Lux Zhang, Luis Enriquez, Mae Zhao, Nick Rutledge

CS 3300

Prof. Rzeszotarski

11/19/2021

Project II: Chess Meta Visualization

Overview

While every game of chess is unique, a statistical analysis of large bodies of chess games reveals correlations between game progress (measured in moves played) and other variables such as which type of piece is moved. Analysis such as this can be helpful to players who wish to improve their game and gain insights into the ways that the rules of chess influence good play.

In this project, we present a visualization of which pieces are moved most often at different points during a chess game, and to which rows, columns, and cells they are most frequently moved. We also allow these results to be filtered by player level and color. The insights we derive from this analysis will be detailed more below.

Data

Our analysis is based on aggregate statistics derived from two datasets of chess game records totaling approximately 91 million games. The first dataset is ChessDB, a free and open chess database of 3.5 million games sourced from Kaggle. These games are predominantly pro games between very high-level players. The second dataset is sourced from lichess.org, the internet's largest open source, free access multiplayer chess website. Lichess publishes complete records of every game played on its service since inception. We used the September 2021 dataset, consisting of 88.1 million games. These records, which uncompress to over 198 GB of textual data, consists of predominantly amateur games between players of all skill levels from beginner to grandmaster, and was chosen to contrast against the pro games from ChessDB.

The two datasets come in two differing formats (see Figure 1 below). Both use algebraic notation to represent chess games: "Qd4+", for instance, means "Move the queen to d4; check". But the details vary. The Kaggle dataset has one game per line; each begins with 17 columns of metadata and then gives the algebraic moves of the chess game in a space-separated list. The Kaggle dataset prepends the color and index of each move; "Qd4+" would become "W13.Qd4+", for example. In contrast, the Lichess dataset puts metadata in brackets, like "[WhiteELO 1340]", separated by newlines, and does not

prepend player and move index. Lichess makes its own addition to the algebraic notation, however, appending Stockfish analyses and timing data to many of the games in line with the algebraic move notation.

ChessDB Format	Lichess Format
95 2000.08.21 1-0 2849 None 49 date_false result_false welo_false belo_true edate_false setup_false fen_false result2_false oyrange_false blen_false ### W1.e4 B1.e6 W2.d4 B2.d5 W3.Nd2 B3.Nc6 W4.Ngf3 B4.dxe4 W5.Nxe4 B5.Nf6 W6.Nxf6+ B6.gxf6 W7.g3 B7.e5 W8.Bg2 B8.Bg4 W9.h3 B9.Bh5 W10.g4 B10.Bg6 W11.c3 B11.Qd7 W12.Be3 B12.h5 W13.dxe5 B13.Qxd1+ W14.Rxd1 B14.fxe5 W15.Nh4 B15.Bc2 W16.Rd2 B16.Bb1 W17.b4 B17.Nxb4 W18.O-O B18.Nxa2 W19.Bxb7 B19.Nxc3 W20.Bxa8 B20.hxg4 W21.hxg4 B21.Be4 W22.Bg5 B22.Bd6 W23.Rc1 B23.Bxa8 W24.Rxc3 B24.Be7 W25.Rxc7	[Event "Rated Rapid game"] [Site "https://lichess.org/3Xuc00Pp"] [Date "2021.09.01"] [Round "-"] [White "Jaiane103"] [Black "ShaianeSH103Rodolfo"] [Result "1/2-1/2"] [WhiteElo "1500"] [BlackElo "1500"] … (some metadata omitted) … 1. h4 { [%clk 0:10:00] } 1... a6 { [%clk 0:10:00] } 2. h5 { [%clk 0:09:45] } 2... a5 { [%clk 0:09:57] } 3. g3 { [%clk 0:09:38] } 3... b5 { [%clk 0:09:55] } 4. e3 { [%clk 0:09:35] } 4... c6 { [%clk 0:09:52] } 5. c3 { [%clk 0:09:33] } … (remainder omitted) …

Figure 1: Raw data formats

```
const DATA = {
    "eventsByMove": [
        {
            "name": "total",
            "data": [90957934, 90952151, 90823678, 90731842, ...]
        }, {
            "name": "pawn",
            "data": [87056190, 81922575, 40696340, 47392449, ...]
        },
        … (about 300 more entries omitted) …
    ]
};
```

