Introduction

Decision-making problems, where an agent must choose the best action from several alternatives at each point in time, are widely encountered in practical applications, such as when companies decide how to structure their websites to encourage shopping behavior. These problems involve a trade-off between exploration and exploitation (Lai & Robbins, 1985; Auer et al., 2002). This trade-off is typically formulated as the multi-armed bandit problem, where each possible action, or arm, has an unknown reward probability distribution, and the agent aims to maximize cumulative rewards over time by selecting the optimal arm (Bouneffouf & Rish, 2019).

In this assignment, we apply the MAB framework to a real-world dataset from Zozo, a high-traffic fashion website. Here, the arms correspond to different fashion items, and the rewards are based on user clicks on the advertisement banner. The goal is to determine which items are best to recommend and identify the most effective positions on the recommendation interface (left, center, or right) to maximize click-through rates (CTR).

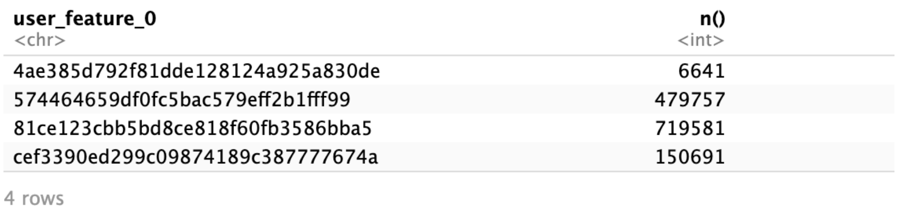
To achieve this, we will first describe and analyze the dataset. Then, we will test several MAB models, including Thompson Sampling, ε-greedy, and Upper Confidence Bound (UCB), to identify the best-performing items and banner positions for maximizing clicks. Additionally, we will perform a heterogeneity analysis to examine how different user segments impact the models' performance, a batch analysis to assess how varying batch sizes affect outcomes, and a parameter analysis to improve the models.

The dataset captures user interactions with recommended fashion items and includes variables such as timestamps of impressions, item IDs, positions on the interface, and click indicator. It also contains propensity scores, which represent the probability of an item being recommended in each position, along with several user-related features such as age and gender.

Heterogeneity and Aggregation Analysis

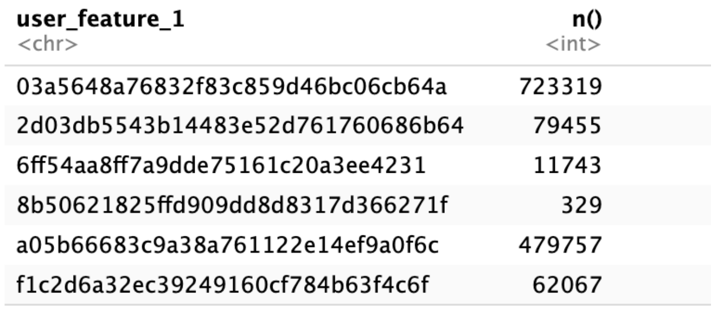
To perform a meaningful heterogeneity and aggregation analysis, our first step is to identify potential clusters within the dataset. This involves grouping the data based on various features and selecting the one that provides the most useful segmentation. Ideally, with a detailed description of the features, we could create more targeted segments, such as isolating middle-aged women . However, since we lack detailed information, we will focus on choosing the feature that best divides the population into meaningful subgroups.

What we are looking for is a feature that on one hand does not create too many subgroups, which could dilute our analysis, and on the other hand ensures that each subgroup has a substantial portion of the sample. This will help us avoid insignificant results and ensure that our analysis captures meaningful differences across segments.

