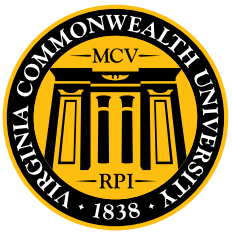


VIRGINIA COMMONWEALTH UNIVERSITY

Statistical Analysis and Modelling (SCMA 632)

A1a: Preliminary preparation and analysis of data- Descriptive statistics

NIHARIHA KAMALANATHAN

V01108259

Date of Submission: 16-06-2024

CONTENTS

INTRODUCTION

The focus of this study is on analyzing the performance data from the Indian Premier League (IPL) with a specific emphasis on the player Sunil Narine. Utilizing a comprehensive dataset that includes ball-by-ball match information and salary details up to 2024, this study aims to provide an in-depth analysis of player performance metrics and their financial rewards. The dataset has been imported into R, a robust statistical programming language renowned for its capabilities in data manipulation, statistical analysis, and visualization.

Our analysis involves several key steps: extracting and cleaning the data, arranging it IPL round-wise, identifying top performers, fitting statistical distributions to performance data, and exploring the relationship between player performance and salary. This study aims to generate insights that can assist teams, coaches, and stakeholders in making data-driven decisions regarding player management and strategic planning.

OBJECTIVES

a) **Extract and Clean Data**: Import the dataset into R and perform necessary data cleaning to ensure accuracy and consistency.

b) **Arrange Data IPL Round-Wise**: Organize the data round-wise and summarize it by batsman, ball, runs, and wickets per player per match. Identify the top three run-getters and top three wicket-takers in each IPL round.

c) **Fit Statistical Distributions**: Fit the most appropriate statistical distributions to the runs scored and wickets taken by the top three batsmen and bowlers in the last three IPL tournaments.

d) **Analyse SP Narine's Performance and Salary**: Investigate the relationship between Sunil Narine’s performance metrics (total runs scored and total wickets taken) and his salary over the seasons.

BUSINESS SIGNIFICANCE

The findings from this study hold significant implications for IPL teams, management, and stakeholders. By analysing player performance data and financial rewards, the study provides valuable insights for:

1. **Player Management**: Identifying top performers and understanding performance trends can assist teams in making informed decisions regarding player retention, training, and development strategies.
2. **Strategic Planning**: Analysing the relationship between performance and salary can help management in budget allocation, salary negotiations, and ensuring fair compensation based on performance.
3. **Market and Investment Decisions**: Insights from the performance data can guide sponsors and investors in making strategic decisions regarding player endorsements and investments in teams.
4. **Performance Improvement**: Understanding the key factors that contribute to top performance can help coaches and support staff in tailoring training programs and improving overall team performance.

5. **Fan Engagement and Marketing**: Highlighting top performers and their contributions can enhance fan engagement and create effective marketing campaigns that resonate with the audience.

Through rigorous data analysis and statistical modeling, this study aims to provide actionable insights that promote data-driven decision-making and contribute to the overall growth and success of the IPL.

