Chess Data Analysis

Addressing the Problem

The problem that we were interested in answering was, Does time affect a player's performance in a chess match?". However, after investigating the data and results we were not satisfied with the results. This led us to broadening the questions and creating our second problem, "Does time affect a player's performance in a chess match?". After analysing the games and results from our new question, we noticed a problem where a player's skill level can affect the results. For example, a player that plays professionally will most likely make better moves compared to an amateur player. As a result, we have decided to split the players based on their elo into separate groups to remove this variability, giving our final problem.

Defining/Refining the Problem:

1. How does time pressure affect a player's performance?
2. Does time affect a player's performance in a chess match?
3. Looking at different skill groups, how does time affect a player's performance during a match?

Data Gathering and Filtering

Lichess:

There are various websites that record and store chess games played by various players. Lichess has one of the largest databases for recorded chess games with over 1.95TB worth of games stored on their website.

https://database.lichess.org/

Games are stored by month as a .pgn.zst file. Each game holds the following values; the type of game, site, date, round, white player name, black player name, result, UTC date, UTC Time, white elo, black elo, white ratting diff, black rating diff, white title, opening name, opening code, time control, termination, and move details. 6% of the games stored include a stockfish engine and games recorded after April of 2017 will have a clock involved in move details.

All of the recorded chess games were played on the lichess.org website, this means that the standard elo value will be different compared to other websites. For new players, their starting elo value will be at 1500 meaning a majority of the games will have an elo value close to that score.

Because there is no real difference in downloading any dataset after April 2017, we have downloaded the 2020 January folder to use in our data analysis. This dataset contains 46,800,709 games.

chess_data_extraction.py

The chess_data_extraction.py is responsible for extracting the data downloaded from the lichess website. Because the file is in a .zst format, the python zstandard library is needed to Decompress and read the file. To read the .pgn chess file, another python library chess.pgn is required. While iterating through each game, several values of the game are extracted. These

are the result of the game, the opening, white elo, black elo, moves, and clock data. Clock data can potentially contain the stockfish evaluation, which is important information needed for this analysis. A check is made to make sure a game has this vital piece of information. If the stockfish evaluation exists, the game data is added onto the pandas dataframe, but if it does not exist, then the game is skipped. We then sorted the dataframe into different elo brackets and wrote the data into csv files. In total, we extracted 1,300,000 games.

move_evaluations.py

The move_evaluations.py script evaluates chess moves using the Stockfish engine to classify the quality of each move in a dataset. The script processes games by iterating through the moves recorded in the dataset, evaluating the board position after each move, and then saving the evaluations for further analysis. However, given our full dataset size (45 million games) this approach proved to be computationally intensive, taking over an hour to process just 8000 games. To address this challenge, we decided to focus on datasets where evaluations were already pre-extracted, significantly reducing processing time. We've left the code here as a foundation to revisit upon in the future, should we decide to attempt a more detailed evaluation or optimize the process for larger datasets.

chess_data_refine.py

The chess_data_refine.py script is a data preprocessing tool designed to clean and structure raw chess game data for subsequent analysis. It processes multiple CSV files containing games, categorized by Elo rating ranges, and extracts key features such as evaluations scores (%eval) and clock times (%clk) from the Clk column. Using regular expressions, the script parses these annotations, storing evaluations and clock times in new columns while removing the original Clk column. Additionally, the script filters out irrelevant or malformed rows, such as those with missing or ambiguous openings (Opening = "?") and repeated headers, ensuring that the output is clean and consistent.

The script processes data in batches, iterating through a predefined list of files representing different Elo ranges, and saves the refined datasets back to their original locations. By retaining only rows with evaluation data and automating the cleaning process, it handles large datasets efficiently. This approach prepares the data for statistical analysis and visualization, while maintaining data integrity.

