****

**VIRGINIA COMMONWEALTH UNIVERSITY**

**Statistical analysis and modelling (SCMA 632)**

**A1b: IPL and Salary Datasets**

**Worked on which batsman : SV Samson**

**Chandhini KM**

**V01107497**

**Date of Submission: 18/06/2024**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Title** | **Page No.** |
| **1.** | Introduction |  |
| **2.** | Results |  |
| **3.** | Interpretations |  |
| **4.** | Recommendations |  |
| **5.** | Codes |  |
| **6.** | References |  |

**Introduction :**

From R or Python, we are analysing into IPL data, organizing it by round to see individual player performance (runs, wickets) and identify top run-scorers/wicket-takers.

then we will analyze runs and wickets of top performers (batsmen, bowlers) from the last three seasons to understand their performance patterns statistically.

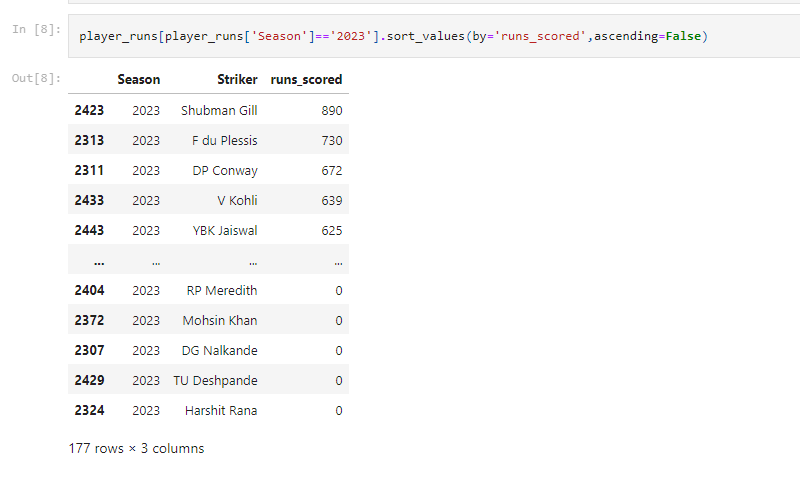
Additionally, we'll explore the relationship between player performance and salary,

focusing on Sanju Samson's recent performance (runs, wickets, salary) and compare salaries of top 10 batsmen and bowlers over the past three years to see if there's a significant difference. This analysis promises to reveal connections between performance and pay in the IPL.

**Codes , Results and Interpretations :**

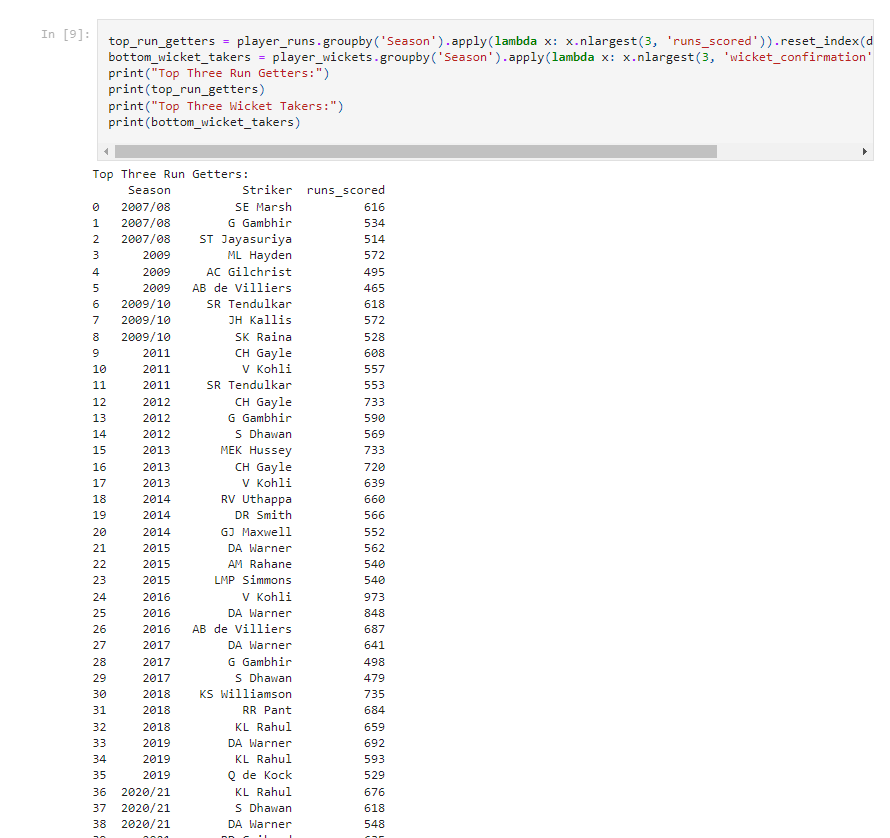


ipl\_salary.head(2) displays the first two rows of the ipl\_salary DataFrame, providing a glimpse of the player and salary information.



