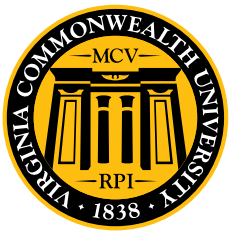

VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A1b: Preliminary preparation and analysis of data – Descriptive Statistics

VIJAYATHITHYAN B B

V01107268

Date of Submission: 18-06-2024

CONTENTS

Analysing Consumption in the state of Bihar using R

INTRODUCTION

The Indian Premier League (IPL) is a major professional Twenty20 cricket league in India, consistently attracting a global audience and commanding significant economic interest. This study delves into the IPL using two datasets: "Cricket_data.csv", containing match details and player performance data, and "Salary 2024.csv", with player salaries for the year 2024.

Through this comprehensive analysis, we aim to gain valuable insights into player performance, salary trends, and potential biases within the IPL. The findings will contribute to a better understanding the league's dynamics and may hold implications for future player recruitment and salary negotiations.

OBJECTIVES

This report aims to analyze Indian Premier League (IPL) data to uncover insights on player performance and salary. This will involve:

1. **Data Exploration and Cleaning:** We will meticulously extract and organize data from "Cricket_data.csv" and "Salary_2024.csv". This process will be done with utmost care, ensuring the data is accurate and reliable. We will organise the data by round, player, and performance metrics (runs, wickets).
2. **Performance Analysis:** Identifying the top three run-scorers and wicket-takers for each IPL round over the past three years.
3. **Statistical Modeling:** We will be fitting appropriate probability distributions to the runs scored and wickets taken by the top performers. This precise modelling will provide us with reliable insights into performance trends.
4. **Performance-Salary Relationship:** Investigating the correlation between a player's past performance and their 2024 salary.
5. **Salary Disparity:** Comparing and analyzing the salary differences between the top 10 batters and wicket-takers over the last three IPL seasons.

BUSINESS SIGNIFICANCE

This analysis of IPL player performance and salaries holds significant value for franchise owners and team management. By examining the distribution of runs and wickets for top performers, franchises can gain insights into player consistency and identify undervalued talent. Additionally, exploring the relationship between performance and salary can inform future player acquisition strategies, potentially leading to more efficient allocation of resources and a competitive edge. Furthermore, highlighting the salary discrepancies between top batters and bowlers provides valuable context for salary negotiations and helps ensure fair compensation across player roles. Ultimately, these findings empower data-driven decision-making, optimizing team rosters and maximizing on-field success.

RESULTS AND INTERPRETATION

A. Arranging the data IPL round-wise and batsmen, ball, runs, and wickets per player per match

Code used:

```
In [20]: grp_data = ipl.groupby(['Season','Innings No', 'Striker',
                                 'Bowler']).agg({'runs_scored': sum,
                                                 'wicket_confirmation':sum}).reset_index()
```

Output:

```
In [21]: grp_data.head()
```

Out[21]:

	Season	Innings No	Striker	Bowler	runs_scored	wicket_confirmation
0	2007/08	1	A Chopra	DP Vijaykumar	1	0
1	2007/08	1	A Chopra	DW Steyn	1	1
2	2007/08	1	A Chopra	GD McGrath	2	0
3	2007/08	1	A Chopra	PJ Sangwan	6	1
4	2007/08	1	A Chopra	RP Singh	9	0

Interpretation: This code snippet restructures the IPL data by grouping it according to season, innings number, batsman, and bowler. It then calculates the total runs scored by each batter and the total wickets taken by each bowler within each specific group.

