EE 325: Probability and Random Processes Programming Assignment 3

GitHub repo: https://github.com/rookie-apoorv/Programming-Assignment-3.git

The Three Systems:

System	K	p_k	a_k
1.	4	$p_1 = 0.2, p_2 = 0.4, p_3 = 0.6, p_4 = 0.65$	$a_1 = a_2 = a_3 = a_4 = 2$
2.	4	$p_1 = 0.2, p_2 = 0.4, p_3 = 0.6, p_4 = 0.65$	$a_1 = 2,\ a_2 = a_3 = 2.5,\ a_4 = 3$
3.	4	$p_1 = 0.2, p_2 = 0.4, p_3 = 0.6, p_4 = 0.65$	$a_1 = 8,\ a_2 = a_3 = a_4 = 2$

Algorithm A

Code for Algorithm A with N = 1000:

https://colab.research.google.com/drive/1v2U3mMcfHddfUpgYRJ60tVOpoBAxDIdd?usp=sharing

Code for Algorithm A with N = 10000:

https://colab.research.google.com/drive/15Rpff_qMrp4qI8SYVf38_nF04S3efYnv?usp=sharing

Code for Algorithm A with N = 10000 using central limit theorem:

https://colab.research.google.com/drive/1q81wnOfAvhQFplKjSdZkxCQuNiM7AQke?usp=sharing

Details of Algorithm A

- We select N1 people, and recommend N1/K people each k type video and observe their behavior.
- For the next N-N1 people we select the best video type from this exploratory phase.
- The best video is taken to be the one with the most number of clicks out of N1/K recommendations.
- We now formulate the probability of choosing the wrong type video after recommending N1 people.
- The correct video type for system 1 is the one which yields the maximum expected revenue.
- The expected revenue is

$$\text{E(Revenue)} = p_k \frac{a_k}{2}$$

- This is maximum for video type 4 in system 1.
- So

P(Selecting wrong video type) = P (selecting type 1,2,3 after N1 recommendation)

= 1 – P (Selecting type 4 video after N1 recommendation)

= 1 – P(number of clicks of type 4 is greater than number of clicks of 1,2,3)

As all the above events are independent this probability can be written as, let nk be number of clicks of video type k out of N1/K recommendations

1 - $\sum_{i=1}^{N_1/K} P$(n_4 = i) * P(n_1 < i) * P (n_2 < i) * P (n_3 < i)

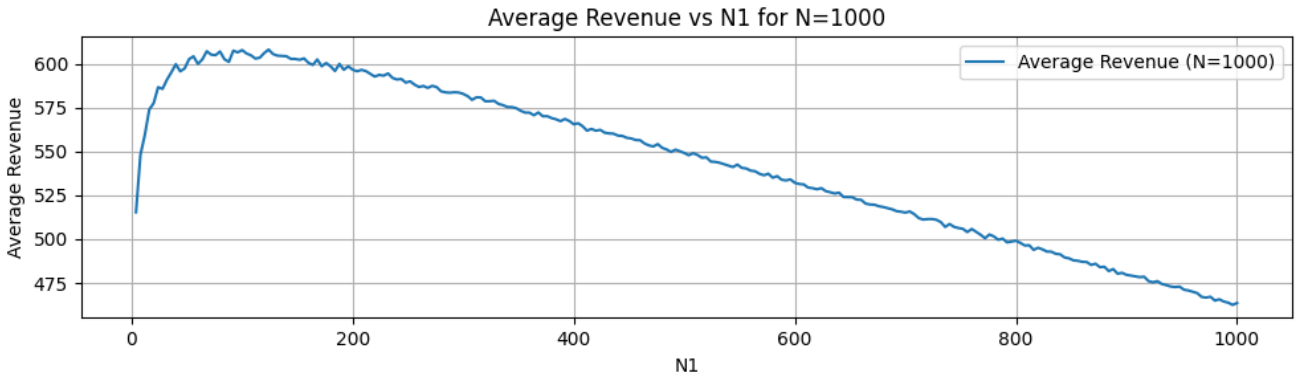
The number of clicks follows binomial distribution with N1/K trials and success probability pk for each k type video.

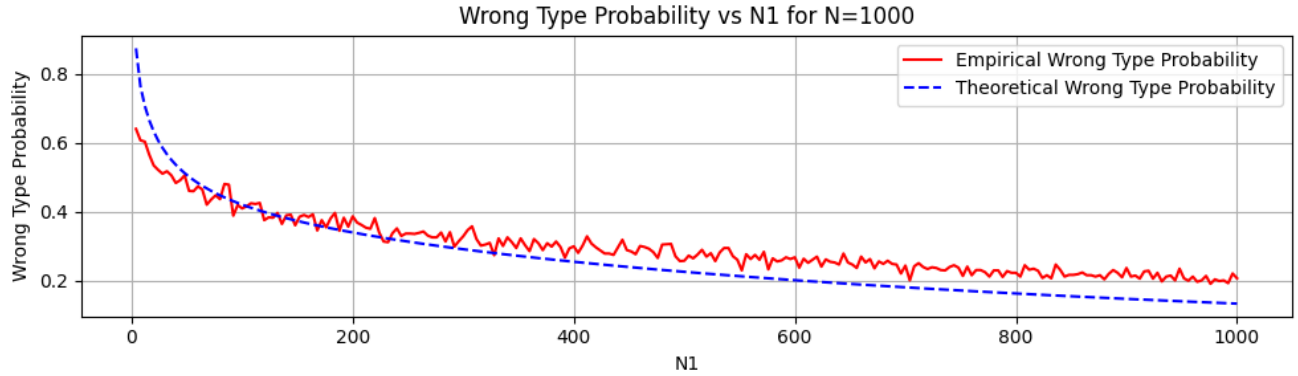
P(n_4 = i) = PMF of binomial distribution at i and P(n_k < i) = CDF of binomial at i.

- Placing values we get

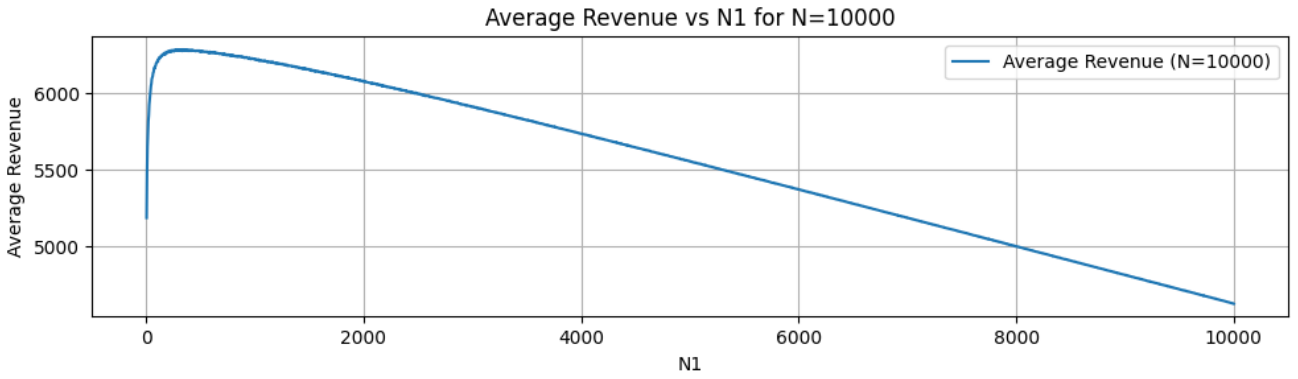
$$1 - \sum_{i=1}^{\frac{N_1}{K}} \binom{\frac{N_1}{K}}{i} (p_4^i)\,(1-p_4)^{\frac{N_1}{K}-i} \left(\sum_{n=0}^{i-1} \binom{\frac{N_1}{K}}{n} (p_1^n)\,(1-p_1)^{\frac{N_1}{K}-n}\right) \left(\sum_{n=0}^{i-1} \binom{\frac{N_1}{K}}{n} (p_2^n)\,(1-p_2)^{\frac{N_1}{K}-n}\right) \left(\sum_{n=0}^{i-1} \binom{\frac{N_1}{K}}{n} (p_3^n)\,(1-p_3)^{\frac{N_1}{K}-n}\right)$$

