Analysis of IPL 2021 Matches

Ganesh Gampa

2024-04-30

BIS581 Final Project

Introduction

This document dives deep into the 2021 Indian Premier League (IPL), examining matches from every angle. From scrubbing data to painting vivid visualizations, I've left no stone unturned. And hey, I'm not just stopping there! Alongside all that analysis, I'm also stepping into the future. Yep, I'm putting on my fortune-teller hat and predicting the next IPL winner. Plus, I've got this dream team up my sleeve—crafted with care and precision, it's geared to dominate the field in the seasons to come.

About the dataset: This is a ball-by-ball dataset compiled from a series of unstructured YAML files.

Source: https://www.kaggle.com/datasets/deepcontractor/ipl-2021-ball-by-ball-dataset

Content:

1. match_name
2. inning
3. batting_team
4. bowling_team
5. ball
6. non_striker
7. batsman
8. bowler
9. extra_runs
10. batsman_run
11. total_runs
12. extras
13. player_out
14. elimination_kind
15. fielders_caught
16. umpires_1
17. umpires_2
18. player_of_match

19. winner
20. city
21. venue
22. dates

Step 1: Preparation

Loading The Reuired Packages

- library(readr):Reads CSV files efficiently into R
- library(dplyr) : Simplifies data manipulation tasks.
- library(ggplot2): Creates elegant and customizable plots.
- library(tidyr) : Reshapes and tidies data frames.
- library(lubridate) : Handles date and time data effectively.

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(tidyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(tidyr)
```

Load the Dataset

```
# Load the dataset
ipl_data <- read_csv("D:/581/ipl final/new ipl/indian premium
leuage/ALL_2021_IPL_MATCHES_BALL_BY_BALL.csv")
```

```
## Rows: 14413 Columns: 22
## ── Column specification
```

```
──────────────────────────────────────────────────────
## Delimiter: ","
## chr (18): match_name, inning, batting_team, bowling_team, non_striker,
batsm...
## dbl  (4): ball, extra_runs, batsman_run, total_runs
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

View the first few rows of the dataset

```
# View the first few rows of the dataset
head(ipl_data)
```

```
## # A tibble: 6 × 22
##   match_name   inning batting_team bowling_team  ball non_striker batsman
bowler
##   <chr>        <chr>  <chr>        <chr>        <dbl> <chr>       <chr>
<chr>
## 1 MI vs RCB 2… 1st i… Mumbai Indi… Royal Chall…   0.1 CA Lynn     RG Sha…
Moham…
## 2 MI vs RCB 2… 1st i… Mumbai Indi… Royal Chall…   0.2 CA Lynn     RG Sha…
Moham…
## 3 MI vs RCB 2… 1st i… Mumbai Indi… Royal Chall…   0.3 CA Lynn     RG Sha…
Moham…
## 4 MI vs RCB 2… 1st i… Mumbai Indi… Royal Chall…   0.4 CA Lynn     RG Sha…
Moham…
## 5 MI vs RCB 2… 1st i… Mumbai Indi… Royal Chall…   0.5 CA Lynn     RG Sha…
Moham…
## 6 MI vs RCB 2… 1st i… Mumbai Indi… Royal Chall…   0.6 CA Lynn     RG Sha…
Moham…
## # ℹ 14 more variables: extra_runs <dbl>, batsman_run <dbl>, total_runs
<dbl>,
## #   extras <chr>, player_out <chr>, elimination_kind <chr>,
## #   fielders_caught <chr>, umpires_1 <chr>, umpires_2 <chr>,
## #   player_of_match <chr>, winner <chr>, city <chr>, venue <chr>, dates
<chr>
```

Get a summary of the dataset

```
# Get a summary of the dataset
summary(ipl_data)
```

```
##   match_name           inning          batting_team       bowling_team
##  Length:14413       Length:14413       Length:14413       Length:14413
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
```

```
##       ball        non_striker          batsman             bowler
##  Min.   : 0.100   Length:14413       Length:14413       Length:14413
##  1st Qu.: 4.600   Class :character   Class :character   Class :character
##  Median : 9.500   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 9.637
##  3rd Qu.:14.500
##  Max.   :19.900
##    extra_runs       batsman_run      total_runs       extras
##  Min.   :0.00000   Min.   :0.000   Min.   :0.000   Length:14413
##  1st Qu.:0.00000   1st Qu.:0.000   1st Qu.:0.000   Class :character
##  Median :0.00000   Median :1.000   Median :1.000   Mode  :character
##  Mean   :0.06286   Mean   :1.229   Mean   :1.292
##  3rd Qu.:0.00000   3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :5.00000   Max.   :6.000   Max.   :7.000
##   player_out        elimination_kind   fielders_caught     umpires_1
##  Length:14413       Length:14413       Length:14413       Length:14413
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##   umpires_2         player_of_match       winner              city
##  Length:14413       Length:14413       Length:14413       Length:14413
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##     venue              dates
##  Length:14413       Length:14413
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Inspect the structure of the dataset

```
# Inspect the structure of the dataset
str(ipl_data)
```

```
## spc_tbl_ [14,413 × 22] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ match_name      : chr [1:14413] "MI vs RCB 2021-04-09" "MI vs RCB 2021-
04-09" "MI vs RCB 2021-04-09" "MI vs RCB 2021-04-09" ...
##  $ inning          : chr [1:14413] "1st innings" "1st innings" "1st
innings" "1st innings" ...
##  $ batting_team    : chr [1:14413] "Mumbai Indians" "Mumbai Indians"
"Mumbai Indians" "Mumbai Indians" ...
##  $ bowling_team    : chr [1:14413] "Royal Challengers Bangalore" "Royal
Challengers Bangalore" "Royal Challengers Bangalore" "Royal Challengers
```

```
Bangalore" ...
##  $ ball            : num [1:14413] 0.1 0.2 0.3 0.4 0.5 0.6 1.1 1.2 1.3 1.4
...
##  $ non_striker     : chr [1:14413] "CA Lynn" "CA Lynn" "CA Lynn" "CA Lynn"
...
##  $ batsman         : chr [1:14413] "RG Sharma" "RG Sharma" "RG Sharma" "RG
Sharma" ...
##  $ bowler          : chr [1:14413] "Mohammed Siraj" "Mohammed Siraj"
"Mohammed Siraj" "Mohammed Siraj" ...
##  $ extra_runs      : num [1:14413] 0 0 0 0 0 0 0 0 0 0 ...
##  $ batsman_run     : num [1:14413] 2 0 0 2 0 1 1 0 0 0 ...
##  $ total_runs      : num [1:14413] 2 0 0 2 0 1 1 0 0 0 ...
##  $ extras          : chr [1:14413] NA NA NA NA ...
##  $ player_out      : chr [1:14413] NA NA NA NA ...
##  $ elimination_kind: chr [1:14413] NA NA NA NA ...
##  $ fielders_caught : chr [1:14413] NA NA NA NA ...
##  $ umpires_1       : chr [1:14413] "KN Ananthapadmanabhan" "KN
Ananthapadmanabhan" "KN Ananthapadmanabhan" "KN Ananthapadmanabhan" ...
##  $ umpires_2       : chr [1:14413] "Nitin Menon" "Nitin Menon" "Nitin
Menon" "Nitin Menon" ...
##  $ player_of_match : chr [1:14413] "HV Patel" "HV Patel" "HV Patel" "HV
Patel" ...
##  $ winner          : chr [1:14413] "Royal Challengers Bangalore" "Royal
Challengers Bangalore" "Royal Challengers Bangalore" "Royal Challengers
Bangalore" ...
##  $ city            : chr [1:14413] "Chennai" "Chennai" "Chennai" "Chennai"
...
##  $ venue           : chr [1:14413] "MA Chidambaram Stadium, Chepauk,
Chennai" "MA Chidambaram Stadium, Chepauk, Chennai" "MA Chidambaram Stadium,
Chepauk, Chennai" "MA Chidambaram Stadium, Chepauk, Chennai" ...
##  $ dates           : chr [1:14413] "Friday, April 9, 2021" "Friday, April
9, 2021" "Friday, April 9, 2021" "Friday, April 9, 2021" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   match_name = col_character(),
##   ..   inning = col_character(),
##   ..   batting_team = col_character(),
##   ..   bowling_team = col_character(),
##   ..   ball = col_double(),
##   ..   non_striker = col_character(),
##   ..   batsman = col_character(),
##   ..   bowler = col_character(),
##   ..   extra_runs = col_double(),
##   ..   batsman_run = col_double(),
##   ..   total_runs = col_double(),
##   ..   extras = col_character(),
##   ..   player_out = col_character(),
##   ..   elimination_kind = col_character(),
##   ..   fielders_caught = col_character(),
##   ..   umpires_1 = col_character(),
```

