OPTIMIZING BATTING LINEUP FOR INDIAN CRICKET PLAYERS BASED ON MATCH LOCATION

A Data-Driven Approach to Maximize Player Performance in India and England

Shahzaib Rahat

Capstone Project

Summary

This project focuses on optimizing the batting lineup for cricket players based on their performance across different match locations using mathematical modeling and simulations. By applying Kernel Density Estimation (KDE), I simulated each player's performance in terms of runs, balls, and run rate to capture realistic data trends. Constraints such as minimum run rates and location-specific batting position eligibility were incorporated to ensure the model adhered to real-world cricket conditions. Using the Simplex method, the objective was to maximize total expected runs while ensuring that each player was assigned to a unique position in the lineup, respecting both performance data and location-based restrictions.

The optimal solution was determined by solving the linear programming problem, which identified the best batting positions for each player based on their expected performance at different locations like India and England. Sensitivity analysis was conducted to evaluate how relaxing certain constraints (such as player position flexibility) could impact the total score. This approach provides a systematic and data-driven way to select the best possible batting lineup for a cricket match, ensuring both realism and performance optimization.

1. Problem Definition

Understanding Key Cricket Concepts

- **Over**: An *over* in cricket consists of 6 consecutive legal deliveries (balls) bowled by the same bowler. In One Day Internationals (ODIs), each team plays a maximum of 50 overs.
- **Run Rate**: Run rate is the average number of runs scored per ball. It is calculated as:

  $$\text{Run Rate} = \text{Total Runs Scored} / \text{Total Balls Faced}$$

- **Batting Lineup**: The batting lineup determines the sequence in which players go out to bat. The order can influence the total score, depending on a player's performance at a particular position.
- **Key Factors Influencing Match Outcomes**:
  - **Location of the Match**: Conditions like pitch type, weather, and crowd support vary by location and can affect player performance.
  - **Batting Lineup Strategy**: Teams strategically decide batting positions based on form, matchups, and past performance in similar conditions.
  - **Opponent Team and Bowlers**: The opposing team's bowling strength influences how well a batsman performs.
  - *(Note: For this project, only Location and Batting Lineup are considered as variables.)*

Problem Statement

In this project, I aim to determine the **optimal batting lineup based on match location** for three top Indian batsmen:

- **KL Rahul**
- **Virat Kohli**
- **Rishabh Pant**

I consider two distinct match locations:

- **India**
- **England**

The goal is to use past performance data (since 2015) to **simulate match outcomes** and identify the batting order that **maximizes the total expected score** of these three players for a given location.

To ensure realism and fairness in the lineup selection, I introduce the following key constraint:

- Each selected player's **expected run rate must be greater than 0.6** in the assigned batting position for that location.

2. Formulation

Dataset Description

The dataset used in this project contains past performance statistics of three Indian cricket players — **KL Rahul, Virat Kohli, and Rishabh Pant** — from matches played in **India and England**. The dataset was sourced from Cricmetric.

Key Variables

In this project, the key variables are:

- **Player_Name**: The batsman under consideration.
- **Location**: Stadium location (India or England).
- **Batting_Position**: The position in the batting order (e.g., 1st, 2nd...).
- **Run** and **Balls**: Used to evaluate batting performance.
- **Run Rate (Av_R.R)**: Essential because matches are limited to a fixed number of overs.

Since **One Day Internationals (ODIs)** are limited to 50 overs per team (i.e., 300 balls), I must not only **maximize total score** but also ensure **good scoring efficiency (run rate)**. That's why both **score** and **balls faced** are considered critical.

Subgrouping Strategy for Simulation

To estimate expected performance, the dataset is grouped into **subgroups based on**:

- Player_Name
- Location
- Batting_Position

Each subgroup contains records with the **same player, location, and batting position**.

The goal of the simulation is to estimate the:

- **Expected total runs**
- **Expected run rate**

for each player at each eligible batting position in both locations.

