OLLSCOIL TEICNEOLAÍOCHTA
BHAILE ÁTHA CLIATH

TECHNOLOGICAL
UNIVERSITY DUBLIN

Student Name:	***Pili Yuan***
Student No:	***C17129329***
Class:	**DT302-4**
Module:	**MRKT-4109**
Title:	**Mobile Game A/B Testing - Cookie Cats**
Date submitted	**6th May 2022**
Lecturer's Name	***Dr. QIANRU SHANG***

Abstract

This report analyses the retention of two different versions of puzzle game Cookie Cats. The two versions have different gate Settings, with gate Settings at Level 30 and Level 40 respectively. A/B Test and Bootstrap analysis were performed on 1-day retention and 7-day retention of Cookie Cats in the report. This report analyses the player retention of puzzle game Cookie Cats and concludes with a recommendation on whether Cookie Cats should not change gate from Level 30 to Level 40 in order to increase and prolong the enjoyment of the game.

Introduction

Puzzle game, also known as the decryption game, refers to the puzzle to solve the main content of the intelligence game, is a kind of video game. It involves solving a given problem using a variety of techniques, including logical reasoning, spatial imagination, pattern recognition, sequence solving, etc. It may also involve movement elements or require ingenuity.

Background

1. Cookie cats

Cookie Cats is a hugely popular mobile puzzle game developed by Tactile Entertainment. It's a classic "connect three"-style puzzle game where the player must connect tiles of the same colour to clear the board and win the level. As players progress through the game's levels, they occasionally encounter levels that force them to wait an indefinite amount of time or make in-app purchases to progress. In addition to driving in-app purchases, these gates serve an important purpose of giving players a forced break from the game, hopefully increasing and prolonging their enjoyment of the game. But where do these gates go? Originally, the first gate was placed at level 30, but in this notebook we will analyse an A/B test where we move the first gate of Cookie Cats from level 30 to level 40.

2. A/B Test

A/B testing is A general framework for hypothesis testing between two groups to determine causality between actions and results, measuring impact only from change (Siroker and Koomen). A complete A/B test consists of the following parts:

1. Analyse the current situation and establish assumptions: analyse the business, determine the highest priority improvement points, make assumptions and put forward optimization suggestions.
2. Set indicators: set the main indicators to measure the pros and cons of the version; Set up auxiliary indicators to assess other impacts.
3. Design and Implement: Design the optimised version of the prototype and complete the Implement.
4. Determine the duration of the test: Determine the duration of the test.
5. Determine the shunt scheme: determine the shunt proportion and other details of each test version.
6. Collect and analyse data: collect experimental data and judge the effectiveness and effect.
7. Give the conclusion:
    a. Determine to release the new version;
    b. Adjust the shunt ratio to continue testing;
    c. Re-develop the optimised iterative scheme and return to Step 1.

The situation is now fully informed and the next step is Set Indicators. The objective is to improve retention rates, conversion rates and user engagement. We need to decide if Cookie Cats should move the gate from level 30 to level 40.

Design - Hypothesis

Hypothesis:

$$\text{Null: } p30 = p40$$

$$\text{Alternative: } p30 > p40$$

Implement

1. EDA of Dataset

First, import the required library *pandas* and *numpy*, and then read the data set. The data we have is from 90,189 that are added together from both gate_30 sitting at 44700 users and gate_40 sitting at 45489 users, these are players that installed the game while the AB test was running.

The variables are:

- userid - a unique number that identifies each player.
- version - whether the player was put in the control group (gate_30- a gate at level 30) or the group with the moved gate (gate_40 - a gate at level 40).
- sum_gamerounds - the number of game rounds played by the player during the first 14 days after install.
- retention_1 - did the player come back and play 1 day after installing?
- retention_7 - did the player come back and play 7 days after installing?

	userid	version	sum_gamerounds	retention_1	retention_7
0	116	gate_30	3	False	False
1	337	gate_30	38	True	False
2	377	gate_40	165	True	False
3	483	gate_40	1	False	False
4	488	gate_40	179	True	True