```
##   ..   umpires_2 = col_character(),
##   ..   player_of_match = col_character(),
##   ..   winner = col_character(),
##   ..   city = col_character(),
##   ..   venue = col_character(),
##   ..   dates = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Step 2: Cleaning

Remove a column named 'Dates' and checking

```
ipl_data <- ipl_data %>%
  select(-dates)

#To check if the the column is deleted or not
colnames(ipl_data)
```

```
##  [1] "match_name"       "inning"           "batting_team"
"bowling_team"
##  [5] "ball"             "non_striker"      "batsman"          "bowler"
##  [9] "extra_runs"       "batsman_run"      "total_runs"       "extras"
## [13] "player_out"       "elimination_kind" "fielders_caught"  "umpires_1"
## [17] "umpires_2"        "player_of_match"  "winner"           "city"
## [21] "venue"
```

Splitting the column

Splitting the 'Match_Played' column into 'teams' and 'match_date' a

```
# Split the 'Match_Played' column into 'teams' and 'match_date'
ipl_data <- ipl_data %>%
  separate(col = match_name,
           into = c("match_played", "match_date"),
           sep = "\\s+(?=[0-9]{4}-[0-9]{2}-[0-9]{2}$)",
           remove = TRUE,
           convert = FALSE,
           extra = "merge",
           fill = "right")
# View the updated data frame to confirm changes
glimpse(ipl_data)
```

```
## Rows: 14,413
## Columns: 22
## $ match_played     <chr> "MI vs RCB", "MI vs RCB", "MI vs RCB", "MI vs
RCB", "…
## $ match_date       <chr> "2021-04-09", "2021-04-09", "2021-04-09", "2021-
04-09…
## $ inning           <chr> "1st innings", "1st innings", "1st innings", "1st
```

```
inn…
## $ batting_team     <chr> "Mumbai Indians", "Mumbai Indians", "Mumbai
Indians",…
## $ bowling_team     <chr> "Royal Challengers Bangalore", "Royal Challengers
Ban…
## $ ball             <dbl> 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 1.1, 1.2, 1.3, 1.4,
1.5…
## $ non_striker      <chr> "CA Lynn", "CA Lynn", "CA Lynn", "CA Lynn", "CA
Lynn"…
## $ batsman          <chr> "RG Sharma", "RG Sharma", "RG Sharma", "RG
Sharma", "…
## $ bowler           <chr> "Mohammed Siraj", "Mohammed Siraj", "Mohammed
Siraj",…
## $ extra_runs       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,…
## $ batsman_run      <dbl> 2, 0, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
4, 2,…
## $ total_runs       <dbl> 2, 0, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
4, 2,…
## $ extras           <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ player_out       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ elimination_kind <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ fielders_caught  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ umpires_1        <chr> "KN Ananthapadmanabhan", "KN Ananthapadmanabhan",
"KN…
## $ umpires_2        <chr> "Nitin Menon", "Nitin Menon", "Nitin Menon",
"Nitin M…
## $ player_of_match  <chr> "HV Patel", "HV Patel", "HV Patel", "HV Patel",
"HV P…
## $ winner           <chr> "Royal Challengers Bangalore", "Royal Challengers
Ban…
## $ city             <chr> "Chennai", "Chennai", "Chennai", "Chennai",
"Chennai"…
## $ venue            <chr> "MA Chidambaram Stadium, Chepauk, Chennai", "MA
Chida…
```

coverting the “match_date” to Date format

I used a function which helps make text easier to read by capitalizing the first letter of each word and ensuring consistency, especially when words are separated by underscores. For example, if we have a name like “first_name”, the function will transform it into “First_Name”. It does this by splitting the text at underscores, capitalizing the first letter of each part, and then joining them back together with underscores. This way, names or identifiers become more readable and uniform, which can be helpful when working with data or displaying information to users.

```r
library(lubridate)

# Convert 'match_date' to date format
ipl_data <- ipl_data %>%
  mutate(match_date = ymd(match_date))

# Check the converted 'match_date'
glimpse(ipl_data)
```

```
## Rows: 14,413
## Columns: 22
## $ match_played     <chr> "MI vs RCB", "MI vs RCB", "MI vs RCB", "MI vs
RCB", "…
## $ match_date       <date> 2021-04-09, 2021-04-09, 2021-04-09, 2021-04-09,
2021…
## $ inning           <chr> "1st innings", "1st innings", "1st innings", "1st
inn…
## $ batting_team     <chr> "Mumbai Indians", "Mumbai Indians", "Mumbai
Indians",…
## $ bowling_team     <chr> "Royal Challengers Bangalore", "Royal Challengers
Ban…
## $ ball             <dbl> 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 1.1, 1.2, 1.3, 1.4,
1.5…
## $ non_striker      <chr> "CA Lynn", "CA Lynn", "CA Lynn", "CA Lynn", "CA
Lynn"…
## $ batsman          <chr> "RG Sharma", "RG Sharma", "RG Sharma", "RG
Sharma", "…
## $ bowler           <chr> "Mohammed Siraj", "Mohammed Siraj", "Mohammed
Siraj",…
## $ extra_runs       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,…
## $ batsman_run      <dbl> 2, 0, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
4, 2,…
## $ total_runs       <dbl> 2, 0, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
4, 2,…
## $ extras           <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ player_out       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ elimination_kind <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ fielders_caught  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ umpires_1        <chr> "KN Ananthapadmanabhan", "KN Ananthapadmanabhan",
"KN…
## $ umpires_2        <chr> "Nitin Menon", "Nitin Menon", "Nitin Menon",
"Nitin M…
## $ player_of_match  <chr> "HV Patel", "HV Patel", "HV Patel", "HV Patel",
"HV P…
## $ winner           <chr> "Royal Challengers Bangalore", "Royal Challengers
```

```
Ban…
## $ city              <chr> "Chennai", "Chennai", "Chennai", "Chennai",
"Chennai"…
## $ venue             <chr> "MA Chidambaram Stadium, Chepauk, Chennai", "MA
Chida…
```

