VIRGINIA COMMONWEALTH UNIVERSITY

Statistical Analysis & Modelling

A1b – Distribution using IPL Dataset

Using R

Submitted by

MONIKA SARADHA

#V01068784

Date of Submission: 06/06/2023

Table of Contents

1. Introduction

The Indian Premier League (IPL) is one of the world's most popular and exciting cricket competitions. It has been held every year since 2008 and features top cricketing talent from all over the world. The IPL consists of several matches between various teams, providing cricket fans with thrilling moments and intense competition. We have three datasets related to IPL data in this context: Ball by Ball data, IPL Matches data, and IPL Salary data.

1.1. About the Data

Ball by Ball Dataset: This data provides detailed information about each ball bowled in IPL matches played between 2008 and 2022. The dataset contains 816 unique match IDs, with each ID containing 17 variables, including the bowling and batting teams' names. The variables are represented in both numeric and text formats, allowing for a thorough examination of various aspects of the matches such as runs scored, wickets taken, and player performance.

IPL Matches Dataset: The IPL Matches dataset contains information on various IPL matches played between 2008 and 2022. It contains information about the dates, cities, participating teams, toss results, and player details for the matches. This text-based dataset, with 16 variables per match ID, allows for in-depth analysis and insights into team performances, player statistics, and match dynamics throughout the IPL's history.

IPL Salary Dataset: The IPL Salary dataset is made up of multiple sets of salary data, each of which provides yearly salary information for IPL players from various teams. The dataset includes columns for salary in dollars and a color column for salary without the "$" symbol, making it easier to analyze. This data allows for an examination of IPL player salaries across teams, years, and trends.

Objective

The goal of analyzing these IPL datasets is to gain insights and valuable information about the matches, players, and team dynamics. We can assess individual player performances, identify trends in batting and bowling strategies, and uncover factors that contribute to team success by analyzing ball-by-ball data. The IPL Matches data allows us to assess the impact of factors such as toss results, home advantage, and team compositions on match outcomes. Furthermore, the IPL

Salary data enables research into player salaries, team spending patterns, and the relationship between player performance and financial investments.

The objective is to analyze IPL data in order to:

- Arrange the data in a circular fashion, recording batsman, ball, runs, and wickets per player per match. Determine the top three run scorers and the bottom three wicket-takers in each IPL round.
- Determine the most appropriate statistical distribution for the top three batsmen and bowlers in the last three IPL tournaments in terms of runs and wickets.
- Examine the relationship between player performance and salary to better understand how performance affects player salaries.

Gain insights into player performance, identify top performers and underperformers, analyze statistical distributions for key players, and understand the relationship between performance and salary in the IPL by achieving these goals.

1.2. Business Significance

Analysis of IPL data can provide significant business value to a variety of stakeholders, including team management, sponsors, broadcasters, and fans. Insights derived from data can help team managers with player selection, strategy formulation, and resource allocation, ultimately improving team performance and increasing chances of success. The analysis can be used by sponsors and broadcasters to identify popular players, optimize marketing strategies, and make sound investment decisions. Furthermore, fans can gain a better understanding of their favorite teams and players, increasing their engagement and enjoyment of the IPL.

Businesses and stakeholders can make data-driven decisions, improve their competitive edge, and maximize the value derived from their involvement in the IPL ecosystem by leveraging the comprehensive IPL datasets. The Indian Premier League (IPL) has had a significant impact on the Indian economy:

- Rise in GDP
- Boosts tourism
- Generates jobs

- Media exposure and viewership
- Cultural diversity
- Hotel and restaurant business
- Increase in tax contribution

2. Results

2.1. Data Preprocessing

Structure of the dataset IPL ball by ball:

```
> str(ipl_ballby)
tibble [225,954 × 18] (S3: tbl_df/tbl/data.frame)
 $ ID              : num [1:225954] 1312200 1312200 1312200 1312200 1312200 1312200 ...
 $ Season          : chr [1:225954] "1312" "1312" "1312" "1312" "1312" ...
 $ innings         : num [1:225954] 1 1 1 1 1 1 1 1 1 1 ...
 $ overs           : num [1:225954] 0 0 0 0 0 0 1 1 1 1 ...
 $ ballnumber      : num [1:225954] 1 2 3 4 5 6 1 2 3 4 ...
 $ batter          : chr [1:225954] "YBK Jaiswal" "YBK Jaiswal" "JC Buttler" "YBK
Jaiswal" ...
 $ bowler          : chr [1:225954] "Mohammed Shami" "Mohammed Shami" "Mohammed Sh
ami" "Mohammed Shami" ...
 $ non-striker     : chr [1:225954] "JC Buttler" "JC Buttler" "YBK Jaiswal" "JC Bu
ttler" ...
 $ extra_type      : chr [1:225954] "NA" "legbyes" "NA" "NA" ...
 $ batsman_run     : num [1:225954] 0 0 1 0 0 0 0 0 4 0 ...
 $ extras_run      : num [1:225954] 0 1 0 0 0 0 0 0 0 0 ...
 $ total_run       : num [1:225954] 0 1 1 0 0 0 0 0 4 0 ...
 $ non_boundary    : num [1:225954] 0 0 0 0 0 0 0 0 0 0 ...
 $ isWicketDelivery: num [1:225954] 0 0 0 0 0 0 0 0 0 0 ...
 $ player_out      : chr [1:225954] "NA" "NA" "NA" "NA" ...
 $ kind            : chr [1:225954] "NA" "NA" "NA" "NA" ...
 $ fielders_involved: chr [1:225954] "NA" "NA" "NA" "NA" ...
 $ BattingTeam     : chr [1:225954] "Rajasthan Royals" "Rajasthan Royals" "Rajasth
an Royals" "Rajasthan Royals" ...
>
```

Inference:

The ipl_ballby dataset contains 225,954 rows and 18 columns. It provides information about various aspects of IPL matches, including player details, innings, overs, runs scored, extras, wickets, and fielding details.