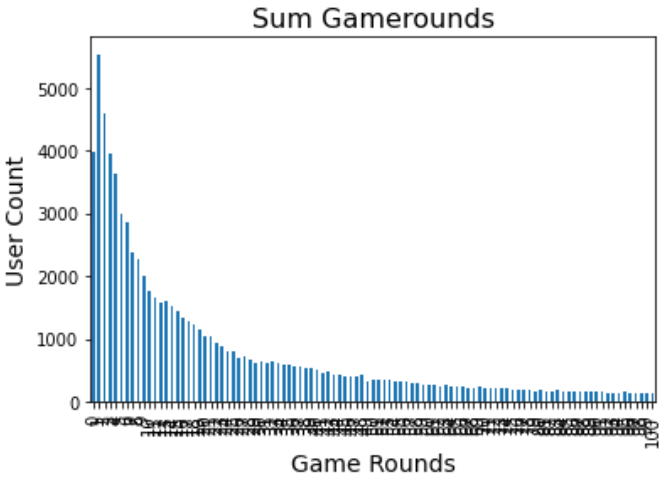
2. Sanity Check

Then we take a look at the number of data in each A/B group. Grouping the data by version (gate_30 and gate_40). As shown in the figure.

version	userid	sum_gamerounds	retention_1	retention_7
gate_30	44700	44700	44700	44700
gate_40	45489	45489	45489	45489

Looks like the number of data in each A/B group are roughly the same which is good Because the two groups have the same sample size, the performance will be better. But since our samples are not identical, we resampled them later with Bootstrap for future analysis.

3. *Distribution of Game Rounds*



- Counting the number of players for each number of game rounds.
- The distribution of players that played 0-100 rounds is plotted.

The output shows the data is highly skewed to the right. Some users downloaded the game but never played (0 rounds). Many users give up after a few rounds of playing. And some of them got hooked and played a lot. The goal of video games is to get the users hooked, so the retention rate can tell the most story. The higher the retention rate is, the easier to retain the users and build a large user base. Next we have a look at the 1-day retention rate.

Retention

1. *1-day retention*

1-day retention is a common metric in the video gaming industry for how fun and engaging a game is. Which means the percentage of players that comes back and plays the game one day after they have installed it.

```
retainers_day_1 = cookie_cats[cookie_cats['retention_1'] == True]\
                              ['userid'].nunique()
total_user = cookie_cats['userid'].nunique()
ret_rate = retainers_day_1/total_user
print(round(ret_rate*100,2), '%')
```

✓ 0.8s

```
44.52 %
```

Overall 1-day retention is 44.52%, almost half of the users came back one day after installing the game.

Now we can take a look at the 1-day retention rate in A/B groups.

```
Group A gate_30 1 day retention rate:  44.82 %
Group B gate_40 1 day retention rate:  44.23 %
```

The 1-day retention rate slightly decreased when the date was moved to level 40 (44.23%) compared to the control when it was at level 30 (44.82%). It is just a small decrease, but it can also make a huge impact in real life retention. Furthermore, it's just the first day, many users haven't even got to level 30, so it's hard to decide which group performed better on this stage. But after 7 days, users should have already reached level 40 which would be more meaningful for our analysis. Next let's take a look at 7-day retention.

2. 7-day retention

```
Group A gate_30 7 day retention rate:  19.02 %
Group B gate_40 7 day retention rate:  18.2 %
```

As with the 1-day retention rate, we see that the 7-day retention rate is lower when the gate is at level 40 (18.2%) than when the door is at level 30 (19.02%). The difference is even larger than for 1-day retention rate. Also, we can see that after a week of install, fewer and fewer people are playing the game as the retention rate decreases all the way from 44.82% to 19.02% for Group A gate_30, 44.23% to 18.2% for Group B gate_40. User engagement dropped significantly after seven days. Since user engagement dropped over time, we can come to another conclusion that when the user is playing the game, the level that they are playing on also impacts their engagement.

