Pakistan Super League Exploratory Data Analysis

For the purpose of this exercise, we'll deploy the very handy cricketdata package developed by Rob J Hyndman, which gathers data from cricsheet and cricinfo.

Loading Libraries

```
library(tidyverse)
library(cricketdata)
library(dplyr)
library(ggplot2)
library(plotly)
```

Loading Ball by Ball and Match Data for PSL 2016-Present from Cricsheet

```
PSL_Ball <- fetch_cricsheet(competition = "psl", gender = "male")

PSL_Match <- fetch_cricsheet("match", "psl", gender = "male")

PSL_Player <- fetch_cricsheet("player", "psl", gender = "male")
```

Understanding the Structure

```
str(PSL_Ball)
```

```
## tibble [63,451 × 33] (S3: tbl_df/tbl/data.frame)
##  $ match_id              : int [1:63451] 1075986 1075986 1075986 1075986 1075986 1075986 1075986 1075986 1075986 1075986 ...
##  $ season                : chr [1:63451] "2016/17" "2016/17" "2016/17" "2016/17" ...
##  $ start_date            : Date[1:63451], format: "2017-02-09" "2017-02-09" ...
##  $ venue                 : chr [1:63451] "Dubai International Cricket Stadium" "Dubai International Cricket Stadium" "Dubai International Cricket Stadium" "D
ubai International Cricket Stadium" ...
##  $ innings               : int [1:63451] 1 1 1 1 1 1 1 1 1 1 ...
##  $ over                  : num [1:63451] 1 1 1 1 1 1 2 2 2 2 ...
##  $ ball                  : int [1:63451] 1 2 3 4 5 6 1 2 3 4 ...
##  $ batting_team          : chr [1:63451] "Peshawar Zalmi" "Peshawar Zalmi" "Peshawar Zalmi" "Peshawar Zalmi" ...
##  $ bowling_team          : chr [1:63451] "Islamabad United" "Islamabad United" "Islamabad United" "Islamabad United" ...
##  $ striker               : chr [1:63451] "Mohammad Hafeez" "Kamran Akmal" "DJ Malan" "DJ Malan" ...
##  $ non_striker           : chr [1:63451] "DJ Malan" "DJ Malan" "Kamran Akmal" "Kamran Akmal" ...
##  $ bowler                : chr [1:63451] "Mohammad Irfan" "Mohammad Irfan" "Mohammad Irfan" "Mohammad Irfan" ...
##  $ runs_off_bat          : int [1:63451] 0 1 0 0 0 0 0 2 0 0 ...
##  $ extras                : int [1:63451] 0 0 0 0 0 0 0 0 0 1 ...
##  $ ball_in_over          : int [1:63451] 1 2 3 4 5 6 1 2 3 4 ...
##  $ extra_ball            : logi [1:63451] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ balls_remaining       : num [1:63451] 119 118 117 116 115 114 113 112 111 110 ...
##  $ runs_scored_yet       : int [1:63451] 0 1 1 1 1 1 1 3 3 4 ...
##  $ wicket                : logi [1:63451] TRUE FALSE FALSE FALSE FALSE FALSE ...
##  $ wickets_lost_yet      : int [1:63451] 1 1 1 1 1 1 1 1 1 1 ...
##  $ innings1_total        : int [1:63451] 190 190 190 190 190 190 190 190 190 190 ...
##  $ innings2_total        : int [1:63451] 175 175 175 175 175 175 175 175 175 175 ...
##  $ target                : num [1:63451] 191 191 191 191 191 191 191 191 191 191 ...
##  $ wides                 : int [1:63451] NA NA NA NA NA NA NA NA NA NA ...
##  $ noballs               : int [1:63451] NA NA NA NA NA NA NA NA NA NA ...
##  $ byes                  : int [1:63451] NA NA NA NA NA NA NA NA NA NA ...
##  $ legbyes               : int [1:63451] NA NA NA NA NA NA NA NA NA 1 ...
##  $ penalty               : int [1:63451] NA NA NA NA NA NA NA NA NA NA ...
##  $ wicket_type           : chr [1:63451] "caught" "" "" "" ...
##  $ player_dismissed      : chr [1:63451] "Mohammad Hafeez" "" "" "" ...
##  $ other_wicket_type     : logi [1:63451] NA NA NA NA NA NA ...
##  $ other_player_dismissed: logi [1:63451] NA NA NA NA NA NA ...
##  $ .groups               : chr [1:63451] "drop" "drop" "drop" "drop" ...
```

```
str(PSL_Match)
```

```
## tibble [269 × 25] (S3: tbl_df/tbl/data.frame)
##  $ match_id       : chr [1:269] "1075986" "1075988" "1075995" "1075997" ...
##  $ balls_per_over : chr [1:269] "6" "6" "6" "6" ...
##  $ team1          : chr [1:269] "Islamabad United" "Karachi Kings" "Islamabad United" "Islamabad United" ...
##  $ team2          : chr [1:269] "Peshawar Zalmi" "Peshawar Zalmi" "Karachi Kings" "Peshawar Zalmi" ...
##  $ gender         : chr [1:269] "male" "male" "male" "male" ...
##  $ season         : chr [1:269] "2016/17" "2016/17" "2016/17" "2016/17" ...
##  $ date           : chr [1:269] "2017/02/09" "2017/02/10" "2017/02/17" "2017/02/18" ...
##  $ event          : chr [1:269] "Pakistan Super League" "Pakistan Super League" "Pakistan Super League" "Pakistan Super League" ...
##  $ match_number   : chr [1:269] "1" "3" "10" "12" ...
##  $ venue          : chr [1:269] "Dubai International Cricket Stadium" "Dubai International Cricket Stadium" "Sharjah Cricket Stadium" "Sharjah Cricket Stad
ium" ...
##  $ city           : chr [1:269] NA NA NA NA ...
##  $ toss_winner    : chr [1:269] "Islamabad United" "Peshawar Zalmi" "Karachi Kings" "Islamabad United" ...
##  $ toss_decision  : chr [1:269] "field" "field" "field" "field" ...
##  $ player_of_match: chr [1:269] "BJ Haddin" "EJG Morgan" "Babar Azam" "Mohammad Sami" ...
##  $ umpire1        : chr [1:269] "Ahsan Raza" "Ahmed Shahab" "Aleem Dar" "Aleem Dar" ...
##  $ umpire2        : chr [1:269] "Shozab Raza" "Ahsan Raza" "RK Illingworth" "RK Illingworth" ...
##  $ reserve_umpire : chr [1:269] "Asif Yaqoob" "Shozab Raza" "Ahsan Raza" "Rashid Riaz" ...
##  $ tv_umpire      : chr [1:269] "Rashid Riaz" "Asif Yaqoob" "Shozab Raza" "Asif Yaqoob" ...
##  $ match_referee  : chr [1:269] "RS Mahanama" "RS Mahanama" "RS Mahanama" "Mohammed Anees" ...
##  $ winner         : chr [1:269] "Islamabad United" "Peshawar Zalmi" "Karachi Kings" "Islamabad United" ...
##  $ winner_wickets : chr [1:269] "7" "7" NA "5" ...
##  $ method         : chr [1:269] "D/L" NA "D/L" NA ...
##  $ winner_runs    : chr [1:269] NA NA "8" NA ...
##  $ outcome        : chr [1:269] NA NA NA NA ...
##  $ eliminator     : chr [1:269] NA NA NA NA ...
```

```
str(PSL_Player)
```

