Chess Patterns Through Data Visualization of 50,000 Lichess Games

Jeremy Neale
Professor: Reza Jafari
Information Visualization, Virginia Tech Summer Session I

July 2025

https://dashapp-1050976616574.us-east1.run.app/

Abstract

This research created over a dozen static plots and dynamic plots of a chess dataset from Lichess, containing over 6 million games. Although only 50,000 samples were used for efficiency, this large sample size allows for in-depth statistical analysis and visuals to shed light on patterns across chess games. A total of 17 conclusions were found. Some of the conclusions were: black tends to make the first capture, turn 7 is the most common turn to castle, black tends to castle queenside more often compared to white, and more. Python was used to load, preprocess, feature engineer, and then create static visuals for the data using many common Python packages. Python code was also delpoyed in a Dockerized container on Google Cloud, where a dashboard of dynamic plots and the static plots are using Dash and Plotly. There are also fully functional tabs that handle outlier removal, Principal Component Analysis (PCA), and normality tests.

1 Introduction

Chess is a fascinating and timeless game. Using a large dataset and modern computing resources, this research aims at finding and visualizing patterns across thousands of chess games, hopefully uncovering truths that lead to advantages for both beginners and advanced players.

2 Description of dataset

Lichess is a free and open-source chess website. A CSV dataset from Kaggle [1] contains over 6 million games, but only 50,000 are used in this project.
The dataset initially has 15 attributes:

- Event: string - Event type ("Classical", "Blitz", "Blitz Tournament", ...)
- White: string - White player's tag
- Black: string - Black player's tag
- Result: string - Result of the game ("1-0" for white win, "0-1" for black win, "1/2-1/2" for draw).
- UTCDate: string - date in UTC format
- UTCTime: string - time in UTC format
- WhiteElo: int - White's Elo (see below)
- BlackElo: int - Black's Elo (see below)
- WhiteRatingDiff: int - White's change in Elo from the game
- BlackRatingDiff: int - Black's change in Elo from the game

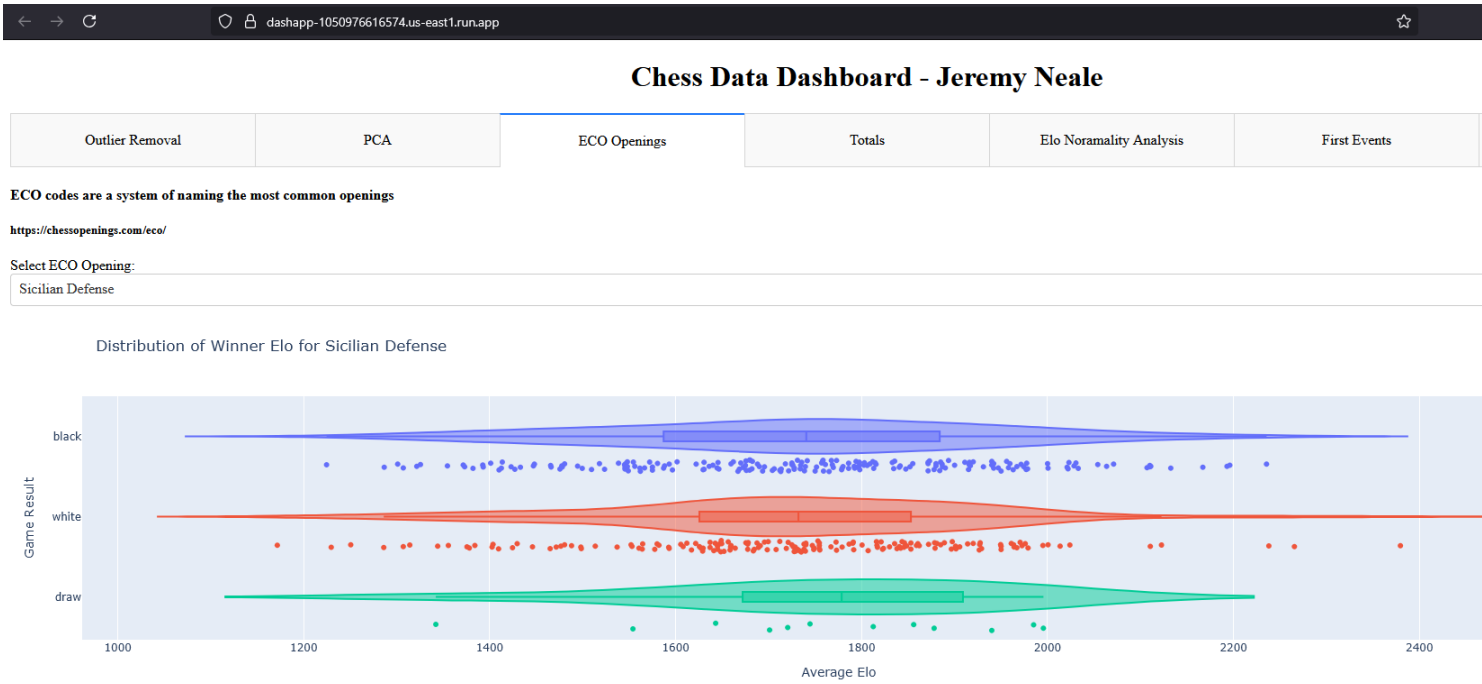

Figure 1: Dashboard of the web application on the "ECO Openings" tab

- ECO: string - 3 character naming system for most common 500 openings
- Opening: string - Expanded name of the ECO opening (example: Queens Gambit Declined).
- TimeControl: string - Time system of the game (example: 180+3, meaning 3 minutes plus 2 seconds gained for each move made).
- Termination: string - How the game ended (checkmate, resign, abandoned)
- AN: string - see below

  I picked this dataset because of the AN column. There is a near-infinite amount of feature engineering that can be done because all the data about the game is in the notation of the moves. There are already some numeric and categorical columns, and Elo serves as a great dependent variable column to see how skill affects decision made in a game like castling and a player moving their queen early.

Chess "Algebraic" notation (AN) is a system of naming each move based on where it moves in the grid and letters for the pieces (N = knight, K = king, R = rook, etc.) and special characters for other details (x = capture, + = check, O-O = short castle, O-O-O = long castle, etc.). Elo is numeric skill rating system that has a formula to calculate how much elo is lost/gained in a win, loss, or draw. It factors in skill differences, so as an example, a 1100 may gain 13 elo for beating a 1200 opponent, but only 2 elo for beating a 850 opponent. This numeric system is beneficial for graphing and analyzing patterns across skill levels.
The only metrics that are used from the 15 base attributes are Event, Result, Elo, Eco, and AN.

3 Pre-processing and Feature Engineering

Most of the data-preprocessing performed was feature engineering using Pandas [7]. The only preprocessing was changing the Result column to a new Result-string column, where "1-0" was changed to "white", "0-1" was changed to "black, and "1/2-1/2" was changed to "draw". A new "avg_elo" column was also made so games could be tracked as a single digit instead of separately. The data was cleaned with .dropna() for many of the graphs, which removes null data.

Event Game type	White White's ID	Black Black's ID	Result Game Result (1-0 White wins) (0-1 Black wins)	UTCDate UTC Date	UTCTime UTC time	WhiteElo White's ELO	BlackElo Black's ELO	WhiteRatingDiff White's rating points difference after the game	BlackRatingDiff Black's rating points difference after the game	ECO Opening in ECO encoding	Opening Opening name	TimeControl Time of the game in seconds.	Termination Reason of the game's end
Blitz 37% Classical 24% Other (2405818) 38%	118945 unique values	115946 unique values	1-0 50% 0-1 46% Other (240218) 4%	2016-06-29 2016-07-30	2021-02-14 2021-02-15	737 3110	728 3108	-595 673	-653 664	A00 7% A40 5% Other (5542427) 89%	Van't Kruijs Opening 2% Scandinavian Defe... 2% Other (6010845) 96%	300+0 17% 180+0 15% Other (4232429) 68%	Normal 68% Time forfeit 32% Other (14759) 0%
Classical	eisaaaa	HAMID449	1-0	2016.06.30	22:00:01	1901	1896	11.0	-11.0	D10	Slav Defense	300+5	Time forfeit
Blitz	go4jas	Sergei1973	0-1	2016.06.30	22:00:01	1641	1627	-11.0	12.0	C20	King's Pawn Opening: 2.b3	300+0	Normal
Blitz tournament	Evangelistaizac	kofuna	1-0	2016.06.30	22:00:02	1647	1688	13.0	-13.0	B01	Scandinavian Defense: Mieses-Kotroc Variation	180+0	Time forfeit

Figure 2: First few lines of the cleaned dataset

New columns made from feature engineering:

- `first_castle_color` – The color of the first player to castle.
- `first_castle_turn` – The earliest castle turn in the game by either player.
- `white_castle` – The type of castle performed (long, short, or neither) by white.
- `black_castle` – The type of castle performed (long, short, or neither) by white.
- `first_white_capture` – The turn of the first white capture.
- `first_black_capture` – The turn of the first black capture.
- `first_capture_turn` – The earliest capture turn in the game by either player.
- `first_capture_color` – The color of the player who made the first capture.
- `num_white_captures` – Total number of captures by white.
- `num_black_captures` – Total number of captures by black.
- `total_captures` – Total number of captures in the game (white + black).
- `num_white_checks` – Total number of checks by white.
- `num_black_checks` – Total number of checks by black.
- `total_checks` – Total number of checks in the game (white + black).
- `first_white_queen_move` – Turn of white's first queen move.
- `first_black_queen_move` – Turn of black's first queen move.
- `first_queen_move_turn` – Earliest queen move turn in the game by either player.
- `first_queen_move_color` – The color of the player who moved the queen first.

These features were all engineered using a few main custom functions. This generic approach to these functions allowed the first 3 to be used for first queen move, first capture, and first castle with a simple parameter change.

The main functions used were:

- first_move_pattern - This function takes 3 parameters: the string of moves (AN from the dataframe) for a specific game, the pattern to look for, and the color. The pattern to look for is either 'Q', 'x', or 'O-O'. The color is either 'white', 'black', or 'either'. It returns the first occurrence (if any) of the pattern by the specified color (it would return the first occurence by either player if 'either' is the third parameter).

- first_move_color - This function takes 2 parameters: the string of moves, and a pattern, similar to the previous function. This function returns the color of whichever player did the pattern first, not the numeric turn number.
- count_occurence - This function tallies up the total occurences of the pattern by both players. This is used to calculate total checks ("+") and total captures ("x").
- detect_castling_type - This function was specific to castling type, and takes the string of moves as a parameter and returns a tuple of which castle style each player used. It returns either "queenside", "kingside", or null for each player.

4 Outlier detection & removal

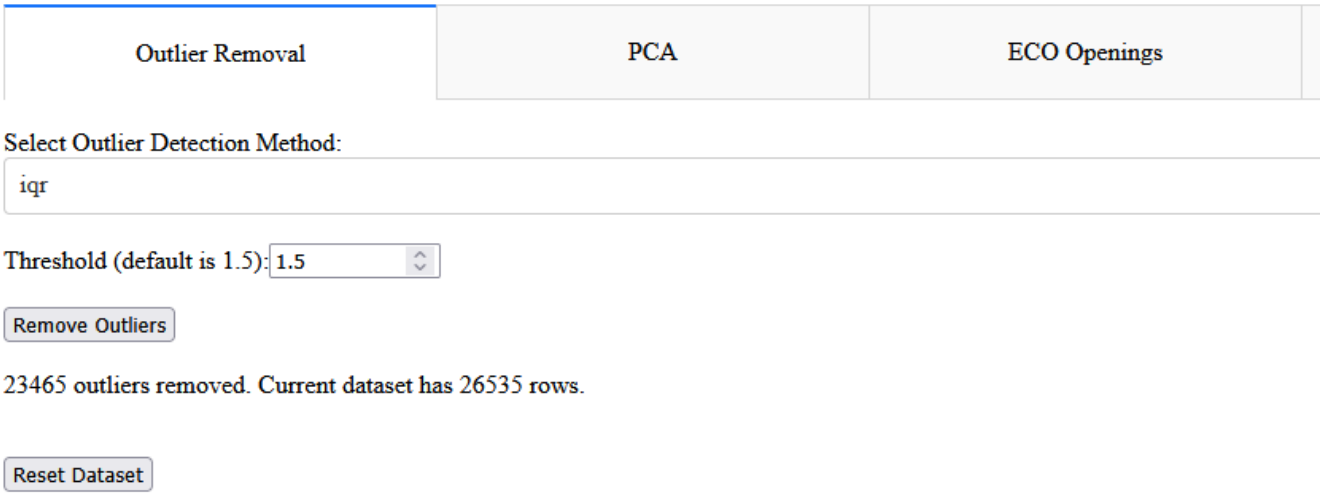

Figure 3: Fully functional outlier removal tab

The web app has two methods of removing outliers for all the graphs: IQR and Z. IQR is the interquartile range, and whatever value is selected is then added onto the third quartile and subtracted from the first quartile. Any values outside of the selected range are removed.
Z is Z-score, which is a measure of how many standard devations away from the mean a value is. Using the same entry box, a user can set the number of Z-scores as a threshold, and any value more Z-scores away from the mean is removed from the dataset.

They are removed from all numeric columns of the df. However, there is a reset button on the outlier tab that reverts the dataset back to it's normal state of 50,000 entries.

Many of the columns don't have ridiculous outliers. The highest elo in the world is right below 3,000, and there are players in the 2,000 range. The only column with an extreme outlier is the first queen move. One game had over 600 moves until a queen was moved, which must have been some sort of experiment. The scatterplot of first queen move is almost unreadable without this outlier removed.

