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.

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.2     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.3     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

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

```
## Warning: package 'plotly' was built under R version 4.3.2
```

```
##
## Attaching package: 'plotly'
##
## The following object is masked from 'package:ggplot2':
##
##     last_plot
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following object is masked from 'package:graphics':
##
##     layout
```

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

```
summary(PSL_Ball)
```

```
##     match_id          season            start_date            venue
##  Min.   : 959175   Length:63451       Min.   :2016-02-04   Length:63451
##  1st Qu.:1128840   Class :character   1st Qu.:2018-03-10   Class :character
##  Median :1211667   Mode  :character   Median :2020-03-12   Mode  :character
##  Mean   :1213270                      Mean   :2020-04-24
##  3rd Qu.:1293023                      3rd Qu.:2022-02-17
##  Max.   :1416494                      Max.   :2024-03-06
##
##     innings           over            ball        batting_team
##  Min.   :1.000   Min.   : 1.00   Min.   : 1.000   Length:63451
##  1st Qu.:1.000   1st Qu.: 5.00   1st Qu.: 2.000   Class :character
##  Median :1.000   Median :10.00   Median : 4.000   Mode  :character
##  Mean   :1.483   Mean   :10.13   Mean   : 3.617
##  3rd Qu.:2.000   3rd Qu.:15.00   3rd Qu.: 5.000
##  Max.   :4.000   Max.   :20.00   Max.   :11.000
##
##  bowling_team         striker          non_striker           bowler
##  Length:63451       Length:63451       Length:63451       Length:63451
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   runs_off_bat       extras         ball_in_over   extra_ball
##  Min.   :0.000   Min.   :0.00000   Min.   :0.000   Mode :logical
##  1st Qu.:0.000   1st Qu.:0.00000   1st Qu.:2.000   FALSE:61248
##  Median :1.000   Median :0.00000   Median :3.000   TRUE :2203
##  Mean   :1.279   Mean   :0.06908   Mean   :3.451
##  3rd Qu.:1.000   3rd Qu.:0.00000   3rd Qu.:5.000
##  Max.   :6.000   Max.   :5.00000   Max.   :7.000
##
##  balls_remaining  runs_scored_yet    wicket        wickets_lost_yet
##  Min.   :  0.00   Min.   :  0.00   Mode :logical   Min.   : 0.00
##  1st Qu.: 33.00   1st Qu.: 36.00   FALSE:60091     1st Qu.: 1.00
##  Median : 62.00   Median : 73.00   TRUE :3360      Median : 2.00
##  Mean   : 61.65   Mean   : 77.07                   Mean   : 2.56
##  3rd Qu.: 91.00   3rd Qu.:113.00                   3rd Qu.: 4.00
##  Max.   :120.00   Max.   :262.00                   Max.   :10.00
##
##  innings1_total  innings2_total      target          wides
##  Min.   : 59.0   Min.   : 60.0   Min.   : 60.0   Min.   :1.00
##  1st Qu.:148.0   1st Qu.:134.0   1st Qu.:149.0   1st Qu.:1.00
##  Median :170.0   Median :157.0   Median :171.0   Median :1.00
##  Mean   :167.3   Mean   :154.7   Mean   :168.3   Mean   :1.21
##  3rd Qu.:187.0   3rd Qu.:173.0   3rd Qu.:188.0   3rd Qu.:1.00
##  Max.   :262.0   Max.   :253.0   Max.   :263.0   Max.   :5.00
##                  NA's   :200                     NA's   :61498
##     noballs           byes          legbyes         penalty
##  Min.   :1.00    Min.   :1.00    Min.   :1.00    Min.   :5
##  1st Qu.:1.00    1st Qu.:1.00    1st Qu.:1.00    1st Qu.:5
##  Median :1.00    Median :1.00    Median :1.00    Median :5
##  Mean   :1.02    Mean   :2.06    Mean   :1.33    Mean   :5
##  3rd Qu.:1.00    3rd Qu.:4.00    3rd Qu.:1.00    3rd Qu.:5
##  Max.   :5.00    Max.   :5.00    Max.   :5.00    Max.   :5
##  NA's   :63201   NA's   :63266   NA's   :62415   NA's   :63450
##  wicket_type        player_dismissed   other_wicket_type other_player_dismissed
##  Length:63451       Length:63451       Mode:logical      Mode:logical
##  Class :character   Class :character   NA's:63451        NA's:63451
##  Mode  :character   Mode  :character
##
##
##
##
##    .groups
##  Length:63451
##  Class :character
##  Mode  :character
##
##
##
##
```

```
summary(PSL_Match)
```

```
##    match_id         balls_per_over        team1              team2
##  Length:269         Length:269         Length:269         Length:269
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##     gender             season              date              event
##  Length:269         Length:269         Length:269         Length:269
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  match_number          venue               city           toss_winner
##  Length:269         Length:269         Length:269         Length:269
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  toss_decision      player_of_match      umpire1            umpire2
##  Length:269         Length:269         Length:269         Length:269
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  reserve_umpire      tv_umpire         match_referee         winner
##  Length:269         Length:269         Length:269         Length:269
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  winner_wickets        method          winner_runs          outcome
##  Length:269         Length:269         Length:269         Length:269
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##   eliminator
##  Length:269
##  Class :character
##  Mode  :character
```

```
summary(PSL_Player)
```

```
##      team              player            match_id
##  Length:5921        Length:5921        Length:5921
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```
sum(is.na(PSL_Ball))
```

```
## [1] 440932
```

```
sum(is.na(PSL_Ball))
```

```
## [1] 440932
```

```
sum(is.na(PSL_Ball))
```

```
## [1] 440932
```

