Kenton Hoffmann

Report for SI 206 Final Project

GOALS

Originally, my goal was to grab data from videogame leaderboards to compare statistics among the top global players. I wanted to see how the win percentage changed with matches played and whether factors (such as type of gaming system,) affected how well players performed. I needed popular games not only for the competitive aspect, but because big gaming companies usually offer a lot of support for people working with data and are more likely to support developer APIs than smaller companies. Thus, another goal I had was to specifically collect the Fortnite leaderboard data, as this was probably the most popular and widely known game of 2018 and has millions of active players. Only a small percentage of players ever reach the global leaderboards, which means they truly represent only the best of the best.

GOALS ACHIEVED

Although I couldn't collect Fortnite data due to constraints, I was able to get leaderboard data for another popular game, Starcraft 2, instead. Blizzard Entertainment, the creators of Starcraft 2, launched developer API support across several of their games last year, and have made it easy for developers to grab what they need from their website. In fact, there was a request tutorial specifically for leaderboards, which, when requested, returned a json file of the top 200 players with their username, wins, losses, their favorite race to play as, and other miscellaneous statistics. From this, I was able to extract the data in a json format insert it into a Sqlite database and create several visualizations from that table.

PROBLEMS FACED

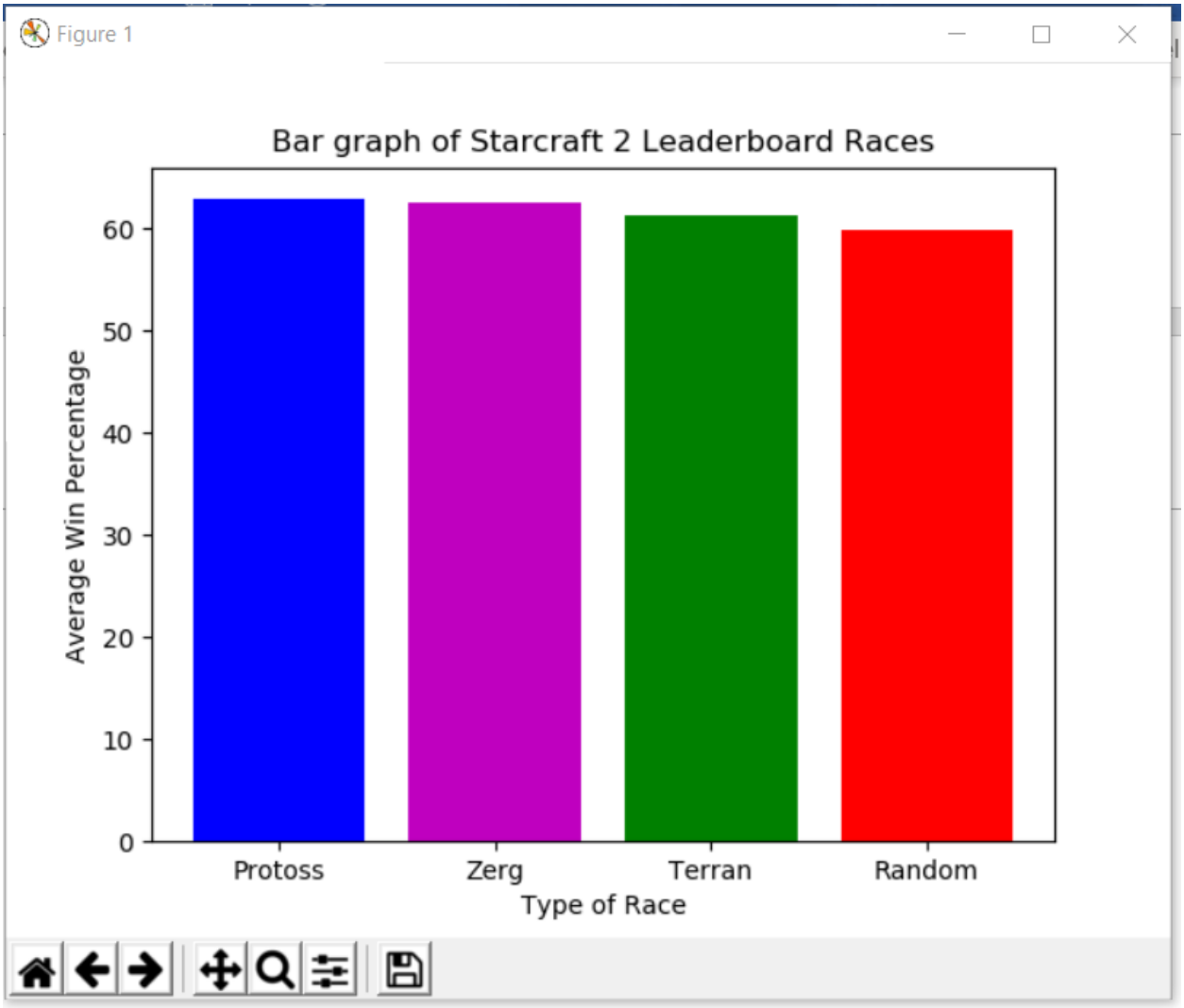
Unfortunately, the main reason that I had to switch my game from Fortnite to Starcraft 2 is because surprisingly, Epic Games doesn't offer official APIs for Fortnite. When I submitted my original plan, I intended to use third party software such as FortniteTracker or Scoutsdk as it appeared that there would be someway to get leaderboard data from their sites. However, the only data they let you access is player specific, meaning that in order to get the top players, I would have to create a different request for each player instead of being able to get all 200 at the same time. Not only would this have been painstakingly slow, but there's a good chance that the website would have limited the number of requests which could have forced me to wait between each request and take several hours, if not days to complete. Furthermore, since none of these APIs are official, when Epic Games updates their game, it conflicts with the way that some of these APIs gather their data and can cause them to stop working completely, which is exactly what happened when they released a new season a few weeks ago.

