Cricket Match Simulation

Laksh Budhrani

Abstract

This project simulates a 4-innings Test cricket match on a ball-by-ball basis using VBA in Excel. The simulation involves two teams, typically "India" and "Australia," determined by a virtual toss. The primary components include initializing teams, setting batting and bowling orders, generating individual ball outcomes, updating scorecards, and logging match results.

Introduction

My project is mainly divided into four parts: data collection for empirical probability distribution for all the batters and bowlers, the theory behind the modelling, the VBA code, and the analysis.

- **Data Collection**

I utilized Cricsheet, which contains ball-by-ball information for 845 Test matches, presented in JSON format. Using Python, I extracted the necessary data for my simulation. Specifically, I collected the number of 0's, 1's, 2's, ..., 6's, and outs that each player has recorded from the total number of balls they have played at the Sydney Cricket Ground (SCG). By dividing the number of balls that resulted in 0 runs by the total number of balls played at SCG, I obtained the probability distribution for a player's 0 run. This method was similarly applied for other runs to derive a comprehensive probability distribution for each player.

- **Batting and Bowling Lineup**

The batting lineup was predetermined based on the squad. For bowlers, I used my personal experience to assign their probability distribution for getting a chance to bowl.

- **Python Files**

The project submission includes the four Python files used for data extraction: data_extraction.py, info_extraction.py, overall_career_stats.py, and prob_distribution.py.

- **Subsequent Steps**

The descriptions of the other steps in my simulation are explained in the following sections of the report. This includes a detailed explanation of the theoretical framework behind the modelling, the implementation of the simulation in VBA, and the analysis of the results.

Model Description

At this point, it's crucial to understand the reason for having separate probability distributions for batsmen and bowlers. Both distributions have distinct meanings. For a batsman, the distribution represents the probability of scoring a particular number of runs on a given ball. Conversely, for a bowler, it represents the probability of conceding that particular score on a given ball.

Initially, my approach for the model involved taking the probability distributions for both the batter and the bowler for each ball, multiplying the corresponding terms, and then normalizing them by dividing each product by the sum of all products. Using the `discreteEmp` function from the BCNN.xla file, we could then determine the specific outcome for that ball.

To illustrate this approach, consider a bowler with a high probability of taking wickets who interacts with two batsmen: one with a low probability of getting out and the other with a high probability. Multiplying the bowler's wicket-taking probability with the first batsman's low out probability results in a lower normalized value compared to the second batsman with a high out probability. This method accounts for the performance of both the batsman and the bowler.

However, this approach has a practical implementation issue as it makes the simulation excessively long. The probability of a batsman getting out is relatively low, and the exit condition for our simulation is when all batsmen are out. Therefore, we introduced a solution by adding a SoftMax temperature to adjust the probabilities. This increases the probabilities of other outcomes by decreasing the highest probability, stabilizing the system effectively.

The stabilizing effect of this system increases the smaller probabilities proportionally, meaning that aside from the highest probability, the smaller probabilities will rise in the same order. Consequently, since bowlers have higher chances of taking wickets, they will achieve wickets earlier in the simulation. Our system stabilizes well at a temperature value of 0.142.

For a concise explanation of this concept, you can refer to this informative video:

https://youtube.com/shorts/XsLK3tPy9SI?si=fAAMroHDslHykUWL

https://medium.com/@harshit158/softmax-temperature-5492e4007f71

Using this model, we represent the entire simulation ball by ball, updating the states continuously.

Simulation Description

Our cricket simulation operates similarly to a state machine, meaning that each iteration of the simulation only requires a few key states to proceed. These essential states in our system include the number of balls, striker, non-striker, bowler, runs (at a particular ball), and innings. Let's delve into each of these components:

Key States in the Simulation

1. **Number of Balls**:
   - This state keeps track of how many balls have been bowled in the current innings. It helps in determining when an over is completed and when to switch the bowler.
2. **Striker**:
   - The batsman currently facing the bowler. This state is crucial as it determines which batter's probability distribution is used for the next ball outcome.
3. **Non-Striker**:
   - The batsman at the non-striker's end. This state is important for handling runs that involve changing ends, such as singles or threes, which might swap the striker and non-striker.
4. **Bowler**:
   - The current bowler delivering the ball. This state helps in determining the probability distribution for the runs or wickets, as different bowlers have different abilities and performance metrics.
5. **Runs (at a particular ball)**:
   - The outcome of each ball, indicating how many runs were scored or if a wicket was taken. This state updates the score and the position of the batsmen, affecting subsequent states.
6. **Innings**:
   - The current phase of the game, which could be the first, second, third, or fourth innings. This state helps in managing the transitions between the different innings and maintaining the overall structure of the Test match.

These states together help maintain the flow of the game and ensure that each ball's outcome is accurately calculated and reflected in the simulation.

