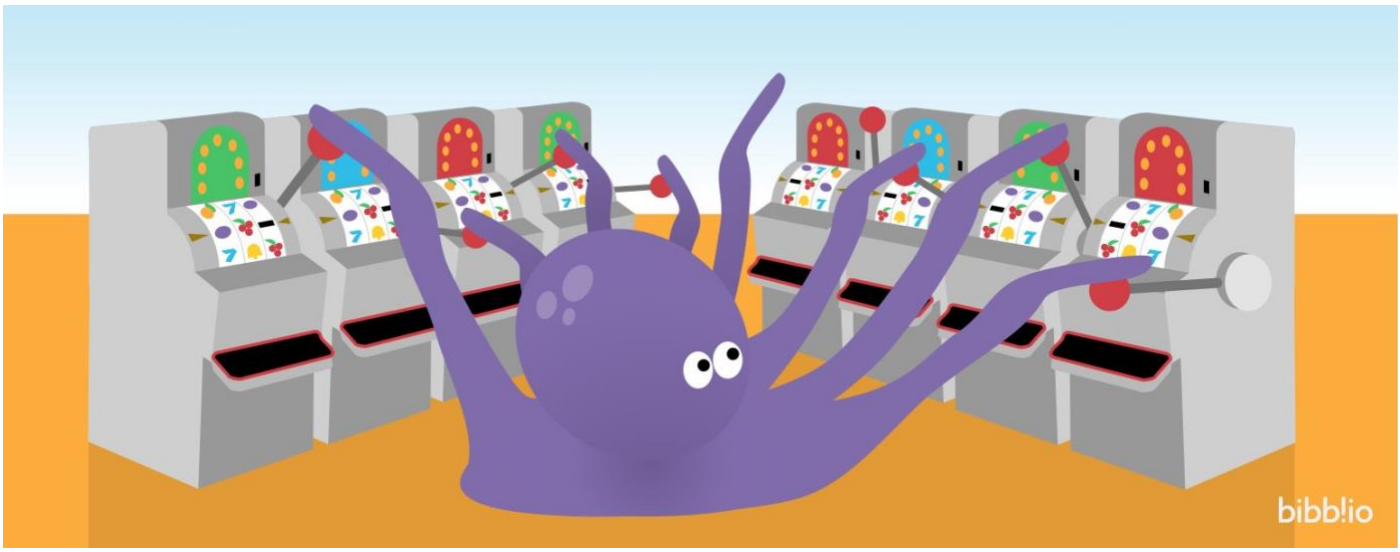
DSC5101 Group Assignment 3

A Study on Historical Surveys Using Multi-Armed Bandit Techniques



November 17, 2019

Utkarsh Chaturvedi	A0206518H
Dasol Kim	A0206452M
Shengnan Sang	A0206529A
Nikhil Mathai Thomas	A0206457A

Table of Contents

1 Executive Summary

This report aims to solve two business problems by implementing, as well as comparing, the Upper Confidence Bound Multi-Armed Bandit (UCB1) and Thompson Sampling deterministic algorithms.

- In the first scenario, the main issue of interest is whether A/B testing and website optimization used during Obama's fundraising campaign can be further improved with the aforementioned testing methods. Ultimately, results are compared with the actual campaign estimates of an additional $60 million in donations.

- The second case involves a commonplace decision for retail stores: among their newly released product lines, which items should be promoted to each distinct customer segment? Ideally, this targeted marketing will improve website conversion rates significantly. In particular, the customer types are divided into 4 groups based on age: less than 30, between 30 and 45, between 46 and 60, and greater than 60. We found significant differences in the product to be marketed to each age-group with a UCB1 test.

2 Multi-Armed Bandits

Similar to A/B testing, the Multi-Armed Bandit ("MAB") testing approach utilizes reinforcement learning to narrow down among a set of competing choices, and determine which variants are the best in terms of realizing the user's end goal. For instance, given both limited marketing budget and time constraint, the consumer-focused business aims to maximize expected profits through higher conversion rates and improved customer engagement.

The trade-off dilemma between exploration and exploitation is fundamental in the multi-arm bandit algorithm. Specifically, the hypothetical slot machine player performs "exploration" by testing new arms, while "exploitation" occurs when the gambler continues to play the most profitable arm in order to maximize expected return. This can be contrasted with A/B testing method that is wholly focused on "exploration" (in the initial phase of the experiment).

