Indian Premier League (IPL) T20 cricket data visualization and analysis

STAT585 Final Project Report

Ashish Jain

27 April 2017

Introduction

Cricket is one of the most popular and played sports in India. There are three standard types of cricket matches, Five-day match (Test Match), 50 overs match (One day match), and Twenty overs match (T20). Although all the formats are quite popular but during the last decade, the popularity of the T20 matches has increased exponentially. The main reason for the popularity of T20 matches is more action in short period of time. A T20 match takes only 3 hours to complete which is pretty short as compared to Test and One day match. In addition to that, the introduction of various T20 tournaments has also added to its popularity. Indian Premier League (IPL) is the biggest tournament among all the T20 tournaments across the globe. It is an annual tournament comprising of two stages. In the first or the league stage, each team plays against the other team twice. In the final stage, the top four teams after the league stage compete for the title. We used the IPL data from the year 2008 to 2016 for our analysis. The raw data is available at Cricsheet website. The raw data consists of the ball by ball details individually for each match (577 files). The raw data has been processed The processed data has been downloaded from Kaggle. For carrying out the analysis we divided this project into two parts. In the first part, we processed the data to get the location coordinates of all the venues which are used in the next part for visualization. In the next part, we developed a shiny application in which the data is grouped by each year, team, and player for visualization and analysis.

The GitHub repository for this project is https://github.com/ashishjain1988/STAT585-FinalProject

Dataset

As mentioned earlier, the IPL dataset has been available at Cricsheet website. The raw data consists ball by ball details (including bowler, batsmen, run scored, match, and teams.) individually for every match (577 files). These files also consist the meta information about the match including the venue, toss winner, the match winner, stadium, and city. Our aim was first to process the data and make it tidy but this step has already been done. We downloaded and used the processed data which is available at Kaggle. The processed data consists of two files. The first file, `matches.csv` consists of meta-information about all the matches as described earlier as a comma separated file. The second file, `deliveries.csv` consists of the ball by ball information of the matches. These files have a common identifier called as "match_id" which is the linking key for these two files.

```r
library(ggplot2)
library(dplyr)
library(tidyr)
# library(ggmap) library(rworldmap) library(ggrepel)
github <- "https://raw.githubusercontent.com/ashishjain1988/STAT585-FinalProject/master/data/"
matches <- read.csv(paste0(github, "matches.csv"), stringsAsFactors = FALSE)
ballbyball <- read.csv(paste0(github, "deliveries1.csv"), stringsAsFactors = FALSE)
location <- read.csv(paste0(github, "location_mapping.csv"),
    stringsAsFactors = FALSE, row.names = 1, header = T)
location[, "city"] <- row.names(location)
```

Methods

Processing of the dataset

Although the data is in a very tidy format, it still required some additional processing steps. In our analysis, we want to point each venue of the matches on a world map with the number of matches for a particular team. For this visualization, we need to extract the longitude and latitude for the venues. For this, we used the `ggmap` and `rworldmap` to extract the details. In total, the matches were hosted by 30 different venues across all seasons. By using `ggmap` we extracted the location information and stored in the `location_mapping.csv` the file which is in the data folder.

```r
cities<-unlist(matches %>% distinct(city) %>% filter(city != ""))
points<-geocode(cities)
row.names(points)<-cities
points<-points[,2:3]
coord2loc <- function(points)
{
  countriesSP <- getMap(resolution='low')
  pointsSP = SpatialPoints(points, proj4string=CRS(proj4string(countriesSP)))
  indices = over(pointsSP, countriesSP)
  #indices$continent   # returns the continent (6 continent model)
  #indices$REGION   # returns the continent (7 continent model)
  indices$ADMIN  #returns country name
  #indices$ISO3 # returns the ISO3 code
}
countries<-coord2loc(as.matrix(points))
points[,3]<-countries
points["Kochi",]$V3<-"India"
write.csv(points,"../data/location_mapping.csv")
#map <- get_map(location = 'India', zoom = 4)
#ggmap(map) + geom_point(aes(x=77.10249,y=28.70406,label="Delhi"))
#mapWorld <- borders("world", colour="gray50", fill="gray50")
#mp <- ggplot() +   mapWorld
#mp <- mp+ geom_point(aes(x=visit.x, y=visit.y) ,color="blue", size=3)
```

Results

In this section, we will discuss various features of the interactive shiny application which is developed to analyze the IPL dataset. The shiny application has three different tabs which are based on the visualization of data based on IPL season, team, and player.