Our VBA code is designed as a collection of independent functions, each serving a specific purpose in the simulation. These functions work together to create a comprehensive and realistic cricket match simulation. The MainSimulation function ties all these individual functions together, coordinating their interactions to execute the simulation seamlessly.

Key Functions Explained

1. **TossSimulation**

This function simulates the toss to decide which team will bat first and which will bowl. It sets up the initial conditions for the simulation by determining the batting and bowling order based on a random toss outcome.

2. **get[batter/bowler][prob/cumprob/values]**

These functions retrieve various probability-related data for batters and bowlers:

- **getBatterProb**: Gets the probability distribution for a particular batter's scores.
- **getBatterCumProb**: Retrieves the cumulative probability distribution for a batter.
- **getBatterValues**: Retrieves the possible score values for a batter.
- **getBowlerProb**: Gets the probability distribution for a particular bowler's scores conceded.
- **getBowlerCumProb**: Retrieves the cumulative probability distribution for a bowler.
- **getBowlerValues**: Retrieves the possible score values for a bowler.

3. **initializeTeams**

This function sets up the teams based on the toss result. It determines which team will bat and which will bowl initially and prepares the teams for the start of the match.

4. **initializeBattingOrder**

Sets up the batting order for the team that is going to bat. This function ensures that the batting order is initialized correctly based on the team composition.

5. **selectBowler**

This function selects the bowler for the current over. It uses the probability distributions for bowlers to determine which bowler will bowl next, ensuring a realistic representation of bowling rotations.

6. **discreteEmpArray**

Used to calculate the outcome of a ball based on discrete empirical probability distributions. It selects a particular value (like runs scored or wickets) based on the probabilities provided, simulating the randomness inherent in cricket.

7. **GenerateBatterScores**

This function is crucial as it generates the score outcome for each ball faced by a batter. It takes into account the batter's and bowler's probability distributions and uses the discreteEmpArray function to determine the result. For example:

- **Inputs**: Batter's name, bowler's name.
- **Process**: Fetches the probability distributions for both the batter and the bowler, computes the combined probabilities by considering SoftMax temperature, and normalizes them.
- **Output**: A score value for the ball, indicating how many runs were scored or if the batter got out.

8. **Results**

Aggregates the scores from all innings and determines the final result of the match. It computes the total runs for both teams and returns the winning team and the margin of victory or declares a draw/tie if applicable.

9. **UpdateGameState**

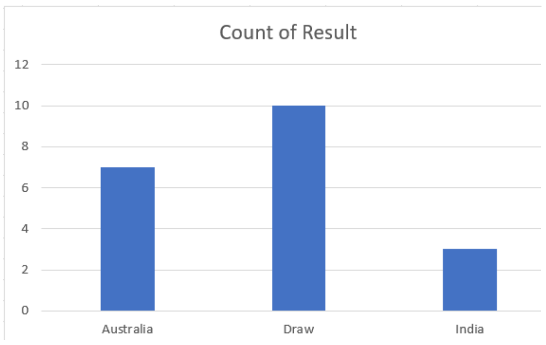
Updates the state of the game after each ball. It modifies the states like the number of balls bowled, the current striker, non-striker, bowler, and the runs scored. This ensures that the game state is always current and accurate for the next iteration.

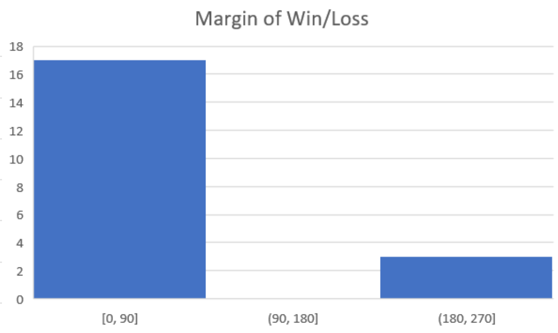
10. **MainSimulation**

The MainSimulation function is the heart of the simulation. It coordinates all the other functions to simulate the match ball by ball. Here's a more detailed explanation:

- **Initialize Teams**: Calls initializeTeams to set up the batting and bowling order.
- **Loop Through Innings**: Manages the flow of the game through all four innings, handling team switches.
- **Ball-by-Ball Simulation**: For each ball, it calls GenerateBatterScores to determine the outcome and UpdateGameState to update the game state accordingly.
- **Track Scores and Results**: Keeps track of scores and uses the Results function to determine the match result at the end of the game.

Analysis





Match Outcomes:

1. **Australia's Dominance**: Australia won 7 out of 20 matches, indicating a stronger performance compared to India, which won only 3 matches. This suggests that in the simulations, Australia generally had a higher probability of winning.
2. **High Draw Frequency**: With 10 out of 20 matches ending in a draw, it indicates that the simulated matches were often closely contested, and neither team was able to secure a decisive victory. This could reflect balanced team strengths or conservative gameplay scenarios.