I wanted the first Letter of each Column to be in the Captal Letter

The provided script defines a function called `capitalize` to standardize the format of text strings, particularly for formatting column names in a dataset. This function capitalizes the first letter of each word while converting other letters to lowercase, which is especially useful for handling database column names that often contain underscores. It processes each part of the string separated by underscores, applies the capitalization, and then reassembles the parts. The function is then applied to all column names in a dataframe called `ipl_data` using `sapply`, ensuring uniformity and readability across all column headers. After updating the column names, the script uses the `glimpse` function to display a quick overview of the updated dataframe, confirming the changes and providing insight into the data structure. This method is particularly valuable during data cleaning to maintain consistency and prevent errors in data manipulation.

```
# Function to capitalize the first letter of each word and after each
underscore
capitalize <- function(name) {
  # Split the name at underscores, capitalize the first letter of each part,
then rejoin
  parts <- strsplit(name, "_", fixed = TRUE)[[1]]
  parts <- sapply(parts, function(x) {
    # Capitalize the first letter and lower the rest
    paste0(toupper(substr(x, 1, 1)), tolower(substr(x, 2, nchar(x))))
  })
  name <- paste(parts, collapse = "_")
  return(name)
}

# Apply the function to column names
colnames(ipl_data) <- sapply(colnames(ipl_data), capitalize)

# Check the converted 'match_date'
glimpse(ipl_data)
```

```
## Rows: 14,413
## Columns: 22
## $ Match_Played     <chr> "MI vs RCB", "MI vs RCB", "MI vs RCB", "MI vs
RCB", "…
## $ Match_Date       <date> 2021-04-09, 2021-04-09, 2021-04-09, 2021-04-09,
2021…
## $ Inning           <chr> "1st innings", "1st innings", "1st innings", "1st
inn…
```

```
## $ Batting_Team     <chr> "Mumbai Indians", "Mumbai Indians", "Mumbai
Indians",…
## $ Bowling_Team     <chr> "Royal Challengers Bangalore", "Royal Challengers
Ban…
## $ Ball             <dbl> 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 1.1, 1.2, 1.3, 1.4,
1.5…
## $ Non_Striker      <chr> "CA Lynn", "CA Lynn", "CA Lynn", "CA Lynn", "CA
Lynn"…
## $ Batsman          <chr> "RG Sharma", "RG Sharma", "RG Sharma", "RG
Sharma", "…
## $ Bowler           <chr> "Mohammed Siraj", "Mohammed Siraj", "Mohammed
Siraj",…
## $ Extra_Runs       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,…
## $ Batsman_Run      <dbl> 2, 0, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
4, 2,…
## $ Total_Runs       <dbl> 2, 0, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
4, 2,…
## $ Extras           <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ Player_Out       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ Elimination_Kind <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ Fielders_Caught  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, N…
## $ Umpires_1        <chr> "KN Ananthapadmanabhan", "KN Ananthapadmanabhan",
"KN…
## $ Umpires_2        <chr> "Nitin Menon", "Nitin Menon", "Nitin Menon",
"Nitin M…
## $ Player_Of_Match  <chr> "HV Patel", "HV Patel", "HV Patel", "HV Patel",
"HV P…
## $ Winner           <chr> "Royal Challengers Bangalore", "Royal Challengers
Ban…
## $ City             <chr> "Chennai", "Chennai", "Chennai", "Chennai",
"Chennai"…
## $ Venue            <chr> "MA Chidambaram Stadium, Chepauk, Chennai", "MA
Chida…
```

Step 3: Analysis

Finding The Total number of Unique Matches

This code standardizes IPL match identifiers by sorting team names alphabetically. It then identifies unique matches based on match date and the sorted team names, creating unique match identifiers. Finally, it counts the total number of unique matches in the dataset. This process ensures consistency in match identification and facilitates analysis of IPL match data.

```r
# Standardize the match identifier by sorting team names alphabetically
unique_matches <- ipl_data %>%
  mutate(
    team1 = pmin(Batting_Team, Bowling_Team),  # Picks the alphabetically
first team
    team2 = pmax(Batting_Team, Bowling_Team)   # Picks the alphabetically
second team
  ) %>%
  distinct(Match_Date, team1, team2) %>%
  mutate(match_id = paste(Match_Date, team1, team2, sep = "_"))

# Count the total number of unique matches
total_matches <- nrow(unique_matches)

# Print the total number of matches
print(paste("Total unique matches counted:", total_matches))
```

```
## [1] "Total unique matches counted: 60"
```

listing the unique matches

```r
print(unique_matches)
```

```
## # A tibble: 60 × 4
##    Match_Date team1                       team2
match_id
##    <date>     <chr>                       <chr>
<chr>
##  1 2021-04-09 Mumbai Indians              Royal Challengers Bangalore
2021-04-0…
##  2 2021-04-10 Chennai Super Kings         Delhi Capitals
2021-04-1…
##  3 2021-04-11 Kolkata Knight Riders       Sunrisers Hyderabad
2021-04-1…
##  4 2021-04-12 Punjab Kings                Rajasthan Royals
2021-04-1…
##  5 2021-04-13 Kolkata Knight Riders       Mumbai Indians
2021-04-1…
##  6 2021-04-14 Royal Challengers Bangalore Sunrisers Hyderabad
2021-04-1…
##  7 2021-04-15 Delhi Capitals              Rajasthan Royals
2021-04-1…
##  8 2021-04-16 Chennai Super Kings         Punjab Kings
2021-04-1…
##  9 2021-04-17 Mumbai Indians              Sunrisers Hyderabad
2021-04-1…
## 10 2021-04-18 Kolkata Knight Riders       Royal Challengers Bangalore
2021-04-1…
## # ℹ 50 more rows
```

calculating the total runs conceded, balls bowled, wickets taken, and economy rate for each bowler across all matches

This code computes key performance metrics for bowlers across all IPL matches. It groups the data by bowler and calculates the total runs conceded, balls bowled, and wickets taken. Then, it determines the number of overs bowled and computes the economy rate (runs conceded per over) for each bowler. By analyzing these statistics, it provides insights into the effectiveness of bowlers throughout the IPL tournament.