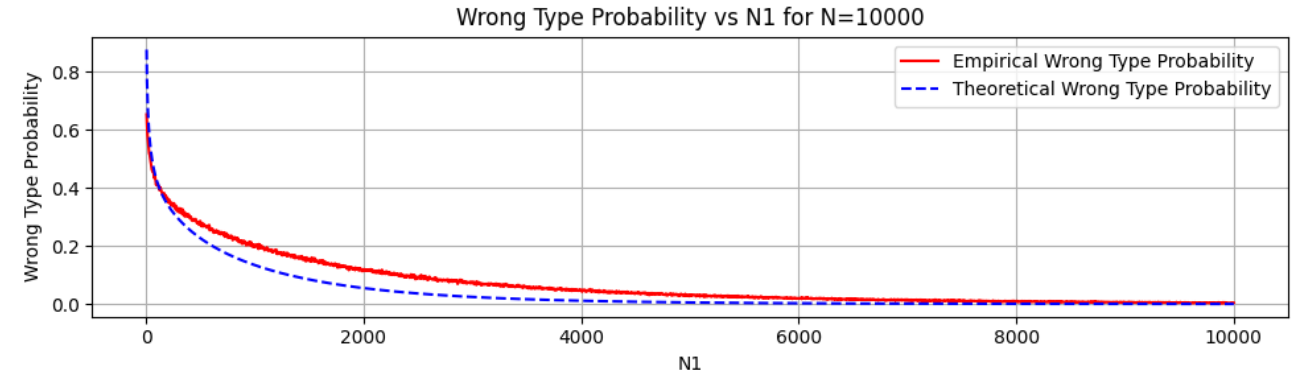
N = 1000 Graph :





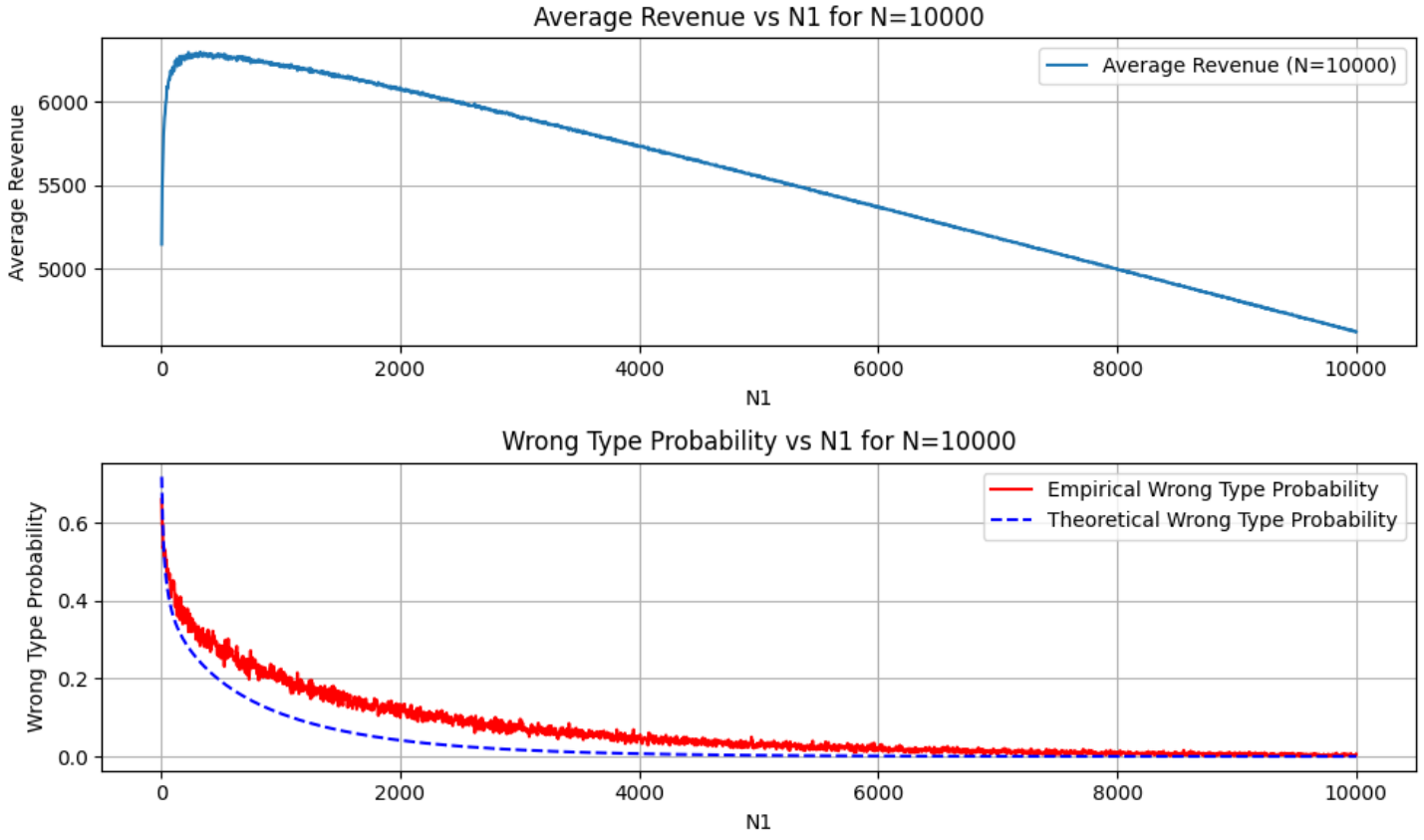
N = 10,000 graph :





N = 10000 using the central limit theorem:

1. **Central Limit Theorem (CLT) Approximation:**
    - This function assumes the revenue distribution for each type, and after enough iterations, it can be approximated as Gaussian using the CLT. This is valid when N1 is large enough, as the sum of revenues will converge to a normal distribution due to the law of large numbers.
    - This is good for N = 10,000 cases since the number of iterations is large enough.
    - The variance is calculated as $(p*a^2)/12$, the variance for a uniform distribution scaled by the click probability.
    - This improves the speed of the algorithm by around $1/4^{th}$ of the runtime
    - The approximation for the theoretical probability holds for this case and is very close to the probability calculated earlier.



Algorithm B:

Summary of the algorithm :

- For each s from 0 to N, recommend type k video with maximum UCB value at the moment.
- Increment nk for the chosen video type.
- Increment mk with a probability of pk.
- Increment Rk (Revenue earned from type k videos) randomly any value between 0 and ak if clicked and 0 if not clicked.
- Do this 1000 times, and calculate the avg value of nk, mk and Rk for each value of s.
- Plot the average values for each k as a function of s.

