League of Legends: Item Balancing
Capstone Project 1 Milestone Report

Sebastian Alvis

Problem Statement

League of Legends is a Multiplayer Online Battle Arena (MOBA) game. It is played by two five-player teams of champions (the in-game characters) racing to destroy the other team's primary building: the nexus. Each nexus is protected by two towers directly in front of it and three lanes of three additional towers and one other building (called the inhibitor) each. The three lanes (known as top, middle, and bottom lane) from each base connect together to form the map. The area between the lanes is known as the jungle, and contains neutral (not on either team) monsters.

Throughout the game, players earn gold in several ways: passively over time, through killing lane minions, jungle monsters, and champions, or destroying neutral objectives, like towers and dragons. Gold is used to purchase items to give the player's champion various stat boosts and abilities. Items form one of the pillars of League of Legends gameplay; understanding which items are useful on which champions is a constantly changing topic as champions and items can be tweaked on a weekly basis.

For the developers of Riot Games (the studio who created and maintains League of Legends), the problem becomes **what items should be changed and why?** This is a task of game balance. They have a strong incentive to keep their game as fair as possible. Fairness in champions and items fosters diversity in which of these are picked in games, and this is where the game is supposed to shine. A lack of diversity can make the game feel stale, and cause players to stop playing and spending money on the game. Making sure items are balanced is probably an ongoing task at Riot Games.

Dataset Description

There are three tables I acquired for this project. The first two are the official lists of champions and items, and were acquired from the Riot Static API / Data Dragon as JSON files, then normalized into tables. These mostly serve to increase readability, and can be joined to the main table. The information in these tables may change from patch to patch (~1-2 weeks), so it may need to be updated if I pick a different set of games.

The JSON format required that I flatten with only the values of the dictionary, and I tacked on the keys afterwards. Additionally, the items' tag column is preserved in the csv format as a unicoded string that looks like a list. I have a line of regex to make it a usable list of strings, but it needs to be reapplied every time the file is read into Pandas.

The champion table contains 141 champions, and I used the same regex fix as above for the champion tags, but did not require any further cleaning.

The item table contains 237 items, of which 80 will be in the final analysis. This involved a boolean column made to filter items that did not build into anything else (full items being the focus of this project), some edge cases for the column, and filtering out other sets of items that are harder to quantify the performance for (Boots, Trinkets, Consumables).

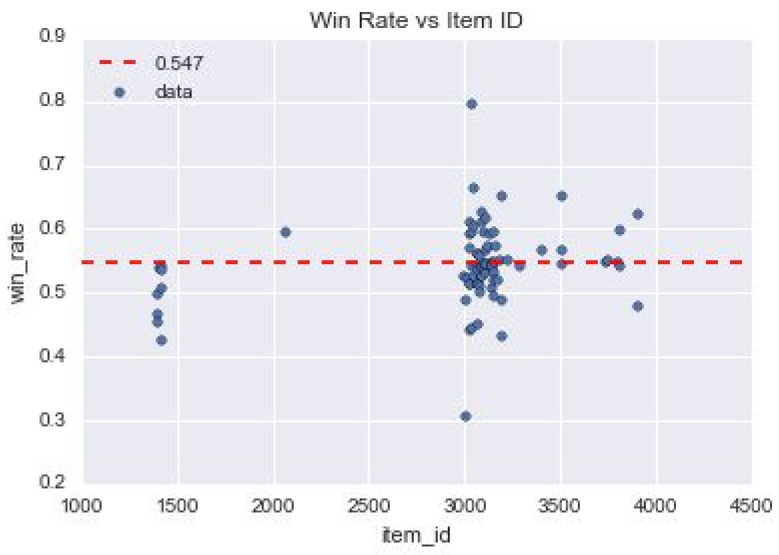
The last table is composed of match-level data. Finding specific matches was fairly complicated. I had to use 5 requests through the API, going from league IDs to summoner IDs to account IDs to match histories to actual games and their timelines. I used a random subset of summoner IDs and a random subset of games from the available match histories to preserve the independence of the data. Acquiring match data had several problems, including the join complexity for the timeline data, missing match data (404), edge case items, and exceeded request rates.

