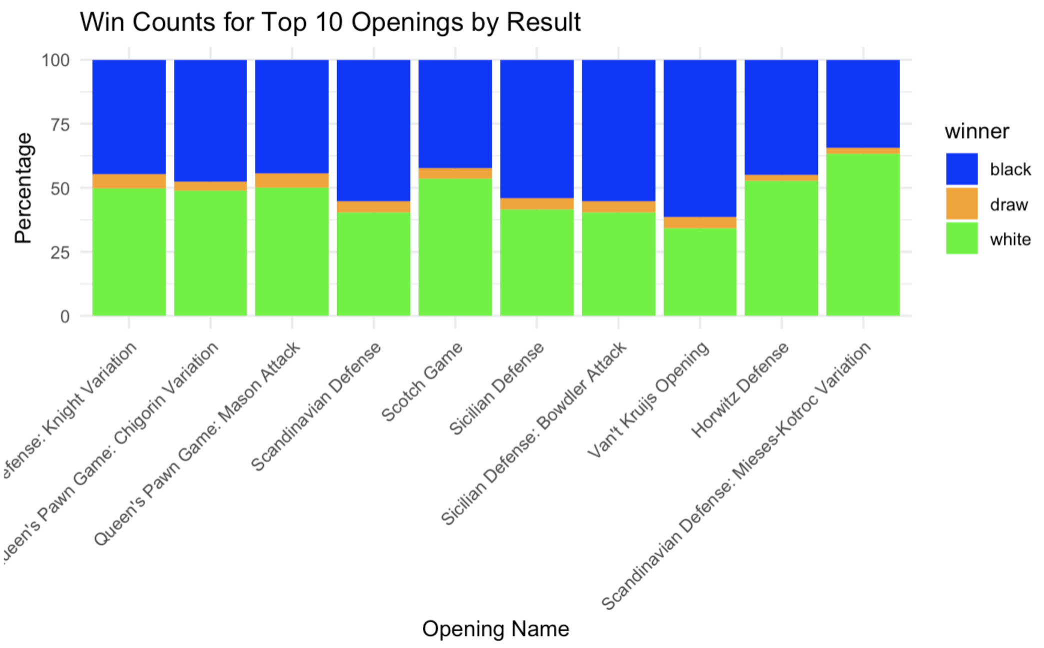
In this milestone, we designed and implemented four visualization components and an interactive Shiny app to explore patterns in chess gameplay data. Each visualization addresses specific questions related to openings, time controls, move frequencies, and player ratings. Below, we discuss each visualization in detail, including the choice of design, implementation, and critical evaluation.