In brief, the epsilon-greedy strategy is commonly used to solve MAB problems. By definition, epsilon is a parameter between 0 and 1 and determined by the user as the total percentage of iterations for exploration. Alternatively, the Thompson Sampling ("TS") approach commonly uses a beta distribution model to illustrate the expected returns for each arm. During each round, the distribution shape adjusts accordingly and ultimately converges to the actual rate of success, for example, the website click-through and conversion rates. TS critically relies on Bayes' Theorem as exploitation dominates over time against exploration.

It can be argued that UCB and TS algorithms each have their advantages. To be more specific, UCB gives more importance to exploration when compared to the other methods. This ensures that the model is more robust towards changes in the customer profiles and shows higher sensitivity to time. On the other hand, exploitation is the main object for TS. Hence, as soon as a favourable option is found, then the algorithm continues to exploit in the subsequent trials, which leaves less likelihood for future exploration, unless a new arm with alpha & beta = 1 is introduced, and has a fair chance against the other arms.

3 Part I: Campaign Background

With the goal of raising Obama's presential rating and profile, members of the 2008 Obama Campaign performed A/B testing to decide which combination of the website would lead to the highest conversions in terms of click-through rate that eventually translates into donation funds. The test included a total combination of 24 test cases using 4 different buttons and 6 different media backgrounds. It was a randomized control trial conducted on approximately 310,000 unique visitors, with each iteration made visible to around 13,000 people. The A/B testing approach helped identify a

certain website styling that had a significantly higher click-through rate than the campaign manager's/teams favorite option. In fact, the winning variation had a sign-up rate of 11.6% compared to the average of 8.26%, which is an improvement of roughly 40.6%.
This above-mentioned approach that was adopted by the campaign team was highly successful since it led to additional 2.88 million email addresses and $60 million donations. However, there are a number of flaws that should be pointed out:

1. The A/B tests were performed during the start of the campaign. Yet as the presidential race advances, an alternative website setup would have been more favorable. This indicates that concentrated learning should be focused towards the start of the campaign.

2. With A/B testing, all the website versions were tested for 13,000 runs. This is not a very efficient approach since those under-performing websites are still being run. Hence, there is a waste of opportunity by running unsuccessful websites multiple times.

4 Part II: Retailer Background

The retailer *"ForeverProfit"* is planning to roll out 6 new products with an objective of identifying customer segments and promoting specific products that benefit them. The consumer types have been divided into 4 groups based on age, and each group is assumed to have different probabilities of purchasing the new products. Moreover, the unique number of visitors to the retailer's website is presumed to be 400,000, which is uniformly distributed among the 4 segments.

5 UCB1 Sampling vs. Thompson vs. Original

To address the two flaws with A/B testing, alternative techniques including UCB1 and TS are adopted below:

To reiterate, the UCB1 model is built with the aim on exploitation, that is running the successful website. However, there is now a time component that encourages exploration of arms that have not been played for a long period. The TS model uses a beta distribution to decide which arm is optimal to choose. Both models have spread out learning throughout the entire campaign duration, and is able to pick up changes in audience sentiment with regards to their most favorable website design.

Part I Results Table:

	Approach				
	AB Testing	UCB1	Thompson		Campaign Favorite
Exploitation	99.68%	99.40%	99.93%		***173,460,000***
Exploration	0.32%	0.60%	0.07%		
Total Donations	242,820,480	242,138,400	243,429,480		
Delta from Campaign Favorite	69,360,480	68,678,400	69,969,480		

Note: "Campaign Favorite" represents the combination initially preferred prior to A/B testing

According to the results from the table above, both UCB1 and TS has their own advantages after running for 100 million iterations. Note that UCB1 gave more importance to exploration when compared to the other two models. This addresses the concern of adjusting robustness in the case of changes in the targeted customers. Whereas for TS, exploitation is the main objective and once a favorable website is identified, then it continues to exploit with less probability of future exploration.

Based on the UCB1 results table from *Figure 1.1*, the maximum u values consistently occurred for the sixth combination, with an average of 0.11602050. Furthermore, *Figure 1.2* demonstrates the n values generated during each of the four runs, and the percentage of playing arm six is approximately 99.4%. Similarly, *Figures 2.1 – 2.3* present a coherent picture after applying TS: combination six was ran the

most, with an average success rate of roughly 0.116 for each trial. This translates into a total of 11,591,775 observed successes overall.

