Sebastian Alvis

Capstone Project 1

Data Wrangling Procedure

Data Selection

I needed a few sets of data for this project.

The first is the items and their information. The item data may vary slightly from patch to patch (~2 weeks), so I may need to update it.

The second is the champions and their information. This is would be mostly for reference, and tags on champion roles (tank, mage, assassin, etc).

The last and largest dataset is the games I will analyze. The game data has item id and champion id numbers, which can be used to join to the other tables.

The game data will need to be chosen more carefully. To try and reduce the amount of variance in player skill level and reduce the expected number of outliers from trolling or throwing games, I will take only match data from the ranked queue, and only ranked games from the Platinum and Diamond leagues. There are 118k players in these leagues, so I will probably have to take a random subset of these players to speed up data acquisition time.

The patch I choose doesn’t matter too much I think, but I should get all the data from the just one patch.

Data Wrangling

The item data was acquired through the Riot Static API / Data Dragon (aka a JSON file) and normalized.

The game data was acquired through the Riot Games API. It was a bit of a process to get to the match data. I found a table online with leagueIds, 1103 of which were in the leagues I wanted. From there, I can get all the summonerIds in these leagueids, get each summonerId’s corresponding accountId, and then get match histories for each account. Then, using the gameId from each match, I can extract match data from a final request and build a table from the matches.

With each match, I made 2 requests. The first got basic match data, including the items that each player ended the game with. The other was a timeline of match events, which I used to get timestamps on each item’s purchase.

Getting match data this way had several problems: the complexity of the join, missing match data (404), and items that weren’t purchased all threw errors, and had to be remedied.

Data Cleaning

Item data was acquired flattened using json\_normalize, with the wrinkle that I had to normalize on the values of the json, then manually set the index of the dataframe to the json’s keys. Item data was then filtered based on the availability on the one map I need, and then I manually selected a subset of columns and renamed them.

Champion data was flattened in the same way as the items, with the same wrinkle. The champion table didn’t need any cleaning, mostly because it is less relevant.

The match data table needed cleaning during creation, since I joined it with some data from the corresponding match timeline table. There were several items that showed up in the final inventories that couldn’t be bought, so I developed a blacklist for them.

No values from any table were missing (as far as I can tell).

I am not sure yet if there are outliers.

Final table should have 935 matches (minus missing matches), with 10 players per match, for a little over 9000 rows.

Some potential sources of bias / other problems:

* I am taking the set of games from a random sample of 1000 players within the 118k players in the leagues I selected. However, if some players are playing considerably more games than others, they may skew the distribution.