Simulation Constraints

In the simulation, both **Runs** and **Balls** are simulated jointly to preserve their relationship.
To ensure realism, these constraints are enforced:

- **Runs cannot be negative**

- **Each player must play at least one ball**
- **Maximum runs scored cannot exceed 6 × Balls faced**

To simulate 1000 realistic combinations of scores and balls while maintaining their relationship, I use **Kernel Density Estimation (KDE)**. This method is explained in the next part of the report and is chosen for its ability to generate realistic, continuous data based on original performance.

Simplex Method and Constraints

Once expected scores and run rates are generated for each player-batting position-location combination, the next step is to **determine the optimal batting lineup** using the **Simplex Method**.

The following **constraints** are enforced in this linear programming step:

- **One player can have only one batting position**
- **Each batting position can be taken by only one player**
- **Run rate must be ≥ 0.6** for a position to be considered valid
- **Player-specific position eligibility**, e.g.:
    - *KL Rahul can only play positions 2, 4, and 5 in India*

Optimization Objective

The objective function is to:

- **Maximize the total expected score of the 3 selected players**,
- Based on the stadium location and valid constraints.
-

3. Data

Data Collection

The player performance data for this project was collected from the website: https://www.cricmetric.com/index.py.
For each of the three selected Indian players — **KL Rahul, Virat Kohli, and Rishabh Pant** — the data was queried using the following parameters:

- **Batting Format:** One Day Internationals (ODIs)
- **Venue Countries:** India and England
- **Batting Positions:** 1 to 7
- **Time Period:** From 2015 onwards
- **Grouping Option on Website:** *Group by Match*

Each query returned match-wise batting statistics for the selected player under the specified conditions. After downloading the results as Excel files, I merged all player-specific datasets into a **main combined dataset**.

Data Structure

The final dataset includes the following columns:

Column Name	Description
Player_Name:	Name of the batsman
Match:	Unique match identifier
Run:	Runs scored in that match
Balls:	Balls faced in that match
Batting_Position:	The player's batting position in that match
Location:	Venue country (India or England)
Av_R.R:	Average Run Rate (Run ÷ Balls) for that match

Data Grouping

The dataset is first grouped by the following three columns:

- **Player_Name**
- **Location**
- **Batting_Position**

I grouped the dataset based on these attributes because a player's performance can vary significantly depending on:

- **Who the player is** (individual batting style and consistency)
- **Where they are playing** (Indian vs. English pitch conditions)
- **What batting position they are assigned** (e.g., opening vs. middle-order)

These three factors most strongly influence batting outcomes and help create more accurate simulations.

After forming these subgroups, I generate **1000 simulations for each group** using the combined probability distribution of **Runs** and **Balls** faced. This enables the modeling of realistic match performances under various conditions.

Grouped Data Overview

After grouping the main dataset by **Player Name**, **Location**, and **Batting Position**, I identified the following **12 unique subgroups**:

- (`KL Rahul`, `England`, 1)
- (`KL Rahul`, `England`, 4)
- (`KL Rahul`, `India`, 2)
- (`KL Rahul`, `India`, 4)
- (`KL Rahul`, `India`, 5)
- (`Rishabh Pant`, `England`, 4)
- (`Rishabh Pant`, `India`, 4)
- (`Rishabh Pant`, `India`, 5)
- (`Virat Kohli`, `England`, 3)
- (`Virat Kohli`, `England`, 4)
- (`Virat Kohli`, `India`, 3)
- (`Virat Kohli`, `India`, 4)

Note: Each group in the dataset has a **unique combination** of **Player Name**, **Location**, and **Batting Position**.
This structure ensures that each subgroup accurately represents a specific match context and is ready for simulation using **joint probability distributions** of **Runs** and **Balls**.