Structure of the dataset IPL salary:

```
> str(ipl_salary)
tibble [1,108 × 42] (S3: tbl_df/tbl/data.frame)
 $ Id          : num [1:1108] 1 1 1 1 1 1 1 1 1 1 ...
 $ Name        : chr [1:1108] "AB de Villiers" "AB de Villiers" "AB de Villiers" "AB
de Villiers" ...
 $ Year        : num [1:1108] 2008 2009 2010 2011 2012 ...
 $ Final Price: num [1:1108] 12048000 14736000 13887000 50600000 55297000 ...
 $ Role        : chr [1:1108] "Wicketkeeper batsman" "Wicketkeeper batsman" "Wicketk
eeper batsman" "Wicketkeeper batsman" ...
 $ Nationality: chr [1:1108] "South African" "South African" "South African" "South
African" ...
 $ Team        : chr [1:1108] "DD" "DD" "DD" "RCB" ...
 $ Ent         : num [1:1108] 0 0 0 0 0 0 0 0 0 0 ...
 $ Age         : num [1:1108] 24 25 26 27 28 29 30 31 32 33 ...
 $ Matches     : num [1:1108] 8 15 7 16 16 14 14 16 16 9 ...
 $ LMatches    : num [1:1108] 0 8 15 7 16 16 14 14 16 16 ...
 $ Runs        : num [1:1108] 95 465 111 312 319 360 395 513 687 216 ...
 $ LRuns       : num [1:1108] 0 95 465 111 312 319 360 395 513 687 ...
 $ HS          : num [1:1108] 26 105 45 65 64 64 89 133 129 89 ...
 $ LHS         : num [1:1108] 0 26 105 45 65 64 64 89 133 129 ...
 $ Ave         : num [1:1108] 19 51.7 15.8 34.7 39.9 ...
 $ LAve        : num [1:1108] 0 19 51.7 15.9 34.7 ...
 $ StrRate     : num [1:1108] 96.9 131 93.3 128.4 161.1 ...
 $ LStrRate    : num [1:1108] 0 96.9 131 93.3 128.4 ...
 $ Fifties     : num [1:1108] 0 3 0 2 3 2 3 2 6 1 ...
 $ LFifties    : num [1:1108] 0 0 3 0 2 3 2 3 2 6 ...
 $ Hundreds    : num [1:1108] 0 1 0 0 0 0 0 1 1 0 ...
 $ LHundreds   : num [1:1108] 0 0 1 0 0 0 0 0 1 1 ...
 $ Fours       : num [1:1108] 5 39 7 21 26 34 26 60 57 12 ...
 $ LFours      : num [1:1108] 0 5 39 7 21 26 34 26 60 57 ...
 $ Sixes       : num [1:1108] 1 12 0 14 15 15 24 22 37 16 ...
```

Inference:

The ipl_salary dataset contains 1,108 rows and 42 columns. It provides information about IPL players' salaries, including their names, nationalities, teams, roles, years, and various performance statistics such as runs scored, batting average, strike rate, number of fifties and hundreds, catches taken, and wickets taken.

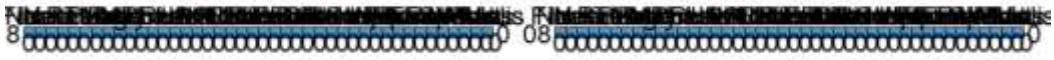
Structure of the dataset IPL matches:

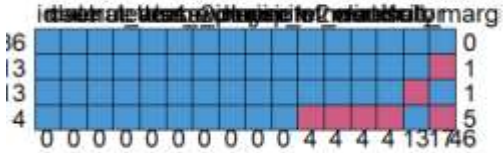
```
> str(ipl_match)
'data.frame':   816 obs. of  16 variables:
 $ id              : int  335982 335983 335984 335985 335986 335987 335988 335989 33
5990 335991 ...
 $ city            : chr  "Bangalore" "Chandigarh" "Delhi" "Mumbai" ...
 $ date            : chr  "4/18/2008" "4/19/2008" "4/19/2008" "4/20/2008" ...
 $ player_of_match: chr  "BB McCullum" "MEK Hussey" "MF Maharoof" "MV Boucher" ...
 $ venue           : chr  "M Chinnaswamy Stadium" "Punjab Cricket Association Stadiu
m, Mohali" "Feroz Shah Kotla" "Wankhede Stadium" ...
 $ neutral_venue   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ team1           : chr  "Royal Challengers Bangalore" "Kings XI Punjab" "Delhi Dar
edevils" "Mumbai Indians" ...
 $ team2           : chr  "Kolkata Knight Riders" "Chennai Super Kings" "Rajasthan R
oyals" "Royal Challengers Bangalore" ...
 $ toss_winner     : chr  "Royal Challengers Bangalore" "Chennai Super Kings" "Rajas
than Royals" "Mumbai Indians" ...
 $ toss_decision   : chr  "field" "bat" "bat" "bat" ...
 $ winner          : chr  "Kolkata Knight Riders" "Chennai Super Kings" "Delhi Dared
evils" "Royal Challengers Bangalore" ...
 $ result          : chr  "runs" "runs" "wickets" "wickets" ...
 $ result_margin   : int  140 33 9 5 5 6 9 6 3 66 ...
 $ eliminator      : chr  "N" "N" "N" "N" ...
 $ umpire1         : chr  "Asad Rauf" "MR Benson" "Aleem Dar" "SJ Davis" ...
 $ umpire2         : chr  "RE Koertzen" "SL Shastri" "GA Pratapkumar" "DJ Harper"
...
```

Inference:

The ipl_match dataset contains 816 observations (rows) and 16 variables (columns). It provides information about IPL matches, including the match ID, city, date, player of the match, venue, whether it is a neutral venue, teams involved, toss details, winner, result (runs/wickets), result margin, whether there was an eliminator, and the names of the umpires.

Missing Values Analysis:





Inference:

It was observed that the datasets ipl_ballby and ipl_salary did not have any missing values, although the dataset ipl_match had around 84 missing values, which had to be worked on.

```
> ipl_matchclean <- na.omit(ipl_match)
> sum(is.na(ipl_matchclean))
[1] 0
```

By means of omitting the NA values the imputation was dealt with. The column method and less significant ones like eliminator were major contributors.

2.2. Player Understanding and Highlights

IPL ball by ball:

```
A tibble: 6 x 20
     ID Season innings overs ballnumber batter  bowler non-s…¹ extra…² batsm…³
  <dbl> <chr>     <dbl> <dbl>      <dbl> <chr>   <chr>  <chr>   <chr>     <dbl>
1312200 1312          1     0          1 YBK Ja… Moham… JC But… NA            0
1312200 1312          1     0          2 YBK Ja… Moham… JC But… legbyes       0
1312200 1312          1     0          3 JC But… Moham… YBK Ja… NA            1
1312200 1312          1     0          4 YBK Ja… Moham… JC But… NA            0
1312200 1312          1     0          5 YBK Ja… Moham… JC But… NA            0
1312200 1312          1     0          6 YBK Ja… Moham… JC But… NA            0
… with 10 more variables: extras_run <dbl>, total_run <dbl>,
  non_boundary <dbl>, iswicketDelivery <dbl>, player_out <chr>, kind <chr>,
  fielders_involved <chr>, BattingTeam <chr>, match <dbl>, new <dbl>, and
  abbreviated variable names ¹`non-striker`, ²extra_type, ³batsman_run
ℹ Use `colnames()` to see all variable names
```

IPL salary:

```
head(ipl_salary)
A tibble: 6 x 42
     Id Name        Year Final…¹ Role  Natio…² Team    Ent   Age Matches LMatc…³
  <dbl> <chr>      <dbl>   <dbl> <chr> <chr>   <chr> <dbl> <dbl>   <dbl>   <dbl>
      1 AB de Vil… 2008  1.20e7 Wick… South … DD        0    24       6       0
      1 AB de Vil… 2009  1.47e7 Wick… South … DD        0    25      15       6
      1 AB de Vil… 2010  1.39e7 Wick… South … DD        0    26       7      15
      1 AB de Vil… 2011  5.06e7 Wick… South … RCB       0    27      16       7
      1 AB de Vil… 2012  5.53e7 Wick… South … RCB       0    28      16      16
      1 AB de Vil… 2013  5.86e7 Wick… South … RCB       0    29      14      16
… with 31 more variables: Runs <dbl>, LRuns <dbl>, HS <dbl>, LHS <dbl>,
  Ave <dbl>, LAve <dbl>, StrRate <dbl>, LStrRate <dbl>, Fifties <dbl>,
  LFifties <dbl>, Hundreds <dbl>, LHundreds <dbl>, Fours <dbl>, LFours <dbl>,
  Sixes <dbl>, LSixes <dbl>, Catches <dbl>, LCatches <dbl>, Stumps <dbl>,
  LStumps <dbl>, Wkts <dbl>, LWkts <dbl>, Econ <dbl>, LEcon <dbl>,
  FourWkts <dbl>, LFourWkts <dbl>, FiveWkts <dbl>, LFiveWkts <dbl>,
  Indian <dbl>, Specialist <dbl>, Status <dbl>, and abbreviated variable …
ℹ Use `colnames()` to see all variable names
```

Inference:

The match column is created in the ipl_ballby data frame by extracting the match ID from the ID column.The new column is created in the ipl_ballby data frame based on the match ID. The match ID values 118, 123, 131, 981, 336, 597, 733, and 734 are replaced with their corresponding adjusted values.The Season column is created in the ipl_ballby data frame based on the adjusted match ID values. Each adjusted match ID value is mapped to its corresponding season (e.g., 392 is mapped to "2009", 419 is mapped to "2010", etc.).