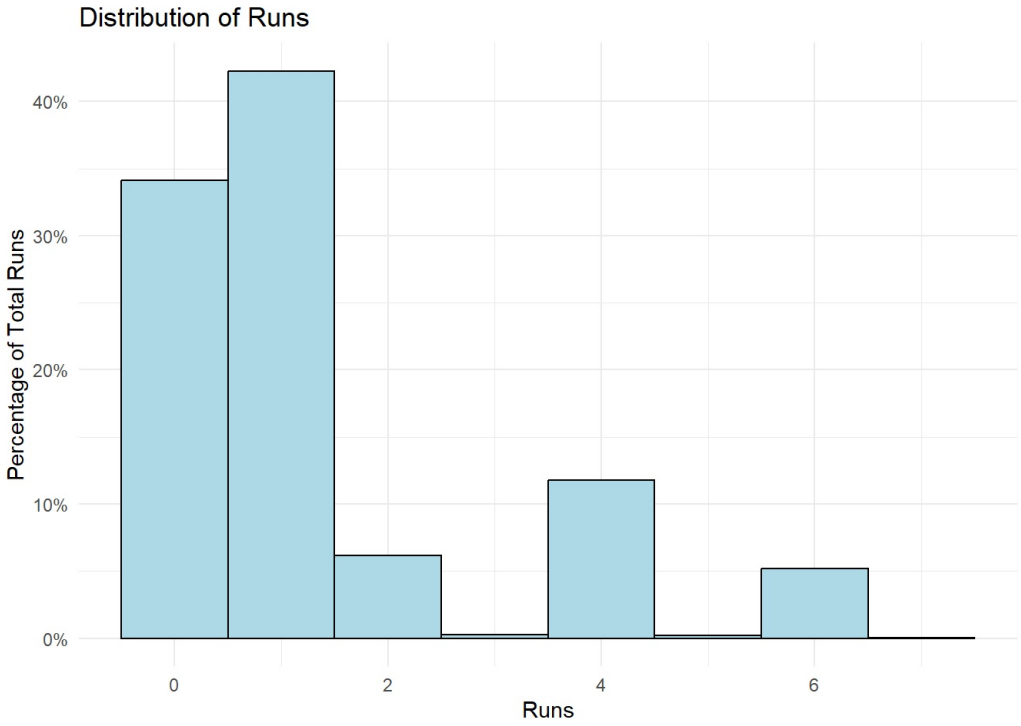
Merging Match and Ball-by-Ball Data

```
PSL_Ball$match_id <- as.character(PSL_Ball$match_id)
PSL_Match$match_id <- as.character(PSL_Match$match_id)
Merged_Data <- merge(PSL_Match, PSL_Ball, by = "match_id")
```

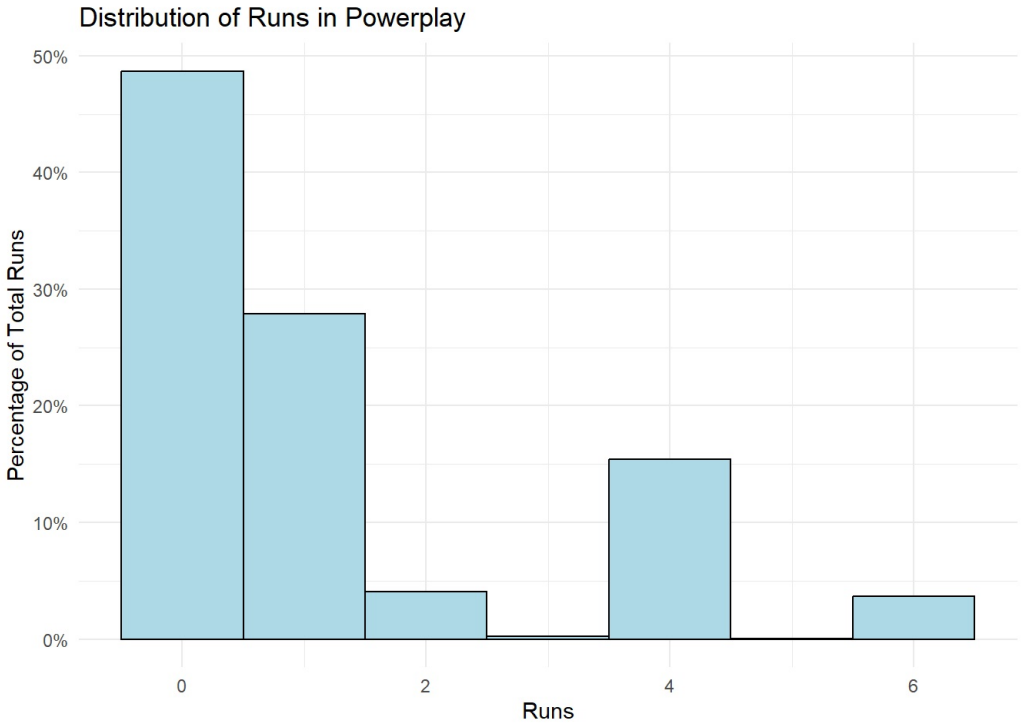
```
library(dplyr)
Merged_Data <- Merged_Data |>
  mutate(season.x = ifelse(season.x %in% c("2020/21", "2021"), "2020/21", season.x))
```