B. Fitting the most appropriate distribution for runs scored and wickets take by the top three batsmen and bowlers in the three IPL tournaments

Code and Result:

1. ***Grouping the data by Season, Striker and Bowler.***

```
In [25]: player_runs = grp_data.groupby(['Season', 'Striker'])
         ['runs_scored'].sum().reset_index()
         player_wickets = grp_data.groupby(['Season', 'Bowler'])
         ['wicket_confirmation'].sum().reset_index()
```

```
In [27]: player_runs[player_runs['Season'] == '2023'].sort_value
         (by = 'runs_scored', ascending = False)
```

```
In [73]: player_wickets[player_wickets['Season'] == '2023'].sort_values
         (by = 'wicket_confirmation', ascending = False)
```

Out[27]:

	Season	Striker	runs_scored
2423	2023	Shubman Gill	890
2313	2023	F du Plessis	730
2311	2023	DP Conway	672
2433	2023	V Kohli	639
2443	2023	YBK Jaiswal	625
...	...	...	...
2404	2023	RP Meredith	0
2372	2023	Mohsin Khan	0
2307	2023	DG Nalkande	0
2429	2023	TU Deshpande	0
2324	2023	Harshit Rana	0

177 rows × 3 columns

Out[73]:

	Season	Bowler	wicket_confirmation
1750	2023	MM Sharma	31
1755	2023	Mohammed Shami	28
1782	2023	Rashid Khan	28
1797	2023	TU Deshpande	24
1770	2023	PP Chawla	23
...	...	...	...
1776	2023	R Tewatia	0
1709	2023	H Sharma	0
1708	2023	Gurnoor Brar	0
1702	2023	DJ Hooda	0
1673	2023	A Badoni	0

137 rows × 3 columns

2. ***Identifying*** top three run getters and wicket taker in all seasons

In [28]:
```
top_run_getters = player_runs.groupby('Season').apply(lambda x: x.nlargest(3,
                  'runs_scored')).reset_index(drop=True)
bottom_wicket_takers = player_wickets.groupby('Season').apply(lambda x: x.nlargest(3,
                  'wicket_confirmation')).reset_index(drop=True)
print("Top Three Run Getters:")
print(top_run_getters)
print("Top Three Wicket Takers:")
print(bottom_wicket_takers)
```

```
Top Three Run Getters:
     Season          Striker  runs_scored
0   2007/08         SE Marsh          616
1   2007/08        G Gambhir          534
2   2007/08    ST Jayasuriya          514
3      2009        ML Hayden          572
4      2009     AC Gilchrist          495
5      2009   AB de Villiers          465
6   2009/10     SR Tendulkar          618
7   2009/10        JH Kallis          572
8   2009/10         SK Raina          528
9      2011         CH Gayle          608
10     2011          V Kohli          557
11     2011     SR Tendulkar          553
12     2012         CH Gayle          733
13     2012        G Gambhir          590
14     2012         S Dhawan          569
15     2013       MEK Hussey          733
16     2013         CH Gayle          720
```

3. *Creating a consolidated data frame containing Strikers, Bowlers and Seasons.*

```
In [30]: ipl_year_id = pd.DataFrame(columns=["id", "year"])
         ipl_year_id["id"] = ipl["Match id"]
         ipl_year_id["year"] = pd.to_datetime(ipl["Date"], dayfirst=True).dt.year
```

```
In [36]: #create a copy of ipl_bbbc dataframe
         ipl_bbbc= ipl.copy()
```

```
In [37]: ipl_bbbc['year'] = pd.to_datetime(ipl["Date"], dayfirst=True).dt.year
```

```
In [38]: ipl_bbbc[["Match id", "year", "runs_scored","wicket_confirmation"
                  ,"Bowler",'Striker']].head()
```

Out[38]:

	Match id	year	runs_scored	wicket_confirmation	Bowler	Striker
0	335982	2008	0	0	P Kumar	SC Ganguly
1	335982	2008	0	0	P Kumar	BB McCullum
2	335982	2008	0	0	P Kumar	BB McCullum
3	335982	2008	0	0	P Kumar	BB McCullum
4	335982	2008	0	0	P Kumar	BB McCullum

---

4. *Finding the most appropriate distribution for runs scored and wickets take by the top three batsmen and bowlers in the three IPL tournaments*

Here Kolmogorov-Smirnov test is used. It is a non-parametric statistical test used to compare the probability distributions of two samples. It is a versatile tool used in various data analysis scenarios, and here is how it's applied in Python:

- **Goodness-of-fit test:** The KS test, a powerful tool, can be used to assess with precision how well a theoretical distribution (e.g., normal, Poisson) fits a given dataset. This helps determine with confidence if the data likely originated from that specific distribution.
- **Comparing two samples:** The KS test can compare the distributions of two independent samples to see if they statistically differ. This is useful in tasks like comparing the batting performances of two players or the bowling strategies of two teams.

