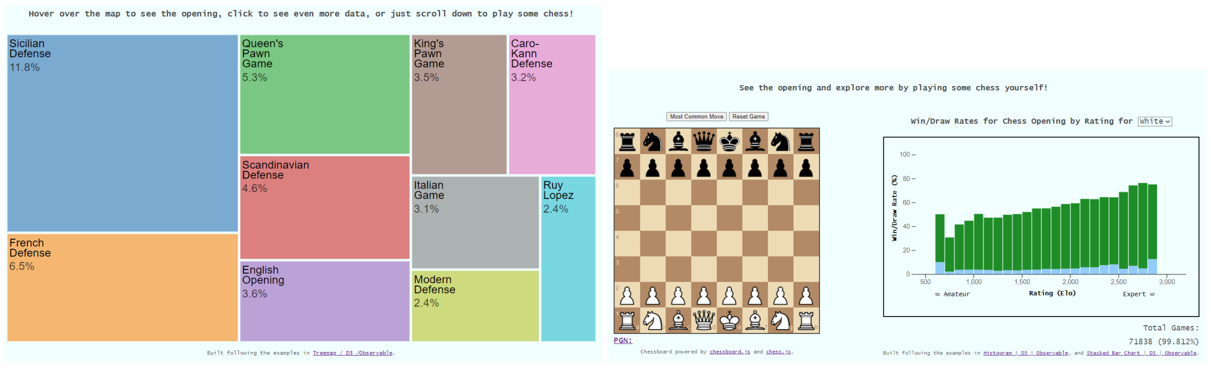
Your Check Mate: How Good is Your Chess Opening?

Justin Lim[*]
MIT

Matthew Tung[†]
MIT



Hover over the map to see the opening, click to see even more data, or just scroll down to play some chess!

See the opening and explore more by playing some chess yourself!

Built following the examples in Treemap / D3 /Observable.

Chessboard powered by chessboard.js and chess.js.

Total Games: 71838 (99.812%)

Built following the examples in Histogram | D3 | Observable, and Stacked Bar Chart | D3 | Observable.

Figure 1: Your Check Mate. Left: Treemap showing the most popular openings in a given skill and time range. Right: Interactive chessboard showing the win and draw rates of each opening.

Abstract

Chess is a complex game with a staggeringly large set of possible positions. Still, the first moves made in each game of chess (the *opening moves*) are very well-studied; most of them even have names. In this paper, we tackle the challenge of producing an interactive visualization that allows users to explore chess openings across several parameters, namely time and player skill. We utilize a treemap and a stacked bar chart to visualize information, such as the popularity and success of different openings. Through interactive elements such as an interactive chessboard, tooltips, and autoscrolling, we created a site to enable a user-friendly exploratory experience, even for users unfamiliar with chess.

Index Terms: Human-centered computing—Visualization—Visualization techniques—Treemaps; Human-centered computing—Visualization—Visualization design and evaluation methods

1 Introduction

Chess is one of the world's oldest and most popular games, whose origins have been traced back as far as the 7th century. Today, chess is played by millions around the world on a daily basis, thanks to the ever-increasing popularity of online chess services, such as Chess.com, which boasts over 20 million users [5], and lichess.org, another popular platform. It has also exploded in popularity over the last year: users on the popular streaming platform Twitch watched 18.3 million hours of chess content in January 2021, nearly as much as was watched in the previous year combined; part of this success may be attributed to the runaway success of Netflix's series *The Queen's Gambit*, which tells the story of a chess prodigy's path to success [11].

The game of chess is extremely complex, but even though there are more than 10^{120} possible chess games [14], the opening moves of the game are extremely well-studied. The Encyclopedia of Chess Openings lists 500 different openings, grouped into categories such as "Flank Openings", "Open Games", and so on [13]. Some openings have been studied to almost thirty moves in, aided by the development of powerful chess engines such as Stockfish [9], which are significantly more powerful than human players.