Summary Statistics

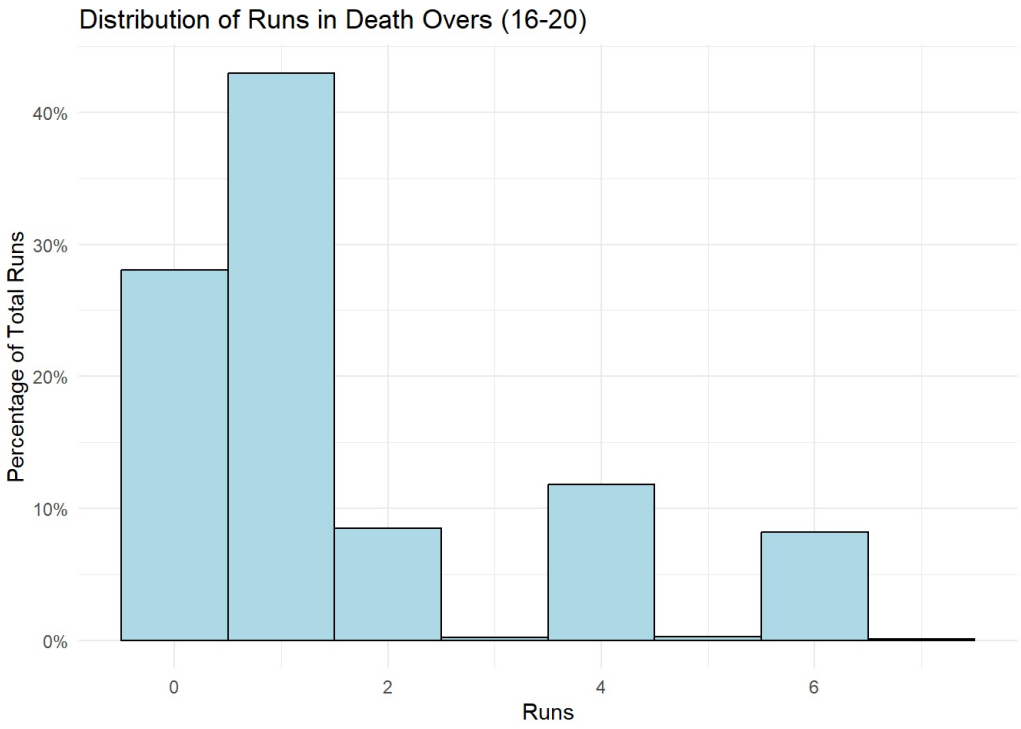
```
# Distribution of runs in Ball by Ball Data
ggplot(Merged_Data, aes(x = runs_off_bat + extras)) +
  geom_histogram(aes(y = after_stat(count) / sum(after_stat(count))), binwidth = 1, fill = "lightblue", color = "
black") +
  scale_y_continuous(labels = scales::percent, name = "Percentage of Total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



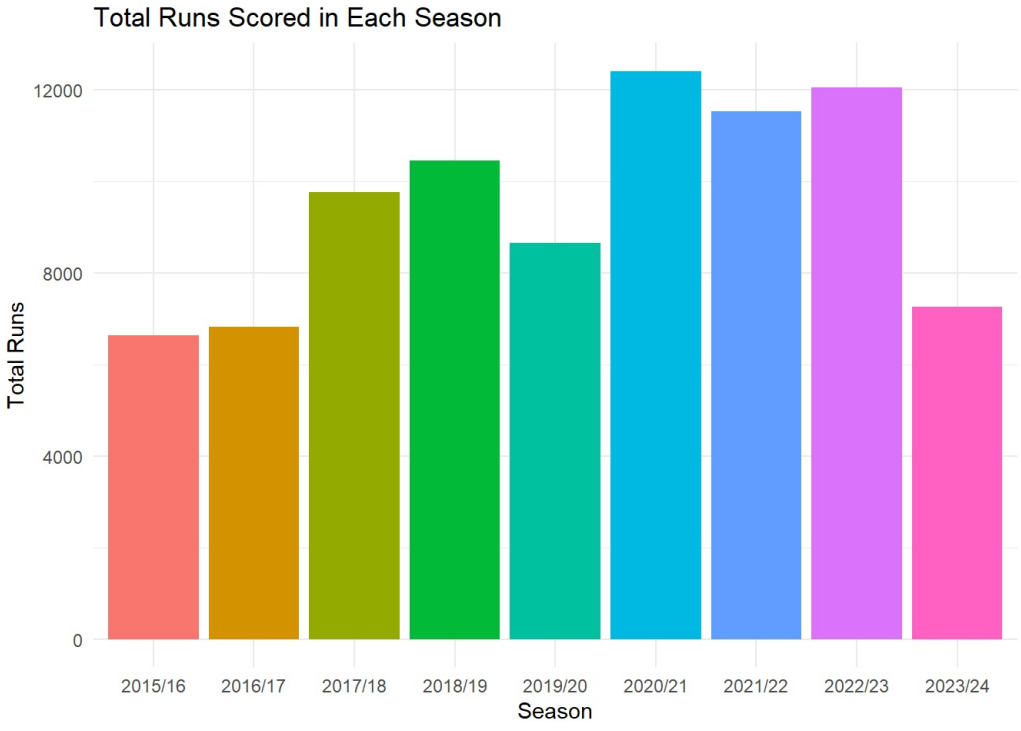

```
# Filter for powerplay
powerplay <- Merged_Data |>
  filter((over >= 1 & over <= 6))