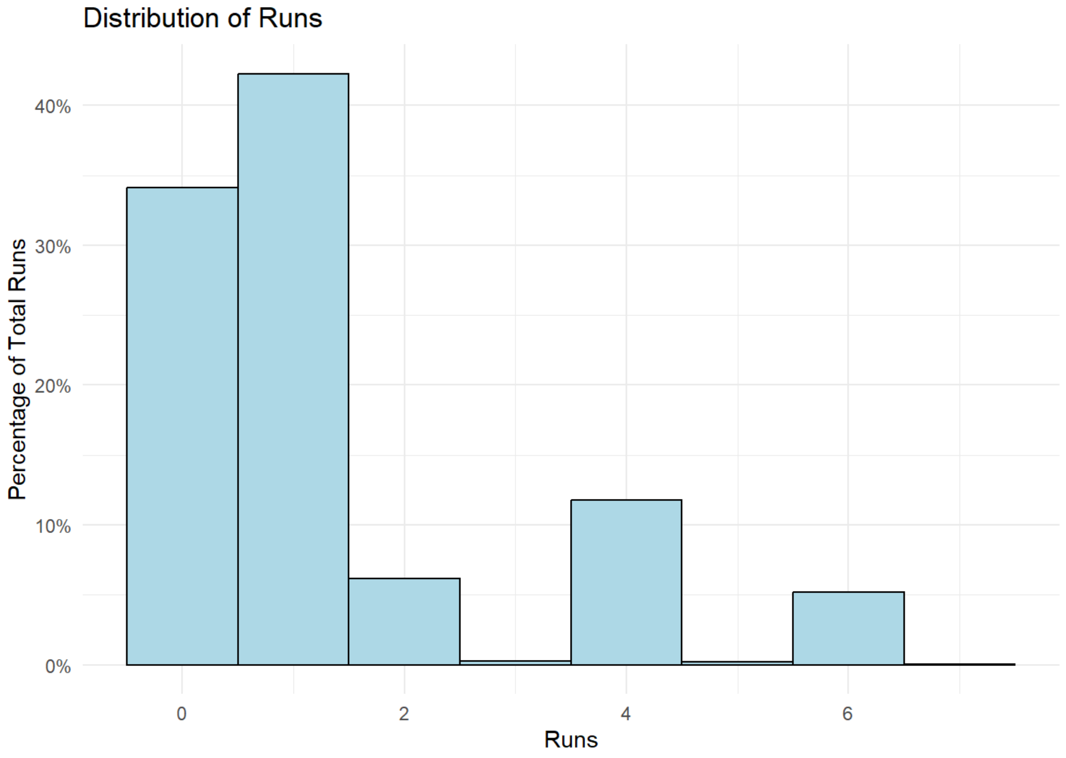
```
## tibble [5,921 × 3] (S3: tbl_df/tbl/data.frame)
##  $ team    : chr [1:5921] "Islamabad United" "Islamabad United" "Islamabad United" "Islamabad United" ...
##  $ player  : chr [1:5921] "DR Smith" "Sharjeel Khan" "BJ Haddin" "SR Watson" ...
##  $ match_id: chr [1:5921] "1075986" "1075986" "1075986" "1075986" ...
```

Merging and Cleaning Match and Ball-by-Ball Data

Summary Statistics

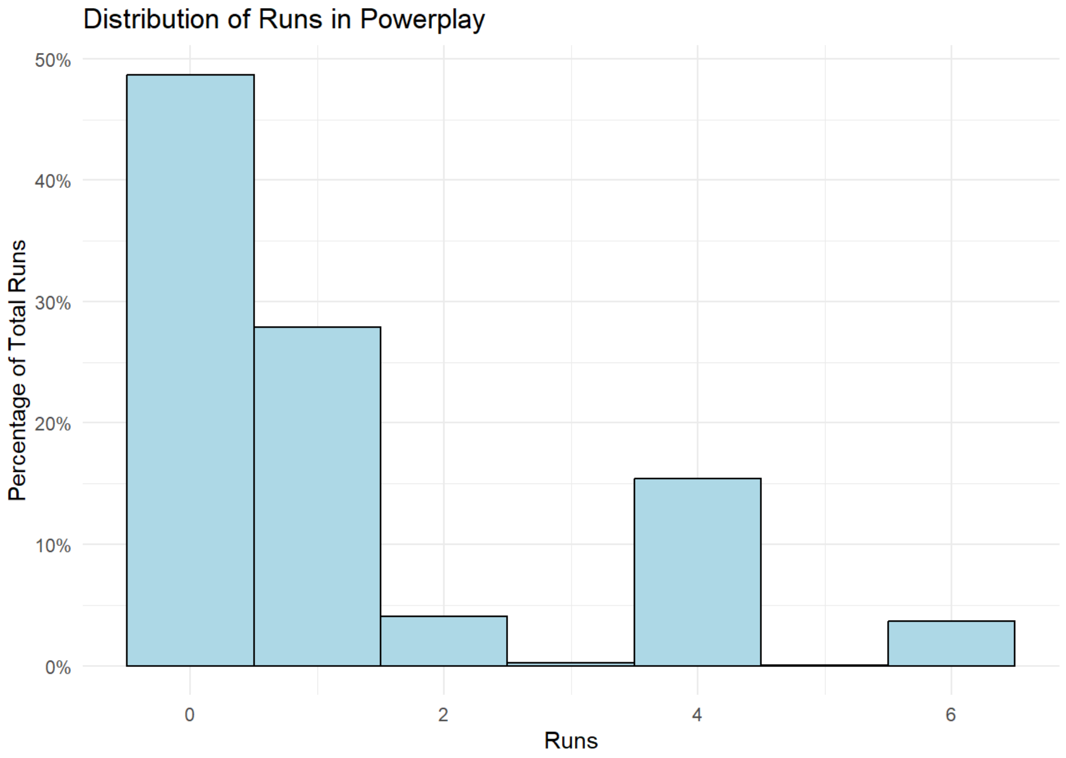
Distribution of Runs Overall

```
# Distribution of runs in Ball by Ball Data
ggplot(Merged_Data, aes(x = runs_off_bat + extras)) +
  geom_histogram(aes(y = after_stat(count) / sum(after_stat(count))), binwidth = 1, fill = "lightblue", color = "black") +
  scale_y_continuous(labels = scales::percent, name = "Percentage of Total Runs") +
  labs(title = "Distribution of Runs", x = "Runs") +
  theme_minimal()
```



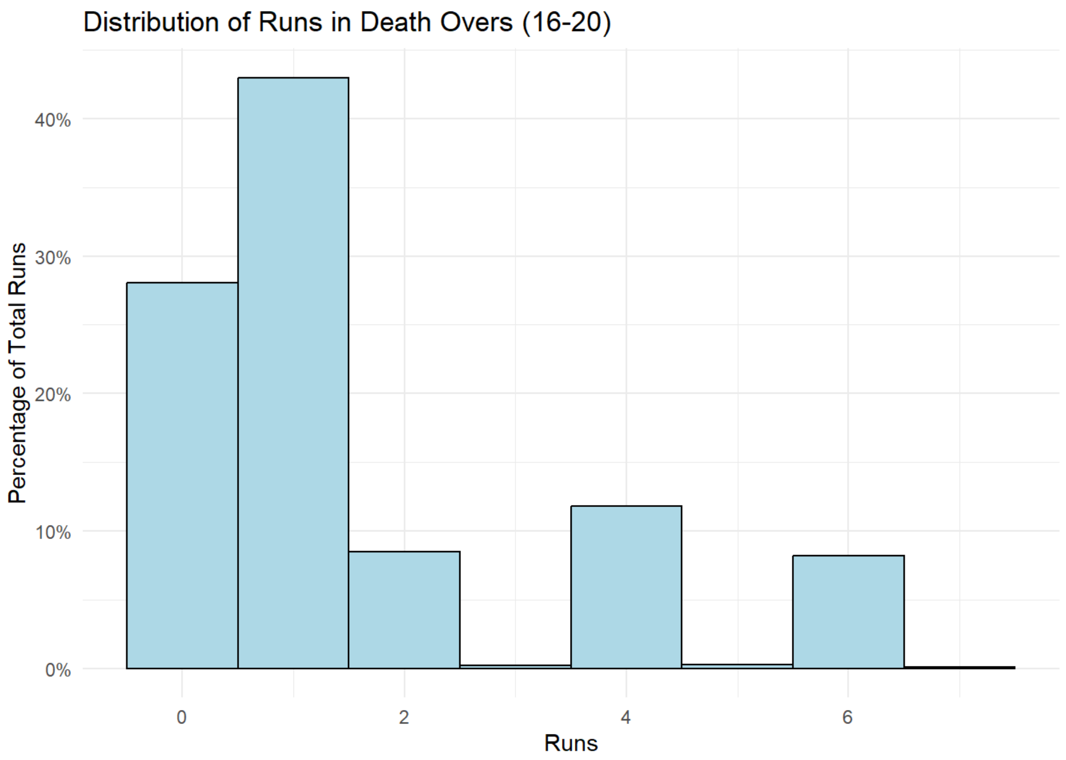
Distribution of Runs in Powerplay

```
# Filter for powerplay
powerplay <- Merged_Data |>
  filter((over >= 1 & over <= 6))

# Plot histogram of runs off bat during powerplay
ggplot(powerplay, aes(x = runs_off_bat)) +
  geom_histogram(aes(y = after_stat(count) / sum(after_stat(count))), binwidth = 1, fill = "lightblue", color = "black") +
  scale_y_continuous(labels = scales::percent, name = "Percentage of Total Runs") +
  labs(title = "Distribution of Runs in Powerplay", x = "Runs") +
  theme_minimal()
```