1. Win Counts for Top 10 Openings by Result

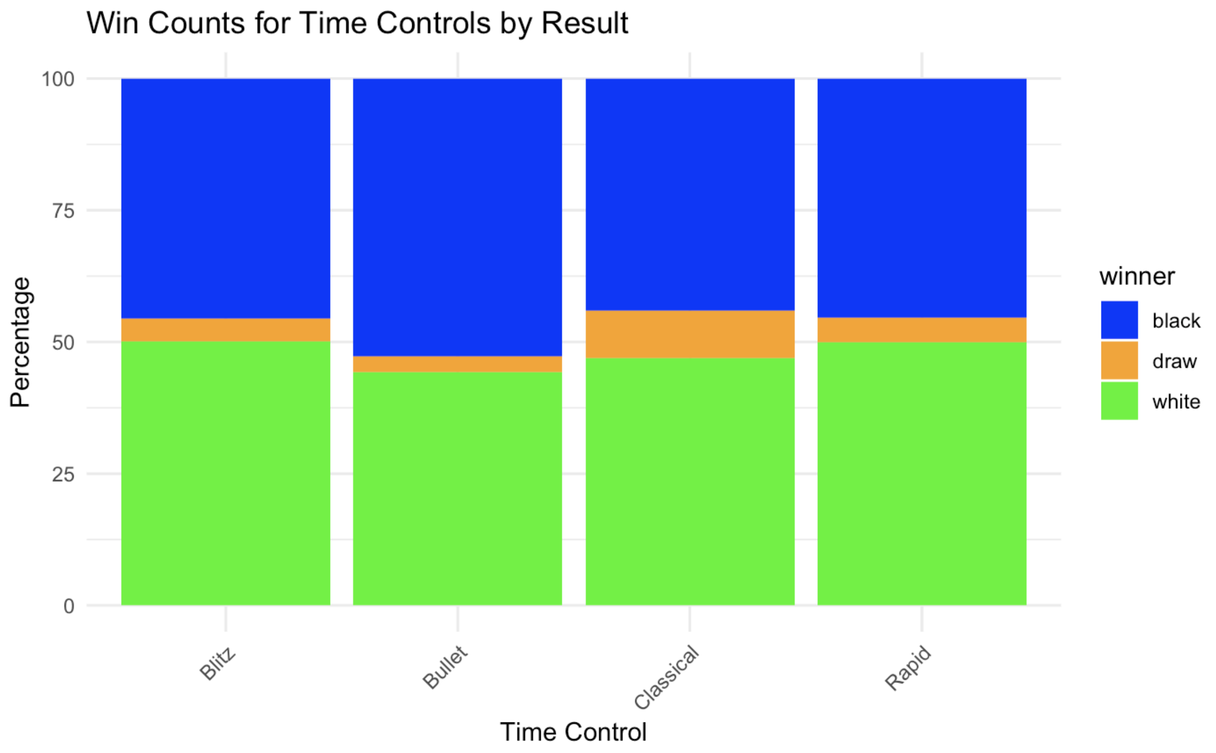


The purpose of this visualization is to analyze the performance of the most common chess openings and understand how often they lead to wins for white, black, or draws. We chose a stacked bar chart because it effectively shows the distribution of results for each opening. To ensure clarity, we limited the analysis to the top 10 most common openings, as including too many categories could make the chart cluttered. The trade-off here is that while stacked bar charts are excellent for showing proportions, they may obscure subtle differences in win rates between openings.

The visualization highlights that certain openings, such as the "Sicilian Defense," favor black, while others, like the "Italian Game," are more balanced. To implement this, we used dplyr to filter the dataset for the top 10 openings and calculate the percentage of wins for each result type. The ggplot2 package was used to create the stacked bar chart, with the reorder() function ensuring that openings are sorted by win percentage. The color scheme distinguishes between results (white, black, and draw), making it easy to interpret.

The stacked bar chart effectively communicates the distribution of results for each opening. However, it may become cluttered if more openings are included. Alternative designs, such as grouped bar charts or small multiples, could be explored to improve readability. Additionally, adding interactivity (e.g., tooltips) could help users explore specific openings in more detail.