# 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()
```



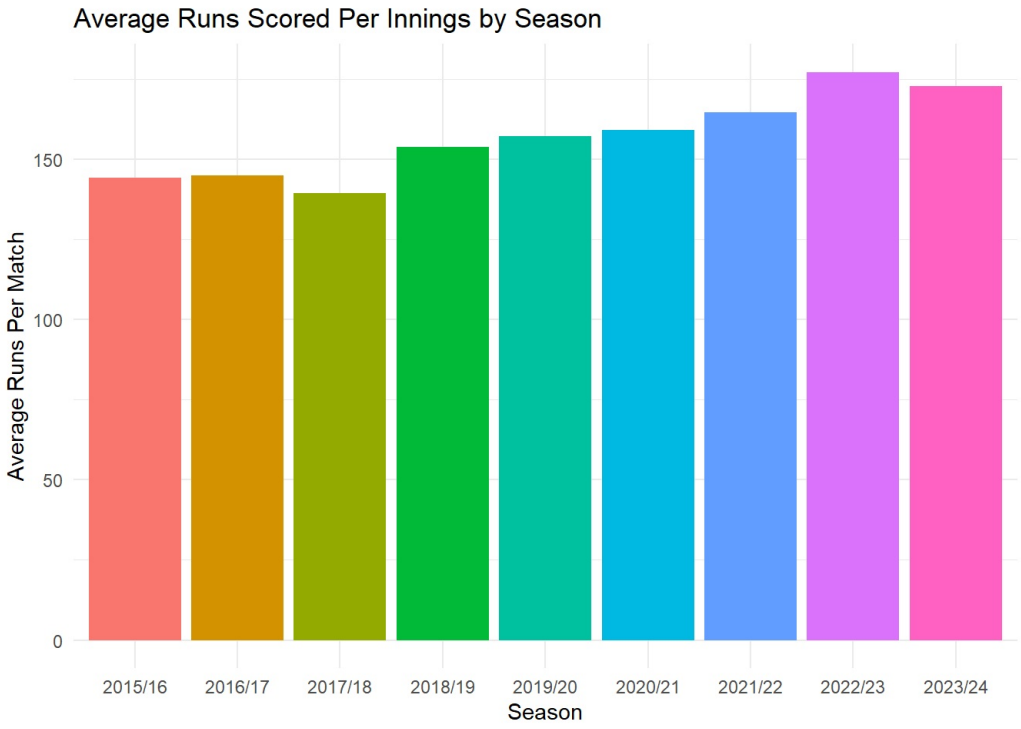
```
# Filter for death overs
death_overs <- Merged_Data |>
  filter((over >= 16 & over <= 20))

# 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_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()
```

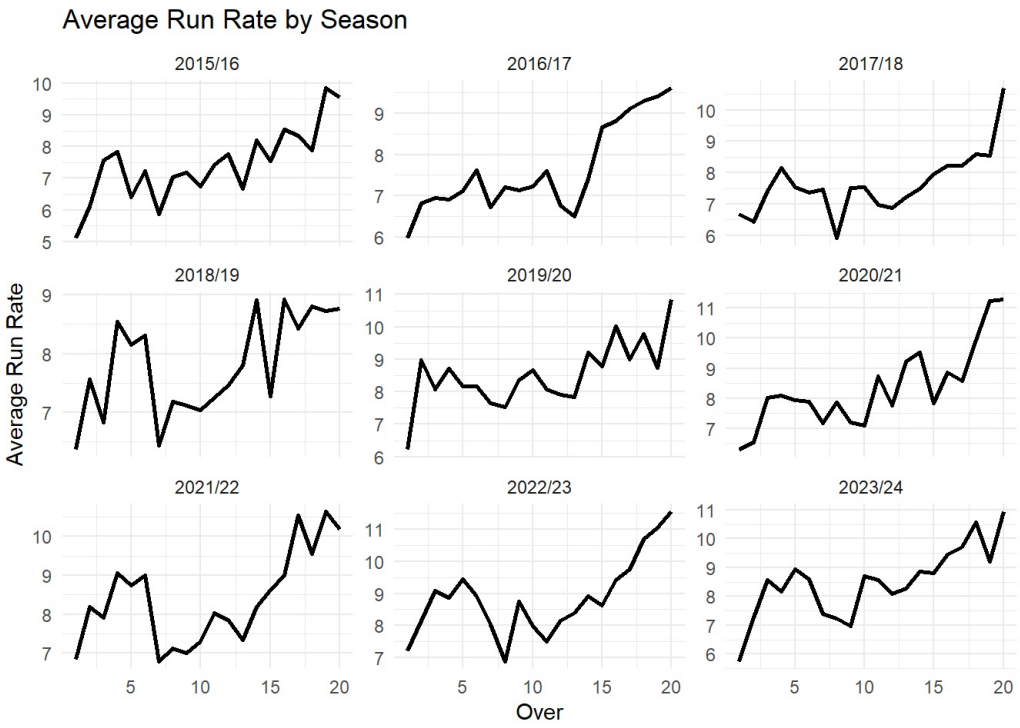


```
Avg_Runs_Per_Match_Season <- Merged_Data |>
  group_by(season.x, match_id, innings) |>
  summarise(Total_Runs = sum(runs_off_bat + extras), .groups = 'drop') |>
  group_by(season.x) |>
  summarise(Avg_Runs_Per_Match = mean(Total_Runs))

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()
```

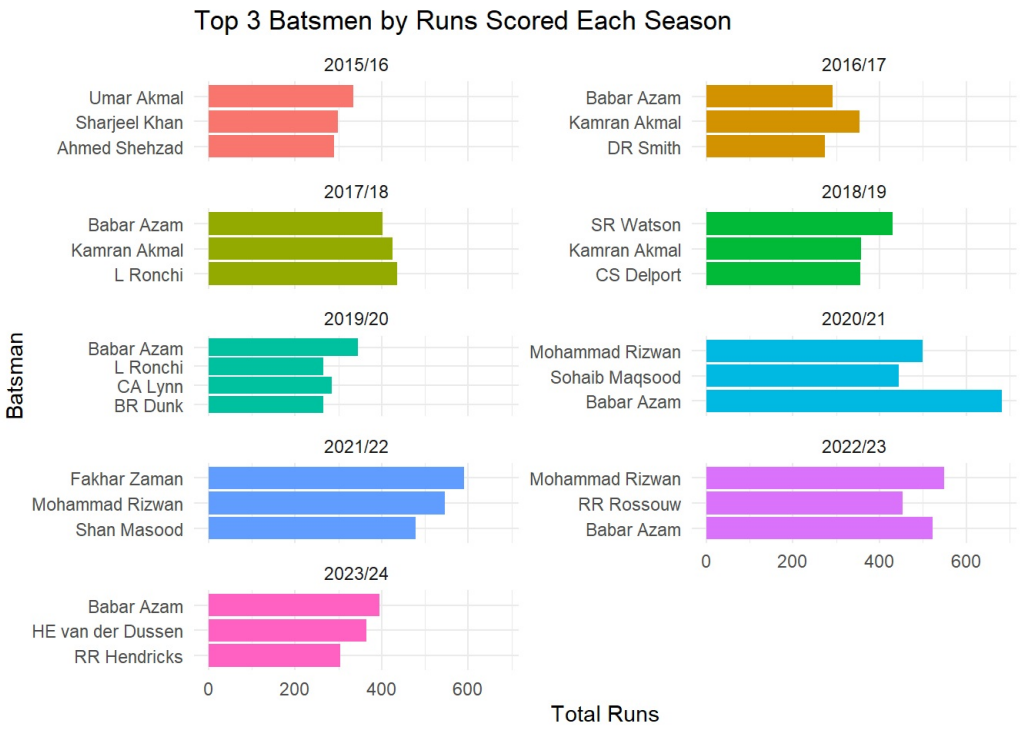



