MPCS 56430 - Scientific Computing
Final Project: Chess Database Analysis

Shane McVeigh smcveigh

1 Introduction

For my project, I applied the high-performance computing and multiprocessing principles covered in the course to a chess game database hosted on Lichess, the most popular open-source online chess website. The questions I looked to answer were:

- What openings are associated with higher and lower levels of play?
- Are any openings gaining or losing popularity?
- Are any time controls gaining or losing popularity?
- Does playing with white or black matter more at different rating levels?
- What is the average rating difference for the winning side?

I felt strongly that there would be data supporting a change in popularity of openings, as this is a phenomenon that has existed in chess for well over 1,000 years. I was less sure of what to expect regarding time controls, but my hypothesis was that lower time controls (less time) would be trending more popular as the average person's attention span drops due to the influence of technology.

2 Background

There are some definitions that are important to understand in order to analyze the chess games. Each of the chess games in the Lichess database follows a standard structure (linked above). Some of the chess terms that are important for my analysis are:

- ECO - this is the opening code that correlates with a sequence of moves at the beginning of the game. Well-documented by 365chess.
- ELO - this is the average rating of a player with the white or black pieces. Many other competitions use ELO rating as well as it works effectively as a peer-ranking system. A good summary by Chess.com can be found here.

3 Methodology & Approach

Lichess database files are stored in a standard format, which supports the Zstandard for compression and decompression. Therefore, it was expedient to use the `Zstandard` python library in order to be able to decompress the game files without storing hundreds of GB on my local machine or a cluster. The `Decompressor` function was most important for my use case and is documented in the Python docs.

I used some HPC methods including a `TextIOWrapper` to reduce the IO buffer size for each file analyzed and put it into a stream rather than streaming the whole file, and also using the `Multiprocessing` library in python, documented here, to create a `Pool` that would be able to run the analysis on multiple large files at once. Since the I/O overhead is quite significant, I limited my parallel processes to 3 to limit buffer overhead on the machine. This still helped significantly speed up my analysis, by a speedup of about 2x (likely did not hit the Amdahl's law limit of 3 due to said overhead).

4 Results & Discussion

4.1 Opening Ratings

Highest-Rated Openings	Lowest-Rated Openings
B69 (Sicilian: Richter-Rauzer) (2578)	C20 (King's Pawn Game) (1336)
D99 (Grunfeld: Smyslov) (2564)	C57 (Italian: Two Knights) (1389)
D76 (Neo-Grunfeld) (2429)	C23 (Bishop's Opening) (1422)
E96 (King's Indian: Orthodox) (2409)	C40 (Damiano) (1428)
B67 (Sicilian: Richter-Rauzer) (2406)	C22 (Center Game) (1431)

There were some clear trends in the source data and high correlations between White and Black's highest and lowest rated openings. I have included just White's ELO rating in the opening results above. For context, 2500 ELO is generally a grandmaster-level player, and 1200-1400 is a decent amateur. So the gaps between the ELO ratings in the table above show a clear divide in some of the openings played by grandmaster-level players. I will certainly be spending some time looking into the Richter-Rauzer for the Sicilian and the Grunfeld (Smyslov) variations, while avoiding some of the more frequent Kings' Pawn openings which comprise the entire lowest-rated openings list.

4.2 Trends from 2017-2025

I sought to answer 2 trend related questions relating to openings and time controls across the period from 2017-2025, evaluated in `popularities.py` and then further refined in the `trends` folder. I found quite interesting results particularly in the time controls data, which showed a consistent upward trend in the proportion of short-time control games and a corresponding downward trend for long-time control games. The full plot for Top 10 most common time controls can be seen at this link. One-minute games used to be the third most common game in 2017, but quickly eclipsed two blitz time control games to become the most common time control.

In terms of the opening data, there was a strong population of openings that kept their spot in the top 10, with some openings joining and leaving over the past 8 years. This follows a long-time trend of certain openings gaining and losing popularity in the chess community. The full plot of the Top 10 most common openings can be seen at this link. There were some interesting openings that became more common (Uncategorized King's Pawn Openings) and less common (C00 - the French defense) over the course of the 8 year period.

4.3 Win Rate Statistics

In `ratings.py`, I evaluated the average ELO difference for the winning side in the most recent monthly upload. I found that the average rating difference for the winning side was 19.51 ELO points, which is quite significant over many millions of games. This lends credence to the effectiveness of ELO as a rating tool.

I also evaluated the win-rate difference (or white advantage) at each rating bucket of 100 ELO. There was a consistent up-slope by rating level (visible at this link). White win-rate was better than black-win rate by about 2.8% at low rating levels, but increased to 4.5% at high rating levels. It was fascinating to see that better players were, in practice, better able to use first-move advantage, even when playing against better players as well.

5 References

(1) "Lichess.Org Open Database." Lichess.Org Open Database, database.lichess.org/#standard_games. Accessed 24 Nov. 2025.
(2) "ECO Codes." 365Chess, www.365chess.com/eco.php. Accessed 24 Nov. 2025.
(3) Chess.com. "Elo Rating System - Chess Terms." Chess.com, www.chess.com/terms/elo-rating-chess.
(4) "Compression.zstd — Compression Compatible with the Zstandard Format." Python Documentation, 2025, docs.python.org/3/library/compression.zstd.html#compression.zstd.ZstdDecompressor. Accessed 24 Nov. 2025.
(5) Python. "Multiprocessing — Process-Based Parallelism — Python 3.8.3rc1 Documentation." Docs.python.org, docs.python.org/3/library/multiprocessing.html.