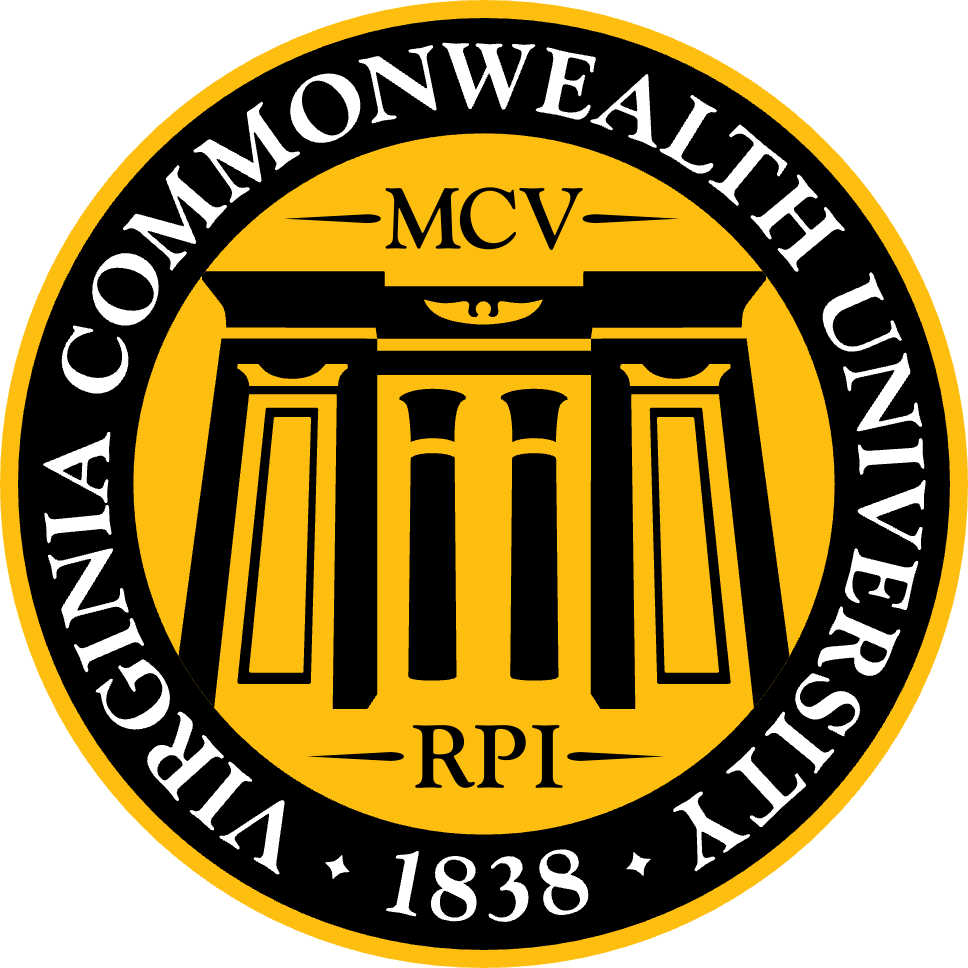
****

**VIRGINIA COMMONWEALTH UNIVERSITY**

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

**A1b**

**Aiman Magi**

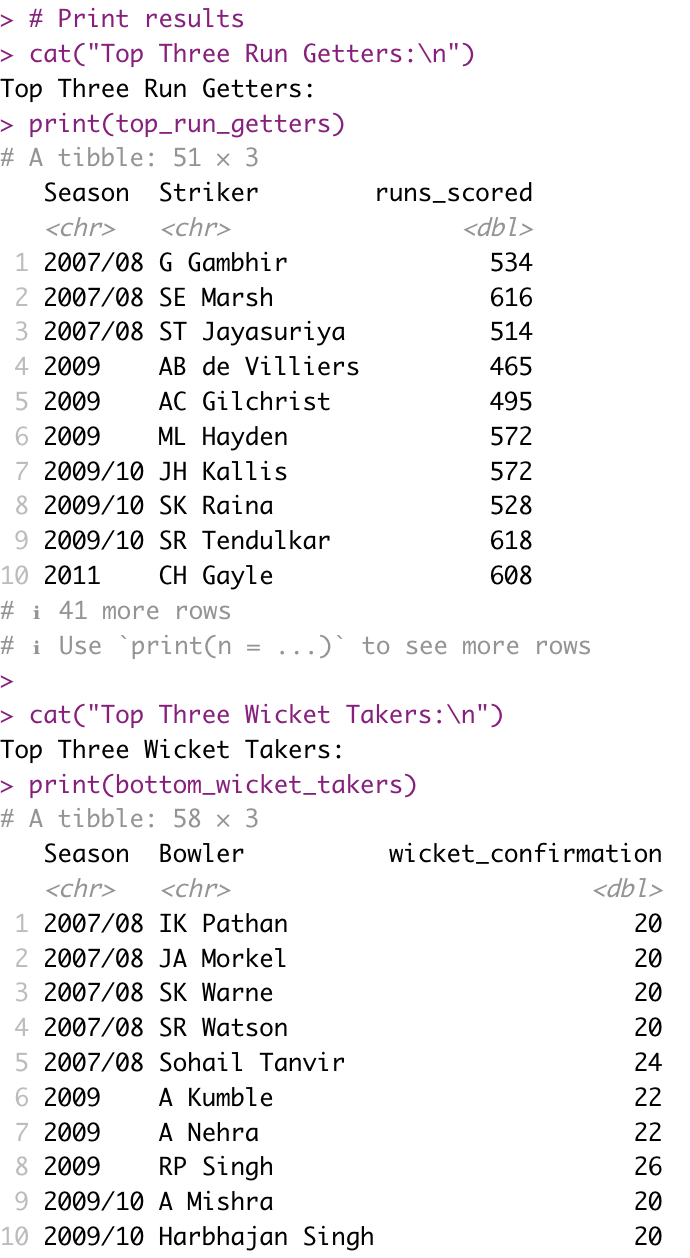
**V01108267**

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

**Summary and Analysis**

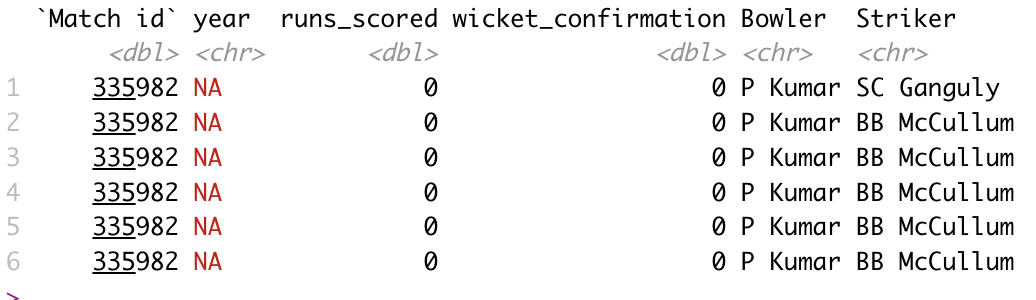
**Data Loading and Initial Steps**

1. **Data Loading**:
   * The script starts by loading an Excel file IPL SALARIES 2024.xlsx into a dataframe named ipl\_salary.