Part II Results Table:

	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6
Customer 1	38978	33812	5271	11168	4883	5889
Customer 2	6554	71967	6004	5154	5345	4977
Customer 3	3893	29441	9822	9189	31098	16558
Customer 4	4264	10893	11055	3794	4079	65916

The MAB approach is ideal when there are different customer segments, each with distinct affinities towards certain products. This model provides an ability to fine-tune which product should be promoted to each customer group.

To tackle the problem in part II, the UCB1 method was applied to determine which product has the highest conversion rates for each customer segment, which helps improve market basket analysis and limits the amount of time spent on advertising a product that has low affinity for the consumer. *Only UCB1 was considered for the experiment because we did not anticipate the addition of a new consumer segment or product to the basket.* Note that the estimated conversions for each type of customer and product combination have been depicted in *Figures 3.1 – 3.2,* with the highest values accentuated in green.

Additionally, the information gained can be served as an input to numerous price discrimination models after adjusting the prices to ensure not only that the products are favorable to the customer, but also that revenue and profitability is maximized.

6 Summary & Recommendations

After weighing the pros and cons of UCB1 and TS methods, we decide to implement both models to solve the two business problems. UCB1 is beneficial especially since customers may continuously update their needs, which is satisfied with this algorithm since it allows for exploration with a time aspect. Furthermore, it is advantageous to use TS when new products are released onto the market. In this event, learnings from previous runs are not lost and it is possible to incorporate new products into the model.

7 Appendix

Figure 1.1: UCB1 – u values

Table for u		Probability to Open				
	combination	Run 1	Run 2	Run 3	Run 4	Average
	1	0.10268486	0.10384415	0.10169836	0.10101998	0.10231184
	2	0.09389403	0.09101164	0.09272058	0.09173963	0.09234147
	3	0.06540840	0.06702209	0.06370422	0.06001590	0.06403765
	4	0.05062263	0.05733595	0.05082011	0.05917281	0.05448788
	5	0.10256565	0.10368123	0.10254406	0.10290237	0.10292333
	6	0.11604518	0.11600111	0.11601505	0.11602064	0.11602050
	7	0.09498878	0.10185430	0.09978570	0.09868460	0.09882835
	8	0.07419391	0.07773521	0.07416908	0.07598815	0.07552159
	9	0.07964452	0.07454844	0.07355979	0.07401999	0.07544319
	10	0.08891099	0.08951719	0.09002698	0.08701292	0.08886702
	11	0.09014267	0.08878616	0.08271850	0.08373397	0.08634533
	12	0.07816934	0.07291420	0.07950966	0.07696050	0.07688843
	13	0.08690911	0.08994269	0.08323588	0.09126498	0.08783817
	14	0.08713626	0.08424861	0.08979809	0.08916133	0.08758607
	15	0.08574794	0.07897041	0.08099420	0.08145488	0.08179186
	16	0.05556842	0.04893314	0.05173765	0.05365493	0.05247354
	17	0.07618276	0.07447276	0.07158106	0.07489224	0.07428221
	18	0.08446745	0.08871972	0.08702224	0.08766217	0.08696790
	19	0.09975275	0.10060648	0.09884428	0.09824250	0.09936150
	20	0.04484305	0.04881530	0.05214249	0.04633323	0.04803352
	21	0.07680220	0.07569639	0.08062450	0.07588048	0.07725089
	22	0.08400210	0.08707337	0.08309634	0.08771117	0.08547075
	23	0.09407349	0.09101526	0.09072539	0.08810264	0.09097920
	24	0.09584427	0.09175148	0.09320722	0.09342892	0.09355797