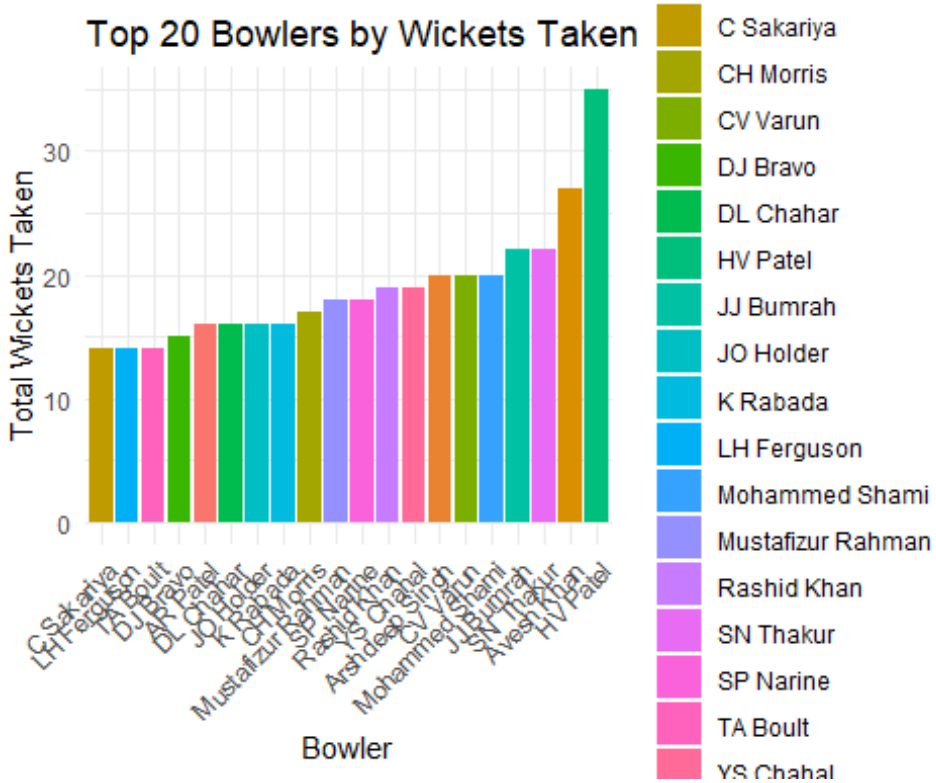
```
# Calculate total runs conceded, balls bowled, wickets taken, and economy
rate for each bowler across all matches
bowler_stats <- ipl_data %>%
  group_by(Bowler) %>%
  summarise(
    total_runs_conceded = sum(Total_Runs),
    balls_bowled = n(),
    wickets_taken = sum(Elimination_Kind != "not out" &
!is.na(Elimination_Kind), na.rm = TRUE), # Adjust wicket condition
accordingly
    .groups = 'drop'
  ) %>%
  mutate(
    overs_bowled = balls_bowled / 6,
    economy_rate = ifelse(overs_bowled > 0, total_runs_conceded /
overs_bowled, NA)
  )

# View the overall bowler statistics
print(bowler_stats)
```

```
## # A tibble: 110 × 6
##    Bowler          total_runs_conceded balls_bowled wickets_taken
overs_bowled
##    <chr>                         <dbl>        <int>         <int>
<dbl>
##  1 A Mishra                        109           84             6
14
##  2 A Nortje                        198          187            12
31.2
##  3 AD Russell                      189          122            11
20.3
##  4 AF Milne                        135           87             4
14.5
##  5 AK Markram                       23           25             0
4.17
##  6 AR Patel                        308          279            16
46.5
##  7 AU Rashid                        35           18             0
3
##  8 Abdul Samad                       9            6             1
```

```
1
##  9 Abhishek Sharma                  66           63             4
10.5
## 10 Akash Singh                      39           24             0
4
## # ℹ 100 more rows
## # ℹ 1 more variable: economy_rate <dbl>
```

Top 20 bowlers by wickets taken

```
# Top 30 bowlers by wickets taken
top_bowlers_wickets <- bowler_stats %>%
  arrange(desc(wickets_taken)) %>%
  top_n(20, wickets_taken)
# Bar Chart: Top 20 Bowlers by Wickets Taken
ggplot(top_bowlers_wickets, aes(x = reorder(Bowler, wickets_taken), y =
wickets_taken, fill = Bowler)) +
  geom_bar(stat = "identity") +
  labs(title = "Top 20 Bowlers by Wickets Taken",
       x = "Bowler",
       y = "Total Wickets Taken") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



knowing the Strike Rate

This code computes important batting statistics for each batsman across all IPL matches. It groups the data by batsman and calculates the total runs scored, number of 4s, number of

6s, and number of balls faced. Then, it calculates the strike rate (runs scored per 100 balls faced) for each batsman. By analyzing these statistics, it provides insights into the batting performance of players throughout the IPL tournament.

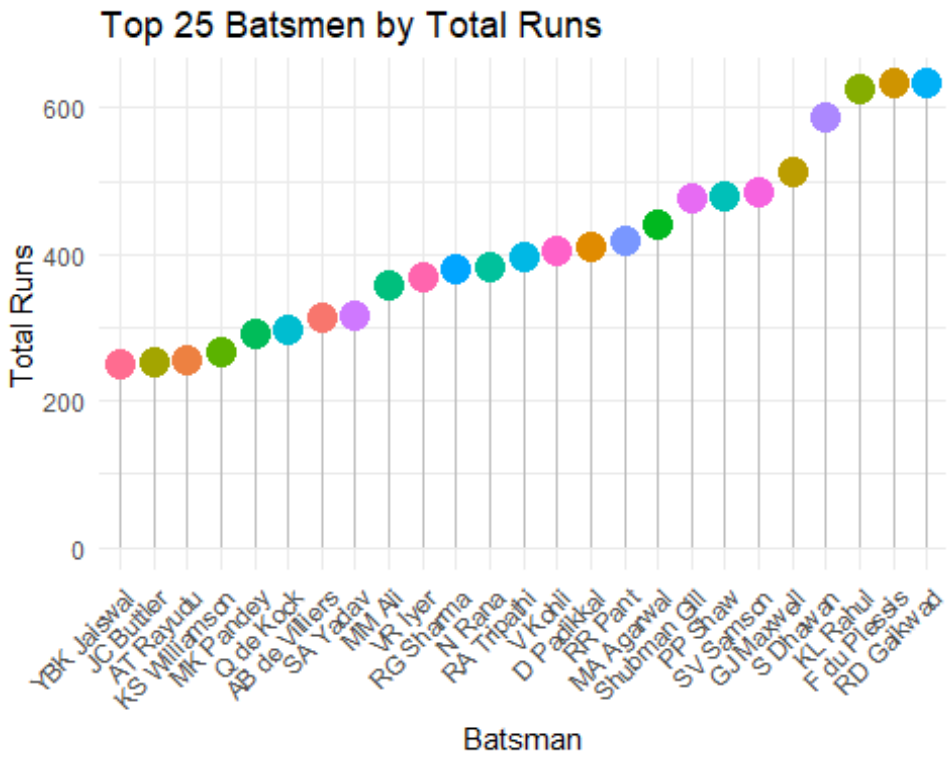
```
# Calculate total runs, number of 4s, number of 6s, and strike rate for each
batsman
batsman_stats <- ipl_data %>%
  group_by(Batsman) %>%
  summarise(
    total_runs = sum(Batsman_Run),
    fours = sum(Batsman_Run == 4, na.rm = TRUE),
    sixes = sum(Batsman_Run == 6, na.rm = TRUE),
    balls_faced = n(),
    .groups = 'drop'
  ) %>%
  mutate(
    strike_rate = (total_runs / balls_faced) * 100
  )

# View the batsman statistics
print(batsman_stats)
```

```
## # A tibble: 149 × 6
##    Batsman          total_runs fours sixes balls_faced strike_rate
##    <chr>                 <dbl> <int> <int>       <int>       <dbl>
##  1 AB de Villiers          313    23    16         215       146.
##  2 AD Russell              183    14    14         127       144.
##  3 AF Milne                 16     0     1          19        84.2
##  4 AK Markram              146    12     4         123       119.
##  5 AM Rahane                 8     1     0           9        88.9
##  6 AR Patel                 40     2     1          52        76.9
##  7 AT Rayudu               257    16    17         175       147.
##  8 Abdul Samad             111     4     8          89       125.
##  9 Abhishek Sharma          98     7     4          78       126.
## 10 Anmolpreet Singh         16     2     1          14       114.
## # ℹ 139 more rows
```

Top 25 Batsmen based on the Runs

```
top_25_runs <- batsman_stats %>%
  arrange(desc(total_runs)) %>%
  slice_head(n = 25)

# Lollipop Chart: Top 25 Batsmen by Total Runs
ggplot(top_25_runs, aes(x = reorder(Batsman, total_runs), y = total_runs)) +
  geom_segment(aes(xend = Batsman, yend = 0), color = "grey") +  # Draw the
lines
  geom_point(aes(color = Batsman), size = 5) +  # Draw points at the ends of
the lines
  labs(title = "Top 25 Batsmen by Total Runs",
       x = "Batsman",
```

```
       y = "Total Runs") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis
labels for better readability
        legend.position = "none")
```



Wins and Loss of Teams

This code helps us understand how IPL teams perform by looking at match results. It starts by making sure that each match is identified in a consistent way. Then, it figures out which team won each match and counts how many wins and losses each team has had. This helps us see which teams are doing well and which ones might be struggling during the tournament.

