Indian Premier League T20 Analysis

Data Science Professional Certificate Program from HarvardX - CYO

Riaz Mohamed Vellamparambil

02/21/2021

Contents

Introduction

This is the submission for the choose-your-own project for the Harvard's multi-part Data Science Professional Certificate series. The goal of this project is to rank players and predict the winner of a match. T20 is a modern format of the original game of cricket where each ball and strike count. A team can acquire players through any of the three ways: the annual player auction, trading players with other teams during the trading windows, and signing replacements for unavailable players. Players sign up for the auction and also set their base price, and are bought by the franchise that bids the highest for them.

About Cricket

https://en.wikipedia.org/wiki/Cricket

https://en.wikipedia.org/wiki/Indian_Premier_League

Data Set

IPL ball by ball https://www.kaggle.com/patrickb1912/ipl-complete-dataset-20082020?select=IPL+Ball-by-Ball+2008-2020.csv

IPL Matches https://www.kaggle.com/patrickb1912/ipl-complete-dataset-20082020?select=IPL+Matches+2008-2020.csv

Executive Summary:

The goal of this project is to rank players by performance. This gives the IPL teams a pool of players as per caliber from which to pick from. We will also build a model to predict the winner of a match.

Steps of Analysis

- **Data Preparation** - We will load the data,
- **Data Exploration** - We will explore the data, plot histograms, create tables and graphs , As part of the data exploration process we will look at three key stats. Bowler Stats, Batsmen Stats and Team Stats
- **Modeling** - We Will create two Models , one to create a pool of playes teams can choose from and predict match winner
- **Conclusion** - We will summarize the results of modeling

Data Preparation

Install and load libraries and MovieLens Dataset To install the required packages, we use if(!require and load the packages from http://cran.us.r-project.org. Load Libraries and the IPL dataset

```
# Install all needed libraries

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(forcats)) install.packages("forcats", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(kableExtra)) install.packages("kableExtra" , repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(tidyr)) install.packages("tidyr", repos = "http://cran.us.r-project.org")
if(!require(stringr)) install.packages("stringr", repos = "http://cran.us.r-project.org")
if(!require(plotly)) install.packages("plotly", repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org")
if (!require(formattable)) install.packages("formattable", repos = "http://cran.us.r-project.org")
```

```
# Load required libraries
library(tidyverse)
library(caret)
library(kableExtra)
library(data.table)
library(dplyr)
library(tidyverse)
library(tidyr)
library(stringr)
library(forcats)
library(ggplot2)
library(plotly)
library(ggthemes)
library(formattable)
```

Import data sets from kaggle into R (I have downloaded the data from kaggle to my github repository)

```
# read the .csv datafiles into R

ball_by_ball <- read.csv("https://raw.githubusercontent.com/riazvm/capstone-ipl/main/data/IPL-Ball-by-Ba
matches <- read.csv("https://raw.githubusercontent.com/riazvm/capstone-ipl/main/data/IPL-Matches-2008-20
```

We will now inspect the data set to view column names

Lets check how the data looks in the two data sets

```
# inspect column nammes in the datasets (Can use colnames or names)
colnames(ball_by_ball)
```

```
##  [1] "id"               "inning"           "over"             "ball"
##  [5] "batsman"          "non_striker"      "bowler"           "batsman_runs"
##  [9] "extra_runs"       "total_runs"       "non_boundary"     "is_wicket"
## [13] "dismissal_kind"   "player_dismissed" "fielder"          "extras_type"
## [17] "batting_team"     "bowling_team"
```

```r
names(matches)
```

```
##  [1] "id"              "city"            "date"            "player_of_match"
##  [5] "venue"           "neutral_venue"   "team1"           "team2"
##  [9] "toss_winner"     "toss_decision"   "winner"          "result"
## [13] "result_margin"   "eliminator"      "method"          "umpire1"
## [17] "umpire2"
```

```r
# Lets check how the data looks in the two data sets
head(matches)
```

```
##       id       city       date player_of_match
## 1 335982  Bangalore 2008-04-18     BB McCullum
## 2 335983 Chandigarh 2008-04-19      MEK Hussey
## 3 335984      Delhi 2008-04-19     MF Maharoof
## 4 335985     Mumbai 2008-04-20      MV Boucher
## 5 335986    Kolkata 2008-04-20       DJ Hussey
## 6 335987     Jaipur 2008-04-21       SR Watson
##                                       venue neutral_venue
## 1                     M Chinnaswamy Stadium             0
## 2 Punjab Cricket Association Stadium, Mohali             0
## 3                           Feroz Shah Kotla             0
## 4                           Wankhede Stadium             0
## 5                               Eden Gardens             0
## 6                    Sawai Mansingh Stadium              0
##                         team1                       team2
## 1 Royal Challengers Bangalore       Kolkata Knight Riders
## 2             Kings XI Punjab         Chennai Super Kings
## 3            Delhi Daredevils            Rajasthan Royals
## 4              Mumbai Indians Royal Challengers Bangalore
## 5       Kolkata Knight Riders             Deccan Chargers
## 6            Rajasthan Royals             Kings XI Punjab
##                   toss_winner toss_decision                     winner  result
## 1 Royal Challengers Bangalore         field       Kolkata Knight Riders    runs
## 2         Chennai Super Kings           bat         Chennai Super Kings    runs
## 3            Rajasthan Royals           bat            Delhi Daredevils wickets
## 4              Mumbai Indians           bat Royal Challengers Bangalore wickets
## 5             Deccan Chargers           bat       Kolkata Knight Riders wickets
## 6             Kings XI Punjab           bat            Rajasthan Royals wickets
##   result_margin eliminator method   umpire1        umpire2
## 1           140          N   <NA> Asad Rauf    RE Koertzen
## 2            33          N   <NA> MR Benson     SL Shastri
## 3             9          N   <NA> Aleem Dar GA Pratapkumar
## 4             5          N   <NA>  SJ Davis      DJ Harper
## 5             5          N   <NA> BF Bowden    K Hariharan
## 6             6          N   <NA> Aleem Dar      RB Tiffin
```

```r
head(ball_by_ball)
```

```
##       id inning over ball     batsman non_striker    bowler batsman_runs
## 1 335982      1    6    5  RT Ponting BB McCullum AA Noffke            1
## 2 335982      1    6    6 BB McCullum  RT Ponting AA Noffke            1
```

```
## 3 335982      1    7    1 BB McCullum  RT Ponting    Z Khan            0
## 4 335982      1    7    2 BB McCullum  RT Ponting    Z Khan            1
## 5 335982      1    7    3  RT Ponting BB McCullum    Z Khan            1
## 6 335982      1    7    4 BB McCullum  RT Ponting    Z Khan            1
##   extra_runs total_runs non_boundary is_wicket dismissal_kind player_dismissed
## 1          0          1            0         0           <NA>             <NA>
## 2          0          1            0         0           <NA>             <NA>
## 3          0          0            0         0           <NA>             <NA>
## 4          0          1            0         0           <NA>             <NA>
## 5          0          1            0         0           <NA>             <NA>
## 6          0          1            0         0           <NA>             <NA>
##   fielder extras_type          batting_team                bowling_team
## 1    <NA>        <NA> Kolkata Knight Riders Royal Challengers Bangalore
## 2    <NA>        <NA> Kolkata Knight Riders Royal Challengers Bangalore
## 3    <NA>        <NA> Kolkata Knight Riders Royal Challengers Bangalore
## 4    <NA>        <NA> Kolkata Knight Riders Royal Challengers Bangalore
## 5    <NA>        <NA> Kolkata Knight Riders Royal Challengers Bangalore
## 6    <NA>        <NA> Kolkata Knight Riders Royal Challengers Bangalore
```

The data matches data set shows all matches played between 2008 and 2020 between teams with a uique match id , we also see have other data that are described by each colum in the matches dataset The ball_by_ball data set has each delivery bowled in the ipl matches and the match id corresponnds to the match id in the matches dataset. An over consists of 6 deliveries by a bowler

Let us check the data set for any anamolies , eg. duplicates, spellings etc. For this we will use the Factor function. Factor is a data structure used for fields that takes only predefined, finite number of values (categorical data). For example: a data field such as marital status may contain only values from single, married, separated, divorced, or widowed.

```
# Let us check the data set for any anamolies , eg. duplicates, spellings etc. For this we will use the
levels(as.factor(ball_by_ball$batting_team))
```

```
##  [1] "Chennai Super Kings"         "Deccan Chargers"
##  [3] "Delhi Capitals"              "Delhi Daredevils"
##  [5] "Gujarat Lions"               "Kings XI Punjab"
##  [7] "Kochi Tuskers Kerala"        "Kolkata Knight Riders"
##  [9] "Mumbai Indians"              "Pune Warriors"
## [11] "Rajasthan Royals"            "Rising Pune Supergiant"
## [13] "Rising Pune Supergiants"     "Royal Challengers Bangalore"
## [15] "Sunrisers Hyderabad"
```

```
levels(as.factor(ball_by_ball$bowling_team))
```

```
##  [1] "Chennai Super Kings"         "Deccan Chargers"
##  [3] "Delhi Capitals"              "Delhi Daredevils"
##  [5] "Gujarat Lions"               "Kings XI Punjab"
##  [7] "Kochi Tuskers Kerala"        "Kolkata Knight Riders"
##  [9] "Mumbai Indians"              "Pune Warriors"
## [11] "Rajasthan Royals"            "Rising Pune Supergiant"
## [13] "Rising Pune Supergiants"     "Royal Challengers Bangalore"
## [15] "Sunrisers Hyderabad"
```

The following are the teams that play the IPL currently Chennai Super Kings, Delhi Capitals, Gujarat Titans, Kolkata Knight Riders, Lucknow Super Giants, Mumbai Indians, Punjab Kings, Rajasthan Royals,

Royal Challengers Bangalore, Sunrisers Hyderabad. We will still need to keep the other teams that were playing the IPL previously to get the top players so the pool from which the teams can choose from is bigger.

Let is now conditionally replace the anomalies we see in the data

```
#Let is now replace the anomalies we see in the correct data

# ball_by_ball data set

ball_by_ball$batting_team[ball_by_ball$batting_team == "Delhi Daredevils"] <- "Delhi Capitals"
ball_by_ball$batting_team[ball_by_ball$batting_team == "Kings XI Punjab"] <- "Punjab Kings"
ball_by_ball$batting_team[ball_by_ball$batting_team == "Gujarat Lions"] <- "Gujarat Titans"
ball_by_ball$batting_team[ball_by_ball$batting_team == "Gujarat Lions"] <- "Gujarat Titans"
ball_by_ball$batting_team[ball_by_ball$batting_team == "Rising Pune Supergiant"] <- "Lucknow Super Gian
ball_by_ball$batting_team[ball_by_ball$batting_team == "Rising Pune Supergiants"] <- "Lucknow Super Gia
ball_by_ball$batting_team[ball_by_ball$batting_team == "Pune Warriors"] <- "Lucknow Super Giants"

ball_by_ball$bowling_team[ball_by_ball$bowling_team == "Delhi Daredevils"] <- "Delhi Capitals"
ball_by_ball$bowling_team[ball_by_ball$bowling_team == "Kings XI Punjab"] <- "Punjab Kings"
ball_by_ball$bowling_team[ball_by_ball$bowling_team == "Gujarat Lions"] <- "Gujarat Titans"
ball_by_ball$bowling_team[ball_by_ball$bowling_team == "Gujarat Lions"] <- "Gujarat Titans"
ball_by_ball$bowling_team[ball_by_ball$bowling_team == "Rising Pune Supergiant"] <- "Lucknow Super Gian
ball_by_ball$bowling_team[ball_by_ball$bowling_team == "Rising Pune Supergiants"] <- "Lucknow Super Gia
ball_by_ball$bowling_team[ball_by_ball$bowling_team == "Pune Warriors"] <- "Lucknow Super Giants"
```

```
# Check if we hav all the correct teams
levels(as.factor(ball_by_ball$batting_team))
```

```
##  [1] "Chennai Super Kings"         "Deccan Chargers"
##  [3] "Delhi Capitals"              "Gujarat Titans"
##  [5] "Kochi Tuskers Kerala"        "Kolkata Knight Riders"
##  [7] "Lucknow Super Giants"        "Mumbai Indians"
##  [9] "Punjab Kings"                "Rajasthan Royals"
## [11] "Royal Challengers Bangalore" "Sunrisers Hyderabad"
```

```
levels(as.factor(ball_by_ball$bowling_team))
```

```
##  [1] "Chennai Super Kings"         "Deccan Chargers"
##  [3] "Delhi Capitals"              "Gujarat Titans"
##  [5] "Kochi Tuskers Kerala"        "Kolkata Knight Riders"
##  [7] "Lucknow Super Giants"        "Mumbai Indians"
##  [9] "Punjab Kings"                "Rajasthan Royals"
## [11] "Royal Challengers Bangalore" "Sunrisers Hyderabad"
```