In this paper, we address the broad challenge of designing an interactive visualization to allow users to explore and learn about chess openings. We identify two axes of interest: skill level, as indicated by the standard chess rating systems, and time. Thus, our goal is to answer the following questions: for a given range of time and player skill, which openings were the most popular? How much success did these players find with these openings? We present a visualization that enables the user to answer these questions. We utilize a treemap and a stacked bar chart to visualize this information. Furthermore, our visualization integrates different types of interaction to enable and guide the user in their own exploration. For instance, users can play out a game of chess on a live chessboard using their own moves, and have the visualization dynamically respond to their play.

2 Related Work

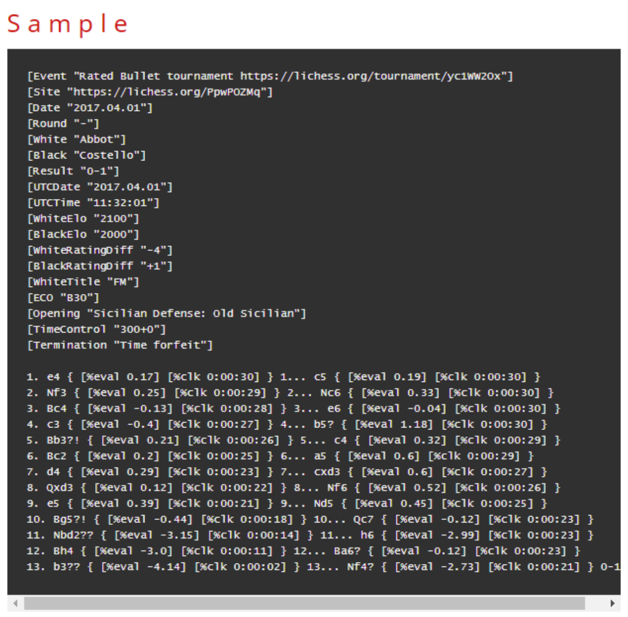
We build upon work from the Javascript libraries chessboard.js [1] and chess.js [2]. The first library providers a visualization of a chessboard, and the second library enables piece movement, validity checking, and other useful features. Our visualization is built entirely in D3.js [3], and various tutorials found on the Observable platform, such as for treemaps [10], histograms [4], and stacked bar charts [8]. Finally, our work relies on the data provided by the lichess.org open database, which contains over 2 billion chess games played on the lichess.org website since January 2013 [6].

3 Methods

3.1 Data Cleaning

The entirety of the lichess open dataset takes up 539GB of disk space in its compressed form. Thus, to make it tractable for an interactive and dynamic visualization, we had to reduce the dataset to contain only the information we needed. Figure 2 shows an example entry from a single game in the dataset. It contains various metadata such as the event the game was played in, the date and time it was played, each player's username and rating, and so on. We distilled this dataset down to the five essential pieces of information we needed:

[*]e-mail: justinl@mit.edu
[†]e-mail: mtung@mit.edu



Sample

```
[Event "Rated Bullet tournament https://lichess.org/tournament/yc1WW2Ox"]
[Site "https://lichess.org/PpwPOZMq"]
[Date "2017.04.01"]
[Round "-"]
[White "Abbot"]
[Black "Costello"]
[Result "0-1"]
[UTCDate "2017.04.01"]
[UTCTime "11:32:01"]
[WhiteElo "2100"]
[BlackElo "2000"]
[WhiteRatingDiff "-4"]
[BlackRatingDiff "+1"]
[WhiteTitle "FM"]
[ECO "B30"]
[Opening "Sicilian Defense: Old Sicilian"]
[TimeControl "300+0"]
[Termination "Time forfeit"]

1. e4 { [%eval 0.17] [%clk 0:00:30] } 1... c5 { [%eval 0.19] [%clk 0:00:30] }
2. Nf3 { [%eval 0.25] [%clk 0:00:29] } 2... Nc6 { [%eval 0.33] [%clk 0:00:30] }
3. Bc4 { [%eval -0.13] [%clk 0:00:28] } 3... e6 { [%eval -0.04] [%clk 0:00:30] }
4. c3 { [%eval -0.4] [%clk 0:00:27] } 4... b5? { [%eval 1.18] [%clk 0:00:30] }
5. Bb3?! { [%eval 0.21] [%clk 0:00:26] } 5... c4 { [%eval 0.32] [%clk 0:00:29] }
6. Bc2 { [%eval 0.2] [%clk 0:00:25] } 6... a5 { [%eval 0.6] [%clk 0:00:29] }
7. d4 { [%eval 0.29] [%clk 0:00:23] } 7... cxd3 { [%eval 0.6] [%clk 0:00:27] }
8. Qxd3 { [%eval 0.12] [%clk 0:00:22] } 8... Nf6 { [%eval 0.52] [%clk 0:00:26] }
9. e5 { [%eval 0.39] [%clk 0:00:21] } 9... Nd5 { [%eval 0.45] [%clk 0:00:25] }
10. Bg5?! { [%eval -0.44] [%clk 0:00:18] } 10... Qc7 { [%eval -0.12] [%clk 0:00:23] }
11. Nbd2?? { [%eval -3.15] [%clk 0:00:14] } 11... h6 { [%eval -2.99] [%clk 0:00:23] }
12. Bh4 { [%eval -3.0] [%clk 0:00:11] } 12... Ba6? { [%eval -0.12] [%clk 0:00:23] }
13. b3?? { [%eval -4.14] [%clk 0:00:02] } 13... Nf4? { [%eval -2.73] [%clk 0:00:21] } 0-1
```

Figure 2: Sample entry from the lichess.org open dataset, representing a game of chess. Screenshot taken from `https://database.lichess.org/`.

1. Date: Since we are interested in changes in chess openings across time, we record the date that each game was played.

2. Result: This is either "1-0", "0-1", or "1/2-1/2", representing a win, loss, and draw for the White player, respectively.

3. WhiteElo, BlackElo: These represent the skill ratings of each individual player according to the Elo rating system [12].

4. Opening: The name of the opening that was played in each game, for instance, the "Sicilian Opening."

5. Moves: The exact sequence of moves that were played in the game, in the format of a PGN. Since we are only interested in the opening of the game, we truncate it to the first 8 moves played, 4 from each player.

In our Github repository, this is done via the Python script `preprocess.py`. We use regular expressions to filter out these irrelevant information, such as the clock status and engine evaluations. We also dealt with inconsistencies in the structure of the data across time. For instance, the datasets from early 2013 had no attribute named "Date"; in these cases, we take the attribute "UTCDate". These were compiled into a TSV format to be easily readable by D3 and Javascript.

Since it is infeasible to consider all 2 billion games in our visualization, we also have to reduce the number of games present in our dataset. To preserve the distribution of games over time, we take the same number of games (2,000) from each month, to obtain a final "small" dataset for each month in TSV format. Each of these "small" files are around 190KB in size. This process is done via the bash script `reduce_data.sh`. The combination of data preprocessing and subsampling enabled us to get a dataset that is representative over time, but still feasible for us to perform computations on. The result is a visualization that is fast and responsive to user input.

3.2 Visual Encodings

We decided on a two-part visualization for this project. The first is a treemap that shows the most popular openings for the selected range of time and rating. The second is a stacked bar chart that shows the win and draw rates for an opening on the interactive chessboard.

3.2.1 Treemap

For our Minimum Viable Prototype (MVP) for this final project, we had chosen a different encoding: a line graph. Upon feedback from our presentation group as well as from peer review, we realized several flaws with this idea. Firstly, it tends to be very cluttered especially for the less common openings, since most of them have similar frequencies. Secondly, we realized it was not friendly to a newcomer to chess, since they do not know the names of specific openings. Thirdly, the trends across time are relatively stable with some notable exceptions, so it would be better if the encoding highlighted those instead.