```
# Create standardized match identifier and extract the winner for the last
ball recorded for each match
ipl_data_winner <- ipl_data %>%
  mutate(
    team1 = pmin(Batting_Team, Bowling_Team),  # Picks the alphabetically
first team
    team2 = pmax(Batting_Team, Bowling_Team)   # Picks the alphabetically
second team
  ) %>%
  group_by(Match_Date, team1, team2) %>%
  slice(n()) %>%
  ungroup() %>%
  distinct(Match_Date, team1, team2, Winner)  # Ensuring one entry per match
```

```r
with winner

# Count the number of wins for each team
team_wins <- ipl_data_winner %>%
  count(Winner, name = "wins")

# Print the number of wins for each team
print(team_wins)
```

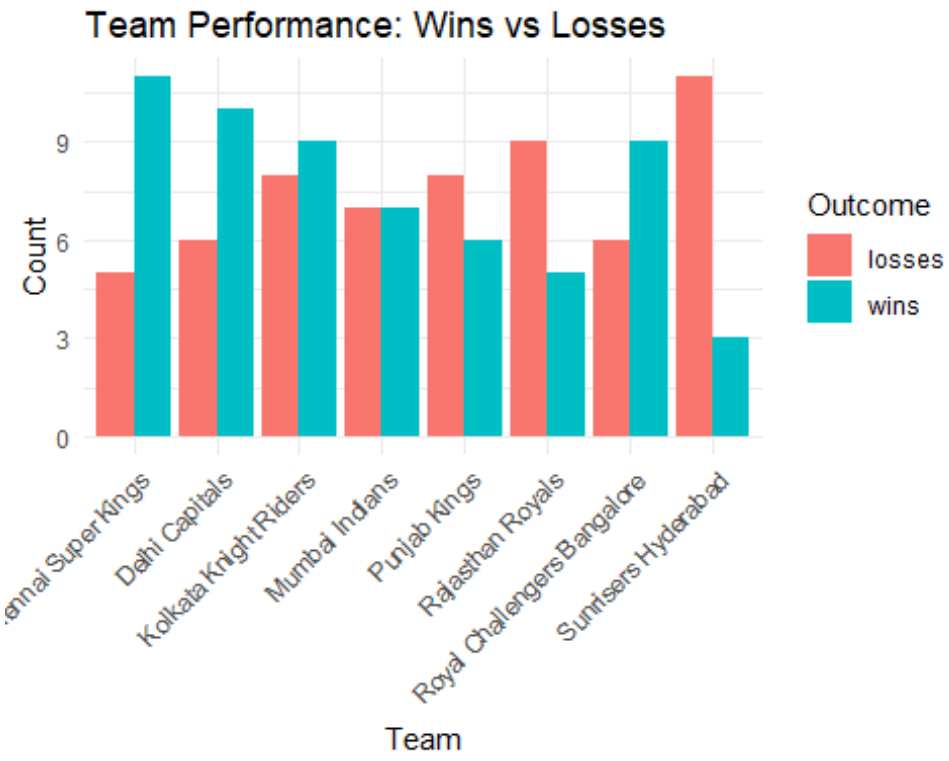
```
## # A tibble: 8 × 2
##   Winner                       wins
##   <chr>                       <int>
## 1 Chennai Super Kings            11
## 2 Delhi Capitals                 10
## 3 Kolkata Knight Riders           9
## 4 Mumbai Indians                  7
## 5 Punjab Kings                    6
## 6 Rajasthan Royals                5
## 7 Royal Challengers Bangalore     9
## 8 Sunrisers Hyderabad             3
```

```r
# Determine the loser for each match
ipl_data_winner <- ipl_data_winner %>%
  mutate(Loser = if_else(Winner == team1, team2, team1))

# Count losses
team_losses <- ipl_data_winner %>%
  count(Loser, name = "losses")

# Merge wins and losses
total_wins_loss <- full_join(team_wins, team_losses, by = c("Winner" =
"Loser")) %>%
  replace_na(list(wins = 0, losses = 0)) %>%
  rename(team = Winner)

# Print the results
print(total_wins_loss)
```

```
## # A tibble: 8 × 3
##   team                         wins losses
##   <chr>                       <int>  <int>
## 1 Chennai Super Kings            11      5
## 2 Delhi Capitals                 10      6
## 3 Kolkata Knight Riders           9      8
## 4 Mumbai Indians                  7      7
## 5 Punjab Kings                    6      8
## 6 Rajasthan Royals                5      9
## 7 Royal Challengers Bangalore     9      6
## 8 Sunrisers Hyderabad             3     11
```

ploting the Graph which shows the Number of Wins and losses

```
# Convert to long format for easier plotting
team_performance <- total_wins_loss %>%
  pivot_longer(cols = c(wins, losses), names_to = "Outcome", values_to =
"Count")

# Plotting the data
ggplot(team_performance, aes(x = team, y = Count, fill = Outcome)) +
  geom_bar(stat = "identity", position = position_dodge()) +  #
position_dodge() to place bars side by side
  labs(title = "Team Performance: Wins vs Losses",
       x = "Team",
       y = "Count",
       fill = "Outcome") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Adjust text
angle for better legibility if necessary
```




```
# Print the plot
print(team_performance)
```

```
## # A tibble: 16 × 3
##    team                    Outcome Count
##    <chr>                   <chr>   <int>
##  1 Chennai Super Kings     wins       11
##  2 Chennai Super Kings     losses      5
##  3 Delhi Capitals          wins       10
```