Let us check if the same type of errors exist in "matches" dataset also for team1, team2, winner, toss_winner and venue variables.

```
# Lets look at the matches dataset for the same anomalies
levels(as.factor(matches$venue))
```

```
##  [1] "Barabati Stadium"
##  [2] "Brabourne Stadium"
```

```
##  [3] "Buffalo Park"
##  [4] "De Beers Diamond Oval"
##  [5] "Dr DY Patil Sports Academy"
##  [6] "Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium"
##  [7] "Dubai International Cricket Stadium"
##  [8] "Eden Gardens"
##  [9] "Feroz Shah Kotla"
## [10] "Green Park"
## [11] "Himachal Pradesh Cricket Association Stadium"
## [12] "Holkar Cricket Stadium"
## [13] "JSCA International Stadium Complex"
## [14] "Kingsmead"
## [15] "M Chinnaswamy Stadium"
## [16] "M.Chinnaswamy Stadium"
## [17] "MA Chidambaram Stadium, Chepauk"
## [18] "Maharashtra Cricket Association Stadium"
## [19] "Nehru Stadium"
## [20] "New Wanderers Stadium"
## [21] "Newlands"
## [22] "OUTsurance Oval"
## [23] "Punjab Cricket Association IS Bindra Stadium, Mohali"
## [24] "Punjab Cricket Association Stadium, Mohali"
## [25] "Rajiv Gandhi International Stadium, Uppal"
## [26] "Sardar Patel Stadium, Motera"
## [27] "Saurashtra Cricket Association Stadium"
## [28] "Sawai Mansingh Stadium"
## [29] "Shaheed Veer Narayan Singh International Stadium"
## [30] "Sharjah Cricket Stadium"
## [31] "Sheikh Zayed Stadium"
## [32] "St George's Park"
## [33] "Subrata Roy Sahara Stadium"
## [34] "SuperSport Park"
## [35] "Vidarbha Cricket Association Stadium, Jamtha"
## [36] "Wankhede Stadium"
```

```r
# The only Anamoly in this set is "M Chinnaswamy Stadium" & "M.Chinnaswamy Stadium"

matches$venue[matches$venue == "M Chinnaswamy Stadium"] <- "M. A. Chidambaram Stadium"
matches$venue[matches$venue == "M. Chinnaswamy Stadium"] <- "M. A. Chidambaram Stadium"
matches$venue[matches$venue == "MA Chidambaram Stadium, Chepauk"] <- "M. A. Chidambaram Stadium"

levels(as.factor(matches$venue))
```

```
##  [1] "Barabati Stadium"
##  [2] "Brabourne Stadium"
##  [3] "Buffalo Park"
##  [4] "De Beers Diamond Oval"
##  [5] "Dr DY Patil Sports Academy"
##  [6] "Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium"
##  [7] "Dubai International Cricket Stadium"
##  [8] "Eden Gardens"
##  [9] "Feroz Shah Kotla"
## [10] "Green Park"
## [11] "Himachal Pradesh Cricket Association Stadium"
```

```
## [12] "Holkar Cricket Stadium"
## [13] "JSCA International Stadium Complex"
## [14] "Kingsmead"
## [15] "M. A. Chidambaram Stadium"
## [16] "M.Chinnaswamy Stadium"
## [17] "Maharashtra Cricket Association Stadium"
## [18] "Nehru Stadium"
## [19] "New Wanderers Stadium"
## [20] "Newlands"
## [21] "OUTsurance Oval"
## [22] "Punjab Cricket Association IS Bindra Stadium, Mohali"
## [23] "Punjab Cricket Association Stadium, Mohali"
## [24] "Rajiv Gandhi International Stadium, Uppal"
## [25] "Sardar Patel Stadium, Motera"
## [26] "Saurashtra Cricket Association Stadium"
## [27] "Sawai Mansingh Stadium"
## [28] "Shaheed Veer Narayan Singh International Stadium"
## [29] "Sharjah Cricket Stadium"
## [30] "Sheikh Zayed Stadium"
## [31] "St George's Park"
## [32] "Subrata Roy Sahara Stadium"
## [33] "SuperSport Park"
## [34] "Vidarbha Cricket Association Stadium, Jamtha"
## [35] "Wankhede Stadium"
```

```
# Contiue other columns
levels(as.factor(matches$team1))
```

```
##  [1] "Chennai Super Kings"         "Deccan Chargers"
##  [3] "Delhi Capitals"              "Delhi Daredevils"
##  [5] "Gujarat Lions"               "Kings XI Punjab"
##  [7] "Kochi Tuskers Kerala"        "Kolkata Knight Riders"
##  [9] "Mumbai Indians"              "Pune Warriors"
## [11] "Rajasthan Royals"            "Rising Pune Supergiant"
## [13] "Rising Pune Supergiants"     "Royal Challengers Bangalore"
## [15] "Sunrisers Hyderabad"
```

```
matches$team1[matches$team1 == "Delhi Daredevils"] <- "Delhi Capitals"
matches$team1[matches$team1 == "Kings XI Punjab"] <- "Punjab Kings"
matches$team1[matches$team1 == "Gujarat Lions"] <- "Gujarat Titans"
matches$team1[matches$team1 == "Gujarat Lions"] <- "Gujarat Titans"
matches$team1[matches$team1 == "Rising Pune Supergiant"] <- "Lucknow Super Giants"
matches$team1[matches$team1 == "Rising Pune Supergiants"] <- "Lucknow Super Giants"
matches$team1[matches$team1 == "Pune Warriors"] <- "Lucknow Super Giants"

levels(as.factor(matches$team2))
```

```
##  [1] "Chennai Super Kings"         "Deccan Chargers"
##  [3] "Delhi Capitals"              "Delhi Daredevils"
##  [5] "Gujarat Lions"               "Kings XI Punjab"
##  [7] "Kochi Tuskers Kerala"        "Kolkata Knight Riders"
##  [9] "Mumbai Indians"              "Pune Warriors"
## [11] "Rajasthan Royals"            "Rising Pune Supergiant"
```

```
## [13] "Rising Pune Supergiants"     "Royal Challengers Bangalore"
## [15] "Sunrisers Hyderabad"
```

```
matches$team2[matches$team2 == "Delhi Daredevils"] <- "Delhi Capitals"
matches$team2[matches$team2 == "Kings XI Punjab"] <- "Punjab Kings"
matches$team2[matches$team2 == "Gujarat Lions"] <- "Gujarat Titans"
matches$team2[matches$team2 == "Gujarat Lions"] <- "Gujarat Titans"
matches$team2[matches$team2 == "Rising Pune Supergiant"] <- "Lucknow Super Giants"
matches$team2[matches$team2 == "Rising Pune Supergiants"] <- "Lucknow Super Giants"
matches$team2[matches$team2 == "Pune Warriors"] <- "Lucknow Super Giants"

levels(as.factor(matches$toss_winner))
```

```
##  [1] "Chennai Super Kings"         "Deccan Chargers"
##  [3] "Delhi Capitals"              "Delhi Daredevils"
##  [5] "Gujarat Lions"               "Kings XI Punjab"
##  [7] "Kochi Tuskers Kerala"        "Kolkata Knight Riders"
##  [9] "Mumbai Indians"              "Pune Warriors"
## [11] "Rajasthan Royals"            "Rising Pune Supergiant"
## [13] "Rising Pune Supergiants"     "Royal Challengers Bangalore"
## [15] "Sunrisers Hyderabad"
```

```
matches$toss_winner[matches$toss_winner == "Delhi Daredevils"] <- "Delhi Capitals"
matches$toss_winner[matches$toss_winner == "Kings XI Punjab"] <- "Punjab Kings"
matches$toss_winner[matches$toss_winner == "Gujarat Lions"] <- "Gujarat Titans"
matches$toss_winner[matches$toss_winner == "Gujarat Lions"] <- "Gujarat Titans"
matches$toss_winner[matches$toss_winner == "Rising Pune Supergiant"] <- "Lucknow Super Giants"
matches$toss_winner[matches$toss_winner == "Rising Pune Supergiants"] <- "Lucknow Super Giants"
matches$toss_winner[matches$toss_winner == "Pune Warriors"] <- "Lucknow Super Giants"

levels(as.factor(matches$winner))
```

```
##  [1] "Chennai Super Kings"         "Deccan Chargers"
##  [3] "Delhi Capitals"              "Delhi Daredevils"
##  [5] "Gujarat Lions"               "Kings XI Punjab"
##  [7] "Kochi Tuskers Kerala"        "Kolkata Knight Riders"
##  [9] "Mumbai Indians"              "Pune Warriors"
## [11] "Rajasthan Royals"            "Rising Pune Supergiant"
## [13] "Rising Pune Supergiants"     "Royal Challengers Bangalore"
## [15] "Sunrisers Hyderabad"
```

```
matches$winner[matches$winner == "Delhi Daredevils"] <- "Delhi Capitals"
matches$winner[matches$winner == "Kings XI Punjab"] <- "Punjab Kings"
matches$winner[matches$winner == "Gujarat Lions"] <- "Gujarat Titans"
matches$winner[matches$winner == "Gujarat Lions"] <- "Gujarat Titans"
matches$winner[matches$winner == "Rising Pune Supergiant"] <- "Lucknow Super Giants"
matches$winner[matches$winner == "Rising Pune Supergiants"] <- "Lucknow Super Giants"
matches$winner[matches$winner == "Pune Warriors"] <- "Lucknow Super Giants"

levels(as.factor(matches$winner))
```

```
##  [1] "Chennai Super Kings"         "Deccan Chargers"
```

```
##  [3] "Delhi Capitals"              "Gujarat Titans"
##  [5] "Kochi Tuskers Kerala"        "Kolkata Knight Riders"
##  [7] "Lucknow Super Giants"        "Mumbai Indians"
##  [9] "Punjab Kings"                "Rajasthan Royals"
## [11] "Royal Challengers Bangalore" "Sunrisers Hyderabad"
```

Methods & Analysis - Data Exploration & Results:

Let us start our exploration of data with finding out a few unique facts about our data.

```
# Let us Check some unique sets

# Total players
n_distinct(ball_by_ball$player)
```

```
## [1] 507
```

```
# Matches set
matches %>% summarize(
  venue=n_distinct(matches$venue),
  teams=n_distinct(c(unique(matches$team1),unique(matches$team2))),
  matches_played=(tot_mat_played = n()),
  noresults=nrow(matches[is.na(matches$winner),])
)
```

```
##   venue teams matches_played noresults
## 1    35    12            816         4
```

```
# Total players, extras are the number of times that
extras_data <- ball_by_ball %>% filter(ball_by_ball$extras_type %in% c("wides","noballs"))
nrow(extras_data)
```

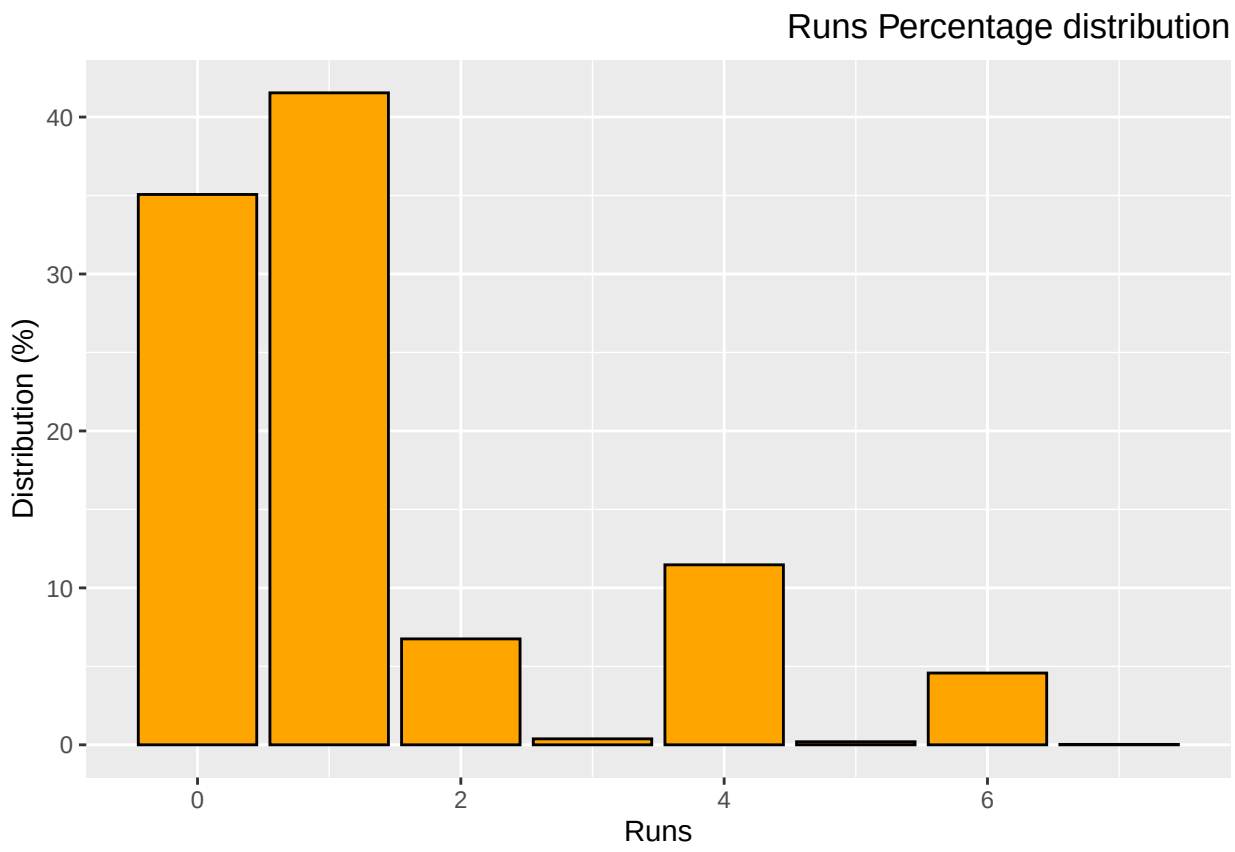
```
## [1] 6616
```

Conclusion: We see a total of 507 players in the ball_by_ball data set. We also have 35 distinct venues, a total of 12 teams , 816 matches played and 4 ended up in no results which means it was a draw. And a total of 6616 extras in wide and noballs. The reason are looking at wides and no balls is because the bowler has to bowl an extra ball for every noball or wide bowled.