Top Run Getters:

```
> top_run_getters
# A tibble: 85 x 4
# Groups:   Season [28]
   Season batter        Total_Balls Total_Runs
   <chr>  <chr>               <int>      <dbl>
 1 1082   DA Warner             462        670
 2 1082   G Gambhir             405        531
 3 1082   S Dhawan              383        495
 4 1136   KS Williamson         522        747
 5 1136   RR Pant               412        717
 6 1136   KL Rahul              426        678
 7 1175   DA Warner             166        274
 8 1175   JM Bairstow           149        255
 9 1175   AD Russell             85        223
10 1178   KL Rahul              369        518
# … with 75 more rows
# ℹ Use `print(n = ...)` to see more rows
>
```

Inference:

Each row represents a player's performance during a specific season, with the total number of balls faced (Total_Balls) and total runs scored (Total_Runs) indicated. In season 1082, for example, the

top three run scorers were DA Warner, G Gambhir, and S Dhawan, with total runs of 670, 531, and 495, respectively.

Low Wicket Takers:

```
low_wicket_takers
A tibble: 436 x 4
Groups:   Season [28]
   Season bowler           Total_Balls Total_Wickets
   <chr>  <chr>                  <int>         <dbl>
 1 1082   AD Mathews                31             0
 2 1082   Ankit Sharma               6             0
 3 1082   Anureet Singh             12             0
 4 1082   BB Sran                   13             0
 5 1082   I Sharma                 114             0
 6 1082   J Yadav                   37             0
 7 1082   KA Pollard                12             0
 8 1082   MN Samuels                18             0
 9 1082   MS Gony                   15             0
10 1082   Mustafizur Rahman         17             0
# ... with 426 more rows
# Use `print(n = ...)` to see more rows
```

Inference:

Each row represents a player's season performance, displaying the total number of balls bowled (Total_Balls) and wickets taken (Total_Wickets). For example, in season 1082, several bowlers, including AD Mathews, Ankit Sharma, Anureet Singh, and others, did not take any wickets.

group_by() function to group the data based on the specified grouping variables, which are "Season" for both top_run_getters and low_wicket_takers.

2.3. Performance Characteristics analysis for Prediction

Economy of Players:

```
Bowler          Economy
> for (i in 1:nrow(sorted_bowlers)) {
+   cat(sorted_bowlers$bowler[i], "\t\t", sorted_
+ }
AC Gilchrist            0
Sachin Baby             0.05363128
AM Rahane               0.05714286
SS Iyer                 0.06278027
DJ Thornely             0.07606973
T Henderson             0.07809524
PH Solanki              0.0797546
M Manhas                0.08131655
DA Warner               0.08275862
RS Gavaskar             0.0855615
A Badoni                0.08633094
M Pathirana             0.0876598
PD Collingwood          0.08794079
Ramandeep Singh                 0.09070918
RV Patel                0.09328622
```

Inference:

The economy was calculated using each player's runs conceded and overs bowled. Total runs conceded by a bowler were calculated by adding the "total_run" column to the converted fraction of the "ballnumber" column (divided by 6 to represent overs). Total overs bowled were calculated by adding the "overs" column to the converted fraction of the "ballnumber" column (divided by 6 to represent overs). The economy was then calculated by dividing the total number of runs conceded by the total number of overs bowled by each player. This calculation calculates the bowler's efficiency in terms of runs conceded per over.

Filtering the recent 3 tournaments based on Season and ID

```
> recent_data <- ipl_ballbyf %>%
+   filter(ID %in% last_three_tournaments$ID & Season %in% last_three_tournaments$Season)
> recent_data
# A tibble: 725 x 18
        ID Season innings overs ballnumber batter bowler non-s...' extra...² batsm...³
     <db1> <fct>     <db1> <db1>      <db1> <chr>  <chr>  <chr>    <chr>       <db1>
 1 1312200 1312          1     0          1 YBK J... Moham... JC But... NA            0
 2 1312200 1312          1     0          2 YBK J... Moham... JC But... legbyes       0
 3 1312200 1312          1     0          3 JC Bu... Moham... YBK Ja... NA            1
 4 1312200 1312          1     0          4 YBK J... Moham... JC But... NA            0
 5 1312200 1312          1     0          5 YBK J... Moham... JC But... NA            0
 6 1312200 1312          1     0          6 YBK J... Moham... JC But... NA            0
 7 1312200 1312          1     1          1 JC Bu... Yash ... YBK Ja... NA            0
 8 1312200 1312          1     1          2 JC Bu... Yash ... YBK Ja... NA            0
 9 1312200 1312          1     1          3 JC Bu... Yash ... YBK Ja... NA            4
10 1312200 1312          1     1          4 JC Bu... Yash ... YBK Ja... NA            0
```

Inference:

We extracted data from the given IPL ball-by-ball data for the three most recent tournaments. It accomplishes this by sorting the data in descending order by the "ID" and "Season" columns, and then selecting the top three distinct combinations of "ID" and "Season." The resulting data is based on the last three tournaments.