```
Average_Run_Rate <- Merged_Data |>
  group_by(season.x, over) |>
  summarise(
    Total_Runs = sum(runs_off_bat + extras, na.rm = TRUE),  # Calculate total runs
    Total_Balls = n(),  # Count total deliveries (rows)
    Total_Overs = Total_Balls / 6.0,  # Convert balls to overs
    Avg_Run_Rate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

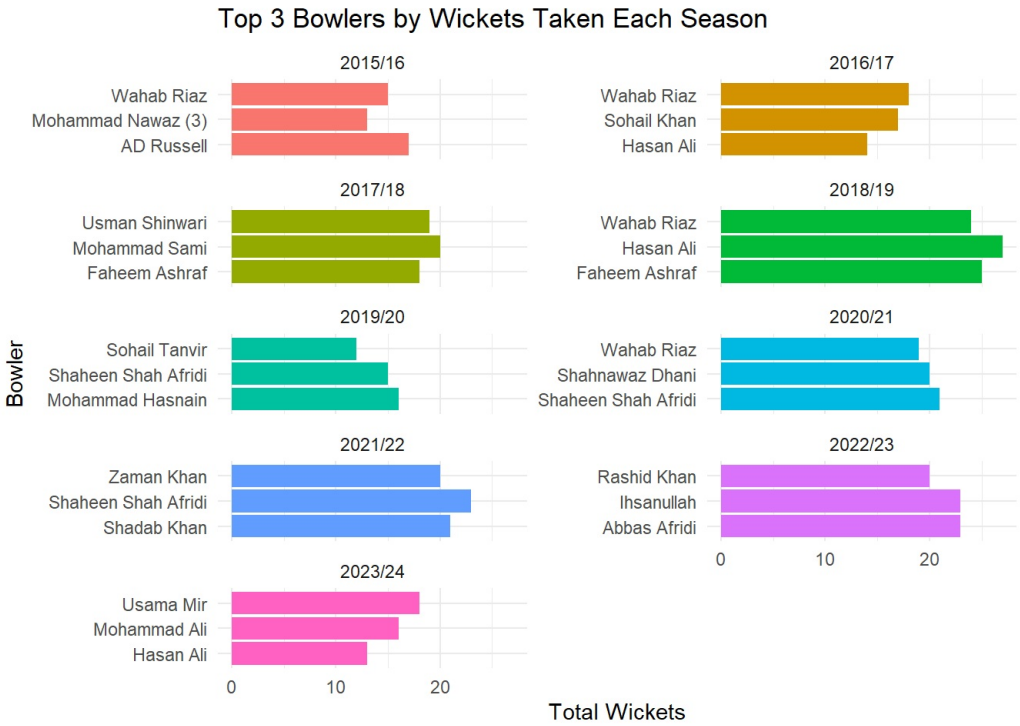
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



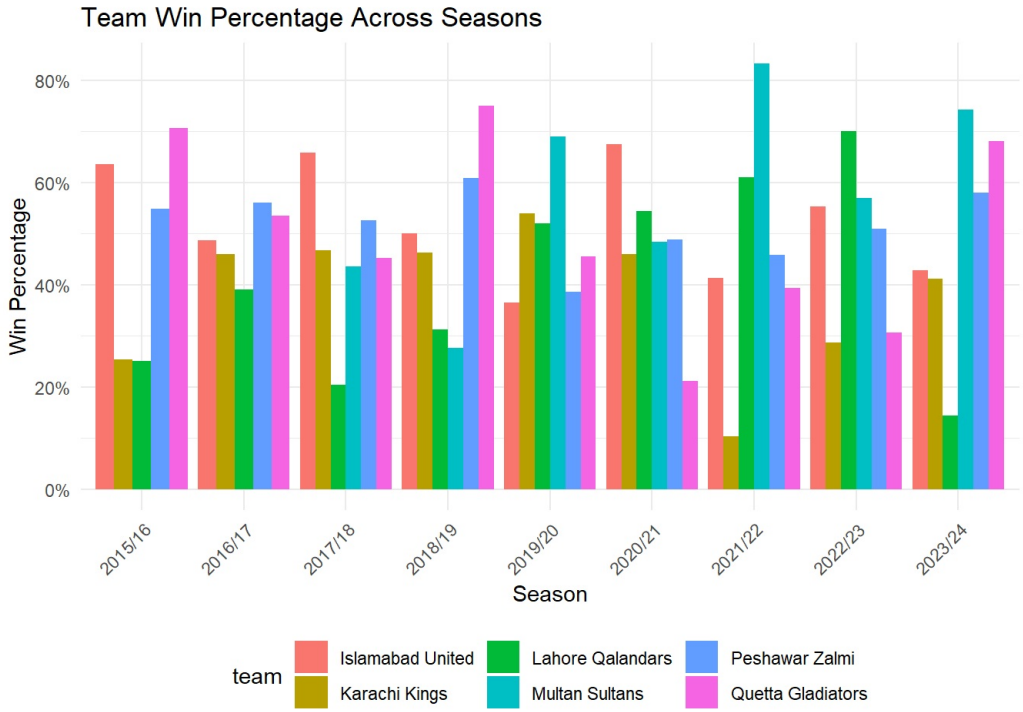

```
Top_Batsmen <- Merged_Data |>
  group_by(season.x, striker) |>
  summarise(Total_Runs = sum(runs_off_bat, na.rm = TRUE), .groups = 'drop') |>
  group_by(season.x) |>
  slice_max(order_by = Total_Runs, n = 3) |>
  ungroup()
ggplot(Top_Batsmen, aes(x = reorder(striker, Total_Runs), y = Total_Runs, fill = season.x)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Top 3 Batsmen 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)
```