The game data is information on the state of the end of the game, including players' characters, final item builds, kill/death/assist (KDA) ratios, etc. The player data from this requests forms most of the final table I use. The timeline table is used to find the timestamp of purchase for each item in each players inventory at the end of the game. This data was cleaned in creation, and the cleaning was mostly logic for items that wouldn't show up as "purchased" in the timeline table. My workaround was essentially a blacklist.

The match table has information for 10 players in each of 925 matches, making 9250 rows.

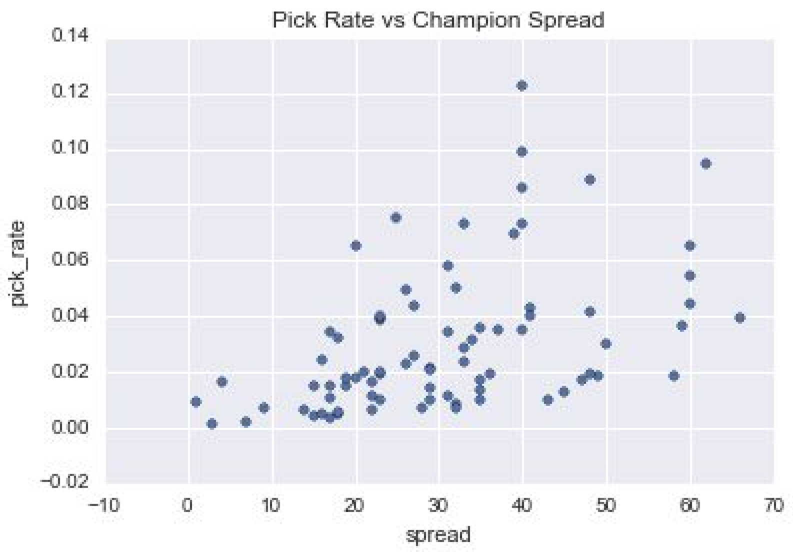
Initial EDA

In the match data, each player builds up to 7 items (of which up to 5 will be relevant to the final analysis, accounting for Boots and Trinkets). From this, data was aggregated on an item-indexed basis to get descriptive statistics on how often items were picked and how often players won games when they picked the item. The lengths of the sets of unique champions to build each item were also recorded. These three metrics - pick rate, win rate, and champion spread, respectively - formed the main metrics by which I would measure item performance and popularity.

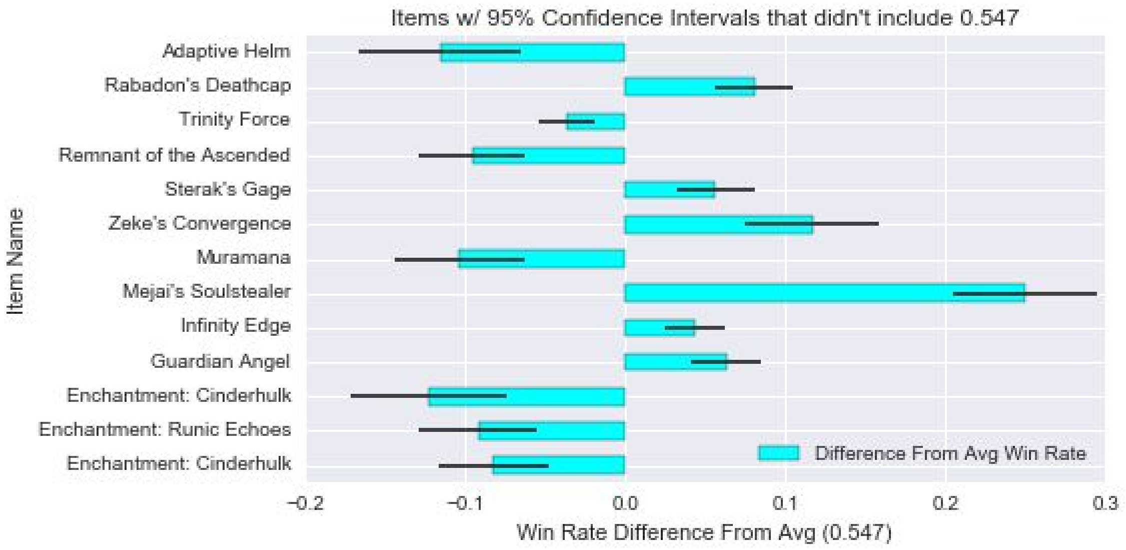


The most relevant variable for telling whether an item is too strong or too weak is the win rate. If an item has an abnormally high win rate, then it is too strong and should be weakened. Conversely, if an item has an abnormally low win rate, then it is too weak and should be strengthened. Both of these can be tested with a simple z-test against the average win rate of all full items and a 95% confidence interval on the true win rate. This is especially easy because the win rate is a Bernoulli variable, as the proportion of games won.