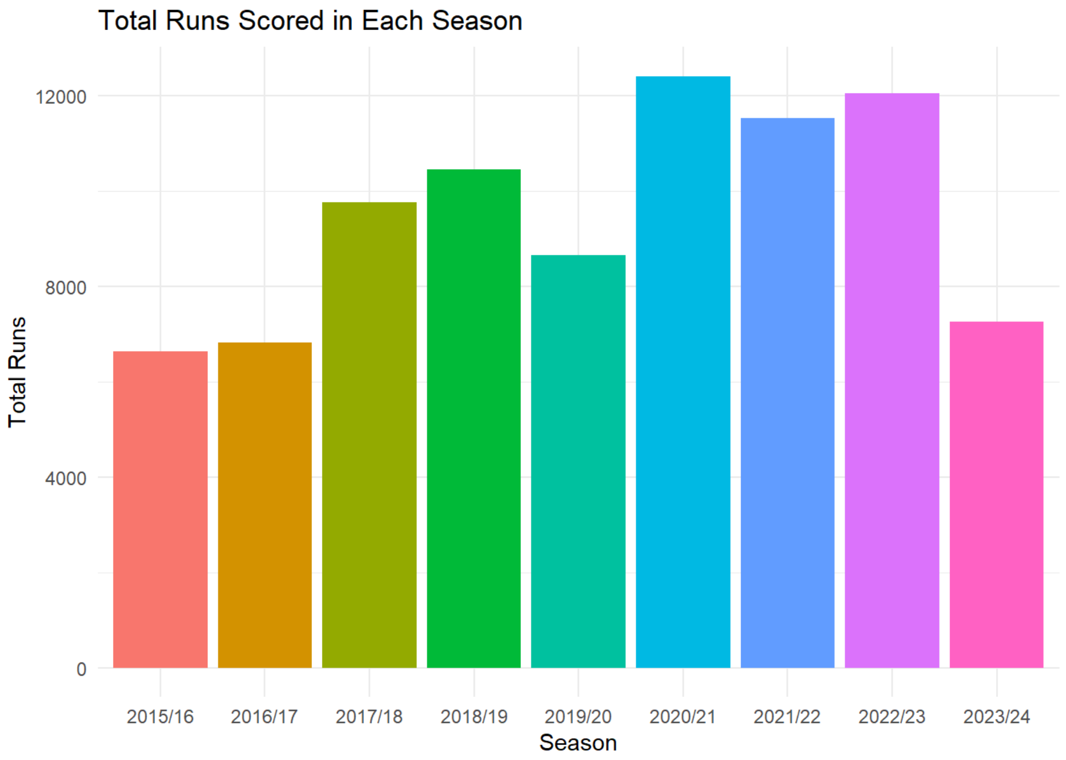
Distribution of Runs in Death Overs (16-20)

```r
# Filter for death overs
death_overs <- Merged_Data |>
  filter((over >= 16 & over <= 20))

# Plot histogram of runs off bat during death overs
ggplot(death_overs, aes(x = runs_off_bat + extras)) +
  geom_histogram(aes(y = after_stat(count) / sum(after_stat(count))), binwidth = 1, fill = "lightblue", color = "black") +
  scale_y_continuous(labels = scales::percent, name = "Percentage of Total Runs") +
  labs(title = "Distribution of Runs in Death Overs (16-20)", x = "Runs") +
  theme_minimal()
```



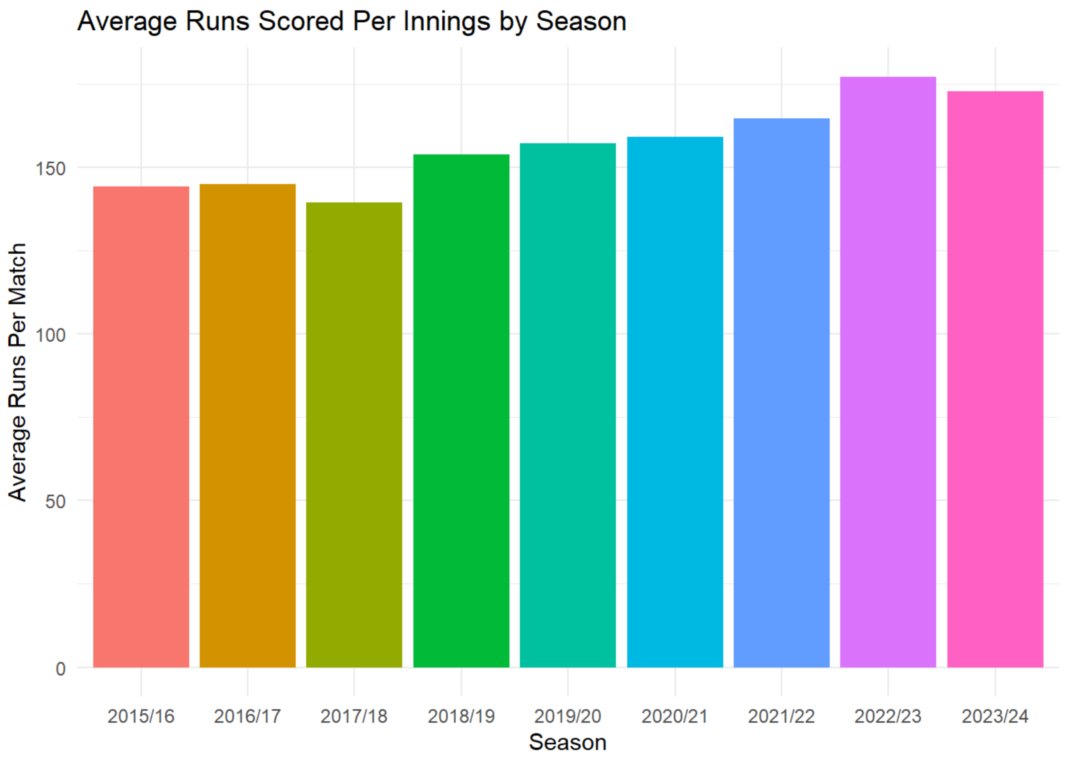
Total Runs Scored by Season

```r
Total_Runs_Season <- Merged_Data |>
  group_by(season.x) |>
  summarise(Total_Runs = sum(runs_off_bat + extras))

ggplot(Total_Runs_Season, aes(x = season.x, y = Total_Runs, fill = season.x)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  labs(title = "Total Runs Scored in Each Season", x = "Season", y = "Total Runs") +
  theme_minimal()
```



