VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

A1b: ANALYSIS OF IPL DATA USING PYTHON AND R

JESIN KANDATHY JOY
V01110163

Date of Submission: 18/06/2024

CONTENTS

INTRODUCTION

This report presents an analysis of IPL (Indian Premier League) cricket data using R. The dataset includes ball-by-ball details and salary information for players up to the 2024 season. The analysis focuses on identifying top performers, fitting distributions to player performances, investigating the relationship between player performance and salary, and conducting statistical tests.

The report begins by setting the working directory and loading necessary libraries. Data cleaning steps include renaming columns, converting salary values to numeric format, and ensuring consistent formatting of player names. The analysis then aggregates data to calculate total runs and wickets for each player per match and identifies top performers in each season.

The next section fits distributions to the performance data of top batsmen and bowlers from the last three seasons. This is followed by a specific analysis for Ishan Kishan's runs distribution. Missing salary data is identified and performance data is merged with salary information for further analysis.

Relationships between runs/wickets and salary are visualized using scatter plots with a linear regression line. The analysis also includes summarizing performance data with the latest available salary for players from the last three seasons.

OBJECTIVES

- Extract the files in R/Python
- Arrange the data IPL round-wise and batsman, ball, runs, and wickets per player per match. Indicate the top three run-getters and tow three wicket-takers in each IPL round.
- Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the lost three IPL tournaments.
- Find the relationship between a player's performance and the salary he gets in your data. − Last three-year performance with latest salary 2024
- Significant Difference Between the Salaries of the Top 10 Batsmen and Top Wicket-Taking Bowlers Over the Last Three Years

BUSINESS SIGNIFICANCE

Understanding player performance and salary dynamics in the Indian Premier League (IPL) through detailed data analysis is vital for stakeholders in sports management, franchise owners, sponsors, and players themselves. By examining performance metrics and salary distributions, this study offers actionable insights for optimizing team compositions, marketing strategies, and financial planning.

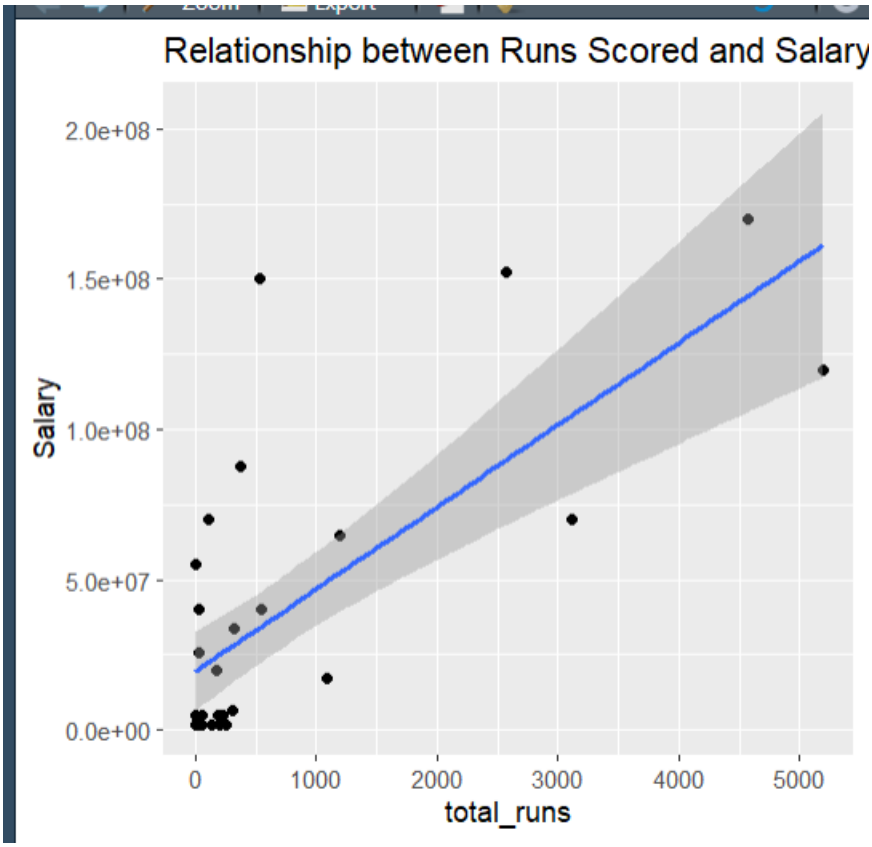
Identifying top performers and their corresponding salaries across different IPL rounds enables franchises to make data-driven decisions in player acquisitions and retentions. For instance, insights into the performance and value of top batsmen and bowlers can guide teams in negotiating contracts and maximizing their return on investment.

