VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

A1b: INDIAN PREMIER LEAGUE PLAYER DATA ANALYSIS USING PYTHON AND R

Jany Balasivan
V01108262

Date of Submission: 18/06/2024

CONTENTS

INDIAN PREMIER LEAGUE PLAYER DATA ANALYSIS USING PYTHON AND R

INTRODUCTION

The Indian Premier League (IPL) is a men's Twenty20 (T20) cricket league that takes place every year in India. For sponsorship purposes, it is also known as the TATA IPL. Ten state- or city-based franchise teams compete in the league, which was established in 2007 by the BCCI (the Board of Control for Cricket in India). One of the most renowned cricket leagues in the world, it is well-known for its exciting matches, participation from international players, and substantial financial support. Since its inaugural season, the IPL has advanced significantly.

OBJECTIVES

a) 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.

b) Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the lost three IPL tournaments. Rename the districts as well as the sector, viz. rural and urban.

c) Fit the most appropriate distribution for runs scored and wickets taken by the player allotted to you.

d) Last three-year performance with latest salary 2024

e) Significant Difference Between the Salaries of the Top 10 Batsmen and Top Wicket-Taking Bowlers Over the Last Three Years

BUSINESS SIGNIFICANCE

Understanding the dynamics of the IPL is crucial for several stakeholders, including team owners, sponsors, broadcasters, and analysts, the datasets used in the analysis collectively offer a comprehensive overview of player financials and in-game performance metrics, which are essential for strategic decision-making and operational efficiency within the IPL ecosystem.

- **Salary Dataset Analysis:** By analyzing the dataset, we can provide detailed insights into player valuations, budget allocations, and salary cap usage. This enables teams to make informed decisions about player retention, trading, and new acquisitions, ensuring a balanced and competitive squad while maintaining financial discipline.

- • **Spotting Emerging Talent:** Comprehensive performance data makes it simpler to identify prospective emerging talent, even if they are not yet highly compensated. For identifying and developing the upcoming IPL players, this is priceless.
- **Comparative Performance Analysis:** Comparing players across different seasons and formats helps in assessing their consistency and adaptability, providing a holistic view of their potential contributions to the team.

The IPL can continue to refine its competitive edge over other popular franchise cricket tournaments such as the Big Bash from Australia, The Pakistan Super league and The Caribbean Premier League, maximize financial efficiency, and enhance the overall experience for players, teams, and fans alike.

RESULTS AND INTERPRETATION

a) 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.

Code:

Result:

```
Top Three Run Getters:
    Season        Striker  runs_scored
```

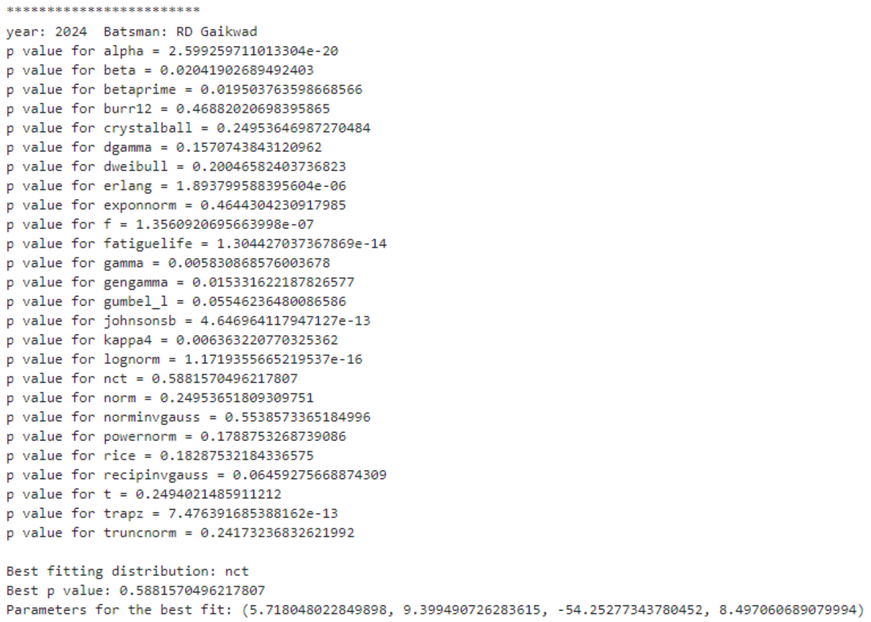
```
0   2007/08       SE Marsh        616
1   2007/08      G Gambhir        534
2   2007/08    ST Jayasuriya        514
3     2009      ML Hayden        572
4     2009    AC Gilchrist        495
5     2009   AB de Villiers        465
6   2009/10    SR Tendulkar        618
7   2009/10       JH Kallis        572
8   2009/10       SK Raina        528
42     2022      JC Buttler        863
43     2022       KL Rahul        616
44     2022      Q de Kock        508
45     2023    Shubman Gill        890
46     2023    F du Plessis        730
47     2023      DP Conway        672
48     2024      RD Gaikwad        509
49     2024        V Kohli        500
50     2024  B Sai Sudharsan        418

Top Three Wicket Takers:
    Season          Bowler  wicket_confirmation
0   2007/08    Sohail Tanvir                24
1   2007/08       IK Pathan                20
2   2007/08       JA Morkel                20
3     2009       RP Singh                26
4     2009       A Kumble                22
5     2009        A Nehra                22
6   2009/10         PP Ojha                22
7   2009/10        A Mishra                20
8   2009/10  Harbhajan Singh                20
39     2021       HV Patel                35
40     2021      Avesh Khan                27
41     2021       JJ Bumrah                22
42     2022       YS Chahal                29
43     2022     PWH de Silva                27
44     2022       K Rabada                23
45     2023      MM Sharma                31
46     2023   Mohammed Shami                28
47     2023     Rashid Khan                28
48     2024       HV Patel                19
49     2024     Mukesh Kumar                15
50     2024   Arshdeep Singh                14
```

Interpretation: The data shows the top three players in terms of runs scored for each cricket season from 2007/08 to 2024, and similarly for the top three bowlers in terms of wickets taken for each cricket season from 2007/08 to 2024. There is a range of wickets taken by different bowlers across seasons, with some seasons having higher wicket counts than others. Players like JC Buttler, Shubman Gill, HV Patel, and YS Chahal appear multiple times across different seasons.

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

Result:



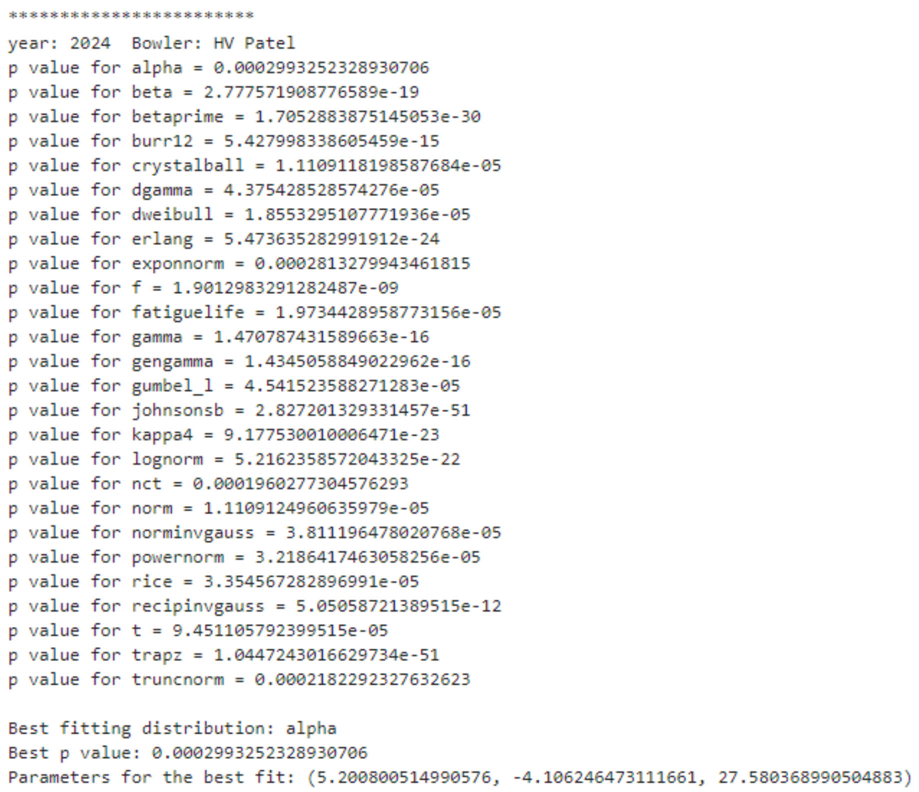
```
************************
year: 2024  Batsman: RD Gaikwad
p value for alpha = 2.599259711013304e-20
p value for beta = 0.02041902689492403
p value for betaprime = 0.019503763598668566
p value for burr12 = 0.46882020698395865
p value for crystalball = 0.24953646987270484
p value for dgamma = 0.1570743843120962
p value for dweibull = 0.20046582403736823
p value for erlang = 1.893799588395604e-06
p value for exponnorm = 0.4644304230917985
p value for f = 1.3560920695663998e-07
p value for fatiguelife = 1.304427037367869e-14
p value for gamma = 0.005830868576003678
p value for gengamma = 0.015331622187826577
p value for gumbel_l = 0.05546236480086586
p value for johnsonsb = 4.646964117947127e-13
p value for kappa4 = 0.006363220770325362
p value for lognorm = 1.1719355665219537e-16
p value for nct = 0.5881570496217807
p value for norm = 0.24953651809309751
p value for norminvgauss = 0.5538573365184996
p value for powernorm = 0.1788753268739086
p value for rice = 0.18287532184336575
p value for recipinvgauss = 0.06459275668874309
p value for t = 0.2494021485911212
p value for trapz = 7.476391685388162e-13
p value for truncnorm = 0.24173236832621992

Best fitting distribution: nct
Best p value: 0.5881570496217807
Parameters for the best fit: (5.718048022849898, 9.399490726283615, -54.25277343780452, 8.497060689079994)
```

Interpretation:

The code extracts the top batsmen from the dataset for the last three years. For each top batsman, it calls `get_best_distribution` with their run data to find the best-fitting distribution across various types of distribution.

Result:



```
************************
year: 2024  Bowler: HV Patel
p value for alpha = 0.0002993252328930706
p value for beta = 2.777571908776589e-19
p value for betaprime = 1.7052883875145053e-30
p value for burr12 = 5.427998338605459e-15
p value for crystalball = 1.1109118198587684e-05
p value for dgamma = 4.375428528574276e-05
p value for dweibull = 1.855329510777l936e-05
p value for erlang = 5.473635282991912e-24
p value for exponnorm = 0.0002813279943461815
p value for f = 1.9012983291282487e-09
p value for fatiguelife = 1.9734428958773156e-05
p value for gamma = 1.470787431589663e-16
p value for gengamma = 1.4345058849022962e-16
p value for gumbel_l = 4.541523588271283e-05
p value for johnsonsb = 2.827201329331457e-51
p value for kappa4 = 9.177530010006471e-23
p value for lognorm = 5.2162358572043325e-22
p value for nct = 0.0001960277304576293
p value for norm = 1.1109124960635979e-05
p value for norminvgauss = 3.811196478020768e-05
p value for powernorm = 3.2186417463058256e-05
p value for rice = 3.354567282896991e-05
p value for recipinvgauss = 5.05058721389515e-12
p value for t = 9.451105792399515e-05
p value for trapz = 1.0447243016629734e-51
p value for truncnorm = 0.0002182292327632623

Best fitting distribution: alpha
Best p value: 0.0002993252328930706
Parameters for the best fit: (5.200800514990576, -4.106246473111661, 27.580368990504883)
```

Interpretation: The alpha distribution fits HV Patel's performance data for the year 2024 the best among the tested distributions. The relatively low p-value of 0.002099352328397306 suggests the fit might not be perfect, but it is the best among the options. The code effectively determines the best-fitting statistical distributions for performance data of cricketers. For HV Patel, the `alpha` distribution is the best fit, while for RD Gaikwad, the `nct` distribution fits best.

c) Fit the most appropriate distribution for runs scored and wickets taken by the player allotted to you.