In [40]:
```
import scipy.stats as st

def get_best_distribution(data):
    dist_names = ['alpha','beta','betaprime','burr12','crystalball',
                  'dgamma','dweibull','erlang','exponnorm','f','fatiguelife',
                  'gamma','gengamma','gumbel_l','johnsonsb','kappa4',
                  'lognorm','nct','norm','norminvgauss','powernorm','rice',
                  'recipinvgauss','t','trapz','truncnorm']
    dist_results = []
    params = {}
    for dist_name in dist_names:
        dist = getattr(st, dist_name)
        param = dist.fit(data)
        params[dist_name] = param
        # Applying the Kolmogorov-Smirnov test
        D, p = st.kstest(data, dist_name, args=param)
        print("p value for "+dist_name+" = "+str(p))
        dist_results.append((dist_name, p))
    # select the best fitted distribution
    best_dist, best_p = (max(dist_results, key=lambda item: item[1]))
    # store the name of the best fit and its p value
    print("\nBest fitting distribution: "+str(best_dist))
    print("Best p value: "+ str(best_p))
    print("Parameters for the best fit: "+ str(params[best_dist]))
    return best_dist, best_p, params[best_dist]
```

In [41]:
```
total_run_each_year = ipl_bbbc.groupby(["year", "Striker"])["runs_scored"].sum().reset_index()
```

---

5. *Listing the top three Strikers and Bowlers in last three years*

In [44]:
```
list_top_batsman_last_three_year = {}
for i in total_run_each_year["year"].unique()[:3]:
    list_top_batsman_last_three_year[i] = total_run_each_year
    [total_run_each_year.year == i][:3]["Striker"].unique().tolist()
```

In [45]:
```
list_top_batsman_last_three_year
```

Out[45]:
```
{2024: ['RD Gaikwad', 'V Kohli', 'B Sai Sudharsan'],
 2023: ['Shubman Gill', 'F du Plessis', 'DP Conway'],
 2022: ['JC Buttler', 'KL Rahul', 'Q de Kock']}
```

In [50]:
```
list_top_bowler_last_three_year = {}
for i in total_wicket_each_year["year"].unique()[:3]:
    list_top_bowler_last_three_year[i] = total_wicket_each_year
    [total_wicket_each_year.year == i][:3]["Bowler"].unique().tolist()
list_top_bowler_last_three_year
```

Out[50]:
```
{2024: ['HV Patel', 'Mukesh Kumar', 'Arshdeep Singh'],
 2023: ['MM Sharma', 'Mohammed Shami', 'Rashid Khan'],
 2022: ['YS Chahal', 'PWH de Silva', 'K Rabada']}
```

6. ***Fitting the most appropriate distribution for wickets take by Arshdeep Singh. The same code can be used to find the most appropriate distribution for runs scored or wickets taken by a Striker or a Bowler respectively.***