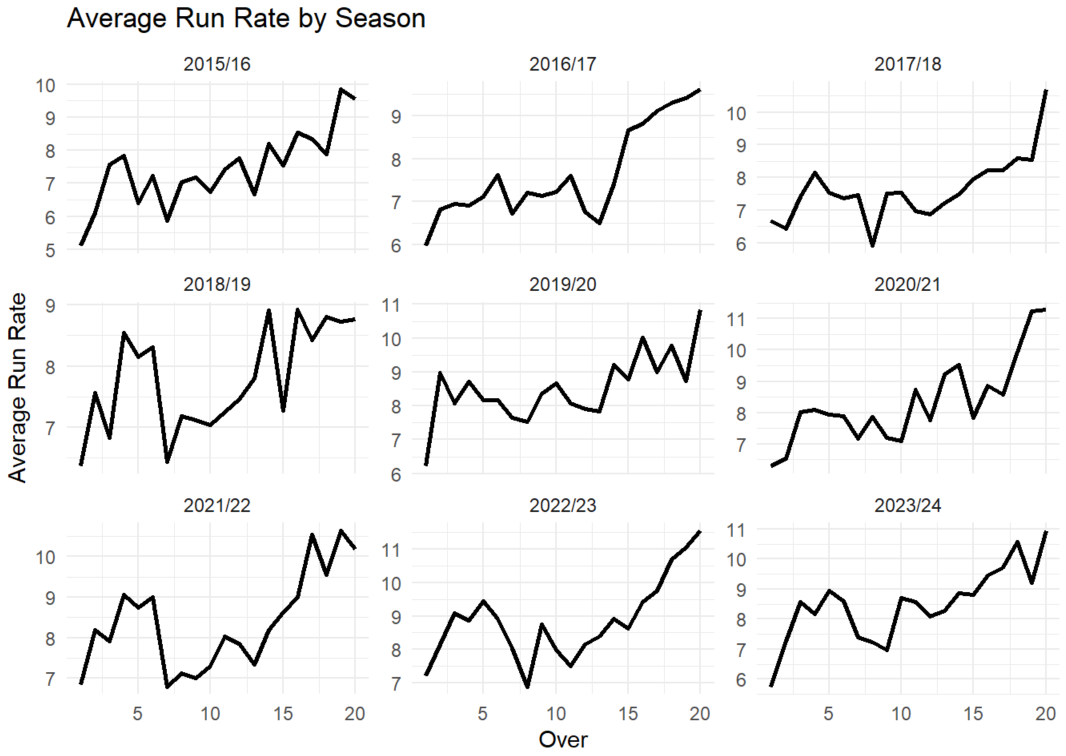
Average Runs Scored Per Innings by Season

```
Avg_Runs_Per_Match_Season <- Merged_Data |>
  group_by(season.x, match_id, innings) |>
  summarise(Total_Runs = sum(runs_off_bat + extras), .groups = 'drop') |>
  group_by(season.x) |>
  summarise(Avg_Runs_Per_Match = mean(Total_Runs))

ggplot(Avg_Runs_Per_Match_Season, aes(x = season.x, y = Avg_Runs_Per_Match, fill = season.x)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  labs(title = "Average Runs Scored Per Innings by Season", x = "Season", y = "Average Runs Per Match") +
  theme_minimal()
```



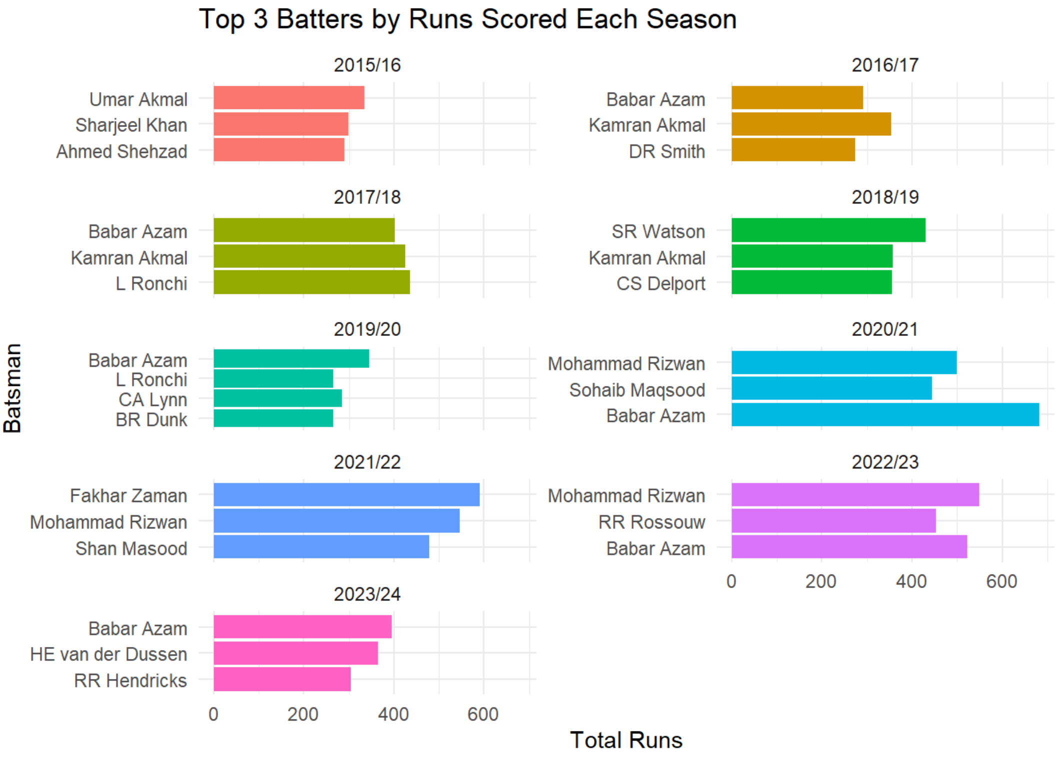
Average Run Rate by Season

```
Average_Run_Rate <- Merged_Data |>
  group_by(season.x, over) |>
  summarise(
    Total_Runs = sum(runs_off_bat + extras, na.rm = TRUE),  # Calculate total runs
    Total_Balls = n(),  # Count total deliveries (rows)
    Total_Overs = Total_Balls / 6.0,  # Convert balls to overs
    Avg_Run_Rate = Total_Runs / Total_Overs, #Calculate Average Run Rate
    .groups = 'drop'
  )

# Step 4: Visualize the average run rate per over for each season
ggplot(Average_Run_Rate, aes(x = over, y = Avg_Run_Rate)) +
  geom_line(linewidth = 1) +
  facet_wrap(~season.x, scales = "free_y") + # Creates a separate plot for each season
  labs(title = "Average Run Rate by Season", x = "Over", y = "Average Run Rate") +
  theme_minimal() +
  theme(legend.position = "bottom")
```



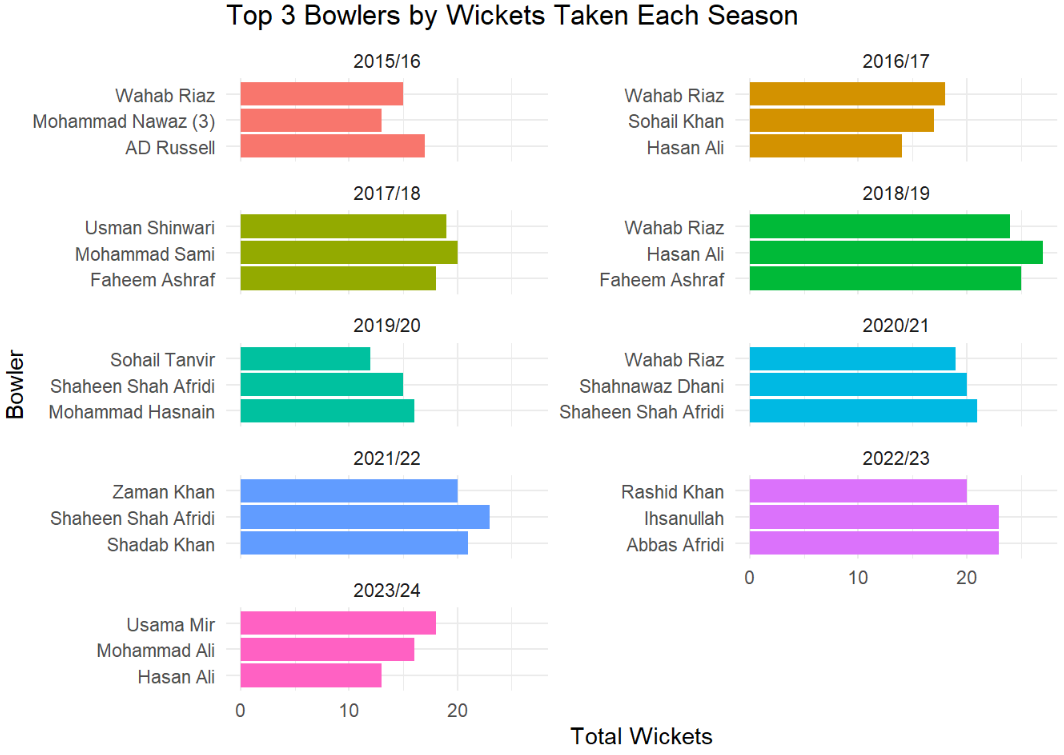
Top Batters

```
Top_Batsmen <- Merged_Data |>
  group_by(season.x, striker) |>
  summarise(Total_Runs = sum(runs_off_bat, na.rm = TRUE), .groups = 'drop') |>
  group_by(season.x) |>
  slice_max(order_by = Total_Runs, n = 3) |>
  ungroup()
ggplot(Top_Batsmen, aes(x = reorder(striker, Total_Runs), y = Total_Runs, fill = season.x)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Top 3 Batters by Runs Scored Each Season", x = "Batsman", y = "Total Runs") +
  theme_minimal() +
  theme(legend.position = "none") +
  facet_wrap(~ season.x, scales = "free_y", ncol = 2)
```