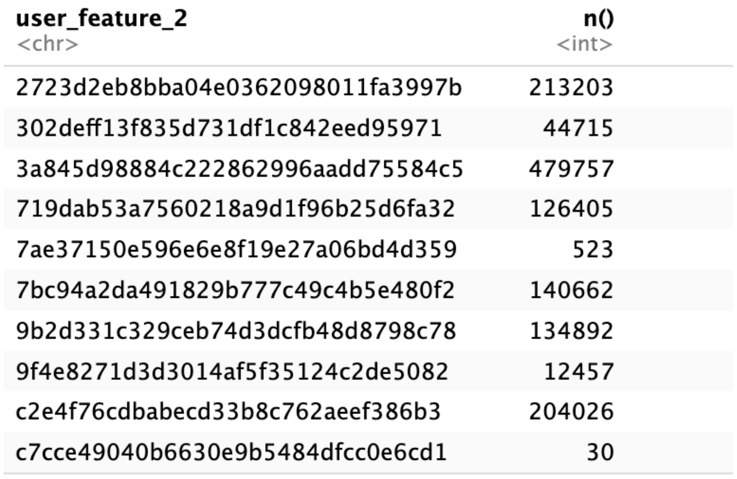


user_feature_0 <chr>	n() <int>
4ae385d792f81dde128124a925a830de	6641
574464659df0fc5bac579eff2b1fff99	479757
81ce123cbb5bd8ce818f60fb3586bba5	719581
cef3390ed299c09874189c387777674a	150691

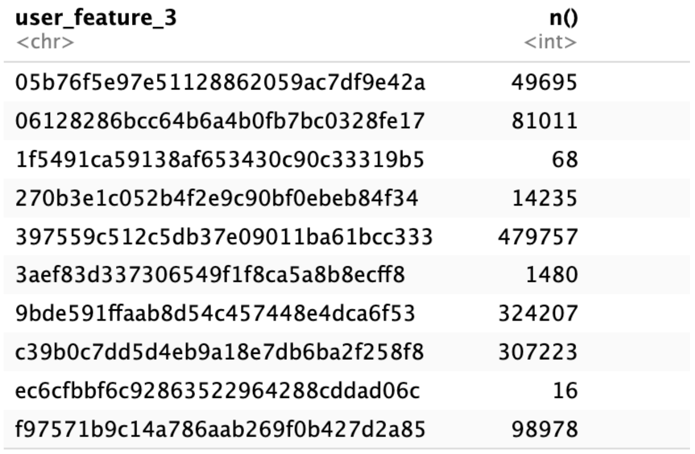
4 rows



user_feature_1 <chr>	n() <int>
03a5648a76832f83c859d46bc06cb64a	723319
2d03db5543b14483e52d761760686b64	79455
6ff54aa8ff7a9dde75161c20a3ee4231	11743
8b50621825ffd909dd8d8317d366271f	329
a05b66683c9a38a761122e14ef9a0f6c	479757
f1c2d6a32ec39249160cf784b63f4c6f	62067



user_feature_2 <chr>	n() <int>
2723d2eb8bba04e0362098011fa3997b	213203
302deff13f835d731df1c842eed95971	44715
3a845d98884c222862996aadd75584c5	479757
719dab53a7560218a9d1f96b25d6fa32	126405
7ae37150e596e6e8f19e27a06bd4d359	523
7bc94a2da491829b777c49c4b5e480f2	140662
9b2d331c329ceb74d3dcfb48d8798c78	134892
9f4e8271d3d3014af5f35124c2de5082	12457
c2e4f76cdbabecd33b8c762aeef386b3	204026
c7cce49040b6630e9b5484dfcc0e6cd1	30