4. Analysis

Project Objective: Optimizing Batting Lineups Based on Location and Position

The goal of this project is to determine the **optimal batting lineup** for three selected Indian men's cricket players — **KL Rahul, Virat Kohli, and Rishabh Pant** — by analyzing their **past performance data**. This is achieved through **simulation techniques** that estimate expected performance under different match conditions, followed by **optimization** to select the most effective batting order.

Key Dataset Highlights

- **Players Considered:** KL Rahul, Virat Kohli, and Rishabh Pant.
- **Time Period:** From **March 2015 to the present**.
- **Locations Analyzed:** Matches played in **India** and **England** only.
- **Batting Positions:** Ranges from positions **1 to 7**, based on each player's historical appearances.
- **Metrics Analyzed:**
    - **Runs** scored in each match
    - **Balls** faced
    - **Derived Run Rate** (Runs ÷ Balls), to measure scoring efficiency

Simulation Strategy

To estimate realistic future performances, I simulate each player's performance in each subgroup (defined by player, location, and batting position) using **1,000 runs of simulation**.

Simulation Technique:

- **Method Used: Kernel Density Estimation (KDE)**
- **Reason:** This method preserves the natural **correlation between Runs and Balls faced**, which is important because scoring more runs generally means facing more balls — but not always in a linear fashion.

Simulation Goals:

From each simulation run, I compute:

- **Expected Runs**
- **Expected Run Rate**

These metrics are later used in a **linear programming model** to optimize the batting order in a way that:

- Maximizes **total expected runs**
- Prevents any **player-position conflicts**
- Ensures **minimum run rate criteria** is met

Before Applying the Simulation

In this project, I simulate two interdependent variables: **Runs** and **Balls Faced**. These are correlated — a high score typically involves facing more balls. However, the relationship is **non-linear** and **context-specific** (varies by player, position, and location).

To maintain their relationship during simulation, I use **Kernel Density Estimation (KDE)**, a concept I learned in **Data 613**.

What is Kernel Density Estimation (KDE), and Why Use It?

KDE is a **non-parametric** technique used to estimate the probability distribution of a variable. It works by placing a smooth curve (e.g., a Gaussian) on each data point and combining them to form a continuous estimate of the underlying distribution.

Simulation Constraints

To ensure realism, the following constraints are applied during simulation:

- **Runs ≥ 0**
- **Balls faced ≥ 1**

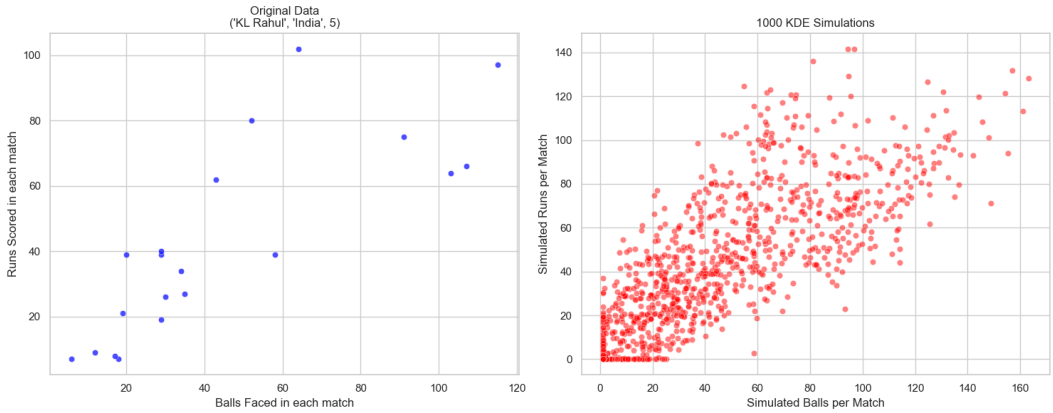
- **Runs ≤ 6 × Balls** (since a maximum of 6 runs can be scored per ball)

Derived Run Rate (sim_av_rr)

During each simulation, I compute a derived variable:

- `sim_av_rr = Runs ÷ Balls`

While this is not the official team run rate used in cricket statistics, it serves a **similar purpose** — measuring **scoring efficiency** at the individual level, which is critical in **limited-overs** formats like ODIs.