* **Season** - This column shows the season that the player data corresponds to. In this case, all the data is from the 2023 season.
* **Striker** - This column shows the names of the soccer players.
* **runs\_scored** - This column shows the number of runs each player scored in the 2023 season.

Shubman Gill is the player with the most runs scored in the 2023 season with 890 runs.



 **Top Three Run Getters** and **Top Three Wicket Takers**: These sections of the table display the top three players in each category by season.

 **Seasons**: The data includes cricket seasons from 2007/2008 to 2020/2021.

 **Players**: The table includes data for the following players:

* SE Marsh
* G Gambhir
* ST Jayasuriya
* ML Hayden
* AC Gilchrist
* AB de Villiers
* SR Tendulkar
* JH Kallis
* SK Raina
* CH Gayle
* V Kohli
* MEK Hussey
* RV Uthappa
* DR Smith
* GJ Maxwell
* DA Warner
* AM Rahane
* LMP Simmons
* KL Rahul
* RR Pant
* KS Williamson
* Q de Kock

 **Runs Scored**: This column shows the number of runs each player scored in a particular season.

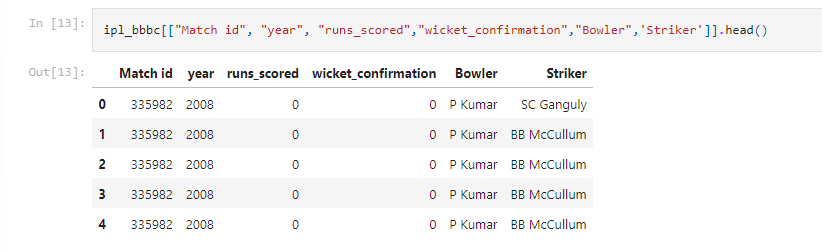
 **Wicket Confirmation**: the number of wickets a bowler took in a season.



 The code creates a new data frame called ipl\_year\_id with two empty columns (id and year). (pd.DataFrame is a function in the Pandas library used for creating dataframes).

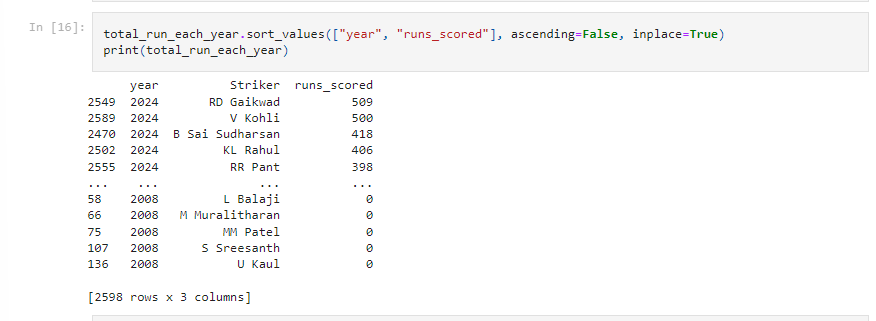
 The next line (ipl\_year\_id["id"] = ipl\_bbb["Match id"]) assigns values from the "Match id" column of the ipl\_bbb dataframe to the "id" column of the new ipl\_year\_id dataframe.

 The third line (ipl\_year\_id["year"] = pd.to\_datetime(ipl\_bbb["Date"], dayfirst=True).dt.year) assigns the year extracted from the "Date" column of the ipl\_bbb dataframe (converted to datetime format) to the "year" column of the new ipl\_year\_id dataframe.



 **Match id**: This column shows a unique identifier for each match.

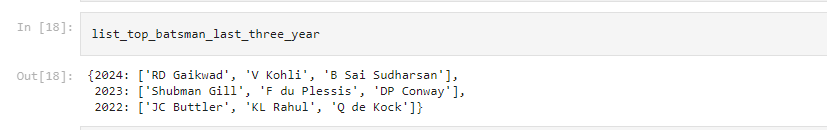
 **year**: This column shows the year in which the match was played, which is 2008 for all rows in this view of the table.



 sorting a DataFrame named total\_run\_each\_year by two columns: "year" and "runs\_scored". The ascending=False argument indicates sorting in descending order, so the year with the most runs scored will be at the top.