Calculating UCBk:

- Using Hoeffding's lemma

$$P\left(p_k - \frac{m_k}{n_k} \geq y\right) \leq e^{-2y^2 n_k}$$

$$P\left(p_k \leq \frac{m_k}{n_k} + y\right) \geq 1 - e^{-2y^2 n_k}$$

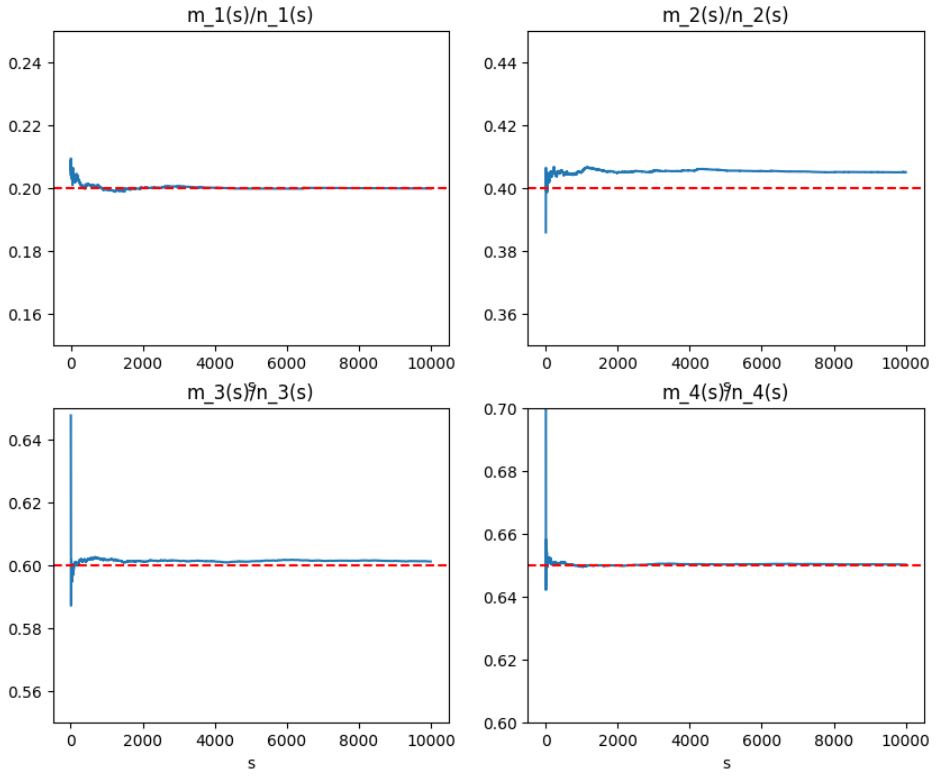
$$1 - e^{-2y^2 n_k} = 1 - \alpha$$

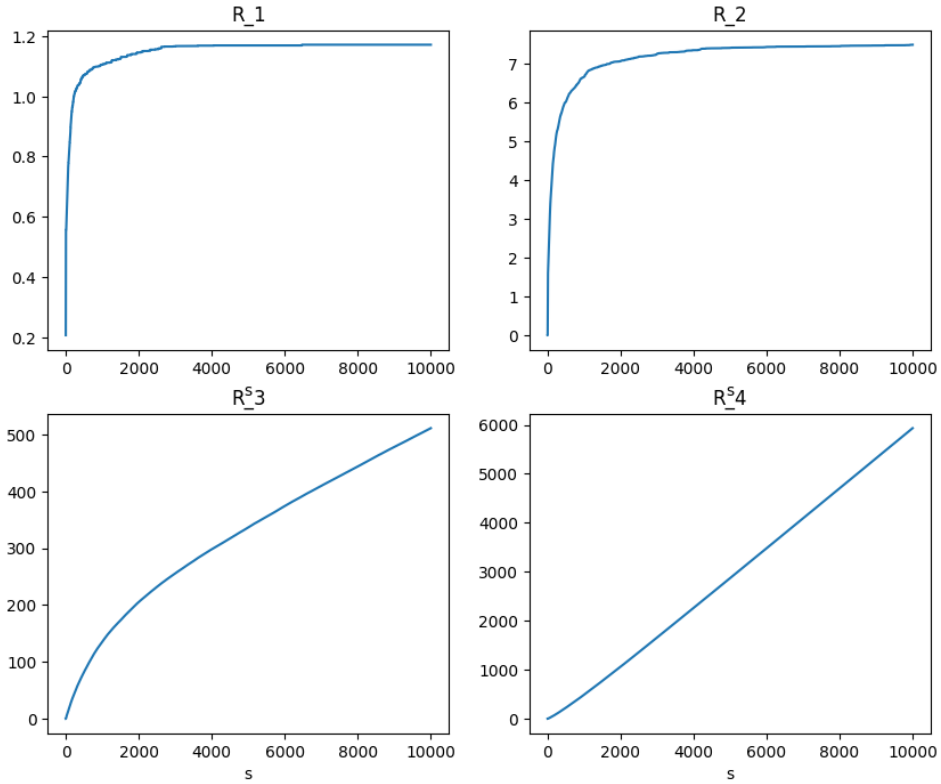
$$y = \sqrt{\frac{\log(1/\alpha)}{2n_k}}$$