In response to these reviews, we chose a treemap encoding to visualize the popularity of chess openings over time. We additionally incorporate several interactive elements to it:

1. When the user hovers over a node in the treemap, a tooltip appears showing what the chess opening looks like, on a chessboard. The tooltip is positioned on the page so that it does not obscure the name of the opening.

2. When the user hovers over a node, the cursor becomes a pointer, showing that the user may click on it.

3. Upon clicking a node, the screen scrolls down to the chessboard, where the chessboard is populated with the opening that the user just clicked on.

The treemap elegantly addresses the flaws of our previous visualization, and provides an effective visualization of each opening's popularity and the tooltips provide much needed context.

3.2.2 Interactive Chessboard

One main drawing component that we had from the beginning of this project was an interactive chessboard. Since our problems revolves around the game of chess and specific moves, we thought it was imperative to have some way to visualize the move themselves and be able to see and play with it. Using the aforementioned chess libraries, we were able to integrate a chessboard onto our site and have it interact with the other visualizations while also allowing the user to play chess normally.

In addition to the front end, one way that one can operate with the chessboard API is via PGN, or Portable Game Notation. A standard plain text format for recording chess games, which can be read by humans and is also supported by most chess software [7], we are able to leverage the API to move the board by inputting a PGN, as well as keep track of it when the user makes a move. Since the lichess database also had PGN data, we able to use it to interact with the board and vice-versa.

So when the user interacts with the treemap and clicks on an opening, it then calls a function to load the corresponding opening PGN to the board, allowing for the user to further visualize the opening. The chessboard's PGN also gets sent to next the visualization, acting as a filtering criteria by checking what games start with the opening PGN. When a user makes a move on the chessboard, it also updates this filtering criteria. Since we want to encourage users to explore the data and see popular moves, we also implemented a function that gets the most common next move. Using the current PGN, it iterates through the data and find the most frequent next move and when the button is clicked, it updates the board and plays said move. We also have a reset button for ease of access to reset the board and data as well as a status bar that informs the user who's turn it is, whether or not there's any data that reflects the current game as well as other standard chess rules (Check, Checkmate, etc.).

In addition to just being a fun addition and draw for the user, we included an interactive chessboard to better visualize the opening and move data, and interact with the other graphs via PGN to do so while also allowing user input.

3.2.3 Stacked Bar Chart

We chose a stacked bar chart to show the success rates of each opening, because both wins and draws are significant outcomes for the chess player; in fact, at the highest levels, most chess games end in a draw. We chose to represent wins in green and draws in blue, drawing on their connotations to a positive and neutral outcome, respectively. To demonstrate changes in the win and draw rates in response to player movement, we animate this graph with transitions to illustrate the direction in which the rates change. This allows the user to visualize whether their move resulted in a better or worse outcome across different ratings. The default win/draw rate is for white, but it can also be set to black. When the user filters via the other encoding, the bar chart updates to reflect it.

We also chose to bin the data points by rating points in increments of 100, as it was sufficient to separate the data without cluttering up the visualization. Since the x-axis represents rating, we added "Amateur" and "Expert" markings to inform users that higher ratings correspond to higher skill. We additionally added several interactive elements to this chart:

1. When the user hovers over a stacked bar, a tooltip appears that displays the total number of games represented in this bar, as well as the exact win and draw rates. Because the stacked bar chart shows percentages, the exact number of games allows the user to contextualize how many games the bar represents.

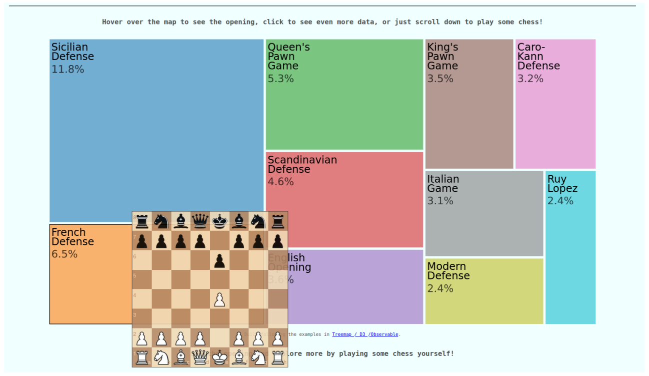
2. The total number of games represented in the current graph, as well as its proportion, is shown on the bottom of the graph at all times. This allows the user to identify whether the moves they made are common or not, allowing them to analyze the opening further and encourage them to use the tooltip.