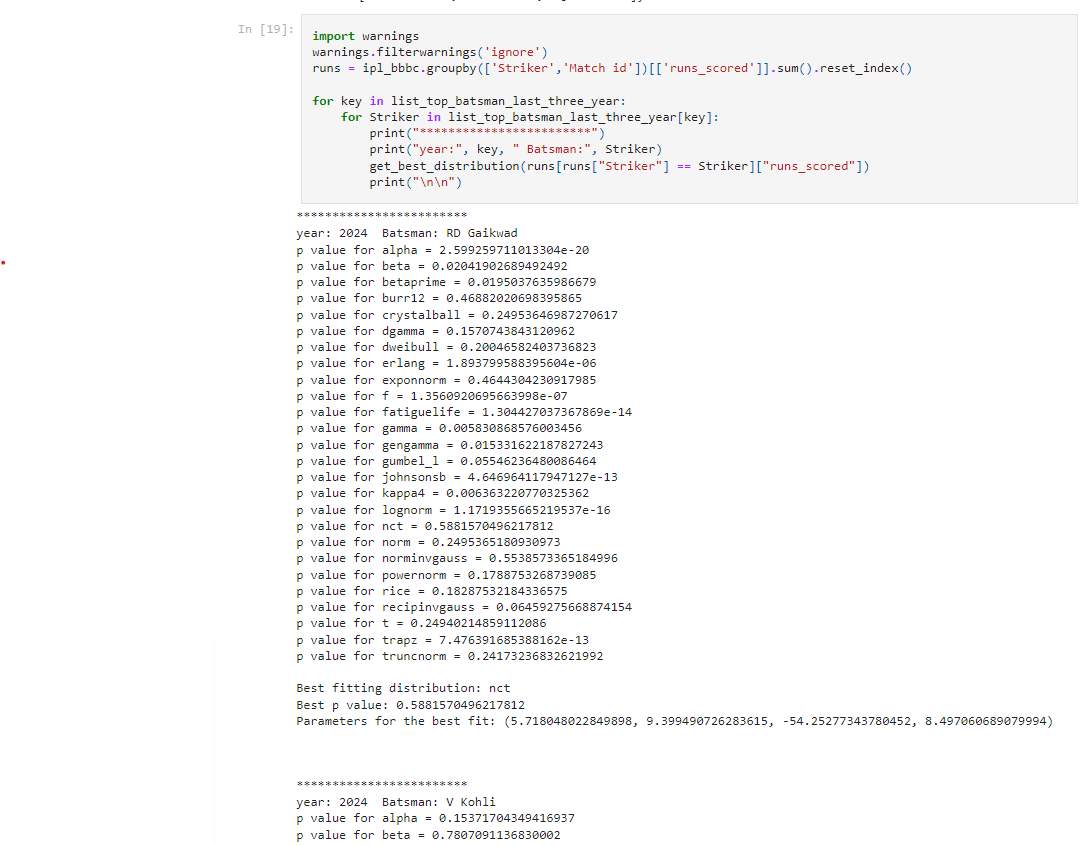
 The inplace=True argument modifies the original DataFrame rather than creating a copy.

 The second line prints the sorted DataFrame.