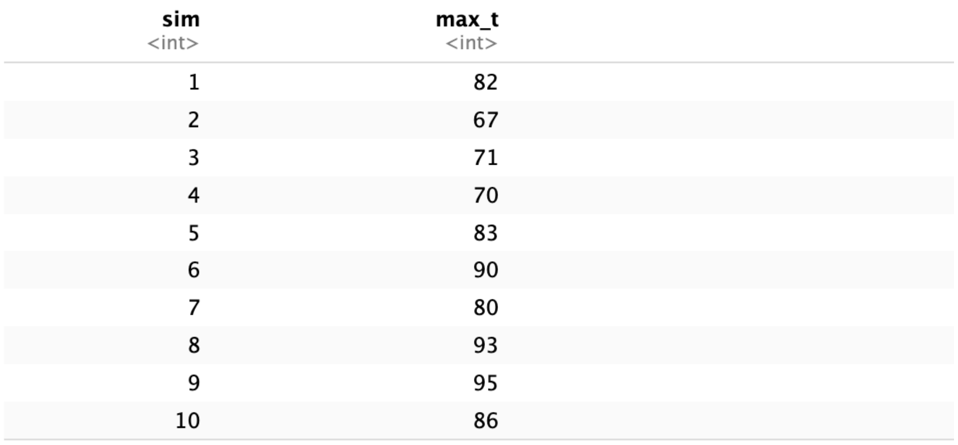


user_feature_3 <chr>	n() <int>
05b76f5e97e51128862059ac7df9e42a	49695
06128286bcc64b6a4b0fb7bc0328fe17	81011
1f5491ca59138af653430c90c33319b5	68
270b3e1c052b4f2e9c90bf0ebeb84f34	14235
397559c512c5db37e09011ba61bcc333	479757
3aef83d337306549f1f8ca5a8b8ecff8	1480
9bde591ffaab8d54c457448e4dca6f53	324207
c39b0c7dd5d4eb9a18e7db6ba2f258f8	307223
ec6cfbbf6c92863522964288cddad06c	16
f97571b9c14a786aab269f0b427d2a85	98978

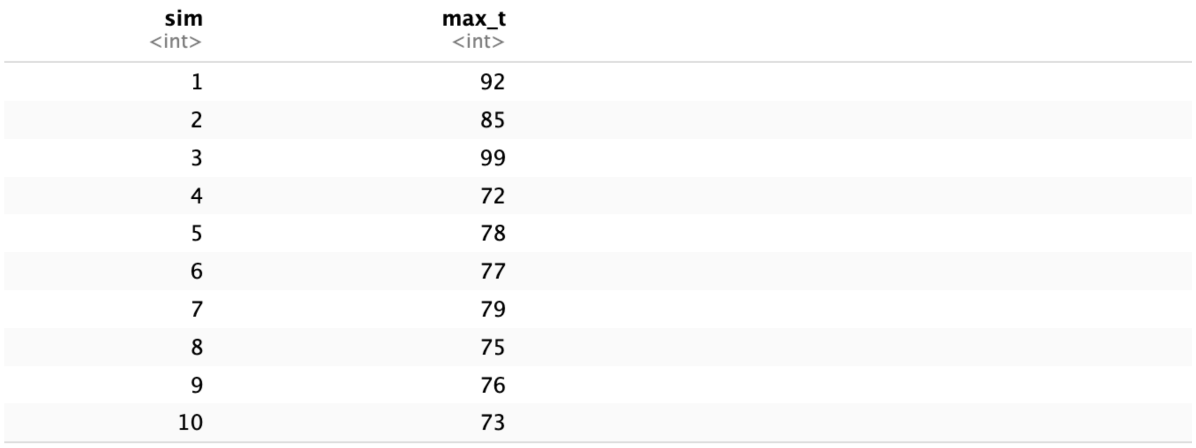
Since some user features, such as features 1, 2, and 3, contain very small sample sizes (e.g., values as low as 16), including these in our analysis would lead to unreliable estimates and reduce the statistical power of the results. Therefore, we are excluding these user features from the analysis to ensure the focus is on groups with larger, more balanced sample sizes, such as feature 0. Now, we divide feature 0 into the four previously identified segments, by creating 4 datasets.

Now we run Thompson Sampling simulations on each of the four segmented datasets based on feature 0 to evaluate how the model performs across different segments of users. We define the datasets in a list and set simulation parameters, such as a horizon size of 6641 impressions and 10 simulation runs per dataset. We set the horizon size to 6641, which matches the number of observations in the smallest segment of the dataset based on user_feature_0. By doing this, we ensure that each segment is treated equally during the simulations, preventing over-representation of larger segments. After running the simulations, we calculate the maximum number of observations (t) in each simulation by grouping the results by the simulation and summarizing the largest t value. Identifying the maximum t for each simulation is necessary to determine the length of each simulation run.