3. A drop-down selection embedded in the title of the bar chart allows users to toggle between player perspectives, so that the user can explore win/draw rates for both the white and black pieces.

3.3 Filtering

With our encodings in place, we wanted to allow the user to filter through the data in order to narrow down the visualization, get information, and even see possible trends. Initially, we had only filtered through the data via the aforementioned PGN, whereas we would go through every data point, which has the PGN of the entire game, and check if its PGN also starts with the opening PGN in question. When the board gets updated, either through the treemap or user input, the bar chart also gets updated, showing the win rates as well as the total number of games that qualify.

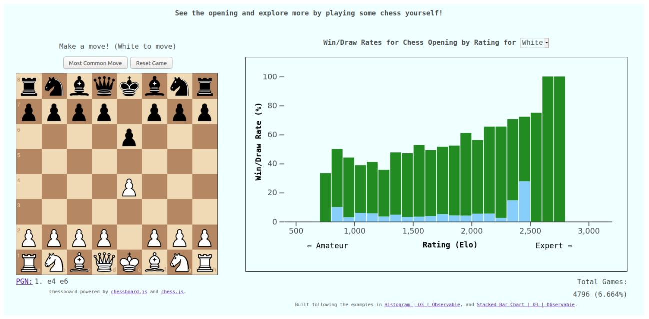
While we still kept this, we wanted to filter across more parameters as we expanded the project, got more feedback, and more data! Based on initial feedback, we thought it would be a good idea to filter across skill level, in order to allow users to explore certain skill levels more in-depth and hopefully add a personalized feeling to it. The lichess database contains the Elo, a calculated skill level based on various criteria, of the players for each game, so we filtered on that via a range slider. We also thought it would be interesting to filter by date, so the user can use two dropdowns to set a start and end date to filter the games based on when they were played. After these two parameters are filtered upon, the treemap as well as the bar chart update, allowing users to see if there are any different trends when they zoom in.


Hover over the map to see the opening, click to see even more data, or just scroll down to play some chess!

Built following the examples in Treemap / D3 /Observable.

See the opening and explore more by playing some chess yourself!

Figure 3: When hovering over a chess opening in the treemap, a tooltip shows the user what the opening looks like, ie. the French Defense.


See the opening and explore more by playing some chess yourself!

Make a move! (White to move)

PGN: 1. e4 e6

Chessboard powered by chessboard.js and chess.js.

Win/Draw Rates for Chess Opening by Rating for White

Total Games: 4796 (6.664%)

Built following the examples in Histogram | D3 | Observable, and Stacked Bar Chart | D3 | Observable.

Figure 4: When clicked, the treemap, sends the opening's PGN, which updates the chessboard and bar chart showing the win rate distribution by skill for that opening.

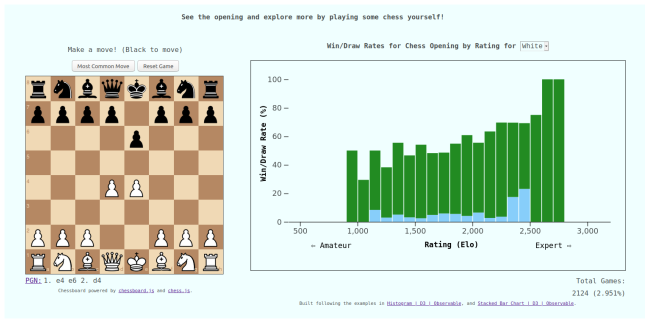
4 Results

In this section, we present a case study that illustrates how our visualizations address the problem we set out to solve. User A wants to explore what chess openings good players like to play, and how well they do.