Comparison between Original Data Points and Simulation Data Points

- The KDE simulation closely replicates the original data's behavior. The mean values of Runs and Balls are nearly identical between the original and simulated data, and the correlation between them is also preserved (0.81 vs 0.82).
- Additionally, the scatter plots show the same upward trend and similar distribution of data points, **confirming that KDE successfully captures the joint relationship between Runs and Balls.**

Another Constraint for Simulation: `sim_av_rr <= 6`

Another constraint I applied is `sim_av_rr <= 6`. This was done for two main reasons:

- **Realistic Simulations**: In cricket, it's impossible for a player to have an average run rate greater than 6 runs per ball. This constraint ensures the simulated values reflect realistic player performance.
- **Outlier Removal**: By removing data points where `sim_av_rr > 6`, I prevent extreme outliers that do not represent typical cricket match scenarios.
- **Preserving Distribution**: While removing unrealistic outliers, this constraint maintains the overall shape of the data's probability distribution, ensuring that the simulation remains consistent with the original data's density.

This approach ensures that the simulations are both realistic and statistically valid.

Next Steps: Expected Values for Optimization

- After filtering the data, I have calculated the expected run, expected balls, and the expected run rate (sim_av_rr) for each group. These expected values will represent the average performance of players across **different locations and batting positions.**
- These expected values will be the key variables used in the **simplex method** to optimize the batting lineup. **The goal is to maximize the total runs** while respecting the constraint that each player can only occupy one position in the lineup.

Simulation Summary DataFrame:

`simulation_summary_df`

	Player_Name	Location	Batting_Position	Expected_Run	Expected_RR
0	KL Rahul	England	1	50.257910	0.611782
1	KL Rahul	England	4	12.824742	0.736363
2	KL Rahul	India	2	49.111232	0.689163
3	KL Rahul	India	4	59.463290	1.006248
4	KL Rahul	India	5	46.253782	1.024319
5	Rishabh Pant	England	4	46.163623	0.811417
6	Rishabh Pant	India	4	30.485436	0.938933
7	Rishabh Pant	India	5	45.945879	1.433688
8	Virat Kohli	England	3	51.885739	0.833646
9	Virat Kohli	England	4	49.089381	0.710002
10	Virat Kohli	India	3	59.005802	0.887102
11	Virat Kohli	India	4	62.900147	0.829727

The `simulation_summary_df` showcases the **expected performance metrics** (i.e., Expected Runs and Expected Run Rate) for each player across various **batting positions** and **locations**. It reflects the possible batting lineup configurations for each player based on the location of the match.

For example:

- **Virat Kohli**: If the match is held in **India**, he can take **Batting Positions 3 and 4** based on his past performance in those positions, as shown in the data. His expected runs and run rates for these positions are included in the table.
- **Rishabh Pant**: For a match in **England**, he can only take **Batting Position 4**. His expected runs and run rate for this position are provided accordingly.

The DataFrame thus helps in identifying the feasible batting positions each player can take based on the location of the match, making it useful for optimizing the batting lineup.

Optimization Problem for Batting Order

Objective Function

The goal is to maximize the total expected runs, denoted as Z. To do this, sum the product of the expected runs and the decision variable for each player and position.

- For each player ii and position jj, multiply the **Expected Runs** for that player at that position by the decision variable xijx_{ij}, where xijx_{ij} is 1 if the player bats at that position, and 0 if they do not.
- Add up the products of these values across all players and positions.

Thus, the objective is to maximize the total expected runs, which can be described as:

"Maximize the total expected runs by summing the expected runs for each player at each position, multiplied by the corresponding decision variable (1 if the player is assigned to that position, and 0 if they are not)."