2. Win Counts for Time Controls by Result

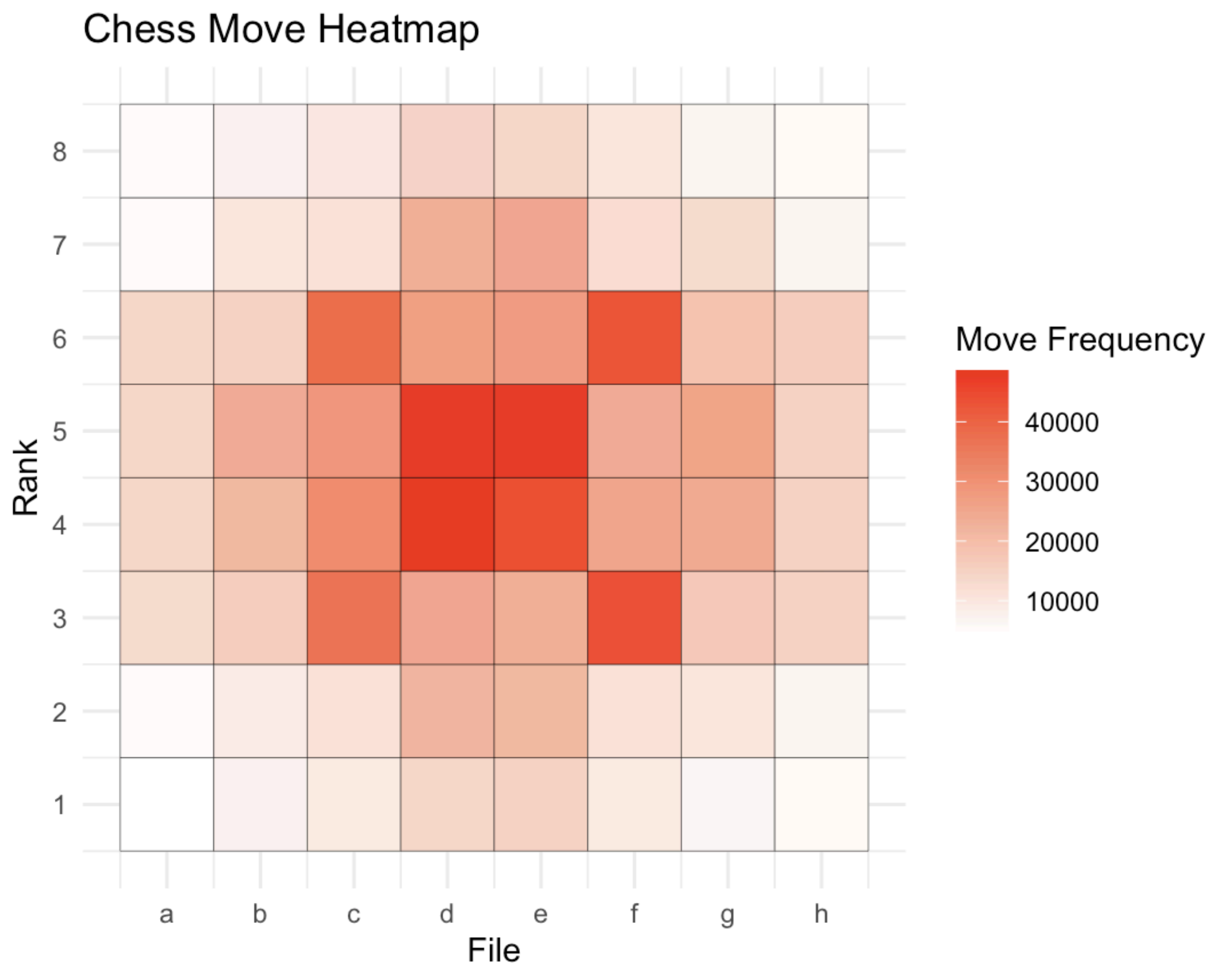


This visualization aims to examine how different time controls (e.g., Bullet, Blitz, Rapid, Classical) influence game outcomes. We chose a stacked bar chart to compare the percentage of wins for each result type across time controls. The design decision to use a stacked bar chart was motivated by its ability to show proportions clearly. However, a trade-off is that subtle differences in win rates between time controls may be obscured due to the stacked nature of the chart.

The visualization reveals that faster time controls, such as Bullet, tend to favor decisive outcomes, while slower controls, like Classical, have a higher percentage of draws. To implement this, we applied a custom function to categorize games into time controls based on the increment_code column. The ggplot2 package was used to create the stacked bar chart, with colors distinguishing between results. The x-axis labels were rotated for better readability.

The stacked bar chart effectively compares results across time controls but may not highlight subtle differences in win rates. Alternative designs, such as grouped bar charts or line charts, could be explored to better emphasize these differences. Additionally, adding interactivity (e.g., filtering by opening) could enhance the user experience.