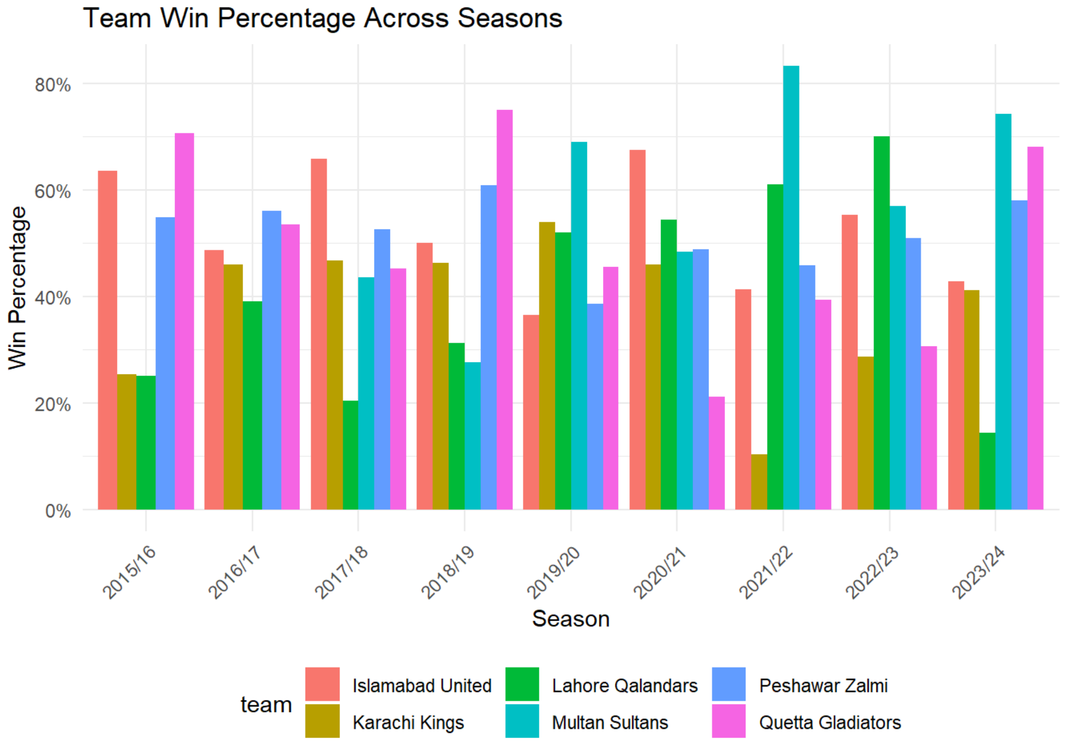
Top Bowlers

```
Top_Bowlers <- Merged_Data |>
  group_by(season.x, bowler) |>
  summarise(Total_Wickets = sum(as.numeric(wicket), na.rm = TRUE), .groups = 'drop_last') |>
  slice_max(order_by = Total_Wickets, n = 3, with_ties = FALSE) |>
  ungroup()

ggplot(Top_Bowlers, aes(x = bowler, y = Total_Wickets, fill = season.x)) +
  geom_col() +
  coord_flip() +
  labs(title = "Top 3 Bowlers by Wickets Taken Each Season", x = "Bowler", y = "Total Wickets") +
  theme_minimal() +
  theme(legend.position = "None") +
  facet_wrap(~ season.x, scales = "free_y", ncol = 2)
```



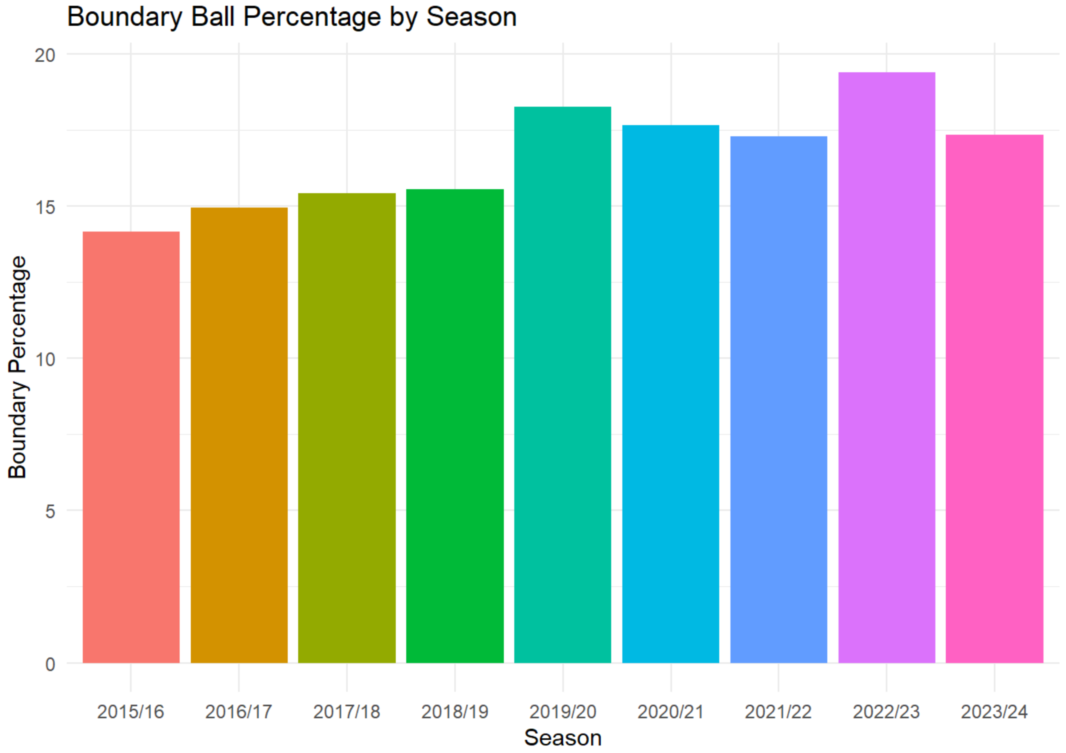
Team Performance Across Seasons

```
# Calculate total matches and wins per team per season
team_stats <- Merged_Data |>
  select(season.x, team1, team2, winner) |>
  mutate(match_played = 1) |>
  pivot_longer(cols = c(team1, team2), names_to = "home_away", values_to = "team") %>%
  group_by(season.x, team) |>
  summarise(Total_Matches = sum(match_played), Wins = sum(winner == team, na.rm = TRUE), .groups = 'drop') |>
  ungroup()

# Calculate win percentage
team_stats <- team_stats |>
  mutate(Win_Percentage = (Wins / Total_Matches) * 100)

ggplot(team_stats, aes(x = season.x, y = Win_Percentage, fill = team)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Team Win Percentage Across Seasons", x = "Season", y = "Win Percentage") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), legend.position = "bottom") +
  scale_y_continuous(labels = function(x) paste0(x, "%"))
```