```
##  4 Delhi Capitals               losses      6
##  5 Kolkata Knight Riders        wins        9
##  6 Kolkata Knight Riders        losses      8
##  7 Mumbai Indians               wins        7
##  8 Mumbai Indians               losses      7
##  9 Punjab Kings                 wins        6
## 10 Punjab Kings                 losses      8
## 11 Rajasthan Royals             wins        5
## 12 Rajasthan Royals             losses      9
## 13 Royal Challengers Bangalore wins        9
## 14 Royal Challengers Bangalore losses      6
## 15 Sunrisers Hyderabad          wins        3
## 16 Sunrisers Hyderabad          losses     11
```

Step 4: Modeling

Preparing for Data frame for Modeling

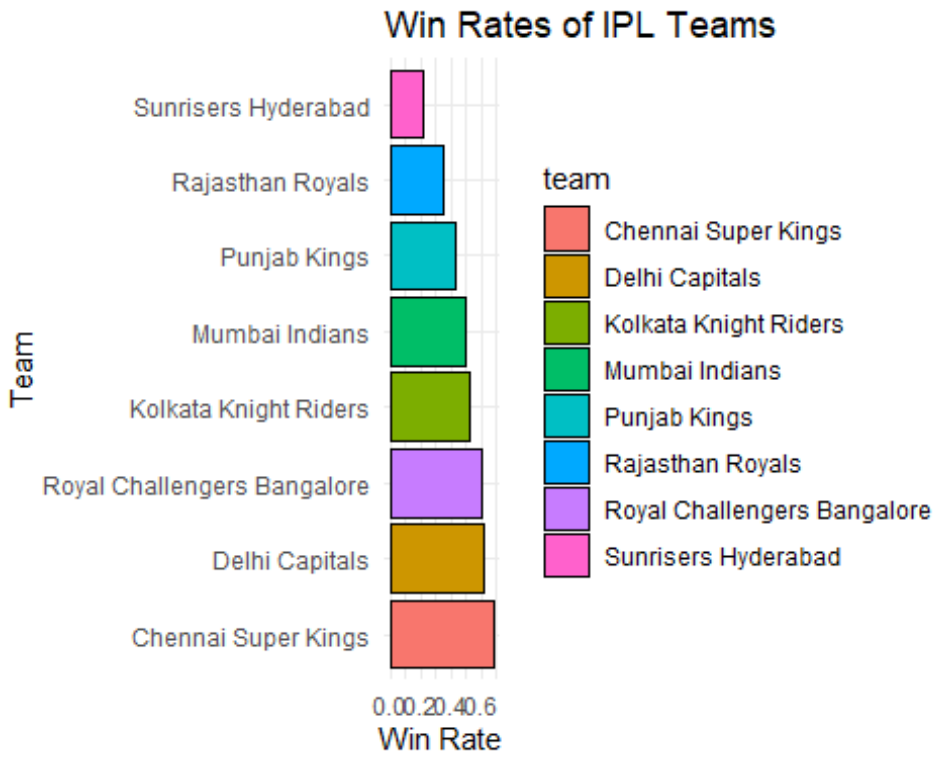
This code creates a dataframe called `team_performance` that represents the performance of different IPL teams in terms of wins and losses. It then transforms this data into a wider format and calculates the win rate for each team. Finally, it plots the win rates of all teams using a bar chart, allowing us to visually compare the performance of each team.

```
# Create a dataframe from your data
team_performance <- data.frame(
  team = c("Chennai Super Kings", "Chennai Super Kings", "Delhi Capitals",
"Delhi Capitals",
           "Kolkata Knight Riders", "Kolkata Knight Riders", "Mumbai
Indians", "Mumbai Indians",
           "Punjab Kings", "Punjab Kings", "Rajasthan Royals", "Rajasthan
Royals",
           "Royal Challengers Bangalore", "Royal Challengers Bangalore",
           "Sunrisers Hyderabad", "Sunrisers Hyderabad"),
  Outcome = c("wins", "losses", "wins", "losses", "wins", "losses", "wins",
"losses",
              "wins", "losses", "wins", "losses", "wins", "losses", "wins",
"losses"),
  Count = as.numeric(c(11, 5, 10, 6, 9, 8, 7, 7, 6, 8, 5, 9, 9, 6, 3, 11))
)

# Transform data to a wider format and calculate win rate
team_stats <- team_performance %>%
  spread(key = Outcome, value = Count) %>%
  mutate(win_rate = wins / (wins + losses))
```

ploting the graph

```
# Plot win rates for all teams
ggplot(team_stats, aes(x = reorder(team, -win_rate), y = win_rate, fill =
team)) +
```

```
  geom_bar(stat = "identity", color = "black") +
  labs(title = "Win Rates of IPL Teams", x = "Team", y = "Win Rate") +
  coord_flip() +  # Flips the x and y axes for better visualization
  theme_minimal()
```



Logistic regression model

```
# Build the logistic regression model
model <- glm(cbind(wins, losses) ~ win_rate, family = binomial, data =
team_stats)
```

Analyzing the Data using Two Teams

This code extracts the win rates for Chennai Super Kings (CSK) and Mumbai Indians (MI) from the team_stats dataframe and assigns them to variables csk_win_rate and mi_win_rate, respectively. It then prepares this data for prediction by creating a new dataframe called prediction_data, which contains the win rates of CSK and MI. These win rates can be used for further analysis or predictive modeling

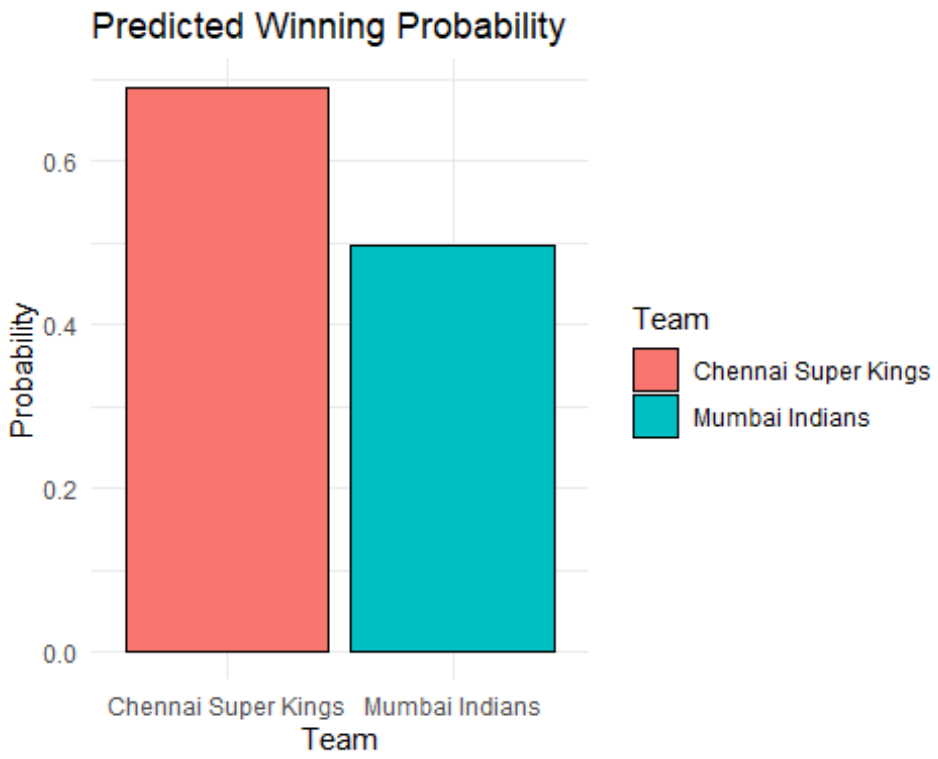
```
# Extract win rates for Chennai Super Kings and Mumbai Indians
csk_win_rate <- team_stats$win_rate[team_stats$team == "Chennai Super Kings"]
mi_win_rate <- team_stats$win_rate[team_stats$team == "Mumbai Indians"]

# Prepare data for prediction
prediction_data <- data.frame(win_rate = c(csk_win_rate, mi_win_rate))
```

Predicting the outcomes

```r
# Predict the outcome
probabilities <- predict(model, newdata = prediction_data, type = "response")
prob_csk_wins <- probabilities[1]  # Probability that Chennai Super Kings
wins
prob_mi_wins <- probabilities[2]  # Probability that Mumbai Indians wins
# Output the predictions
cat("Probability that Chennai Super Kings wins:", prob_csk_wins, "\n")

## Probability that Chennai Super Kings wins: 0.688951

cat("Probability that Mumbai Indians wins:", prob_mi_wins, "\n")

## Probability that Mumbai Indians wins: 0.4972369
```

Ploting The Result

```r
# Data frame for plotting predictions
prediction_plot_data <- data.frame(
  Team = c("Chennai Super Kings", "Mumbai Indians"),
  Probability = probabilities
)