Figure 2: Post-processed, aggregated data (data.js)

To handle these datasets, we developed an ETL pipeline (see etl.js) that can parse, aggregate, and extract all relevant information from the entire 210 GB of data in about 50 minutes. The ETL code requires Node to run and outputs a single file, data.js, which aggregates the entire 210 GB dataset into 0.6 MB of time series data for about 300 categories, each representing a different event that might take place during a move. For instance, the “pawn” event means that a pawn was moved; the “pos_f5_mid” event means that a player of middle skill level (ELO) moved a piece to position f5. This data is what the visualization actually displays.

Figure 2, above, shows an excerpt from the aggregate data file. The "data" members show the number of times that the "name" event occurred on each move across all of the games in the dataset. For instance, Black's first move is the 2nd move of the game, so Black opens with pawn in 81,922,575 of the 90,952,151 games that reached the 2nd move.

Visual Design Rationale

Our goal in this project was to allow users to improve their chess game by visualizing how chess games progress as they are played. For that reason, we did not want to create an overly reductive display, because serious chess players would get little out of it. On the other hand, we did not want to overwhelm the audience with information – we wanted to surface what was important and hide what did not matter.

Our initial idea was to display the dataset using line graphs. For instance, the x-axis might display time (measured in moves played), and the y-axis might display how often a pawn was moved, a piece was moved to position A1, or so on. But we quickly realized that, with many line graphs competing for space, the visual story was being obscured. (See Figure 3.) Further, we found that many very interesting patterns develop in long-lived games (100-200 moves in length), but these kinds of games are rare enough that the graphs were trending down toward the x-axis and the data was not as visible.

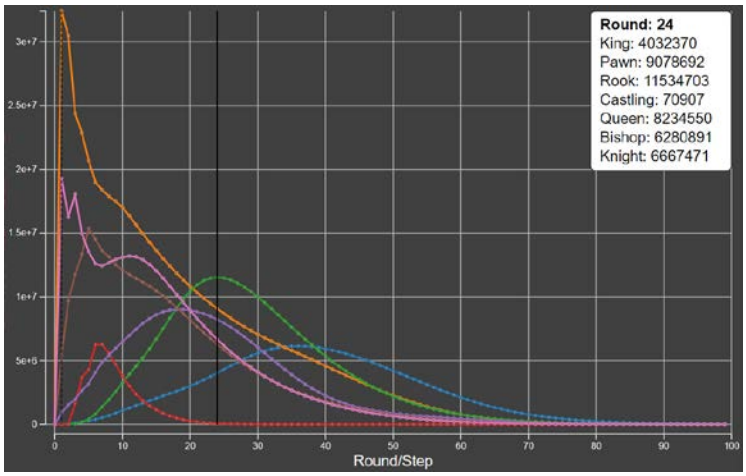

Figure 3: Crowding in a line graph display

For that reason, we decided to also add an option to display the data as a normalized stacked area plot and as a stacked area plot. (See Figure 4 below.) Stacking the areas solves the first problem above: now the line graphs do not interfere with each other. And normalizing solves the second problem by ensuring that the entire vertical height of the graph is always used.

Additional testing showed that, for positional analysis (how often do players move to all 64 positions during a certain move?), the graph was simply inadequate. We would have to visualize every destination from A1 to H8 on a single graph. So, we also decided to add a heatmap which would show how frequently a certain location was a destination during a particular move.