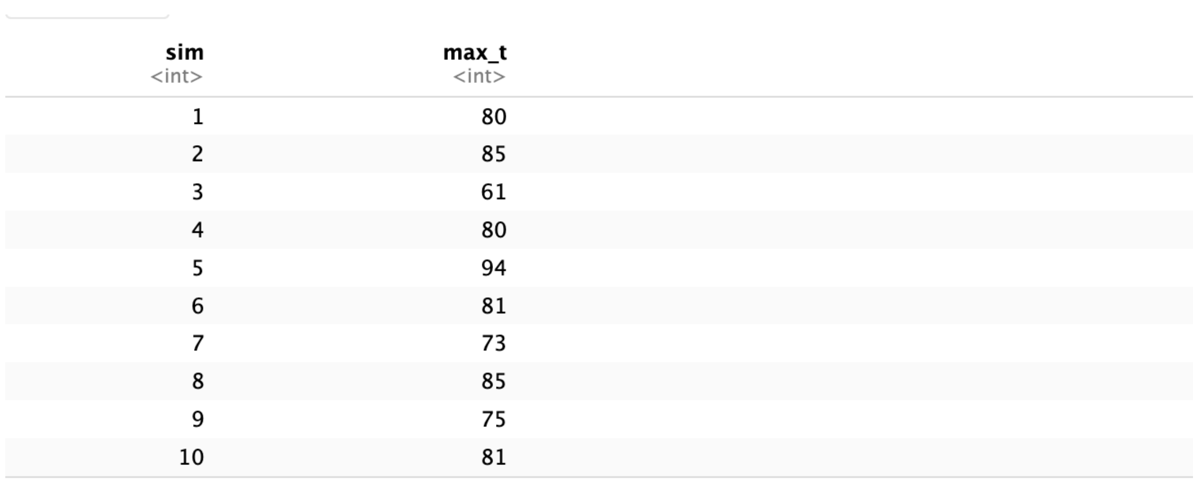
# This section calculates the maximum number of arm pulls per round for each dataset of user features.



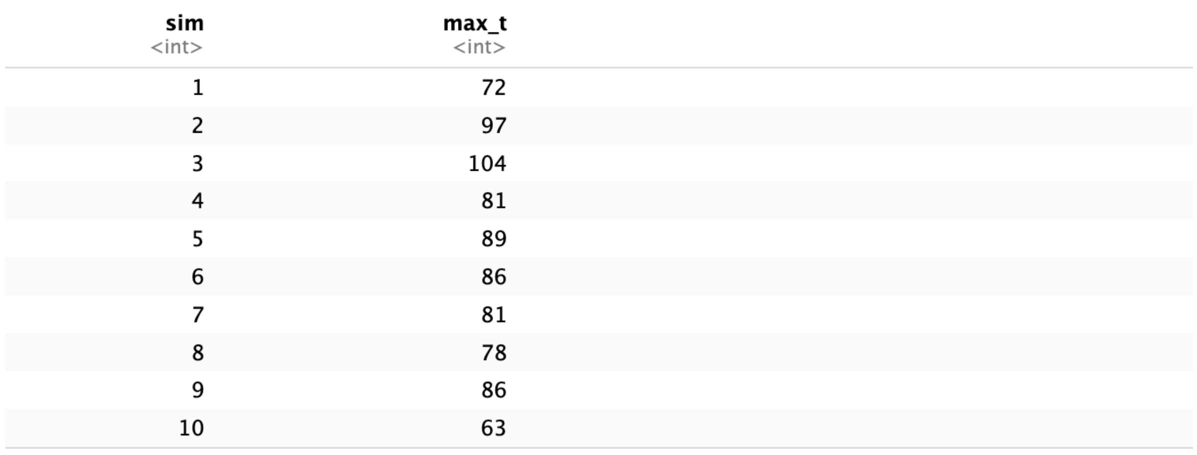
sim	max_t
<int>	<int>
1	82
2	67
3	71
4	70
5	83
6	90
7	80
8	93
9	95
10	86



