Monitoring Game Updates

Character balances for the online game—Battlerite

1. Introduction

For this project, I would like to aid game developers in evaluating character balances for online multiplayer arena-based games. Often, games will have large character disparities in which some will be extremely powerful and others will be far too weak. This is undesirable for both the developers, because players prefer games where they can play a character without fear of being at a disadvantage, so I would like to evaluate character balances by monitoring win-rates and pick-rates. Specifically, I will be working with data from a game called Battlerite.

Battlerite is a quick team-based arena game with many characters for players to choose from. Once the players pick their character of choice, they are randomly paired into teams and enter an arena with the objective of defeating the other team in combat. The first team to win 3 total rounds, wins the match. The game has two main formats, casual and ranked; and two game modes, two-verse-two and three-verse-three, which I will abbreviate as 2v2 and 3v3 accordingly.

The formats and game modes result in very different styles of play. The casual format is for players who want to play for fun or try out new characters. The ranked format, on the other hand, is a more serious test of skill where players try to climb through the following leagues: Bronze, Silver, Gold, Platinum, Champion, and Grand Champion. The 2v2 game mode is more dependent on individual player skill, while the 3v3 game mode is more dependent on teamwork.

Because of the different playstyles within the game, I believe the problem of character balances relies heavily on the format, casual or ranked, and game mode, 2v2 or 3v3. I will evaluate each character's win-rate and pick-rate overall, by format, and by game mode. This will allow developers to see which characters have advantages or disadvantages for each format and each game mode. I will then create several time-series plot examples to evaluate

how win-rates and pick-rates for characters change over time. This allows developers to implement balance changes and monitor the difference in win-rates and pick-rates over time.

To achieve all of this, I will pull match data using Battlerite's API, then store the data in a DataFrame with columns to determine the winner as well as which characters, formats, and game modes were played. I will store all of my work in Jupyter Notebooks written in Python 3, and notate my work in Markdown for easy readability.

2. Data Wrangling

2.1. Data Collection

I began by exploring the API, and familiarizing myself with the structure of the data. Once I understood the structure of the data, I created a function to extract the necessary information and store them in an easily accessible DataFrame. I then created another function to get a chunk of data using a specified date. Using these functions I collected data from various days and hours for the months of January 2018 to April 2018. It is important to note that I only collected data for the first week of April, because of the date I began this process, April 10th.

2.2. Character Mappings

Once I finished collecting my data, I created a mapping between the character IDs and their corresponding names using two files I found through the official documentation; a json file, **gameplay.json**, and an ini file, **English.ini**. The **gameplay.json** file includes two IDs for each character, a 'typeID' and a 'name'. The 'typeID' is the ID stored in my collected data, while the 'name' is an ID in the **English.ini** file that can be used to find each character's name.

I built a DataFrame using the **gameplay.json** file to store each character's 'typeID' and 'name' ID, then built a second DataFrame using the **English.ini** file to store the mappings of each 'name' ID and its value. I then merged these two DataFrames using an inner join to map each 'typeID' with the correct name.