IPL seasons analysis

In this section, the user can visualize and analyze the data grouped together by the season. The user can use various plots which are described below for a specific season by selecting the season in the select widget. In addition to that, the user can also filter the teams or players by selecting the minimum number of seasons played through the slider bar. Now we talk about the different plots that are there in the shiny application. The shiny page for the season analysis is shown below.

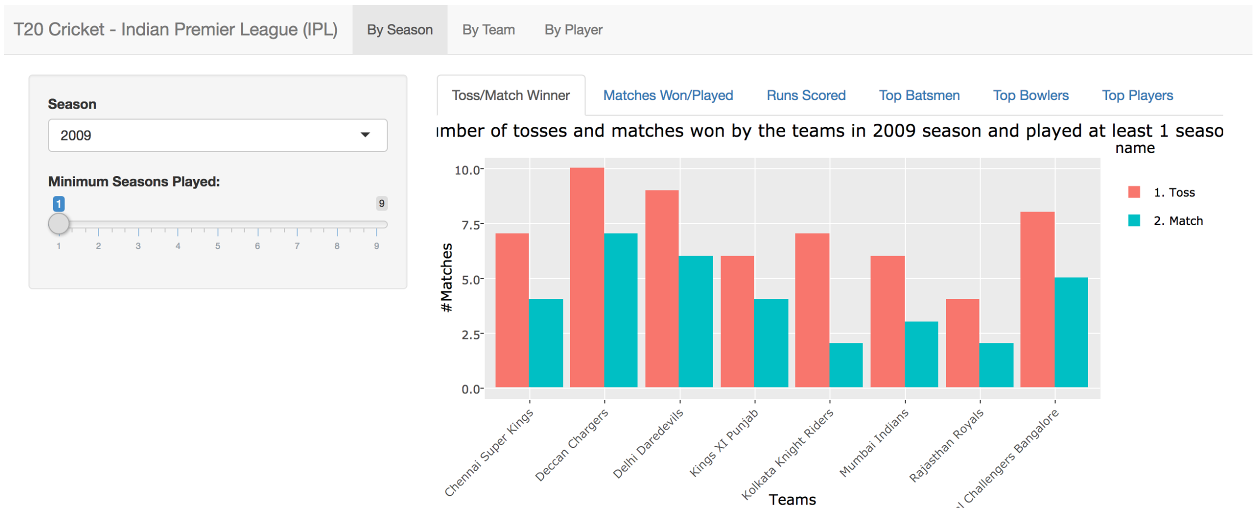
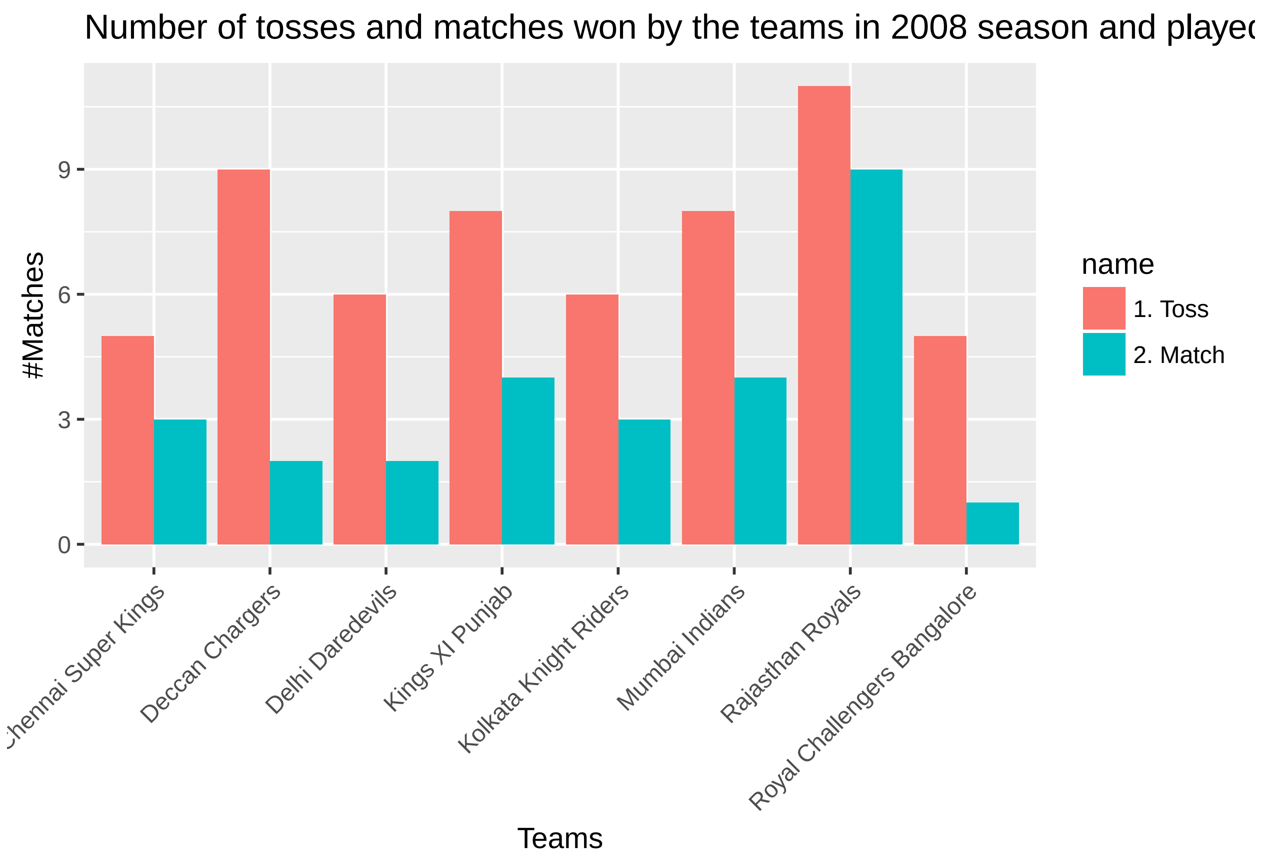