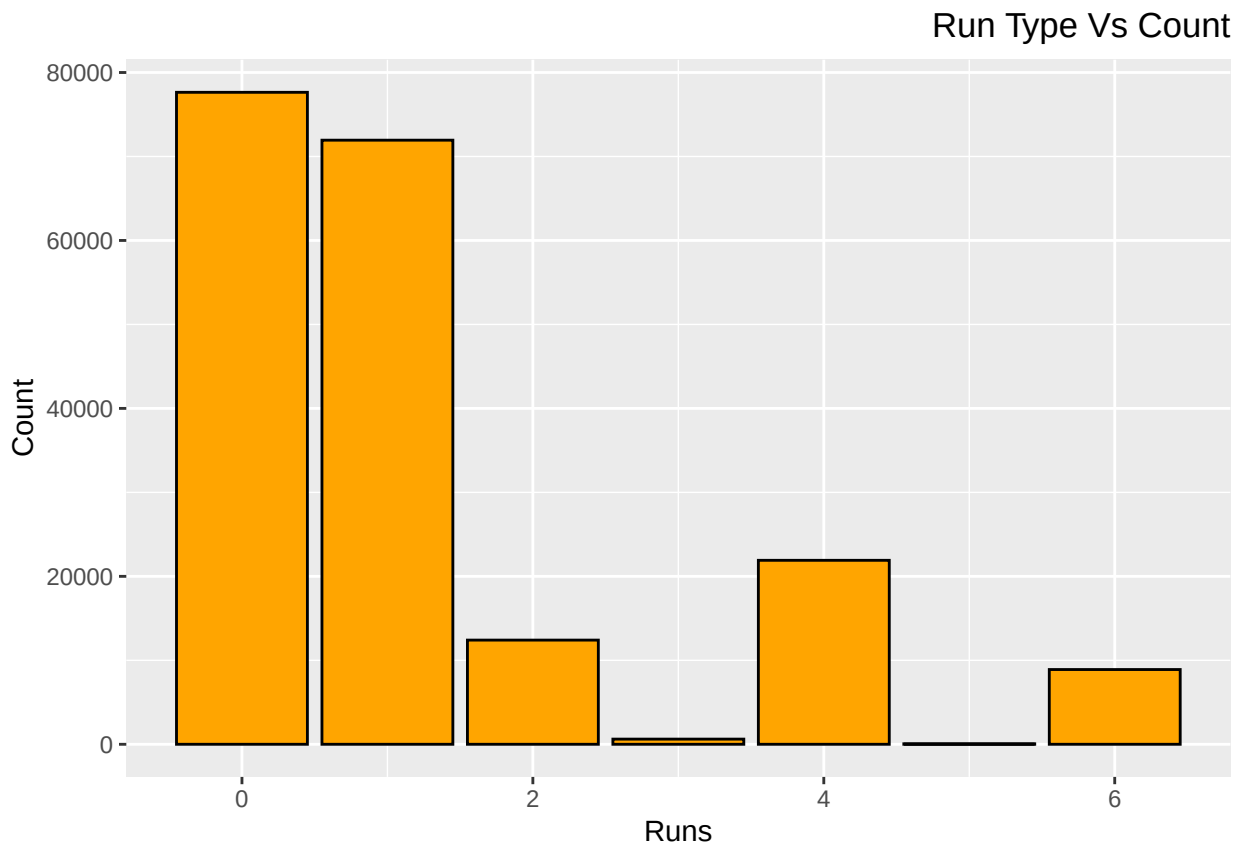
Now let us go into further analysis and check how many times a different run was scored by a player. Runs are scored by batsmen and bowlers bowl deliveries.

```
# Create a Player role

delivery <- ball_by_ball %>%
   gather(playerRole, player, batsman:bowler) %>%
  mutate(playerRole=as.factor(playerRole))

# Visual Representation: Let us look at how the runs are distributed as percentage

delivery %>% filter(playerRole == "batsman") %>%
  group_by(runs = total_runs) %>%
  summarize(count = n()) %>%
  mutate(distribution = percent(count / sum(count)))  %>%
  ggplot(aes(x = runs , y = distribution*100))+
  ggtitle("Runs Percentage distribution")+
  xlab("Runs")+
```

```
  ylab("Distribution (%)")+
  geom_bar(stat = "identity",  fill = "orange", color = "black")+
  theme(plot.title = element_text(hjust = 1.0))
```




```
# Visual Representation: Let us see number of times a batsman had scored a different run

delivery %>%
  filter(playerRole == "batsman") %>%
  group_by(run = batsman_runs) %>%
  summarize(count = n()) %>%
  mutate(percent = percent(count / sum(count))) %>%
  ggplot(aes(x = run , y = count))+
  ggtitle("Run Type Vs Count")+
  xlab("Runs")+
  ylab("Count")+
  geom_bar(stat = "identity",  fill = "orange", color = "black")+
  theme(plot.title = element_text(hjust = 1.0))
```



Conclusion: From the above visuals we do see that 7 runs is rare. So we will omit the 7 run. We also saw that there were were about 6616 extras.

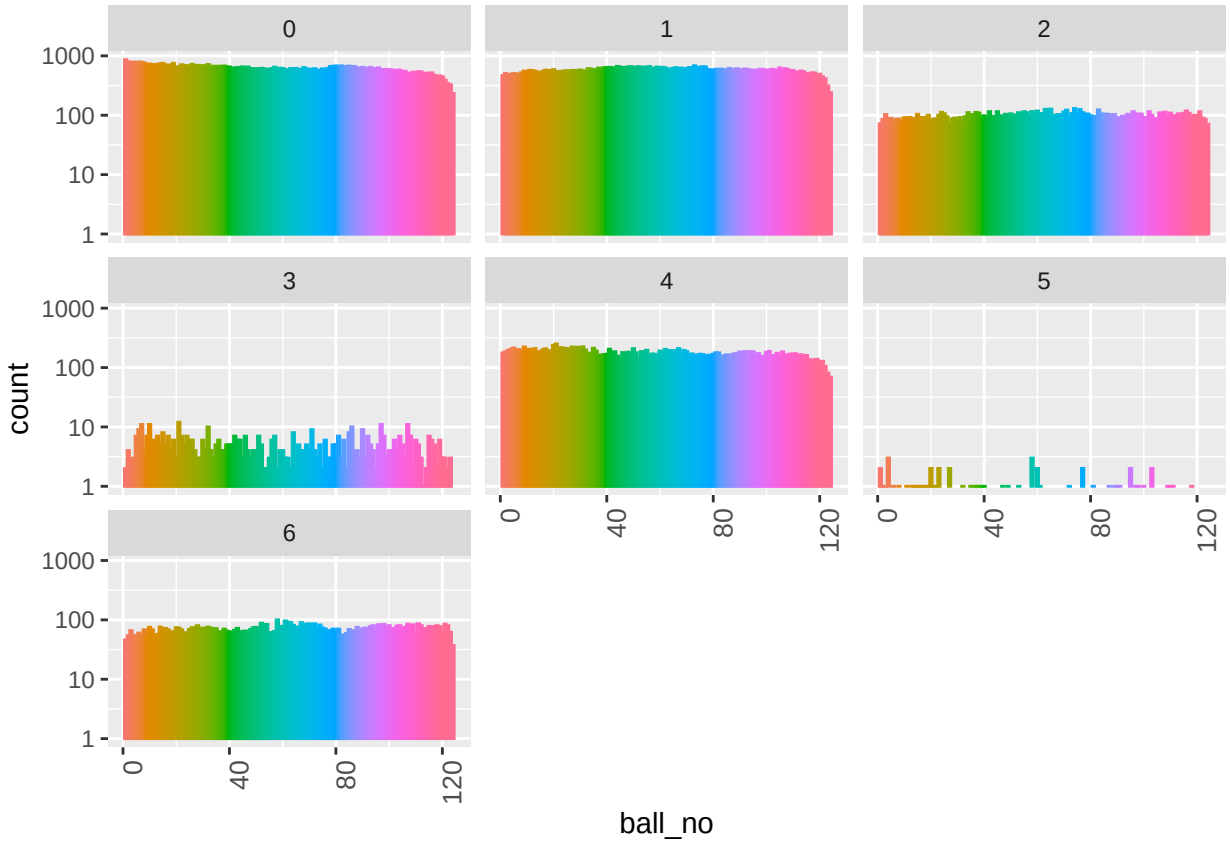
Lets us now look if there is a correlation between a ball and the type of run scored. There are 20 overs in a T20 match for each innings. Each over has 6 deliveries with a total of 120 deliveries. If we take the average number of extras we will find that there is about 4 deliveries in an innings.

```
# Major run types scored on each ball and correlation between them
runs_and_batsman <- delivery %>%
  group_by(id, inning) %>%
  mutate(ball_no = 1:n()) %>%
  ungroup() %>%
  filter(playerRole == "batsman") %>%
  filter(batsman_runs != "" & batsman_runs != "7" & ball_no %in% 1: 124) %>%
  group_by(ball_no, batsman_runs) %>%
  summarize(count=n())

runs_and_batsman
```

```
## # A tibble: 791 x 3
## # Groups:   ball_no [124]
##    ball_no batsman_runs count
##      <int>        <int> <int>
##  1       1            0   864
##  2       1            1   468
##  3       1            2    72
```

```
##  4       1            3     2
##  5       1            4   176
##  6       1            5     2
##  7       1            6    46
##  8       2            0   797
##  9       2            1   504
## 10       2            2    85
## # ... with 781 more rows
```

```
# Visualization on how runs stack up over the innings

runs_and_batsman %>%
  ggplot(aes(ball_no,count, col=factor(ball_no))) +
  geom_col() +
  facet_wrap( ~ batsman_runs) +
  scale_y_log10()+
  theme(axis.text.x = element_text(
    angle = 90,
    size = 10,
    hjust = 1
  ),
  legend.position = "none")
```




```
# Correlation
cor(runs_and_batsman$batsman_runs, as.numeric(runs_and_batsman$ball_no))
```

```
## [1] -0.02449583
```

Conclusion: From the visualization we see that there is no correlation between the delivery and the runs scored .

We can see that the number of no runs scored decreases as the iniings progresses. We also see the 1s and 2s are steady throughout the innings.

Most teams try to bat out their innings by stabilizing thoughtout the innings

```
# Run trends during the innings
delivery %>%
  group_by(id, inning) %>%
  mutate(ball_no = 1:n()) %>%
  ungroup() %>%
  filter (inning %in% 1:2 & playerRole == "batsman" & ball_no %in% 1:124) %>%
  group_by(ball_no=as.numeric(ball_no), runs = as.factor(batsman_runs)) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  arrange(ball_no) %>%
  top_n(5) %>%
  ggplot(aes(ball_no,count,col=runs, shape = runs)) +
  geom_point(size = 2)
```

Conclusion: From the visualization we see that 0s and 1's are the most prominent run types throughout the innings but are high in the first few overs and taper down towards the end of the innings. We also see

the 4s and the 6's higher during both the initial part of the innings. This is because of the power play in T20 where the field is restricted.

Lets look at how players performed during the IPL. Like all games the MVP dictates the player capability. We will look at the player_of_match field to pick the top 25 playes in the IPL so far.

Palyer Statistics : Batsman

```
# top 25 players by player_of_match
matches %>%
  group_by(mvp=player_of_match) %>%
  summarize(player_of_match= n()) %>%
  arrange(desc(player_of_match)) %>%
  head(n = 25)
```

```
## # A tibble: 25 x 2
##    mvp            player_of_match
##    <chr>                    <int>
##  1 AB de Villiers              23
##  2 CH Gayle                    22
##  3 RG Sharma                   18
##  4 DA Warner                   17
##  5 MS Dhoni                    17
##  6 SR Watson                   16
##  7 YK Pathan                   16
##  8 SK Raina                    14
##  9 G Gambhir                   13
## 10 V Kohli                     13
## # ... with 15 more rows
```

All teams look for batsman who are hard hitters , who can score boundaries at every ball. Let us see who are the players with maximum boundaries

```
# Top 25 basmen with boundaries (boundary is a 4 or a 6)
delivery %>%
  filter(playerRole == "batsman" & batsman_runs == 6 | batsman_runs == 4) %>%
  group_by(player, batsman_runs) %>%
  summarize(n = n()) %>%
  spread(batsman_runs, n) %>%
  rename(sixes = `6`,fours = `4`) %>%
  select(sixes, fours) %>%
  arrange(desc(sixes)) %>%
  head(25)
```

```
## # A tibble: 25 x 3
## # Groups:   player [25]
##    player         sixes fours
##    <chr>          <int> <int>
##  1 CH Gayle         349   810
##  2 AB de Villiers   235   724
##  3 MS Dhoni         216   613
##  4 RG Sharma        214   911
##  5 V Kohli          202  1055
##  6 KA Pollard       198   572
##  7 DA Warner        195   946
##  8 SK Raina         194  1045
##  9 SR Watson        190  1028
## 10 RV Uthappa       163   883
## # ... with 15 more rows
```

Let us look at the top 25 batsmen with the maximum strike rate (Runs scored vs deliveries faced). The higher the better

```
# Top 25 basmen 25 batsmen with the maximum strike rate (Runs scored vs deliveries faced)
delivery %>%
  filter(playerRole == "batsman") %>%
  group_by(player) %>%
  summarize(runs_scored=sum(batsman_runs), balls_faced=n(), strike_rate=sum(batsman_runs)/n()) %>%
  arrange(desc(strike_rate)) %>%
  head(25)
```

```
## # A tibble: 25 x 4
##    player          runs_scored balls_faced strike_rate
##    <chr>                 <int>       <int>       <dbl>
##  1 B Stanlake                5           2        2.5
##  2 Umar Gul                 39          19        2.05
##  3 RS Sodhi                  4           2        2
##  4 Shahid Afridi            81          46        1.76
##  5 I Malhotra                7           4        1.75
##  6 TU Deshpande             21          12        1.75
##  7 AD Russell             1517         882        1.72
##  8 LJ Wright               106          63        1.68
##  9 Abdul Samad             111          66        1.68
## 10 KMDN Kulasekara           5           3        1.67
## # ... with 15 more rows
```

Conclusion: From the above analysis we know that the top 25 playes with the maximum runs are also the top boundary hitters. Strike rate cannot be taken to look into the top players because it is dependent on the number of balls faced. The top 25 in strike rate do not figure in the top players.

Now that we have seen the batsmen let us also look at the top bowlers in the IPL. A bowler can be concluded as a good bowler by the number of wickets taken and their econoomy rate (the less number of runs an over)

Palyer Statistics : Bowler

```
# Let us check the the distict set of dimissal types
levels(as.factor(delivery$dismissal_kind))
```

```
## [1] "bowled"                "caught"                "caught and bowled"
## [4] "hit wicket"            "lbw"                   "obstructing the field"
## [7] "retired hurt"          "run out"               "stumped"
```

Top bowlers with maximum number of wickets

```
# Top 25 bowlers  with maximum number of wickets
delivery %>%
  filter(playerRole == "bowler" &
      dismissal_kind != "" & dismissal_kind != "retired hurt" & dismissal_kind != "run out" & dismissal
  ) %>%
  group_by(player) %>%
  summarise(total_wickets=n())%>%
  arrange(desc(total_wickets)) %>%
  head(25)
```

```
## # A tibble: 25 x 2
##    player          total_wickets
##    <chr>                   <int>
##  1 SL Malinga                170
##  2 A Mishra                  160
##  3 PP Chawla                 156
##  4 DJ Bravo                  153
##  5 Harbhajan Singh           150
##  6 R Ashwin                  138
##  7 B Kumar                   136
##  8 SP Narine                 127
##  9 YS Chahal                 121
## 10 UT Yadav                  119
## # ... with 15 more rows
```

Bowlers who conceded maximum runs

```
# Top 25 bowlers  with maximum number of runs conceded
delivery %>%
  filter(playerRole == "bowler")  %>%
  group_by(player) %>%
  summarise(runsconceded=sum(batsman_runs), deliveries_bowled=n())%>%
  arrange(desc(runsconceded)) %>%
  head(25)
```

```
## # A tibble: 25 x 3
##    player          runsconceded deliveries_bowled
##    <chr>                  <int>             <int>
##  1 PP Chawla               4196              3285
##  2 Harbhajan Singh         3868              3451
```

```
##  3 A Mishra                3788             3233
##  4 DJ Bravo                3659             2846
##  5 R Ashwin                3581             3327
##  6 UT Yadav                3446             2642
##  7 RA Jadeja               3408             2759
##  8 SL Malinga              3193             2974
##  9 B Kumar                 3132             2795
## 10 P Kumar                 3106             2637
## # ... with 15 more rows
```