3. Exploratory Data Analysis

3.1. Check if the match data includes every character

Next, I began to explore my data. I needed to ensure that the data included as many characters as possible, so I did a quick assessment and discovered that there were many characters who were not included in the dataset. Many characters were missing in the higher leagues, especially in the 3v3 game mode. The following characters were found to be missing:

```
Format : bronze
Game mode: overall
   Missing characters: None
Game mode: 2v2
   Missing (amount): 1
   Missing characters: Taya
Game mode: 3v3
   Missing (amount): 2
   Missing characters: Lucie, Varesh

Format : silver
Game mode: overall
   Missing characters: None
Game mode: 2v2
   Missing characters: None
Game mode: 3v3
   Missing (amount): 1
   Missing characters: Varesh

Format : platinum
Game mode: overall
   Missing characters: None
Game mode: 2v2
   Missing (amount): 1
   Missing characters: Taya
Game mode: 3v3
   Missing characters: None

Format : diamond
Game mode: overall
   Missing characters: None
Game mode: 2v2
   Missing (amount): 1
   Missing characters: Zander
Game mode: 3v3
   Missing (amount): 3
   Missing characters: Blossom, Iva, Pestilus

Format : champion
Game mode: overall
   Missing (amount): 9
   Missing characters: Ashka, Bakko, Blossom, Destiny, Jamila, Jumong, Oldur, Sirius, Ulric
Game mode: 2v2
   Missing (amount): 11
   Missing characters: Ashka, Bakko, Blossom, Destiny, Jamila, Jumong, Oldur, Pestilus, Sirius, Ulric, Zander
Game mode: 3v3
   Missing (amount): 20
   Missing characters: Alysia, Ashka, Bakko, Blossom, Destiny, Ezmo, Freya, Iva, Jamila, Jumong, Lucie, Oldur, Pearl, Poloma, R
aigon, Ruh Kaan, Sirius, Thorn, Ulric, Varesh

Format : grand_champ
Game mode: overall
   Missing (amount): 20
   Missing characters: Alysia, Ashka, Blossom, Destiny, Freya, Iva, Jade, Jamila, Lucie, Pearl, Poloma, Raigon, Rook, Ruh Kaan,
Shifu, Sirius, Taya, Thorn, Ulric, Varesh
Game mode: 2v2
   Missing (amount): 20
```

3.2. Import newly collected data, and repeat

Because my first collection of data was so sparse, I decided to spend more time collecting data. I repeated a similar process, included a few more days and hours to my collection, and assessed the number of missing characters in my newly acquired dataset. This time only one combination had missing characters, which was satisfactory for my brief analysis:

```
Format : grand_champ
Game mode: overall
   Missing characters: None
Game mode: 2v2
   Missing characters: None
Game mode: 3v3
   Missing (amount): 1
   Missing characters: Jamila
```

4. Character Statistics

4.1. Win rates

Now that I have finished the preliminary steps, I can take a quick look at each character's win rates. To do this, I simply collect the number of wins for each character and divide by the number of times each character was played. In doing so, I made sure to account for each format and game mode combination and ensured that any characters with missing data resulted in a win rate of zero.

4.2. Pick rates

Next, I take a quick look at each character's pick rates. To do this, I simply collect the number of times each character was played and divide by the total number of characters played. I then repeat the same process used to view the win rates. As with the win rates, I made sure to do this for each format and game mode combination and ensured that any characters with missing data resulted in a pick rate of zero.

4.3. Statistics table

After computing the win rates and pick rates, I decided to display the statistics in an easily viewable table. I did this by rearranging the indexes and columns of the win rate and pick rate dataframes, then merging the two together to create the table displayed below:

	stat	win_rate										pick_rate		
	format	overall	casual	ranked	bronze	silver	gold	platinum	diamond	champion	grand_champ	overall	casual	ranked
character	game_mode													
Alysia	2V2	0.493289	0.501329	0.482524	0.432343	0.504769	0.487903	0.464891	0.425926	0.133333	1.0	0.034475	0.031549	0.039363
	3V3	0.503205	0.514226	0.482873	0.517442	0.488333	0.482927	0.477419	0.404255	0.500000	0.0	0.040585	0.037334	0.048352
	overall	0.497414	0.507089	0.482653	0.463158	0.499462	0.485999	0.470263	0.415842	0.333333	0.5	0.036779	0.033895	0.042265
Ashka	2V2	0.494462	0.508079	0.465495	0.469363	0.448759	0.474916	0.480315	0.500000	0.612903	1.0	0.056853	0.061825	0.048548
	3V3	0.508435	0.513175	0.495421	0.468384	0.481959	0.530541	0.483516	0.547170	0.518519	0.0	0.064475	0.067042	0.058343

5. Inferential Statistics

5.1. Create functions

Moving on, I'd like to perform some inferential statistics on the data. To begin, I will have to create two functions.