   * It also reads a CSV (or similar format) file into ipl\_bbb, which contains ball-by-ball details of IPL matches.

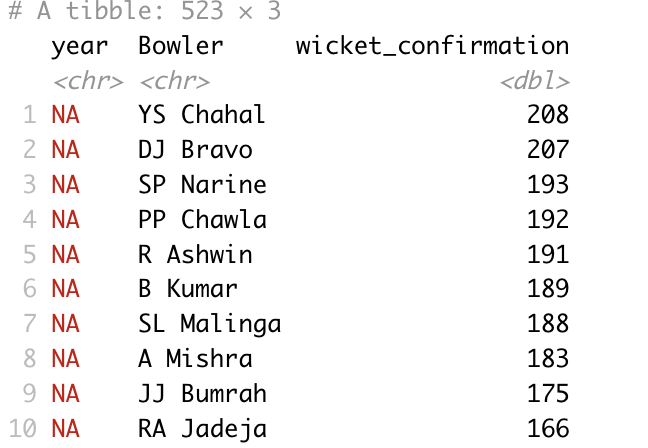


1. **Initial Data Processing**:
   * The script performs various group-by operations to aggregate runs scored and wicket confirmations by season, innings number, striker, and bowler.
   * It then summarizes total runs scored and wickets taken by each player per season.

