ChessViz M2

7th May 2023

Alexandru Tifui
Emma Chirlomez
Theodor Moroianu

Project Goal

As amateur chess players, we wanted to create a tool one can use to see the effectiveness of various openings, strategies or individual moves throughout chess games and player ***ELOs***.

An opening, a move or a game board that might lead with a high probability to a win for low-rated players might be disastrous against skilled players, or vice-versa. This implies there is no universal ranking of potential moves given a chess board, when taking into account the skill (***ELO***) of the two players. With ChessViz, we aim at providing a straightforward, easy to use visualization tool, where players of any rating can explore historical data of chess games, for their particular ***ELO*** range. For this purpose, our visualizations often include a chess board, where players can examine various chess data.

Architecture and Used Tools

The application we developed follows the conventional architecture of a single-page web application. Its components are:

1. The backend API server, which computes and sends to the UI data to be visualized. The server is written in Python 3, and heavily relies on:
    - Flask, a simple web framework we use for communicating with the clients.
    - Python-Chess, a Python package for parsing PGN files (chess archives).
2. The frontend server, serving static assets to browser clients, and proxying API requests to the backend server.
3. The web UI, written with Typescript in React. For the visualizations, we use:
    - ***D3***, for which we built a custom React wrapper.
    - ***Chessground***, a chess board visualizer created by lichess.org, for which we also built a custom React wrapper.
    - ***BlueprintJS***, a UI toolkit for a nicer user interface.

The web application itself (i.e. the data sent to the client when the page is loaded) does not include any data, making it load and render very fast. When a visualization is generated by the user, the app makes API calls, requesting from the API only the required data, making the rendering visualizations very fast and responsive, especially considering we are parsing over 24GB of chess data for our project.

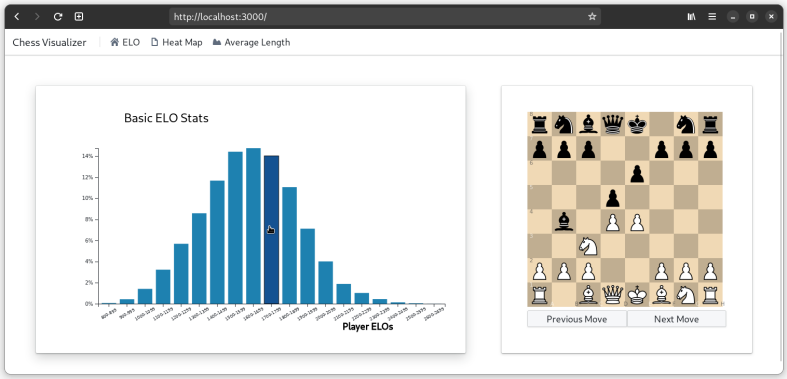
User Interface of ChessViz

ChessViz is still under construction, but we have already generated a few plots, for providing an end-to-end MVP, and showcasing the potential usefulness of our app.

The application has a menu at the top, where users can select various visualizations.

When a user selects a visualization, the main content of the page is replaced by the visualization.

Some visualizations are split-screen, where the left part of the screen is a D3 (possibly interactive) visualization users can click on to see additional information on the right part of the screen, such as moves on the chessboard or another visualization.



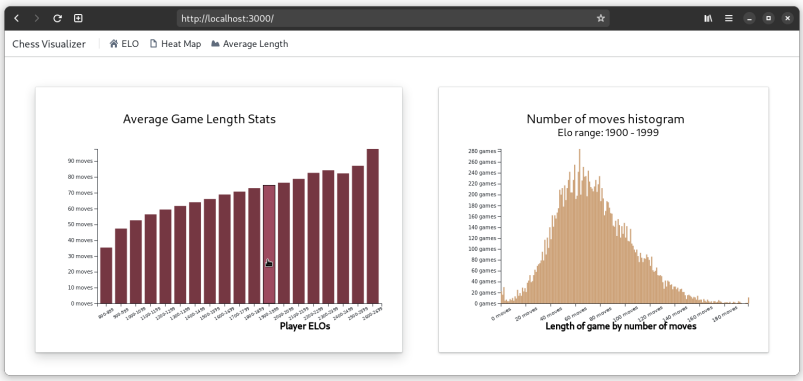
Visualizations

For representing our chess data in a meaningful way, we took inspiration from the following lectures:

- ***Histograms. Pie charts, Line charts*** - Lecture "*Do and dont in viz*".
- ***D3.js Library*** - Lecture "*D3.js*".
- ***D3 interactive components*** - Lecture "*Interactions*", Lecture "*More interactive d3.js*".
- ***Formatting and colors*** - Lecture "Perception colors".

As of now we implemented the following visualizations we want for the final product:

- An interactive wrapper over a chess board view.
- Histogram of number of games per ***ELO*** rating range. The chart is clickable, displaying for each bucket a random game played between



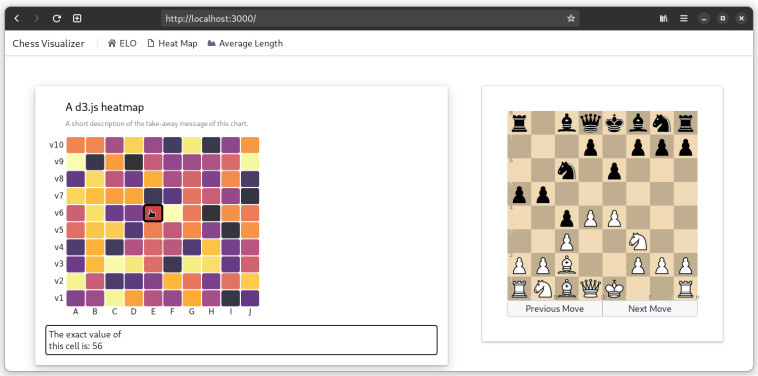
two players within the ***ELO*** range.

- Histogram of the average game duration (as number of moves) per ***ELO*** range. The chart is also clickable, displaying a more precise distribution of the number of moves.

We want to additionally implement:

- A line chart, following a few players, to see their rating changes throughout the month.
- Bar chart of number of quick defeats / abandoned games per ***ELO*** rating range.
- For each ***ELO*** range, a wide range of visualizations such as:
    - Most used openings (clickable pie charts connected to a chessboard view).
    - For most used openings, show the outcome and frequency (pie charts).
    - For each piece, a heatmap of the frequency of that piece on the board.

We have also implemented a proof-of-concept D3 heatmap view, connected to a chessboard (a click on the heatmap triggers a change in the game and move displayed in the chessboard view), which at the moment displays dummy data.



Functional Prototype - Running The Project

The easiest way to run the project is to follow the instructions provided in the "*readme.md*" file on github. More concretely:

1. Clone the repository.
2. Generate the backend data, by running in the backend folder:

```
$ cd backend/
$ python src/main.py generate-and-run
```

3. Start the frontend server, by running:

```
$ cd frontend/
$ npm i
$ npm run start
```

The app should then be accessible on http://localhost:3000/.