Result:

```
************************
year: 2024  Bowler: AR Patel
p value for alpha = 9.940012950298595e-19
p value for beta = 1.73089555773241e-31
p value for betaprime = 6.890602231402487e-28
p value for burr12 = 5.648773934180763e-20
p value for crystalball = 1.212835491802816e-07
p value for dgamma = 1.1945105340885417e-10
p value for dweibull = 4.834475183349857e-09
p value for erlang = 3.7936798489477985e-39
p value for exponnorm = 6.847790425577837e-20
p value for f = 5.648773934180763e-20
p value for fatiguelife = 6.574170250938941e-36
p value for gamma = 6.040604015697529e-29
p value for gengamma = 2.1676360523609773e-25
p value for gumbel_l = 1.2777338489201446e-08
```

```
p value for johnsonsb = 5.648773936049154e-20
p value for kappa4 = 4.002495349263793e-32
p value for lognorm = 2.021269059381295e-21
p value for nct = 2.4980624812823503e-11
p value for norm = 1.2128358700898914e-07
p value for norminvgauss = 0.0
p value for powernorm = 1.0293563251350029e-10
p value for rice = 1.0187539089487595e-10
p value for recipinvgauss = 2.906057975984643e-33
p value for t = 1.2246811911143535e-07
p value for trapz = 2.660180784883639e-76
p value for truncnorm = 5.651069592111697e-20

Best fitting distribution: t
Best p value: 1.2246811911143535e-07
Parameters for the best fit: (566.4804912469504, 0.8886439252542147, 0.8544360334114212)
```