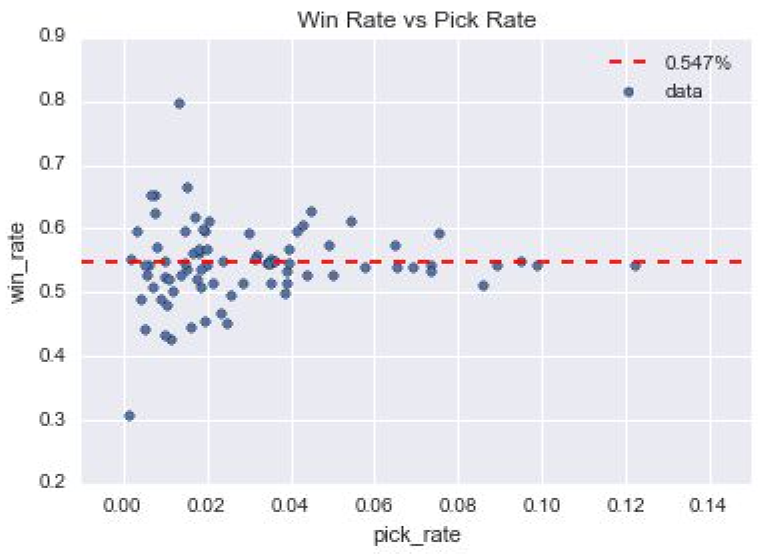
The next two variables to be explored are pick rate and champion spread. An item's popularity can indicate strength, and if a player on each team in a game build the same item, then the win rate will be moved towards 50%, as only one of the teams can win. Pick rate is the proportion of item slots that have a given item (7 item slots per player) but could be defined to be the proportion of players who purchase an item (I would just multiply all the pick rates by 7). Champion spread is a measure of the set of champions that build a given item, and can indicate an item's excessive or limited versatility, especially if some of the champions building it are unexpected (as determined by champion and item tags).



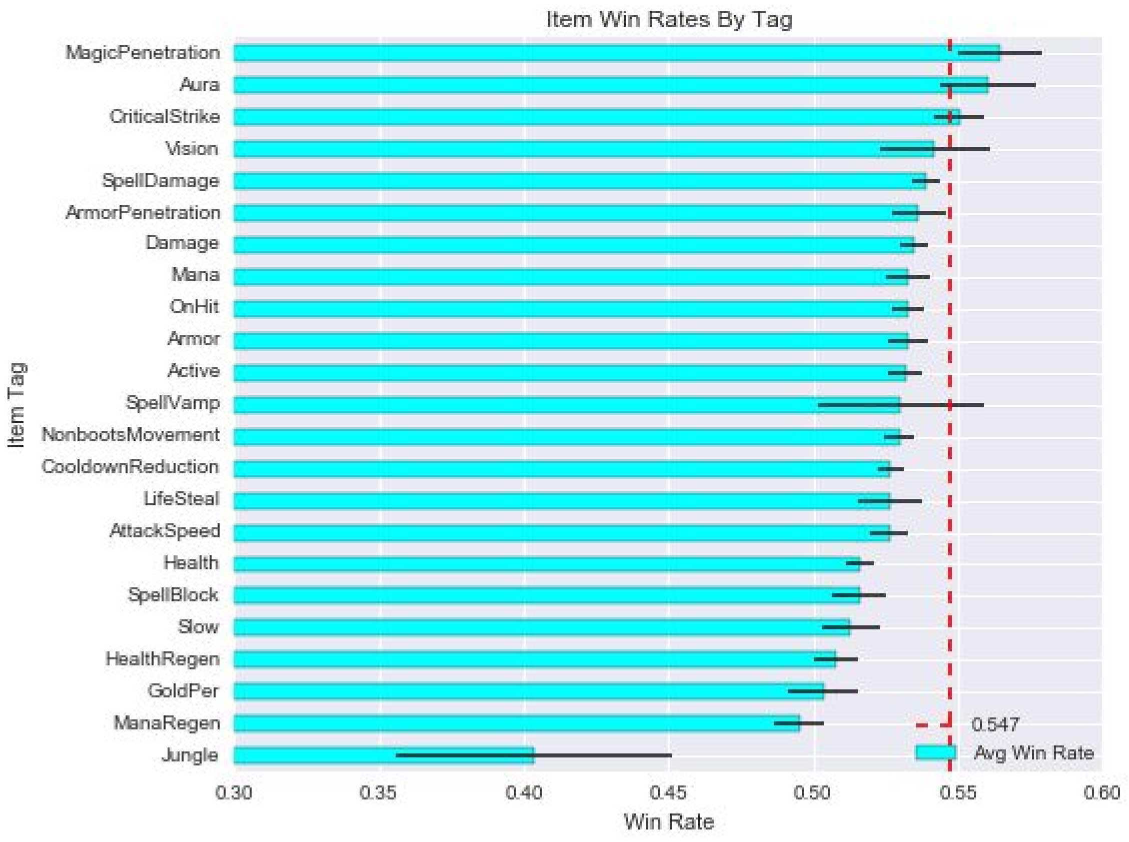
Pick rate and champion spread are expected to correlate. It is important to find correlations to pick rate and win rate, as the two primary metrics for answering the initial question. The highest correlation to win rate is through champion spread, and the highest to pick rate is through champion spread, and then total item cost.