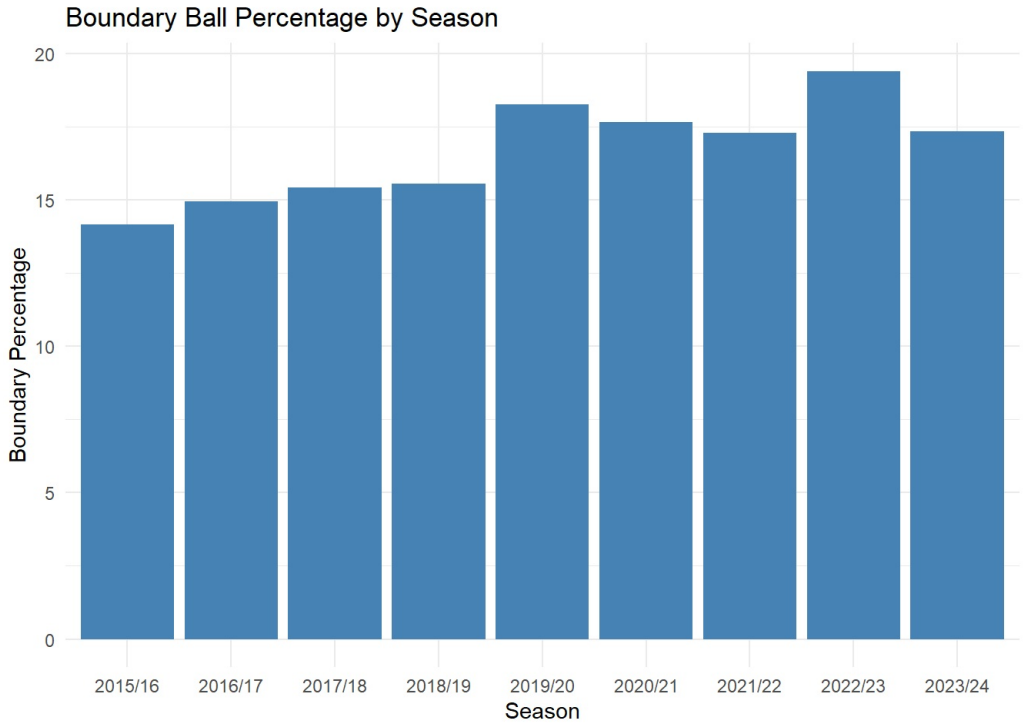
```
Top_Bowlers <- Merged_Data |>
  group_by(season.x, bowler) |>
  summarise(Total_Wickets = sum(as.numeric(wicket), na.rm = TRUE), .groups = 'drop_last') |>
  slice_max(order_by = Total_Wickets, n = 3, with_ties = FALSE) |>
  ungroup()

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)
```




```
# 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 <- 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)) +
  geom_col(fill = "steelblue") + # Bar plot
  labs(title = "Boundary Ball Percentage by Season",
       x = "Season",
       y = "Boundary Percentage") +
  theme_minimal()
```

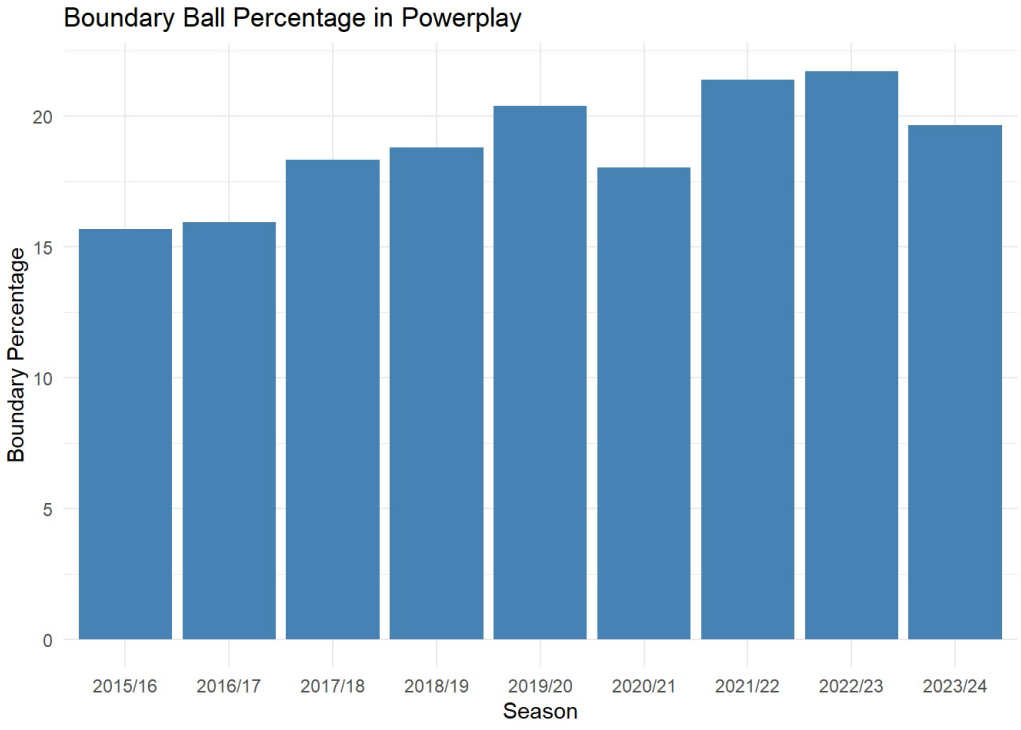


```r
library(dplyr)

# Assuming Merged_Data already contains ball-by-ball data along with match and team information
# 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) %>%
  summarise(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')
```