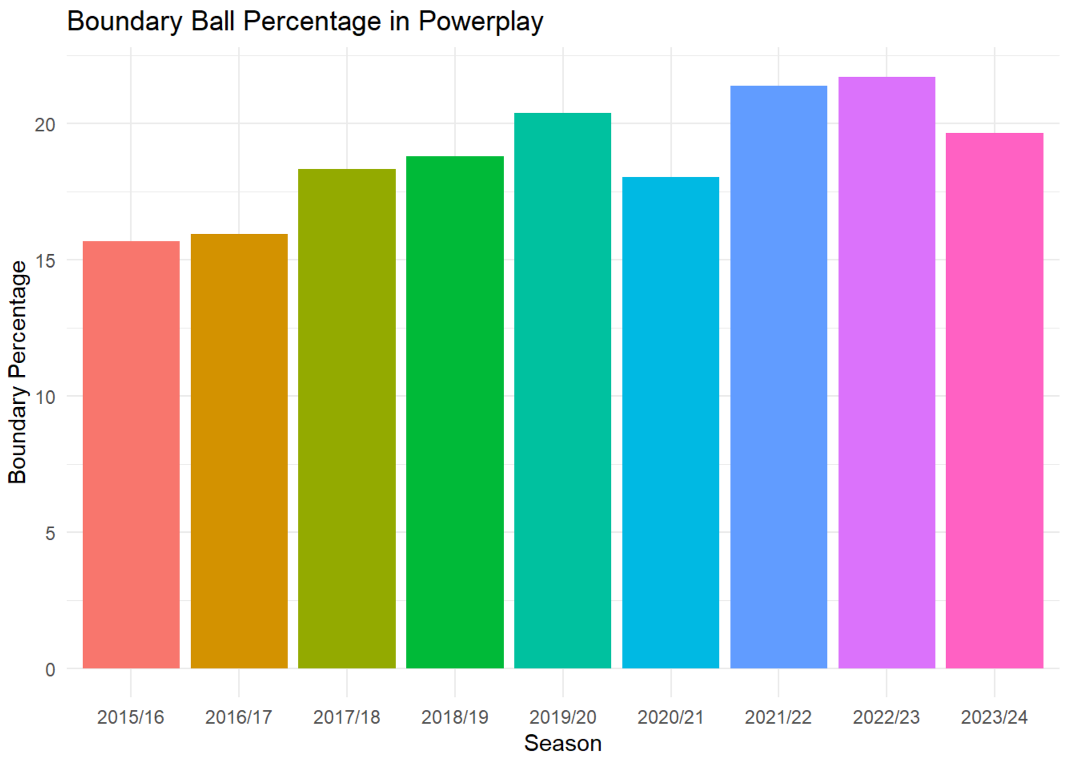
Boundary Balls

```r
boundary_balls <- Merged_Data |>
  filter(runs_off_bat %in% c(4, 6)) |>
  mutate(boundary = 1) # Mark boundary balls

# Step 2: Calculate Boundary Ball Percentage
boundary_percentage_by_season <- Merged_Data |>
  group_by(season.x) |>
  summarise(total_balls = n(), # Total number of deliveries
            boundaries = sum(runs_off_bat %in% c(4, 6)), # Number of boundary balls
            boundary_percentage = (boundaries / total_balls) * 100) # Calculate percentage

# Step 3: Visualization
ggplot(boundary_percentage_by_season, aes(x = season.x, y = boundary_percentage, fill = season.x)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  labs(title = "Boundary Ball Percentage by Season",
       x = "Season",
       y = "Boundary Percentage") +
  theme_minimal()
```



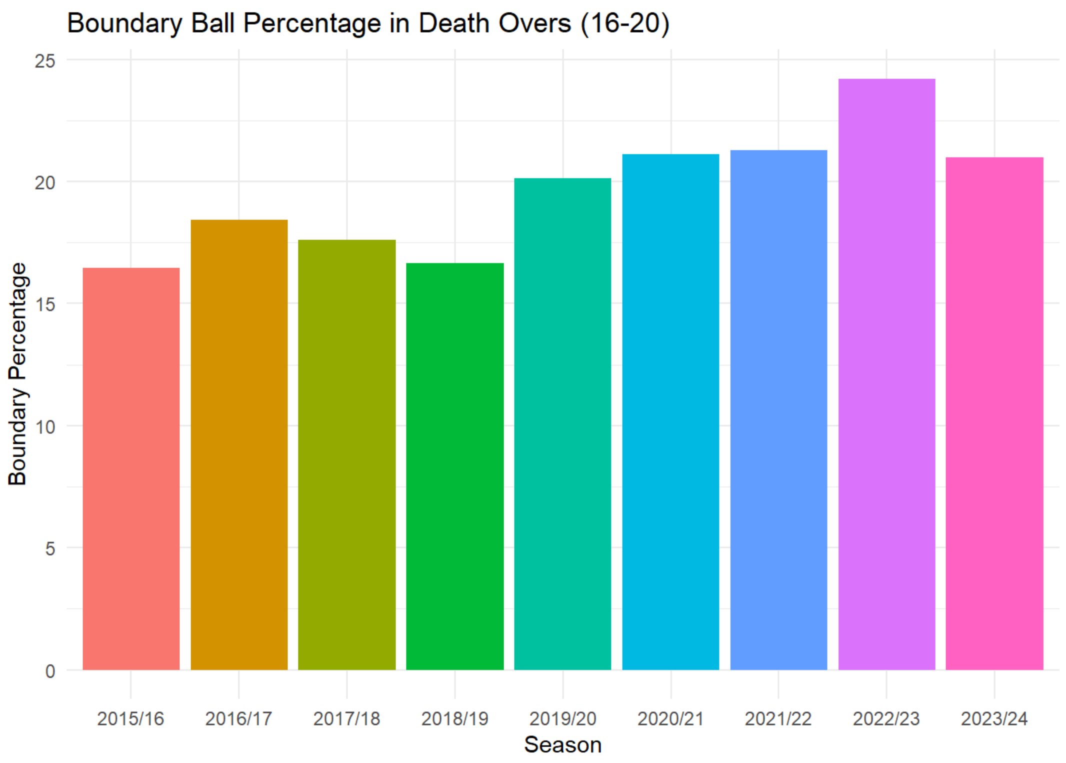
Boundary Balls in Powerplay

```
boundary_balls <- powerplay |>
  filter(runs_off_bat %in% c(4, 6)) |>
  mutate(boundary = 1) # Mark boundary balls

# Step 2: Calculate Boundary Ball Percentage
boundary_percentage_by_season <- powerplay |>
  group_by(season.x) |>
  summarise(total_balls = n(), # Total number of deliveries
            boundaries = sum(runs_off_bat %in% c(4, 6)), # Number of boundary balls
            boundary_percentage = (boundaries / total_balls) * 100) # Calculate percentage

# Step 3: Visualization
  ggplot(boundary_percentage_by_season, aes(x = season.x, y = boundary_percentage,   fill = season.x)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  labs(title = "Boundary Ball Percentage in Powerplay",
       x = "Season",
       y = "Boundary Percentage") +
  theme_minimal()
```