Interpretation: After running the code for `Rashid khan` performance with the ball in terms of wickets, the results of fitting various statistical distributions to the performance of bowler AR Patel for the years 2022, 2023, and 2024. The best fitting distribution is the t-distribution has been consistent across all three years: 1.2246811911143535 ×10^{-7}.

```
Result:
************************
year: 2024 batsman: AR Patel
p value for alpha = 1.4283049006330874e-19
p value for beta = 0.08501064188004714
p value for betaprime = 9.200747163367085e-11
p value for burr12 = 0.4240145784486461
p value for crystalball = 0.029775015720014397
p value for dgamma = 0.008447321543132325
p value for dweibull = 0.006765151003550240 5
p value for erlang = 0.0012434310409705773
p value for exponnorm = 0.44275294718405667
p value for f = 1.0828276463613638e-16
p value for fatiguelife = 0.22678195858041206
p value for gamma = 0.01941581626513733
p value for gengamma = 2.3537360809311073e-06
p value for gumbel_l = 4.928627051090389e-06
p value for johnsonsb = 0.2513706078100967
p value for kappa4 = 2.0042162915949264e-21
p value for lognorm = 1.3560827213335702e-29
p value for nct = 0.3161622541605552
p value for norm = 0.029775039515882007
p value for norminvgauss = 0.38491885655137925
p value for powernorm = 0.011133425872538627
p value for rice = 0.011334555411159908
p value for recipinvgauss = 0.0786826137997505
p value for t = 0.022362973623296645
p value for trapz = 1.5833056939085992e-52
p value for truncnorm = 0.07692749313709646

Best fitting distribution: exponnorm
Best p value: 0.44275294718405667
Parameters for the best fit: (2821.2456145120113, -0.014686214458803193, 0.005082734964125436)
```

```
************************
year: 2023 batsman: AR Patel
p value for alpha = 1.4283049006330874e-19
p value for beta = 0.08501064188004714
p value for betaprime = 9.200747163367085e-11
p value for burr12 = 0.4240145784486461
p value for crystalball = 0.029775015720014397
p value for dgamma = 0.008447321543132325
p value for dweibull = 0.0067651510035502405
p value for erlang = 0.0012434310409705773
p value for exponnorm = 0.44275294718405667
p value for f = 1.0828276463613638e-16
p value for fatiguelife = 0.22678195858041206
p value for gamma = 0.01941581626513733
p value for gengamma = 2.3537360809311073e-06
p value for gumbel_l = 4.928627051090389e-06
p value for johnsonsb = 0.2513706078100967
p value for kappa4 = 2.0042162915949264e-21
p value for lognorm = 1.3560827213335702e-29
p value for nct = 0.3161622541605552
p value for norm = 0.029775039515882007
p value for norminvgauss = 0.38491885655137925
p value for powernorm = 0.011133425872538627
p value for rice = 0.011334555411159908
p value for recipinvgauss = 0.0786826137997505
p value for t = 0.022362973623296645
p value for trapz = 1.5833056939085992e-52
p value for truncnorm = 0.07692749313709646

Best fitting distribution: exponnorm
Best p value: 0.44275294718405667
Parameters for the best fit: (2821.2456145120113, -0.014686214458803193, 0.005082734964125436)

************************
year: 2022 batsman: AR Patel
p value for alpha = 1.4283049006330874e-19
p value for beta = 0.08501064188004714
p value for betaprime = 9.200747163367085e-11
p value for burr12 = 0.4240145784486461
p value for crystalball = 0.029775015720014397
p value for dgamma = 0.008447321543132325
p value for dweibull = 0.0067651510035502405
p value for erlang = 0.0012434310409705773
p value for exponnorm = 0.44275294718405667
p value for f = 1.0828276463613638e-16
p value for fatiguelife = 0.22678195858041206
p value for gamma = 0.01941581626513733
p value for gengamma = 2.3537360809311073e-06
p value for gumbel_l = 4.928627051090389e-06
p value for johnsonsb = 0.2513706078100967
p value for kappa4 = 2.0042162915949264e-21
p value for lognorm = 1.3560827213335702e-29
p value for nct = 0.3161622541605552
p value for norm = 0.029775039515882007
p value for norminvgauss = 0.38491885655137925
p value for powernorm = 0.011133425872538627
p value for rice = 0.011334555411159908
```

```
p value for recipinvgauss = 0.0786826137997505
p value for t = 0.022362973623296645
p value for trapz = 1.5833056939085992e-52
p value for truncnorm = 0.07692749313709646

Best fitting distribution: exponnorm
Best p value: 0.44275294718405667
Parameters for the best fit: (2821.2456145120113, -0.014686214458803193, 0.005082734964125436)
```

Interpretation:

For each of the years 2022, 2023, and 2024, the best fitting distribution is the Exponentially Modified Normal (exponnorm) distribution. The highest p-value obtained for the exponnorm distribution is consistent across all three years: 0.4427. This indicates that the exponnorm distribution provides a relatively good fit to the data compared to other distributions.

d) Find the relationship between a player's performance and the salary he gets in your data.

Result:

```
Correlation between Salary and Runs: 0.3349654749323617
Correlation between Salary and Wickets: 0.2127466075152879
```

Interpretation:

To combine player salary data with their performance data (runs scored) for the year 2024, we had to use fuzzy string matching which uses to match player names between the two datasets. Uses the `fuzzywuzzy` library to match player names from the salary data to the player names in the run data. The value of 0.3349 indicates a moderate positive correlation between salary and runs scored. The moderate correlation between salary and runs scored suggests that salaries may be more reflective of batting performance compared to bowling performance, However, in both cases, the correlations are not strong, indicating that many factors influence player performance beyond just their salaries.