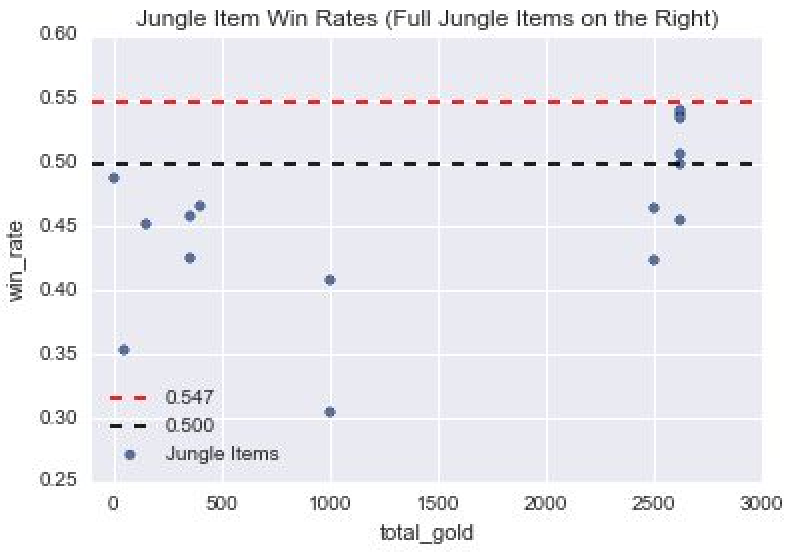
The primary test done here was the z-test for the proportion of wins. If the game is perfectly balanced, then each items' win rate should be the same (null hypothesis). For the data I have, the relevant full items have an average win rate of 0.547. The test is done to see if this value falls within the 95% confidence interval (alpha = 0.05) of each item's individual win rate. 13 of the 80 relevant items had win rates outside their confidence interval, 7 of them too low, and 6 too high. Most notable was an item with a win rate 0.80.



However, I am realizing that I should probably normalize win rate with pick rate. Items that are built by everyone have a win rate pushed to 0.50, but items that aren't built as often can have more expected fluctuations in the win rate. For items with high pick rates, it would be interesting to check how they perform players on both sides don't pick it.



The average win rate for each tag / type of item was tested in the same way, but since the number of data points were much higher, the confidence intervals were more tightly bound. No set of items had win rates that were too high, but several sets of items, most notably the Jungle items, had win rates that were abnormally low, centered around 0.50.



This behavior is different from the rest of the full items, because one player per team per game almost always builds a jungle item, and they generally build the jungle item first. That should enforce a win rate close to 0.50, unlike the rest of the items.

Several questions still remain. I want to find the most important factors in determining pick rate and win rate, but with the amount of data present (especially columns), this is probably easier to do with machine learning. Things that could be analyzed are if items that give certain bonuses or have certain bonus-to-item-cost efficiencies give good win rates, or if items with high pick rates / spread have lots of champions outside the expected set of champion tags, and if the time an item is bought influences its win rate.