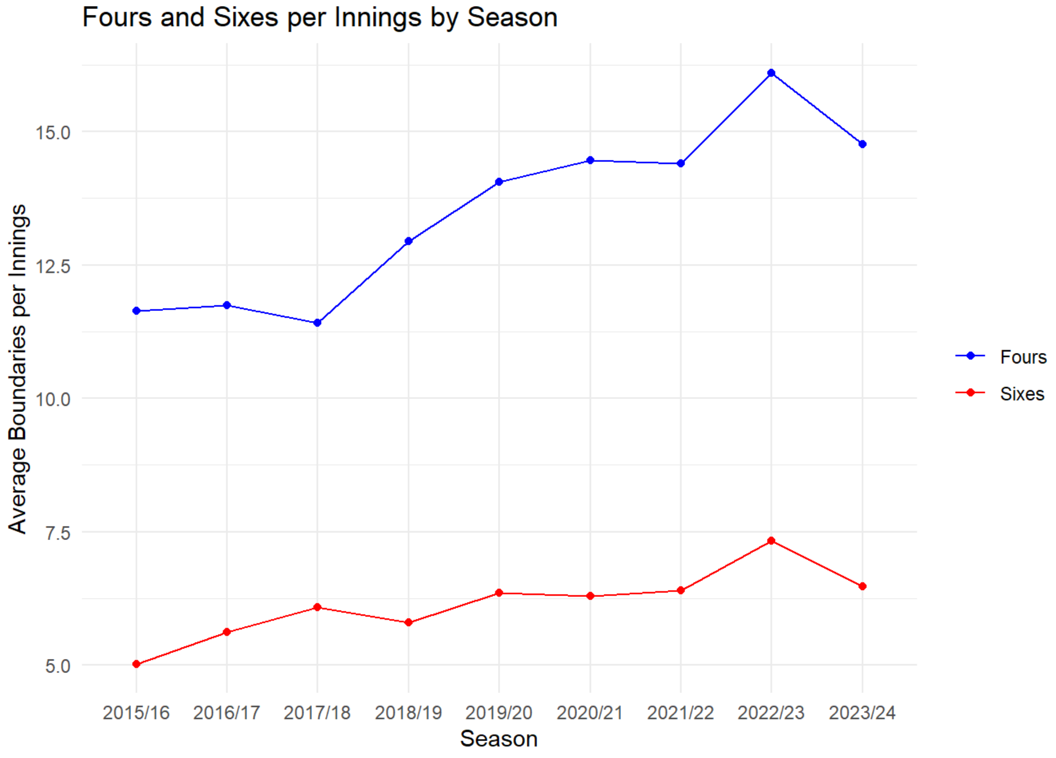
Boundary Balls in Death Overs (16-20)

```
boundary_balls <- death_overs |>
  filter(runs_off_bat %in% c(4, 6)) |>
  mutate(boundary = 1) # Mark boundary balls

# Step 2: Calculate Boundary Ball Percentage
boundary_percentage_by_season <- death_overs |>
  group_by(season.x) |>
  summarise(total_balls = n(), # Total number of deliveries
            boundaries = sum(runs_off_bat %in% c(4, 6)), # Number of boundary balls
            boundary_percentage = (boundaries / total_balls) * 100) # Calculate percentage

# Step 3: Visualization
  ggplot(boundary_percentage_by_season, aes(x = season.x, y = boundary_percentage,   fill = season.x)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  labs(title = "Boundary Ball Percentage in Death Overs (16-20)",
       x = "Season",
       y = "Boundary Percentage") +
  theme_minimal()
```



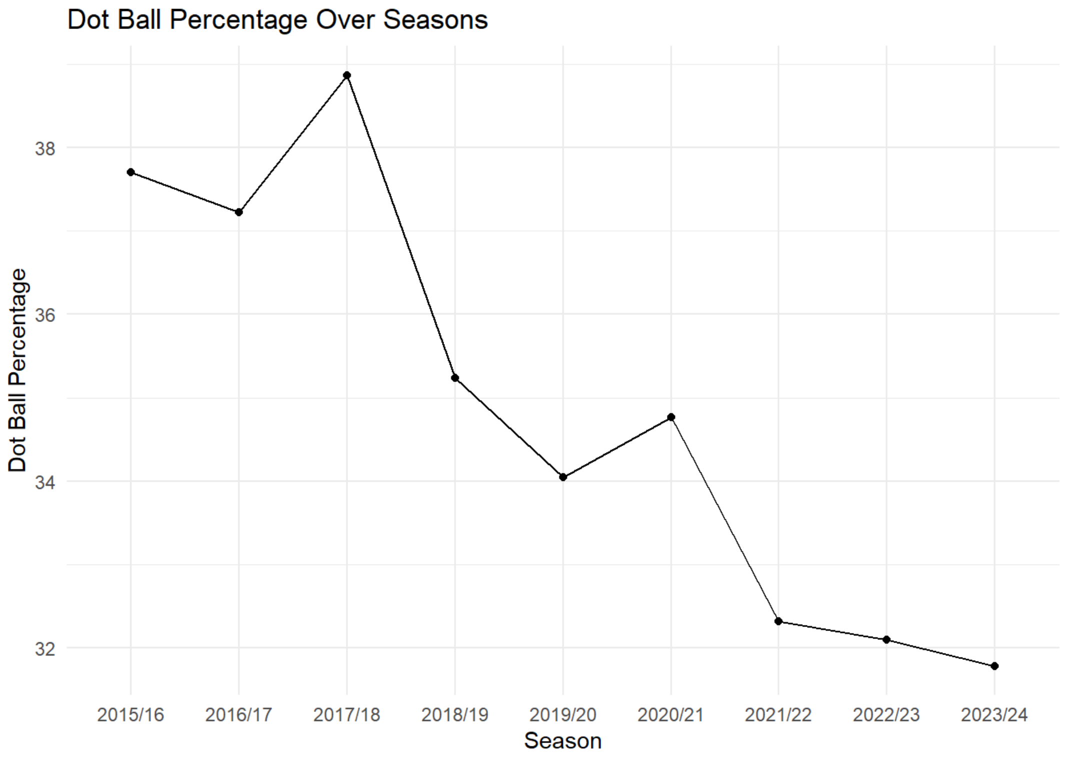
Boundaries per Innings

```r
# Step 1: Prepare the data
boundaries_per_innings <- Merged_Data |>
  filter(runs_off_bat %in% c(4, 6)) |>
  mutate(boundary_type = ifelse(runs_off_bat == 4, "Fours", "Sixes")) |>
  group_by(season.x, match_id, innings, boundary_type) |>
  summarise(total_boundaries = n(), .groups = 'drop') |>
  group_by(season.x, boundary_type) |>
  summarise(avg_boundaries_per_innings = mean(total_boundaries), .groups = 'drop')

# Step 2: Visualize the data
ggplot(boundaries_per_innings, aes(x = season.x, y = avg_boundaries_per_innings, color = boundary_type, group = boundary_type)) +
  geom_line() +  # Line plot
  geom_point() +
  scale_color_manual(values = c("Fours" = "blue", "Sixes" = "red")) +  # Assign custom colors
  labs(title = "Fours and Sixes per Innings by Season", x = "Season", y = "Average Boundaries per Innings") +
  theme_minimal() +
  theme(legend.title = element_blank())  # Remove the legend title
```