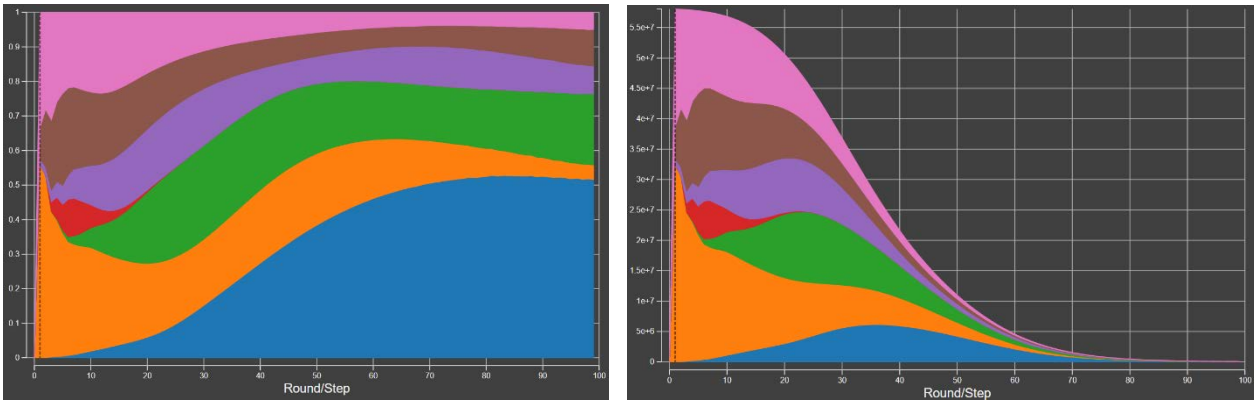

Figure 4: Normalized and unnormalized stacked area graphs of the same data.

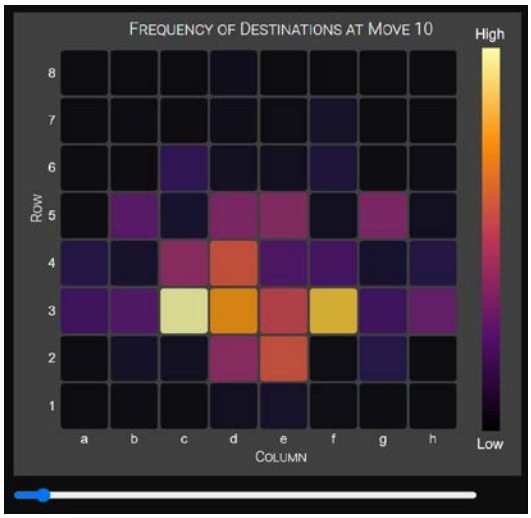

Figure 5: Move destination heatmap displaying Black's 15th move.

The stacked area charts and line chart display the data by varying datapoint positions on the vertical common scale at any given move number on the horizontal axis. The different categories (in the case of Figure 4, piece types) are displayed using a 10-hue ordinal scale through D3's schemeCategory10, the colors of which vary in hue distinctively, and different colors should be adequately distinguishable for people with color blindness.

The heatmap, which is shown above in Figure 5, allows the viewer to see all 64 channels of destination data in a single display. The slider at the bottom allows for the user to change the time which is being displayed on the heatmap.

The rows and columns of the heatmap resembles the layout of the chessboard, and the squares' position on the common horizontal and vertical scales denote each positions on the chessboard. The colors of the heatmap are provided by D3's "inferno", a perceptually uniform quantitative sequential color scale. Higher values are warmer and brighter, and lower values are cooler and darker. This color scale spans from a dark purple to a bright, desaturated yellow, and the shift in saturation, hue, and luminosity ensures that there is no ambiguities between any two squares.

Interactivity

In terms of interactivity, we wanted to offer the user the ability to filter the data in a variety of ways. For one, we allow the data to be broken into low, middle, and high skill level of play. We do this by examining the ELO scores of the players on both sides. Generally, if a player's ELO is below 1225, they are in the bottom 20% of the player base on Lichess, and if their ELO is above 1875, they are in the top 20%. All data in the visualization is filterable at these stopping points, which partition the games into low, middle, and high skill levels. This is shown below in Figure 6.