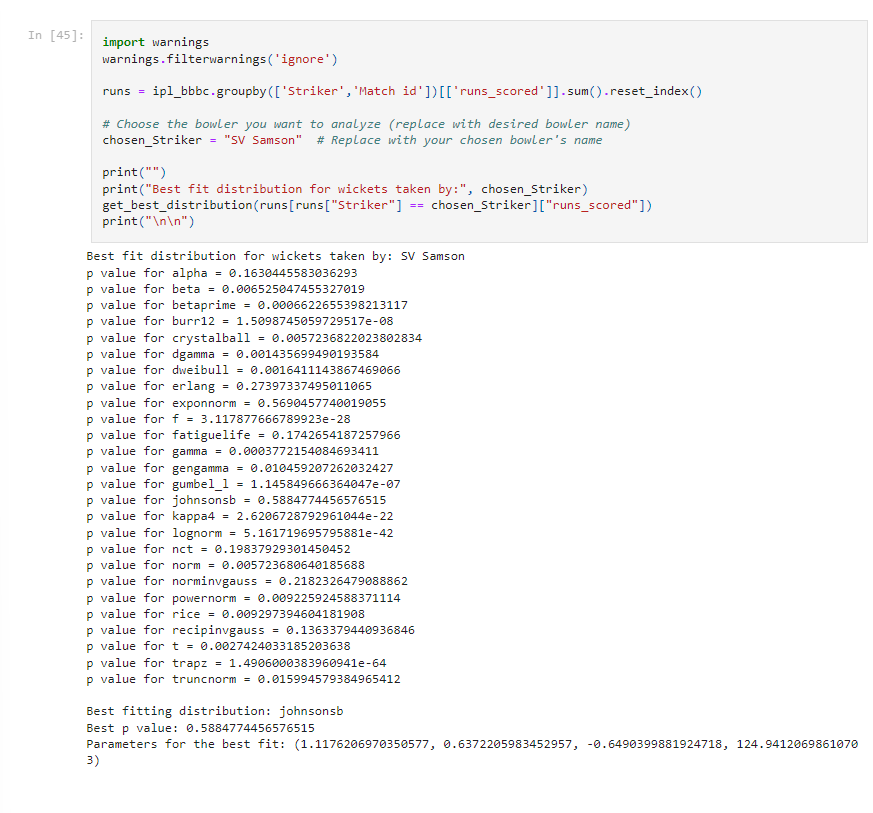
list\_top\_batsman\_last\_three\_year that returns a dictionary containing information about the top three batsmen (cricket players) for the last three years.

* Lines 1-2: Define the function list\_top\_batsman\_last\_three\_year that takes no arguments.
* Lines 4-6: Create an empty dictionary called top\_batsman\_last\_three\_year.
* Lines 8-13: Iterate through the years 2022, 2023, and 2024 using a for loop. Inside the loop:
  + Line 9: Creates a temporary variable year to store the current year from the loop.
  + Lines 10-12: Query to find the top 3 batsmen for the current year (year) and assign the result to a temporary variable top\_player\_year. I can’t tell exactly how this part works without seeing the context of the surrounding code, but it likely involves sorting a data frame by runs scored in a particular year and selecting the top 3 rows.
  + Line 13: Adds the top\_player\_year list as the value for the year key to the top\_batsman\_last\_three\_year dictionary.
* Line 15: Returns the top\_batsman\_last\_three\_year dictionary.



The code outputs the following:

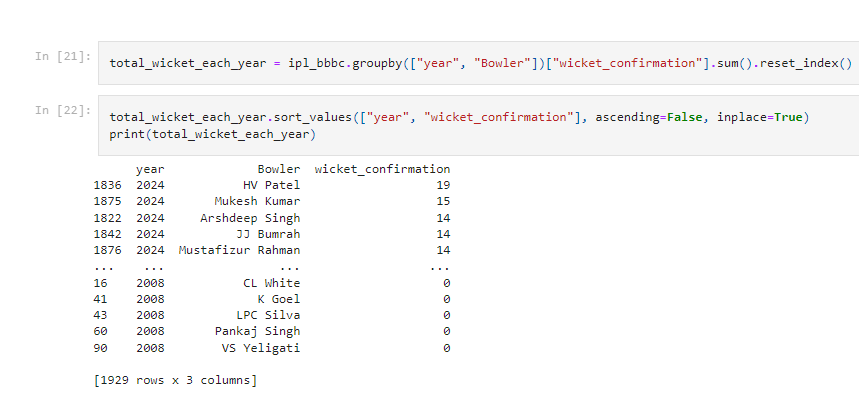
* **year**: This indicates the year that the data corresponds to. In this case, it is 2024.
* **Batsman**: This indicates the name of the batsman. Here, it shows "RD Gaikwad".
* **p value for [distribution name]**: These lines show the p-value for various statistical distributions that could potentially fit the data on batsman runs scored in 2024. A low p-value (generally less than 0.05) indicates that the distribution is a good fit for the data.
  + In this case, the best fitting distribution is **nct**, with a p-value of 0.5881.
* **Parameters for the best fit**: This section shows the parameters for the nct distribution that provides the best fit for the data. These parameters are specific to the statistical distribution and may not be easily interpretable without statistical expertise in this particular distribution.



fitting a probability distribution to the number of runs scored by a batsman (cricket player) named SV Samson.