The first function will simply compute the difference between the desired statistic, win rate or pick rate.

The second function will bootstrap the data several times. Boostrapping refers to a test in which a dataset is resampled, and a test statistic is computed for the resampled data; in this case, the computed test statistic will be the difference function that I just created.

5.2. Hypothesis tests

Now that my functions are complete, I can perform some hypothesis tests. I would like to see if there are any statistically significant differences in the win rates and pick rates between formats and game modes. I believe the ranked format is the most important because

it is played by serious players who care about the game, so it should be no surprise that the game should be balanced around the ranked format. With that said, I will compare the statistics of ranked 2v2 matches to ranked 3v3 matches to determine statistical significance between game modes, and then compare the statistics of bronze league matches to champion league matches to determine statistical significance between leagues.

The results of my hypothesis tests show that there is indeed a difference between 2v2 and 3v3 game modes because 44% of the characters have significantly different win rates, while 78% of the characters have significantly different pick rates. This means that game developers must take game modes into account when they implement balance changes.

In addition, my tests show that there is also a difference between leagues. In comparing bronze league to champion league, I discovered that 55% of the characters have significantly different win rates and 85% of the characters have significantly different pick rates. This means that game developers must take player skill into account when they implement balance changes.

6. Monitoring Changes

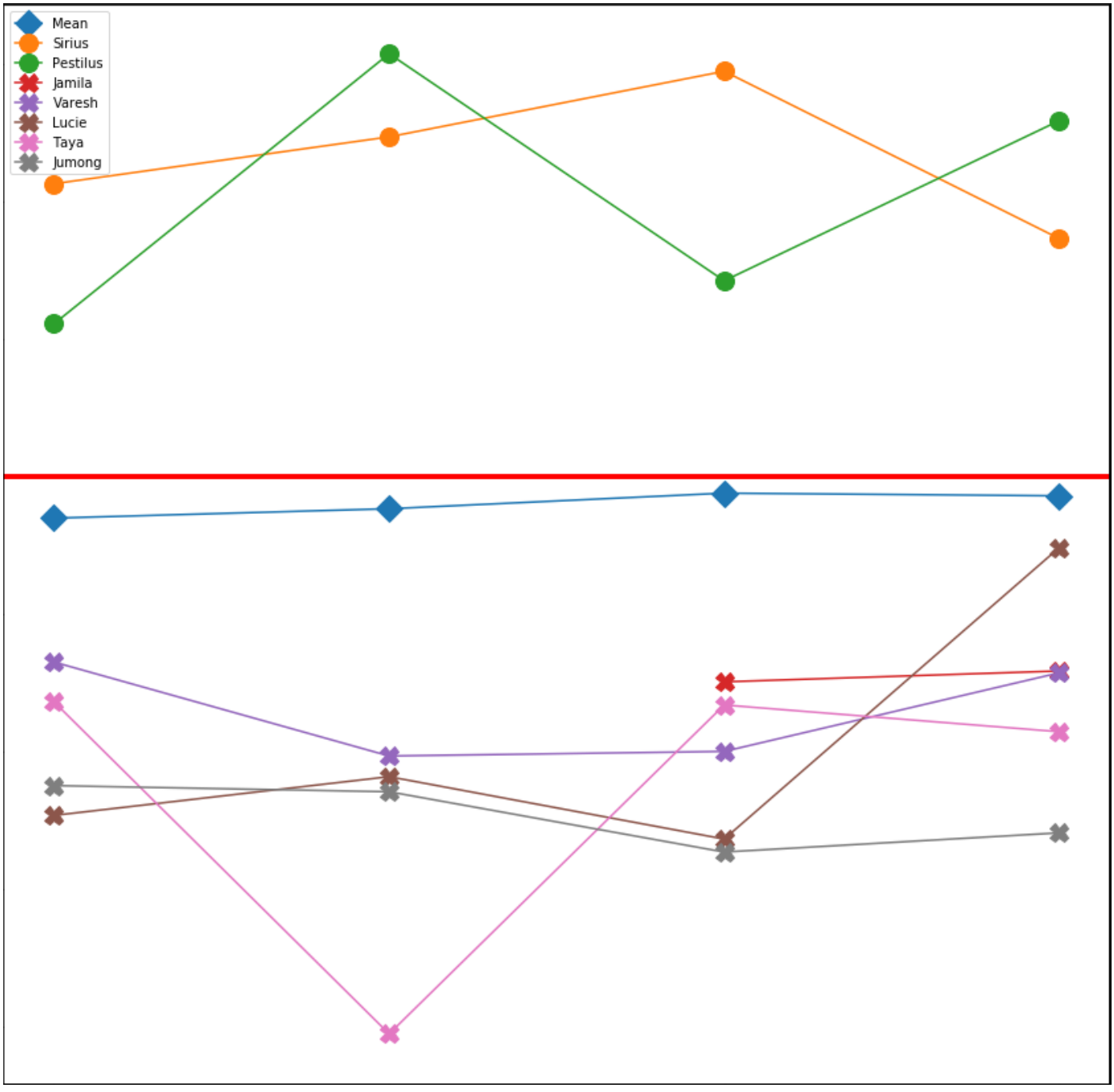
6.1. Find imbalanced characters

The last part of my project involves rooting out problematic characters, and implementing updates to fix them.

To begin, I will look for characters with excessively high and excessively low win rates, which I deem to be at least one standard deviation from the mean. I will then plot the monthly win rate of each character. Recall that the ranked format is the most important format to monitor for balance changes because it covers the player base most committed to the game, thus I will continue to focus on the ranked format.

The results show that the following characters are imbalanced, by my standards:

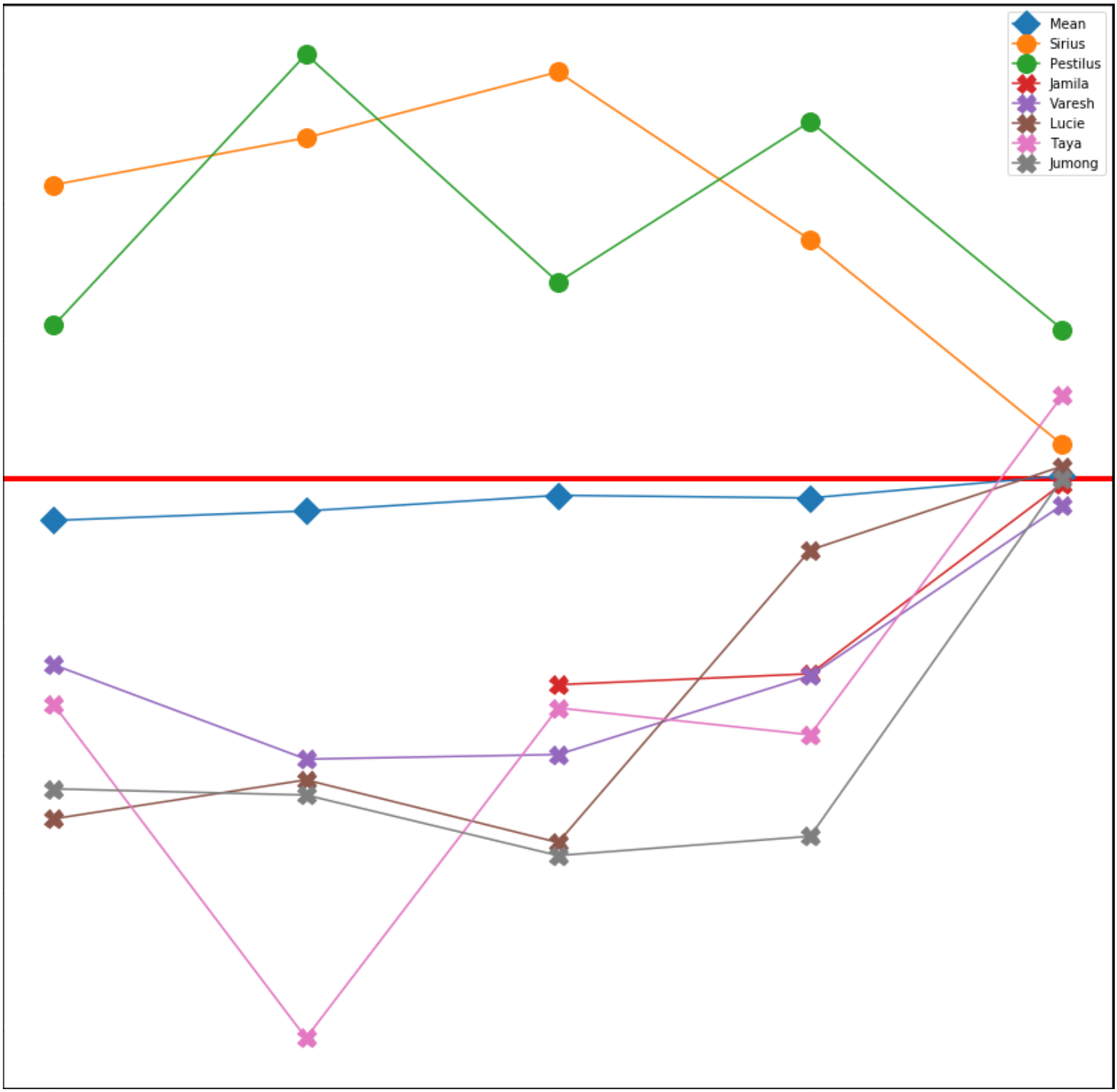
```
Excessively high win rates:
 character  game_mode
Sirius     overall      0.547324
Pestilus   overall      0.540463
Name: ranked, dtype: float64

Excessively low win rates:
 character  game_mode
Jamila     overall      0.470655
Varesh     overall      0.465686
Lucie      overall      0.458049
Taya       overall      0.453735
Jumong     overall      0.451389
```



6.2. Simulated updates

Now that I have determined which characters need balancing, I will simulate an update by creating random data, then plot the monthly win rates to monitor the changes to each character. To keep things simple, I will only simulate ranked matches since we are only concerned with monitoring ranked matches.

My simulated update appears to have been successful for the most part.



Pestilus remains quite strong with a win rate greater than 52%, and requires a bit more work to bring him in line with the other characters.

Taya on the other hand, has been improved far too much, and will need some tweaking to stabilize her improvements.

The remaining characters of interest have been balanced quite well and are much closer to a 50% win rate. The average win rates of all characters is now at 50%, meaning that the characters are much more balanced overall.

This simulation demonstrates how game developers can implement balance changes to specific characters, and monitor the effect of such changes.

7. Conclusions

I have pulled data from the Battlerite API, and created a table to easily view the win rates and pick rates of each character by format (casual, ranked, etc.) and game mode (2v2, 3v3), which will allow game developers to get an overall sense of each character's current standing.

I then determined that there is a statistically significant difference between the win rates and pick rates of characters between game modes and leagues, which is an extremely important piece of information. This lets game developers know that they need to target different characters in different ways, so that there is no character bias between game modes or player skill.

I then determined which characters needed balance changes, and created a method to visualize their win rates over time, allowing game developers to monitor the most imbalanced characters before and after implementing updates.

Overall, I believe my work will prove extremely useful because it allows game developers to get a quick glance at the win rates and pick rates of each character in general, and allows them to identify specific characters that need immediate changes.