Figure 1.2: UCB1 – n values

Table for n						%			
	combination	Run 1	Run 2	Run 3	Run 4	Run 1	Run 2	Run 3	Run 4
	1	84511	101479	73895	67452	0.1%	0.1%	0.1%	0.1%
	2	31461	24821	28505	26270	0.0%	0.0%	0.0%	0.0%
	3	6146	6565	5761	5032	0.0%	0.0%	0.0%	0.0%
	4	3694	4587	3719	4884	0.0%	0.0%	0.0%	0.0%
	5	83059	98880	83174	87549	0.1%	0.1%	0.1%	0.1%
	6	99441015	99386303	99463821	99471845	99.4%	99.4%	99.5%	99.5%
	7	34762	75608	57864	50859	0.0%	0.1%	0.1%	0.1%
	8	8963	10703	8966	9791	0.0%	0.0%	0.0%	0.0%
	9	11815	9135	8714	8903	0.0%	0.0%	0.0%	0.0%
	10	21111	22141	22982	18503	0.0%	0.0%	0.0%	0.0%
	11	23130	20983	14096	14976	0.0%	0.0%	0.0%	0.0%
	12	10925	8462	11747	10278	0.0%	0.0%	0.0%	0.0%
	13	18341	22859	14537	25289	0.0%	0.0%	0.0%	0.0%
	14	18626	15478	22584	21534	0.0%	0.0%	0.0%	0.0%
	15	16980	11422	12754	13087	0.0%	0.0%	0.0%	0.0%
	16	4319	3515	3827	4063	0.0%	0.0%	0.0%	0.0%
	17	9871	9104	7963	9280	0.0%	0.0%	0.0%	0.0%
	18	15651	20886	18524	19347	0.0%	0.0%	0.0%	0.0%
	19	57432	64141	51829	48421	0.1%	0.1%	0.1%	0.0%
	20	3122	3503	3874	3259	0.0%	0.0%	0.0%	0.0%
	21	10182	9657	12490	9739	0.0%	0.0%	0.0%	0.0%
	22	15202	18605	14417	19416	0.0%	0.0%	0.0%	0.0%
	23	31975	24831	24249	19954	0.0%	0.0%	0.0%	0.0%
	24	37707	26332	29708	30269	0.0%	0.0%	0.0%	0.0%

Figure 2.1: Thompson – a values

Table for a					
	combination	Run 1	Run 2	Run 3	Run 4
	1	1139	1408	604	1119
	2	423	473	342	232
	3	59	49	47	82
	4	17	55	33	73
	5	899	1561	2428	848
	6	11596081	11590484	11584503	11596033
	7	606	633	309	400
	8	91	102	110	126
	9	65	108	107	161
	10	139	100	176	327
	11	72	223	163	219
	12	152	192	100	92
	13	217	363	302	231
	14	354	155	284	163
	15	148	124	139	66
	16	32	26	27	19
	17	115	114	49	61
	18	148	114	265	85
	19	441	485	1116	666
	20	25	88	31	17
	21	113	74	59	117
	22	232	136	188	131
	23	164	201	397	211
	24	544	270	801	311

Figure 2.2: Thompson – (a +b) values

Table for a+b						%		
	combination	Run 1	Run 2	Run 3	Run 4	Run 1	Run 2	Run 3	Run 4
	1	10983	13523	6152	10957	0.01%	0.01%	0.01%	0.01%
	2	4497	4932	3681	2591	0.00%	0.00%	0.00%	0.00%
	3	888	774	723	1090	0.00%	0.00%	0.00%	0.00%
	4	384	836	595	1029	0.00%	0.00%	0.00%	0.00%
	5	8879	14911	22682	8385	0.01%	0.01%	0.02%	0.01%
	6	99933035	99924836	99915575	99936920	99.93%	99.92%	99.92%	99.94%
	7	6093	6430	3403	4280	0.01%	0.01%	0.00%	0.00%
	8	1163	1326	1401	1587	0.00%	0.00%	0.00%	0.00%
	9	951	1407	1406	1938	0.00%	0.00%	0.00%	0.00%
	10	1667	1303	2043	3557	0.00%	0.00%	0.00%	0.00%
	11	1021	2502	1970	2483	0.00%	0.00%	0.00%	0.00%
	12	1840	2258	1331	1224	0.00%	0.00%	0.00%	0.00%
	13	2432	3870	3332	2622	0.00%	0.00%	0.00%	0.00%
	14	3798	1843	3142	1921	0.00%	0.00%	0.00%	0.00%
	15	1810	1572	1643	933	0.00%	0.00%	0.00%	0.00%
	16	570	499	521	411	0.00%	0.00%	0.00%	0.00%
	17	1446	1474	763	916	0.00%	0.00%	0.00%	0.00%
	18	1798	1471	2966	1159	0.00%	0.00%	0.00%	0.00%
	19	4627	4999	10781	6723	0.00%	0.00%	0.01%	0.01%
	20	495	1201	547	389	0.00%	0.00%	0.00%	0.00%
	21	1450	1050	895	1488	0.00%	0.00%	0.00%	0.00%
	22	2647	1689	2211	1644	0.00%	0.00%	0.00%	0.00%
	23	1929	2332	4245	2429	0.00%	0.00%	0.00%	0.00%
	24	5645	3010	8040	3372	0.01%	0.00%	0.01%	0.00%