Constraints for Simplex Method

1. **Each player bats once:**
    - For each player ii, they can only be assigned to one batting position. The sum of the decision variables across all positions for player ii should equal 1.
    - This ensures that each player will only occupy one batting position.
2. **Each position has one player:**
    - For each position jj, only one player can bat in that position. The sum of the decision variables across all players for position jj should equal 1.
    - This ensures that each batting position is filled by exactly one player.

Location Restrictions (India example):

1. **KL Rahul**: He can only be assigned to **Batting Position 2, 4, or 5**. The constraint ensures that he occupies one of these positions, and the sum of all possible positions he can take (Position 2, Position 4, and Position 5) must equal 1. This means that exactly one of these positions will be selected for KL Rahul.
2. **Rishabh Pant**: He can only be assigned to **Batting Position 4 or 5**. The constraint ensures that he occupies one of these two positions, and the sum of these possibilities must equal 1. Only one of these positions will be selected for Rishabh Pant.
3. **Virat Kohli**: He can only be assigned to **Batting Position 3 or 4**. The constraint ensures that he occupies one of these two positions, and the sum of these possibilities must equal 1. One of these positions will be selected for Virat Kohli.

Binary Constraints:

This ensures that each player can only be assigned to one position at a time, and no position can be assigned to more than one player.

5. Solution

After simulating the performance of each player using Kernel Density Estimation (KDE) and applying all the defined constraints, the optimal batting lineup for the three Indian batsmen was determined for each stadium location.

Stadium Location: India

- **KL Rahul → Batting Position 4**
- **Rishabh Pant → Batting Position 5**
- **Virat Kohli → Batting Position 3**

Stadium Location: England

- **KL Rahul → Batting Position 1**
- **Rishabh Pant → Batting Position 4**
- **Virat Kohli → Batting Position 3**

These lineups maximize the total expected score while meeting the required constraints, including a minimum run rate of 0.6 and eligibility of each player for specific batting positions.

Sensitivity Report: Variables

The table below shows the decision variables — in this case, the assignment of players to batting positions.

Variable	Value	Reduced Cost
x_KL Rahul_1	1.00	0.0000
x_KL Rahul_4	0.00	0.0000
x_Rishabh Pant_4	1.00	0.0000
x_Virat Kohli_3	1.00	0.0000
x_Virat Kohli_4	0.00	0.0000

Interpretation:

- **Value 1.00**: This means the player-position pair was selected in the optimal solution.
  - KL Rahul is assigned to position 1.
  - Rishabh Pant is assigned to position 4.
  - Virat Kohli is assigned to position 3.
- **Value 0.00**: The variable was not selected.
- **Reduced Cost = 0.0000** for all variables, indicating optimality for those selected.

Sensitivity Report: Constraints

The table below shows the constraints, such as player assignments, position assignments, and run-rate constraints.

Constraint	Shadow Price	Slack
player_KL Rahul	12.8247	0.0000
player_Rishabh Pant	46.1636	0.0000

Constraint	Shadow Price	Slack
player_Virat Kohli	49.0894	0.0000
pos_1	37.4332	0.0000
pos_4	0.0000	0.0000
pos_3	2.7964	0.0000

Interpretation:

- **Slack = 0.0000**: All constraints are tight, meaning they are exactly met.
- **Shadow Price**: This tells how much the objective function (total score) would improve if a constraint is relaxed by 1 unit.
    - Example: If KL Rahul could be assigned more than one position, the score could improve by 12.82 runs.
    - If another player could be assigned to position 1, the score could increase by 37.43 runs.

Key Takeaways:

- The optimal batting lineup, based on the constraints and run-rate threshold, is:
    - **KL Rahul → Position 1**
    - **Rishabh Pant → Position 4**
    - **Virat Kohli → Position 3**
- All constraints are fully utilized, meaning the solution is efficient.
- Shadow prices provide insight into the impact of relaxing certain constraints on the total score.