Figure 1: Season analysis

Toss Winner

This plot compares the count of toss and matches winner across different teams in a season. The information is displayed as a histogram. This plot can be used to see the effect of winning the toss on the match result for teams. The user can also hover over the bars and get the details. For example, the plot below compares how many times a team won a match after winning the toss in 2008 season.

```r
Toss <- matches %>% filter(season == 2008) %>% group_by(toss_winner) %>%
    count() %>% mutate(winner = toss_winner) %>% select(-toss_winner)
Match <- matches %>% filter(season == 2008, toss_winner == winner) %>%
    group_by(winner) %>% filter(winner != "") %>% count()
Toss$name <- "1. Toss"
Match$name <- "2. Match"
team_seasons <- matches %>% gather(key = team, value = teamname,
    c(5:6)) %>% distinct(season, teamname) %>% group_by(teamname) %>%
    count() %>% filter(n >= 1)
stat_data_toss <- rbind(Toss, Match) %>% filter(winner %in% team_seasons$teamname)
gg <- ggplot(stat_data_toss, aes(winner, n, fill = name)) + geom_histogram(position = "dodge",
    stat = "identity") + ggtitle(paste("Number of tosses and matches won by the teams in",
    2008, "season and played at least", 1, "seasons")) + theme(axis.text.x = element_text(angle = 45,
    hjust = 1)) + xlab("Teams") + ylab("#Matches")
gg
```



Comparison of total matches played and won

This plot compares the total matches played and total match won across different teams in a season. The information is displayed as a histogram. This plot can be used to check the performance of the teams in a season. This will also tell us the winner of a particular season. The user can also hover over the bars and get the details.

Comparison of average runs scored by teams

This plot compares the average runs scored across different teams in a season. The information is displayed as a boxplot. This plot can be used to check the performance of the teams in a season by displaying their batting performance of the teams in a season. The user can also hover over the bars and get the details about the mean, outlier scores, upper and lower quartile.

Top batsmen

This plot compares the total runs scored by batsmen in a season. The information is displayed as a histogram. This plot shows the top 10 batsmen in a season based on the number of runs scored. The number of runs scored by the batsmen is displayed above the plot.

Top bowlers

This plot compares the total runs scored by batsmen in a season. The information is displayed as a histogram. This plot shows the top 10 batsmen in a season based on the number of runs scored. The number of runs scored by the batsmen is displayed above the plot.

Team performance analysis

In this section, the user can visualize and analyze the data grouped together for a team across all seasons. The user can use various plots which are described below for a team by selecting the team in the select widget. In addition to that, the user can also filter the teams by selecting the minimum number of seasons played through the slider bar (Automatic update of team select widget). Now we talk about the different plots that are there for this. The shiny page for the team analysis is shown below.