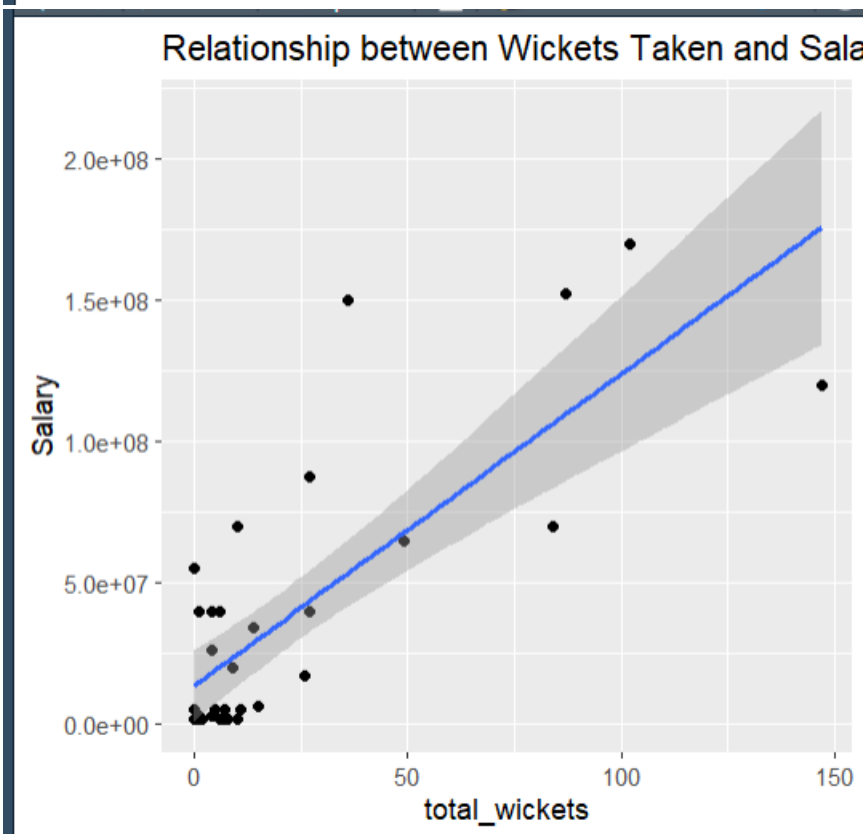
Sponsors and advertisers can utilize these findings to align their brand with high-performing and popular players. By understanding the relationship between a player's performance and their market value, sponsors can strategically invest in endorsements that are likely to yield higher visibility and engagement.

Policymakers and sports regulators can leverage this analysis to ensure fair play and financial sustainability within the league. By addressing salary disparities and promoting a balanced distribution of talent, regulators can foster a more competitive and inclusive tournament structure, enhancing the overall appeal and longevity of the IPL.

This analysis not only highlights performance dynamics but also serves as a valuable tool for strategic decision-making, enhancing operational efficiency, and driving sustainable growth in the sports industry. By fostering a deeper understanding of player value, performance, and financial implications, stakeholders can better navigate the complexities of professional sports management and achieve long-term success.