SOCIAL MEDIA REPORT

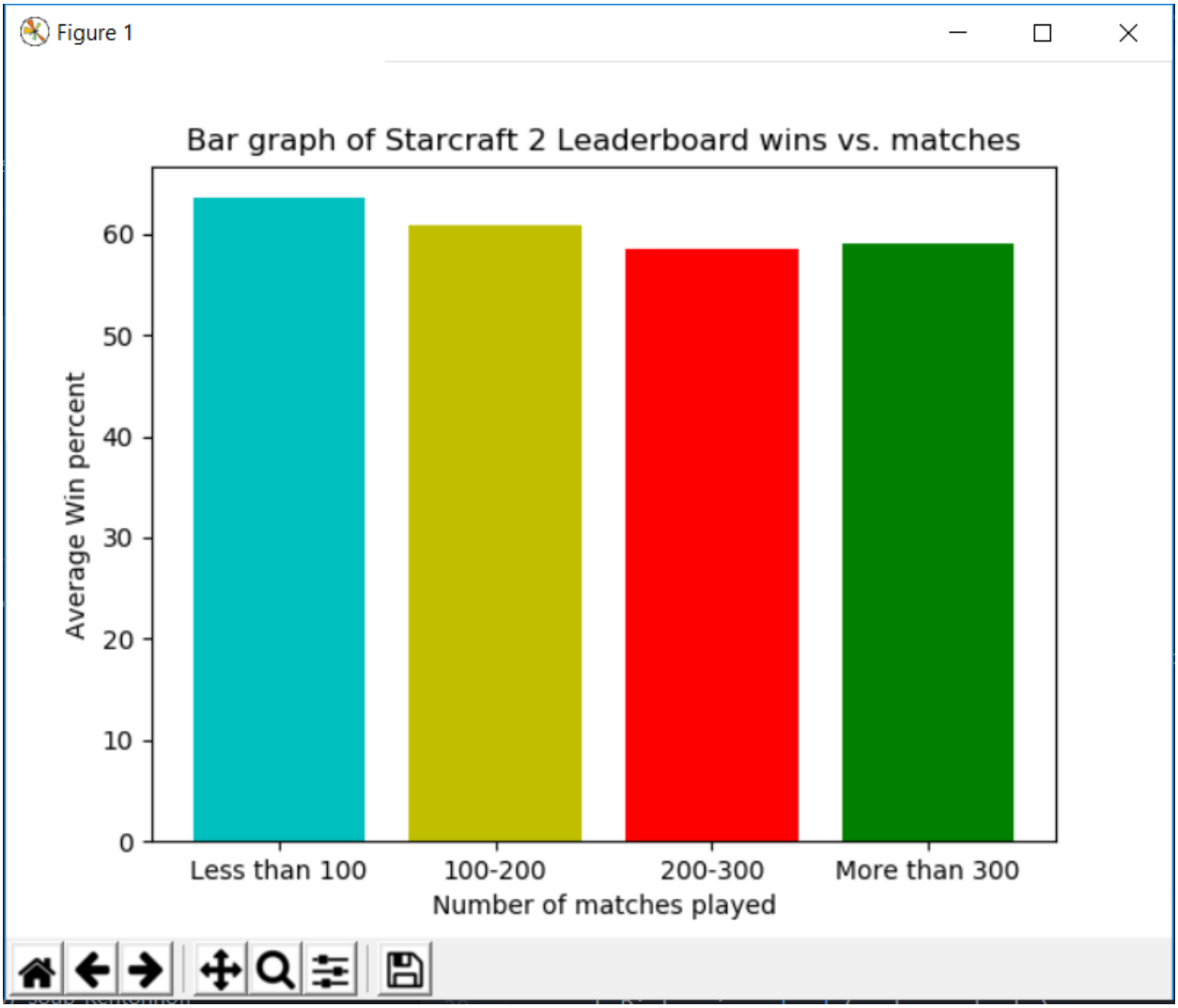
In Starcraft 2, there are three races that players can choose from: Protoss, Zerg, and Terran. I created a list for each race, as well as a random list for players that had no preference which race to play, that contains the win percentages of all the top players who played as that race. Then, I added all the win percentages together and divided them by the length of the list to get the average win percent for each race. The program prints out the number of players for each list, the list's average win percentage, and a bar graph visualizing the results. Additionally, I also created a second function that groups players based on number of matches played instead and calculated the average win percent of each group. This second function only has a bar graph, as I created it as an additional visualization for extra credit.

```
A@LAPTOP-318PP4SB MINGW64 ~/desktop/si206/final-project-Kentonhoff (master)
$ py final.py
Number of Protoss players: 72
Average Protoss win percentage: 62.88293285272677
----------------------------
Number of Zerg players: 56
Average Zerg win percentage: 62.58544349218301
----------------------------
Number of Terran players: 46
Average Terran win percentage: 61.237928897989015
----------------------------
Number of Random players: 6
Average Random win percentage: 59.87401767672106
----------------------------
```