Figure 3 shows the fully-functional outlier removal tab. This updates the dataframe, which in turn updates all the graphs (the page may need to be refreshed). After pressing the "Reset Dataset" button and refreshing, the dataset and graphs go back to the original 50,000 line dataframe.

5 Principal Component Analysis (PCA)

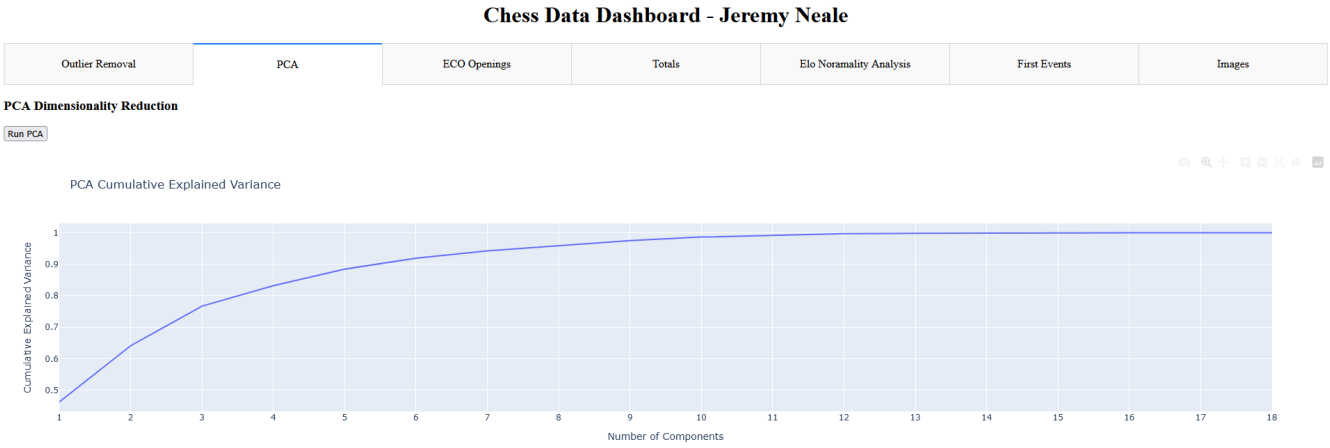

Figure 4: PCA on the 50,000 line dataset

This simple tab in the dashboard runs PCA on the current dataset (which can be modified from its original state in the outlier removal tab) [5] [6]. It uses the scikit-learn Python package and the PCA class to do most of the computation, and then Plotly to graph the number of components explained by the cumulative variance.

Figure 4 shows the cumulative variance explained by the number of components. We can see that it resembles a logarithmic function, suggesting diminishing returns.

6 Normality test

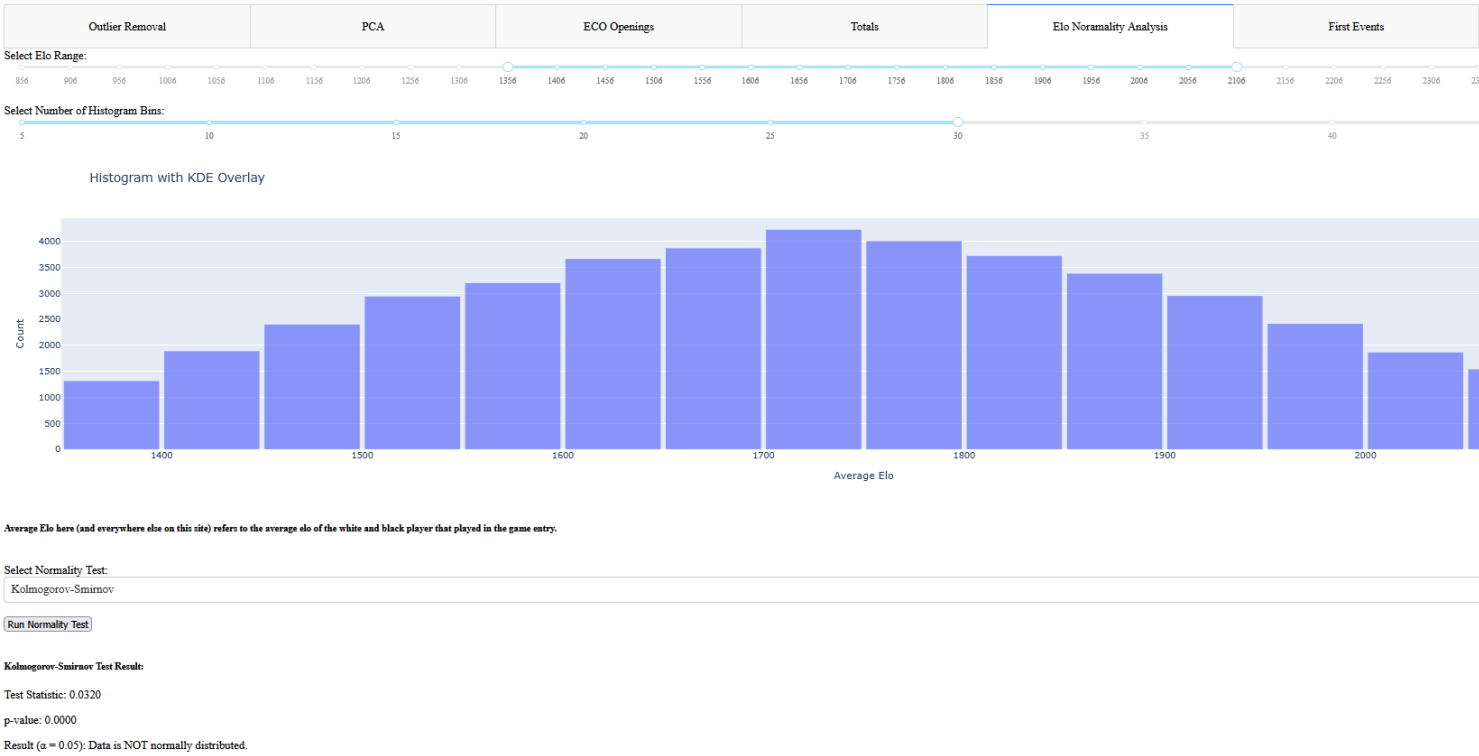

Figure 5: Normality test run on the Elo normality tab, with 2 sliders that control the Elo range and number of bins

Figure 5 shows the Elo Normality Analysis tab, where a normality test can be run on the selected Elo range [2]. A dynamic histogram plot shows the distribution, with the number of bins also as a slider than updates live.

3 different normality tests are available from the dropdown menu:

- Shapiro-Wilk
- Kolmogorov-Smirnov
- D'Agostino-Pearson

The data is not normally distributed, and the p-value is mostly 0.0000. This is because such a large sample of data is being used. There is even a warning in the terminal whenver the code runs, because it says anything more than 5,000 sample is unlikely to be normally distributed.

Elo is also typically skewed to the right, with a small number of International Masters and Grand Masters having very high elos compared to everyone else.

7 Heatmap & Pearson correlation coefficient matrix

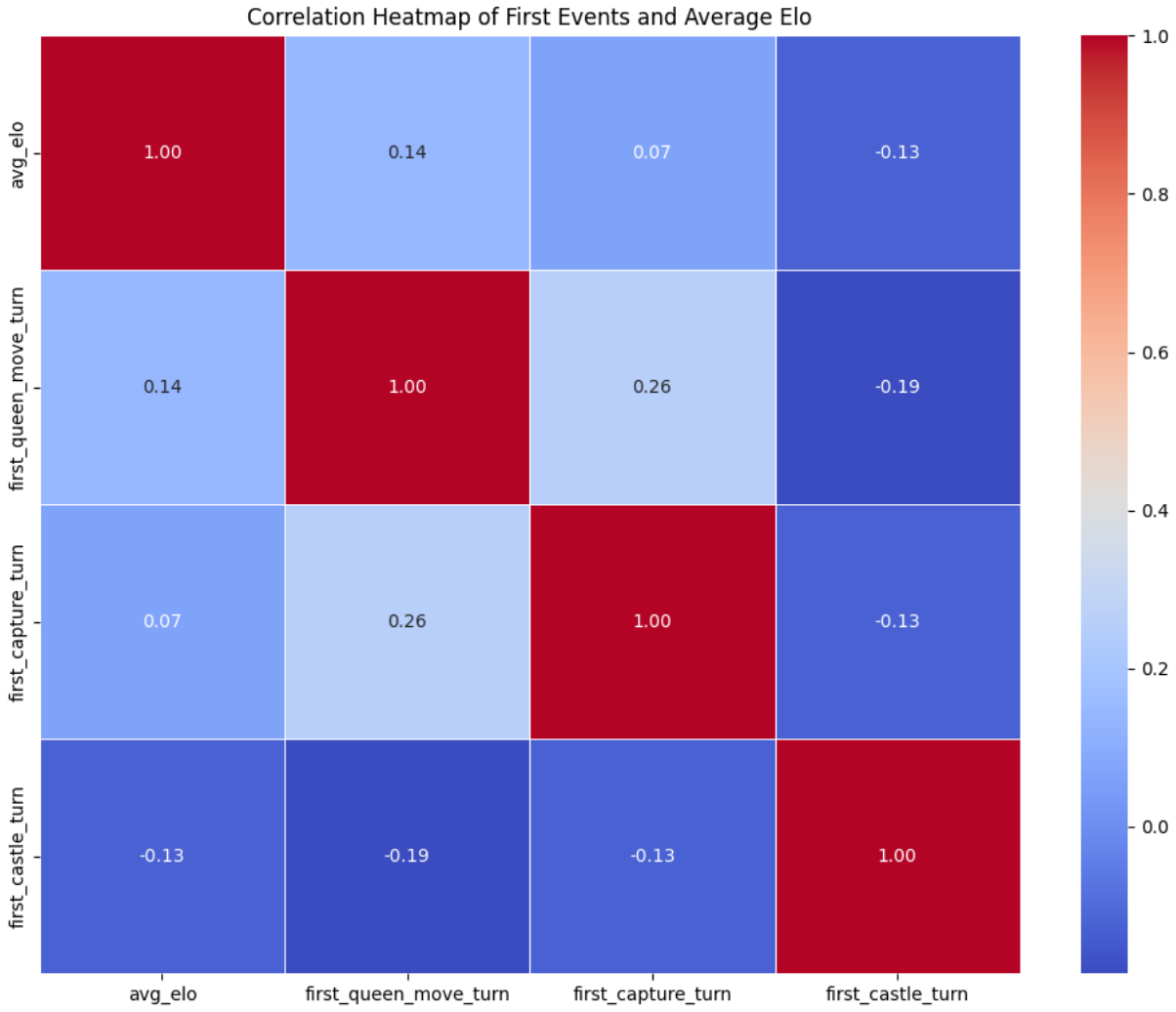

Figure 6: Correlation matrix heatmap of first events and average elo

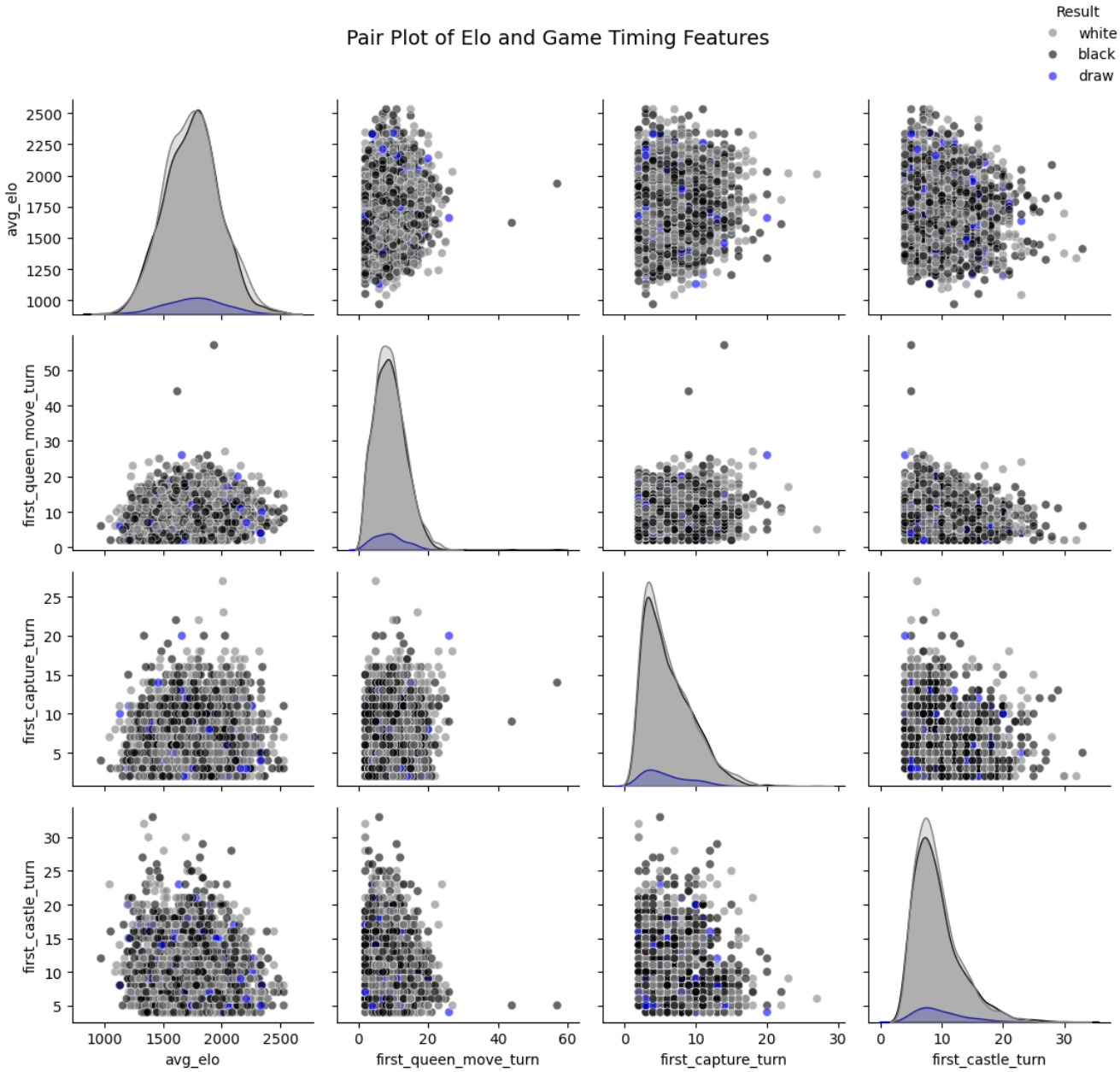

Figure 7: Pair Scatter Plot of first events and average elo

Figures 6 and 7 show the relationship between all of the first events that were feature engineered and average elo of the match. This was one of the first ideas that came to my mind when I found this dataset; I wanted to study how higher level players did things compared to lower level players. I've always learned that you shouldn't bring your queen out too early and you should castle very early on. I wanted to see if higher level players follow these trends, and how they influence the result of a chess match.