Code

```
# Load required libraries
library(dplyr)
library(readr)
library(readxl)
library(ggplot2)

# Set working directory
setwd("C:\\Users\\aravi\\OneDrive\\Desktop\\VCU\\Boot camp class\\Assignment 2")

install.packages("fitdistrplus")
# Load datasets
```

```
ipl_bbb <- read_csv('IPL_ball_by_ball_updated till 2024.csv',show_col_types = FALSE)
ipl_salary <- read_excel('IPL SALARIES 2024.xlsx')

# Display the first two rows of ipl_salary
head(ipl_salary, 2)

# Group the data and aggregate
grouped_data <- ipl_bbb %>%
  group_by(Season, `Innings No`, Striker, Bowler) %>%
  summarise(runs_scored = sum(runs_scored, na.rm = TRUE),
            wicket_confirmation = sum(wicket_confirmation, na.rm = TRUE)) %>%
  ungroup()

# Summarise player runs and wickets
player_runs <- grouped_data %>%
  group_by(Season, Striker) %>%
  summarise(runs_scored = sum(runs_scored, na.rm = TRUE)) %>%
  ungroup()

player_wickets <- grouped_data %>%
  group_by(Season, Bowler) %>%
  summarise(wicket_confirmation = sum(wicket_confirmation, na.rm = TRUE)) %>%
  ungroup()

# Sort player runs for season 2023
player_runs_2023 <- player_runs %>%
  filter(Season == '2023') %>%
  arrange(desc(runs_scored))

# Get top 3 run-getters and bottom 3 wicket-takers per season
top_run_getters <- player_runs %>%
  group_by(Season) %>%
  top_n(3, runs_scored) %>%
  ungroup()

bottom_wicket_takers <- player_wickets %>%
  group_by(Season) %>%
  top_n(3, wicket_confirmation) %>%
  ungroup()

# Print results
cat("Top Three Run Getters:\n")
print(top_run_getters)
```

```
cat("Top Three Wicket Takers:\n")
print(bottom_wicket_takers)

# Create a dataframe for match id and year
ipl_year_id <- data.frame(
  id = ipl_bbb$`Match id`,
  year = format(as.Date(ipl_bbb$Date, format = "%d/%m/%Y"), "%Y")
)

# Create a copy of ipl_bbb dataframe and add a year column
ipl_bbbc <- ipl_bbb %>%
  mutate(year = format(as.Date(Date, format = "%d/%m/%Y"), "%Y"))

# Display the first few rows of the modified dataframe
head(ipl_bbbc %>% select(`Match id`, year, runs_scored, wicket_confirmation, Bowler,
Striker))
head(ipl_bbbc %>% select(`Match id`, year, runs_scored, wicket_confirmation, Bowler,
Striker))
# Load required libraries
library(dplyr)
library(fitdistrplus)
library(data.table)

# Define a function to get the best distribution
get_best_distribution <- function(data) {
  dist_names <- c('norm', 'lnorm', 'gamma', 'weibull', 'exponential', 'logis', 'cauchy')
  dist_results <- list()
  params <- list()
  for (dist_name in dist_names) {
    fit <- fitdist(data, dist_name)
    ks_test <- ks.test(data, dist_name, fit$estimate)
    p_value <- ks_test$p.value
    cat("p value for", dist_name, "=", p_value, "\n")
    dist_results[[dist_name]] <- p_value
    params[[dist_name]] <- fit$estimate
  }
  best_dist <- names(which.max(unlist(dist_results)))
  best_p <- max(unlist(dist_results))
  cat("\nBest fitting distribution:", best_dist, "\n")
  cat("Best p value:", best_p, "\n")
  cat("Parameters for the best fit:", params[[best_dist]], "\n")
  return(list(best_dist, best_p, params[[best_dist]]))
}

# Total runs each year
total_run_each_year <- ipl_bbbc %>%
  group_by(year, Striker) %>%
  summarise(runs_scored = sum(runs_scored, na.rm = TRUE)) %>%
  ungroup() %>%
  arrange(year, desc(runs_scored))

print(total_run_each_year)

list_top_batsman_last_three_year <- list()
for (i in unique(total_run_each_year$year)[1:3]) {
  list_top_batsman_last_three_year[[as.character(i)]] <- total_run_each_year %>%
    filter(year == i) %>%
    top_n(3, runs_scored) %>%
    pull(Striker)
}