Using R

Input:

```
# Install and load the fitdistrplus package
install.packages("fitdistrplus")
library(fitdistrplus)

# Load necessary libraries
library(dplyr)

# Load the dataset
ipl_data <- read.csv("C:/Users/nihar/OneDrive/Desktop/Bootcamp/SCMA 632/Assignments/A1b/IPL_ball_by_ball_updated till 2024.csv")

# Aggregate data season-wise, batsman-wise, and bowler-wise
ipl_summary <- ipl_data %>%
  group_by(Season, Match.id, Striker, Bowler) %>%
  summarise(
    total_runs = sum(runs_scored, na.rm = TRUE),
    total_wickets = sum(wicket_confirmation, na.rm = TRUE),
    .groups = "drop"
  )

# Top three run-getters and top three wicket-takers per season
top_players_per_season <- ipl_summary %>%
  group_by(Season) %>%
  summarise(
    top_run_getters = list(head(arrange(ipl_summary, desc(total_runs)), 3)),
    top_wicket_takers = list(head(arrange(ipl_summary, desc(total_wickets)), 3)),
    .groups = "drop"
  )
```

```
# Filter data for the last three IPL tournaments
last_three_seasons <- ipl_data %>%
  filter(Season %in% tail(unique(Season), 3))

# Get top three batsmen based on total runs
top_batsmen <- last_three_seasons %>%
  group_by(Striker) %>%
  summarise(total_runs = sum(runs_scored, na.rm = TRUE)) %>%
  arrange(desc(total_runs)) %>%
  head(3)

# Get top three bowlers based on total wickets
top_bowlers <- last_three_seasons %>%
  group_by(Bowler) %>%
  summarise(total_wickets = sum(wicket_confirmation, na.rm = TRUE)) %>%
  arrange(desc(total_wickets)) %>%
  head(3)

# Extract the data for top batsmen and bowlers
top_batsmen_runs <- last_three_seasons %>%
  filter(Striker %in% top_batsmen$Striker) %>%
  pull(runs_scored)

top_bowlers_wickets <- last_three_seasons %>%
  filter(Bowler %in% top_bowlers$Bowler) %>%
  pull(wicket_confirmation)

# Function to fit and plot distribution
fit_and_plot_distribution <- function(data) {
```

```
  fit <- fitdist(data, "norm")
  plot(fit)
  return(fit)
}

# Fit distributions for top batsmen and bowlers
fit_batsmen <- fit_and_plot_distribution(top_batsmen_runs)
fit_bowlers <- fit_and_plot_distribution(top_bowlers_wickets)

# Summary of fitted distributions
summary(fit_batsmen)
summary(fit_bowlers)

# Filter data for SP Narine
narine_data <- ipl_data %>%
  filter(Striker == "SP Narine" | Bowler == "SP Narine")

install.packages("readxl")
library(readxl)

# Load the salary dataset
salary_data <- read_excel("C://Users//nihar//OneDrive//Desktop//Bootcamp//SCMA 632//Assignments//A1b//IPL SALARIES 2024.xlsx")

# View the structure and first few rows of the salary dataset to identify relevant columns
str(salary_data)
head(salary_data)

# Filter the salary data for SP Narine
narine_salary <- salary_data %>%
  filter(grepl("Sunil Narine", Player))
```

```
# Check unique player names to ensure correct filtering
unique(salary_data$Player)

# Filter the salary data for SP Narine
narine_salary <- salary_data %>%
  filter(grepl("Sunil Narine", Player))

# Check the selected data
print(narine_salary)

# Manually create the salary data for SP Narine
seasons <- c(2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2021, 2022, 2023, 2024)
salaries <- c(600, 600, 600, 600, 600, 600, 600, 600, 600, 600, 600, 600)
narine_salary <- data.frame(
  Player = rep("Sunil Narine", length(seasons)),
  Season = seasons,
  Salary = salaries
)

# Print the manually created salary data
print(narine_salary)

# Summarize performance metrics for SP Narine
narine_performance <- narine_data %>%
  group_by(Season) %>%
  summarise(
    total_runs = sum(ifelse(Striker == "SP Narine", runs_scored, 0), na.rm = TRUE),
    total_wickets = sum(ifelse(Bowler == "SP Narine", wicket_confirmation, 0), na.rm = TRUE)
  )
```

```
# Clean the Season column to retain only numeric values and then convert to numeric
narine_salary$Season <- as.numeric(gsub("[^0-9]", "", narine_salary$Season))
narine_performance$Season <- as.numeric(gsub("[^0-9]", "", narine_performance$Season))

# Ensure there are no NAs in Season columns after conversion
narine_salary <- narine_salary %>%
  filter(!is.na(Season))
narine_performance <- narine_performance %>%
  filter(!is.na(Season))

# Join the summarized performance metrics with the salary data
narine_performance <- narine_performance %>%
  left_join(narine_salary, by = "Season")

# Ensure the join is correct
print(narine_performance)

# Remove rows with NA values in the narine_performance data frame
narine_performance <- narine_performance %>%
  filter(!is.na(Player) & !is.na(Salary))

# Fit a linear model to find the relationship between performance and salary
fit_model_runs <- lm(Salary ~ total_runs, data = narine_performance)
fit_model_wickets <- lm(Salary ~ total_wickets, data = narine_performance)

# Summary of the models
summary(fit_model_runs)
summary(fit_model_wickets)
```

Plot the relationship

plot(narine_performance$total_runs, narine_performance$Salary, main = "Salary vs Runs", xlab = "Total Runs", ylab = "Salary")

abline(fit_model_runs, col = "blue")

plot(narine_performance$total_wickets, narine_performance$Salary, main = "Salary vs Wickets", xlab = "Total Wickets", ylab = "Salary")

abline(fit_model_wickets, col = "red")