Based on the pair scatter plot, we can conclude that there seems to be a negative linear relationship between first turn capturing and first turn castling. There is also a weaker negative linear relationship between first castle turn and first queen move turn.

Both of these make sense, because some openings go for trading pawns in the center or bringing your queen out, and it is difficult/impossible to capture pieces early and also castle early.

We can also see a very slight correlation between bringing a queen out early and a piece being capture earlier.

This makes sense because there are a few common queen traps that aim to win a pawn at the center with a check on the king.

8 Statistics

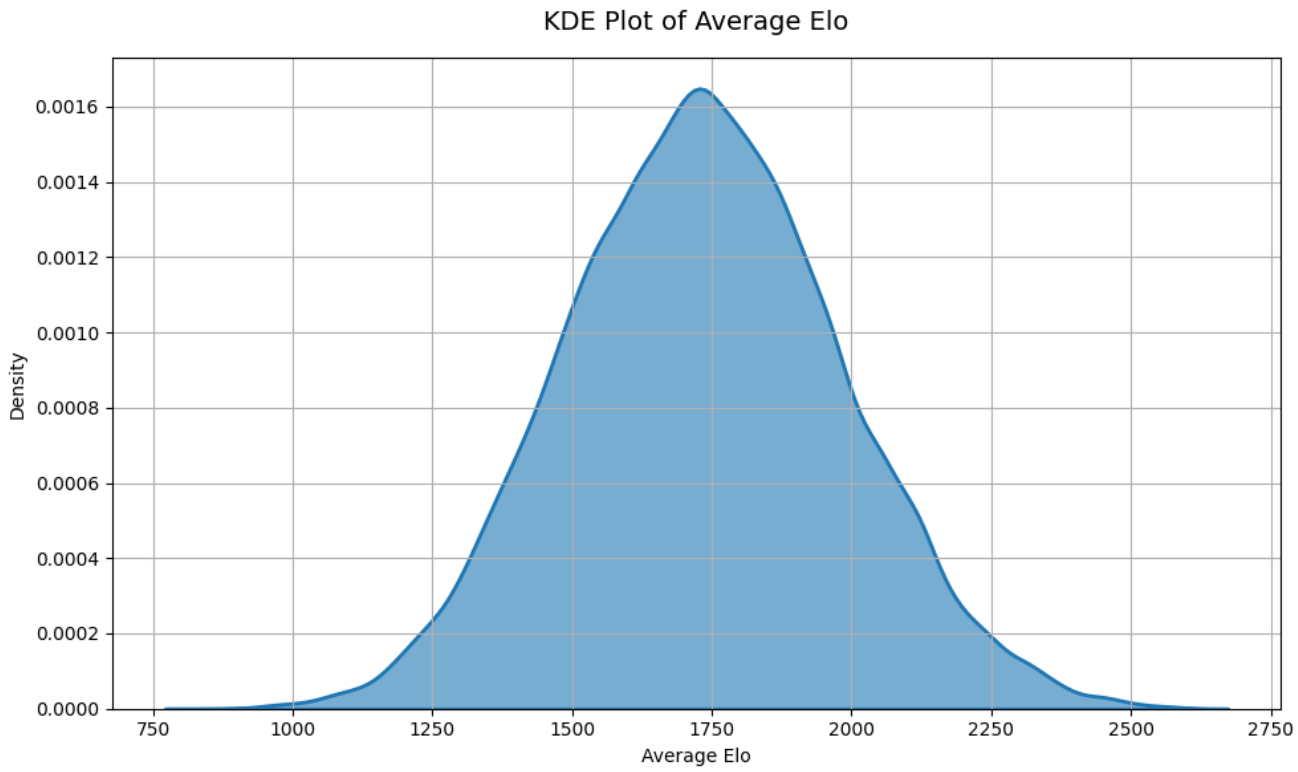

Figure 8: KDE Plot of average elo

The KDE of elo in figure 8 shows the distribution of elo for the chess matches. The elos here are extremely high, and online research showed me that Lichess uses a different elo system than Chess.com (most popular chess website). On Chess.com, the average elo is about 800, while on Lichess it is around 1500. We can see that while this distribution appears somewhat Gaussian, we determined that it is not with 3 different normality tests.

There are many other statistical measures we can perform on the dataset. We can see that 37% of all games played are Blitz, and 24% are classical. We can see that the universal win rates are White at 50%, Black at 46%, and 4% end in a draw. The next section displays many useful statistical metrics of the dataset.

9 Data visualization

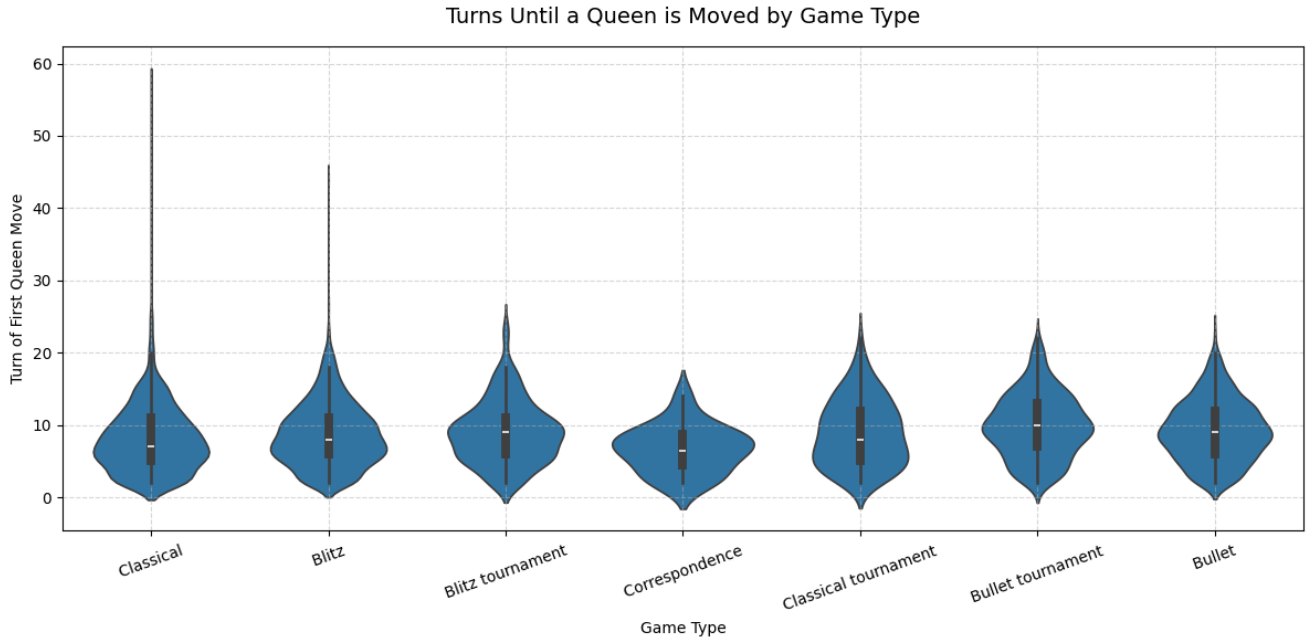

Figure 9: Violin plot of first queen move by game type

Figure 9 shows a violin plot of the first queen move by game type. Like many of the first event charts, I wanted to uncover if there were any factors that led players to making certain moves first, like bringing a queen out. Bringing a queen out early is too widely considered a rash an undisciplined move, and my intuition made me think that shorter games (Rapid and Blitz) would have more overly aggressive players.

We can see from the violin plot that it is actually the opposite. Classical appears to have a lower median than bullet and blitz, which is very surprising. Classical is the longest time format, so I would have assumed that players are more patient and wait to bring their queen out.

Conclusion: Players in longer time formats tend to bring out their queen earlier.

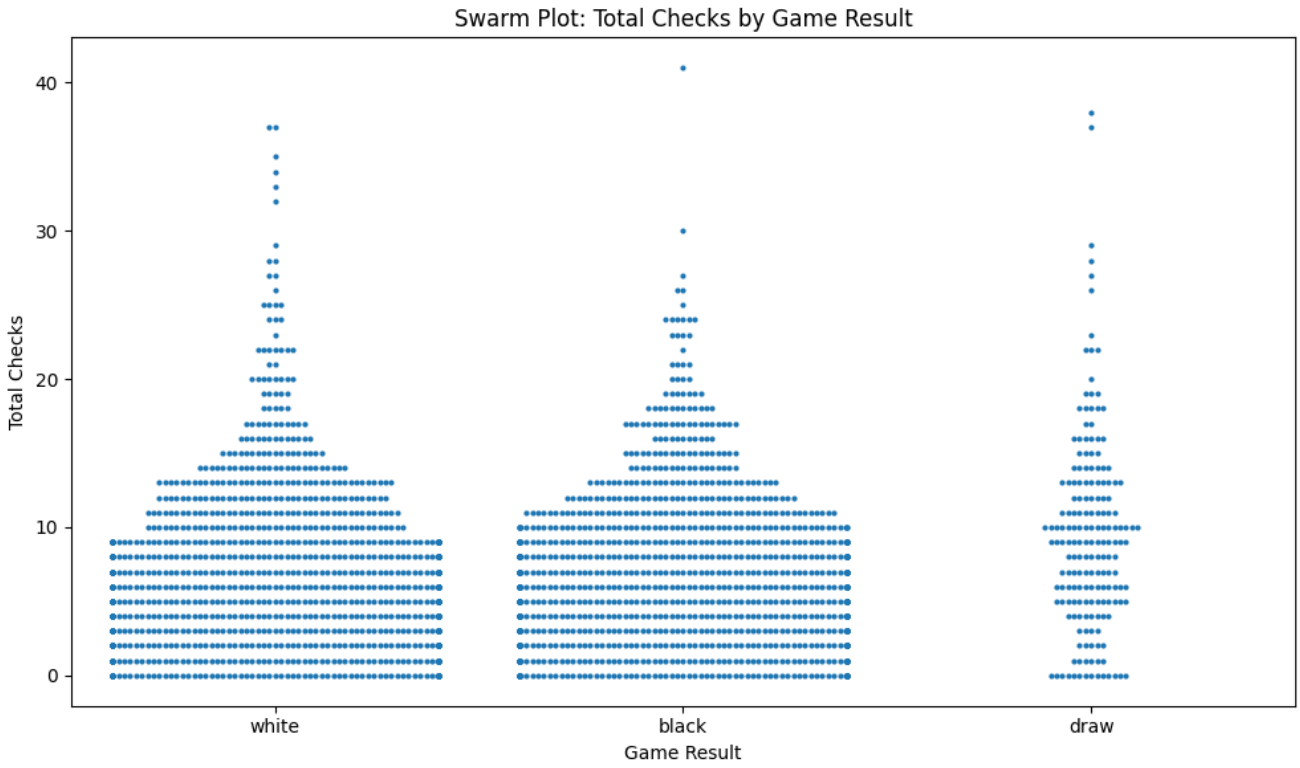

Figure 10: Swarm plot of total checks by game result

Figure 10 shows a swarm plot of total checks by game result. I wanted to make this plot to see if when certain colors won if they were likely to have more checks than if the other color won. Aside from a few outliers, I don't think there is any conclusive evidence that either of those is the case. The only conclusion that I think could be made is that games with draws tend to have slightly more checks. This is unsurprising because one of the main ways to get a draw is through repitition, and one of the main ways to force that is through perpetual check.

Conclusion: Draws tend to have more checks.