3. Chess Move Heatmap

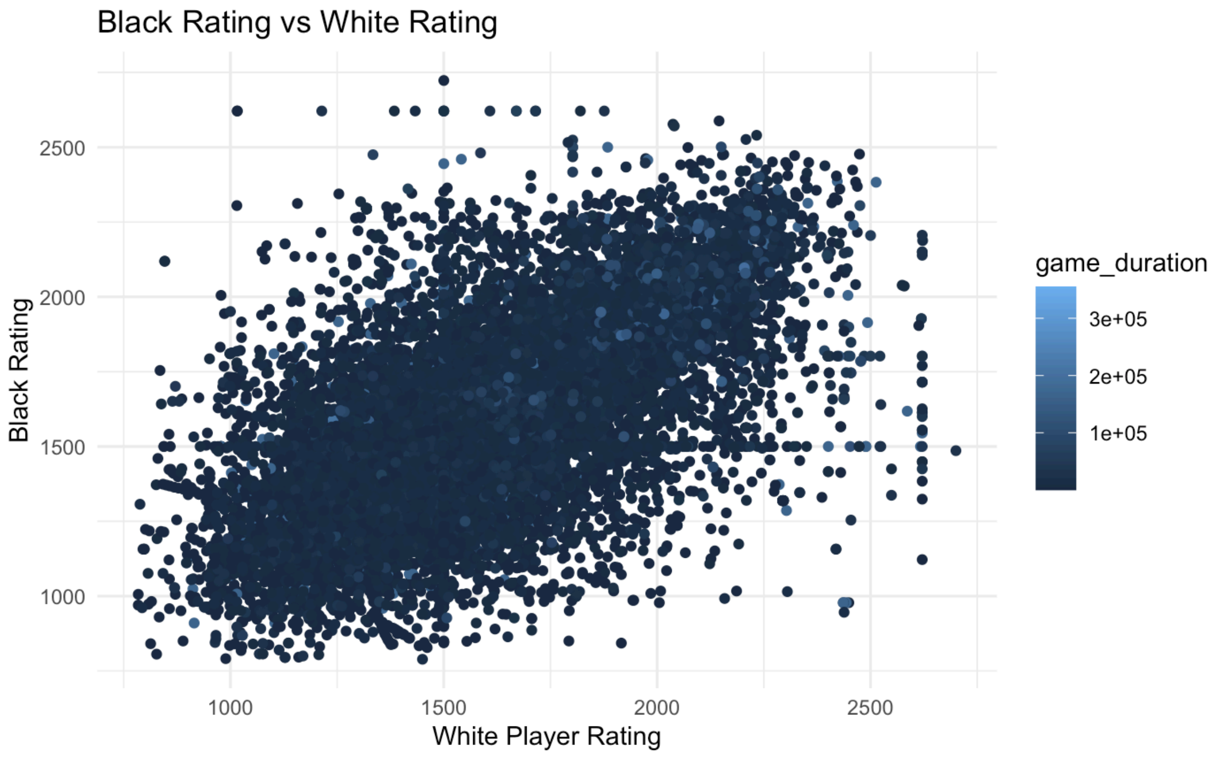


The purpose of this visualization is to identify hotspots of activity on the chessboard by visualizing the frequency of moves on each square. We chose a heatmap because it is well-suited for representing spatial patterns. The trade-off is that while heatmaps excel at showing spatial data, they do not capture temporal aspects of move sequences, such as the order in which squares are used.

The heatmap shows that central squares, such as e4 and d4, are the most frequently used, reflecting their strategic importance in chess. To implement this, we used stringr to extract square moves from the moves column and counted their occurrences. The ggplot2 package was used to create the heatmap, with a gradient color scale representing move frequencies. The coord_fixed() function ensured that the squares maintained their aspect ratio, making the heatmap visually accurate.

The heatmap effectively highlights spatial patterns in move frequencies but is limited in its ability to show temporal patterns. Adding interactivity, such as tooltips or animations, could help users explore move sequences over time. Additionally, filtering the data by game phase (e.g., opening, middlegame, endgame) could provide deeper insights.