Bootstrap

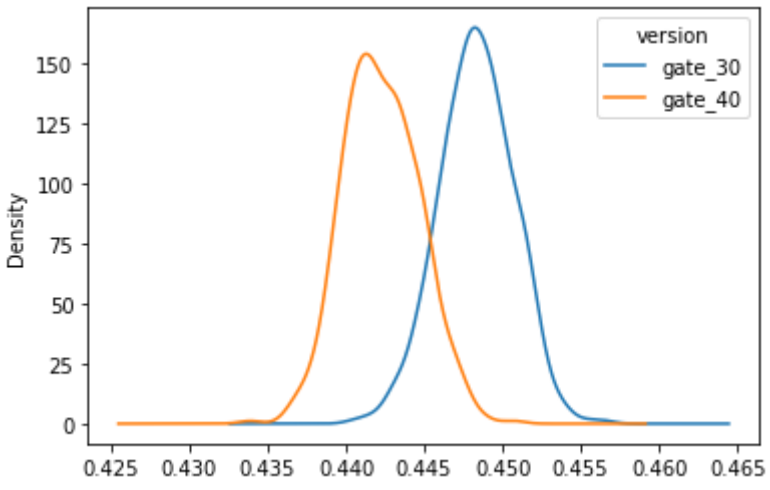
1. 1-day retention

In order to get the certainty of these retention numbers, we will repeatedly resample the dataset 500 times (with replacement) and calculate the 1-day retention and 7-day retention for those samples. The variation in 1-day retention will give us an indication of how uncertain the retention numbers are. We will use bootstrap - sampling with replacement.

First, creating lists with bootstrapped means for each A/B group. Then transforming the list into DataFrame as follows. To do this in python, we need to import resample from sklearn.utils. For this task, we resample 500 times with replacement, but in real life situations it could be much more like 10,000 times.

version	gate_30	gate_40
retention_1	0.448054	0.439051
retention_1	0.446286	0.440150
retention_1	0.451141	0.444965
retention_1	0.448121	0.443536
retention_1	0.446488	0.443382
...	...	...
retention_1	0.446711	0.444349
retention_1	0.447718	0.441623
retention_1	0.448210	0.444525
retention_1	0.449351	0.444987
retention_1	0.447405	0.439711

Kernel Density Estimate plot was made to show the bootstrap distribution. It represents the bootstrap uncertainty over what the underlying 1-day retention could be for the two A/B groups.



Looking only at this figure, we can see that there are indeed some differences, albeit small, between A/B groups. To be more clear, let's zoom in on the difference in 1-day retention.

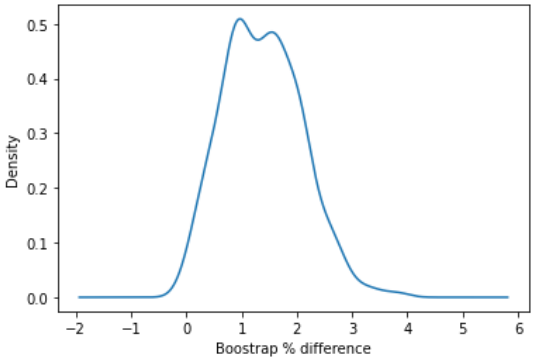
2. *Difference in 1-day retention*

PERCENT DIFFERENCE FORMULA

$$p = \frac{|a - b|}{(a + b) \div 2} \times 100$$

According to the Percent difference formula above, the percent difference between A group and B group can be calculated. And graphed as follows:

version	gate_30	gate_40	diff (%)
retention_1	0.448054	0.439051	2.029634
retention_1	0.446286	0.440150	1.384416
retention_1	0.451141	0.444965	1.378459
retention_1	0.448121	0.443536	1.028424
retention_1	0.446488	0.443382	0.698030
...	...	...	...
retention_1	0.446711	0.444349	0.530205
retention_1	0.447718	0.441623	1.370648
retention_1	0.448210	0.444525	0.825607
retention_1	0.449351	0.444987	0.976036
retention_1	0.447405	0.439711	1.734661