Top 3 Bowlers with least economy recent 3 tournaments:

```
+   cat(sorted_bowlers$bowler[i],  "\t\t ",
+ }
HH Pandya               0.0647482
Rashid Khan             0.07826087
GJ Maxwell              0.08153477
```

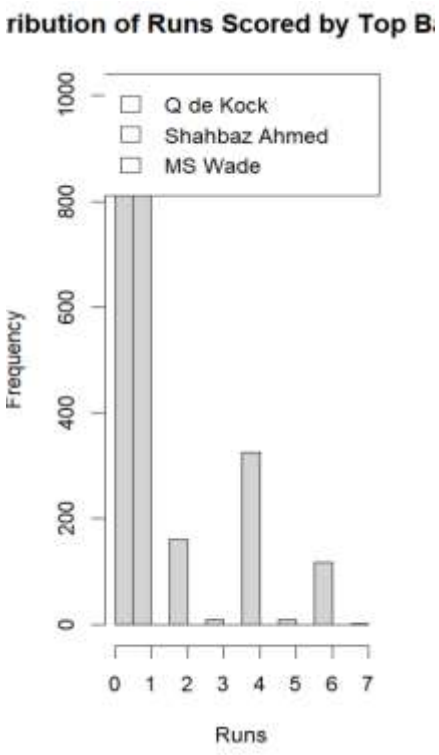
Top 3 Batsmen in the recent 3 tournaments:

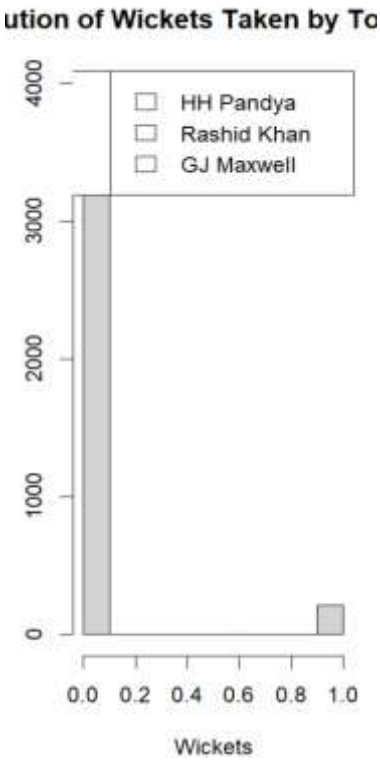
```
+ }
Q de Kock              600
Shahbaz Ahmed          400
MS Wade                400
```

Inference:

For the batsmen, it shows the strike rate, batting average, and consistency. For the bowlers, it displays the economy. From the results, we observe that Q de Kock, Shahbaz Ahmed, and MS Wade have comparable statistics. Among the bowlers, HH Pandya has the lowest economy, followed by Rashid Khan and GJ Maxwell.

Histogram of runs scored and wicket taken by top 3 Bastmen and Bowlers respectively:





GOF for Top 3 Batsmen:

```
> print(runs_gof)
Goodness-of-fit statistics
                               1-mle-norm
Kolmogorov-Smirnov statistic    0.3373518
Cramer-von Mises statistic     47.9524492
Anderson-Darling statistic    260.1045515

Goodness-of-fit criteria
                               1-mle-norm
Akaike's Information Criterion   9758.291
Bayesian Information Criterion   9769.974
>
```

Inference:

Appropriate distribution for the runs scored data is the normal distribution. The Kolmogorov-Smirnov statistic of 0.3373518 for the runs scored data indicates that the fit to a normal distribution is not ideal, as a smaller value indicates a better fit. The Cramer-von Mises and Anderson-Darling statistics are also relatively high, indicating a poorer fit.

GOF for Top 3 Bowlers:

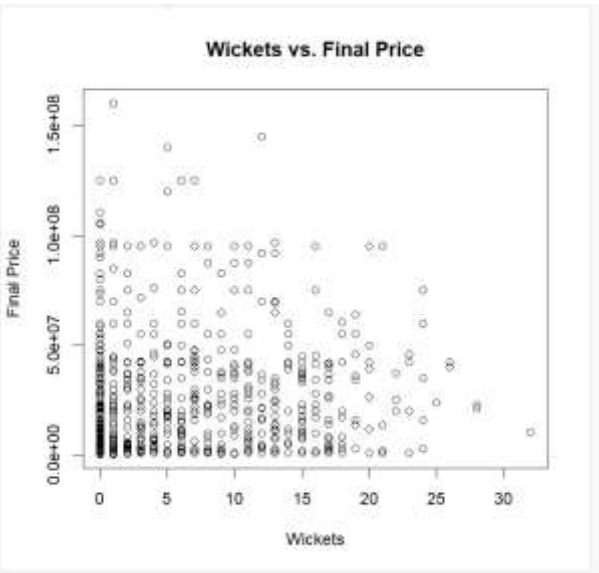
```
> print(wickets_gof)
Chi-squared statistic:  5.314183
Degree of freedom of the Chi-squared distribution:  1
Chi-squared p-value:  0.02115251
   the p-value may be wrong with some theoretical counts < 5
Chi-squared table:
       obscounts  theocounts
<= 0 3935.000000 3939.898900
<= 1  203.000000  193.281654
> 1     0.000000    4.819446

Goodness-of-fit criteria
                              1-mle-pois
Akaike's Information Criterion   1631.993
Bayesian Information Criterion   1638.321
```

Inference:

Appropriate distribution for the wickets taken data is the Poisson distribution. The Chi-squared p-value of 0.02115251 for the wickets taken data indicates that the fit to a Poisson distribution is statistically significant, indicating a reasonable fit. However, because the p-value may be inaccurate when theoretical counts are less than 5, more investigation may be required.