# Plot predicted probabilities
ggplot(prediction_plot_data, aes(x = Team, y = Probability, fill = Team)) +
  geom_bar(stat = "identity", color = "black") +
  labs(title = "Predicted Winning Probability", x = "Team", y =
"Probability") +
  theme_minimal()
```



Now let's predict for the entire Teams

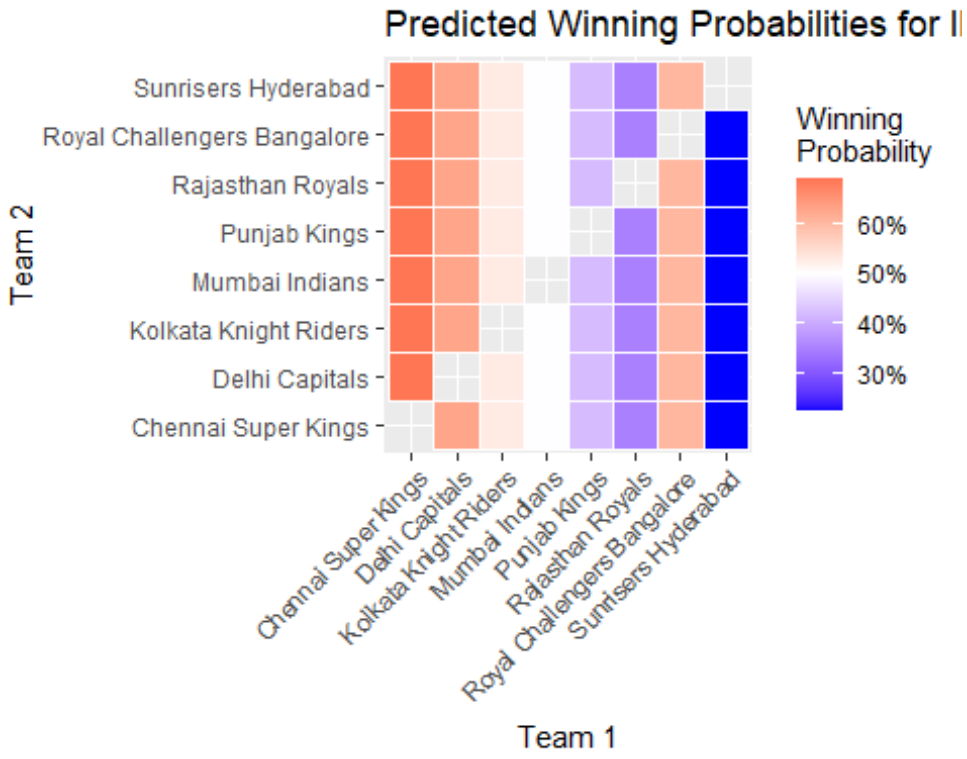
```
# Transform data to a wider format and calculate win rate
team_stats <- team_performance %>%
  spread(key = Outcome, value = Count) %>%
  mutate(win_rate = wins / (wins + losses))

# Build the logistic regression model
model <- glm(cbind(wins, losses) ~ win_rate, family = binomial, data =
team_stats)

# Create all possible matchups
matchups <- expand.grid(team1 = team_stats$team, team2 = team_stats$team,
stringsAsFactors = FALSE)
matchups <- matchups[matchups$team1 != matchups$team2,]  # Remove matchups of
a team against itself

# Add win rates for each team in the matchups
matchups <- matchups %>%
  left_join(team_stats, by = c("team1" = "team")) %>%
  rename(win_rate1 = win_rate) %>%
  left_join(team_stats, by = c("team2" = "team")) %>%
  rename(win_rate2 = win_rate)

# Prepare a new dataset for prediction that includes the win_rate for each
team in the format expected by the model
# The model expects a single 'win_rate' column, so we need to do this
```

```
separately for each team
prediction_data <- with(matchups, data.frame(win_rate = win_rate1))
prediction_data2 <- with(matchups, data.frame(win_rate = win_rate2))

# Predict probabilities for each team in the matchups
matchups$prob_team1_wins <- predict(model, newdata = prediction_data, type =
"response")
matchups$prob_team2_wins <- predict(model, newdata = prediction_data2, type =
"response")
```

Plotting the Result graph

This code generates a heatmap visualizing the predicted winning probabilities for all possible IPL team matchups. Each cell in the heatmap represents a matchup between two teams, with the x-axis indicating "Team 1" and the y-axis indicating "Team 2." The color of each cell corresponds to the predicted probability of "Team 1" winning the matchup, ranging from blue (lower probability) to red (higher probability). The midpoint of the color gradient represents a 50% probability. This visualization allows for a quick assessment of the relative strengths of different IPL teams and their potential outcomes in matchups. Additionally, the x-axis text is angled for better readability.

```
# Plot all matchups and their predicted probabilities
ggplot(matchups, aes(x = team1, y = team2, fill = prob_team1_wins)) +
  geom_tile(color = "white") +  # Use tiles to create a heatmap
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint =
0.5,
                       name = "Winning\nProbability", labels =
scales::percent_format()) +
  labs(title = "Predicted Winning Probabilities for IPL Teams",
       x = "Team 1",
       y = "Team 2") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



I want My Dream Team to win more matches ao have created Dataset for it.

The script selects the top 6 batsmen and bowlers for the Sunrisers Hyderabad based on their performance, removes these top players from the general pool, and then randomly assigns the remaining players to different teams. Finally, it combines all the player assignments, ensuring that Sunrisers Hyderabad retains its top players while other teams receive the rest of the available players.

```
# Select top players for Sunrisers Hyderabad
top_batsmen <- batsman_stats %>% arrange(desc(total_runs)) %>% head(6)
top_bowlers <- bowler_stats %>% arrange(desc(wickets_taken)) %>% head(6)

# Remove these top players from the pool
remaining_batsmen <- setdiff(batsman_stats, top_batsmen)
remaining_bowlers <- setdiff(bowler_stats, top_bowlers)

# Create a function to randomly assign players to teams
assign_players_to_teams <- function(players, teams, column) {
  num_players <- nrow(players)
  if (num_players > 0) {
    sample_teams <- sample(teams, num_players, replace = TRUE)
    return(data.frame(Player = players[[column]], Team = sample_teams))
  } else {
    return(data.frame(Player = character(0), Team = character(0)))
  }
}
```

```r
# Define the teams
teams <- c("Chennai Super Kings", "Delhi Capitals", "Kolkata Knight Riders",
"Mumbai Indians", "Punjab Kings", "Rajasthan Royals", "Royal Challengers
Bangalore", "Sunrisers Hyderabad")

# Assign remaining players to teams
batsmen_assignments <- assign_players_to_teams(remaining_batsmen, teams,
"Batsman")
bowlers_assignments <- assign_players_to_teams(remaining_bowlers, teams,
"Bowler")

# Combine top players into Sunrisers Hyderabad
sunrisers_batsmen <- data.frame(Player = top_batsmen$Batsman, Team =
rep("Sunrisers Hyderabad", nrow(top_batsmen)))
sunrisers_bowlers <- data.frame(Player = top_bowlers$Bowler, Team =
rep("Sunrisers Hyderabad", nrow(top_bowlers)))

# Combine all player assignments
final_batsmen_assignments <- rbind(sunrisers_batsmen, batsmen_assignments)
final_bowlers_assignments <- rbind(sunrisers_bowlers, bowlers_assignments)
```