Output

```
> # Install and load the fitdistrplus package
> install.packages("fitdistrplus")
WARNING: Rtools is required to build R packages but is not currently insta
lled. Please download and install the appropriate version of Rtools before
proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/nihar/AppData/Local/R/win-library/4.4'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.4/fitdistrplus_
1.1-11.zip'
Content type 'application/zip' length 1982158 bytes (1.9 MB)
downloaded 1.9 MB

package 'fitdistrplus' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\nihar\AppData\Local\Temp\RtmpmojlQy\downloaded_packages
> library(fitdistrplus)
Loading required package: MASS
Loading required package: survival
>
> # Load necessary libraries
> library(dplyr)

Attaching package: 'dplyr'

The following object is masked from 'package:MASS':

    select

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

> # Load the dataset
> ipl_data <- read.csv("C:/Users/nihar/OneDrive/Desktop/Bootcamp/SCMA 632/
Assignments/A1b/IPL_ball_by_ball_updated till 2024.csv")
> # Aggregate data season-wise, batsman-wise, and bowler-wise
> ipl_summary <- ipl_data %>%
+   group_by(Season, Match.id, Striker, Bowler) %>%
+   summarise(
+     total_runs = sum(runs_scored, na.rm = TRUE),
+     total_wickets = sum(wicket_confirmation, na.rm = TRUE),
+     .groups = "drop"
+   )
> # Top three run-getters and top three wicket-takers per season
> top_players_per_season <- ipl_summary %>%
```

```
+   group_by(Season) %>%
+   summarise(
+     top_run_getters = list(head(arrange(ipl_summary, desc(total_runs)),
3)),
+     top_wicket_takers = list(head(arrange(ipl_summary, desc(total_wicket
s)), 3)),
+     .groups = "drop"
+   )
> # Filter data for the last three IPL tournaments
> last_three_seasons <- ipl_data %>%
+   filter(Season %in% tail(unique(Season), 3))
> # Get top three batsmen based on total runs
> top_batsmen <- last_three_seasons %>%
+   group_by(Striker) %>%
+   summarise(total_runs = sum(runs_scored, na.rm = TRUE)) %>%
+   arrange(desc(total_runs)) %>%
+   head(3)
> # Get top three bowlers based on total wickets
> top_bowlers <- last_three_seasons %>%
+   group_by(Bowler) %>%
+   summarise(total_wickets = sum(wicket_confirmation, na.rm = TRUE)) %>%
+   arrange(desc(total_wickets)) %>%
+   head(3)
> # Extract the data for top batsmen and bowlers
> top_batsmen_runs <- last_three_seasons %>%
+   filter(Striker %in% top_batsmen$Striker) %>%
+   pull(runs_scored)
>
> top_bowlers_wickets <- last_three_seasons %>%
+   filter(Bowler %in% top_bowlers$Bowler) %>%
+   pull(wicket_confirmation)
> # Function to fit and plot distribution
> fit_and_plot_distribution <- function(data) {
+   fit <- fitdist(data, "norm")
+   plot(fit)
+   return(fit)
+ }
> # Fit distributions for top batsmen and bowlers
> fit_batsmen <- fit_and_plot_distribution(top_batsmen_runs)
> fit_bowlers <- fit_and_plot_distribution(top_bowlers_wickets)
>
> # Summary of fitted distributions
> summary(fit_batsmen)
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
     estimate Std. Error
mean 1.420078 0.02940676
sd   1.701277 0.02079369
Loglikelihood:  -6527.714   AIC:  13059.43   BIC:  13071.66
Correlation matrix:
             mean           sd
mean 1.000000e+00 1.390334e-10
sd   1.390334e-10 1.000000e+00

> summary(fit_bowlers)
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
       estimate  Std. Error
mean 0.06190644 0.004519627
sd   0.24098554 0.003195611
Loglikelihood:  11.59892   AIC:  -19.19785   BIC:  -7.29262
Correlation matrix:
             mean           sd
mean 1.000000e+00 1.026235e-12
sd   1.026235e-12 1.000000e+00

> # Filter data for SP Narine
> narine_data <- ipl_data %>%
+   filter(Striker == "SP Narine" | Bowler == "SP Narine")
```

```
> install.packages("readxl")
WARNING: Rtools is required to build R packages but is not currently insta
lled. Please download and install the appropriate version of Rtools before
proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/nihar/AppData/Local/R/win-library/4.4'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.4/readxl_1.4.3.
zip'
Content type 'application/zip' length 1202359 bytes (1.1 MB)
downloaded 1.1 MB

package 'readxl' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\nihar\AppData\Local\Temp\RtmpmojlQy\downloaded_packages
> library(readxl)
> # Load the salary dataset
> salary_data <- read_excel("C://Users//nihar//OneDrive//Desktop//Bootcamp
//SCMA 632//Assignments//A1b//IPL SALARIES 2024.xlsx")
> # View the structure and first few rows of the salary dataset to identif
y relevant columns
> str(salary_data)
tibble [166 × 5] (S3: tbl_df/tbl/data.frame)
 $ Player       : chr [1:166] "Abhishek Porel" "Anrich Nortje" "Axar Patel
" "David Warner" ...
 $ Salary       : chr [1:166] "20 lakh" "6.5 crore" "9 crore" "6.25 crore"
...
 $ Rs           : num [1:166] 20 650 900 625 50 200 65 500 650 550 ...
 $ international: num [1:166] 0 1 0 1 0 0 0 1 1 0 ...