2.4. Factors Influencing Salary of Players

Correlation:



Inference:

The correlation coefficient between Runs and Final.Price is approximately 0.471, indicating a moderate positive linear relationship between these two variables. This suggests that there is a tendency for higher Runs to be associated with higher Final Prices.

The correlation coefficient between Wkts and Final.Price is approximately 0.056, indicating a weak positive linear relationship between these two variables. This suggests that there is a slight tendency for higher Wickets to be associated with higher Final Prices, but the relationship is not as strong as in the case of Runs and Final Prices.

Wickets alone wouldn't primarily affect the value of a bowler other parameters included in the economy also affects the result.

3. Recommendation

3.1. Business Implications

- Analyzing player performance allows teams to make informed decisions about team selection and strategy.
- Understanding the relationship between performance and salary can help you evaluate a player's worth during auctions and negotiations.
- Sponsorship and branding: Identifying top performers helps sponsors select appropriate ambassadors and maximize brand exposure.
- Fan Engagement: Sharing information about player performance improves fan engagement and the overall fan experience.
- Analysis of player performance data assists teams in tailoring strategies to individual strengths and weaknesses.

3.2. Business Recommendations

- Player Investment: Invest strategically in high-performing players who contribute significantly to team success.
- Talent Development entails identifying promising players and providing them with training and opportunities for advancement.
- Collaboration with sponsors to leverage the popularity and performance of top players for brand visibility.
- Fan Interaction: Using social media campaigns and events that highlight top performers and encourage participation, engage fans.
- Performance-based Contracts: In order to incentivize consistent performance, consider including performance-based clauses in player contracts.

Implementing these recommendations in the IPL ecosystem can result in improved team performance, increased brand value, increased fan engagement, and effective player management.

4. Reference:

- Manager, S. (2021, April 24). Impact of IPL on Indian Economy - The Sports School Blog. The Sports School – Integrated School for Sports & Academics. https://thesportsschool.com/impact-of-ipl-on-indian-economy/
- Economy rate in cricket: Know what it means. (2022, April 9). SportsAdda. https://www.sportsadda.com/cricket/features/what-is-economy-rate-cricket

5. Codes

5.1. R-studio

```
library(readxl)

ipl_ballby<-
read_excel("C:\\Users\\monis\\OneDrive\\Desktop\\SCM\\IPL_Ball_by_Ball_2008_2022.xlsx")

ipl_salary<-read_excel("C:\\Users\\monis\\OneDrive\\Desktop\\SCM\\IPL salary 2008_2022.xlsx")

ipl_match<-read.csv("C:\\Users\\monis\\OneDrive\\Desktop\\SCM\\IPL Matches 2008-2020.csv")

#Data Understanding

summary(ipl_ballby)

str(ipl_ballby)

print(colnames(ipl_ballby))

summary(ipl_salary)

str(ipl_salary)

print(colnames(ipl_salary))

summary(ipl_match)

str(ipl)

print(colnames(ipl_match))

#Missing Value Analysis

sum(is.na(ipl_ballby))

sum(is.na(ipl_salary))

sum(is.na(ipl_match))

is.na(ipl_match)

#Column removal (missing data)

ipl_match <- ipl_match[, !names(ipl_match) %in% c("method")]

md.pattern(ipl_ballby)

md.pattern(ipl_salary)

md.pattern(ipl_match)
```

```
missing_values <- colSums(is.na(ipl_match))
```

 # Display columns with missing values

```
columns_with_missing <- names(missing_values[missing_values > 0])
print(columns_with_missing)
```

 # Remove missing values from Ipl match dataset

```
ipl_matchclean <- na.omit(ipl_match)
sum(is.na(ipl_matchclean))
#-----------------------------------------------------------------
```