simple aggregation model

The code defines a function to estimate the performance of a cricket team by calculating the total runs scored and wickets taken by its players. It specifically measures these statistics for the Sunrisers Hyderabad team, providing a basic indicator of the team's offensive and defensive capabilities. This simple model helps understand the team's potential strength in a straightforward way.

```r
# Simple model to predict winning percentage
# This part is highly simplified and just a placeholder
winning_percentage <- function(team_name) {
  total_runs_scored <- sum(batsman_stats$total_runs[batsman_stats$Batsman
%in% final_batsmen_assignments$Player[final_batsmen_assignments$Team ==
team_name]])
  total_wickets_taken <- sum(bowler_stats$wickets_taken[bowler_stats$Bowler
%in% final_bowlers_assignments$Player[final_bowlers_assignments$Team ==
team_name]])
  return(list(runs = total_runs_scored, wickets = total_wickets_taken))
}

# Calculate for Sunrisers Hyderabad as an example
sunrisers_performance <- winning_percentage("Sunrisers Hyderabad")
print(sunrisers_performance)
```

```
## $runs
## [1] 4823
##
```

```
## $wickets
## [1] 197
```

comparative simulation model

This script is designed to simulate a series of cricket matches based on basic team statistics such as total runs scored and wickets taken. It begins by compiling these statistics for each team, providing a measure of their offensive and defensive strengths. Using these metrics, the script sets up potential matches between all teams, ensuring that no team plays against itself. It then predicts the outcomes of these matches by comparing the ratio of runs scored to wickets taken for each team against their opponents. The team with the higher ratio in any given match is predicted to win. After simulating all the matches, the script visually represents the results in a bar chart, which shows the number of wins each team accumulates. This visualization helps to quickly identify which teams are predicted to perform better over the season, based on their calculated offensive and defensive capabilities.

```
# Create aggregated statistics for each team
team_stats <- data.frame(
  Team = unique(final_batsmen_assignments$Team),
  Total_Runs = sapply(unique(final_batsmen_assignments$Team), function(t) {
    sum(batsman_stats$total_runs[batsman_stats$Batsman %in%
final_batsmen_assignments$Player[final_batsmen_assignments$Team == t]])
  }),
  Total_Wickets = sapply(unique(final_bowlers_assignments$Team), function(t)
{
    sum(bowler_stats$wickets_taken[bowler_stats$Bowler %in%
final_bowlers_assignments$Player[final_bowlers_assignments$Team == t]])
  })
)

# Ensure team_stats is correctly populated
print(team_stats)
```

```
##                                                    Team Total_Runs
## Sunrisers Hyderabad                 Sunrisers Hyderabad       4823
## Rajasthan Royals                       Rajasthan Royals       3046
## Chennai Super Kings                 Chennai Super Kings        977
## Mumbai Indians                           Mumbai Indians        434
## Royal Challengers Bangalore Royal Challengers Bangalore       3190
## Kolkata Knight Riders             Kolkata Knight Riders       1305
## Delhi Capitals                           Delhi Capitals       1238
## Punjab Kings                               Punjab Kings       2703
##                             Total_Wickets
## Sunrisers Hyderabad                   197
## Rajasthan Royals                       74
## Chennai Super Kings                    84
## Mumbai Indians                         99
## Royal Challengers Bangalore            86
## Kolkata Knight Riders                  70
```

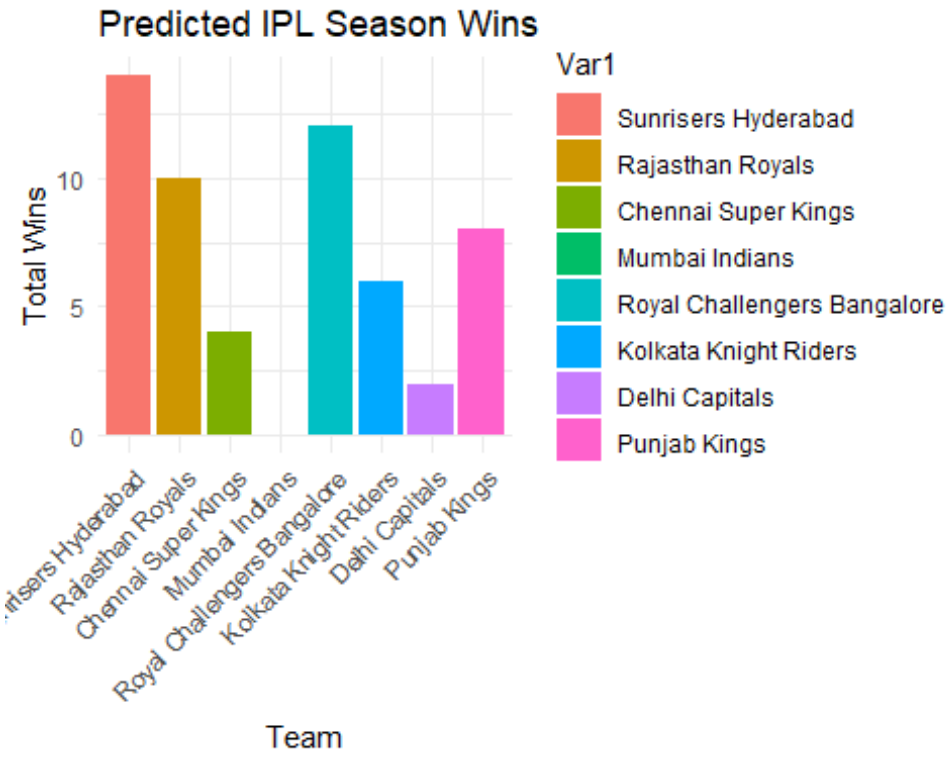
```
## Delhi Capitals                          66
## Punjab Kings                            41
```

```r
# Generate all possible matches excluding self-matches
matches <- expand.grid(Home = team_stats$Team, Away = team_stats$Team)
matches <- matches[matches$Home != matches$Away,]

# Predict outcomes based on a simple comparative logic
predict_match_outcome <- function(home, away) {
  home_stats <- team_stats[team_stats$Team == home,]
  away_stats <- team_stats[team_stats$Team == away,]
  home_score <- home_stats$Total_Runs / away_stats$Total_Wickets
  away_score <- away_stats$Total_Runs / home_stats$Total_Wickets

  if (home_score > away_score) home else away
}

# Simulate the outcomes
matches$Winner <- mapply(predict_match_outcome, matches$Home, matches$Away)

# Check results
print(head(matches))
```

```
##                          Home                Away              Winner
## 2           Rajasthan Royals Sunrisers Hyderabad Sunrisers Hyderabad
## 3        Chennai Super Kings Sunrisers Hyderabad Sunrisers Hyderabad
## 4             Mumbai Indians Sunrisers Hyderabad Sunrisers Hyderabad
## 5 Royal Challengers Bangalore Sunrisers Hyderabad Sunrisers Hyderabad
## 6      Kolkata Knight Riders Sunrisers Hyderabad Sunrisers Hyderabad
## 7             Delhi Capitals Sunrisers Hyderabad Sunrisers Hyderabad
```

Plotting The Result

```r
library(ggplot2)

# Count wins per team
win_counts <- table(matches$Winner)

# Create a bar plot of win counts
ggplot(data = as.data.frame(win_counts), aes(x = Var1, y = Freq, fill =
Var1)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(title = "Predicted IPL Season Wins", x = "Team", y = "Total Wins") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```


Predicted IPL Season Wins
Total Wins
10
5
0
Team
Rajasthan Royals
Chennai Super Kings
Mumbai Indians
Royal Challengers Bangalore
Kolkata Knight Riders
Delhi Capitals
Punjab Kings
Var1
Sunrisers Hyderabad
Rajasthan Royals
Chennai Super Kings
Mumbai Indians
Royal Challengers Bangalore
Kolkata Knight Riders
Delhi Capitals
Punjab Kings