Lets check for the bowlers with good economy rate, which is the runs conceded vs the deliveries bowled The lower the economy rate the better

```
# Top 25 bowlers  with best economy rate
delivery %>%
  filter(playerRole == "bowler") %>%
  group_by(player) %>%
  summarize(runs_conceded=sum(batsman_runs), deliveries_bowled=n(), economy_rate=sum(batsman_runs)/n())
  arrange(economy_rate) %>%
  head(25)
```

```
## # A tibble: 25 x 4
##    player        runs_conceded deliveries_bowled economy_rate
##    <chr>                 <int>             <int>        <dbl>
##  1 AC Gilchrist              0                 1        0
##  2 DA Warner                 1                 2        0.5
##  3 AS Roy                    8                15        0.533
##  4 NB Singh                 14                25        0.56
##  5 LA Carseldine             5                 7        0.714
##  6 SS Mundhe                 5                 7        0.714
##  7 Sachin Baby               8                10        0.8
##  8 AM Rahane                 5                 6        0.833
##  9 DJ Thornely              38                44        0.864
## 10 SM Harwood               61                67        0.910
## # ... with 15 more rows
```

Lets check for the bowlers with good strike rate, which is the wickets taken vs the deliveries bowled The lower the strike rate the better

```
# Top 25 bowlers  with best strike rate

t_wickets <- delivery %>%
  filter(playerRole == "bowler" & dismissal_kind != "" & dismissal_kind != "retired hurt" & dismissal_ki
  group_by(player) %>%
  summarize(total_wickets = n())

top_max_deliveries <- delivery %>%
  filter(playerRole == "bowler") %>%
  group_by(player) %>%
  summarize(total_delivery = n()) %>%
  arrange(desc(total_delivery))
```

```
top_max_deliveries %>%
  full_join(t_wickets, by = "player") %>%
  mutate(strike_rate = total_delivery/ total_wickets) %>%
  arrange(strike_rate) %>%
  head(25)
```

```
## # A tibble: 25 x 4
##    player        total_delivery total_wickets strike_rate
##    <chr>                  <int>         <int>       <dbl>
##  1 AC Gilchrist               1             1        1
##  2 Sachin Baby               10             2        5
##  3 AM Rahane                  6             1        6
##  4 LA Carseldine              7             1        7
##  5 SS Mundhe                  7             1        7
##  6 DAJ Bracewell             25             3        8.33
##  7 AS Joseph                 55             6        9.17
##  8 Shoaib Akhtar             46             5        9.2
##  9 D du Preez                43             4       10.8
## 10 R Ninan                   36             3       12
## # ... with 15 more rows
```

Conclusion: From the above analysis there are bowlers who show up in the top 25 because they have bowled lesser number of deliveries. If the total number of wickets are taken into consideration as compared to the higest wicket takers its a few and these would not matter. We can eliminate the bowler strike rate during modeling our algorithm.

Let is now see how the top batsmen contributed to the overall IPL totals

```
# The total runs in the IPL
total_ipl_runs <- delivery %>%
  filter(playerRole == "batsman") %>%
  summarize(total_runs_ipl = sum(total_runs))
total_ipl_runs
```

```
##   total_runs_ipl
## 1         252794
```

```
# The total runs by the top 25 batsman in the IPL
total_runs_by_top_25 <-  delivery %>%
  filter(playerRole == "batsman") %>%
  group_by(player) %>%
  summarize(tot_runs_top_batsmman = sum(batsman_runs))%>%
  arrange(desc(tot_runs_top_batsmman)) %>%
  head(n = 25)

sum(total_runs_by_top_25$tot_runs_top_batsmman)
```

```
## [1] 96321
```

```
# What is the percentage that they contributed

percent(sum(total_runs_by_top_25$tot_runs_top_batsmman)/total_ipl_runs$total_runs_ipl)
```

```
## [1] 38.10%
```

Conclusion: From the above analysis the top 25 batsmen contribute to 38% of the total runs in the IPL.

Let is now see how the top boowlers contributed to the overall IPL wickets

```
# The total wickets in the IPL
total_ipl_wickets <- delivery %>%
  filter(playerRole == "bowler" & dismissal_kind != "") %>%
  summarize(overall_wickets = n())

total_ipl_wickets
```

```
##   overall_wickets
## 1            9495
```

```
# The wickets runs by the top 25 bowlers in the IPL
total_wickets_by_top_25 <- delivery %>%
  filter(playerRole == "bowler" &
           dismissal_kind != "" & dismissal_kind != "retired hurt" & dismissal_kind != "run out" & dism
  ) %>%
  group_by(player) %>%
  summarise(total_wickets=n())%>%
  arrange(desc(total_wickets)) %>%
  head(25)

total_wickets_by_top_25
```

```
## # A tibble: 25 x 2
##    player          total_wickets
##    <chr>                   <int>
##  1 SL Malinga                170
##  2 A Mishra                  160
##  3 PP Chawla                 156
##  4 DJ Bravo                  153
##  5 Harbhajan Singh           150
##  6 R Ashwin                  138
##  7 B Kumar                   136
##  8 SP Narine                 127
##  9 YS Chahal                 121
## 10 UT Yadav                  119
## # ... with 15 more rows
```

```
# What is the percentage that they contributed

percent(sum(total_wickets_by_top_25$total_wickets)/total_ipl_wickets)
```

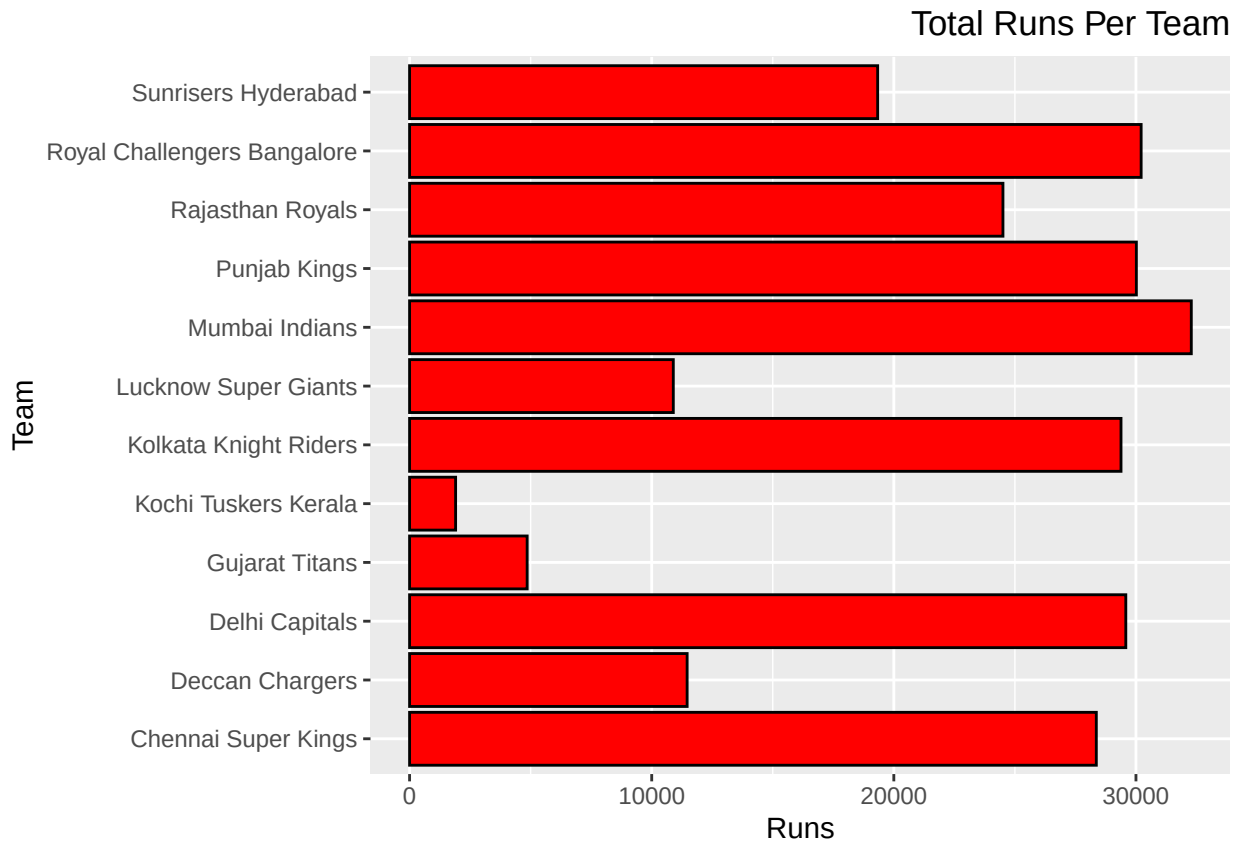
```
## [1] 30.30%
```

Conclusion: From the above analysis the top 25 bowlers contibute to 30% of the total wickets taken in the IPL

Team Statistics

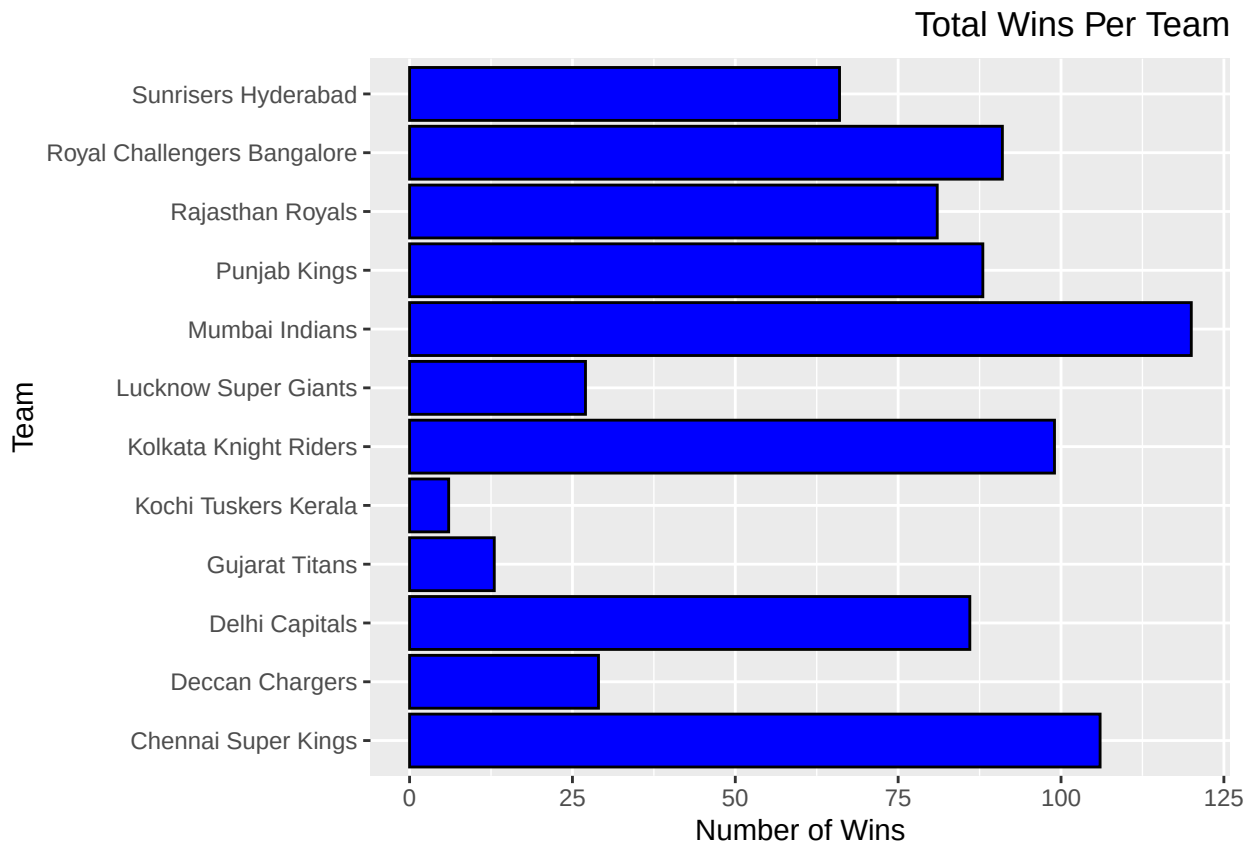
Let us analyze some Team statistics

```
# Visual Total runs scored by each team

delivery %>% filter(playerRole == "batsman") %>%
  group_by(team = batting_team) %>%
  summarize(teamruns = sum(total_runs)) %>%
  ggplot(aes(x = teamruns , y = team))+
  ggtitle("Total Runs Per Team")+
  xlab("Runs")+
  ylab("Team")+
  geom_bar(stat = "identity",  fill = "red", color = "black")+
  theme(plot.title = element_text(hjust = 1.0))
```




```
# Match Winners

matches %>%
  group_by(team = winner) %>%
  filter(winner != "") %>%
  summarize(count = n()) %>%
  ggplot(aes(x = count , y = team))+
  ggtitle("Total Wins Per Team")+
  xlab("Number of Wins")+
```

```
  ylab("Team")+
  geom_bar(stat = "identity",  fill = "blue", color = "black")+
  theme(plot.title = element_text(hjust = 1.0))
```



Conclusion: The top 3 teams are Mumbai Indians, Chennai Super Kinds and Kolkata Knight Riders , both in terms of wins and runs.