sim	max_t
<int>	<int>
1	92
2	85
3	99
4	72
5	78
6	77
7	79
8	75
9	76
10	73

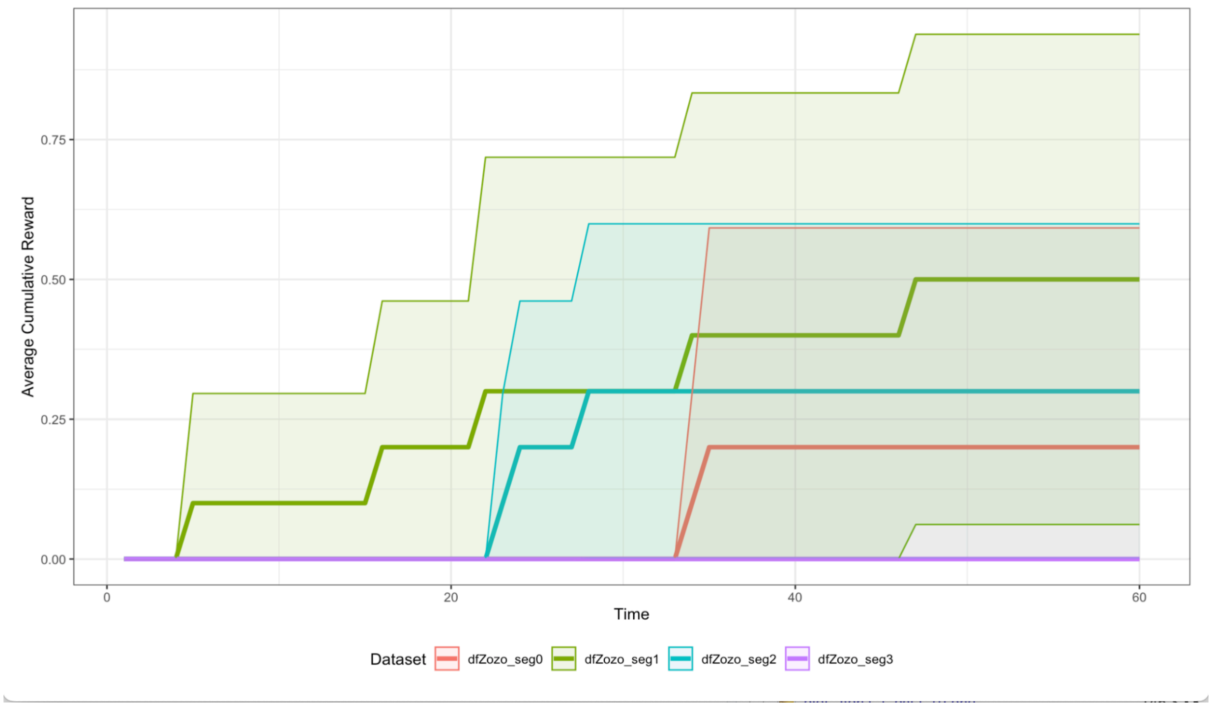


sim	max_t
<int>	<int>
1	80
2	85
3	61
4	80
5	94
6	81
7	73
8	85
9	75
10	81



sim	max_t
<int>	<int>
1	72
2	97
3	104
4	81
5	89
6	86
7	81
8	78
9	86
10	63

Once we have the maximum t for all simulations and datasets, we use the smallest maximum t across them to ensure uniformity when building our comparison. To identify which segment is more likely to click, we calculate the cumulative rewards for each dataset at each time step, average these rewards across simulations, and generate confidence intervals to account for uncertainty. This approach allows us to assess the performance of each segment in terms of click behavior.