Margin of Wins:

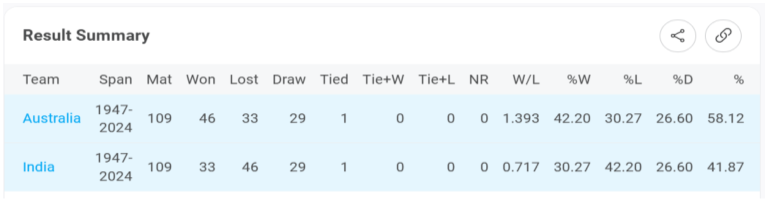
1. **Close Margins**: 17 matches had winning margins in the range of 0-90 runs. This suggests that most victories were closely contested, with small margins indicating competitive matches where both teams performed similarly.
2. **Larger Margins**: Only 3 matches had winning margins in the range of 180-270 runs, indicating less frequent but more dominant performances by the winning team in these simulations. These larger margins suggest that occasionally one team significantly outperformed the other.

Author's View

Comparing the simulation results with real statistics of India versus Australia matches, the outcomes are quite satisfying. This match and the corresponding statistics are specific to the Sydney Cricket Ground (SCG), which is Australia's home ground. Historically, players' performance tends to be better when they play at their home grounds. This trend is reflected in the simulation results, where Australia won more matches.

Additionally, the analysis highlights that India versus Australia matches are closely contested. As a cricket fan, I know that India and Australia have been among the top teams in cricket for a long time, leading to highly competitive and closely fought matches. The high number of draws and the narrow margins of victory in the simulation align well with the real-world rivalry between these two cricketing giants, providing a realistic and insightful representation of their matches at the SCG.

This analysis not only validates the simulation model but also underscores the intense and competitive nature of India versus Australia cricket matches, reinforcing the expectation that these encounters are always thrilling and unpredictable.



Result Summary

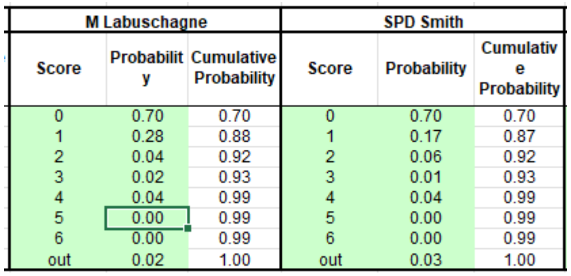
Team	Span	Mat	Won	Lost	Draw	Tied	Tie+W	Tie+L	NR	W/L	%W	%L	%D	%
Australia	1947-2024	109	46	33	29	1	0	0	0	1.393	42.20	30.27	26.60	58.12
India	1947-2024	109	33	46	29	1	0	0	0	0.717	30.27	42.20	26.60	41.87

Hypothetical Scenario 1: Impact of Steve Smith and Marnus Labuschagne's Recent Form

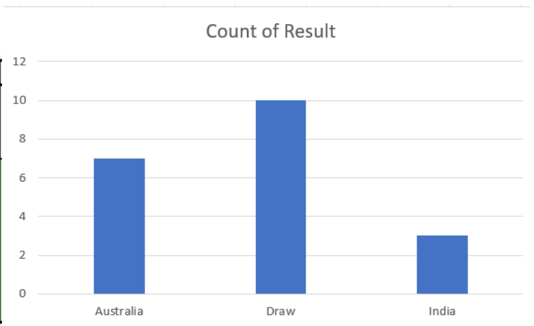
Steve Smith and Marnus Labuschagne are renowned as some of the best middle-order batsmen for Australia, and their impressive statistics reflect their exceptional skills. However, over the past 1-2 years, their performance has not been at the same level as it once was. This recent dip in form could significantly influence match outcomes.

Scenario: To explore this, we will adjust the probability distributions of Steve Smith and Marnus Labuschagne to reflect their recent performance dip. By doing so, we can assess whether India's chances of winning the series increase under these altered conditions.

Purpose: The goal is to determine the extent to which the recent decline in form of these key Australian batsmen affects the overall match dynamics and whether it tilts the balance in favor of India.



M Labuschagne			SPD Smith		
Score	Probability	Cumulative Probability	Score	Probability	Cumulative Probability
0	0.70	0.70	0	0.70	0.70
1	0.28	0.88	1	0.17	0.87
2	0.04	0.92	2	0.06	0.92
3	0.02	0.93	3	0.01	0.93
4	0.04	0.99	4	0.04	0.99
5	0.00	0.99	5	0.00	0.99
6	0.00	0.99	6	0.00	0.99
out	0.02	1.00	out	0.03	1.00



Results of the Hypothetical Scenario

Surprisingly, the change in form for Steve Smith and Marnus Labuschagne did not significantly affect Australia's match results. Despite adjusting their probability distributions to reflect their recent performance dip, the overall outcomes remained largely the same.

