League of Legends Item Balancing: Exploratory Data Analysis

Sebastian Alvis

Capstone Project 1

For the project, the question to keep in mind is: How would a developer for League of Legends know when they need to change an item in the game? To this end, I have tables of item stats, match data (specifically what items players ended the game with and when they purchased them), summary statistics on each item, and summary statistics on each type of item (as defined by the tags).

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 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 the champion spread, and the highest to pick rate is through 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.