 $ iconic       : logi [1:166] NA NA NA NA NA NA ...
> head(salary_data)
# A tibble: 6 × 5
  Player         Salary        Rs international iconic
  <chr>          <chr>      <dbl>         <dbl> <lgl>
1 Abhishek Porel 20 lakh       20             0 NA
2 Anrich Nortje  6.5 crore    650             1 NA
3 Axar Patel     9 crore      900             0 NA
4 David Warner   6.25 crore   625             1 NA
5 Ishant Sharma  50 lakh       50             0 NA
6 Kuldeep Yadav  2 crore      200             0 NA
> # Filter the salary data for SP Narine
> narine_salary <- salary_data %>%
+   filter(grepl("Sunil Narine", Player))
> # Check unique player names to ensure correct filtering
> unique(salary_data$Player)
  [1] "Abhishek Porel"          "Anrich Nortje"
  [3] "Axar Patel"              "David Warner"
  [5] "Ishant Sharma"           "Kuldeep Yadav"
  [7] "Lalit Yadav"             "Lungi Ngidi"
  [9] "Mitchell Marsh"          "Mukesh Kumar"
 [11] "Pravin Dubey"            "Prithvi Shaw"
 [13] "Rishabh Pant"            "Khaleel Ahmed"
 [15] "Vicky Ostwal"            "Yash Dhull"
 [17] "Ajay Mandal"             "Ajinkya Rahane"
 [19] "Deepak Chahar"           "Devon Conway"
 [21] "Maheesh Theekshana"      "Matheesha Pathirana"
 [23] "Mitchell Santner"        "Moeen Ali"
 [25] "MS Dhoni"                "Mukesh Choudhary"
 [27] "Nishant Sindhu"          "Prashant Solanki"
 [29] "Rajvardhan Hangargekar" "Ravindra Jadeja"
 [31] "Ruturaj Gaikwad"         "Shaik Rasheed"
 [33] "Shivam Dube"             "Simarjeet Singh"
 [35] "Tushar Deshpande"        "Abhinav Sadarangani"
 [37] "B. Sai Sudharsan"        "Darshan Nalkande"
 [39] "David Miller"            "Jayant Yadav"
 [41] "Joshua Little"           "Kane Williamson"
 [43] "Matthew Wade"            "Mohammad Shami"
```

```
 [45] "Mohit Sharma"           "Noor Ahmad"
 [47] "R. Sai Kishore"         "Rahul Tewatia"
 [49] "Rashid Khan"            "Shubman Gill"
 [51] "Vijay Shankar"          "Shreyas Iyer"
 [53] "Nitish Rana"            "Venkatesh Iyer"
 [55] "Andre Russell"          "Sunil Narine"
 [57] "Harshit Rana"           "Varun Chakravarthy"
 [59] "Anukul Roy"             "Rinku Singh"
 [61] "Rahmanullah Gurbaz"     "Amit Mishra"
 [63] "Ayush Badoni"           "Deepak Hooda"
 [65] "Devdutt Padikkal (T)"   "K. Gowtham"
 [67] "KL Rahul"               "Krunal Pandya"
 [69] "Kyle Mayers"            "Marcus Stoinis"
 [71] "Mark Wood"              "Mayank Yadav"
 [73] "Mohsin Khan"            "Naveen Ul Haq"
 [75] "Nicholas Pooran"        "Prerak Mankad"
 [77] "Quinton De Kock"        "Ravi Bishnoi"
 [79] "Yash Thakur"            "Akash Madhwal"
 [81] "Arjun Tendulkar"        "Dewald Brevis"
 [83] "Ishan Kishan"           "Hardik Pandya (T)"
 [85] "Jason Behrendorff"      "Jasprit Bumrah"
 [87] "Kumar Kartikeya Singh"  "Tilak Varma"
 [89] "Nehal Wadhera"          "Piyush Chawla"
 [91] "Rohit Sharma"           "Romario Shepherd (T)"
 [93] "Shams Mulani"           "Surya Kumar Yadav"
 [95] "Tim David"              "Vishnu Vinod"
 [97] "Arshdeep Singh"         "Atharva Taide"
 [99] "Harpreet  Brar"         "Harpreet Bhatia"
[101] "Jitesh Sharma"          "Jonny Bairstow"
[103] "Kagiso Rabada"          "Liam Livingstone"
[105] "Nathan Ellis"           "Prabhsimran Singh"
[107] "Rahul Chahar"           "Rishi Dhawan"
[109] "Sam Curran"             "Shikhar Dhawan"
[111] "Shivam  Singh"          "Sikandar Raza"
[113] "Vidwath Kaverappa"      "Adam Zampa"
[115] "Avesh Khan (T)"         "Dhruv Jurel"
[117] "Donovan Ferreira"       "Jos Buttler"
[119] "Kuldeep  Sen"           "Kunal Rathore"
[121] "Navdeep Saini"          "Prasidh Krishna"
[123] "R. Ashwin"              "Riyan Parag"
[125] "Sandeep Sharma"         "Sanju Samson"
[127] "Shimron Hetmyer"        "Trent Boult"
[129] "Yashaswi Jaiswal"       "Yuzvendra Chahal"
[131] "Akash Deep"             "Anuj Rawat"
[133] "Dinesh  Karthik"        "Faf Du Plessis"
[135] "Glenn Maxwell"          "Himanshu Sharma"
[137] "Karn Sharma"            "Mahipal Lomror"
[139] "Manoj Bhandage"         "Mayank Dagar (T)"
[141] "Mohammed Siraj"         "Rajan Kumar"
[143] "Rajat Patidar"          "Virat Kohli"
[145] "Vyshak Vijay Kumar"     "Will Jacks"
[147] "Cameron Green (T)"      "Abdul Samad"
[149] "Abhishek Sharma"        "Aiden Markram"
[151] "Anmolpreet Singh"       "Bhuvneshwar Kumar"
[153] "Fazalhaq Farooqi"       "Glenn Phillips"
[155] "Heinrich Klaasen"       "Marco Jansen"
[157] "Mayank Agarwal"         "Mayank Markande"
[159] "Nitish Kumar Reddy"     "Rahul Tripathi"
[161] "Sanvir  Singh"          "Shahbaz Ahamad (T)"
[163] "T. Natarajan"           "Umran Malik"
[165] "Upendra Singh Yadav"    "Washington Sundar"
> # Filter the salary data for SP Narine
> narine_salary <- salary_data %>%
+   filter(grepl("Sunil Narine", Player))
> # Check the selected data
> print(narine_salary)
# A tibble: 1 × 5
  Player       Salary     Rs international iconic