It is important to note that even small changes in the probability distribution can have a larger impact due to the application of softmax temperature. A slight increase in the probability of getting out can theoretically lead to a more significant effect on match dynamics. However, after running the experiment three times, the distribution of match outcomes remained consistent. Although the specific results varied for each trial due to the inherent randomness of the simulation, the overall distribution pattern was nearly identical.

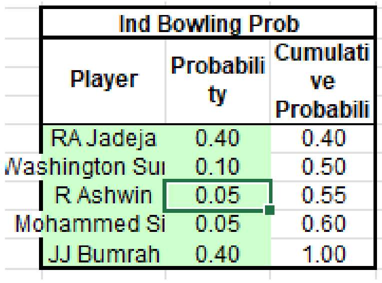
This indicates that Australia's team performance is resilient and not heavily dependent on the form of individual players like Smith and Labuschagne, at least within the scope of our simulation. It suggests that other factors and players contribute significantly to maintaining Australia's competitive edge in these simulations.

Hypothetical Scenario 2: Increasing Reliance on Jadeja and Bumrah

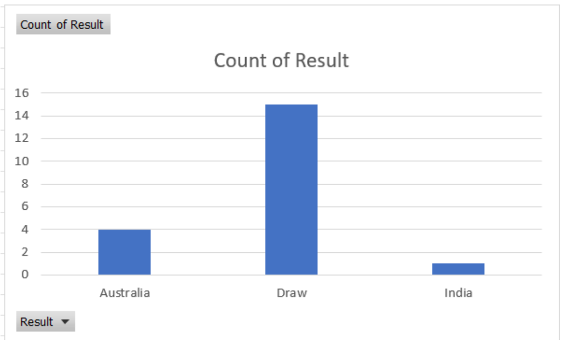
In Test matches, there is no limit to the number of overs a bowler can bowl. This flexibility allows teams to rely more heavily on their key bowlers in crucial situations. For this hypothetical scenario, we will explore what happens if India increases its reliance on two of its top bowlers, Ravindra Jadeja and Jasprit Bumrah. Specifically, we will adjust their probability distribution for getting overs and assess whether this increases India's chances of winning.

Scenario: Modify the probability distributions to reflect a higher likelihood of Jadeja and Bumrah bowling more overs compared to other bowlers in the team.

Purpose: The goal is to determine if increasing the bowling workload of Jadeja and Bumrah, who are known for their effectiveness, can enhance India's chances of winning the series.



Ind Bowling Prob		
Player	Probability	Cumulative Probabili
RA Jadeja	0.40	0.40
Washington Sui	0.10	0.50
R Ashwin	0.05	0.55
Mohammed Si	0.05	0.60
JJ Bumrah	0.40	1.00



Results of the Hypothetical Scenario
Yes, the changes did make a difference. With the increased reliance on Jadeja and Bumrah, the simulation results showed 15 draws, 4 losses, and 1 win for India out of 20 matches.

This indicates that their increased bowling workload helped decrease the loss percentage, although it also slightly reduced the winning percentage. Essentially, relying more on Jadeja and Bumrah led to more matches ending in a draw, which is not a negative outcome as it reduced the overall number of losses.

In conclusion, while this strategy may not significantly increase the winning percentage, it does enhance the team's ability to avoid losses. This makes relying more on Jadeja and Bumrah a potentially viable tactic for achieving more stable and consistent match outcomes.

Future Work/Summary

There are several aspects that can be considered for further enhancing the simulation model:

1. **Player Comparison**: The current model does not differentiate between players who have the same probability distribution but different skill levels. In reality, one player could be much better than another despite similar statistical probabilities. Future work could incorporate a qualitative assessment or ranking system to better reflect player abilities.

2. **Extras and Wickets**: The model could be expanded to include factors like extras (no balls, wides, byes) and different types of wickets (bowled, caught, LBW). This would provide a more comprehensive simulation of real match scenarios.

3. **External Factors**: Probability distributions could be adjusted based on external factors such as weather conditions, pitch type, dew, and time of day (morning/evening). These elements can significantly influence player performance and match outcomes, adding another layer of realism to the simulation.

4. **Dynamic Adjustments**: Incorporating dynamic adjustments to player probabilities based on in-match events (e.g., a player gaining confidence after scoring a century or a bowler finding rhythm) could make the simulation more responsive and lifelike.

5. **Team Strategies**: Exploring different team strategies, such as aggressive versus defensive play styles or adapting to opponent weaknesses, could offer insights into optimal approaches for various match conditions.

By addressing these factors, future enhancements to the simulation model can provide deeper insights and a more accurate representation of cricket matches.