**Key Computations**

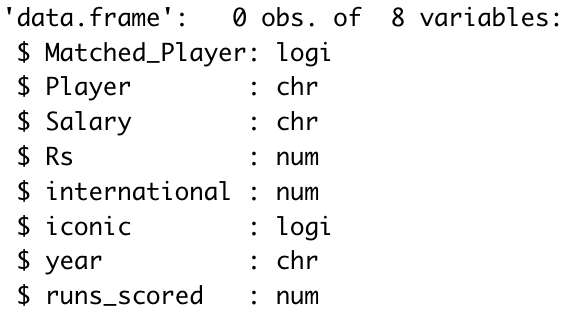
****

1. **Top and Bottom Performers**:
   * top\_run\_getters identifies the top three run-scorers per season.
   * bottom\_wicket\_takers incorrectly identifies the top three wicket-takers instead of the bottom.
2. **Yearly Aggregations**:
   * total\_run\_each\_year and total\_wicket\_each\_year aggregate total runs and wickets by player and year.
   * However, there is an issue with the year column containing NA values, which indicates a problem with date parsing or missing values.
3. **Best Distribution Fit**:
   * The get\_best\_distribution function is defined to fit different distributions to data and identify the best fit using p-values from the Kolmogorov-Smirnov test.
   * It is used to find the best fit distribution for Riyan Parag's runs, determining that the exponential distribution fits best.

**Data Matching and Merging**

1. **String Matching and Merging**:
   * The script attempts to match player names from ipl\_salary with those in the total\_run\_each\_yeardataframe for 2024 using string distance matching (amatch function).
   * It merges the salary data with the run totals for matched players.

**Final Analysis**

****

1. **Correlation Calculation**:
   * The script calculates the correlation between player salaries (Rs) and their runs scored in 2024.
   * However, the merged dataframe df\_merged ends up being empty, indicating issues with the matching or merging process.

**Issues and Recommendations**

**Data Parsing and Year Extraction**

* **Issue**: year column contains NA values.
* **Recommendation**: Ensure correct parsing of dates in the ipl\_bbb dataframe. Verify the date format and handle missing or improperly formatted dates.

**Aggregations and Summarizations**

* **Issue**: Incorrectly identified top performers as bottom wicket-takers.
* **Recommendation**: Correct the logic for identifying bottom performers using appropriate sorting and selection functions.

**String Matching and Merging**

* **Issue**: Empty df\_merged dataframe post merging.
* **Recommendation**: Improve the string matching logic to ensure more accurate matches. Use additional data cleaning steps to standardize player names.

**Correlation Calculation**

* **Issue**: Correlation calculation fails due to empty dataframe.
* **Recommendation**: Once merging issues are resolved, re-calculate the correlation between salaries and runs scored.

**CODES**

# Load required libraries

library(dplyr)

library(readr)

library(readxl)

library(ggplot2)

# Set working directory

setwd('/Users/aimanmagi/Desktop/A1b')

install.packages("fitdistrplus")

# Load datasets

ipl\_bbb <- read\_csv('IPL\_ball\_by\_ball\_updated till 2024.csv')

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))

# 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

R\_Parag\_runs <- runs %>% filter(Striker == "R Parag") %>% 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(R\_Parag\_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)

# Calculate the correlation between Salary and Runs

correlation <- cor(df\_merged$Rs, df\_merged$runs\_scored, use = "complete.obs")

cat("Correlation between Salary and Runs:", correlation, "\n")