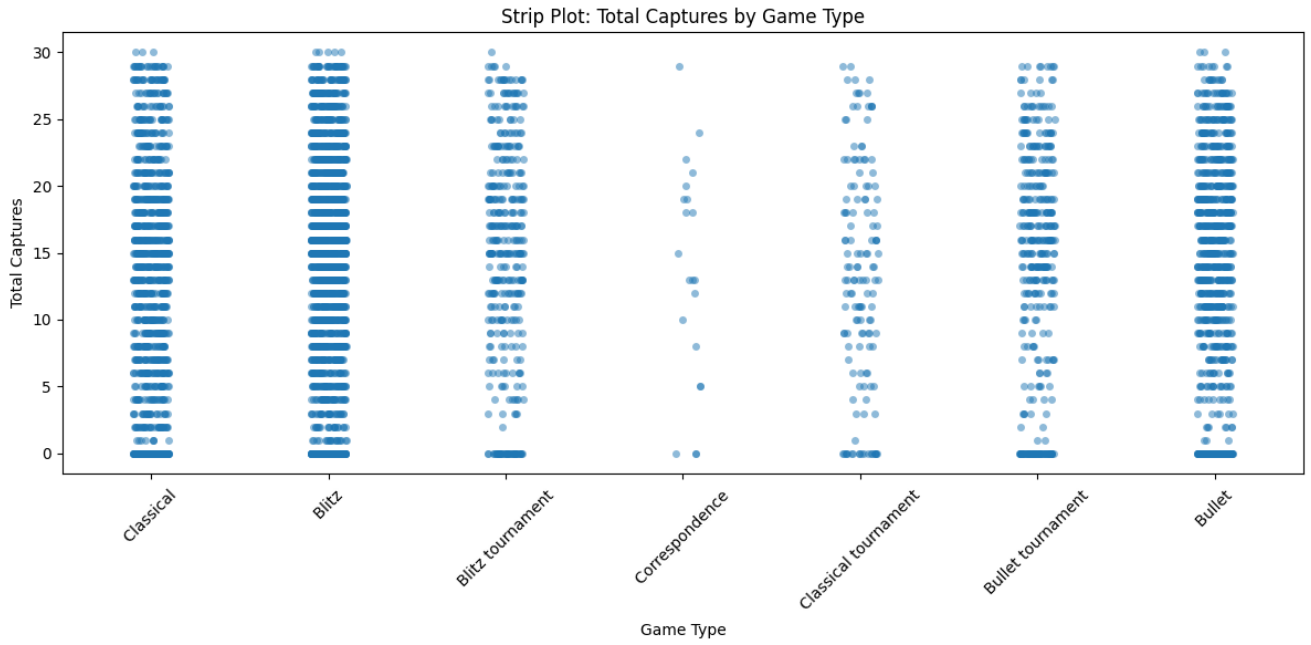

Figure 11: Strip plot of total captures by game type

Figure 11 shows a strip plot of total captures by game type. I personally do not like the strip plot and find it hard to visually interpret. I was hoping to see if certain game styles had more captures, and my intuition assumed that quicker games would have more captures due to trading off pieces more freely. I don't think there's enough evidence to make any conclusions.

No conclusion.

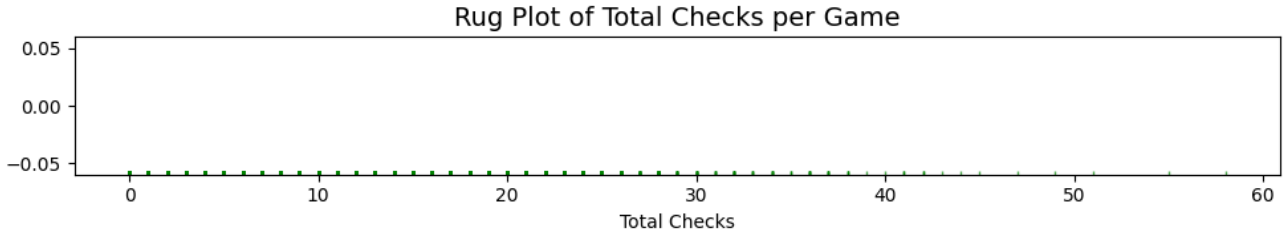

Figure 12: Rug plot of total checks per game

Figure 12 is a rug plot of total checks per game. Similar to the strip plot, I am not a fan of rug plots and found this chart almost impossible to interpret. I'm not sure if there was an issue with the data or the chart creation or scaling, but there is almost nothing I can interpret here.

No conclusion.

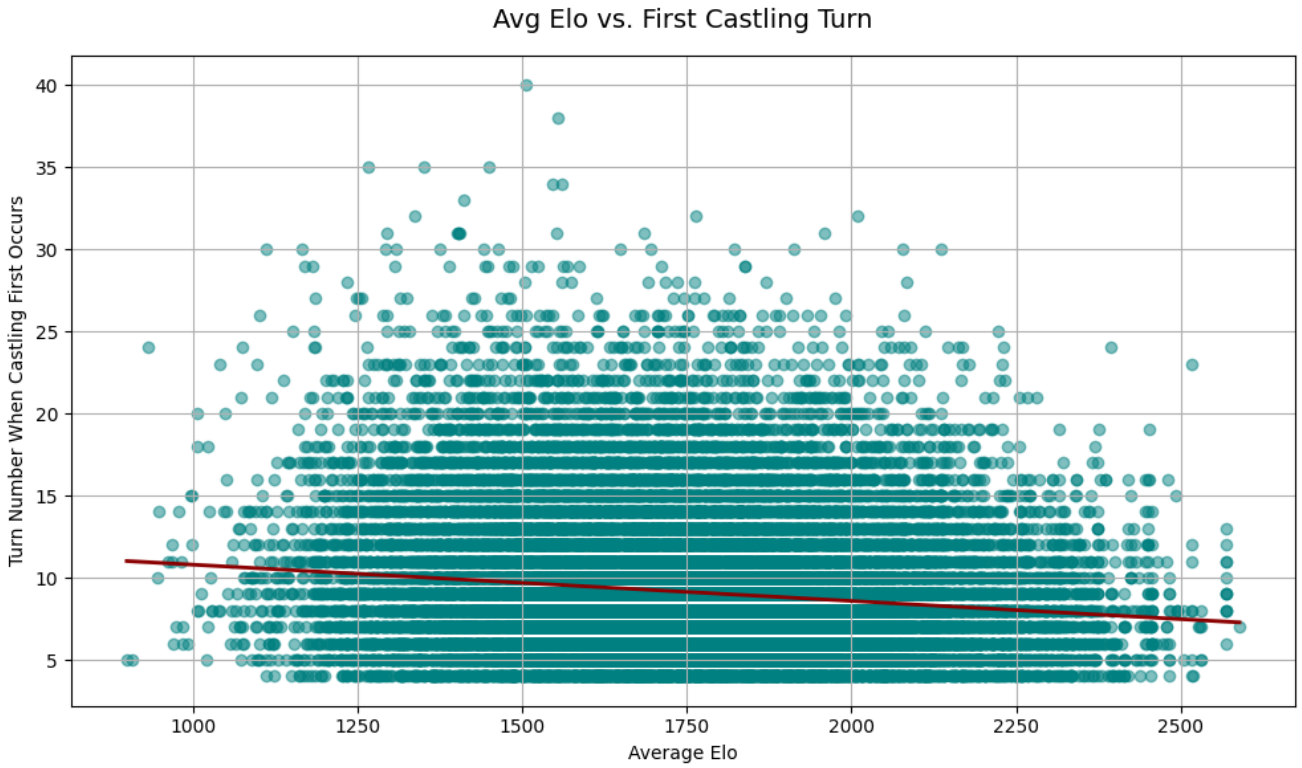

Figure 13: Reg plot of average elo vs first castling turn

Figure 13 shows a reg plot of average elo vs first turn castling from either player. This was one of the first charts I was eager to make when I initially had the idea of doing a chess dataset, and I wanted to see how much earlier stronger players castled and if that pattern was true at all. We can see that while it is a slight trend, there is a clear negative linear relationship between elo and turn numbers until a player castles. With this sample size, we can make a conclusion, even with a slight slope.

Conclusion: Higher elo players tend to castle earlier.

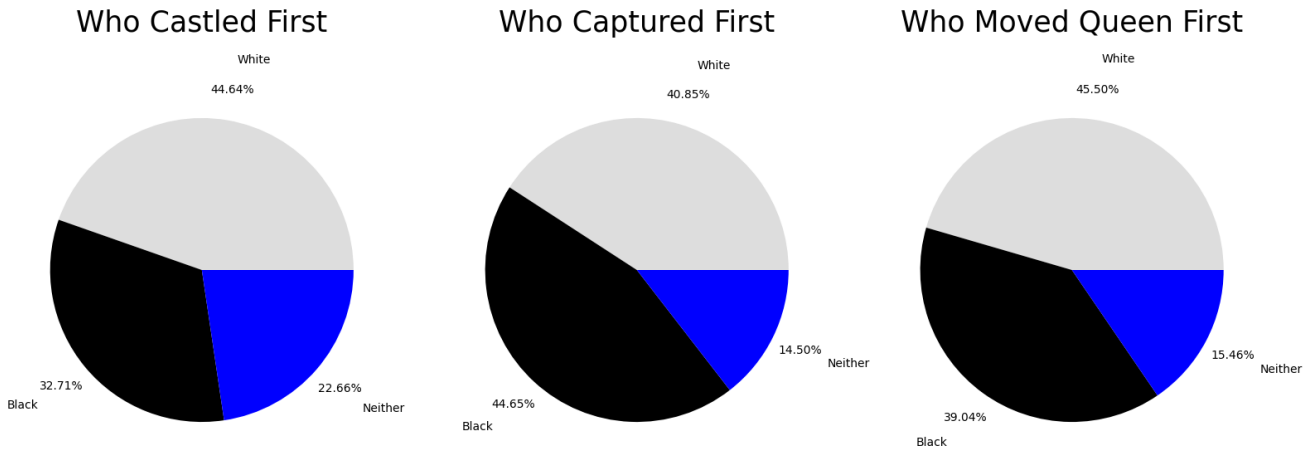

Figure 14: 3 pie charts of first events by color

Figure 14 shows 3 pie charts of the 3 measured first events by color. This was also one I was curious to see when I found this dataset. My intuition thought that white would castle first (since they go first), but black may be the agressor and bring their queen out first.

Based on the data, white castles first, by a margin of over 10%. White also tends to move their queen first, with a smaller margin of about 5%. These are unsurpring considering that white goes first.

However, the surprising result here is that black tends to capture the first piece and a significant rate, with a margin of almost 4% more often. This initially sounds surprising, but many gambits (tactical sacrafices) in openings are generally performed by white, such as the Queen's Gambit, King's Gambit, Vienna Gambit, etc. Black also has common openings trying to trade off white's center pawn, such as the French or Caro-Kann.

To my surprise, white tends to move their queen first much more often, with a margin of over 5%. This is mostly due to the fact that white goes first, but could also because their are a few popular white openings like the Scholar's mate that bring the Queen to c3, looking to attack the weak f7 pawn.

Conclusion: White tends to castle first and move its queen first, but black tends to make the first capture.

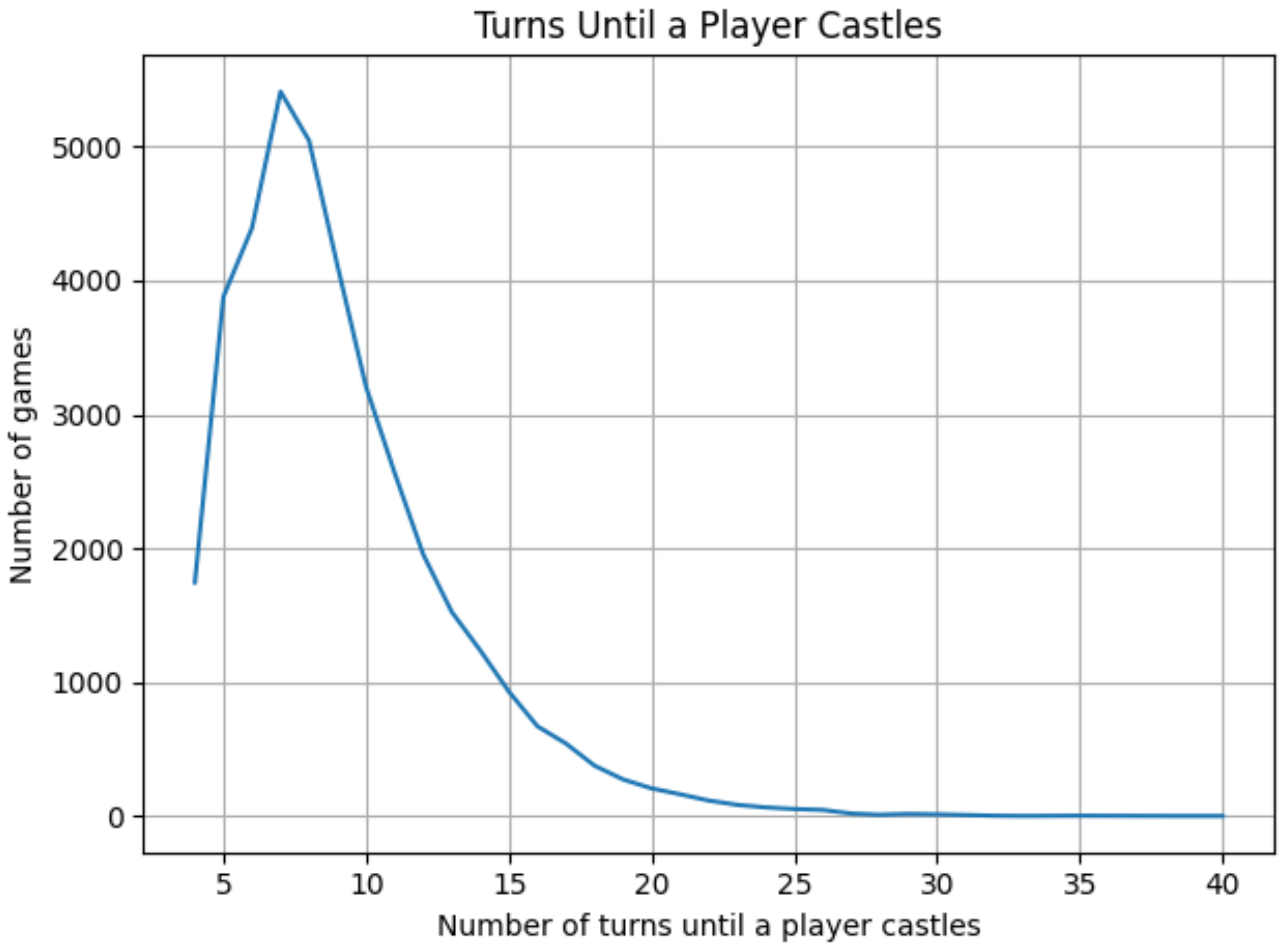

Figure 15: Line plot of number of games vs number of turns until a player castles

Figure 15 Shows the number of turns until a player castles. The most common number of turns until a player castles is 7, with 8 and 6 as second and third. The minimum number is 4, since a player would need to move a pawn to make room for the bishop to move, the knight to move, and then the fourth turn can be a kingside castle. 5 moves is the minimum to castle queenside, since the queen also needs to be moved out of the way. This data is skewed right because their is a minimum of 4, but no maximum, with some games taking a long time until someone castles.