What is printed in the terminal when the code runs.



First bar graph created highlighting win percentages for races.



Second bar graph highlighting win percentages based on the number of matches played.

INSTRUCTIONS FOR RUNNING THE CODE

To run the code, you'll need the main file called final.py and the cache titled "KenStarcraftCache.json". The access token listed for requests only works for 24 hours, and in order to request a new one, I must visit their website and enter my client id and secret. **I discussed this with Sonal Doomra at office hours and I verified that the access token does get the necessary data, and she said that if the program has the json needed to run I should be fine.** The sqlite3 connection needs to be changed to match the directory of the person running the files, as it is set to my computer's directory ('/Users/A/Desktop/si206/final-project-Kentonhoff/Players.sqlite'), but it should be able to create itself once the program runs if the path is correct. If the program runs successfully, the first thing you should see is the bar graph visualizing Races vs win percentage. Once you close this graph, another graph highlighting

number of matches played against win percentage should appear. Close this graph again to see the final output that displays the number of players for each race, as well as the average win percentage for the players of each race.

DOCUMENTATION (SEE DOCSTRINGS IN FINAL.PY FILE FOR MORE INFO.)

The first thing the program does is that it tries to read in the cache from the CACHE_FNAME file. If it doesn't exist, it makes a request to the API which then returns a json file of the top 200 players on the leaderboard. The next function *setupplayertable* sets up the players.sqlite database, creating the table if it does not exist. The function inserts the display name, favorite race, points, wins, and losses of a player into the database. The function then commits the changes to the database. Following the creation of the database, the next function *getrace* selects data from the favorite_race, wins, and losses columns in the database. It prints the number of players for each race and the average win percentage for that race. It also creates a bar graph to visualize this data, with the race name as the x label and win percentage as the y label. The next function

DOCUMENTATION FOR RESOURCES

Date	Issue Description	Location of resource	Result (did it solve the issue)
12/8/2018	Fortnite APIs not working as intended; cannot access leaderboard data	https://develop.battle.net/documentation/api-reference/starcraft-2-community-api	Switched to Starcraft 2 API, which allowed me to grab the desired data from leaderboards.
12/9/2018	Had trouble determining what colors matplotlib allows.	https://matplotlib.org/2.0.2/api/colors_api.html	Changed colors to match the ones listed on the website
12/10/18	Needed better understanding of Sqlite insert	https://www.sqlite.org/lang_insert.html, https://www.sqlite.org/lang_conflict.html	I believe I understand insert, ignore, replace a bit better.
12/10/18	np.broadcast(*args[:32])ValueError: shape mismatch: objects cannot be broadcast to a single shape	https://stackoverflow.com/questions/36668771/axes3d-numpy-arrays-error-while-trying-to-build-3d-plot	When I tried creating my bar graph, I forgot to put a comma between 2 items in a list that I was going to use to label one of the axes. Once I correctly inserted all the commas, the graph constructed properly.

12/13/18	Needed to remember format for docstrings	Debugging.pptx	Created docstrings for each function and initial setup.