In [70]:
```
import warnings
warnings.filterwarnings('ignore')

wickets = ipl_bbbc.groupby(['Bowler','Match id'])[['wicket_confirmation']].sum().reset_index()

# Choose the bowler you want to analyze (replace with desired bowler name)
chosen_bowler = "Arshdeep Singh"  # Replace with your chosen bowler's name

print("**********")
print("Best fit distribution for wickets taken by:", chosen_bowler)
get_best_distribution(wickets[wickets["Bowler"] == chosen_bowler]["wicket_confirmation"])
print("\n\n")
```

```
**********
Best fit distribution for wickets taken by: Arshdeep Singh
p value for alpha = 0.002547644307209551
p value for beta = 3.7725133611153275e-15
p value for betaprime = 5.062381659741898e-22
p value for burr12 = 4.603956720503075e-14
p value for crystalball = 0.0002501762149918564
p value for dgamma = 0.00028566200697101806
p value for dweibull = 0.0016211491850549598
p value for erlang = 2.269289539862191e-12
p value for exponnorm = 0.0019097947631203649
p value for f = 0.000227258408802241
p value for fatiguelife = 2.169103029961132e-15
p value for gamma = 6.618486511618167e-29
p value for gengamma = 5.948936850168967e-23
p value for gumbel_l = 0.00026864389982599567
p value for johnsonsb = 5.472387372640376e-24
p value for kappa4 = 8.181970339328129e-12
p value for lognorm = 1.9909678840157557e-12
p value for nct = 0.0014257070102449143
p value for norm = 0.00025017539197677184
p value for norminvgauss = 0.0001290021448063343
p value for powernorm = 0.00047137775975730436
p value for rice = 0.00047472774494963083
p value for recipinvgauss = 1.9623061606588953e-10
p value for t = 0.004473243416688644
p value for trapz = 1.1911079182772876e-29
p value for truncnorm = 0.00034221379785853717

Best fitting distribution: t
Best p value: 0.004473243416688644
Parameters for the best fit: (4.822497644715119, 1.1162819391895469, 0.9153269129308039)
```

Interpretation: Thus the fitting distribution for wickets taken by Arshdeep Singh is the T-Test.

C. Finding the relationship between a player's performance and the salary he gets as per the data.

The names of players are in different format in database. Thus, it is required to regularize the names to proceed with further analysis.

Code and Results:

```
In [57]: from fuzzywuzzy import process

         # Convert to DataFrame
         df_salary = ipl_salary.copy()
         df_runs = R2024.copy()

         # Function to match names
         def match_names(name, names_list):
             match, score = process.extractOne(name, names_list)
             return match if score >= 80 else None  # Use a threshold score of 80

         # Create a new column in df_salary with matched names from df_runs
         df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x, df_runs['Striker'].tolist()))

         # Merge the DataFrames on the matched names
         df_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Striker')
```

```
In [74]: df_merged
```

Out[74]:

	Player	Salary	Rs	international	iconic	Matched_Player	year	Striker	runs_scored
0	Abhishek Porel	20 lakh	20	0	NaN	Abishek Porel	2024	Abishek Porel	202
1	Anrich Nortje	6.5 crore	650	1	NaN	A Nortje	2024	A Nortje	4
2	Axar Patel	9 crore	900	0	NaN	AR Patel	2024	AR Patel	149
3	David Warner	6.25 crore	625	1	NaN	TH David	2024	TH David	217
4	David Miller	3 crore	300	1	NaN	TH David	2024	TH David	217
...	...	...	...	...	...	...	...	...	...
106	Anmolpreet Singh	20 lakh	20	0	NaN	Anmolpreet Singh	2024	Anmolpreet Singh	0
107	Heinrich Klaasen	5.25 crore	525	1	NaN	H Klaasen	2024	H Klaasen	295
108	Marco Jansen	4.2 crore	420	1	NaN	M Jansen	2024	M Jansen	1
109	Rahul Tripathi	8.5 crore	850	0	NaN	RA Tripathi	2024	RA Tripathi	31
110	Washington Sundar	8.75 crore	875	0	NaN	Washington Sundar	2024	Washington Sundar	0

111 rows × 9 columns

Now, finding the relationship between a player's performance and the salary he gets as per the data is possible using the following codes,

```
In [59]: # Calculate the correlation
         correlation = df_merged['Rs'].corr(df_merged['runs_scored'])

         print("Correlation between Salary and Runs:", correlation)
```

```
Correlation between Salary and Runs: 0.3061248376582168
```

Interpretation: The correlation coefficient is a statistical measure that indicates the strength and direction of a linear relationship between two variables. It ranges from -1 to 1. A positive correlation (+1) indicates that as one variable increases, the other also tends to increase. A negative correlation (-1) indicates that as one variable increases, the other tends to decrease. A correlation of 0 indicates no linear relationship between the variables. This correlation is for a Striker's performance and his remuneration. Accordingly, the correlation of 0.30612 indicate that there is a weak positive correlation. Salary is not the sole consideration of a player's performance, but other factors, such as, player's experience, reputation, past performance etc. also determines their current performance.