Conclusion: Turns 4-11 are the most common number of turns until a player castles, with 7 being the mode. The data is skewed right.

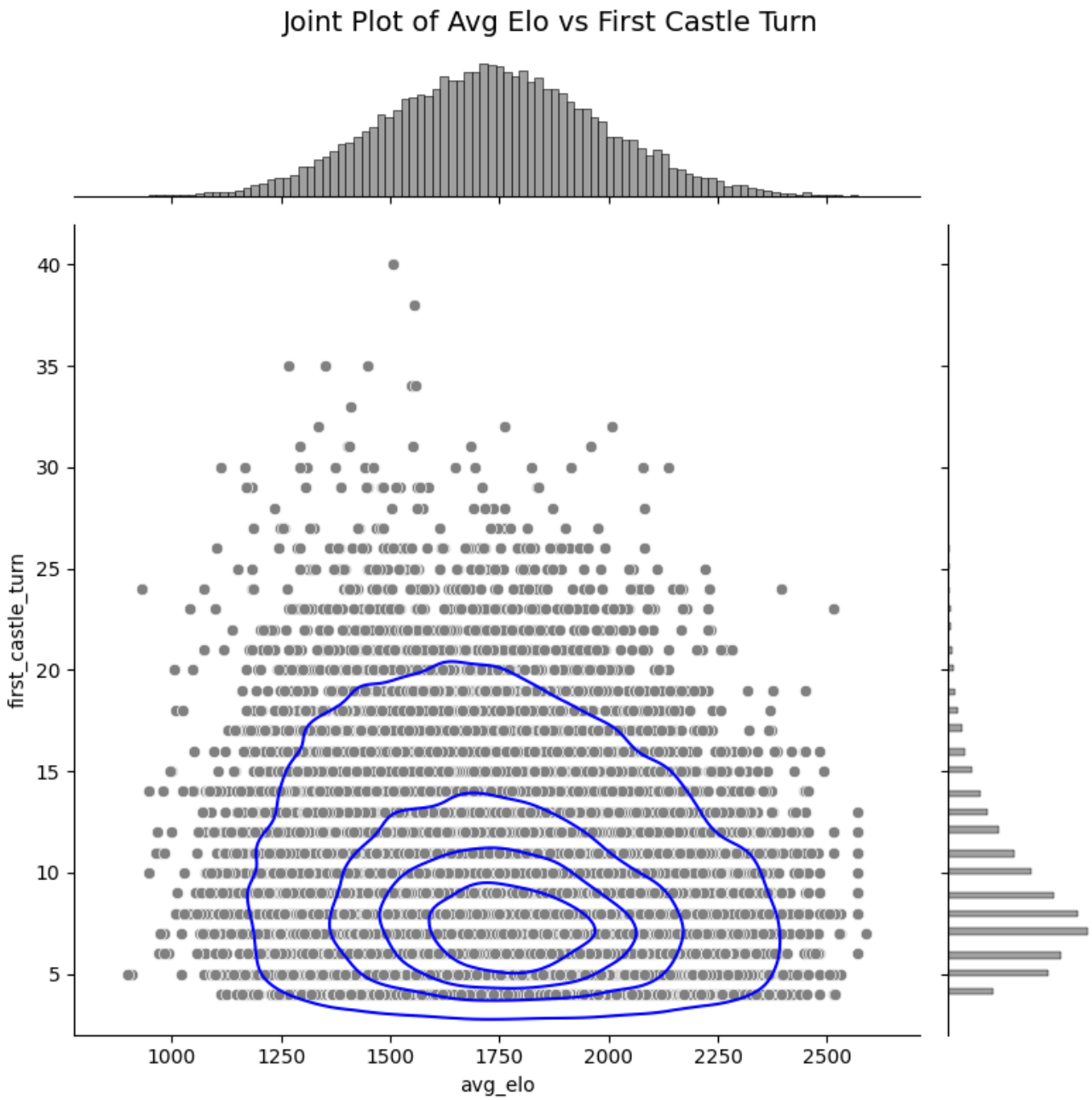

Figure 16: Joint plot of average elo vs first turn castling

Figure 16 is a joint plot that shows a similar theme across the first event plots, comparing first castle with average elo. This plot is interesting because it shows the histograms on the axes, and the scatterplot with KDE in the graph. We can see a slight downward trend in the scatterplot, suggesting (similar to the reg plot) that higher elo players tend to castle earlier.

Conclusion: Higher elo players tend to castle earlier.

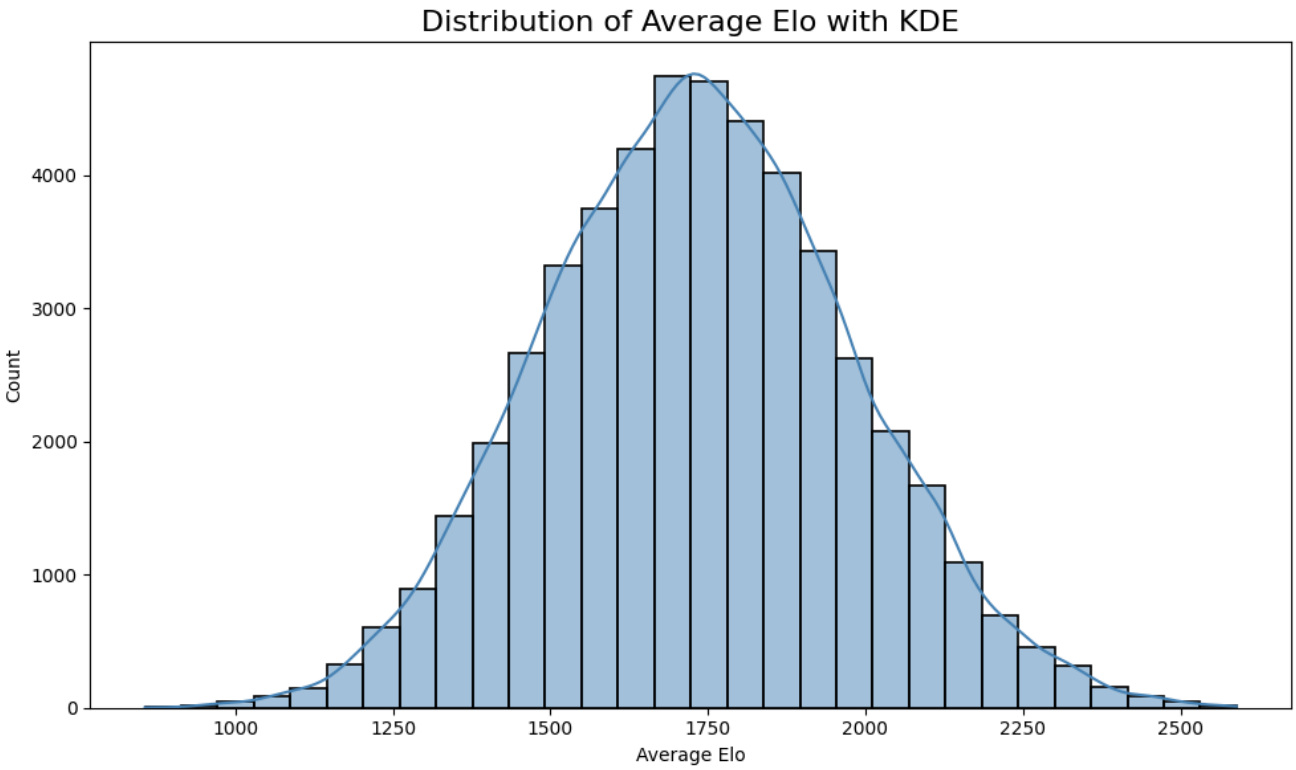

Figure 17: Average elo histogram with KDE overlay

Figure 17 shows another histogram of the average elo distribution with a KDE overlay. Similar to the previous elo distribution graph, this appears roughly normally distributed, but all 3 normality tests confirm that it is not Gaussian.

No conclusion.

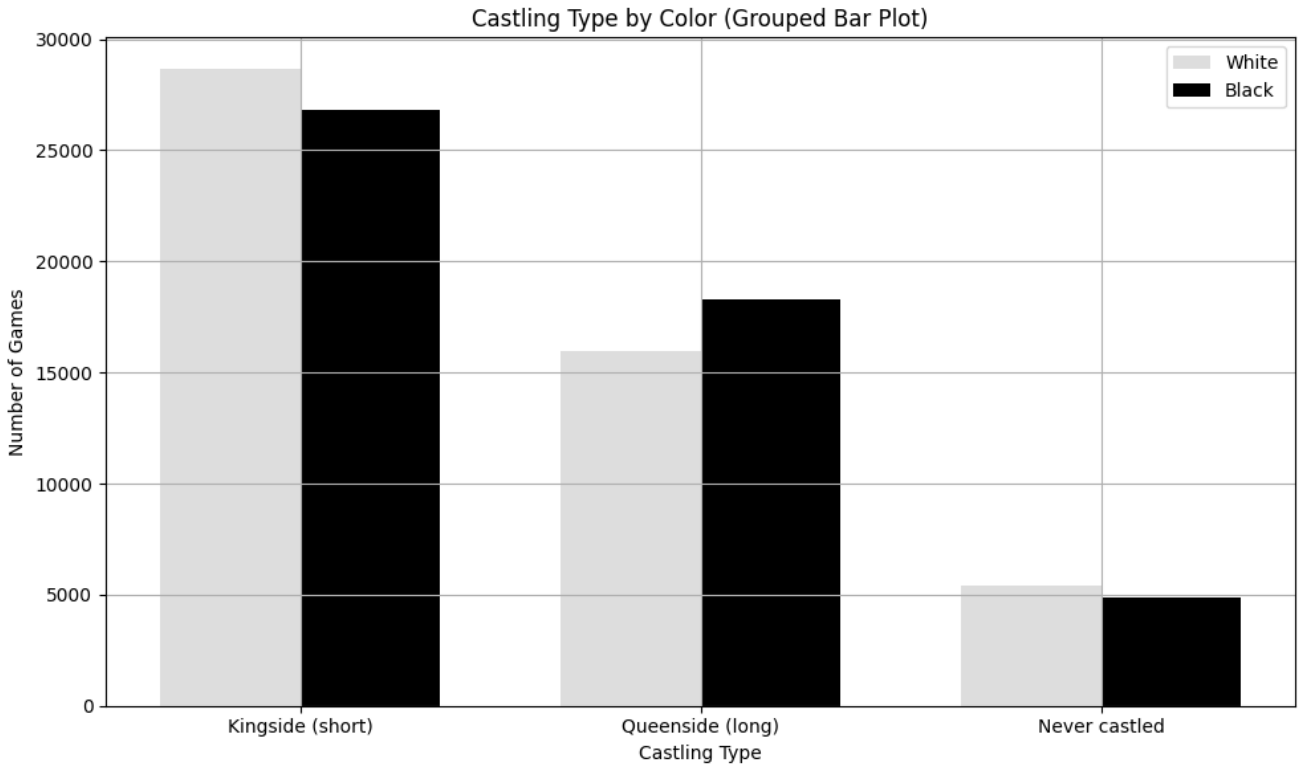

Figure 18: Grouped bar plot of castling type by color

Figure 18 shows castling type by color. This was one of the charts I was very curious about, because it is generally accepted that castling kingside is better and safer. However, castling queenside against kingside tends to make for a sharper game and is considered more confrontational. My intuition said that black would tend to castle queenside more and take the greater risk, since black is fighting from behind from the start of the game and more able to take on risk.

We can see that in general, castling kingside is far more common. However, we can see that white tends to play it safe more, castling kingside much more than queenside. For black, while they also tend to castle kingside more often, the gap between kingside and queenside castles is much slimmer. Black castles queenside at a significant rate higher when compared to white.

Conclusion: Black tends to castle queenside more frequently than white.