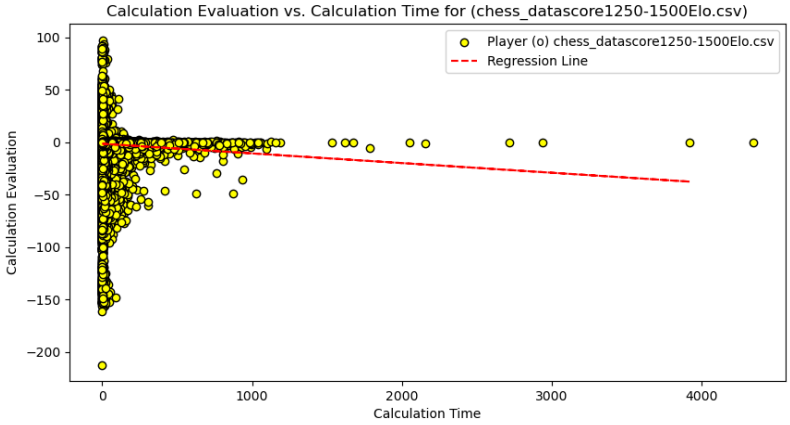
chess_data_getperformance.py

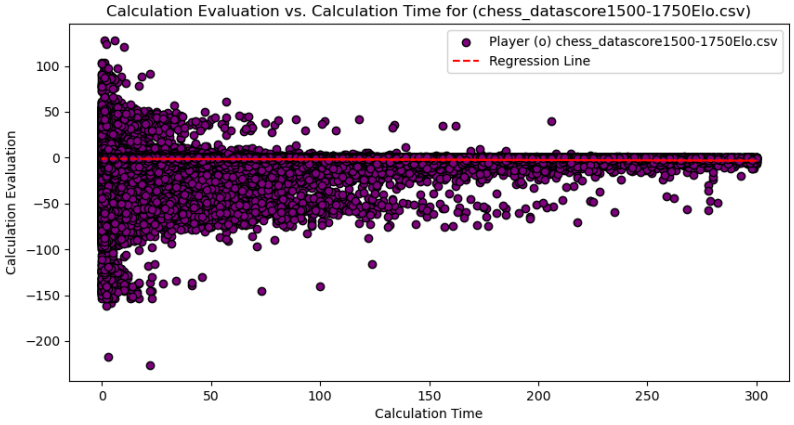
The chess_data_getperformance.py script calculates the performance of a player's move. This is through checking the evaluation score before and after the player's move. The chess_data_get _ performance.py also determines whether the move made is by white or black and the time spent on the move. If the calculated time is negative, meaning the difference between the start and end time is < 0, then that move is excluded from the dataset. The data calculated is stored into .csv files based on the elo range of the player. These .csv files are stored in a folder directory of the user's choice.

Visualizing Data

plot_avg_evals.py

The plot_avg_evals.py script is a visualization tool designed to analyze and plot the relationship between calculation time (calc_time) and evaluation scores (Calc_Eval) for chess players across different Elo rating ranges. It processes multiple datasets, generates scatter plots for each Elo range, and overlays regression lines to capture trends in the data. We included only 2 of the visualizations within the report and the rest can be found on the github repository.The resulting visualizations provide insights into how decision time correlates with evaluation scores for both white and black players.





We chose to show these two graphs as this is where the majority of the data is situated, mainly due to the fact that any new player that plays on lichess will start off with a rating of 1500 Elo. The first plot corresponds to the 1250-1500 Elo bracket, showcasing a broad range of calculation times, extending up to 4000 seconds for some moves. A red regression line is overlaid to highlight trends. It indicates a slight downward trend, hinting at a possible negative correlation between long calculation times and evaluation quality for players at this skill level. Furthermore, the downward slope suggests that longer calculation times might be associated with diminishing returns in evaluation quality, potentially pointing to overthinking or indecisiveness in lower-rated players. The second plot, for the 1500-1750 Elo bracket, depicts a dense cluster of data near shorter calculation times, with minimal variance in evaluations. The regression line in this plot shows no strong correlation, suggesting that calculation time has a limited impact on evaluations in this range. This more consistent relationship between time spent and move quality, reflects improved efficiency in decision-making among slightly more experienced players.

Data Analysis

chess_data_decision_time_mann_whitney.py

The chess_data_decision_time_mann_whitney.py script conducts statistical tests to determine whether there is a significant difference in player improvement based on the time they spend deciding their moves. Specifically, it uses the Mann-Whitney U-test, to compare moves made with “short” calculation times (less than 60 seconds) versus “long” calculation times (60 seconds or more).

Elo Range	White player p-value	Black player p-value
0-1000	2.8961593235141955e-10	1.1716819870537879e-10
1000-1250	1.2924230530847963e-97	4.2950033448196025e-83
1250-1500	2.7354433110113262e-191	4.620633078216946e-198
1500-1750	0.0	9.086627817170332e-297
1750-2000	3.225107201895036e-247	2.2221156337127994e-249
2000-2250	2.6185500189503097e-61	1.0505994362748187e-85
2250+	0.003271276643493218	0.003909403535141548

The results show that across all Elo ranges, the p-values for both white and black players are extremely low, indicating significant differences between short and long calculation times in terms of player improvement. This suggests that the amount of time players spend deciding their moves has a measurable impact on their performance. For lower Elo ranges (0-1000 or 1000-1250), the differences are significant but less pronounced compared to higher Elo ranges (1500-1750 or 1750-2000), where the p-values are extremely small. Interestingly, in the 2250+ Elo range, while the differences are still significant, the p-values are relatively larger compared to other brackets, which may indicate more consistency in decision-making effectiveness at the highest levels. Overall, the script reveals how decision time influences player performance across skill levels and emphasizes its importance as a factor in chess strategy.

chess_data_indepthtime_mann_whitney.py

After performing a Mann-Whitney U-test on moves made within 60 seconds and moves made beyond 60 seconds, we decided to make a more in depth analysis based on the time it takes to make the move. For example, making a p-value test based on the evaluation scores that took 0 seconds and the evaluation scores that took 1 second.

Because the dataset is not normal, we are unable to use Tukey and Anova to compare all the time groups in this manner. As a result, we would still need to use mann-whitney and would have to iterate through all possible combinations for each time group. Because the data results would be too long, we have only displayed what is needed to make conclusions. The rest of the data can be found on the github repository under the chess_data_analysis folder.