$$UCB = \frac{m_k}{n_k} + \sqrt{\frac{\log(1/\alpha)}{2n_k}}$$

PLOTS: **For N = 10,000**

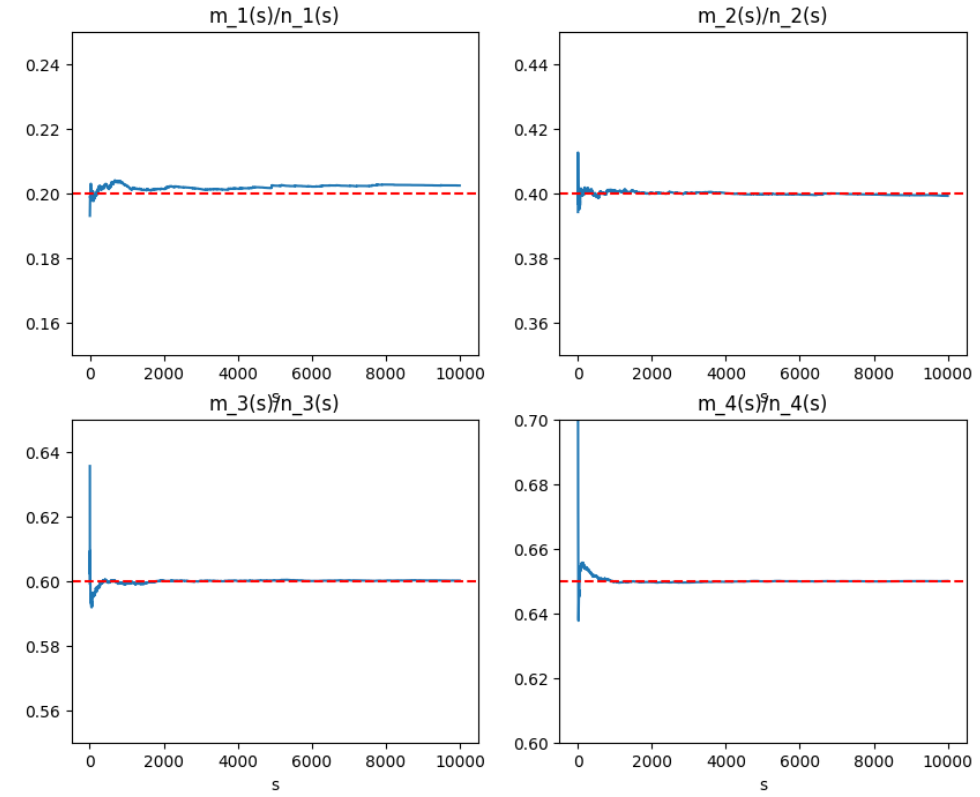
- <u>Alpha = 0.1</u>

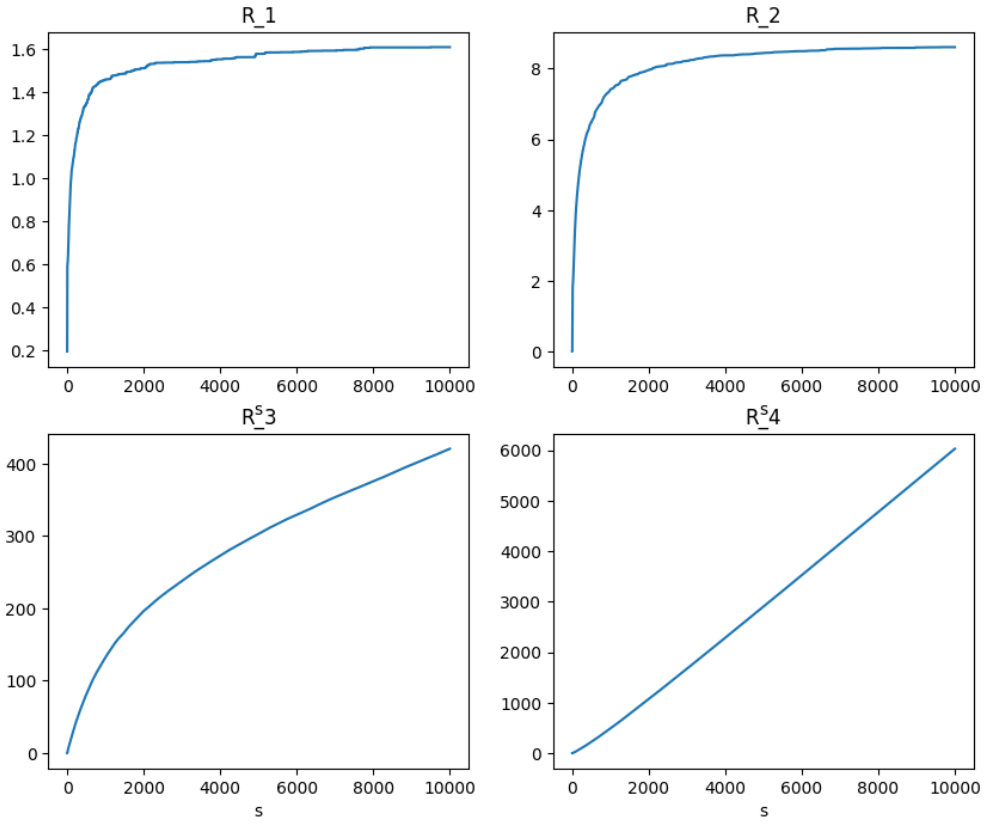




6451.546884852386 = total_revenue after N people

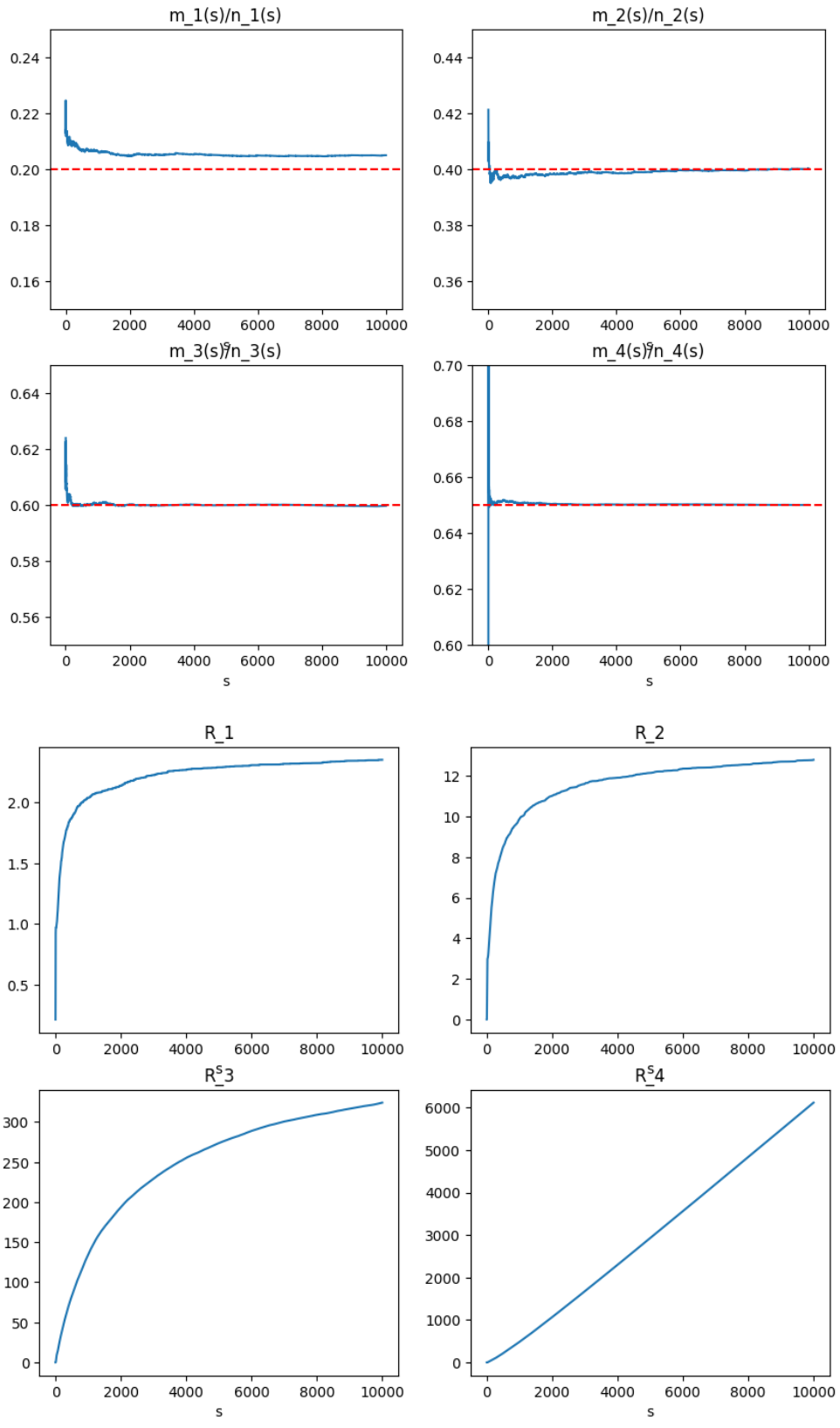
- Alpha = 0.05





Total_revenue_N_people = 6459.072892797279

- Alpha = 0.01



Total_revenue_N_people = 6457.328603952969

For **system1**: The total revenue for N people turns out to be

Alpha	Total Revenue from N people	Best expected value
0.1	6451	6,500
0.05	6459	6,500
0.01	6457	6,500

- The above gives the sample average of total revenue for 10,000 users for system 1.

Discussion on effects of various parameters:

Effect of α

- It can be observed that decreasing the value of alpha and hence decreasing our tolerance for the upper bound improves the algorithm's performance.
- This can be seen in terms of total revenue rise from alpha = 0.1 to alpha = 0.05.
- The R_k plot for k = 3 becomes flatter on increasing the value of alpha.
- This means the algorithm recommends type 3 lesser often for lower values of alpha and more recommends type 4, hence increasing the revenue.
- Observe that as alpha approaches 0 the algorithm becomes random, which should not generate high revenues on average, hence we hypothesize the existence of an optimal alpha.

Effect of N

- Increasing the value of N, causes the average revenue to increase, as the algorithm will have more time for using the best value it has discovered for more users hence increasing the revenue. For lower N values, system does not get enough time for exploitation.

Effect of p

- The higher the pk value the more likely algorithm is to recommend that video type. In the above algorithm irrespective of which video type gives maximum revenue, it recommends the one with higher pk.