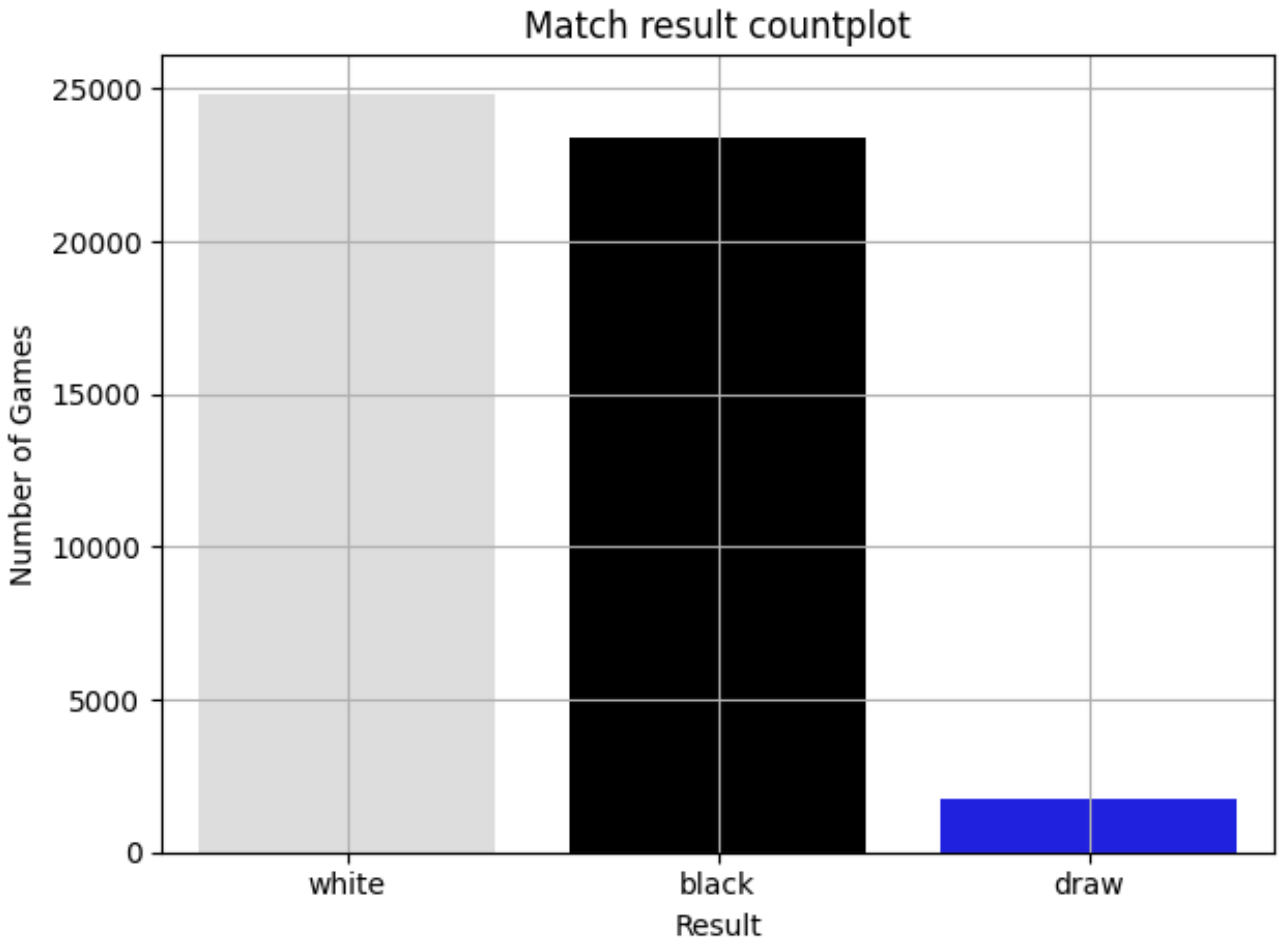

Figure 19: Countplot of match results

Figure 19 is a count plot of match results by color. Unsurprisingly, white wins games at a slightly higher percentage than black. This is because white makes the first move, and is up "tempo" for the entire opening phase of the game. Draws make up a small portion of game results.

Conclusion: White wins games more often than black.

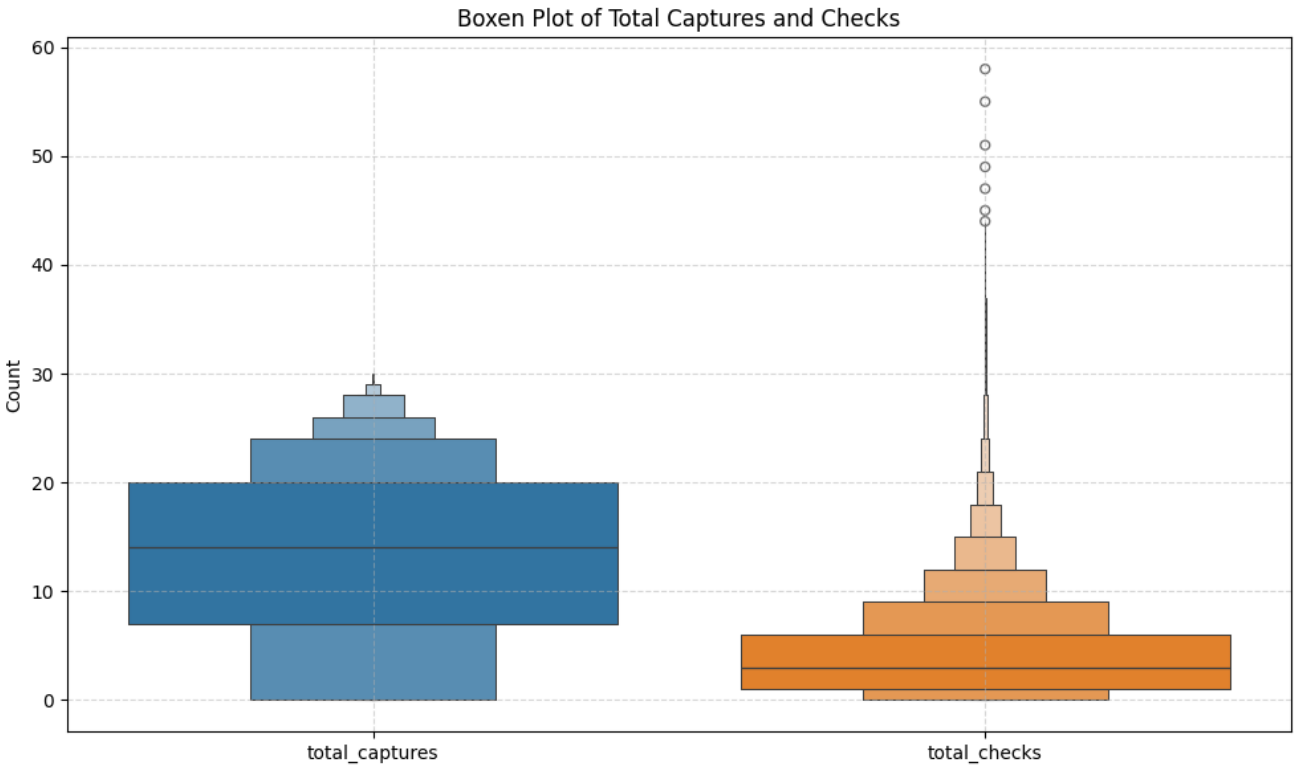

Figure 20: Boxen plot of total captures and checks

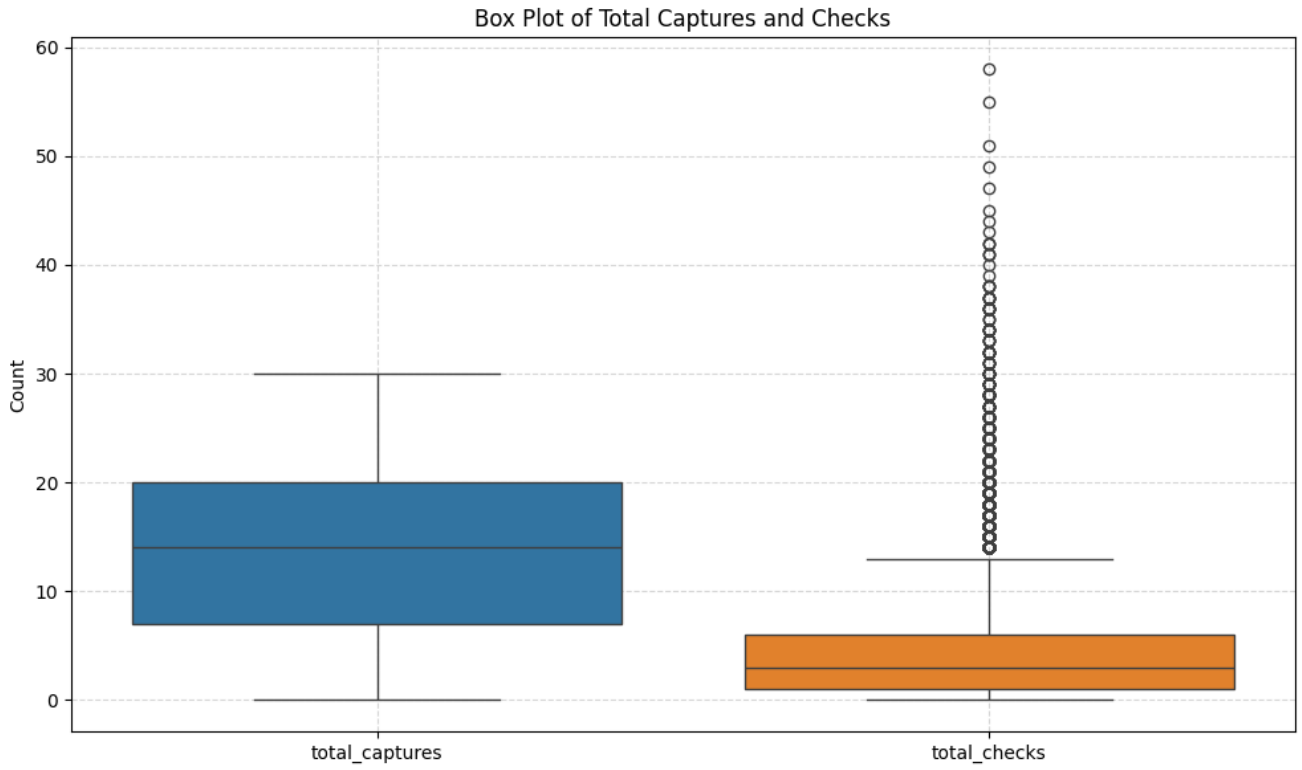

Figure 21: Box plot of total captures and checks

Figure 20 and 21 both show the distribution of total captures and total checks across all the games. This is interesting, as it mainly highlights the difference between the two chart types.

One thing to notice is that the total captures is capped at 30, meaning there are no outliers. This is because there are 32 pieces, and once the 30th piece is captured, there are only two kings left and the game is ruled a draw by insufficient material. However, there is a near-infinite amount of checks that can be in a game, and we can see some outliers in the data, with one game having 59 checks. There is in fact a limited number of moves a chess game can have, although the number is incredibly high. This is due to the rare **50-move rule**, where the game is ruled in a draw if no piece has been captured, or no pawn has advanced for 50 moves.

We can also see that total captures is very evently distributed, while total checks is skewed "right" here. The low and median value are very close together with checks (less than 5 apart), while the median captures fall almost perfectly between the low and high value.

Conclusion: Total captures has no outliers due to constraints, but total checks has many outliers. Total captures is much more evenly distributed than total checks, which is skewed right.

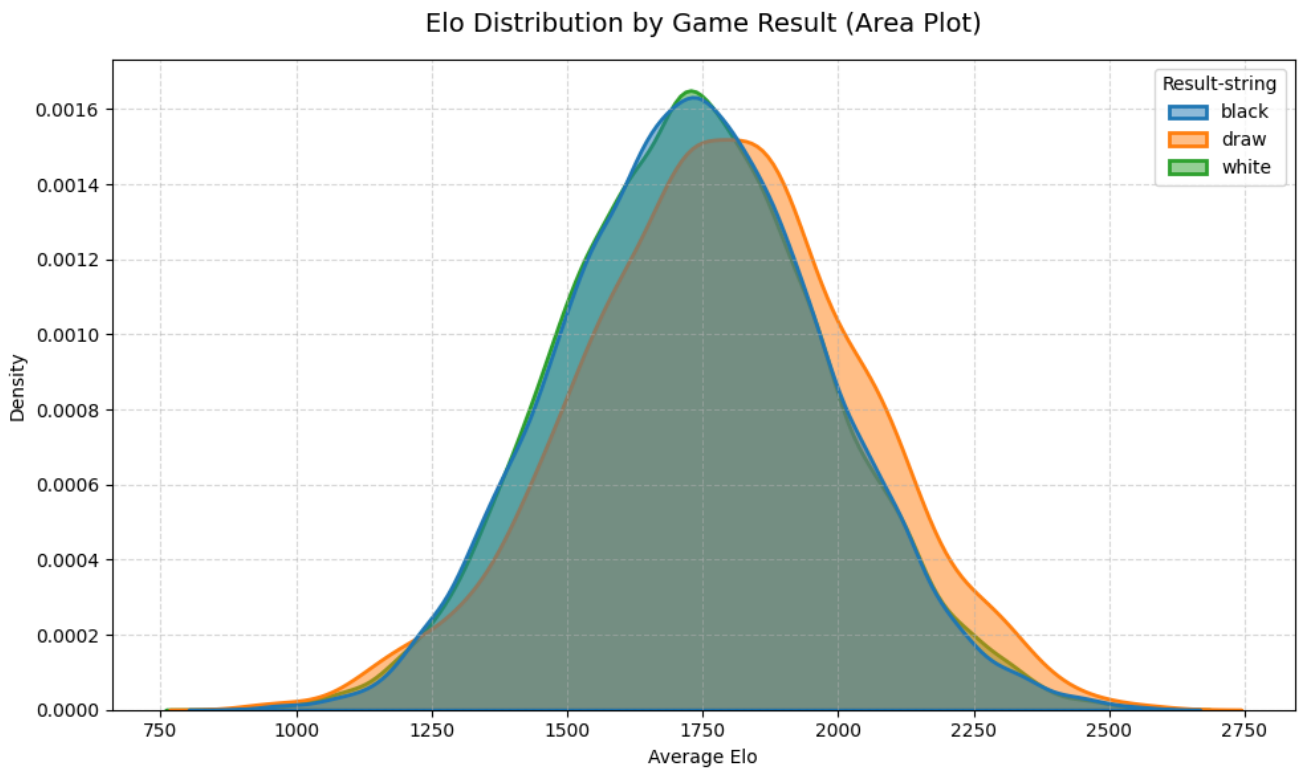

Figure 22: Dist plot of elo distribution by game result

Figure 22 shows a dist plot of elo distribution by game result. I was hoping for a more definitive answer with white vs black game results, but they appear almost identical. My initial intuition was that higher rated players win more often with white, and lower level players are more of a 50/50 split, but there isn't any conclusive evidence to support that.