Does winning a toss matter to the team and does electing to field or bat matter to the team

```
# Which team won how many tosses
toss_winners <- matches %>%
  group_by(toss_winner) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  rename(team = toss_winner, number_of_tosses_won = count)
toss_winners
```

```
## # A tibble: 12 x 2
##    team                        number_of_tosses_won
##    <chr>                                      <int>
##  1 Mumbai Indians                               106
##  2 Delhi Capitals                               100
##  3 Kolkata Knight Riders                         98
##  4 Chennai Super Kings                           97
##  5 Rajasthan Royals                              87
##  6 Royal Challengers Bangalore                   87
```

```
##  7 Punjab Kings                                85
##  8 Sunrisers Hyderabad                         57
##  9 Deccan Chargers                             43
## 10 Lucknow Super Giants                        33
## 11 Gujarat Titans                              15
## 12 Kochi Tuskers Kerala                         8
```

```r
# Matches played by teams
team1_played <- matches %>%
  group_by(team1) %>%
  summarize(count1 = n()) %>%
  arrange(team1) %>%
  rename(team = team1)

team2_played <- matches %>%
  group_by(team2) %>%
  summarize(count2 = n()) %>%
  arrange(team2) %>%
  rename(team = team2)

matches_team <- team1_played %>%
  full_join(team2_played, by = "team")

matches_team
```

```
## # A tibble: 12 x 3
##    team                        count1 count2
##    <chr>                        <int>  <int>
##  1 Chennai Super Kings             94     84
##  2 Deccan Chargers                 39     36
##  3 Delhi Capitals                 102     92
##  4 Gujarat Titans                  16     14
##  5 Kochi Tuskers Kerala             7      7
##  6 Kolkata Knight Riders           95     97
##  7 Lucknow Super Giants            37     39
##  8 Mumbai Indians                  97    106
##  9 Punjab Kings                    92     98
## 10 Rajasthan Royals                70     91
## 11 Royal Challengers Bangalore    108     87
## 12 Sunrisers Hyderabad             59     65
```

```r
# Number of wins by eacxh team
winners <- matches %>%
  group_by(winner) %>%
  filter(winner != "") %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  rename(team = winner, matches_won = count)
winners
```

```
## # A tibble: 12 x 2
##    team                        matches_won
##    <chr>                             <int>
```

```
##  1 Mumbai Indians                       120
##  2 Chennai Super Kings                  106
##  3 Kolkata Knight Riders                 99
##  4 Royal Challengers Bangalore           91
##  5 Punjab Kings                          88
##  6 Delhi Capitals                        86
##  7 Rajasthan Royals                      81
##  8 Sunrisers Hyderabad                   66
##  9 Deccan Chargers                       29
## 10 Lucknow Super Giants                  27
## 11 Gujarat Titans                        13
## 12 Kochi Tuskers Kerala                   6
```

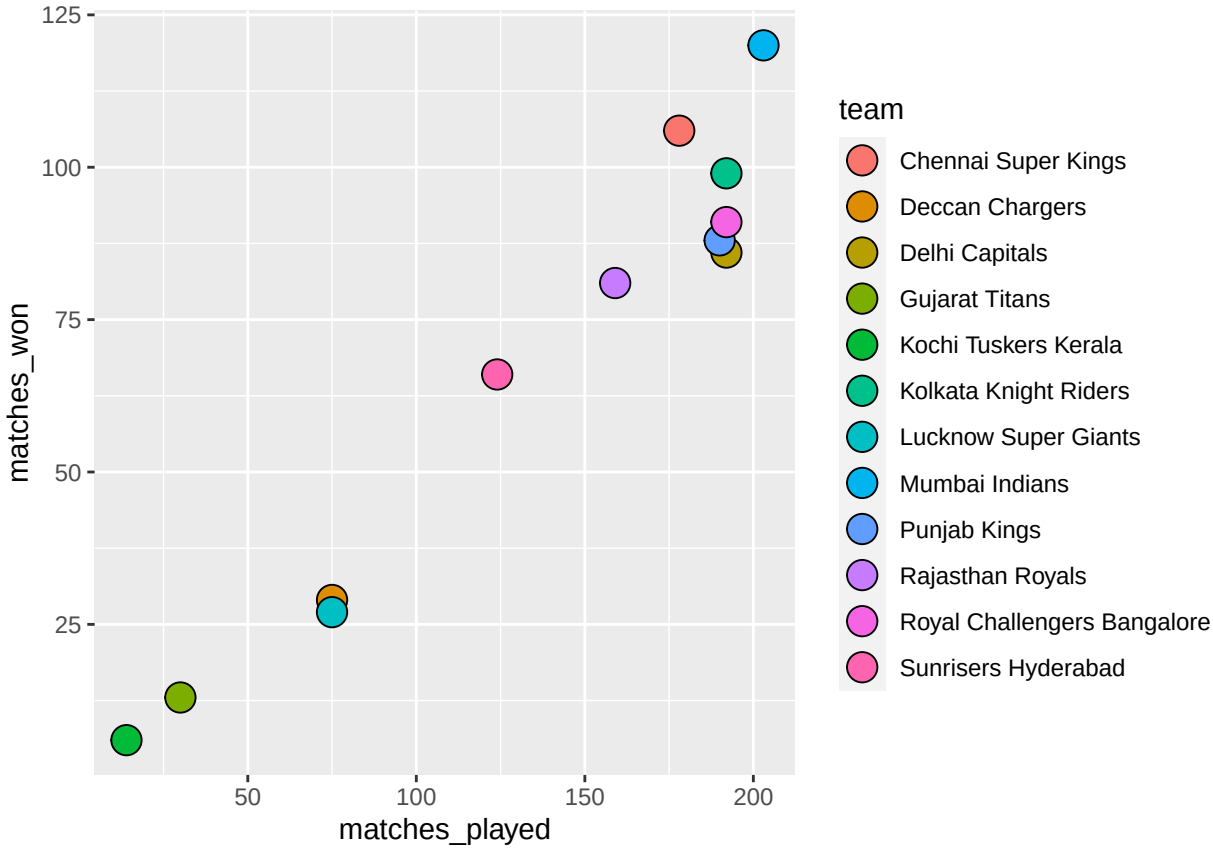
```
# Number of losses by each team

team1_lost <- matches %>%
  filter(winner != "") %>%
  filter(as.character(winner) != as.character(team1)) %>%
  group_by(team1) %>%
  summarize(count1 = n()) %>%
  arrange(team1) %>%
  rename(team = team1)
team2_lost <- matches %>%
  filter(winner != "") %>%
  filter(as.character(winner) != as.character(team2)) %>%
  group_by(team2) %>%
  summarize(count2 = n()) %>%
  arrange(team2) %>%
  rename(team = team2)
losers <- team1_lost %>%
  full_join(team2_lost, by = "team") %>%
  mutate(matches_lost = count1 + count2) %>%
  select(-count1,-count2) %>%
  arrange(desc(matches_lost))
losers
```

```
## # A tibble: 12 x 2
##    team                         matches_lost
##    <chr>                               <int>
##  1 Delhi Capitals                        106
##  2 Punjab Kings                          102
##  3 Royal Challengers Bangalore           101
##  4 Kolkata Knight Riders                  93
##  5 Mumbai Indians                         83
##  6 Rajasthan Royals                       78
##  7 Chennai Super Kings                    72
##  8 Sunrisers Hyderabad                    58
##  9 Lucknow Super Giants                   48
## 10 Deccan Chargers                        46
## 11 Gujarat Titans                         17
## 12 Kochi Tuskers Kerala                    8
```

```r
# Team matches , wins and losses
teams <- matches_team %>%
  right_join(toss_winners, by = "team")%>%
  right_join(winners, by = "team") %>%
  right_join(losers, by = "team") %>%
  mutate(matches_played = matches_won + matches_lost)

teams
```

```
## # A tibble: 12 x 7
##    team   count1 count2 number_of_tosse~ matches_won matches_lost matches_played
##    <chr>   <int>  <int>            <int>       <int>        <int>          <int>
##  1 Chenn~     94     84               97         106           72            178
##  2 Decca~     39     36               43          29           46             75
##  3 Delhi~    102     92              100          86          106            192
##  4 Gujar~     16     14               15          13           17             30
##  5 Kochi~      7      7                8           6            8             14
##  6 Kolka~     95     97               98          99           93            192
##  7 Luckn~     37     39               33          27           48             75
##  8 Mumba~     97    106              106         120           83            203
##  9 Punja~     92     98               85          88          102            190
## 10 Rajas~     70     91               87          81           78            159
## 11 Royal~    108     87               87          91          101            192
## 12 Sunri~     59     65               57          66           58            124
```

```r
# Visualization of matches polayes vs won
teams %>%
  ggplot(aes(matches_played , matches_won, fill = team)) +
  geom_point(shape=21, size=5)
```



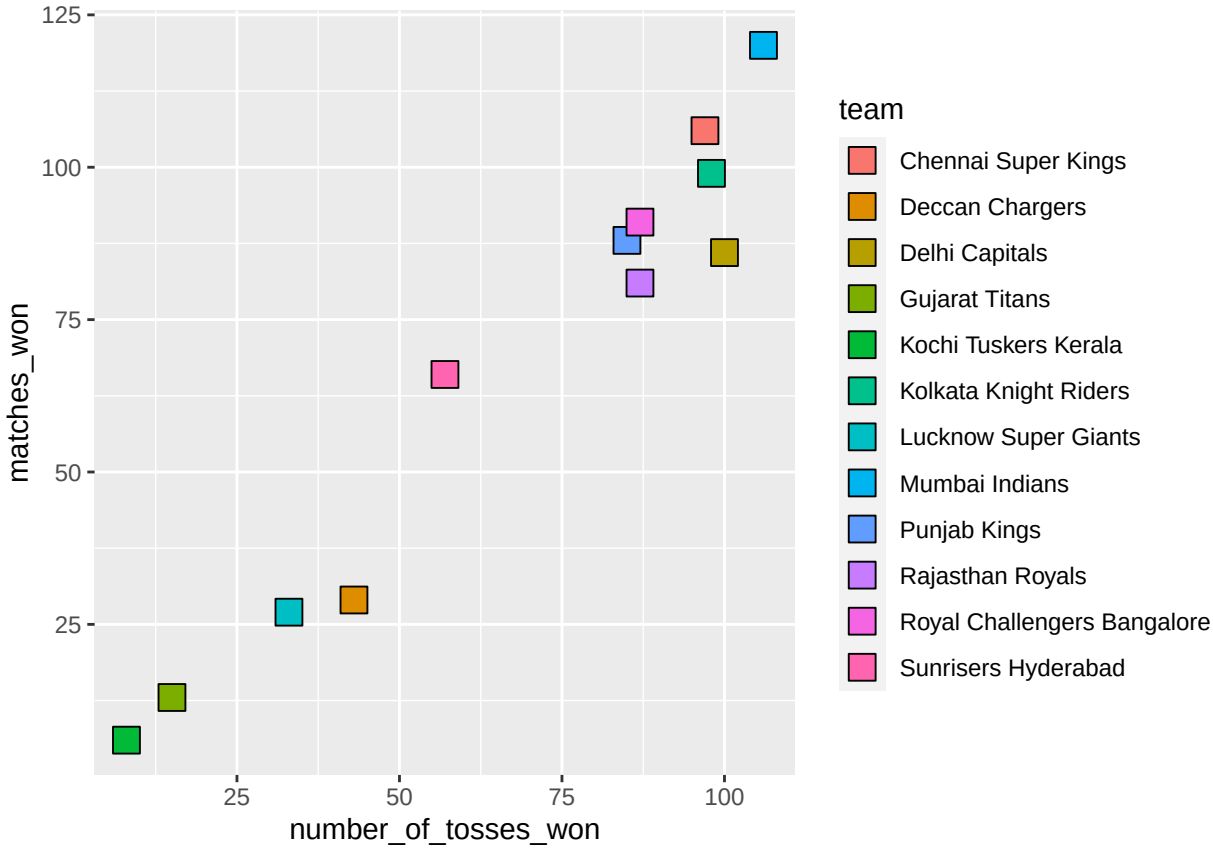

```
# Correlation between mamtches played to matches won
cor(teams$matches_played, teams$matches_won)
```

```
## [1] 0.9721357
```

Conclusion: The more the matches the more wins. The correlation between matches played to matches won is also positive

How does matches won to tosses one correlate

```
# Lets see the correlation between matches won and tosses won

teams %>%
  ggplot(aes(number_of_tosses_won, matches_won, fill = team)) +
  geom_point(shape=22, size = 5)
```




```
cor(teams$matches_won, teams$number_of_tosses_won)
```

```
## [1] 0.976548
```

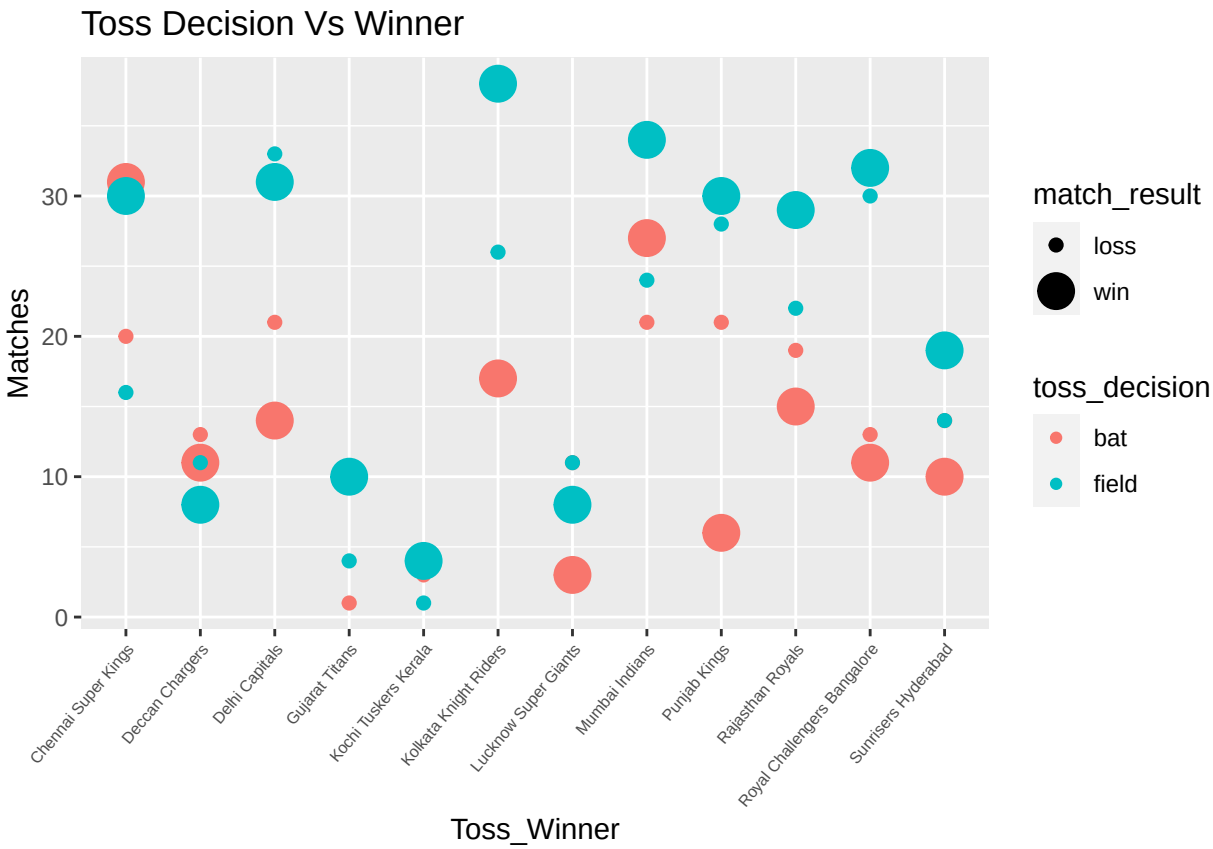
Conclusion: Teams that won tosses mostly won matches as well. And there is a high correlation between the two