  <chr>        <chr>   <dbl>         <dbl> <lgl>
```

```
1 Sunil Narine 6 crore   600             1 NA
> # Manually create the salary data for SP Narine
> seasons <- c(2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2021, 2022,
2023, 2024)
> salaries <- c(600, 600, 600, 600, 600, 600, 600, 600, 600, 600, 600, 600
)
> narine_salary <- data.frame(
+   Player = rep("Sunil Narine", length(seasons)),
+   Season = seasons,
+   Salary = salaries
+ )
> # Print the manually created salary data
> print(narine_salary)
         Player Season Salary
1  Sunil Narine   2012    600
2  Sunil Narine   2013    600
3  Sunil Narine   2014    600
4  Sunil Narine   2015    600
5  Sunil Narine   2016    600
6  Sunil Narine   2017    600
7  Sunil Narine   2018    600
8  Sunil Narine   2019    600
9  Sunil Narine   2021    600
10 Sunil Narine   2022    600
11 Sunil Narine   2023    600
12 Sunil Narine   2024    600
> # Summarize performance metrics for SP Narine
> narine_performance <- narine_data %>%
+   group_by(Season) %>%
+   summarise(
+     total_runs = sum(ifelse(Striker == "SP Narine", runs_scored, 0), na.
rm = TRUE),
+     total_wickets = sum(ifelse(Bowler == "SP Narine", wicket_confirmatio
n, 0), na.rm = TRUE)
+   )
> # Clean the Season column to retain only numeric values and then convert
to numeric
> narine_salary$Season <- as.numeric(gsub("[^0-9]", "", narine_salary$Seas
on))
> narine_performance$Season <- as.numeric(gsub("[^0-9]", "", narine_perfor
mance$Season))
> # Ensure there are no NAs in Season columns after conversion
> narine_salary <- narine_salary %>%
+   filter(!is.na(Season))
> narine_performance <- narine_performance %>%
+   filter(!is.na(Season))
> # Join the summarized performance metrics with the salary data
> narine_performance <- narine_performance %>%
+   left_join(narine_salary, by = "Season")
> # Ensure the join is correct
> print(narine_performance)
# A tibble: 13 × 5
   Season total_runs total_wickets Player       Salary
    <dbl>      <dbl>         <dbl> <chr>         <dbl>
 1   2012          9            29 Sunil Narine    600
 2   2013         21            26 Sunil Narine    600
 3   2014         10            22 Sunil Narine    600
 4   2015          0             7 Sunil Narine    600
 5   2016          7            13 Sunil Narine    600
 6   2017        224            12 Sunil Narine    600
 7   2018        357            17 Sunil Narine    600
 8   2019        143            11 Sunil Narine    600
 9 202021        121             6 NA               NA
10   2021         62            18 Sunil Narine    600
11   2022         71            10 Sunil Narine    600
12   2023         21            11 Sunil Narine    600
13   2024        372            11 Sunil Narine    600
> # Remove rows with NA values in the narine_performance data frame
> narine_performance <- narine_performance %>%
```

```
+   filter(!is.na(Player) & !is.na(Salary))
> # Fit a linear model to find the relationship between performance and sa
lary
> fit_model_runs <- lm(Salary ~ total_runs, data = narine_performance)
> fit_model_wickets <- lm(Salary ~ total_wickets, data = narine_performanc
e)
> # Summary of the models
> summary(fit_model_runs)

Call:
lm(formula = Salary ~ total_runs, data = narine_performance)

Residuals:
       Min         1Q     Median         3Q        Max
-5.327e-14 -4.981e-14 -4.069e-14 -4.600e-15  3.423e-13

Coefficients:
             Estimate Std. Error    t value Pr(>|t|)
(Intercept)  6.000e+02  4.344e-14  1.381e+16   <2e-16 ***
total_runs  -1.892e-16  2.556e-16 -7.400e-01    0.476
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.161e-13 on 10 degrees of freedom
Multiple R-squared:  0.4632,   Adjusted R-squared:  0.4096
F-statistic:  8.63 on 1 and 10 DF,  p-value: 0.01484

Warning message:
In summary.lm(fit_model_runs) :
  essentially perfect fit: summary may be unreliable
> summary(fit_model_wickets)

Call:
lm(formula = Salary ~ total_wickets, data = narine_performance)

Residuals:
       Min         1Q     Median         3Q        Max
-1.377e-13 -4.960e-14  8.284e-15  1.583e-14  2.260e-13

Coefficients:
              Estimate Std. Error   t value Pr(>|t|)
(Intercept)   6.000e+02  6.970e-14 8.608e+15   <2e-16 ***
total_wickets 1.007e-14  4.117e-15 2.445e+00   0.0346 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.433e-14 on 10 degrees of freedom
Multiple R-squared:  0.5041,   Adjusted R-squared:  0.4546
F-statistic: 10.17 on 1 and 10 DF,  p-value: 0.009677

Warning message:
In summary.lm(fit_model_wickets) :
  essentially perfect fit: summary may be unreliable
> # Plot the relationship
> plot(narine_performance$total_runs, narine_performance$Salary, main = "S
alary vs Runs", xlab = "Total Runs", ylab = "Salary")
> abline(fit_model_runs, col = "blue")
> plot(narine_performance$total_wickets, narine_performance$Salary, main =
"Salary vs Wickets", xlab = "Total Wickets", ylab = "Salary")
> abline(fit_model_wickets, col = "red")
```