We can see that without a doubt, higher rated players draw more. This could be due to many factors, but skill is without a doubt the leading cause here. Drawing a game from a losing position is a skillful manuever, and can save a player from a loss. Stronger players are seemingly better at forcing draws, through perpetual check or zugzwang, German for "compulsion to move", a chess tactic where you force your opponent to make a move that hurts them no matter the move. This tactic can be used with great skill to force a draw from a losing position.

Conclusion: Draws are more common for players with higher elos.

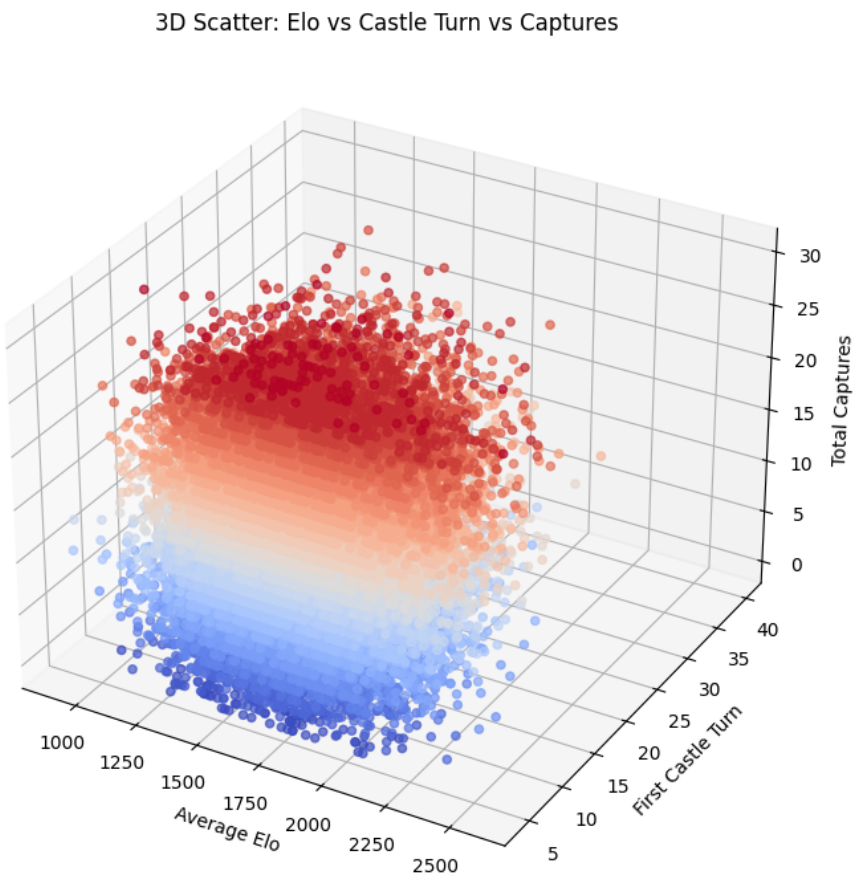

Figure 23: 3d Scatter plot of elo vs first castle turn vs total captures

Figure 23 shows a 3-d scatter plot of elo vs castle turn vs captures. I wanted to see the multivariate relationship between a few different variables, and I thought skill and first event decisions would show some insight.

Unfortunately, this graph is quite difficult to interpret in three dimensions, so no conclusions can be reached.

The 3-d version on the dashboard is far easier to interpret, since it can be moved around in any direction, making it much easier to perceive where the points are.

No conclusion.

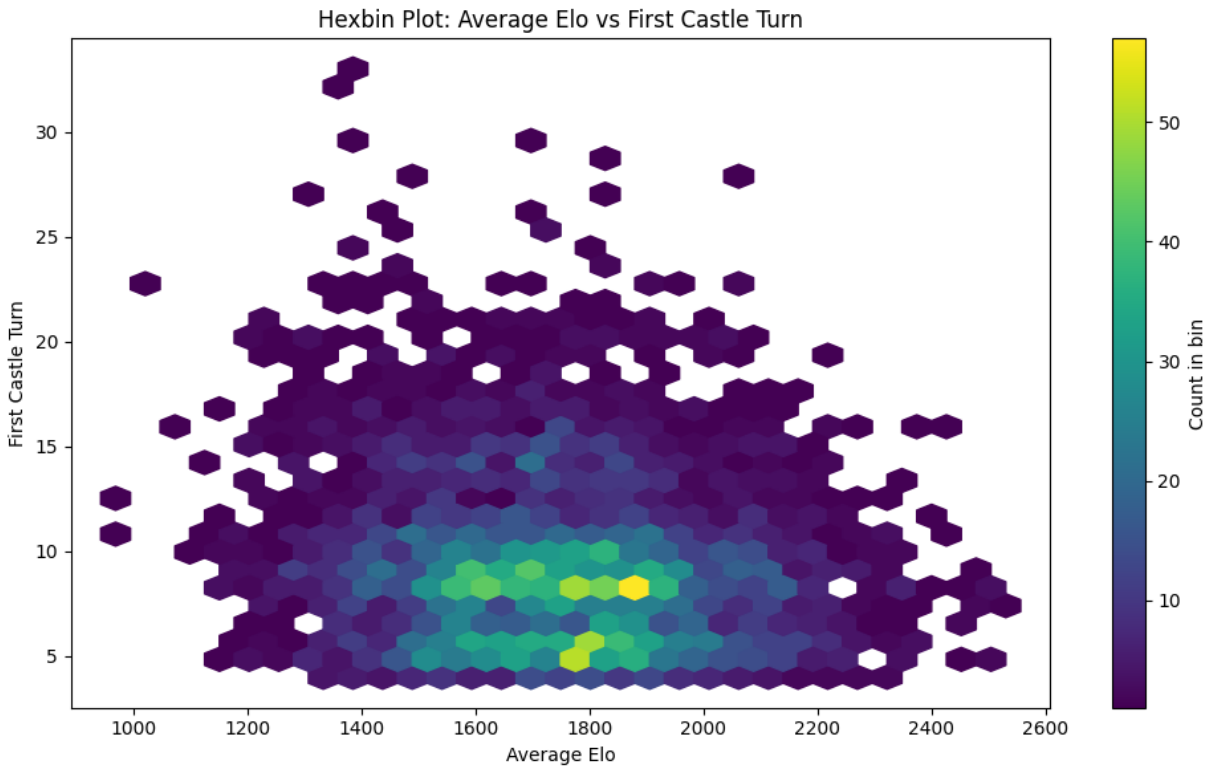

Figure 24: Hexbin plot of average elo compared to first turn castling

Figure 24 shows a hexbin plot of avg elo and first turn castling. Similar to the other castle plots, we see a slight negative relationship between elo and castling, indicating that higher players castle earlier. We see a concentration similar to the joint KDE plot around 5-7 turns, and around 1600-1850 elo.

Conclusion: Higher elo players tend to castle slightly earlier.

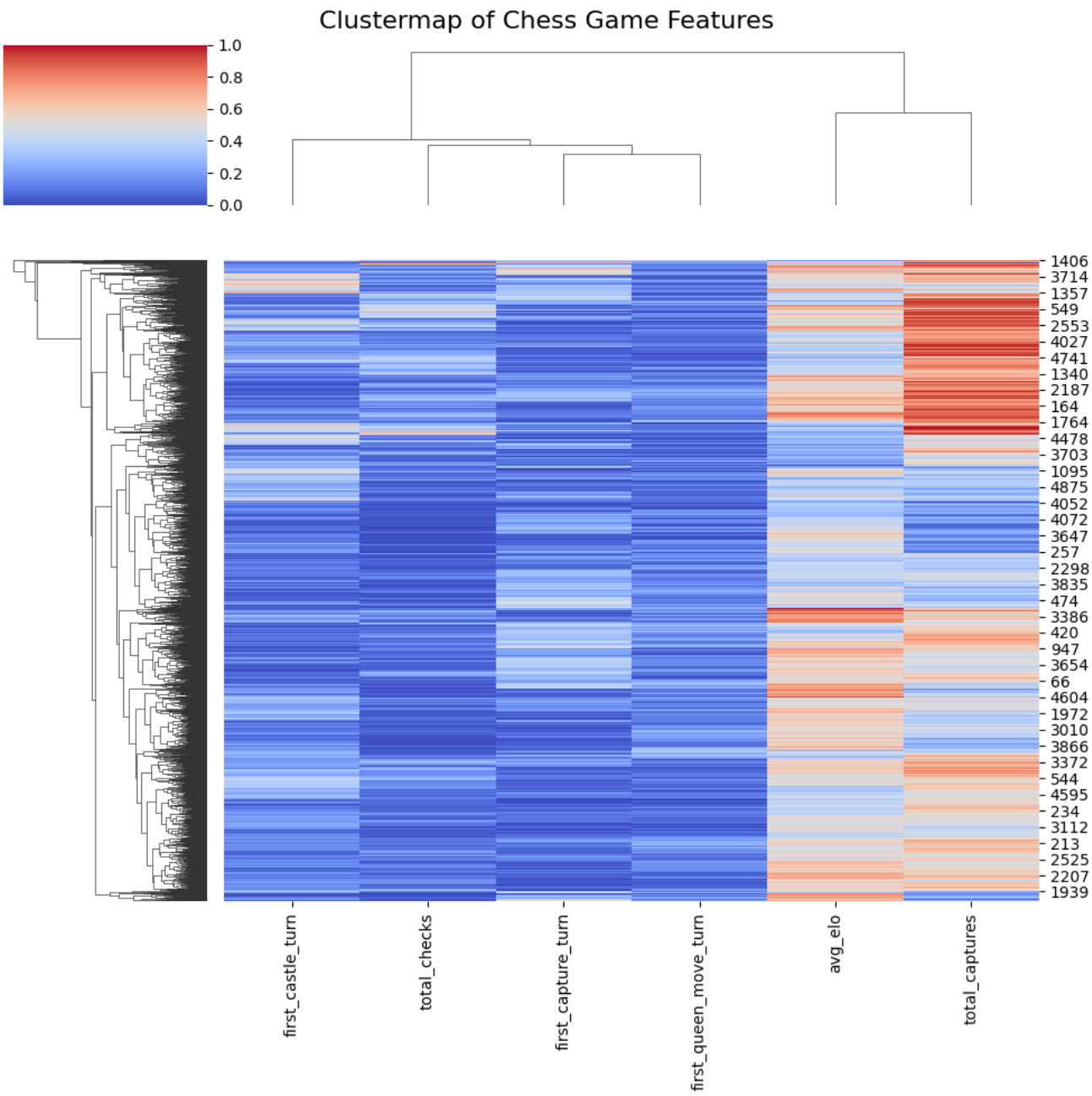

Figure 25: Clustermap of numerical attributes

Figure 25 shows a clustermap of many of the numerical attributes from the dataframe. We can see that average elo and total captures are more dense than the others, suggesting similar patterns. We can interpret that higher elo games tend to be more conservative early on, while lower elo games jump into action quicker.

Conclusion: Higher elo games tend to be more conservative, and total captures is a good indicator of how other aspects of the game went.

10 Dashboard

The Outlier Removal, PCA, Elo Normality Analysis, and ECO Openings tabs have previously been displayed, so this section will show the remaining tabs. The dashboard was created with Plotly and Dash [8] [4].

Figure 26: The Totals tab, which shows the total checks and captures and a strip plot below (not pictured)

Figure 26 shows the totals tab, which has a multiple selection dynamic input that displays white, black, draw, or a combination of them on the plots below.