Let us now check the decision to bat or field after winning the toss affects the wins

```
# Toss wins and toss decisions

toss_winners <- matches %>%
  group_by(toss_winner) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  rename(team = toss_winner, number_of_tosses_won = count)


toss_wins_results <- matches %>%
  filter(winner != "") %>%
  mutate(match_result = ifelse(toss_winner==winner, "win", "loss")) %>%
  group_by(toss_winner, toss_decision, match_result) %>%
  summarize(match_result_count = n())

toss_wins_results
```

```
## # A tibble: 46 x 4
## # Groups:   toss_winner, toss_decision [24]
##    toss_winner         toss_decision match_result match_result_count
##    <chr>               <chr>         <chr>                     <int>
##  1 Chennai Super Kings bat           loss                         20
##  2 Chennai Super Kings bat           win                          31
##  3 Chennai Super Kings field         loss                         16
##  4 Chennai Super Kings field         win                          30
##  5 Deccan Chargers     bat           loss                         13
##  6 Deccan Chargers     bat           win                          11
##  7 Deccan Chargers     field         loss                         11
##  8 Deccan Chargers     field         win                           8
##  9 Delhi Capitals      bat           loss                         21
## 10 Delhi Capitals      bat           win                          14
## # ... with 36 more rows
```

```
# Visualization : Effect of toss wins vs decision to bat or field
toss_wins_results %>%
  ggplot(aes(toss_winner, match_result_count, col = toss_decision, size = match_result)) +
  geom_point()  +
  ggtitle("Toss Decision Vs Winner")+
  xlab("Toss_Winner")+
  ylab("Matches")+
  theme(axis.text.x = element_text(
    angle = 50,
    size = 6,
    hjust = 1
  ),
  legend.position = "right")
```

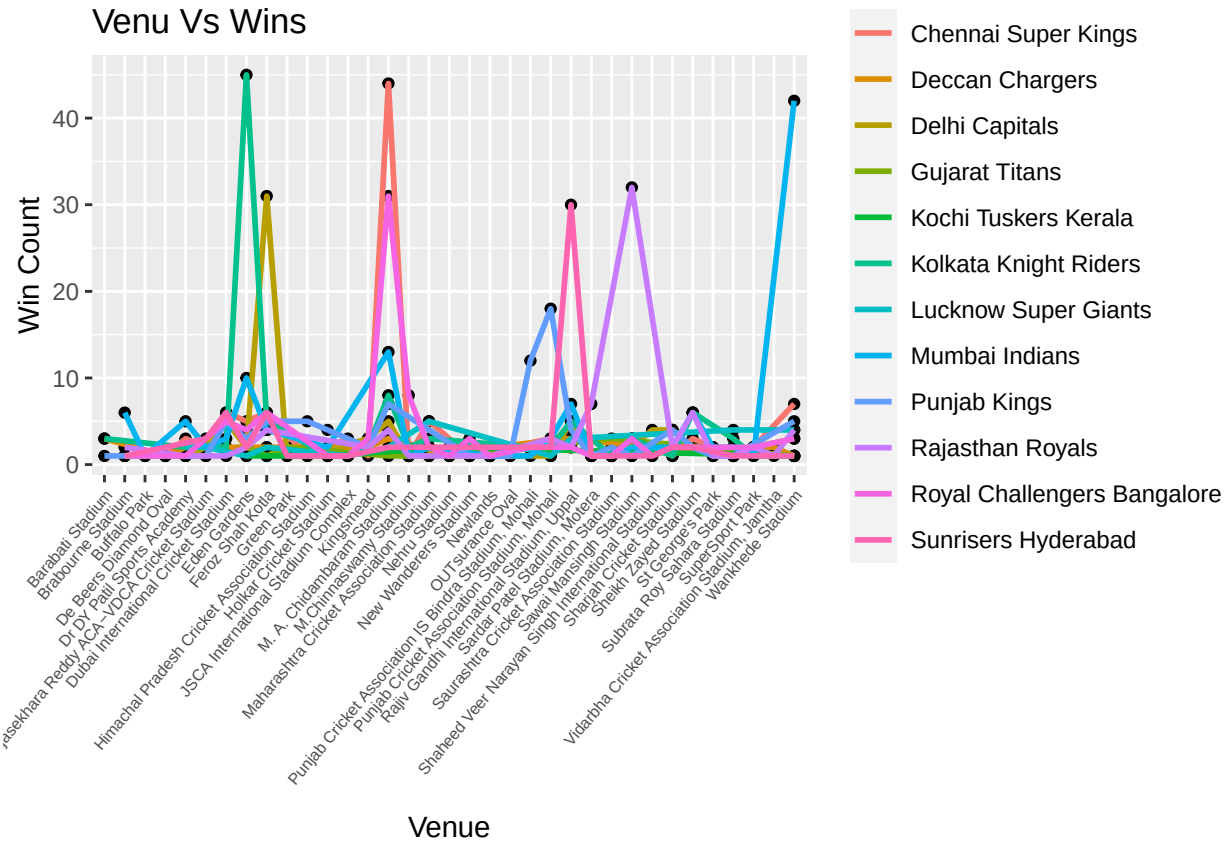


Conclusion: From the plot we see that the teams that have elected to field first after winning the toss have won more games (bigger circles). The decision to win the toss thus effects the result of a match.

Let us check if venue plays a part in the result of a game

At this point, let us check how many runs were scored batting first and how many were scored batting second in totals and run type wise.

```
# Venue effect on team wins
matches %>%
  filter(winner != "") %>%
  group_by(venue, winner) %>%
  summarize(count = n()) %>%
  ggplot(aes(venue, count, group = winner)) +
  geom_point() +
  ggtitle("Venu Vs Wins")+
  xlab("Venue")+
  ylab("Win Count")+
  geom_line(aes(col = winner), size = 1) +
  theme(axis.text.x = element_text(
    angle = 50,
    size = 6,
    hjust = 1
  ),
  legend.position = "right", legend.title.align=0)
```



Conclusion: From the plot we can see that the home venues result in more wins ffor the teams as inidicated by the peaks

We have enough data to Start our model

Model 1 : Build a model to rank the best players depending on his value

Factors determining the value of a player 1) High Strike Rate (Runs/deliveries faced) for a batsman 2) Low Economy Rate (Runs given/ deliveries bowled) 3) Socre against top bowlers and economy against top batsmen 4) Runs Scored by the team and wickets dismissed

High Strike Rate (Runs/deliveries faced) for a batsman

Low Economy Rate (Runs given/ deliveries bowled)

Let us determine the batsmen with highest strike rates. For this let is create a table that contains the averages

```
# batsmen stats
batsmen_avgs <- delivery %>%
  filter(playerRole == "batsman") %>%
  group_by(player) %>%
  summarize(total_deliveries = n(), tot_runs = sum(batsman_runs)) %>%
  summarize (
    avg_deliveries = mean(total_deliveries),
    avg_runs = mean(tot_runs),
    median_deliveries = median(total_deliveries),
    median_runs = median(tot_runs)
  )
as_tibble(batsmen_avgs)
```

```
## # A tibble: 1 x 4
##   avg_deliveries avg_runs median_deliveries median_runs
##            <dbl>    <dbl>             <int>       <int>
## 1           360.     447.                74          76
```

```
# bowler stats
bowler_avgs <- delivery %>%
  filter(playerRole == "bowler") %>%
  group_by(player) %>%
  summarize(total_deliveries = n(), tot_runs = sum(batsman_runs)) %>%
  summarize (
    avg_deliveries = mean(total_deliveries),
    avg_runs = mean(tot_runs),
    median_deliveries = median(total_deliveries),
    median_runs = median(tot_runs)
  )
as_tibble(bowler_avgs)
```

```
## # A tibble: 1 x 4
##   avg_deliveries avg_runs median_deliveries median_runs
##            <dbl>    <dbl>             <dbl>       <dbl>
## 1           461.     571.              200.        270.
```

Model: Reguliration - We will calculate the strike rate and economy rates using regularization. As we have already seen during our analysis the strike rate can be high but most of the players with high strike rates do not figure into the top 25 players. If they face just 2 deliveries and hit a boundary in the first their strike rate can be very high. Hence we adjust this by using median as the Error.

```
strike_rates_regularized <- delivery %>%
  filter(playerRole == "batsman") %>%
  group_by(player) %>%
```

```
  summarize(batsman_reg_str_rate = (sum(batsman_runs) + batsmen_avgs$median_runs) / (n() +
                                                                    batsmen_avgs$median_deliveries)) %>%
  arrange(desc(batsman_reg_str_rate))

strike_rates_regularized %>%
  head(25)
```

```
## # A tibble: 25 x 2
##    player         batsman_reg_str_rate
##    <chr>                         <dbl>
##  1 AD Russell                     1.67
##  2 N Pooran                       1.50
##  3 SP Narine                      1.50
##  4 AB de Villiers                 1.48
##  5 V Sehwag                       1.47
##  6 HH Pandya                      1.47
##  7 GJ Maxwell                     1.45
##  8 RR Pant                        1.45
##  9 CH Morris                      1.44
## 10 BCJ Cutting                    1.43
## # ... with 15 more rows
```

```
economy_rates_regularized <- delivery %>%
  filter(playerRole == "bowler") %>%
  group_by(player) %>%
  summarize(bowler_reg_eco_rate = (sum(batsman_runs) + bowler_avgs$median_runs) /
              (n() + bowler_avgs$median_deliveries)) %>%
  arrange(bowler_reg_eco_rate)
economy_rates_regularized %>%
  head(25)
```

```
## # A tibble: 25 x 2
##    player         bowler_reg_eco_rate
##    <chr>                        <dbl>
##  1 Rashid Khan                   1.06
##  2 M Muralitharan                1.07
##  3 DW Steyn                      1.08
##  4 SL Malinga                    1.09
##  5 R Ashwin                      1.09
##  6 A Kumble                      1.10
##  7 SP Narine                     1.11
##  8 Sohail Tanvir                 1.11
##  9 SW Tait                       1.13
## 10 DP Nannes                     1.13
## # ... with 15 more rows
```

Now let us merge the two sets and get the top players and rank them. When we merge these two sets we will have a few not applicable values , the reason is because a batsman may not be a bowler and vice versa. If a Batsman is nt a bowler we will use the avg balls and for bwlers who are not batsman we will use the median runs

```r
# Top rate players based on strike rates & economy rates
top_players <- strike_rates_regularized %>%
  full_join(economy_rates_regularized, by = "player") %>%
  mutate(batsman_reg_str_rate = replace_na(
    batsman_reg_str_rate,
    batsmen_avgs$median_runs / batsmen_avgs$avg_deliveries
  )) %>%
  mutate(bowler_reg_eco_rate = replace_na(bowler_reg_eco_rate, bowler_avgs$avg_runs /
                                    bowler_avgs$median_deliveries)) %>%
  mutate(value_of_player = 100 * (batsman_reg_str_rate + 1 / bowler_reg_eco_rate))
top_players %>%
  arrange(desc(value_of_player)) %>%
  select(player, value_of_player) %>%
  mutate(rank = row_number()) %>%
  head(100) %>%
  knitr::kable()
```

player	value_of_player	rank
SP Narine	239.6926	1
AD Russell	238.1415	2
CH Morris	226.0868	3
CH Gayle	224.3507	4
MM Ali	221.8452	5
JC Archer	220.9454	6
YK Pathan	220.6706	7
KH Pandya	220.0010	8
HH Pandya	219.1316	9
Rashid Khan	218.9008	10
GJ Maxwell	218.5992	11
KA Pollard	217.6115	12
Harbhajan Singh	216.3953	13
SR Watson	215.5835	14
SK Raina	214.9529	15
K Gowtham	214.3356	16
V Sehwag	214.0680	17
KK Cooper	213.6447	18
JA Morkel	213.1761	19
BCJ Cutting	213.1244	20
MF Maharoof	211.5580	21
DA Warner	211.1297	22
Mohammad Nabi	210.3603	23
Bipul Sharma	209.6259	24
KP Pietersen	209.2578	25
Shahid Afridi	209.2227	26
R Tewatia	208.9314	27
ST Jayasuriya	207.9850	28
RN ten Doeschate	207.6467	29
CR Brathwaite	205.9549	30
AC Gilchrist	205.7979	31
RG Sharma	205.3444	32
Yuvraj Singh	205.0433	33
BA Stokes	204.7791	34
N Rana	204.6793	35
AR Patel	204.5710	36
Umar Gul	204.4119	37
M Morkel	204.3903	38
SA Yadav	204.2229	39
JP Duminy	203.9741	40
SN Khan	203.7613	41
STR Binny	203.6427	42
SM Curran	203.6399	43
JD Ryder	203.5471	44
A Ashish Reddy	203.3675	45
LJ Wright	203.1412	46
KS Williamson	202.9263	47
DR Smith	202.6501	48
A Symonds	202.6235	49
Shakib Al Hasan	202.1672	50
SM Pollock	201.8618	51
RA Jadeja	201.3684	52
V Kohli	201.3601	53
RA Tripathi	200.9361	54
Abdul Samad	200.6530	55
DJ Bravo	200.5200	56
C de Grandhomme	200.2356	57

This table gives us the top players bowlers , batsmen and allrounders and theur value.