print(list_top_batsman_last_three_year)

# Suppress warnings
options(warn = -1)

# Runs for each batsman
runs <- ipl_bbbc %>%
  group_by(Striker, `Match id`) %>%
  summarise(runs_scored = sum(runs_scored, na.rm = TRUE)) %>%
  ungroup()

for (key in names(list_top_batsman_last_three_year)) {
  for (Striker in list_top_batsman_last_three_year[[key]]) {
    cat("************************\n")
    cat("year:", key, " Batsman:", Striker, "\n")
    get_best_distribution(runs %>% filter(Striker == Striker) %>% pull(runs_scored))
    cat("\n\n")
  }
}

# Total wickets each year
total_wicket_each_year <- ipl_bbbc %>%
  group_by(year, Bowler) %>%
  summarise(wicket_confirmation = sum(wicket_confirmation, na.rm = TRUE)) %>%
  ungroup() %>%
  arrange(year, desc(wicket_confirmation))

print(total_wicket_each_year)

list_top_bowler_last_three_year <- list()
for (i in unique(total_wicket_each_year$year)[1:3]) {
  list_top_bowler_last_three_year[[as.character(i)]] <- total_wicket_each_year %>%
    filter(year == i) %>%
    top_n(3, wicket_confirmation) %>%
    pull(Bowler)
}

print(list_top_bowler_last_three_year)

# Load required libraries
library(dplyr)
library(stringdist)
library(fitdistrplus)

# Suppress warnings
options(warn = -1)

# Aggregate wickets data
wickets <- ipl_bbbc %>%
  group_by(Bowler, `Match id`) %>%
  summarise(wicket_confirmation = sum(wicket_confirmation, na.rm = TRUE)) %>%
  ungroup()

# Get best distribution for top bowlers in the last three years
for (key in names(list_top_bowler_last_three_year)) {
  for (bowler in list_top_bowler_last_three_year[[key]]) {
    cat("***********************\n")
    cat("year:", key, " Bowler:", bowler, "\n")
    get_best_distribution(wickets %>% filter(Bowler == bowler) %>%
pull(wicket_confirmation))
    cat("\n\n")
  }
}

# Load necessary libraries
library(dplyr)
library(fitdistrplus)
```

```
# Filter the runs scored by R Parag
Rashid_khan_runs <- runs %>% filter(Striker == "Rashid Khan") %>% pull(runs_scored)

# Function to fit the best distribution
get_best_distribution <- function(data) {
  # Fit different distributions
  fit_norm <- fitdist(data, "norm")
  fit_pois <- fitdist(data, "pois")
  fit_exp <- fitdist(data, "exp")

  # Compare the distributions
  gof_stat <- gofstat(list(fit_norm, fit_pois, fit_exp), fitnames = c("Normal", "Poisson",
"Exponential"))

  # Print the goodness-of-fit statistics
  print(gof_stat)

  # Return the best fit distribution
  best_fit <- names(which.min(gof_stat$aic))
  return(best_fit)
}

# Fit the distribution to R Parag's runs scored and get the best distribution
best_distribution <- get_best_distribution(Rashid_khan_runs)

# Print the best distribution
print(paste("Best fitting distribution:", best_distribution))

# Filter total runs for the year 2024
R2024 <- total_run_each_year %>%
  filter(year == 2024)

# Function to match names using string distance
match_names <- function(name, names_list) {
  match <- amatch(name, names_list, method = "jw", maxDist = 0.2)
  if (!is.na(match)) {
    return(names_list[match])
  } else {
    return(NA)
  }
}
```

```
# Create a new column in ipl_salary with matched names from R2024
ipl_salary$Matched_Player <- sapply(ipl_salary$Player, function(x) match_names(x,
R2024$Striker))

# Merge the dataframes on the matched names
df_merged <- merge(ipl_salary, R2024, by.x = "Matched_Player", by.y = "Striker")

# Display structure of the merged dataframe
str(df_merged)

df_cleaned <- na.omit(df_merged)
correlation <- cor(df_cleaned$Rs, df_cleaned$runs_scored)
cat("Correlation between Salary and Runs:", correlation, "\n")
```