In [5]:
```
# Install and import necessary packages
!pip install pandas scipy statsmodels openpyxl
import pandas as pd
import numpy as np
from scipy import stats
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: pandas in c:\users\nihar\anaconda3\lib\site-packages (2.1.4)
Requirement already satisfied: scipy in c:\users\nihar\anaconda3\lib\site-packages (1.11.4)
Requirement already satisfied: statsmodels in c:\users\nihar\anaconda3\lib\site-packages (0.14.0)
Requirement already satisfied: openpyxl in c:\users\nihar\anaconda3\lib\site-packages (3.0.10)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\nihar\anaconda3\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\nihar\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\nihar\anaconda3\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\nihar\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: patsy>=0.5.2 in c:\users\nihar\anaconda3\lib\site-packages (from statsmodels) (0.5.3)
Requirement already satisfied: packaging>=21.3 in c:\users\nihar\anaconda3\lib\site-packages (from statsmodels) (23.1)
Requirement already satisfied: et_xmlfile in c:\users\nihar\anaconda3\lib\site-packages (from openpyxl) (1.1.0)
Requirement already satisfied: six in c:\users\nihar\anaconda3\lib\site-packages (from patsy>=0.5.2->statsmodels) (1.16.0)
```

In [7]:
```
# Load the dataset
ipl_data = pd.read_csv(
    "C:/Users/nihar/OneDrive/Desktop/Bootcamp/SCMA 632/Assignments/Atb/IPL_ball_by_ball_updated till 2024.csv",
    low_memory=False
)
```