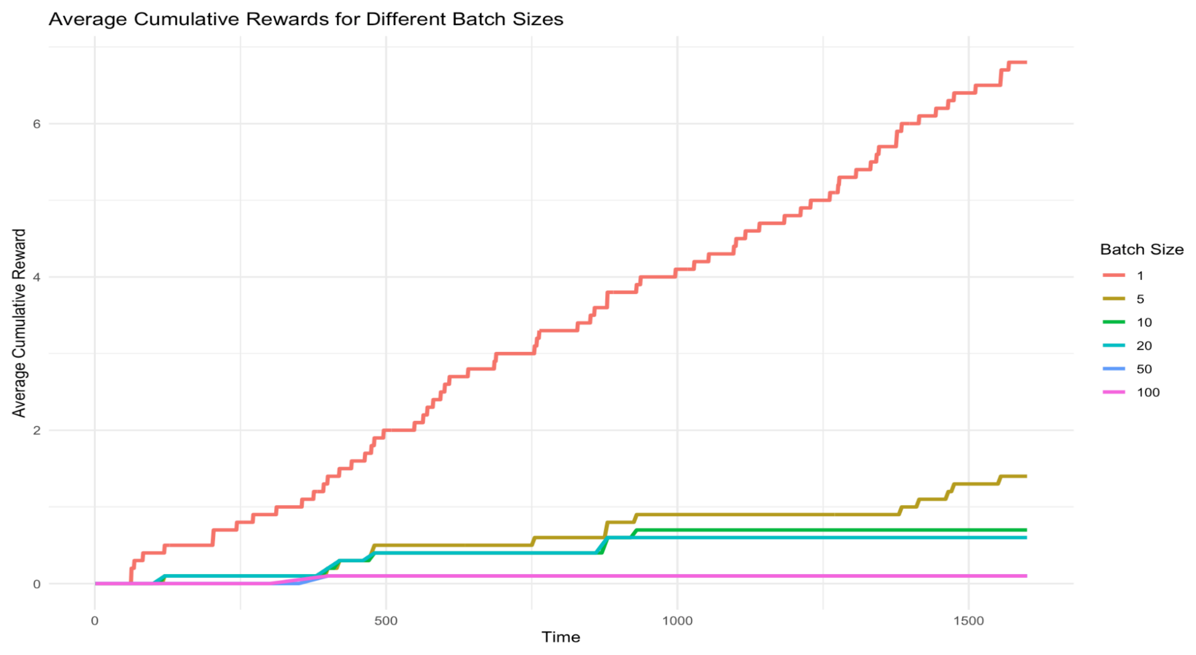


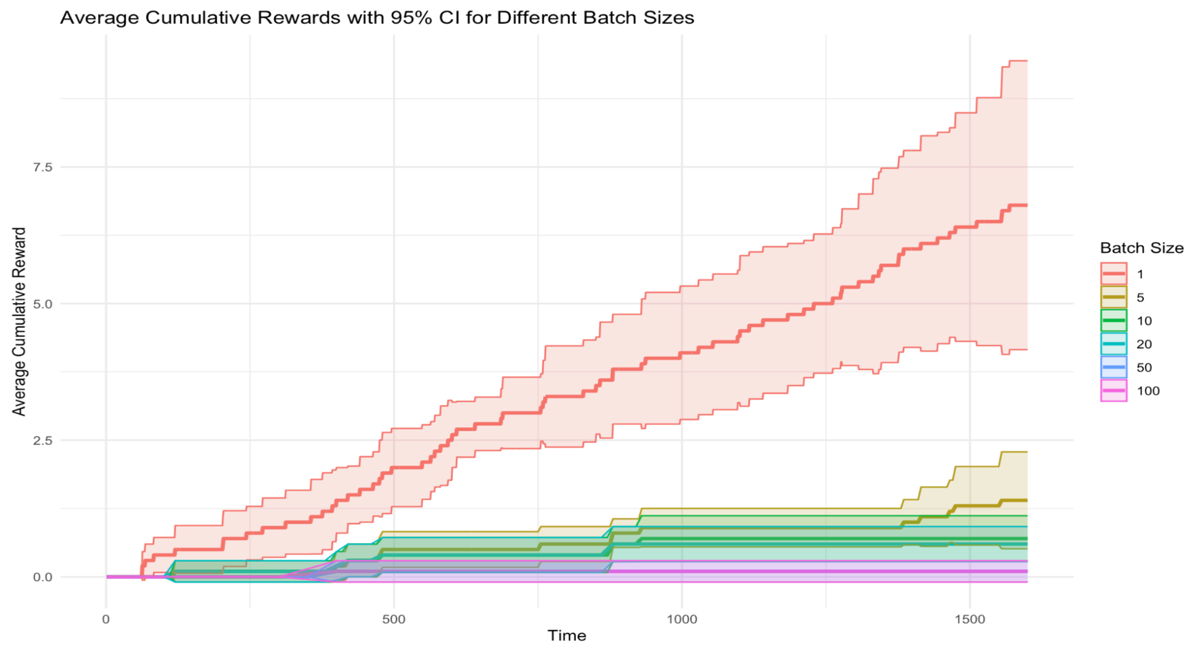
This graph shows the average cumulative reward over time for four user segments based on feature 0. The green line (dfZozo_feat1) appears to perform the best, accumulating the highest number of rewards (clicks), while the purple line (dfZozo_feat3) shows no reward, indicating that this group does not interact with the banner. However, the overlapping confidence intervals, represented by the shaded areas, introduce uncertainty into these results. Due to this overlap, we cannot definitively determine which segment is more likely to click, as the true performance differences between the segments are not significant.

# Batch Sensitivity Analysis
# Batch Sensitivity Analysis with the Contextual Package

Here we assess how different batch sizes affect a multi-armed bandit model's performance when applying the TS algorithm. Instead of updating the model after every single user interaction (which would be a batch size of 1), we can group several interactions together into "batches" and update the model after processing each batch. For each batch size, we simulate the relationship between item recommendations and user clicks, collect the results, and store them for comparison.

Once we have the maximum t for all simulations and depending on the batch size, we use the smallest maximum t across them to ensure uniformity when building our comparison. Then we identify which batch has higher average cumulative rewards at each time step and generate confidence intervals to account for uncertainty.