A sample result for Elo ranges 1250-1500 and 1500-1750 are as follows:

Time 1	Time 2	p-val (1250-1500)	p-val (1500-1750)
0 (s)	1 (s)	1.350843e-215	0.000000e+00
1 (s)	2 (s)	0.000000e+00	0.000000e+00

2 (s)	3 (s)	0.000000e+00	0.000000e+00
3 (s)	4 (s)	1.471288e-133	3.713576e-137
4 (s)	5 (s)	1.228704e-40	5.312213e-61
5 (s)	6 (s)	6.297282e-37	4.118404e-43
6 (s)	7 (s)	1.669967e-23	6.405558e-25
7 (s)	8 (s)	3.192424e-22	4.351653e-15
8 (s)	9 (s)	4.470184e-12	7.129074e-19
9 (s)	10 (s)	3.541753e-09	4.719528e-08
10 (s)	11 (s)	7.742693e-09	7.926885e-09
11 (s)	12 (s)	6.869311e-08	2.945917e-06
12 (s)	13 (s)	1.782499e-03	2.053869e-03
13 (s)	14 (s)	1.488172e-02	1.461529e-03
14 (s)	15 (s)	1.808943e-02	6.681926e-03
15 (s)	16 (s)	4.662482e-03	4.923660e-01 (False)

The results show that across different elo ranges, the p-value is extremely low when comparing low time spent moves with each other indicating there is a significant difference between the two groups. However when comparing with later times, the differences diminish, as evident by the larger p-value and scatter plot displayed by the plot_avg_evals.py, and eventually becomes non-significant or not enough data to conclude any significant difference.

Conclusion

Our analysis sought to answer the question: **"Looking at different skill groups, how does time affect a player's performance during a match?"** Through an iterative process of refining our problem and analyzing the data, we were able to draw meaningful insights into how calculation time influences performance at different skill levels. By segmenting the dataset into Elo brackets, we addressed variability caused by differences in player skill, enabling us to isolate the impact of time pressure on performance.

The results from our Mann-Whitney U-tests and deeper statistical analysis reveal that decision time plays a significant role in chess performance. For lower Elo ranges, the impact of time spent on moves was pronounced, though diminishing returns were evident with longer calculation times, potentially due to overthinking. In higher Elo ranges, players demonstrated more consistency, with significant differences between short and long calculation times persisting but becoming less variable at the highest skill levels. Additionally, granular

comparisons of individual time intervals (e.g., 0-1 seconds, 1-2 seconds) showed that significant differences exist for shorter time intervals but gradually diminish as the time spent increases. This suggests that while rapid decisions are critical in certain situations, there is a threshold where additional time spent may no longer yield improvements in move quality.

Ultimately, our findings confirm that decision time is an important factor in chess performance, but its impact varies by skill level and time spent. These results not only highlight the complexity of time management in chess but also provide a foundation for further exploration into how time pressure shapes strategic decision-making. Future work could explore additional factors, such as psychological aspects or opening-specific trends, to build upon these insights.

Workflow

1. Download a dataset of games from the lichess.org open database.
2. Run chess_data_extraction.py to extract relevant features from the raw dataset.
3. Run chess_data_refine.py to process and clean the extracted data.
4. Run chess_data_getperformance.py to calculate player improvement metrics, such as evaluation changes between moves, based on the refined dataset.
5. Run plot_avg_evals.py to generate a scatter plot to visualize the relationship between calculation time and evaluation scores for different Elo ranges.
6. Run chess_data_decision_time_mann_whitney.py to perform statistical tests to analyze the impact of decision time on player improvement.
7. Run chess_data_indepthtime_mann_whitney.py to perform a more detailed analysis by iterating through all possible decision time groups.

Project Experience Summary

Lazar

- Created impactful visualizations of chess player performance by implementing plot_avg_evals.py, generating scatter plots and regression analyses to showcase the relationship between calculation time and evaluation scores across various Elo ranges.
- Conducted advanced statistical analyses using chess_data_decision_time_mann_whitney.py, revealing significant differences in player performance based on decision times through Mann-Whitney U-tests.
- Designed and implemented chess move quality assessments with move_evaluations.py, leveraging the Stockfish engine to evaluate move performance and understand player decision-making patterns (was not used due to computational restrictions).
- Collaboratively developed and optimized the chess_data_refine.py script, by implementing a parse_clk_column function that was used to extract evaluation scores and clock times.

Preston

- Extracted critical features such as player Elo ratings, moves, and evaluation data by implementing chess_data_extraction.py, transforming raw Lichess datasets into structured CSV files for analysis.
- Designed and ran the chess_data_getperformance.py script to calculate player improvement metrics, enabling a clear understanding of the impact of moves on evaluation scores across different Elo brackets.
- Conducted in-depth statistical time analysis with chess_data_indepthtime_mann_whiteny.py, iteratively comparing evaluation scores for different decision time groups to uncover nuanced patterns in player performance.
- Collaboratively developed and optimized the chess_data_refine.py script by removing any games that have no moves and additional headers added on during the data extraction process.