The user can go onto the site and decide to explore the French Defense, as seen in Fig. 3. After clicking on the treemap, they are brought down to the chessboard and bar chart where the opening is played and the win/draw rate across skill level distribution is shown. They can see that lower rated players (less than 1500) tend to succeed more than higher rated players, see Fig. 4. They can decide to take a closer look into the game and play another move, including the most common one, see Fig. 5, and see how the distribution typically stays consistent for a few moves before branching. user A can also go back to the beginning and look at the openings distribution more closely by filtering by rating and/or year. When filtering for lower rated players, the user can see that it is more equally distributed, as seen in Fig. 6, while openings like the Sicilian and French Defense are much more popular (more than 20% of games), for higher rated players, see Fig. 7. This indicates that more skilled players tend to stick towards a certain meta while lower players do not, whether it be due to lack of skill, willingness to try different approaches, or some other unknown reason. This is an example of what user can do and what insights they may get when working with our site.

5 Discussion

We observe that the interactivity offered by our interactive chessboard is a major attraction for users. Our peer reviewers enjoyed the experience of "playing chess" and having a dynamic visualization


See the opening and explore more by playing some chess yourself!

Make a move! (Black to move)

PGN: 1. e4 e6 2. d4

Chessboard powered by chessboard.js and chess.js.

Total Games: 2124 (2.951%)

Built following the examples in Histogram | D3 | Observable, and Stacked Bar Chart | D3 | Observable.

Figure 5: The user can click on the Most Common Move button to cause the visualization to filter for the most common next move based on the PGN and play it, updating the graph, whose distribution stays relatively similar.

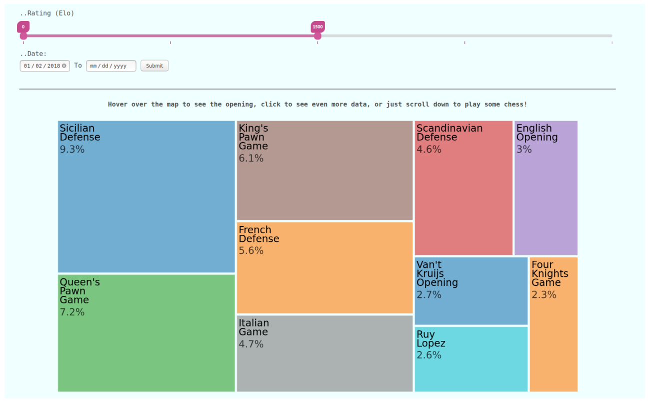

Hover over the map to see the opening, click to see even more data, or just scroll down to play some chess!

Figure 6: The treemap when the skill level is filtered from 0-1500. The distribution looks fairly equal.

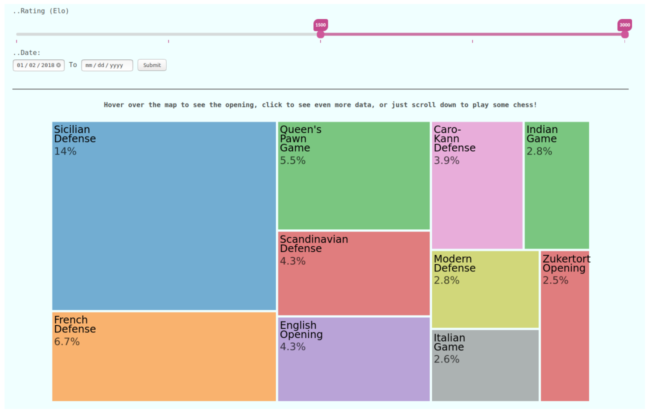

Hover over the map to see the opening, click to see even more data, or just scroll down to play some chess!

Figure 7: The treemap when the skill level is filtered from 1500-3000. The distribution looks slightly less equal.

that responded to their actions. Thus, it seems that in exploratory datasets such as this one, where there are many possible actions and parameters that can be chosen, a good strategy to engage the user is to allow them the option to make those decisions.