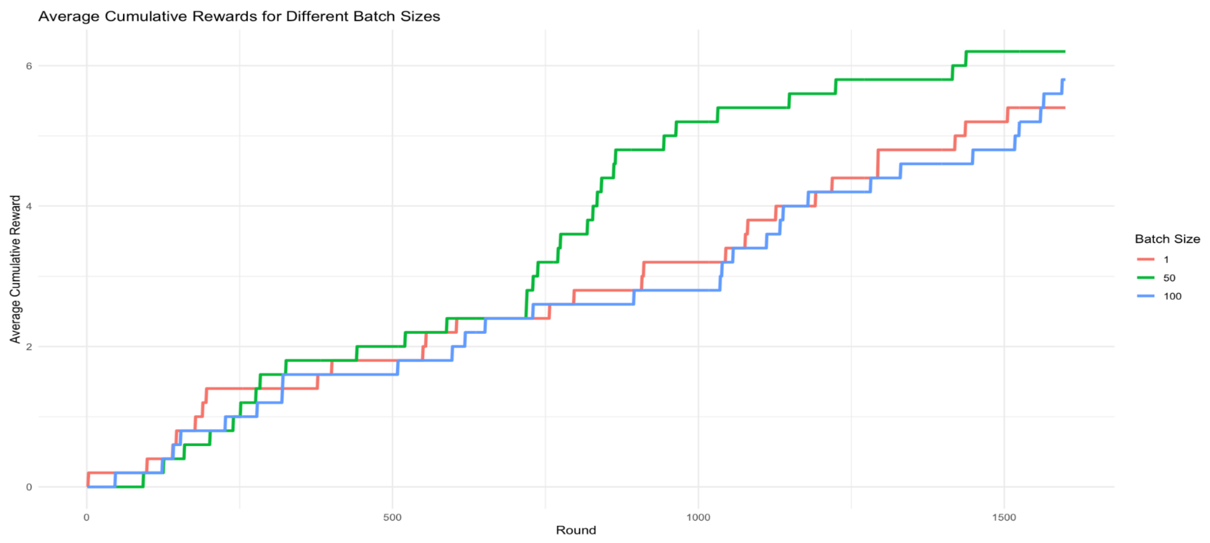


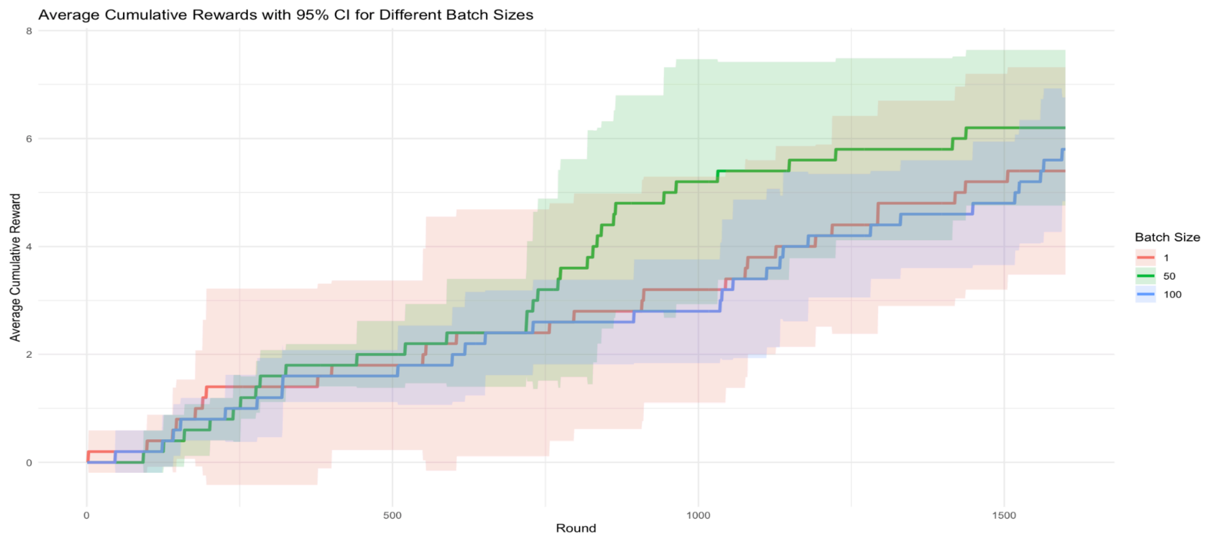


This graph displays the average cumulative rewards over time for different batch sizes (1, 5, 10, 20, 50, 100). The batch size of 1, represented by the red line, achieves the highest cumulative rewards but with wider confidence intervals, indicating greater variability in performance. As batch sizes increase, the cumulative rewards decrease, with batch sizes like 50 and 100 showing more stable but slower reward accumulation, reflected in narrower confidence intervals. This suggests that smaller batch sizes, which update the model more frequently, lead to faster learning and higher rewards, but with more uncertainty, while larger batch sizes result in more consistent, though less optimal, performance over time.

# Batch Sensitivity Analysis without the Contextual Package

Now we will perform Thompson Sampling by hand without the contextual package for batching but due to our limited computing power we will only test for 5 simulations with 3 batch sizes.





The plot shows that a batch size of 50 achieves higher cumulative rewards compared to batch sizes of 1 and 100 in the Thompson Sampling simulation. This is likely because batch size 50 strikes an effective balance between exploration and exploitation, allowing the algorithm to gather enough observations in each batch to make more informed decisions without delaying updates too much. In contrast, a batch size of 1 may be too exploratory, causing slower cumulative reward growth as it frequently tests different actions, while batch size 100 delays feedback and updates, resulting in slower adaptation and underperformance.