The plot above shows that the more likely percent difference is around 1% to 2%, and most of the distribution is above 0%. But what is the probability that the difference is above 0% when the gate is at level 30?

```
boot_1d['diff'] = boot_1d['gate_30']-boot_1d['gate_40']
prob_1d = (boot_1d['diff'] > 0).sum() / len(boot_1d['diff']) * 100
print(prob_1d , '%')
```

✓ 0.1s

```
96.6 %
```

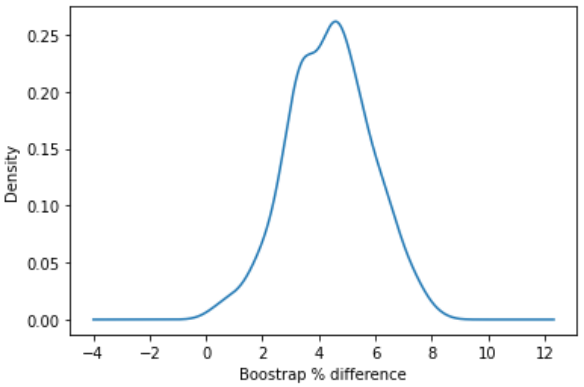
The bootstrap analysis shows that the probability for 1-day retention is quite high (96.6%) when the gate is at level 30. However, we want to prove the hypothesis more with 7-day retention.

3. 7-day retention

We perform the same bootstrap method with 7-day retention.

1. Creating a list with bootstrapped means for each AB-group
2. Transforming the list to a DataFrame
3. Adding a column with the percent difference between the two AB-groups
4. Plotting the bootstrap percent difference
5. Calculating the probability that 7-day retention is greater when the gate is at level 30

version	gate_30	gate_40	diff (%)	diff
retention_7	0.189038	0.182813	3.347898	0.006225
retention_7	0.191745	0.182879	4.733055	0.008866
retention_7	0.192640	0.181582	5.909570	0.011057
retention_7	0.191633	0.181670	5.337651	0.009963
retention_7	0.188031	0.179978	4.376928	0.008054
...	...	...	...	...
retention_7	0.187919	0.181692	3.369582	0.006227
retention_7	0.190313	0.181341	4.828503	0.008973
retention_7	0.188971	0.179714	5.021711	0.009257
retention_7	0.192506	0.182396	5.393338	0.010110
retention_7	0.191432	0.182418	4.822283	0.009014




```
boot_7d['diff'] = boot_7d['gate_30']-boot_7d['gate_40']
prob_7d = (boot_7d['diff'] > 0).sum() / len(boot_7d['diff']) * 100
print(prob_7d , '%')
```

✓ 0.1s

```
99.8 %
```

The bootstrap results for 7-day retention shows that retention is much higher when the gate is at level 30 than when the gate is at level 40. The probability was 100% for the first 5 times I ran the code, then 99.8% is the only time I got that is not 100%.

Conclusion

Both the normal retention rate and the bootstrap analysis show that there is strong evidence that 7-day retention is higher when the gate is at level 30 than when it is at level 40. The decision is **not** to move the gate from level 30 to level 40 if we want to keep retention high (

both 1-day and 7-day retention). One might imagine that the further back the gate is set, the more interesting it is. But the analysis tells us otherwise. Over time, players become engaged with the game. Having a gate at a certain level to force players to rest increases retention. It's not always better to set the gate too late. If the gate is set too late, as in the example of Level 40, the player is likely to get tired of the game before they reach the gate. So it's important to be smart about setting up the gate.

In terms of analysis methods, besides using bootstrap to test A/B test, we can also calculate p-value. If p-value is greater than 0.05, null hypothesis is rejected. If p-value is less than 0.05, null hypothesis is accepted.

Of course, we can also look at other parameters, such as the number of game rounds or the built-in payment model of the game for both AB groups. But the retention rate is one of the most important indicators. If we can't retain our player base, then there's no point in spending as much money as they have in the game.

Reference

Siroker, Dan, and Pete Koomen. *A/B Testing: The Most Powerful Way to Turn Clicks Into Customers*. Wiley, 2013. Accessed 6 May 2022.