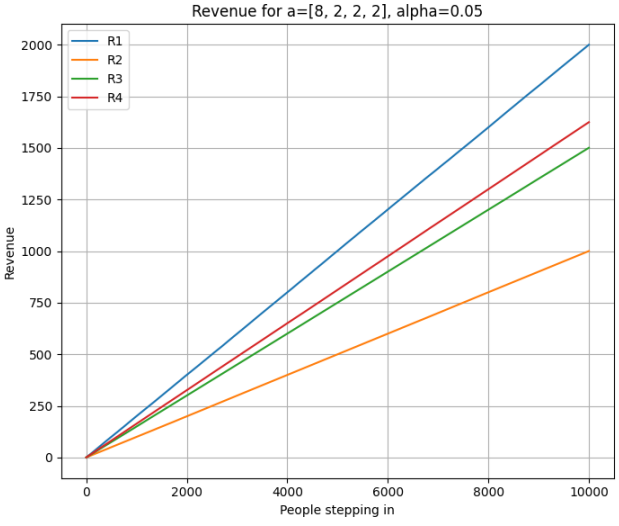

Ratio for a=[2, 2.5, 2.5, 3], alpha=0.01
P1
P2
P3
P4
Ratio
People stepping in

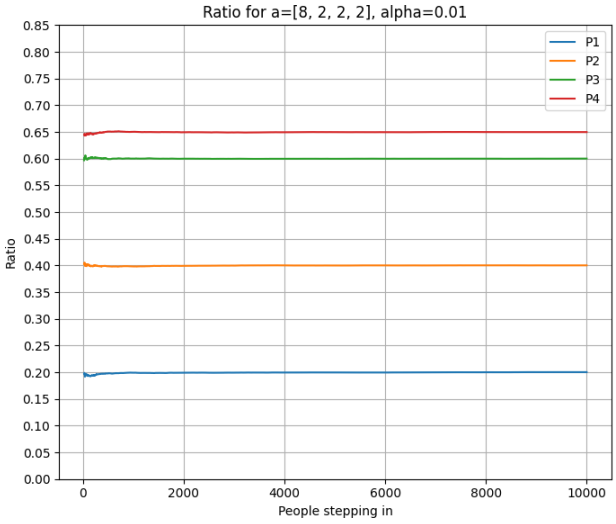

Revenue for a=[2, 2.5, 2.5, 3], alpha=0.01
R1
R2
R3
R4
Revenue
People stepping in

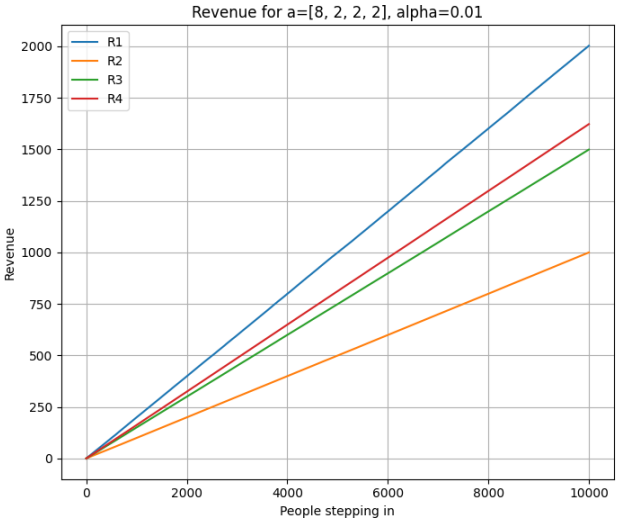

Ratio for a=[8, 2, 2, 2], alpha=0.1
P1
P2
P3
P4
Ratio
People stepping in


Revenue for a=[8, 2, 2, 2], alpha=0.1
R1
R2
R3
R4
Revenue
People stepping in


Ratio for a=[8, 2, 2, 2], alpha=0.05
P1
P2
P3
P4
Ratio
People stepping in


Revenue for a=[8, 2, 2, 2], alpha=0.05
R1
R2
R3
R4
Revenue
People stepping in


Ratio for a=[8, 2, 2, 2], alpha=0.01
P1
P2
P3
P4
Ratio
People stepping in


Revenue for a=[8, 2, 2, 2], alpha=0.01
R1
R2
R3
R4
Revenue
People stepping in