#b) Arrange the data IPL round wise and batsman and ball and runs and wickets per player per match. Indicate the top three run getters and low three wicket takers in each IPL round.

```
# Set the Season based on the match ID
ipl_ballby$match <- as.numeric(substr(ipl_ballby$ID, 1, 3))
ipl_ballby$new <- ifelse(ipl_ballby$match == 118, 117,
                         ifelse(ipl_ballby$match == 123, 121,
                                ifelse(ipl_ballby$match == 131, 130,
                                       ifelse(ipl_ballby$match == 981, 980,
                                              ifelse(ipl_ballby$match == 336, 335,
                                                     ifelse(ipl_ballby$match == 597, 598,
                                                            ifelse(ipl_ballby$match == 733, 729,
                                                                   ifelse(ipl_ballby$match == 734, 729, ipl_ballby$match)))))))



#Ballbyball
ipl_ballby$Season <- ifelse(ipl_ballby$new == 392, "2009",
                            ifelse(ipl_ballby$new == 419, "2010",
                                   ifelse(ipl_ballby$new == 501, "2011",
                                          ifelse(ipl_ballby$new == 548, "2012",
                                                 ifelse(ipl_ballby$new == 598, "2013",
                                                        ifelse(ipl_ballby$new == 729, "2014",
```

```
                                        ifelse(ipl_ballby$new == 829, "2015",
                                            ifelse(ipl_ballby$new == 980, "2016",
                                                ifelse(ipl_ballby$new == 108, "2017",
                                                    ifelse(ipl_ballby$new == 113, "2018",
                                                        ifelse(ipl_ballby$new == 117, "2019",
                                                            ifelse(ipl_ballby$new == 121, "2020",
ifelse(ipl_ballby$new == 125, "2021",
                                                                ifelse(ipl_ballby$new == 130,
"2022", "unknown"))))))))))))

#View the modified data
head(ipl_ballby)
head(ipl_salary)
head(ipl_match)

library(dplyr)
sorted_data <- arrange(ipl_ballby, Season, batter)
summary_data <- sorted_data %>%
  group_by(Season, batter) %>%
  summarize(Total_Balls = n(),
            Total_Runs = sum(total_run),
            .groups = 'drop')

top_run_getters <- summary_data %>%
  group_by(Season) %>%
  top_n(3, Total_Runs) %>%
  arrange(Season, desc(Total_Runs))

sorted_data <- arrange(ipl_ballby, Season, bowler)
```

```
summary_data <- sorted_data %>%
  group_by(Season, bowler) %>%
  summarize(Total_Balls = n(),
            Total_Wickets = sum(isWicketDelivery),
            .groups = 'drop')

low_wicket_takers <- summary_data %>%
  group_by(Season) %>%
  top_n(-3, Total_Wickets) %>%
  arrange(Season, Total_Wickets)

highest_wicket_takers <- summary_data %>%
  group_by(Season) %>%
  top_n(3, Total_Wickets) %>%
  arrange(Season, Total_Wickets)

top_run_getters
low_wicket_takers
highest_wicket_takers
#------------------------------------------------------------------------
```