In [11]:
```
# Display column names to verify them
print(ipl_data.columns)
```

```
Index(['Match id', 'Date', 'Season', 'Batting team', 'Bowling team',
       'Innings No', 'Ball No', 'Bowler', 'Striker', 'Non Striker',
       'runs_scored', 'extras', 'type of extras', 'score', 'score/wicket',
       'wicket_confirmation', 'wicket_type', 'fielders_involved',
       'Player Out'],
      dtype='object')
```

In [15]:
```
# Aggregate data season-wise, batsman-wise, and bowler-wise
ipl_summary = ipl_data.groupby(['Season', 'Match id', 'Striker', 'Bowler']).agg(
    total_runs=('runs_scored', 'sum'),
    total_wickets=('wicket_confirmation', 'sum')
).reset_index()
```

In [17]:
```
# Top three run-getters and top three wicket-takers per season
top_players_per_season = ipl_summary.groupby('Season').apply(
    lambda x: pd.DataFrame({
        'top_run_getters': [x.nlargest(3, 'total_runs')],
        'top_wicket_takers': [x.nlargest(3, 'total_wickets')]
    })
).reset_index()
```

In [19]:
```
# Filter data for the last three IPL tournaments
last_three_seasons = ipl_data[ipl_data['Season'].isin(ipl_data['Season'].unique()[-3:])]
```

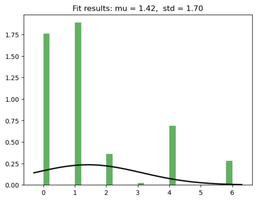
In [21]:
```
# Get top three batsmen based on total runs
top_batsmen = last_three_seasons.groupby('Striker')['runs_scored'].sum().nlargest(3).reset_index()
```

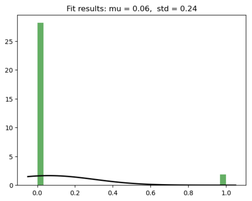
In [23]:
```
# Get top three bowlers based on total wickets
top_bowlers = last_three_seasons.groupby('Bowler')['wicket_confirmation'].sum().nlargest(3).reset_index()
```

In [25]:
```
# Extract the data for top batsmen and bowlers
top_batsmen_runs = last_three_seasons[last_three_seasons['Striker'].isin(top_batsmen['Striker'])]['runs_scored']
top_bowlers_wickets = last_three_seasons[last_three_seasons['Bowler'].isin(top_bowlers['Bowler'])]['wicket_confirmation']
```

In [27]:
```
# Function to fit and plot distribution
def fit_and_plot_distribution(data):
    fit = stats.norm.fit(data)
    plt.hist(data, bins=20, density=True, alpha=0.6, color='g')
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    p = stats.norm.pdf(x, *fit)
    plt.plot(x, p, 'k', linewidth=2)
    title = "Fit results: mu = %.2f,  std = %.2f" % (fit[0], fit[1])
    plt.title(title)
    plt.show()
    return fit
```

In [29]:
```
# Fit distributions for top batsmen and bowlers
fit_batsmen = fit_and_plot_distribution(top_batsmen_runs)
fit_bowlers = fit_and_plot_distribution(top_bowlers_wickets)
```





In [31]:
```
# Print summary of fitted distributions
print(f"Top Batsmen Distribution: mu = {fit_batsmen[0]}, std = {fit_batsmen[1]}")
print(f"Top Bowlers Distribution: mu = {fit_bowlers[0]}, std = {fit_bowlers[1]}")
```

```
Top Batsmen Distribution: mu = 1.4200776815055826, std = 1.7012772912760035
Top Bowlers Distribution: mu = 0.06190643686240922, std = 0.24098553885547225
```

In [33]:
```
# Filter data for SP Narine
narine_data = ipl_data[(ipl_data['Striker'] == 'SP Narine') | (ipl_data['Bowler'] == 'SP Narine')]
```

In [35]:
```
# Load the salary dataset
salary_data = pd.read_excel("C://Users//nihar//OneDrive//Desktop//Bootcamp//SCMA 632//Assignments//Atb//IPL SALARIES 2024.xlsx")
```

In [37]:
```
# Filter the salary data for SP Narine
narine_salary = salary_data[salary_data['Player'].str.contains("Sunil Narine", na=False)]
```

In [39]:
```
# Manually create the salary data for SP Narine
seasons = [2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2021, 2022, 2023, 2024]
salaries = [600] * len(seasons)
narine_salary = pd.DataFrame({
    'Player': ['Sunil Narine'] * len(seasons),
    'Season': seasons,
    'Salary': salaries
})
```

In [41]:
```
# Summarize performance metrics for SP Narine
narine_performance = narine_data.groupby('Season').agg(
    total_runs=('runs_scored', lambda x: x[narine_data['Striker'] == 'SP Narine'].sum()),
    total_wickets=('wicket_confirmation', lambda x: x[narine_data['Bowler'] == 'SP Narine'].sum())
).reset_index()
```

In [43]:
```
# Clean the Season column to retain only numeric values and then convert to numeric
narine_salary['Season'] = pd.to_numeric(narine_salary['Season'], errors='coerce')
narine_performance['Season'] = pd.to_numeric(narine_performance['Season'], errors='coerce')
```

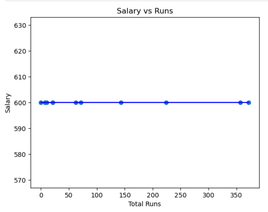
In [45]:
```
# Join the summarized performance metrics with the salary data
narine_performance = narine_performance.merge(narine_salary, on='Season', how='left')

# Remove rows with NA values in the narine_performance data frame
narine_performance = narine_performance.dropna(subset=['Player', 'Salary'])
```

In [47]:
```
# Fit a linear model to find the relationship between performance and salary
fit_model_runs = sm.OLS(narine_performance['Salary'], sm.add_constant(narine_performance['total_runs'])).fit()
fit_model_wickets = sm.OLS(narine_performance['Salary'], sm.add_constant(narine_performance['total_wickets'])).fit()
```