4. Game Duration vs. Rating Difference

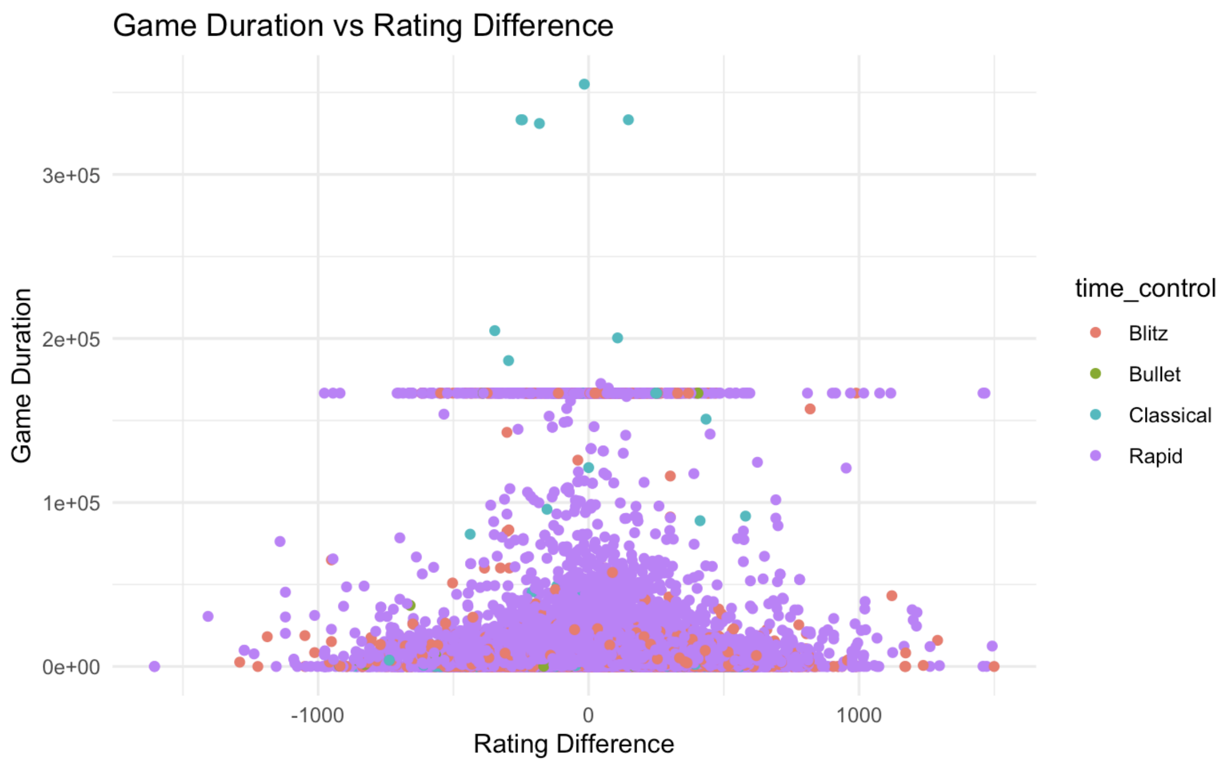


This visualization explores the relationship between game duration and the rating difference between players, stratified by time control. We chose a scatter plot because it effectively shows relationships between two continuous variables. The trade-off is that scatter plots can become cluttered with large datasets, so we filtered out extreme values to improve readability.

The scatter plot indicates that games with smaller rating differences tend to have longer durations, especially in Classical time controls. To implement this, we calculated game duration and rating difference, then used ggplot2 to create the scatter plot. Points were colored by time control to highlight patterns. The theme_minimal() function ensured a clean and modern design.

The scatter plot effectively shows the relationship between game duration and rating difference but can become cluttered with large datasets. Adding interactivity, such as zooming and filtering, could improve usability. Additionally, incorporating trend lines or regression analysis could provide deeper insights into the relationship.

5. Shiny App: Favorite Openings by Rating



The purpose of the Shiny app is to provide an interactive exploration of chess openings and their popularity across different rating groups. We chose a scatter plot of white ratings versus black ratings, colored by the top openings, and a table summarizing the top openings for each rating group. The trade-off is that while the app is intuitive, users unfamiliar with chess terminology may need additional context.

The app allows users to dynamically adjust the number of top openings displayed and view both a visualization and a table summarizing the results. The makePlot() function generates the scatter plot, while the makeTable() function creates the table. The app uses dplyr for data manipulation, ggplot2 for visualization, and Shiny for interactivity. The scale_color_viridis_d() function ensures an accessible color scheme.

The app provides an interactive and user-friendly way to explore chess openings but can become cluttered if too many openings are displayed. Adding filters (e.g., by time control or result) could enhance its analytical capabilities. Additionally, optimizing performance for larger datasets would improve the user experience.

The visualizations and Shiny app provide valuable insights into chess gameplay, addressing questions related to openings, time controls, move frequencies, and player ratings. While each visualization has its strengths, there are trade-offs in terms of readability, information density, and interactivity. For the final interface, we plan to refine the designs based on user feedback and add more advanced features, such as interactivity and statistical analysis.