D. Significant difference between the Salaries of the top 10 batsmen and top wicket-taking bowlers over the last three years.

The names of players are in different format in database. Thus, it is required to regularize the names to proceed with further analysis. Two data frames are created namely, df_merged and df_merged1 that contain strikers' matched names and bowlers' matched names.

Code and Results:

In [140]:
```
from fuzzywuzzy import process

# Convert to DataFrame
df_salary = ipl_salary.copy()
df_runs = R2024.copy()
df_wickets = W2024.copy()

# Function to match names
def match_names(name, names_list):
    match, score = process.extractOne(name, names_list)
    return match if score >= 80 else None  # Use a threshold score of 80

# Create a new column in df_salary with matched names from df_runs
df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x, df_runs['Striker'].tolist()))


# Merge the DataFrames on the matched names
df_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Striker')
```

In [141]:
```
# Convert to DataFrame
df_salary = ipl_salary.copy()
df_runs = R2024.copy()
df_wickets = W2024.copy()

# Function to match names
def match_names(name, names_list):
    match, score = process.extractOne(name, names_list)
    return match if score >= 80 else None  # Use a threshold score of 80

# Create a new column in df_salary with matched names from df_runs
df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x, df_wickets['Bowler'].tolist()))


# Merge the DataFrames on the matched names
df_merged1 = pd.merge(df_salary, df_wickets, left_on='Matched_Player', right_on='Bowler')
```

Once these two data frames are created we can now find the significant difference between the Salaries of the top 10 batsmen and top wicket-taking bowlers over the last three years using the following code.

In [166]:
```
from fuzzywuzzy import process  # Assuming fuzzywuzzy is already imported

# Filter last three years' data (assuming 'year' column exists)
last_three_years_data = merged_df[merged_df['year_x'].isin([2022, 2023, 2024])]

# Function to get top players
def get_top_players(data, n, role):
  if role == "Matched_Player":
    top_players = last_three_years_data.groupby('Matched_Player')['runs_scored'].sum().sort_values(ascending=False).head(n)
  elif role == "Bowler":
    top_players = last_three_years_data.groupby('Bowler')['wicket_confirmation'].sum().sort_values(ascending=False).head(n)
  else:
    raise ValueError("Invalid role. Choose 'Matched_Player' or 'Bowler'")
  return top_players.index.tolist()

# Get top 10 batsmen and bowlers
top_10_batsmen = get_top_players(last_three_years_data, 10, "Matched_Player")
top_10_bowlers = get_top_players(last_three_years_data, 10, "Bowler")

# Calculate average salary (assuming 'Rs_x' column holds salary information)
avg_salary_batsmen = last_three_years_data[last_three_years_data['Matched_Player'].isin(top_10_batsmen)]['Rs_x'].mean()
avg_salary_bowlers = last_three_years_data[last_three_years_data['Bowler'].isin(top_10_bowlers)]['Rs_x'].mean()

# Calculate difference
difference = avg_salary_batsmen - avg_salary_bowlers

# Print result
print("Significant Difference in Salary (Batsmen - Bowlers):", difference)
```

```
Significant Difference in Salary (Batsmen - Bowlers): 221.42676767676772
```

Interpretation: The output is a numerical value representing the difference between the average salaries of the top 10 batsmen and the top 10 bowlers over the last three years, based on the 'Rs_x' column assumed to hold salary information. For instance, if the output is 221.42, it indicates that, on average, the top 10 batsmen earn Rs. 221.42 more than the top 10 bowlers in the last three years, according to the data.