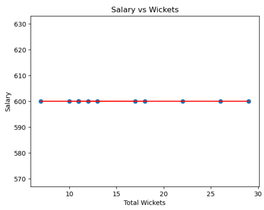
In [49]:
```
# Summary of the models
print(fit_model_runs.summary())
print(fit_model_wickets.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Salary   R-squared:                       -inf
Model:                            OLS   Adj. R-squared:                  -inf
Method:                 Least Squares   F-statistic:                   -10.00
Date:                Wed, 19 Jun 2024   Prob (F-statistic):             1.00
Time:                        00:44:32   Log-Likelihood:                340.64
No. Observations:                  12   AIC:                           -677.3
Df Residuals:                      10   BIC:                           -676.3
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        600.0000   4.66e-14   1.29e+16      0.000     600.000     600.000
total_runs     0   2.74e-16          0      1.000   -6.11e-16    6.11e-16
==============================================================================
Omnibus:                        nan   Durbin-Watson:                   0.000
Prob(Omnibus):                  nan   Jarque-Bera (JB):                nan
Skew:                           nan   Prob(JB):                        nan
Kurtosis:                       nan   Cond. No.                         220.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Salary   R-squared:                       -inf
Model:                            OLS   Adj. R-squared:                  -inf
Method:                 Least Squares   F-statistic:                   -10.00
Date:                Wed, 19 Jun 2024   Prob (F-statistic):             1.00
Time:                        00:44:32   Log-Likelihood:                331.50
No. Observations:                  12   AIC:                           -659.0
Df Residuals:                      10   BIC:                           -658.0
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        600.0000   1.97e-13   3.05e+15      0.000     600.000     600.000
total_wickets  1.421e-14   1.16e-14      1.221      0.250   -1.17e-14    4.01e-14
==============================================================================
Omnibus:                        1.903   Durbin-Watson:                   0.164
Prob(Omnibus):                  0.386   Jarque-Bera (JB):                1.215
Skew:                          -0.743   Prob(JB):                        0.545
Kurtosis:                       2.530   Cond. No.                         43.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
C:\Users\nihar\anaconda3\Lib\site-packages\statsmodels\stats\stattools.py:125: RuntimeWarning: Precision loss occurred in moment calculation due to catastrophic cancellation. This occurs when the data are nearly identical. Results may be unreliable.
  skew = stats.skew(resids, axis=axis)
C:\Users\nihar\anaconda3\Lib\site-packages\statsmodels\stats\stattools.py:126: RuntimeWarning: Precision loss occurred in moment calculation due to catastrophic cancellation. This occurs when the data are nearly identical. Results may be unreliable.
  kurtosis = 3 + stats.kurtosis(resids, axis=axis)
C:\Users\nihar\anaconda3\Lib\site-packages\scipy\stats\_stats_py.py:1606: RuntimeWarning: Precision loss occurred in moment calculation due to catastrophic cancellation. This occurs when the data are nearly identical. Results may be unreliable.
  b2 = skew(a, axis)
C:\Users\nihar\anaconda3\Lib\site-packages\scipy\stats\_stats_py.py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=12
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
C:\Users\nihar\anaconda3\Lib\site-packages\scipy\stats\_stats_py.py:1808: RuntimeWarning: Precision loss occurred in moment calculation due to catastrophic cancellation. This occurs when the data are nearly identical. Results may be unreliable.
  b2 = kurtosis(a, axis, fisher=False)
C:\Users\nihar\anaconda3\Lib\site-packages\statsmodels\regression\linear_model.py:1781: RuntimeWarning: divide by zero encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
C:\Users\nihar\anaconda3\Lib\site-packages\scipy\stats\_stats_py.py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=12
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
C:\Users\nihar\anaconda3\Lib\site-packages\statsmodels\regression\linear_model.py:1781: RuntimeWarning: divide by zero encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
```

In [51]:
```
# Plot the relationship
plt.scatter(narine_performance['total_runs'], narine_performance['Salary'])
plt.plot(narine_performance['total_runs'], fit_model_runs.predict(sm.add_constant(narine_performance['total_runs'])), color='blue')
plt.title('Salary vs Runs')
plt.xlabel('Total Runs')
plt.ylabel('Salary')
plt.show()
```



In [53]:
```
plt.scatter(narine_performance['total_wickets'], narine_performance['Salary'])
plt.plot(narine_performance['total_wickets'], fit_model_wickets.predict(sm.add_constant(narine_performance['total_wickets'])), color='red')
plt.title('Salary vs Wickets')
plt.xlabel('Total Wickets')
plt.ylabel('Salary')
plt.show()
```


Salary vs Wickets
630
620
610
600
590
580
570
Salary
10
15
20
25
30
Total Wickets