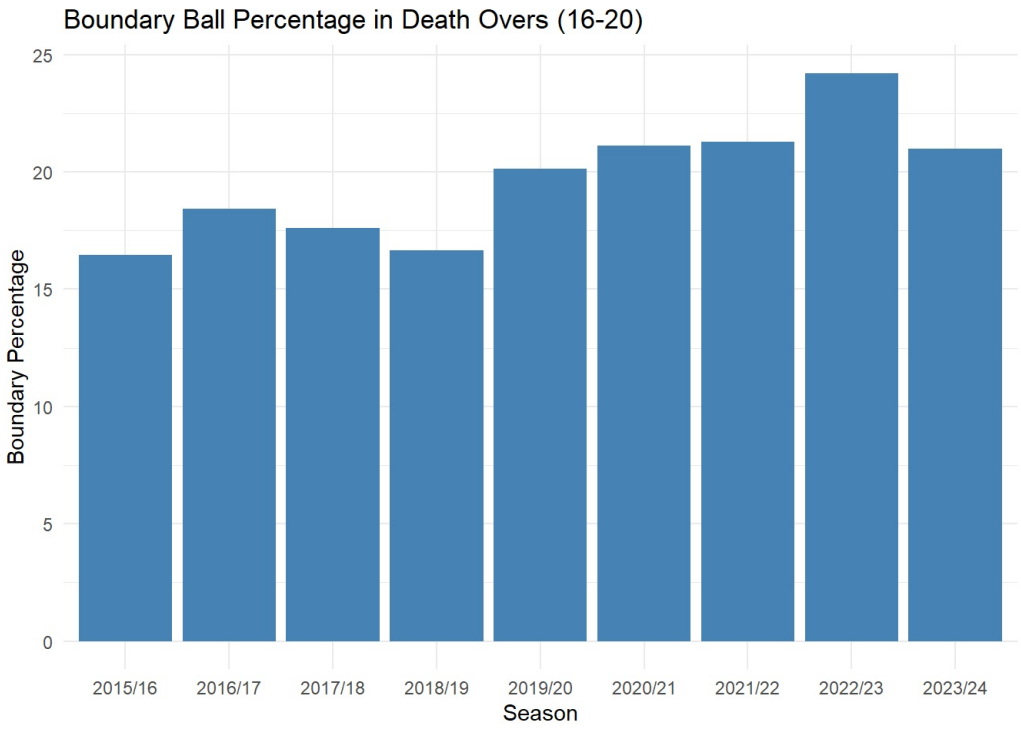
```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## ℹ Please use `reframe()` instead.
## ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
Run_Percentage$dot_ball_percentage <- as.numeric(Run_Percentage$dot_ball_percentage)
```

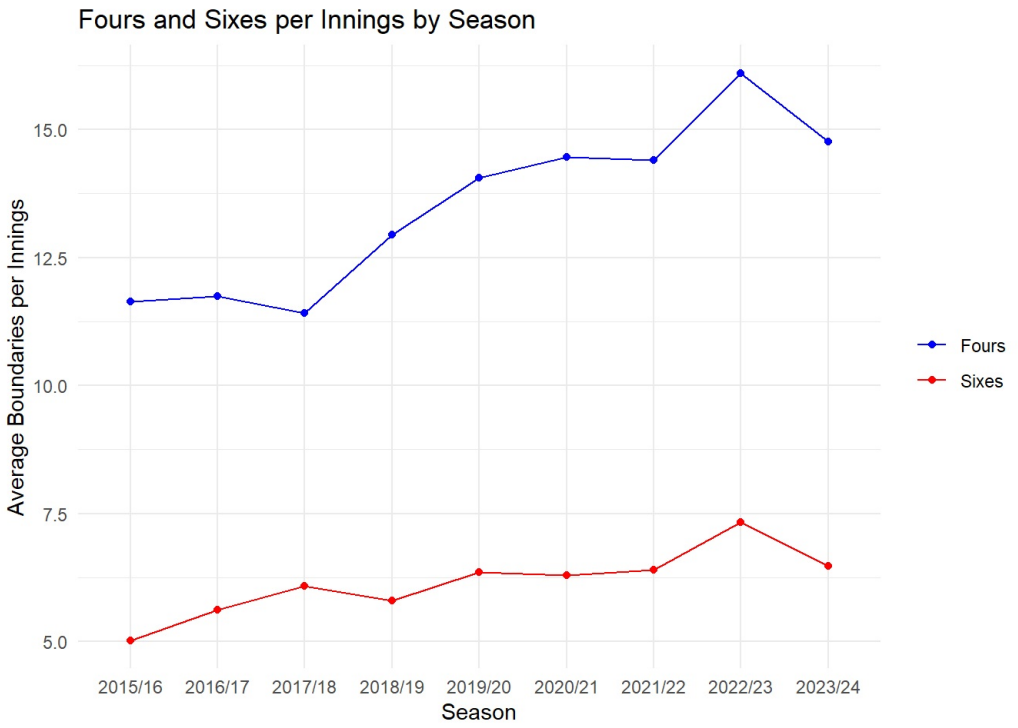
```r
boundary_balls <- powerplay |>
  filter(runs_off_bat %in% c(4, 6)) |>
  mutate(boundary = 1) # Mark boundary balls

# 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)) +
  geom_col(fill = "steelblue") + # Bar plot
  labs(title = "Boundary Ball Percentage in Powerplay",
       x = "Season",
       y = "Boundary Percentage") +
  theme_minimal()
```