Adapting the algorithm for System 2 and 3

- Algorithm 2 and 3 have different maximum revenue values for different video types.
- This will affect the optimum choice of the video type.
- Till now we have been recommending videos on the basis of UCB which is the upper bound on the values of p.
- The expected revenue for each video recommendation is

  E(Revenue) = $p_k \frac{a_k}{2}$

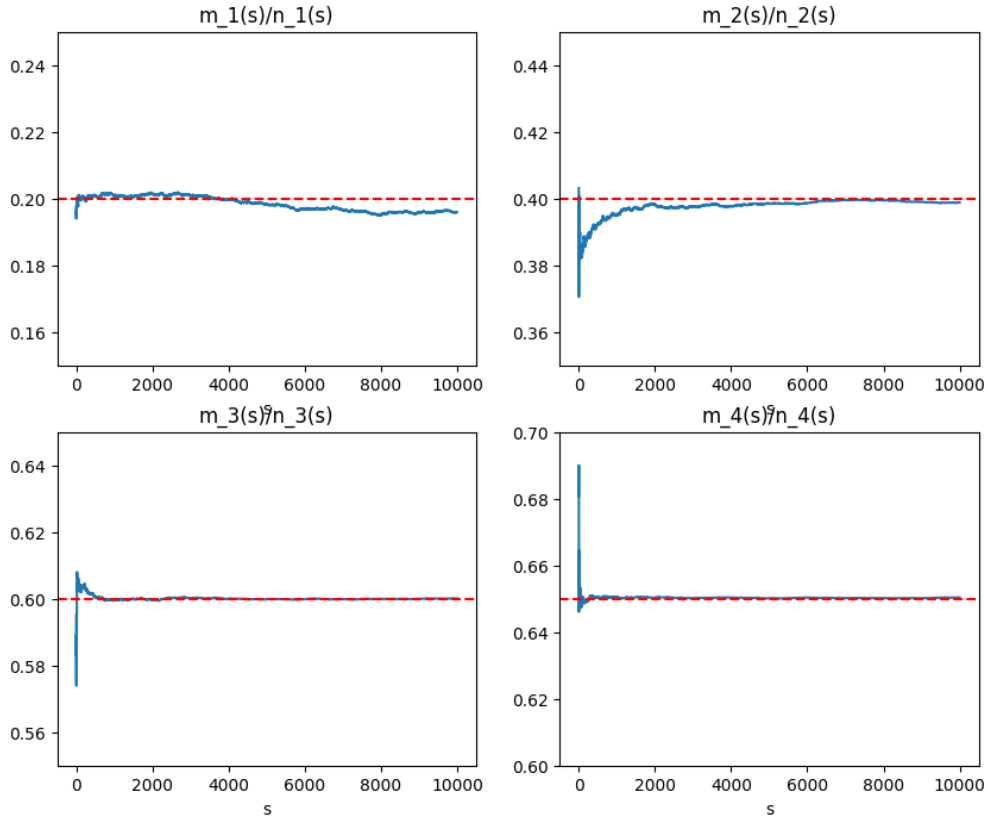
  - Hence the recommendation should also take into account the revenue collected for a video type for further recommendations.
  - For this we have updated the value of UCB_k as follows

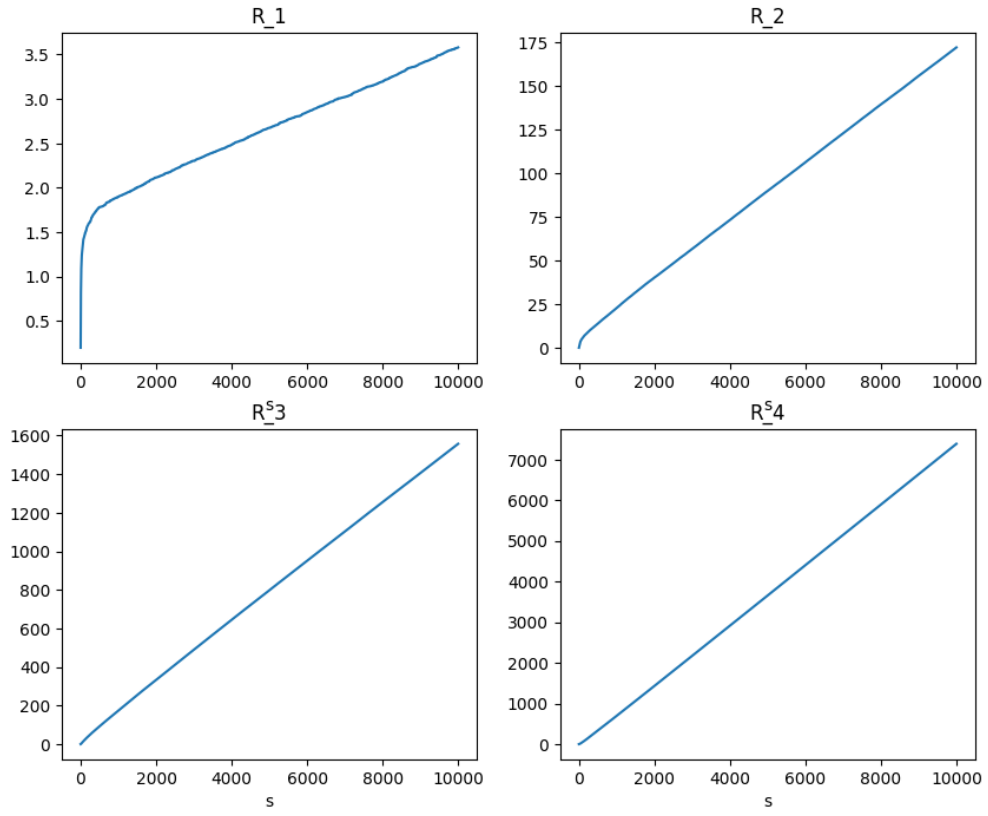
    $UCB_k = \left(\frac{R_k}{m_k}\right)\left(\frac{m_k}{n_k} + \sqrt{\left\{\frac{\log(1/\alpha)}{2n_k}\right)}\right\}$

  - This takes into account the revenue accumulated for a particular video type so far, more the revenue collected, higher the chances of the video to be recommended.
  - The following are the plots obtained for system 2 and 3 for the three values of alpha and N=10,000

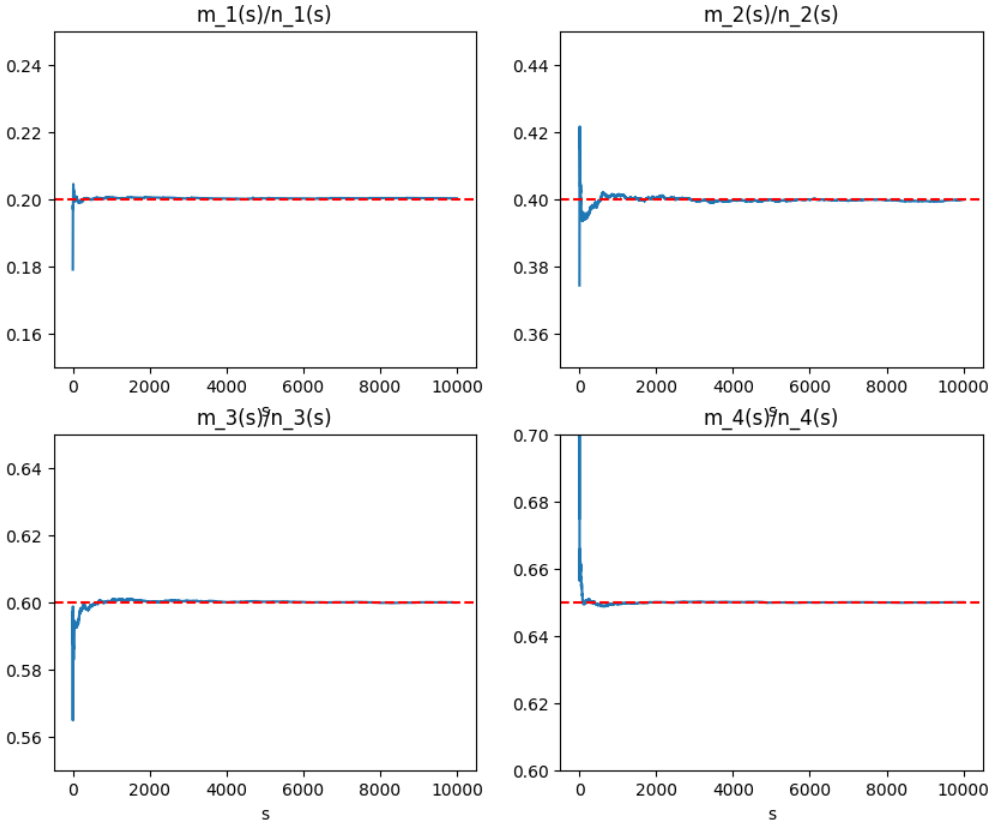
System 2:

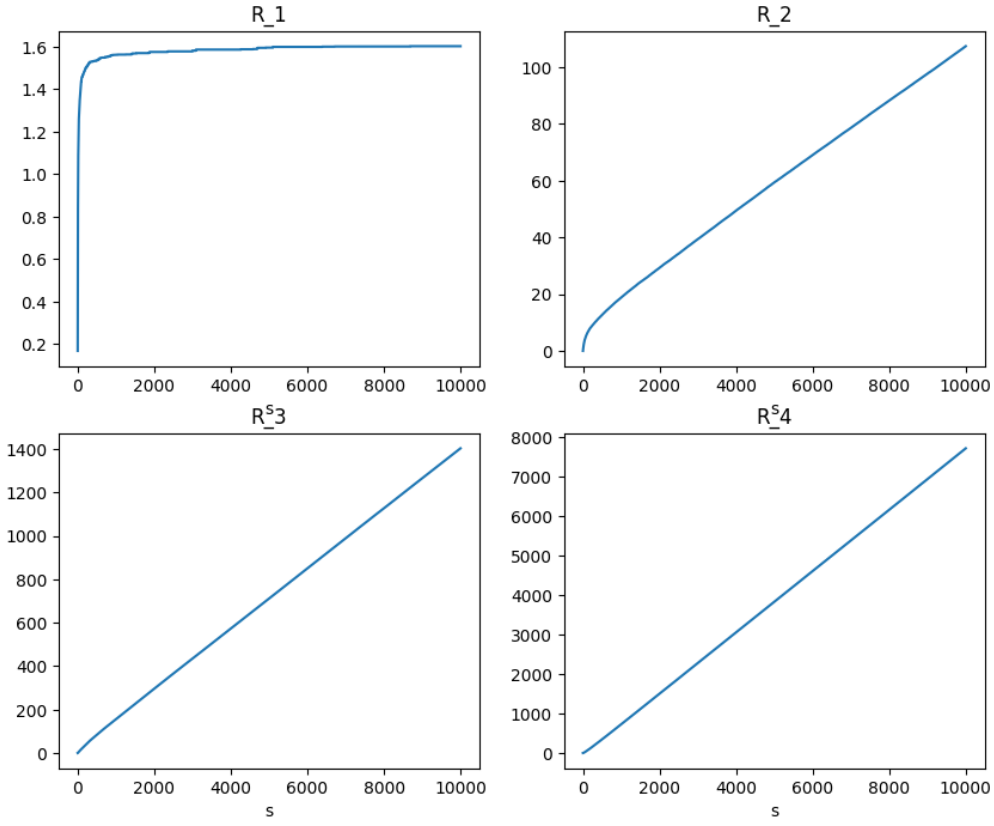
- Alpha = 0.1



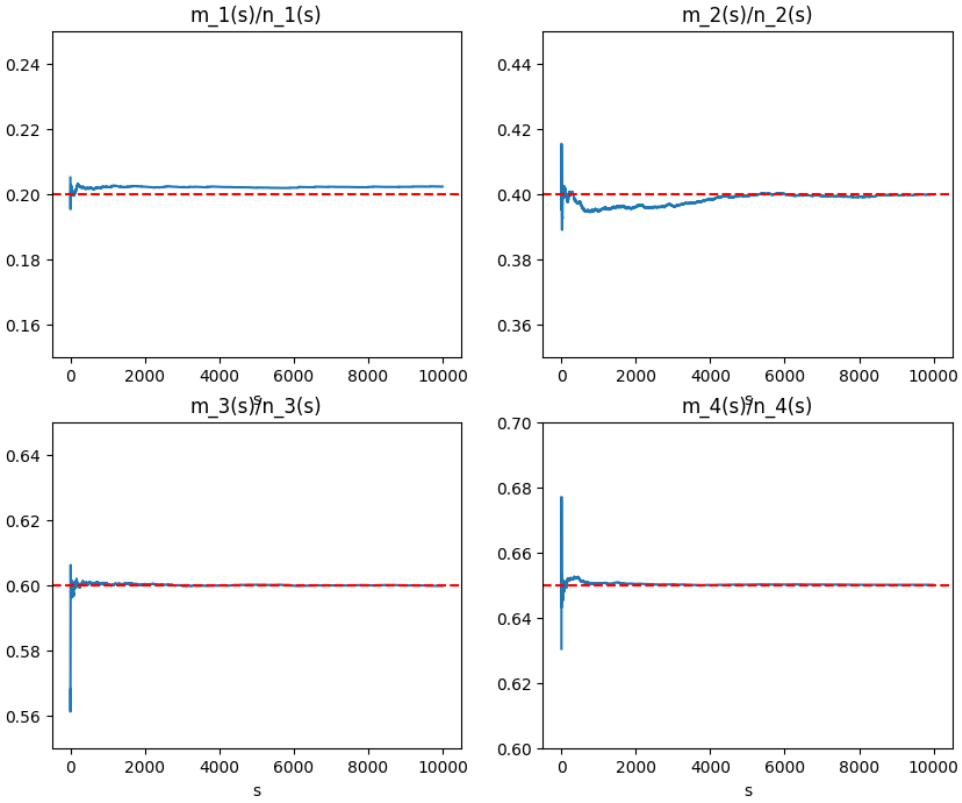


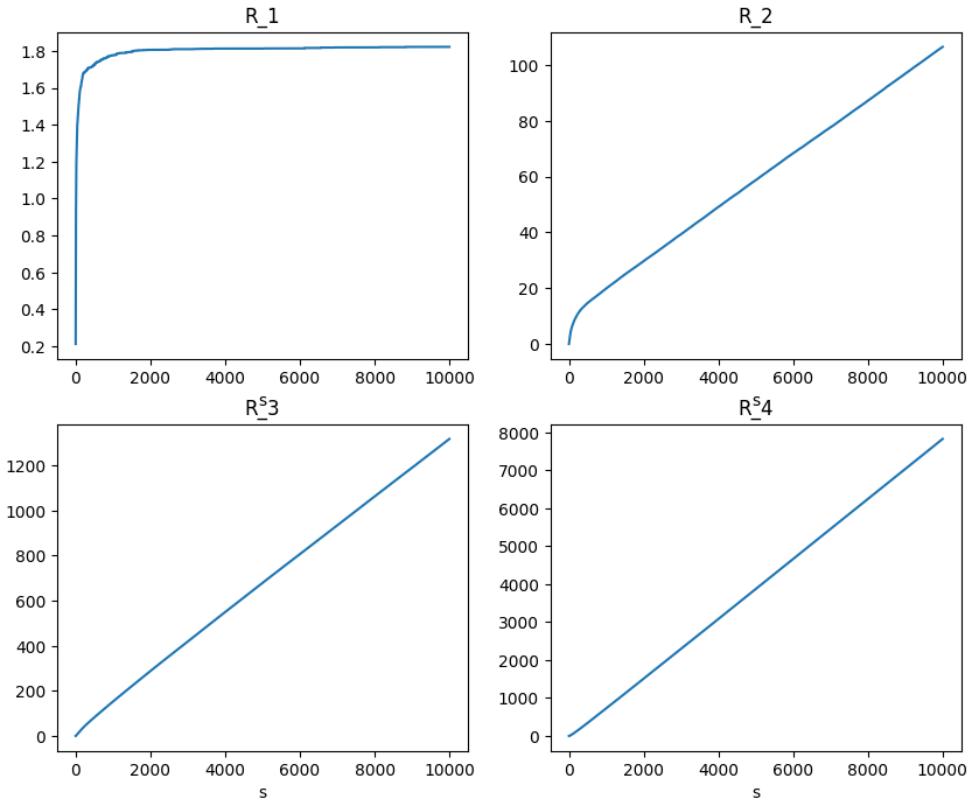
- Alpha = 0.05



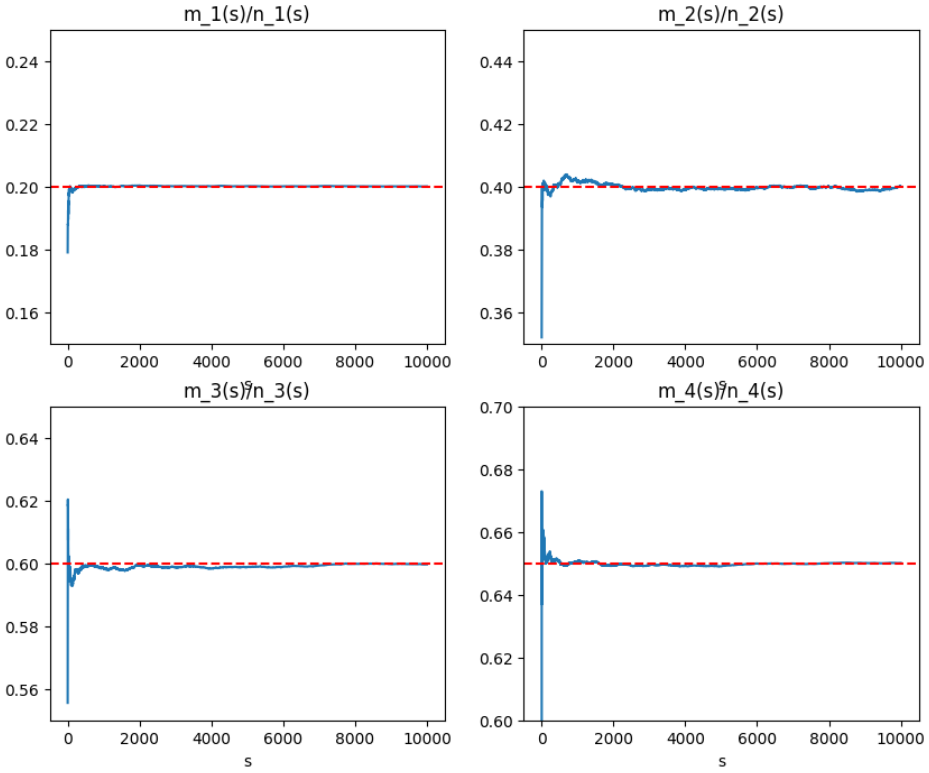


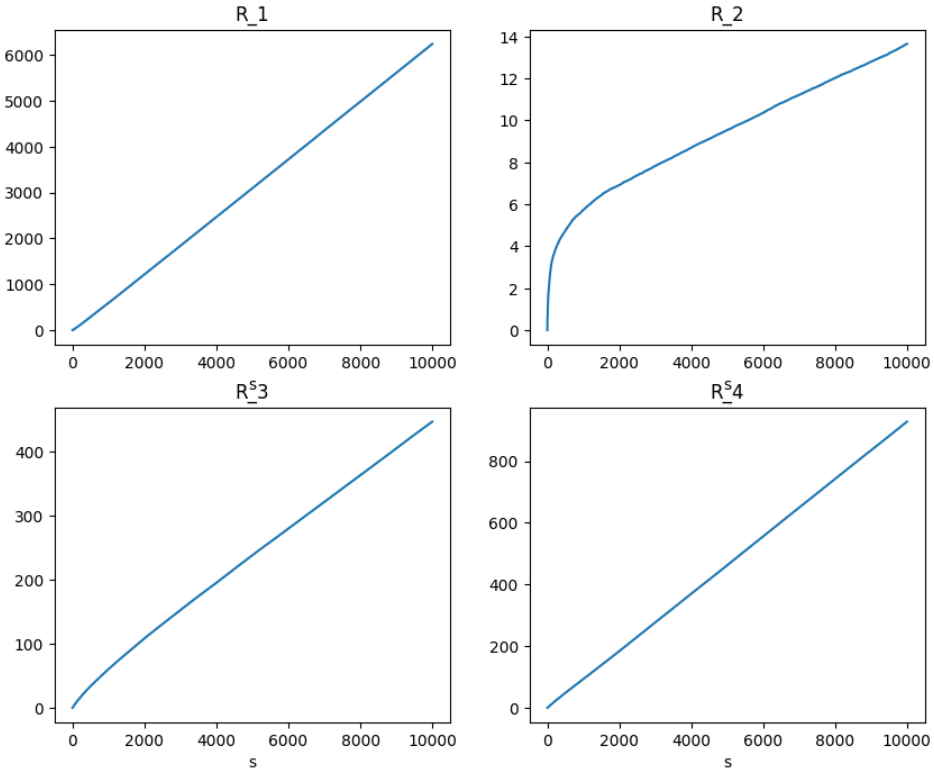
- Alpha = 0.01



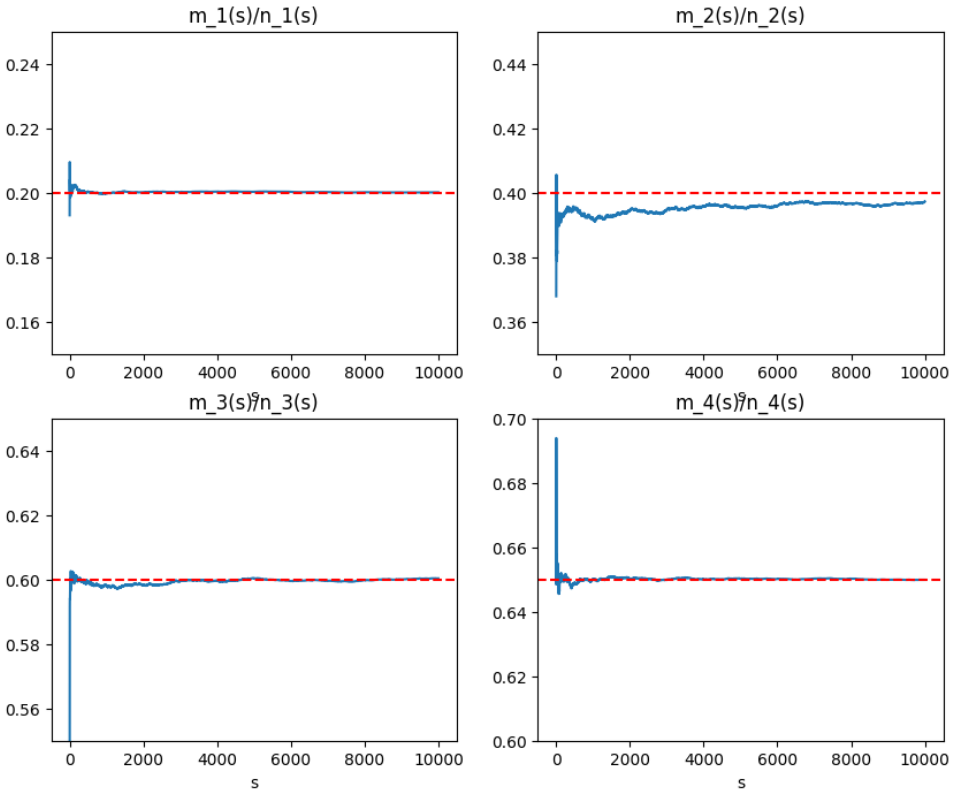


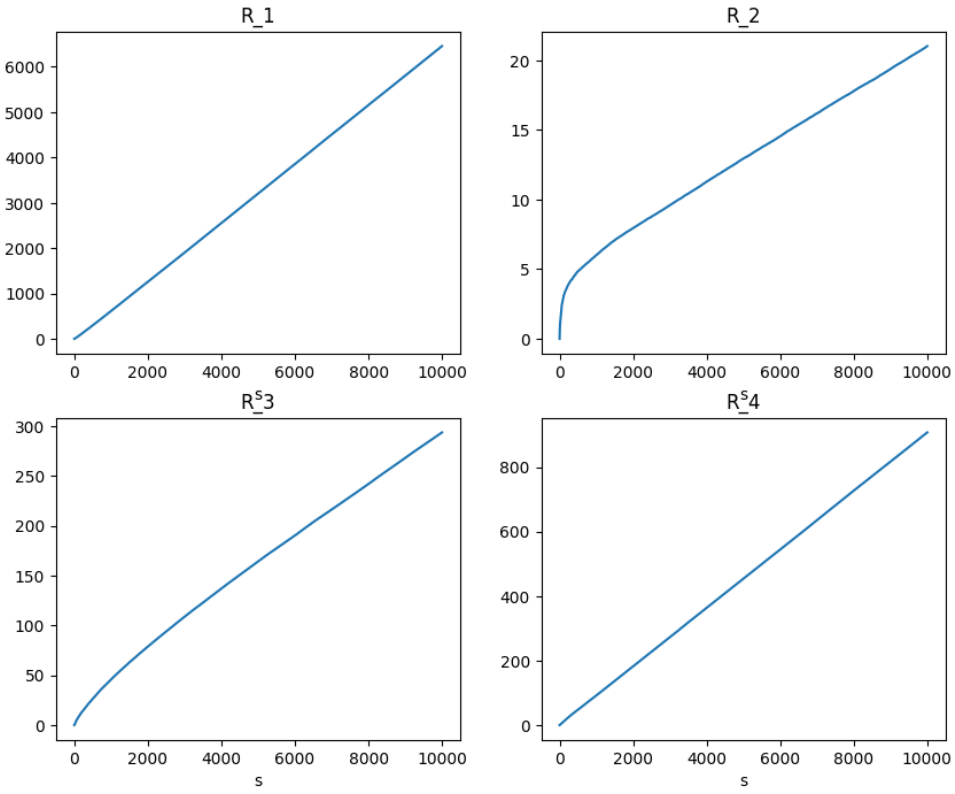
- **System 3:**
  - Alpha = 0.1



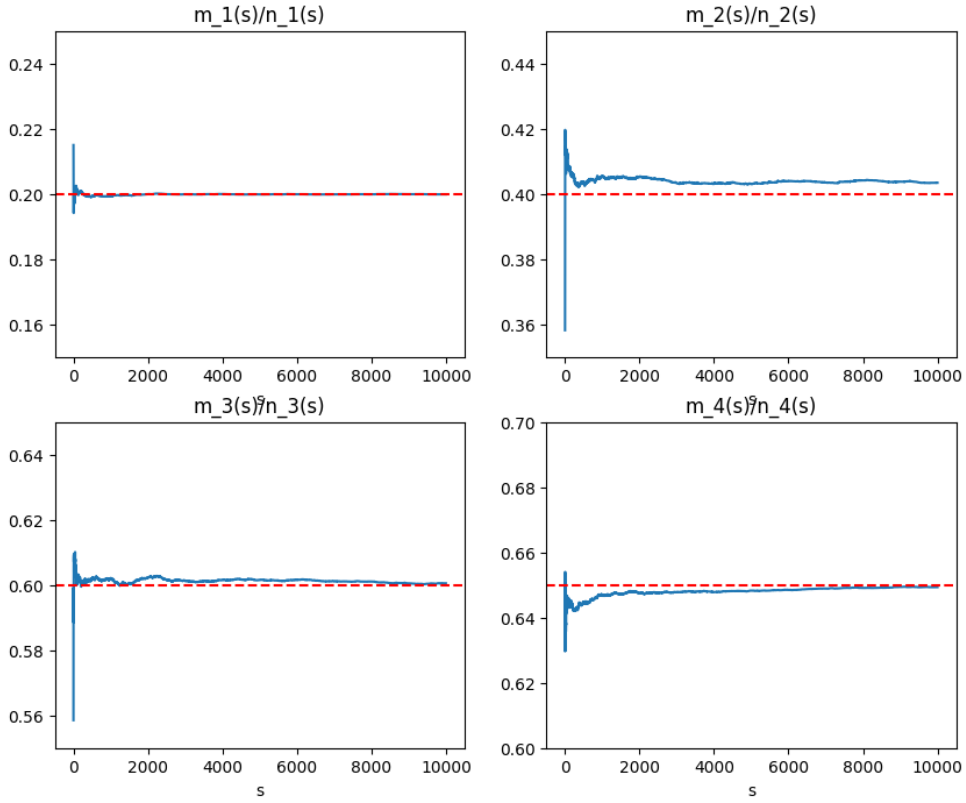


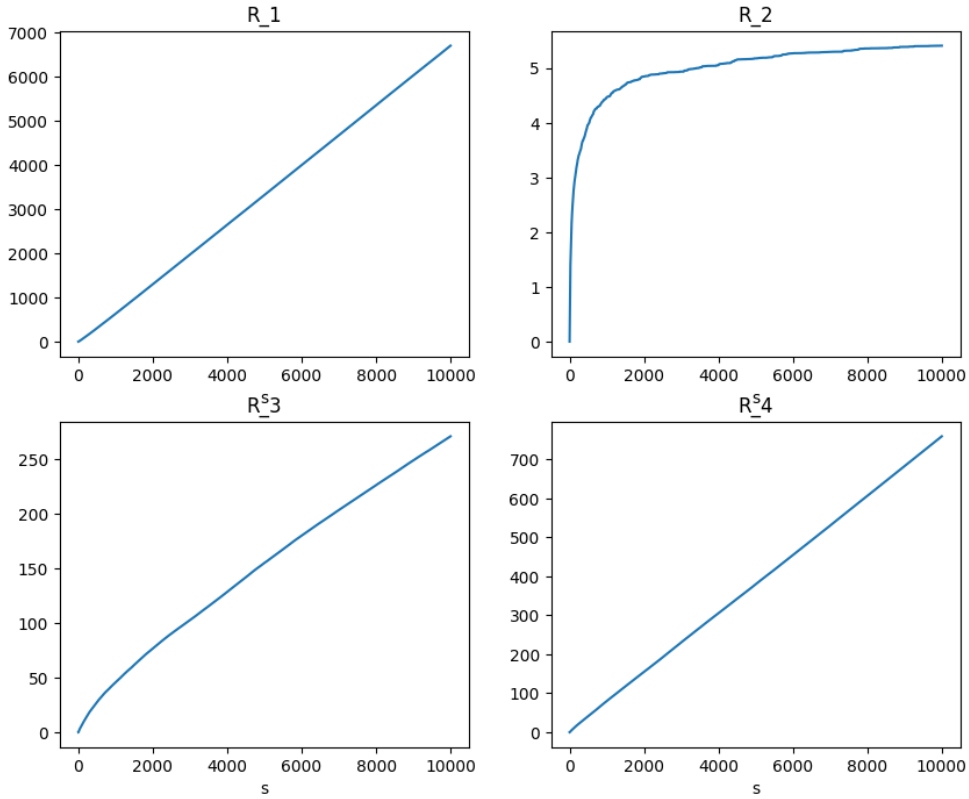
- Alpha = 0.05





- Alpha = 0.01





- Values of total revenues for system 2 and 3 at various alpha values
    - **System 2**

Alpha	Total Revenue from N people	Best expected value
0.1	9117	9,750
0.05	9226	9,750
0.01	9257	9,750

    - **System 3:**

Alpha	Total Revenue from N people	Best expected value
0.1	7668	8,000
0.05	7678	8,000
0.01	7728	8,000

- In this algorithm, we are assuming the p values to be constant over time.
- This assumption is not very realistic as user preferences change from time to time, in the real world.
- To tackle this problem we can reset the UCB values from time to time, so essentially we are cycling through exploration and exploitation multiple times.
- Instead of completely resetting UCB values we can keep a moving window of weights assigned to each iteration, such that the data collected from long ago recommendations holds less importance in determining ucb values.