We will now introduce an anamoly which is how the top bowlers and batsmen performed against each other

```
# Top 25 batsman and bowlers
top_batsmen <- strike_rates_regularized %>%
  head(25)
top_batsmen
```

```
## # A tibble: 25 x 2
##    player         batsman_reg_str_rate
##    <chr>                         <dbl>
##  1 AD Russell                     1.67
##  2 N Pooran                       1.50
##  3 SP Narine                      1.50
##  4 AB de Villiers                 1.48
##  5 V Sehwag                       1.47
##  6 HH Pandya                      1.47
##  7 GJ Maxwell                     1.45
##  8 RR Pant                        1.45
##  9 CH Morris                      1.44
## 10 BCJ Cutting                    1.43
## # ... with 15 more rows
```

```
top_bowlers <- economy_rates_regularized %>%
  head(25)
top_bowlers
```

```
## # A tibble: 25 x 2
##    player         bowler_reg_eco_rate
##    <chr>                        <dbl>
##  1 Rashid Khan                   1.06
##  2 M Muralitharan                1.07
##  3 DW Steyn                      1.08
##  4 SL Malinga                    1.09
##  5 R Ashwin                      1.09
##  6 A Kumble                      1.10
##  7 SP Narine                     1.11
##  8 Sohail Tanvir                 1.11
##  9 SW Tait                       1.13
## 10 DP Nannes                     1.13
## # ... with 15 more rows
```

```
# batsmen strike rate against top_bowlers bowlers
# we will use the original set

top_25_bat_strikerate<- ball_by_ball %>%
  filter(bowler %in% top_bowlers$player) %>%
  group_by(player = batsman) %>%
  summarize(strike_rate = (sum(batsman_runs) + batsmen_avgs$median_runs) / (n() +batsmen_avgs$avg_deliv
  arrange(desc(strike_rate))
top_25_bat_strikerate %>%
  mutate(rank = row_number())%>%
  head(25)
```

```
## # A tibble: 25 x 3
##    player         strike_rate  rank
##    <chr>                <dbl> <int>
##  1 SR Watson            0.901     1
##  2 AB de Villiers       0.895     2
##  3 DA Warner            0.869     3
##  4 RV Uthappa           0.850     4
##  5 SK Raina             0.841     5
##  6 CH Gayle             0.807     6
##  7 MS Dhoni             0.801     7
##  8 YK Pathan            0.793     8
##  9 V Kohli              0.768     9
## 10 KL Rahul             0.766    10
## # ... with 15 more rows
```

```
# bowler ecorate rate against top_bowlers bowlers
# we will use the original set

top_25_bow_ecorate <- ball_by_ball %>%
  filter(batsman %in% top_batsmen$player) %>%
  group_by(player = bowler) %>%
  summarize(eco_rate = (sum(batsman_runs) + bowler_avgs$avg_runs) / (n() +bowler_avgs$median_deliveries)
  arrange(eco_rate)
top_25_bow_ecorate %>%
  mutate(rank = row_number())%>%
  head(25)
```

```
## # A tibble: 25 x 3
##    player          eco_rate  rank
##    <chr>              <dbl> <int>
##  1 R Ashwin            1.69     1
##  2 Rashid Khan         1.69     2
##  3 JJ Bumrah           1.70     3
##  4 Harbhajan Singh     1.71     4
##  5 SP Narine           1.74     5
##  6 B Kumar             1.76     6
##  7 YS Chahal           1.77     7
##  8 DS Kulkarni         1.80     8
##  9 DJ Bravo            1.81     9
## 10 DW Steyn            1.82    10
## # ... with 15 more rows
```

Now let us merge the two sets and get the top players and rank them. When we merge these two sets we will have a few not applicable values , the reason is because a batsman may not be a bowler and vice versa. If a Batsman is not a bowler we will use the avg balls and for bowlers who are not batsman we will use the median runs

```
# Top class players based on strike rates & economy rates against each other
top_class_players <- top_25_bat_strikerate %>%
  full_join(top_25_bow_ecorate, by = "player") %>%
  mutate(strike_rate = replace_na(strike_rate,
    batsmen_avgs$median_runs / batsmen_avgs$avg_deliveries
  )) %>%
```

```
  mutate(eco_rate = replace_na(eco_rate, bowler_avgs$avg_runs /
                                         bowler_avgs$median_deliveries)) %>%
  mutate(value_of_player = 100 * (strike_rate + 1 / eco_rate))


top_class_players %>%
  arrange(desc(value_of_player)) %>%
  select(player, value_of_player, strike_rate,eco_rate) %>%
  mutate(rank = row_number()) %>%
  head(100) %>%
  knitr::kable()
```

player	value_of_player	strike_rate	eco_rate	rank
SR Watson	140.41070	0.9013531	1.989044	1
SK Raina	126.43855	0.8409679	2.361734	2
YK Pathan	125.17856	0.7927657	2.178555	3
AB de Villiers	124.45383	0.8953332	2.863647	4
DA Warner	122.14993	0.8694098	2.840187	5
RV Uthappa	119.90184	0.8498134	2.863647	6
CH Gayle	118.82764	0.8074671	2.625986	7
RA Jadeja	116.81515	0.6435367	1.906160	8
MS Dhoni	114.98060	0.8006010	2.863647	9
KL Rahul	111.53433	0.7661382	2.863647	10
S Dhawan	111.23947	0.7527486	2.780511	11
V Kohli	111.16608	0.7683365	2.912698	12
RG Sharma	110.73739	0.7407221	2.727384	13
JH Kallis	110.20275	0.5997736	1.991024	14
Yuvraj Singh	108.53280	0.7001425	2.596152	15
G Gambhir	108.14346	0.7322295	2.863647	16
JA Morkel	108.01888	0.5894232	2.037632	17
JP Duminy	106.93410	0.6611917	2.450084	18
SP Narine	106.55846	0.4906443	1.739311	19
SV Samson	106.38129	0.7146079	2.863647	20
DJ Bravo	105.95480	0.5061888	1.807144	21
BB McCullum	105.17320	0.7025269	2.863647	22
KD Karthik	105.16400	0.7024350	2.863647	23
KA Pollard	104.83535	0.5960182	2.210749	24
AC Gilchrist	104.00770	0.6908720	2.863647	25
AT Rayudu	103.35064	0.6843013	2.863647	26
RR Pant	102.86789	0.6794739	2.863647	27
V Sehwag	102.78068	0.6734356	2.821900	28
AM Rahane	102.75678	0.6783628	2.863647	29
IK Pathan	102.51404	0.5776102	2.234486	30
DR Smith	101.47326	0.6103512	2.472913	31
DPMD Jayawardene	101.31355	0.6639305	2.863647	32
GJ Maxwell	100.93858	0.5997075	2.440939	33
WP Saha	100.37245	0.6545195	2.863647	34
MA Agarwal	100.30709	0.6538659	2.863647	35
MK Pandey	99.68638	0.6476588	2.863647	36
SE Marsh	99.61708	0.6469658	2.863647	37
PA Patel	99.40426	0.6448376	2.863647	38
Harbhajan Singh	98.37966	0.3989173	1.709755	39
SPD Smith	98.36306	0.6351153	2.869315	40
SR Tendulkar	98.23047	0.6341753	2.872495	41
MK Tiwary	98.04644	0.6312325	2.863427	42
BJ Hodge	97.90157	0.6258995	2.831929	43
SA Yadav	97.55840	0.6263789	2.863647	44
AR Patel	97.36526	0.4265020	1.827650	45
M Vijay	97.30963	0.6115973	2.766259	46
F du Plessis	95.94784	0.6102734	2.863647	47
SS Iyer	95.72057	0.6080006	2.863647	48
N Rana	95.65067	0.5852893	2.693839	49
STR Binny	94.58025	0.5316566	2.414608	50
AD Russell	94.11716	0.4807739	2.172035	51
R Dravid	93.90454	0.5898403	2.863647	52
A Symonds	93.73354	0.5510777	2.588945	53
AD Mathews	92.17746	0.4848238	2.288587	54
AJ Finch	92.14606	0.5802278	2.930551	55
LRPL Taylor	91.77478	0.5681865	2.860729	56
MEK Hussey	91.59369	0.5667319	2.863647	57

```r
# Top class players based on strike rates & economy rates against each other
top_class_players <- top_25_bat_strikerate %>%
  full_join(top_25_bow_ecorate, by = "player") %>%
  mutate(strike_rate = replace_na(strike_rate,
    batsmen_avgs$median_runs / batsmen_avgs$avg_deliveries
  )) %>%
  mutate(eco_rate = replace_na(eco_rate, bowler_avgs$avg_runs /
                                          bowler_avgs$median_deliveries)) %>%
  mutate(value_of_player = 100 * (strike_rate + 1 / eco_rate))


top_class_players %>%
  arrange(desc(value_of_player)) %>%
  select(player, value_of_player, strike_rate,eco_rate) %>%
  mutate(rank = row_number()) %>%
  head(100) %>%
  knitr::kable()
```

player	value_of_player	strike_rate	eco_rate	rank
SR Watson	140.41070	0.9013531	1.989044	1
SK Raina	126.43855	0.8409679	2.361734	2
YK Pathan	125.17856	0.7927657	2.178555	3
AB de Villiers	124.45383	0.8953332	2.863647	4
DA Warner	122.14993	0.8694098	2.840187	5
RV Uthappa	119.90184	0.8498134	2.863647	6
CH Gayle	118.82764	0.8074671	2.625986	7
RA Jadeja	116.81515	0.6435367	1.906160	8
MS Dhoni	114.98060	0.8006010	2.863647	9
KL Rahul	111.53433	0.7661382	2.863647	10
S Dhawan	111.23947	0.7527486	2.780511	11
V Kohli	111.16608	0.7683365	2.912698	12
RG Sharma	110.73739	0.7407221	2.727384	13
JH Kallis	110.20275	0.5997736	1.991024	14
Yuvraj Singh	108.53280	0.7001425	2.596152	15
G Gambhir	108.14346	0.7322295	2.863647	16
JA Morkel	108.01888	0.5894232	2.037632	17
JP Duminy	106.93410	0.6611917	2.450084	18
SP Narine	106.55846	0.4906443	1.739311	19
SV Samson	106.38129	0.7146079	2.863647	20
DJ Bravo	105.95480	0.5061888	1.807144	21
BB McCullum	105.17320	0.7025269	2.863647	22
KD Karthik	105.16400	0.7024350	2.863647	23
KA Pollard	104.83535	0.5960182	2.210749	24
AC Gilchrist	104.00770	0.6908720	2.863647	25
AT Rayudu	103.35064	0.6843013	2.863647	26
RR Pant	102.86789	0.6794739	2.863647	27
V Sehwag	102.78068	0.6734356	2.821900	28
AM Rahane	102.75678	0.6783628	2.863647	29
IK Pathan	102.51404	0.5776102	2.234486	30
DR Smith	101.47326	0.6103512	2.472913	31
DPMD Jayawardene	101.31355	0.6639305	2.863647	32
GJ Maxwell	100.93858	0.5997075	2.440939	33
WP Saha	100.37245	0.6545195	2.863647	34
MA Agarwal	100.30709	0.6538659	2.863647	35
MK Pandey	99.68638	0.6476588	2.863647	36
SE Marsh	99.61708	0.6469658	2.863647	37
PA Patel	99.40426	0.6448376	2.863647	38
Harbhajan Singh	98.37966	0.3989173	1.709755	39
SPD Smith	98.36306	0.6351153	2.869315	40
SR Tendulkar	98.23047	0.6341753	2.872495	41
MK Tiwary	98.04644	0.6312325	2.863427	42
BJ Hodge	97.90157	0.6258995	2.831929	43
SA Yadav	97.55840	0.6263789	2.863647	44
AR Patel	97.36526	0.4265020	1.827650	45
M Vijay	97.30963	0.6115973	2.766259	46
F du Plessis	95.94784	0.6102734	2.863647	47
SS Iyer	95.72057	0.6080006	2.863647	48
N Rana	95.65067	0.5852893	2.693839	49
STR Binny	94.58025	0.5316566	2.414608	50
AD Russell	94.11716	0.4807739	2.172035	51
R Dravid	93.90454	0.5898403	2.863647	52
A Symonds	93.73354	0.5510777	2.588945	53
AD Mathews	92.17746	0.4848238	2.288587	54
AJ Finch	92.14606	0.5802278	2.930551	55
LRPL Taylor	91.77478	0.5681865	2.860729	56
MEK Hussey	91.59369	0.5667319	2.863647	57

We will now create the pool of 200 top players based on the above two models. We are taking into account players ranked by strikerate and economy and also how they performed against each other. There maybe other anomolies we can introduce to this to arrive at the best pool of players but we will restrict our findings to these two.

```
# Top pool players based on strike rate and economy for bowlers and their performance with each other
# we will again join the two sets and replace the strike rate and economy rate as previously done
world_top_players <- top_players %>%
  select(-value_of_player) %>%
  full_join(top_class_players, by = "player") %>%
  mutate(strike_rate = replace_na(strike_rate,
                                  batsmen_avgs$median_runs / batsmen_avgs$avg_deliveries
  )) %>%
  mutate(eco_rate = replace_na(eco_rate, bowler_avgs$avg_runs /
                                 bowler_avgs$median_deliveries)) %>%
  mutate(player_value = 100 * ((batsman_reg_str_rate + strike_rate) + 1 /
                                 (bowler_reg_eco_rate + eco_rate)))  %>%
  select(player, value_of_player) %>%
  arrange(desc(value_of_player)) %>%
  mutate(rank = row_number())
world_top_players %>%
  head(200)
```

```
## # A tibble: 200 x 3
##    player         value_of_player  rank
##    <chr>                    <dbl> <int>
##  1 SR Watson                 140.     1
##  2 SK Raina                  126.     2
##  3 YK Pathan                 125.     3
##  4 AB de Villiers            124.     4
##  5 DA Warner                 122.     5
##  6 RV Uthappa                120.     6
##  7 CH Gayle                  119.     7
##  8 RA Jadeja                 117.     8
##  9 MS Dhoni                  115.     9
## 10 KL Rahul                  112.    10
## # ... with 190 more rows
```

Model 2: Predict the winner of a Match

From our analysis of the matches we have seen the following factors matter

```
1)  venue, toss, toss decision, team battiing first and the team batting next
2)  the data set we have is extremely small
```

To predict a match winner it is known that the (KNN and Random Forest), with 49.77% accuracy (logistic Regression), almost 17% better than a random decision (benchmark) which has 33%. We will also look at the Naive Bayes methodology.