This visualization also demonstrates the importance of the process of onboarding. We do so in three ways: (1) by visualizing the named openings to users when they hover over the treemap, (2) by allowing the user to populate the chessboard by selecting specific openings, and (3) through instructive text positioned right above each visualization, telling the user how they should interact with them. For instance, above the treemap, we include the text "Hover over the map to see the opening, click to see even more data, or just scroll down to play some chess!" These elements reduce the barrier to exploration for a new user.

Finally, performance and responsiveness is critical to the user experience. In our visualization, we employed various strategies to keep load times low, such as preprocessing our data to keep only the necessary information, and taking care to only filter our dataset when it is necessary. For instance, when the user makes a move on the chessboard, only the stacked bar chart is reloaded, and not the treemap. When we expanded our dataset to contain more games across a wider range of time, we experienced serious lag as the data got filtered and updated, detrimentally affecting the user's experience to the point where they sometimes did not understand what was happening. In order to have a good range of data, we decided to sample the available datafiles across time in order to have a wide range of data without being too big. While we definitely wanted to have more data available, we realized the importance of balancing that with performance for sake of the UX, especially when a big draw was the dynamic aspect of the chessboard.

6 Future Work

Our system could be extended in several ways. Firstly, we could take advantage of the ability of treemaps to visualize hierarchical data by showing continuations of openings within each node in the treemap. For instance, within "Sicilian Opening", we could populate nodes showing the most common next moves after the Sicilian Opening is played. Such an addition would add depth to the exploration. It also further reduces the barrier to usability, since a newcomer to chess can visualize the commonly-played moves in this way.

Furthermore, we can create more efficient data structures to store the chess games, so that more games can be included in the visualization. This could be done by indexing or grouping our data by time or rating, so that filtering will be efficient. We can also employ hashing to reduce the size of our dataset, for instance, by mapping each opening to a number, and storing that number in our data file, instead of the full string.

Finally, the lichess open dataset contains information beyond those explored in this project, allowing many more interesting questions to be asked. For instance, what are the types of mistakes that are made at each level? We may utilize the engine evaluations of each move to determine when a mistake is made, and attempt to visualize player behavior in this way. Another interesting question or point of exploration would be the endgame. Much like the opening, there is a lot of analysis that goes into the endgame of chess, so we could characterize the types of endgames and look at how successful players are in converting them into wins.

References

[1] chessboard.js. https://github.com/oakmac/chessboardjs/. Accessed: 2021-05-18.

[2] chess.js. https://github.com/jhlywa/chess.js. Accessed: 2021-05-18.

[3] d3.js. https://d3js.org/. Accessed: 2021-05-18.

[4] Histogram. https://observablehq.com/@d3/histogram. Accessed: 2021-05-18.

[5] How many chess players are there in the world?. https://www.chess.com/article/view/how-many-chess-players-are-there-in-the-world. Accessed: 2021-05-18.

[6] lichess.org open database. https://database.lichess.org/. Accessed: 2021-05-18.

[7] Portable game notation (pgn). https://www.chess.com/terms/chess-pgn. Accessed: 2021-05-18.

[8] Stacked bar chart. https://observablehq.com/@d3/stacked-bar-chart. Accessed: 2021-05-18.

[9] Stockfish chess engine. https://github.com/official-stockfish/Stockfish. Accessed: 2021-05-18.

[10] Treemap. https://observablehq.com/@d3/treemap. Accessed: 2021-05-18.

[11] H. Chitkara. How chess.com built a streaming empire. https://www.protocol.com/chess-streaming-twitch-hikaru-botez. Accessed: 2021-05-18.

[12] A. E. Elo. ”8.4 logistic probability as a rating basis”. the rating of chessplayers, pastpresent., 2008.

[13] D. Hooper, K. Whyld, et al. *The Oxford companion to chess*. Oxford University Press Oxford, 1984.

[14] C. E. Shannon. Xxii. programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275, 1950.