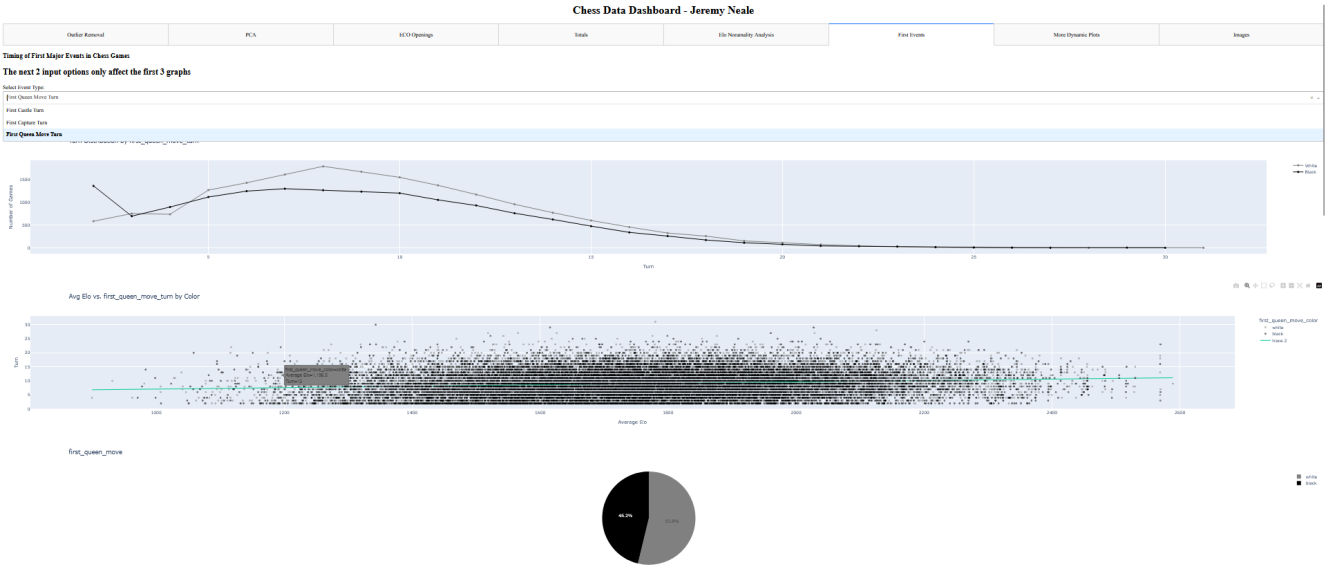

Figure 27: The First Events tab, with the first 3 graphs that take 2 dynamic input parameters

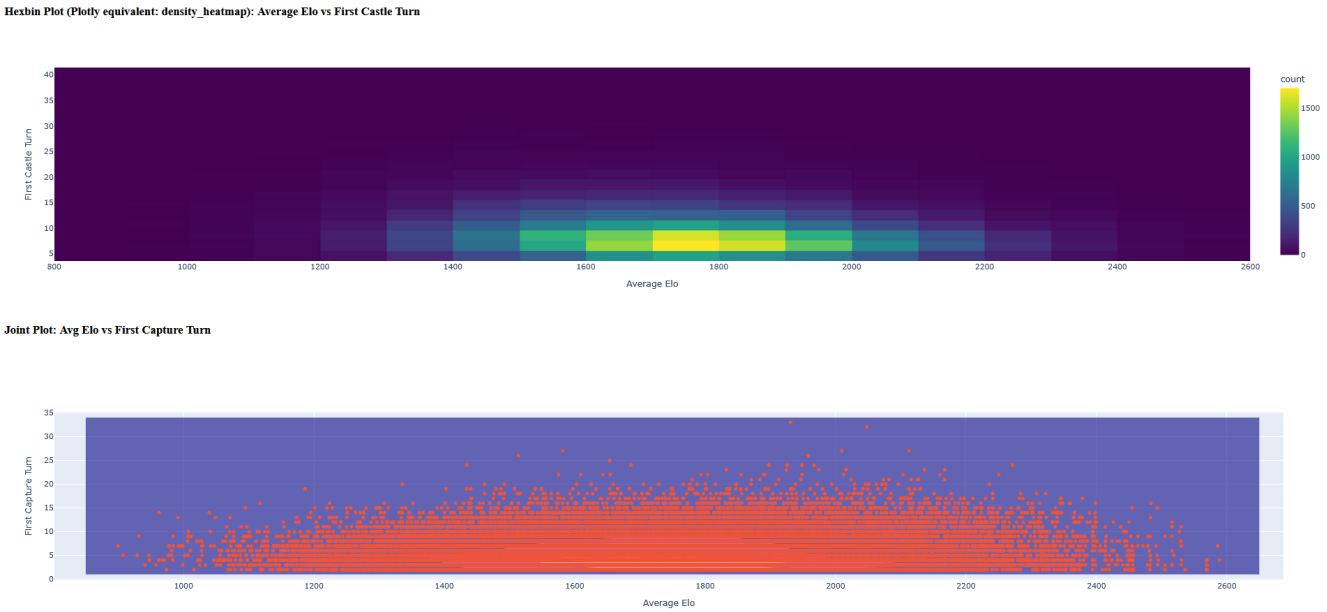

Figure 28: 2 more plots from the First Events tab

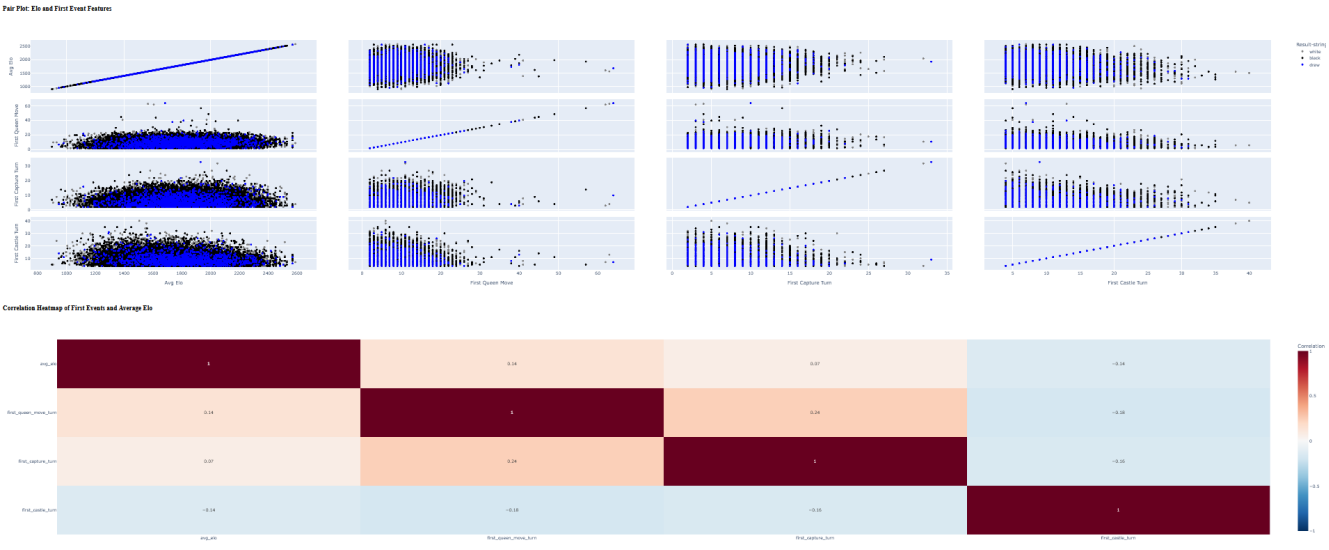

Figure 29: More plots from the First Events tab

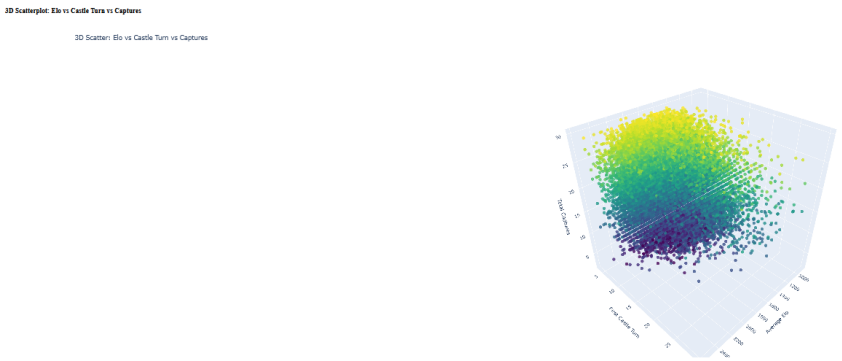

Figure 30: The last plot from the First Events tab

Figures 27, 28, 29, and 30 show the First Events tab, which has two dynamic selection options: a dropdown to select the event (first castle, first queen move, or first castle) and a RadioItem to select if the chart will show the data combined or separated by color. These 2 inputs only affect the first 3 graphs, the line plot, scatter plot, and pie chart.

The other images are the other dynamic graphs on this tab, all made with Plotly Express or Plotly Graph Objects [4] [3].

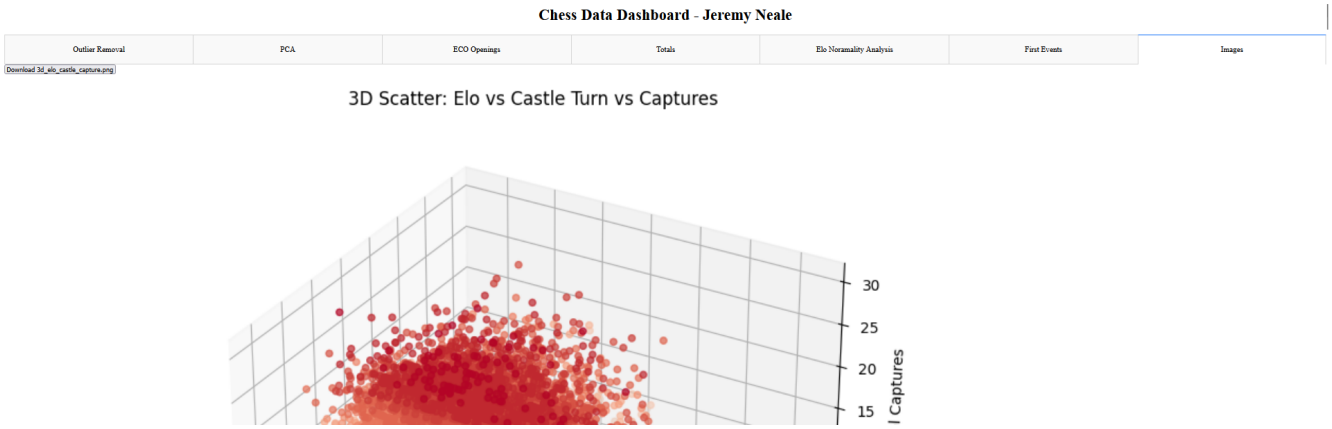

Figure 31: The Images tab

Figure 31 shows the images tab, which shows all the static plots, along with a download button for each. This tab is not shown in full because they are the same pictures from the Data Visualization section.

Figure 32: The More Dynamic Plots tab

Figure 32 shows the More Dynamic Plots tab. These plots didn't fit into the other main categories, so they are in their own tab. Both of these plots are dynamic, since they allow users to select items and see the count, zoom, download, and do all the other benefits of Plotly graphs.

Closing noes for the dashbaord Many of these use Plotly Graph Objects (`go.Figure(...)`) [3], since these plots are not all available with Plotly Express. All Plotly Express figures are actually built on Graph Objects under the hood, but not all plot types (such as hexbin and joint plots) are supported by Plotly Express. Thus, I had to manually create these plots as best I could with the Graph Objects package.

A few of the static plots (QQ and cluster map) did not have ways to graph in Plotly or even go.Figure(), so I left them as static plots in the Images tab.

There are more dynamic plots on the website than are shown here in the tab screenshots.

11 Conclusion

After creating all of these plots, I can say that many of my intuitions were correct, while some of them were wrong. I was the most surprised to see that black tends to make the first capture, but this made sense after I remember how common gambits are for white and not black.

The interactive python web application is great because it allows customization. The dashboard is fully-functional, the only drawback being the Totals tab takes a bit longer to load (maybe 10 seconds). Users can manipulate the data by selecting the outlier range. They can choose their own range of elo and the number of bins and perform a normality test. They can do a PCA with the trimmed dataset. I think the most useful graph of all is the ECO openings graph, which allows a user to select an opening they're curious about and see how elo affects the distribution of wins and losses for both colors and draws. I hope this dashboard helps people see trends across this many chess games and find new insights that I missed. I hope people use it to find new ECO openings that will dominate their elo range.

I plan to make a YouTube video that deep-dives into the graphs for fun, showing the process I used to make it as well as showing the graphs and dashboard.

Summary of conclusions:

- The elo distribution is not Gaussian.
- There is a minor positive correlation (r=0.26) between the first turn a capture occurs and the first turn a queen is moved.

- There is a negative correlation between the first turn a capture occurs and the first turn a castle occurs.
- Players in longer time formats (Classical) tend to bring out their queen earlier than shorter time formats (Blitz, Bullet).
- Draws tend to have more checks.
- Higher elo players tend to castle earlier.
- White tends to castle and move its queen first.
- Black tends to make the first capture.
- Turn 7 is the most common turn for the first castle to occur (minimum moves to castle is 4).
- Black tends to castle queenside more frequently than white.
- Kingside castle is more common for both black and white.
- White wins more games than black.
- Total captures is fairly evenly distributed between 0-30 (no more or less can occur outside this range).
- Total checks is skewed right.
- Draws are more common for players with higher elos.
- Certain openings are better for certain colors at different elo ranges.
- Certain ECO openings are more prone to draws, and this range tends to be at higher elos.

12 Future work

The only major change I want to make in the future is to allow people to change the dataset. With the code open-sourced, this would be a quick fix for code-savvy users, but a URL upload box would be great, the only problem is that the data would need to be formatted the same.

One small but useful update could also be to choose the number of rows from the 6 million line dataset to use. Right now it is hard coded to 50,000.

References

[1] Arevel. Chess games dataset, 2022. URL: https://www.kaggle.com/datasets/arevel/chess-games.

[2] SciPy Developers. scipy.stats — statistical functions (shapiro, kstest, normaltest), 2024. URL: https://docs.scipy.org/doc/scipy/reference/stats.html.

[3] Plotly Technologies Inc. Graph objects in python, 2024. URL: https://plotly.com/python/graph-objects/.

[4] Plotly Technologies Inc. Plotly express — high-level interface for plotly in python, 2024. URL: https://plotly.com/python/plotly-express/.

[5] Scikit learn Developers. sklearn.decomposition.pca - scikit-learn 1.3.2 documentation, 2024. URL: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[6] Scikit learn Developers. sklearn.preprocessing.minmaxscaler - scikit-learn 1.3.2 documentation, 2024. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html.

[7] The pandas development team. pandas: Powerful data structures for data analysis, time series, and statistics, 2024. URL: https://pandas.pydata.org/.

[8] Plotly. Dash core components documentation, 2024. URL: https://dash.plotly.com/dash-core-components.