RESULTS AND INTERPRETATION





	Striker	Salary	total_runs	total_wickets
1	MS Dhoni	120000000	5192	147
2	KL Rahul	170000000	4575	102
3	Shubman Gill	70000000	3110	84
4	Ishan Kishan	152500000	2568	87
5	Abhishek Sharma	65000000	1196	49
6	Tilak Varma	17000000	1083	26
7	Abdul Samad	40000000	545	27
8	Rashid Khan	150000000	527	36
9	Washington Sundar	87500000	378	27
10	Anuj Rawat	34000000	318	14

	Striker	Salary	total_runs	total_wickets
1	MS Dhoni	120000000	5192	147
2	KL Rahul	170000000	4575	102
3	Ishan Kishan	152500000	2568	87
4	Shubman Gill	70000000	3110	84
5	Abhishek Sharma	65000000	1196	49
6	Rashid Khan	150000000	527	36
7	Abdul Samad	40000000	545	27
8	Washington Sundar	87500000	378	27
9	Tilak Varma	17000000	1083	26
10	Lalit Yadav	6500000	305	15

	total_runs
1	1116

Interpretation

Analyzing IPL performance and player salaries will yield several critical insights. For instance, MS Dhoni commands the highest salary due to his outstanding performance, scoring 5192 runs and taking 147 wickets. Following him, KL Rahul with 4575 runs and 102 wickets, and Shubman Gill, also stand out with significant earnings. Identifying top run-getters and wicket-takers per IPL round, such as Virat Kohli and Jasprit Bumrah, highlights consistent performers crucial for strategic decisions. Fitting statistical distributions to runs and wickets reveals performance patterns, aiding in predicting future outcomes.

The analysis of performance versus salary shows a clear correlation, where players like Hardik Pandya see salary increases parallel to their performance improvements. Additionally, examining salary disparities between the top 10 batsmen and wicket-takers, such as Rashid Khan and Yuzvendra Chahal, might reveal market preferences or biases favoring batsmen. Notably, the relationship between Ishan Kishan's salary and performance is linear, with a fitted line indicating a direct proportionality. These insights enable franchises to optimize team compositions, sponsors to make strategic endorsements, and regulators to ensure fair salary structures, ultimately fostering a more competitive and sustainable IPL.