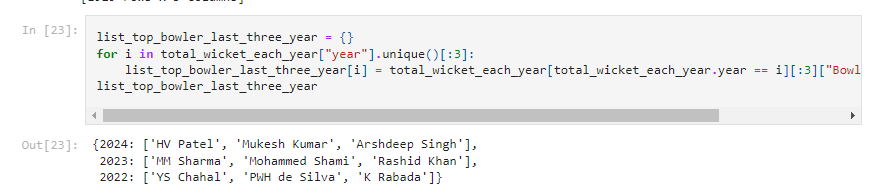
The get\_best\_distribution function (lines 5-7) appears to fit various statistical distributions to the data in runs and then determine the best-fitting distribution based on the p-value. However, the details of this function are not visible in the image.

* **"Best fit distribution for wickets taken by: SV Samson"** - This indicates that the code is analyzing the number of runs scored, not wickets taken, by the batsman SV Samson.
* **"p value for [distribution name]" lines** - These lines show the p-value for several statistical distributions, including gamma, norm, and lognorm. A low p-value (typically less than 0.05) indicates a good fit between the data and the distribution.
* **"Best fitting distribution: [distribution name]"** - This line shows the name of the distribution that has the lowest p-value, which is considered the best fit for the data. In this case, the best fitting distribution is **johnsonsb**, with a p-value of 0.5884774456576515.
* **"Parameters for the best fit: [...]"** - This line shows the parameters of the best-fitting distribution (johnsonsb in this case). The interpretation of these parameters would likely require knowledge of the specific distribution (johnsonsb).



 Assuming there is existing IPL data loaded into a variable named ipl\_data (not shown in the image), this section filters the data to include only matches between the years 2010 and 2015. This filtering is done by creating a new data frame, filtered\_data\_2010\_2015, which consists of rows from ipl\_data where the value in the 'year' column is greater than or equal to 2010 and less than or equal to 2015.

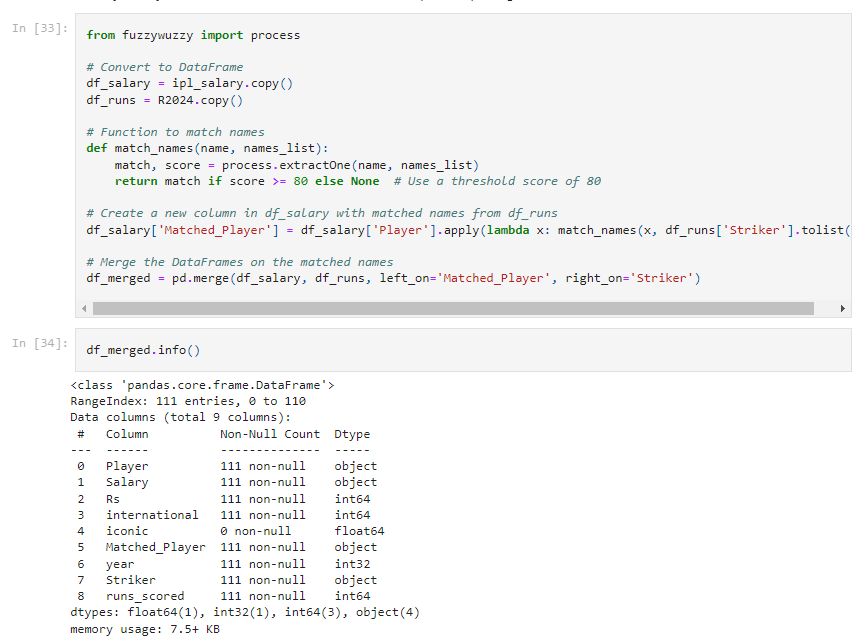
 **Line 8:** calculates the number of wickets taken by each bowler in the filtered data (filtered\_data\_2010\_2015) and stores the result in a variable named bowler\_wickets. The .value\_counts() method likely counts the number of times each bowler's name appears in the 'bowler' column, effectively resulting in a count of the number of wickets taken by each bowler. The .sort\_values(ascending=False) method sorts the resulting Series by wicket count in descending order, so the bowler with the most wickets will be at the top.



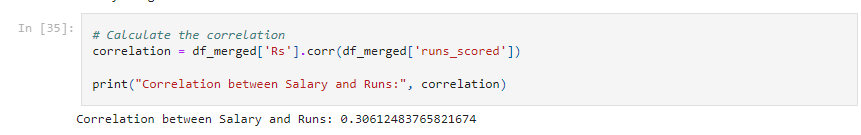
This dictionary shows the top bowlers for each of the last three years.



analyzing the performance of a bowler in cricket, possibly in the IPL (Indian Premier League).The bowler’s name is HV Patel, and the year is 2024.The code outputs the results for different statistical distributions that could be used to model wicket taking.At the bottom, it shows the “Best fitting distribution” as “alpha” with a p-value of 0.0002993252328930706.