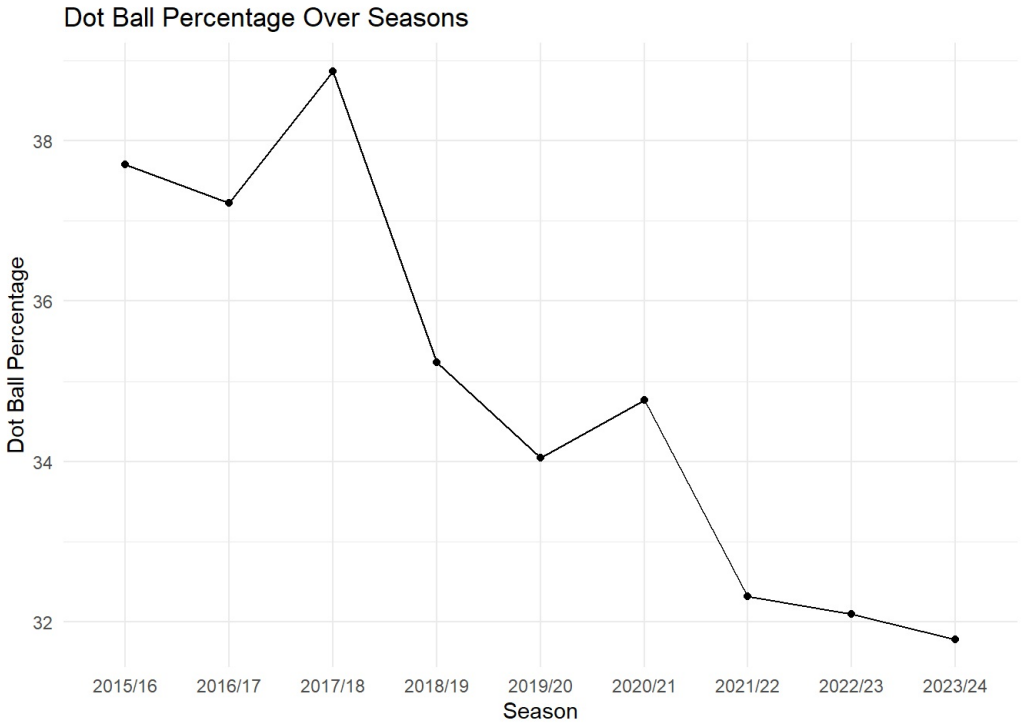
```
boundary_balls <- death_overs |>
  filter(runs_off_bat %in% c(4, 6)) |>
  mutate(boundary = 1) # Mark boundary balls

# 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)) +
  geom_col(fill = "steelblue") + # Bar plot
  labs(title = "Boundary Ball Percentage in Death Overs (16-20)",
       x = "Season",
       y = "Boundary Percentage") +
  theme_minimal()
```



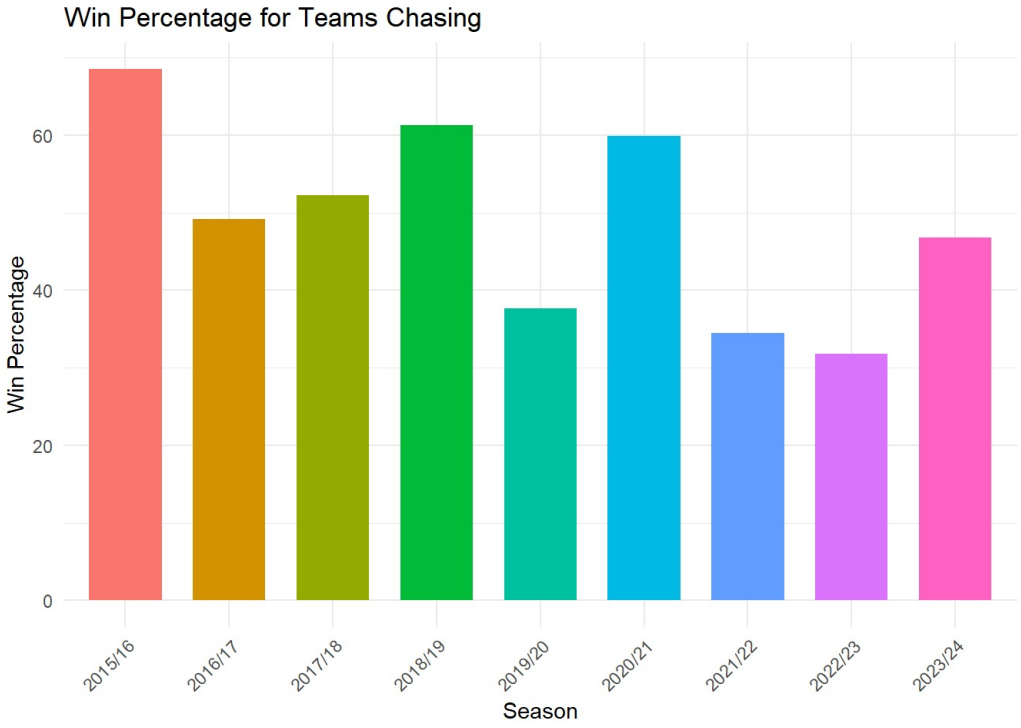

```
# Step 1: Prepare the data
boundaries_per_innings <- Merged_Data |>
  filter(runs_off_bat %in% c(4, 6)) |>
  mutate(boundary_type = ifelse(runs_off_bat == 4, "Fours", "Sixes")) |>
  group_by(season.x, match_id, innings, boundary_type) |>
  summarise(total_boundaries = n(), .groups = 'drop') |>
 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



```
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
# 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()
```