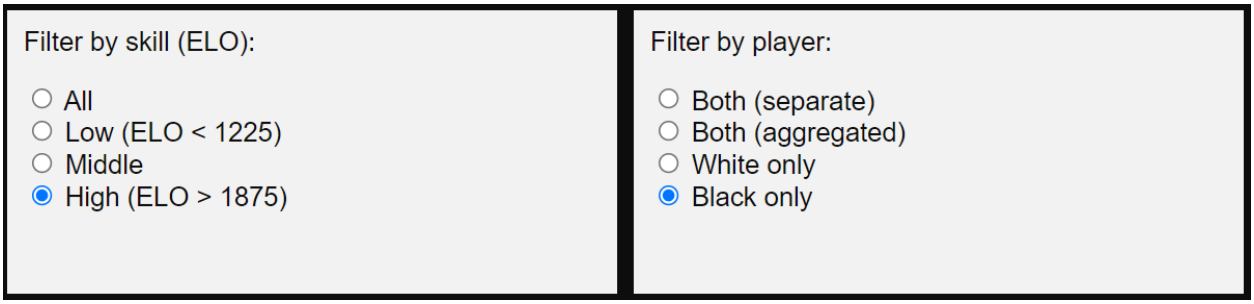

Figure 6: Filtering options.

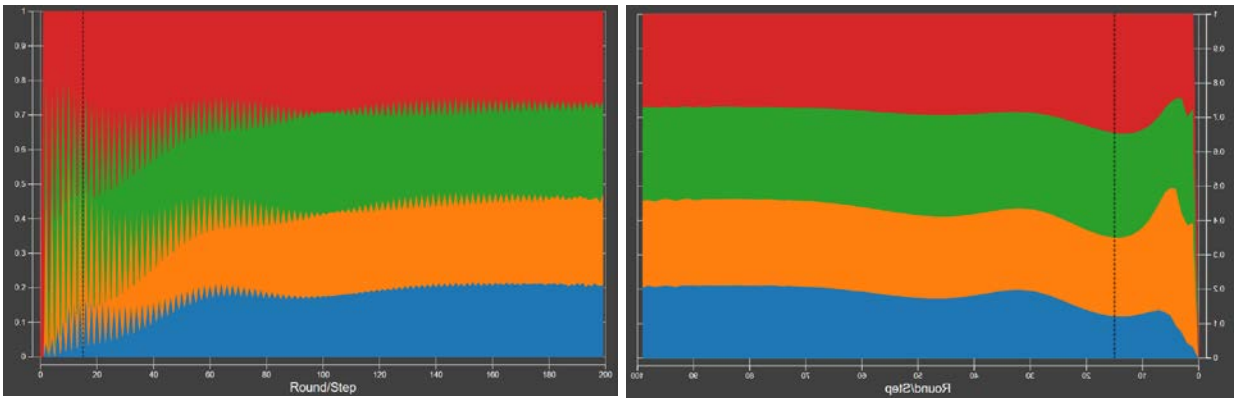

Figure 7: Jitter because of white/black player asymmetry, solved on right by Both (aggregated) filtering.

We also offer filtering by player, which is a more subtle option. Chess is not fully symmetric; the white player's view of the board is not exactly the same as the black player's view, because the king and queen positions are set by absolute position and not relative to the player. For that reason, play is inherently asymmetric. We ran into a problem where if we displayed *every* move, we would get wild jitter as in Figure 7, above, because play alternates between white and black. The filter option "Both (aggregated)" combines adjacent white and black moves into a single data point, solving this problem.

The Story

Here are just a few of the insights we were surprised by after having developed this visualization:

- **High-level play is systematically different from low-level play.** High level players finish matches slower (have fewer early checkmates), move to the center of the board more quickly, have more consistent and clear opening moves, and favor their rooks more in the endgame than lower-level players. They also exhibit a slightly greater bias toward the center of the board.

- **The right side of the board (columns E to H) is more active than the left.** This is true at all skill levels, but especially pronounced after the first 10 moves and at higher levels of play. Probably because of the king/queen asymmetry, players tend to push their pieces onto the right-hand side.

- **Pawns dominate the early game, then rooks the midgame, then kings the endgame.** Roughly speaking, from White's 1st move to Black's 20th move, the pawn is the most common piece played. The rook dominates from White's 21st move to Black's 36th. After that, the king is the most common move.

- **The center of the board is usually the most important position.** Players prefer to push their pieces to the center quickly in the early game, and then as games progress, the most common action is toward the center.

Overall, what we hoped this visualization would convey is that chess is a vastly complex game with untold opportunities for data mining. Players of chess may intuitively feel the flow of the game, but we wanted to make that flow more explicitly visual so that both experienced and novice players could see it for themselves. We believe that the project does precisely that.