#c) Fit the most appropriate distribution for runs scored and wickets take by the top three batsmen and bowlers in the lost three IPL tournaments.

```
library(dplyr)
library(MASS)
library(fitdistrplus)

ipl_ballbyf<-
read_excel("C:\\Users\\monis\\OneDrive\\Desktop\\SCM\\IPL_Ball_by_Ball_2008_2022.xlsx")
```

```
colnames(ipl_ballbyf)
# Calculate economy for each player
player_economy <- ipl_ballbyf %>%
  group_by(bowler) %>%
  summarise(
    Total_Runs_Conceded = sum(total_run),
    Total_Overs_Bowled = sum(overs) + sum(ballnumber) / 6,
    Economy = Total_Runs_Conceded / Total_Overs_Bowled
  )

# View the economy of players
print(player_economy)

# Sort the bowlers based on economy in ascending order
sorted_bowlers <- player_economy %>%
  arrange(Economy)

# Print the bowlers and their economy
cat("Bowlers and their Economy (Low to High):\n")
cat("Bowler\t\tEconomy\n")
for (i in 1:nrow(sorted_bowlers)) {
  cat(sorted_bowlers$bowler[i], "\t\t", sorted_bowlers$Economy[i], "\n")
}

library(dplyr)

# Convert ipl_ballbyf to a tibble
ipl_ballbyf <- as_tibble(ipl_ballbyf)
```

```
# Calculate Strike Rate, Batting Average, and Consistency for all players
player_stats <- ipl_ballbyf %>%
  group_by(batter) %>%
  summarise(total_balls_faced = sum(batsman_run != 0),
            total_runs_scored = sum(batsman_run),
            total_dismissals = sum(!is.na(player_out)),
            total_innings_played = n_distinct(innings),
            half_centuries = sum(batsman_run >= 50),
            centuries = sum(batsman_run >= 100)) %>%
  mutate(strike_rate = (total_runs_scored / total_balls_faced) * 100,
         batting_average = total_runs_scored / total_dismissals,
         consistency = ((half_centuries + centuries) / total_innings_played) * 100)

# Sort the players based on the three parameters in descending order
top_batsmen <- player_stats %>%
  arrange(desc(strike_rate), desc(batting_average), desc(consistency)) %>%
  head(3)

# Print the top 3 batsmen
cat("Top 3 Batsmen:\n")
cat("Batsman\t\tStrike Rate\tBatting Average\tConsistency\n")
for (i in 1:nrow(top_batsmen)) {
  cat(top_batsmen$batter[i],   "\t\t",   top_batsmen$strike_rate[i],   "\t\t",   top_batsmen$batting_average[i],
"\t\t", top_batsmen$consistency[i], "\n")
}

#to find recent 3 tournaments
sorted_data <- ipl_ballbyf %>%
  arrange(desc(ID), desc(Season))
```

```
# Get the last three tournaments
last_three_tournaments <- sorted_data %>%
  distinct(ID, Season) %>%
  slice_head(n = 3)
recent_data <- ipl_ballbyf %>%
  filter(ID %in% last_three_tournaments$ID & Season %in% last_three_tournaments$Season)
recent_data

# Calculate economy for each player in the last three tournaments
player_economy <- recent_data %>%
  group_by(bowler) %>%
  summarise(
    Total_Runs_Conceded = sum(total_run),
    Total_Overs_Bowled = sum(overs) + sum(ballnumber) / 6,
    Economy = Total_Runs_Conceded / Total_Overs_Bowled
  )

# Sort the bowlers based on economy in ascending order
sorted_bowlers <- player_economy %>%
  arrange(Economy)

# Print the top 3 bowlers and their economy
cat("Top 3 Bowlers (Lowest Economy) in the last three IPL tournaments:\n")
cat("Bowler\t\tEconomy\n")
for (i in 1:min(3, nrow(sorted_bowlers))) {
  cat(sorted_bowlers$bowler[i], "\t\t", sorted_bowlers$Economy[i], "\n")
}
```

```
# Calculate Strike Rate, Batting Average, and Consistency for all players in the last three tournaments
player_stats <- recent_data %>%
  group_by(batter) %>%
  summarise(
    total_balls_faced = sum(batsman_run != 0),
    total_runs_scored = sum(batsman_run),
    total_dismissals = sum(!is.na(player_out)),
    total_innings_played = n_distinct(innings),
    half_centuries = sum(batsman_run >= 50),
    centuries = sum(batsman_run >= 100)
  )
  mutate(
    strike_rate = (total_runs_scored / total_balls_faced) * 100,
    batting_average = total_runs_scored / total_dismissals,
    consistency = ((half_centuries + centuries) / total_innings_played) * 100
  )

# Sort the players based on the three parameters in descending order
top_batsmen <- player_stats %>%
  arrange(desc(strike_rate), desc(batting_average), desc(consistency)) %>%
  head(3)

# Print the top 3 batsmen
cat("Top 3 Batsmen in the last three IPL tournaments:\n")
cat("Batsman\t\tStrike Rate\tBatting Average\tConsistency\n")
for (i in 1:nrow(top_batsmen)) {
  cat(top_batsmen$batter[i],   "\t\t",   top_batsmen$strike_rate[i],   "\t\t",   top_batsmen$batting_average[i],
"\t\t", top_batsmen$consistency[i], "\n")
}
```

```
library(fitdistrplus)

# Extract runs scored and wickets taken by the top three batsmen
top_batsmen <- c("Q de Kock", "Shahbaz Ahmed", "MS Wade")
runs_scored <- ipl_ballbyf$total_run[ipl_ballbyf$batter %in% top_batsmen]

# Extract wickets taken by the top three bowlers
top_bowlers <- c("HH Pandya", "Rashid Khan", "GJ Maxwell")
wickets_taken <- ipl_ballbyf$isWicketDelivery[ipl_ballbyf$bowler %in% top_bowlers]

# Plot histograms of runs scored and wickets taken
hist(runs_scored, main = "Distribution of Runs Scored by Top Batsmen", xlab = "Runs", ylab = "Frequency")
hist(wickets_taken, main = "Distribution of Wickets Taken by Top Bowlers", xlab = "Wickets", ylab = "Frequency")

# Add names to the histograms
legend("topright", legend = top_batsmen, fill = "white")
legend("topright", legend = top_bowlers, fill = "white")

# Fit distributions to the data
runs_fit <- fitdist(runs_scored, "norm")
wickets_fit <- fitdist(wickets_taken, "pois")

# Evaluate goodness of fit
runs_gof <- gofstat(runs_fit)
wickets_gof <- gofstat(wickets_fit)
```

```
# Print the goodness of fit statistics
print(runs_gof)
print(wickets_gof)

#-----------------------------------------------------------------
```

#d) Find the relationship between the performance of a player and the salary he gets in the data available to you.

```
library(readxl)
library(dplyr)
library(tidyr)
library(purrr)

ipl_salary <- read_excel("C:\\Users\\monis\\OneDrive\\Desktop\\SCM\\IPL salary 2008_2022.xlsx")

# Check the column names
names(ipl_salary)

# Extract the 'Runs' and 'Final.Price' columns
x <- ipl_salary$Runs
y <- ipl_salary$`Final Price`

# Calculate the correlation between 'Runs' and 'Final.Price'
correlation_runs <- cor(x, y)

# Perform a correlation test between 'Runs' and 'Final.Price'
cor_test_runs <- cor.test(x, y)

# Extract the 'Wkts' and 'Final.Price' columns
```

```
x1 <- ipl_salary$Wkts
y1 <- ipl_salary$`Final Price`

# Calculate the correlation between 'Wkts' and 'Final.Price'
correlation_wkts <- cor(x1, y1)

# Perform a correlation test between 'Wkts' and 'Final.Price'
cor_test_wkts <- cor.test(x1, y1)

# Plot the relationship between 'Runs' and 'Final.Price'
plot(x, y, main = "Runs vs. Final Price", xlab = "Runs", ylab = "Final Price")

# Plot the relationship between 'Wkts' and 'Final.Price'
plot(x1, y1, main = "Wickets vs. Final Price", xlab = "Wickets", ylab = "Final Price")



print("Correlation between Runs and Final.Price:")
print(correlation_runs)
print("")

print("Correlation between Wkts and Final.Price:")
print(correlation_wkts)
print("")
```