Figure 2.3: Thompson – probabilities

Probabilities			
Run 1	Run 2	Run 3	Run 4
0.104	0.104	0.098	0.102
0.094	0.096	0.093	0.090
0.066	0.063	0.065	0.075
0.044	0.066	0.055	0.071
0.101	0.105	0.107	0.101
0.116	0.116	0.116	0.116
0.099	0.098	0.091	0.093
0.078	0.077	0.079	0.079
0.068	0.077	0.076	0.083
0.083	0.077	0.086	0.092
0.071	0.089	0.083	0.088
0.083	0.085	0.075	0.075
0.089	0.094	0.091	0.088
0.093	0.084	0.090	0.085
0.082	0.079	0.085	0.071
0.056	0.052	0.052	0.046
0.080	0.077	0.064	0.067
0.082	0.077	0.089	0.073
0.095	0.097	0.104	0.099
0.051	0.073	0.057	0.044
0.078	0.070	0.066	0.079
0.088	0.081	0.085	0.080
0.085	0.086	0.094	0.087
0.096	0.090	0.100	0.092

Figure 3.1 UCB1 – u values

	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6
Run 1						
Customer 1	0.10257391	0.10137923	0.07203115	0.08954847	0.07126899	0.07727660
Customer 2	0.08030906	0.11708528	0.08802118	0.07778066	0.08890998	0.08856683
Customer 3	0.06582598	0.10042912	0.08095121	0.08222781	0.09377017	0.09321725
Customer 4	0.06015675	0.07413864	0.07936850	0.05847554	0.05847554	0.09362759
Run 2						
	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6
Customer 1	0.10287755	0.09756596	0.07688451	0.08793474	0.07766522	0.08019201
Customer 2	0.08920188	0.11781856	0.09045303	0.08703471	0.08328009	0.07664437
Customer 3	0.06590486	0.09926665	0.08857595	0.08502365	0.10052347	0.08678074
Customer 4	0.06079501	0.07458157	0.07026183	0.05365597	0.05556910	0.09303012
Run 3						
	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6
Customer 1	0.10196013	0.10264159	0.07153966	0.08915977	0.06960501	0.07625899
Customer 2	0.09188282	0.11604611	0.08666215	0.08787879	0.08402072	0.08160649
Customer 3	0.06590850	0.09509721	0.08590476	0.08177255	0.10013622	0.09451899
Customer 4	0.05539736	0.07774712	0.07494122	0.05204352	0.05412021	0.09418449
Run 4						
	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6
Customer 1	0.10248299	0.10321775	0.07830015	0.08780584	0.07301452	0.07464761
Customer 2	0.09089776	0.11436983	0.08350046	0.08115236	0.08213592	0.08328727
Customer 3	0.06616352	0.09685377	0.08236244	0.08521352	0.09929778	0.09150327
Customer 4	0.05241090	0.07643072	0.07794418	0.05425056	0.05797101	0.09499917

Figure 3.2: UCB1– n values

	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6
Run 1						
Customer 1	39201	33932	4623	11893	4476	5875
Customer 2	4271	73297	6044	3857	6321	6210
Customer 3	3965	39849	8116	8744	20145	19181
Customer 4	4721	9752	13935	4395	4395	62802
Run 2						
	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6
Customer 1	45073	24486	5983	11156	6219	7083
Customer 2	6177	73401	6578	5561	4695	3588
Customer 3	3763	30272	12001	9515	33813	10636
Customer 4	5132	10874	8326	3802	4103	67763
Run 3						
	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6
Customer 1	35916	38386	4501	11485	4152	5560
Customer 2	7749	70179	5908	6270	5213	4681
Customer 3	3869	21294	10500	8169	35971	20197
Customer 4	3863	12322	10208	3401	3677	66529
Run 4						
	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6
Customer 1	35723	38443	5977	10136	4684	5037
Customer 2	8020	70989	5485	4929	5150	5427
Customer 3	3975	26349	8669	10327	34462	16218
Customer 4	3339	10624	11752	3576	4140	66569