Naive Bayes – https://en.wikipedia.org/wiki/Naive_Bayes_classifier Logistic Regression – https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148#:~:text=Logistic%20regression%20is%20a%20clas
KNN Algorithm (Nearest Neighbour) – https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
Random Forest – https://en.wikipedia.org/wiki/Random_forest

Data Sets: train and test datasets

Create the training and test sets

```
# Create a new dataset. We are going to be taking matches as our base data set to work out of
# Let us remove  Gujarat Titans and Kochi Tuskers Kerala because they  no more take part in the IPL
best_teams <- teams %>% arrange(desc(matches_played)) %>%
  head(8)
best_teams
```

```
## # A tibble: 8 x 7
##   team    count1 count2 number_of_tosse~ matches_won matches_lost matches_played
##   <chr>    <int>  <int>            <int>       <int>        <int>          <int>
## 1 Mumbai~     97    106              106         120           83            203
## 2 Delhi ~    102     92              100          86          106            192
## 3 Kolkat~     95     97               98          99           93            192
## 4 Royal ~    108     87               87          91          101            192
## 5 Punjab~     92     98               85          88          102            190
## 6 Chenna~     94     84               97         106           72            178
## 7 Rajast~     70     91               87          81           78            159
## 8 Sunris~     59     65               57          66           58            124
```

```
match_predictor_ds <- matches %>%
  select(team1,team2, winner, venue, toss_winner, toss_decision) %>%
  filter(winner != ""  & team1 %in% best_teams$team & team2 %in% best_teams$team)
any(is.na(match_predictor_ds))
```

```
## [1] FALSE
```

```
summary(match_predictor_ds)
```

```
##     team1              team2              winner             venue
##  Length:628         Length:628         Length:628         Length:628
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  toss_winner        toss_decision
##  Length:628         Length:628
##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

```
levels(as.factor(match_predictor_ds$team1))
```

```
## [1] "Chennai Super Kings"         "Delhi Capitals"
## [3] "Kolkata Knight Riders"       "Mumbai Indians"
## [5] "Punjab Kings"                "Rajasthan Royals"
## [7] "Royal Challengers Bangalore" "Sunrisers Hyderabad"
```

```
match_predictor_ds$team1[match_predictor_ds$team1 == "Mumbai Indians"] <- "Mumbai"
match_predictor_ds$team1[match_predictor_ds$team1 == "Delhi Capitals"] <- "DC"
match_predictor_ds$team1[match_predictor_ds$team1 == "Kolkata Knight Riders"] <- "KKR"
```

```
match_predictor_ds$team1[match_predictor_ds$team1 == "Royal Challengers Bangalore"] <- "RCB"
match_predictor_ds$team1[match_predictor_ds$team1 == "Punjab Kings"] <- "Punjab"
match_predictor_ds$team1[match_predictor_ds$team1 == "Chennai Super Kings"] <- "CSK"
match_predictor_ds$team1[match_predictor_ds$team1 == "Sunrisers Hyderabad"] <- "Hyd"
match_predictor_ds$team1[match_predictor_ds$team1 == "Rajasthan Royals"] <- "Rajasthan"


levels(as.factor(match_predictor_ds$team2))
```

```
## [1] "Chennai Super Kings"         "Delhi Capitals"
## [3] "Kolkata Knight Riders"       "Mumbai Indians"
## [5] "Punjab Kings"                "Rajasthan Royals"
## [7] "Royal Challengers Bangalore" "Sunrisers Hyderabad"
```

```
match_predictor_ds$team2[match_predictor_ds$team2 == "Mumbai Indians"] <- "Mumbai"
match_predictor_ds$team2[match_predictor_ds$team2 == "Delhi Capitals"] <- "DC"
match_predictor_ds$team2[match_predictor_ds$team2 == "Kolkata Knight Riders"] <- "KKR"
match_predictor_ds$team2[match_predictor_ds$team2 == "Royal Challengers Bangalore"] <- "RCB"
match_predictor_ds$team2[match_predictor_ds$team2 == "Punjab Kings"] <- "Punjab"
match_predictor_ds$team2[match_predictor_ds$team2 == "Chennai Super Kings"] <- "CSK"
match_predictor_ds$team2[match_predictor_ds$team2 == "Sunrisers Hyderabad"] <- "Hyd"
match_predictor_ds$team2[match_predictor_ds$team2 == "Rajasthan Royals"] <- "Rajasthan"

levels(as.factor(match_predictor_ds$winner))
```

```
## [1] "Chennai Super Kings"         "Delhi Capitals"
## [3] "Kolkata Knight Riders"       "Mumbai Indians"
## [5] "Punjab Kings"                "Rajasthan Royals"
## [7] "Royal Challengers Bangalore" "Sunrisers Hyderabad"
```

```
match_predictor_ds$winner[match_predictor_ds$winner == "Mumbai Indians"] <- "Mumbai"
match_predictor_ds$winner[match_predictor_ds$winner == "Delhi Capitals"] <- "DC"
match_predictor_ds$winner[match_predictor_ds$winner == "Kolkata Knight Riders"] <- "KKR"
match_predictor_ds$winner[match_predictor_ds$winner == "Royal Challengers Bangalore"] <- "RCB"
match_predictor_ds$winner[match_predictor_ds$winner == "Punjab Kings"] <- "Punjab"
match_predictor_ds$winner[match_predictor_ds$winner == "Chennai Super Kings"] <- "CSK"
match_predictor_ds$winner[match_predictor_ds$winner == "Sunrisers Hyderabad"] <- "Hyd"
match_predictor_ds$winner[match_predictor_ds$winner == "Rajasthan Royals"] <- "Rajasthan"


levels(as.factor(match_predictor_ds$toss_winner))
```

```
## [1] "Chennai Super Kings"         "Delhi Capitals"
## [3] "Kolkata Knight Riders"       "Mumbai Indians"
## [5] "Punjab Kings"                "Rajasthan Royals"
## [7] "Royal Challengers Bangalore" "Sunrisers Hyderabad"
```

```
match_predictor_ds$toss_winner[match_predictor_ds$toss_winner == "Mumbai Indians"] <- "Mumbai"
match_predictor_ds$toss_winner[match_predictor_ds$toss_winner == "Delhi Capitals"] <- "DC"
match_predictor_ds$toss_winner[match_predictor_ds$toss_winner == "Kolkata Knight Riders"] <- "KKR"
match_predictor_ds$toss_winner[match_predictor_ds$toss_winner == "Royal Challengers Bangalore"] <- "RCB
```

```
match_predictor_ds$toss_winner[match_predictor_ds$toss_winner == "Punjab Kings"] <- "Punjab"
match_predictor_ds$toss_winner[match_predictor_ds$toss_winner == "Chennai Super Kings"] <- "CSK"
match_predictor_ds$toss_winner[match_predictor_ds$toss_winner == "Sunrisers Hyderabad"] <- "Hyd"
match_predictor_ds$toss_winner[match_predictor_ds$toss_winner == "Rajasthan Royals"] <- "Rajasthan"


# Create a Train Set and a Test Set

set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
options(digits=3)

test_index <- createDataPartition(y = match_predictor_ds$winner, times = 1, p = 0.1, list = FALSE)

# create a test & train set
train <- match_predictor_ds[-test_index,]
temp <- match_predictor_ds[test_index,]

# Matching team1,team2,venue,toss_decision and toss_winner in both train and test sets
test <- temp %>%
  semi_join(train, by = "team1") %>%
  semi_join(train, by = "team2") %>%
  semi_join(train, by = "venue") %>%
  semi_join(train, by = "toss_decision") %>%
  semi_join(train, by = "toss_winner")
# Add rows removed from temp set back into train set
removed <- anti_join(temp, test)
train <- rbind(train, removed)
rm(removed, temp, test_index)

#Let us explore the train data set

train %>% as_tibble()
```

```
## # A tibble: 562 x 6
##    team1    team2    winner  venue                     toss_winner toss_decision
##    <chr>    <chr>    <chr>   <chr>                     <chr>       <chr>
##  1 RCB      KKR      KKR     M. A. Chidambaram Stadium RCB         field
##  2 Punjab   CSK      CSK     Punjab Cricket Associati~ CSK         bat
##  3 DC       Rajasth~ DC      Feroz Shah Kotla          Rajasthan   bat
##  4 Mumbai   RCB      RCB     Wankhede Stadium          Mumbai      bat
##  5 Rajasth~ Punjab   Rajast~ Sawai Mansingh Stadium    Punjab      bat
##  6 CSK      Mumbai   CSK     M. A. Chidambaram Stadium Mumbai      field
##  7 Punjab   Mumbai   Punjab  Punjab Cricket Associati~ Mumbai      field
##  8 RCB      Rajasth~ Rajast~ M. A. Chidambaram Stadium Rajasthan   field
##  9 CSK      KKR      CSK     M. A. Chidambaram Stadium KKR         bat
## 10 RCB      CSK      CSK     M. A. Chidambaram Stadium CSK         bat
## # ... with 552 more rows
```

```
test %>% as_tibble()
```

```
## # A tibble: 66 x 6
##    team1    team2    winner  venue                     toss_winner toss_decision
##    <chr>    <chr>    <chr>   <chr>                     <chr>       <chr>
```

```
##  1 Punjab   DC       Punjab  Punjab Cricket Associati~ DC          bat
##  2 KKR      RCB      KKR     Eden Gardens              KKR         bat
##  3 CSK      Punjab   CSK     M. A. Chidambaram Stadium Punjab      field
##  4 Rajasth~ DC       Rajast~ Sawai Mansingh Stadium    Rajasthan   field
##  5 KKR      Rajasth~ Rajast~ Eden Gardens              Rajasthan   field
##  6 Punjab   Mumbai   Mumbai  SuperSport Park           Punjab      bat
##  7 DC       Mumbai   DC      SuperSport Park           DC          field
##  8 Rajasth~ DC       DC      Sardar Patel Stadium, Mo~ DC          field
##  9 Rajasth~ KKR      Rajast~ Sardar Patel Stadium, Mo~ Rajasthan   bat
## 10 CSK      Rajasth~ CSK     M. A. Chidambaram Stadium CSK         bat
## # ... with 56 more rows
```

Algorithms and Fit

F score : In statistical analysis of binary classification, the F-score or F-measure is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. Precision is also known as positive predictive value, and recall is also known as sensitivity in diagnostic binary classification.

We are not worried abut negative outcomes and hence going with an F score. F1 score: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6 , we will use F1 score. Naive Bayes

```
# Since all the variables in the  dataset are variables that deermine he winner we will calulcae the F1
# for all classes
naive <- train(winner ~ ., method = "naive_bayes", data = train)
predicion_test_naive <- predict(naive, test)
f1_score_nb <- confusionMatrix(predicion_test_naive, as.factor(test$winner))$byClass[,"F1"]
f1_score_nb <- as.data.frame(t(f1_score_nb)) %>% mutate(averageF1Score = rowMeans(.))
# Table
# Make column names more readable
colnames(f1_score_nb) = gsub("Class: ", "", colnames(f1_score_nb))
# F1 table for different models
f1_model_table <- data.frame(Model = "Naive") %>% bind_cols(f1_score_nb)
f1_model_table %>% knitr::kable()
```

Model	CSK	DC	Hyd	KKR	Mumbai	Punjab	Rajasthan	RCB	averageF1Score
Naive	NA	NA	0.5	0.571	NA	0.292	0.667	NA	NA

We can see from the observation that the prediction oes nt satisfy all teams and hence the Naive Bayes algorithm cannot be used in this case

Logistic Regression

```
# Logistic Regression

logistic_regression <- train(winner ~ ., method = "lda", data = train)
predicion_test_lr <- predict(logistic_regression, test)
f1_score_lr <- confusionMatrix(predicion_test_lr, as.factor(test$winner))$byClass[,"F1"]
f1_score_lr <- as.data.frame(t(f1_score_lr)) %>% mutate(averageF1Score = rowMeans(.))
f1_score_lr
```

```
##   Class: CSK Class: DC Class: Hyd Class: KKR Class: Mumbai Class: Punjab
## 1      0.667     0.154      0.364        0.5         0.538         0.462
##   Class: Rajasthan Class: RCB averageF1Score
## 1            0.526        0.4          0.451
```

```
# Table
colnames(f1_score_lr) = gsub("Class: ", "", colnames(f1_score_lr))

f1_model_table <- bind_rows(f1_model_table,
                      data.frame(Model = "LDA") %>% bind_cols(f1_score_lr))
f1_model_table %>% knitr::kable()
```

Model	CSK	DC	Hyd	KKR	Mumbai	Punjab	Rajasthan	RCB	averageF1Score
Naive	NA	NA	0.500	0.571	NA	0.292	0.667	NA	NA
LDA	0.667	0.154	0.364	0.500	0.538	0.462	0.526	0.4	0.451

Random Forest

```
# Random Forest (multiple trees)

trainctrl <- trainControl(method="cv")
random_forest <- train(winner ~ ., method = "rf", data = train,  trControl=trainctrl)
predicion_test_rf <- predict(random_forest, test)
f1_score_rf <- confusionMatrix(predicion_test_rf,  as.factor(test$winner))$byClass[,"F1"]
f1_score_rf <- as.data.frame(t(f1_score_rf)) %>% mutate(averageF1Score = rowMeans(.))
f1_score_rf
```

```
##   Class: CSK Class: DC Class: Hyd Class: KKR Class: Mumbai Class: Punjab
## 1      0.533     0.444        0.4      0.632         0.593         0.429
##   Class: Rajasthan Class: RCB averageF1Score
## 1            0.667      0.571          0.534
```

```
colnames(f1_score_rf) = gsub("Class: ", "", colnames(f1_score_rf))
f1_model_table <- bind_rows(f1_model_table,
                            data.frame(Model = "RF") %>% bind_cols(f1_score_rf))
f1_model_table %>% knitr::kable()
```

Model	CSK	DC	Hyd	KKR	Mumbai	Punjab	Rajasthan	RCB	averageF1Score
Naive	NA	NA	0.500	0.571	NA	0.292	0.667	NA	NA
LDA	0.667	0.154	0.364	0.500	0.538	0.462	0.526	0.400	0.451
RF	0.533	0.444	0.400	0.632	0.593	0.429	0.667	0.571	0.534

Nearest Neighbour

```
# KNN (Nearest neighbour)

knn <- train(winner ~ ., method = "knn", data = train,)
predicion_test_knn <- predict(knn, test)
f1_score_knn <- confusionMatrix(predicion_test_knn, as.factor(test$winner))$byClass[,"F1"]
f1_score_knn <- as.data.frame(t(f1_score_knn)) %>% mutate(averageF1Score = rowMeans(.))
f1_score_knn
```

```
##   Class: CSK Class: DC Class: Hyd Class: KKR Class: Mumbai Class: Punjab
## 1        0.5       0.5      0.667      0.556          0.64          0.75
##   Class: Rajasthan Class: RCB averageF1Score
## 1            0.667      0.714          0.624
```

```
# Table
colnames(f1_score_knn) = gsub("Class: ", "", colnames(f1_score_knn))
f1_model_table <- bind_rows(f1_model_table,
                      data.frame(Model = "KNN") %>% bind_cols(f1_score_knn))
f1_model_table %>% knitr::kable()
```

Model	CSK	DC	Hyd	KKR	Mumbai	Punjab	Rajasthan	RCB	averageF1Score
Naive	NA	NA	0.500	0.571	NA	0.292	0.667	NA	NA
LDA	0.667	0.154	0.364	0.500	0.538	0.462	0.526	0.400	0.451
RF	0.533	0.444	0.400	0.632	0.593	0.429	0.667	0.571	0.534
KNN	0.500	0.500	0.667	0.556	0.640	0.750	0.667	0.714	0.624

Conclusion:

We have created a pool of players considering different factors from which IPL teams can pick from. There are always biases that can be added to the analysis Eg. What if a top player moved teams , that may change the winner of a game. We also see from the analysis that LDA gives us the best results in terms of predicting a winner. Random forest as it considers multiple trees is a little slow compared to the other algorithms. KNN and Random forest results are nearly similar.

I also want to take this oppertunity to thanks all my peers , staff and the excellent teachers at the program.