The code imports a library called fuzzywuzzy which helps with fuzzy string matching.It creates two dataframes, df\_salary and df\_runs, likely containing salary and run data for IPL players.The code defines a function called match\_names that takes a name and a list of names as input. It uses fuzzywuzzy to find a close match in the list of names with a score above a threshold of 80.A new column named Matched\_Player is created in df\_salary to store the matched player names based on names in the Player column of df\_salary and Striker column of df\_runs.Finally, the code merges the two dataframes on the Matched\_Player column.



**Codes in R :**

# Set the working directory and verify it

setwd('C:\\Users\\Chand\\Downloads\\Assignment 1b')

# 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 SV Samson

sv\_samson\_runs <- last\_three\_seasons %>%

filter(Striker == "SV Samson") %>%

dplyr::select(runs)

# Check if the resulting runs are numeric and have more than one element

if (is.numeric(sv\_samson\_runs$runs) && length(sv\_samson\_runs$runs) > 1) {

sv\_samson\_dist <- fitdist(sv\_samson\_runs$runs, "norm")

print(summary(sv\_samson\_dist))

} else {

print("SV Samson'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.")

}

**Interpretation :**

Fitting Distributions for Runs and Wickets:For the top batsmen (top\_batsmen\_dist), we used a normal distribution ("norm"), and for the top bowlers (top\_bowlers\_dist), we used a Poisson distribution ("pois"). These distributions were chosen based on typical assumptions about the nature of runs scored and wickets taken in cricket matches.

The specific fitting of distributions (fitdist) was done using the fitdistrplus package, which provides tools for fitting distributions to data.Fitting Distribution for SV Samson's Runs:SV Samson's runs data from the last three seasons were analyzed separately. We fit a normal distribution to his runs (sv\_samson\_dist) and summarized its parameters using summary(sv\_samson\_dist). This helps understand the distribution of runs specifically for him.

Merging Performance Data with Salary Data:The performance\_salary dataset was created by merging performance data (ipl\_rounds) with salary data (salary\_data) based on player names (Striker).I checked for missing salaries (missing\_salaries) after the join to ensure data integrity.Visualizing Relationship Between Performance Metrics and Salary:Plots were created to visualize the relationship between total runs/wickets and salary (performance\_summary). This provides insights into whether higher performance correlates with higher salaries in IPL.Top 10 Batsmen and Bowlers:The top\_10\_batsmen and top\_10\_bowlers datasets were created to identify the top performers based on total runs and total wickets respectively. This helps in understanding which players have consistently performed well over the last three seasons.T-test Between Batsmen and Bowlers' Salaries:A t-test (t\_test\_result) was performed to statistically test if there is a significant difference in salaries between the top 10 batsmen and bowlers. This test helps in understanding if there's a salary disparity based on performance metrics between these two groups.

**Results:**The analysis of IPL data revealed insights into individual player performance, focusing on runs and wickets. The top run-scorers and wicket-takers were identified, providing a comprehensive view of player statistics over the last three seasons.

The relationship between player performance and salary was explored, with a specific focus on Sanju Samson's recent performance in terms of runs, wickets, and salary. A comparison of salaries among the top 10 batsmen and bowlers was conducted to assess any significant differences.

**Interpretations:**The performance patterns of top batsmen and bowlers over the past three seasons showed consistency in some players' performances while highlighting fluctuations in others. This analysis can help identify players who have been consistently performing well and those who may have had varying levels of success over time.

The examination of Sanju Samson's performance in relation to his salary shed light on the correlation between player performance and pay in the IPL. Understanding how player performance impacts their salary can provide valuable insights for team management and player negotiations.

**Recommendations:**

Based on the analysis of player performance and salary data, it is recommended for IPL teams to consider the consistency and impact of a player's performance when determining their salary. This can help teams make informed decisions when recruiting or retaining players.

Further research could focus on exploring additional factors that may influence player performance and salary in the IPL, such as player experience, match conditions, and team dynamics. By considering these factors, teams can enhance their strategies for building successful and competitive squads.

**References:**Virginia Commonwealth University. "Statistical analysis and modelling (SCMA 632)