Code1.1-UCB – Rcode

```
library(dplyr)
##All possible combinations - Image(i) -inrows
comb = c(0.102,0.0923,0.063,0.056,
         0.103,0.116,0.098,0.077,
         0.078,0.0903,0.087,0.075,
         0.088,0.0877,0.0826,0.055,
         0.076,0.0864,0.100,0.051,
         0.074,0.085,0.091,0.094)

Donation = 21
Target_pop = 100000000
len = length(comb)
play = 0
Total_Donation=0

####UCB1
u = c(0,0,0,0,
      0,0,0,0,
      0,0,0,0,
      0,0,0,0,
      0,0,0,0,
      0,0,0,0)
uPlus = c(0,0,0,0,
      0,0,0,0,
      0,0,0,0,
      0,0,0,0,
      0,0,0,0,
      0,0,0,0)

n = c(0,0,0,0,
      0,0,0,0,
      0,0,0,0,
      0,0,0,0,
      0,0,0,0,
      0,0,0,0)
```

```
###initial run

for(i in 1:len)
{
  u[i]=rbinom(1,1,comb[i])
  n[i]=n[i]+1
}
Total_Donation=sum(u)*21


for(i in 1:(Target_pop-len))
{
  if(i%%100000==0)
  {print(i)}
  for(j in 1:len)
  {
  uPlus[j]=u[j]+sqrt(2 * log10(i)/n[j])
  }
  index=which.max(uPlus)
  play=rbinom(1,1,comb[index])
  Total_Donation=Total_Donation + play*21
  u[index]=(u[index]*n[index]+play)/(n[index] + 1)
  n[index] = n[index] + 1
}

sum(n)
##TotalDonation
Total_Donation

##position of best version
which.max(u)
##prob of best version
max(u)
u
n
```

Code1.2 - Thompson – Rcode

```
#library(dplyr)
##All possible combinations - Image(i) -inrows
comb = c(0.102,0.0923,0.063,0.056,
         0.103,0.116,0.098,0.077,
         0.078,0.0903,0.087,0.075,
         0.088,0.0877,0.0826,0.055,
         0.076,0.0864,0.100,0.051,
         0.074,0.085,0.091,0.094)

Donation = 21
Target_pop = 100000000
len = length(comb)
play = 0
Total_Donation=0

####Thompson
a = c(1,1,1,1,
      1,1,1,1,
      1,1,1,1,
      1,1,1,1,
      1,1,1,1,
      1,1,1,1)
b = c(1,1,1,1,
      1,1,1,1,
      1,1,1,1,
      1,1,1,1,
      1,1,1,1,
      1,1,1,1)
beta = c(0,0,0,0,
         0,0,0,0,
         0,0,0,0,
         0,0,0,0,
         0,0,0,0,
         0,0,0,0)
```

```
###initial run

for(i in 1:(Target_pop))
{
  if(i%%100000==0)
  {print(i)}
  for(j in 1:len)
  {
    beta[j]=rbeta(1,a[j],b[j])
  }
  index=which.max(beta)
  play=rbinom(1,1,comb[index])
  Total_Donation=Total_Donation + play*21
  if (play>0) {
    a[index]=a[index]+1
  }
  if (play<1) {
    b[index]=b[index]+1
  }
}


##TotalDonation
Total_Donation

a
b
a+b
which.max(a+b)
a/(a+b)
```

Code2.1 - UCB – Rcode

```
#library(dplyr)

##All possible combinations - Image(i) -inrows
comb = c(0.102,0.0923,0.063,0.056,
         0.103,0.116,0.098,0.077,
         0.078,0.0903,0.087,0.075,
         0.088,0.0877,0.0826,0.055,
         0.076,0.0864,0.100,0.051,
         0.074,0.085,0.091,0.094)
cust = matrix(comb,4,6)
rnam<-c('c1','c2','c3','c4')
row.names(cust)<-rnam
cnam<-c('P1','P2','P3','P4','P5','P6')
colnames(cust)<-cnam
cust

Target_pop = 400000
len = ncol(cust)
iter =nrow(cust)
play = 0
####UCB2
ze = c(0,0,0,0,
       0,0,0,0,
       0,0,0,0,
       0,0,0,0,
       0,0,0,0,
       0,0,0,0)
u = matrix(ze,4,6)
uPlus = matrix(ze,4,6)
n = matrix(ze,4,6)
```

```
###initial run
for (i in 1:iter)
  {
  for(j in 1:len)
  {
    u[i,j]=rbinom(1,1,cust[i,j])
    n[i,j]=n[i,j]+1
  }
}



for(t in 1:(Target_pop/iter - len))
{

for(i in 1:iter)
{
  for(j in 1:len)
  {
    uPlus[i,j]=u[i,j]+sqrt(2 * log10(t)/n[i,j])
  }
  index=which.max(uPlus[i,])
  play=rbinom(1,1,cust[i,index])
  u[i,index]=(u[i,index]*n[i,index]+play)/(n[i,index] + 1)
  n[i,index] = n[i,index] + 1
}
}

n
u
```