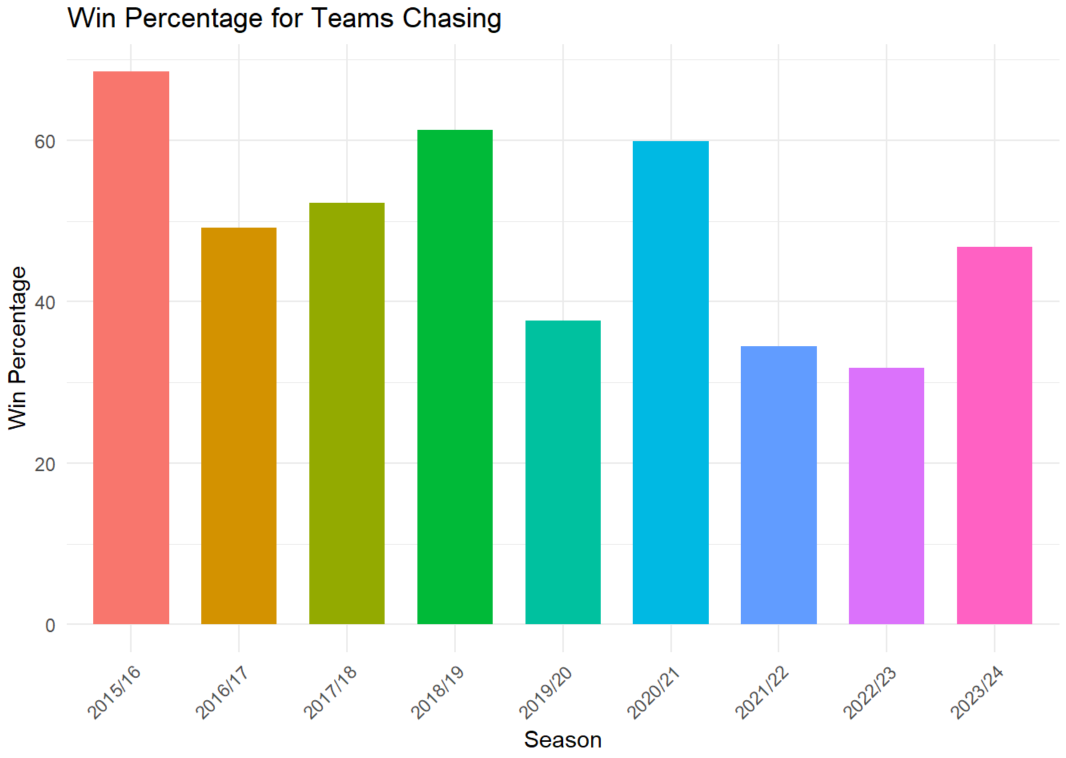
Dot Balls

```
dot_ball_percentage <- Merged_Data |>
  mutate(is_dot_ball = ifelse(runs_off_bat == 0 & is.na(wides) & is.na(noballs), 1, 0)) |>
  group_by(season.x) |>
  summarise(
    dot_balls = sum(is_dot_ball, na.rm = TRUE),
    total_deliveries = n() + sum(!is.na(wides) | !is.na(noballs), na.rm = TRUE),
    dot_ball_percentage = (dot_balls / total_deliveries) * 100
  ) |>
  ungroup() |>
  mutate(season.x = factor(season.x, levels = unique(season.x))) |>
  arrange(season.x)

ggplot(dot_ball_percentage, aes(x = season.x, y = dot_ball_percentage, group = 1)) +
  geom_line() +  # Ensure a single group for connecting lines
  geom_point() +
  labs(title = "Dot Ball Percentage Over Seasons", x = "Season", y = "Dot Ball Percentage") +
  theme_minimal()
```



Chasing Totals

```
Merged_Data <- Merged_Data |>
  mutate(chasing_win = ifelse(winner == team2, 1, 0))

# Now calculate win percentages for teams chasing, grouped by season
win_percentage_by_season <- Merged_Data |>
  group_by(season.x) |>
  summarise(total_matches = n(),
            chasing_wins = sum(chasing_win, na.rm = TRUE),  # Ensure NA values are handled
            win_percentage = (chasing_wins / total_matches) * 100) |>
  ungroup()  # Ungroup to ensure further operations aren't affected by grouping

# Visualize win percentage for teams chasing over seasons
ggplot(win_percentage_by_season, aes(x = season.x, y = win_percentage, fill = season.x)) +
  geom_bar(stat = "identity", width = 0.7) +  # Bar chart with slightly reduced bar width for clarity
  labs(title = "Win Percentage for Teams Chasing",
       x = "Season", y = "Win Percentage") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels for better readability
        legend.position = "none")
```



Models / Hypothesis

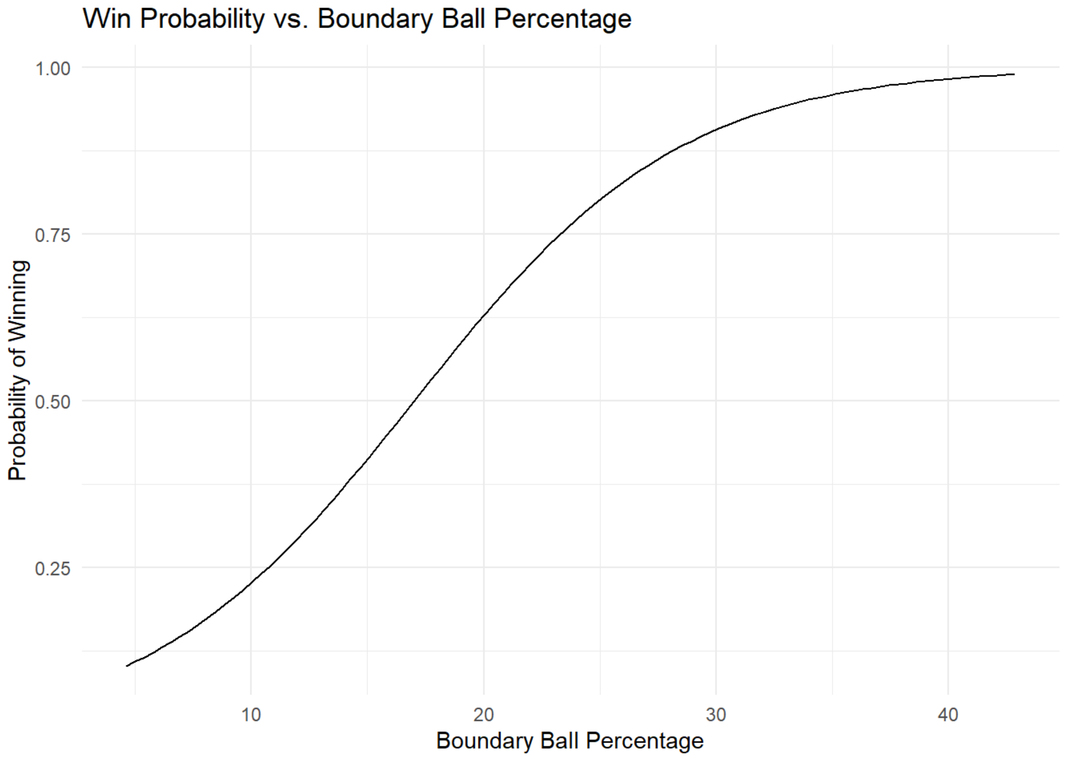
Logistic Regression Model: Boundary Ball Percentage and Winning Probability

```
# Calculate boundary ball percentage
Run_Percentage <- Merged_Data %>%
  mutate(boundary_ball = ifelse(runs_off_bat %in% c(4, 6), 1, 0),
         dot_ball = ifelse(runs_off_bat == 0 & is.na(extras), 1, 0)) %>%
  group_by(match_id, batting_team) %>%
  reframe(total_boundaries = sum(boundary_ball),
            total_dot_balls = sum(dot_ball),
            total_balls = n(),
            boundary_ball_percentage = (total_boundaries / total_balls) * 100,
            dot_ball_percentage = (total_dot_balls / total_balls) * 100,
            win = ifelse(batting_team == winner, 1, 0),
            .groups = 'drop')
 Run_Percentage$dot_ball_percentage <- as.numeric(Run_Percentage$dot_ball_percentage)
```

```
# Logistic regression model
# Fit the logistic regression model using only boundary_ball_percentage
model <- glm(win ~ boundary_ball_percentage, data = Run_Percentage, family = "binomial")

# Summary of the model
summary(model)
```

```
##
## Call:
## glm(formula = win ~ boundary_ball_percentage, family = "binomial",
##     data = Run_Percentage)
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -2.972606   0.036272  -81.95   <2e-16 ***
## boundary_ball_percentage  0.174711   0.002101   83.16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 86235  on 62216  degrees of freedom
## Residual deviance: 77725  on 62215  degrees of freedom
##   (1234 observations deleted due to missingness)
## AIC: 77729
##
## Number of Fisher Scoring iterations: 4
```

```
boundary_seq <- seq(min(Run_Percentage$boundary_ball_percentage, na.rm = TRUE),
                    max(Run_Percentage$boundary_ball_percentage, na.rm = TRUE), length = 100)

pred_data <- data.frame(boundary_ball_percentage = boundary_seq)
pred_data$win_prob <- predict(model, newdata = pred_data, type = "response")

ggplot(pred_data, aes(x = boundary_ball_percentage, y = win_prob)) +
  geom_line() +
  labs(title = "Win Probability vs. Boundary Ball Percentage",
       x = "Boundary Ball Percentage", y = "Probability of Winning") +
  theme_minimal()
```


Win Probability vs. Boundary Ball Percentage
Probability of Winning
Boundary Ball Percentage