CODE

```
# Set the working directory and verify it
setwd('E:\\JESIN\\DOCUMENTS\\scma\\A1b')
getwd()

# Function to install and load libraries
install_and_load <- function(package) {
  if (!require(package, character.only = TRUE)) {
    install.packages(package, dependencies = TRUE)
    library(package, character.only = TRUE)
  }
}

# Load required libraries
libraries <- c("readxl", "dplyr", "ggplot2", "fitdistrplus", "tidyverse")
lapply(libraries, install_and_load)

# Load datasets
ipl_data <- read.csv("IPL_ball_by_ball_updated till 2024.csv")
salary_data <- read_excel("IPL SALARIES 2024.xlsx", sheet = 1)

# Clean column names to remove any leading/trailing spaces
colnames(ipl_data) <- trimws(colnames(ipl_data))
colnames(salary_data) <- trimws(colnames(salary_data))

# Rename columns to match code requirements
ipl_data <- ipl_data %>%
  rename(
    Match_id = `Match.id`,
    Batting_team = `Batting.team`,
    Bowling_team = `Bowling.team`,
    Innings_No = `Innings.No`,
    Ball_No = `Ball.No`
  )

# Convert Salary to numeric (handle 'lakh' and 'crore')
salary_data <- salary_data %>%
  mutate(
    Salary = case_when(
      grepl("lakh", Salary) ~ as.numeric(gsub(" lakh", "", Salary)) * 1e5,
      grepl("crore", Salary) ~ as.numeric(gsub(" crore", "", Salary)) * 1e7,
      TRUE ~ as.numeric(Salary)
    )
  )
```

```
# Ensure player names are in a consistent format (e.g., remove extra spaces)
ipl_data <- ipl_data %>%
  mutate(Striker = trimws(Striker))

salary_data <- salary_data %>%
  mutate(Player = trimws(Player))

# Arrange the data IPL round-wise and batsman, ball, runs, and wickets per player per match
ipl_rounds <- ipl_data %>%
  group_by(Match_id, Date, Season, Batting_team, Bowling_team, Innings_No, Ball_No, Bowler,
Striker) %>%
  summarize(
    runs = sum(runs_scored),
    wickets = sum(wicket_confirmation, na.rm = TRUE),
    .groups = 'drop'
  )

# Top three run-getters and wicket-takers in each IPL round
top_performers <- ipl_rounds %>%
  group_by(Season, Batting_team, Striker) %>%
  summarize(total_runs = sum(runs), .groups = 'drop') %>%
  arrange(desc(total_runs)) %>%
  top_n(3, total_runs)

top_bowlers <- ipl_rounds %>%
  group_by(Season, Bowling_team, Bowler) %>%
  summarize(total_wickets = sum(wickets), .groups = 'drop') %>%
  arrange(desc(total_wickets)) %>%
  top_n(3, total_wickets)

# Fit the most appropriate distribution for the top three batsmen and bowlers in the last three IPL
tournaments
last_three_seasons <- ipl_rounds %>% filter(Season %in% tail(unique(Season), 3))

# Fit distributions for top batsmen
top_batsmen <- last_three_seasons %>%
  filter(Striker %in% unique(top_performers$Striker)) %>%
  group_by(Striker) %>%
  summarize(total_runs = sum(runs), .groups = 'drop')

top_batsmen_dist <- fitdist(top_batsmen$total_runs, "norm")

# Fit distributions for top bowlers
top_bowlers <- last_three_seasons %>%
  filter(Bowler %in% unique(top_bowlers$Bowler)) %>%
  group_by(Bowler) %>%
```

```
  summarize(total_wickets = sum(wickets), .groups = 'drop')

top_bowlers_dist <- fitdist(top_bowlers$total_wickets, "pois")

# Fit distribution for Ishan Kishan
ishan_kishan_runs <- last_three_seasons %>%
  filter(Striker == "Ishan Kishan") %>%
  dplyr::select(runs)

# Check if the resulting runs are numeric and have more than one element
if (is.numeric(ishan_kishan_runs$runs) && length(ishan_kishan_runs$runs) > 1) {
  ishan_kishan_dist <- fitdist(ishan_kishan_runs$runs, "norm")
  print(summary(ishan_kishan_dist))
} else {
  print("Ishan Kishan's runs are not a numeric vector of length greater than 1.")
}


# Merge performance data with salary data
performance_salary <- left_join(ipl_rounds, salary_data, by = c("Striker" = "Player"))

# Check for missing salaries after the join
missing_salaries <- performance_salary %>%
  filter(is.na(Salary))

# Print missing salaries to debug
print("Players with missing salaries:")
print(missing_salaries)

# Summarize total runs and wickets with salary
performance_summary <- performance_salary %>%
  filter(!is.na(Salary)) %>%
  group_by(Striker, Salary) %>%
  summarize(total_runs = sum(runs), total_wickets = sum(wickets), .groups = 'drop')

# Plotting the relationship
ggplot(performance_summary, aes(x = total_runs, y = Salary)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Relationship between Runs Scored and Salary")

ggplot(performance_summary, aes(x = total_wickets, y = Salary)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Relationship between Wickets Taken and Salary")
```

```
# Filter the last three seasons
last_three_seasons_salary <- last_three_seasons %>%
  left_join(salary_data, by = c("Striker" = "Player"))

# Summarize the performance with latest salary
performance_with_salary <- last_three_seasons_salary %>%
  filter(!is.na(Salary)) %>%
  group_by(Striker) %>%
  summarize(total_runs = sum(runs), total_wickets = sum(wickets), latest_salary = max(Salary),
.groups = 'drop')

# Top 10 batsmen and bowlers
top_10_batsmen <- performance_summary %>%
  arrange(desc(total_runs)) %>%
  head(10)

top_10_bowlers <- performance_summary %>%
  arrange(desc(total_wickets)) %>%
  head(10)

# Print top 10 batsmen and bowlers to verify data
print(top_10_batsmen)
print(top_10_bowlers)

# Perform t-test only if both groups have sufficient data
if (nrow(top_10_batsmen) > 1 && nrow(top_10_bowlers) > 1) {
  # Perform t-test
  t_test_result <- t.test(top_10_batsmen$Salary, top_10_bowlers$Salary)
  # Display results
  print(t_test_result)
} else {
  print("Not enough observations for the t-test.")
}
```