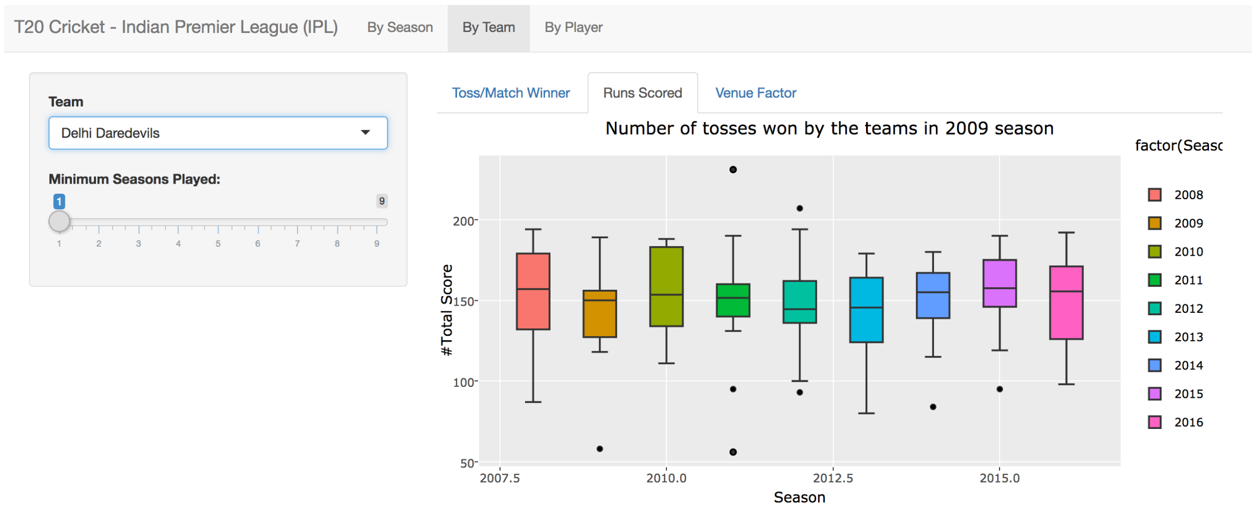


Figure 2: Team analysis

Comparison of total matches played and won

This plot compares the total matches played and total match won by a team across the seasons. The information is displayed as a histogram. This plot can be used to check the performance of a team across different seasons. The user can also hover over the bars and get the details.

Comparison of average runs scored

This plot compares the average runs scored by a team across the seasons. The information is displayed as a boxplot. This plot can be used to check the performance of a team by checking their batting performance across the seasons. The user can also hover over the bars and get the details about the mean, outlier scores, upper and lower quartile.

Location Impact

This plot contains the information about the matches won by a team at a particular venue. The information is displayed by using `ggmap`. The plot maps the location of the venue by plotting the longitude and latitude on a world map and labeling the number of matches played and won at that location. For example the plot below show the location where Gujrat Lions played. The labels consists of the venue name, number of matches played and won.

```
teamName<-"Gujarat Lions"
matchPlayed <- matches %>% gather(key=team,value=teamname,c(5:6)) %>% filter(teamname == teamName)
%>% select(teamname,city) %>% group_by(city) %>% count()
matchWon <- matches %>% filter(winner == teamName) %>% select(winner,city) %>%group_by(city)
%>% count()
names(matchPlayed)<-c("city","Matches.Played")
df<-data.frame(setdiff(matchPlayed$city,matchWon$city),0)
names(df)<-c("city","n")
dd<-rbind(matchWon,df)
names(dd)<-c("city","Matches.Won")
matchl<-merge(matchPlayed,dd,by.x="city",by.y="city")
matchlocation<-merge(matchl,location,by.x="city",by.y="city")
map <- get_map(location = unique(matchlocation$V3), zoom = 4)
ggmap(map) + geom_point(data = matchlocation,aes(x=lon,y=lat)) +
  geom_label_repel(data = matchlocation,aes(lon, lat, label = paste(city,Matches.Played,Matches.Won)),
  fontface = 'bold', color = 'red',box.padding = unit(0.35, "lines"),point.padding = unit(0.5, "lines")
  theme_classic(base_size = 10)
```

Player's performance analysis

In this section, the user can visualize and analyze the data grouped for a player across all seasons. The user can use various plots which are described below for a player to analyze his performance by selecting the player in the select widget. In addition to that, the user can also filter the players by selecting the minimum number of seasons played through the slider bar (Automatic update of the player select widget). The shiny page for the player analysis is shown below.

Average runs scored across seasons

This plot compares the average runs scored by a batsman across seasons. The information is displayed as a boxplot. This plot can be used to check the performance of a batsman by checking his batting performance across the seasons. The user can also hover over the bars and get the details about the mean, highest score, upper and lower quartile. For example, the plot below shows the average runs of Virat Kohli across all seasons.

```
player<-"V Kohli"
runs_by_season<-ballbyball %>% filter(player == batsman) %>% group_by(Season,match_id) %>% summarise(to
ggplot(runs_by_season, aes(x=Season, y = totalscore,fill = factor(Season))) + geom_boxplot() +
      ggtitle(paste("Total Runs for", player, "across all seasons")) +
        xlab("Season") + ylab("#Runs")
```

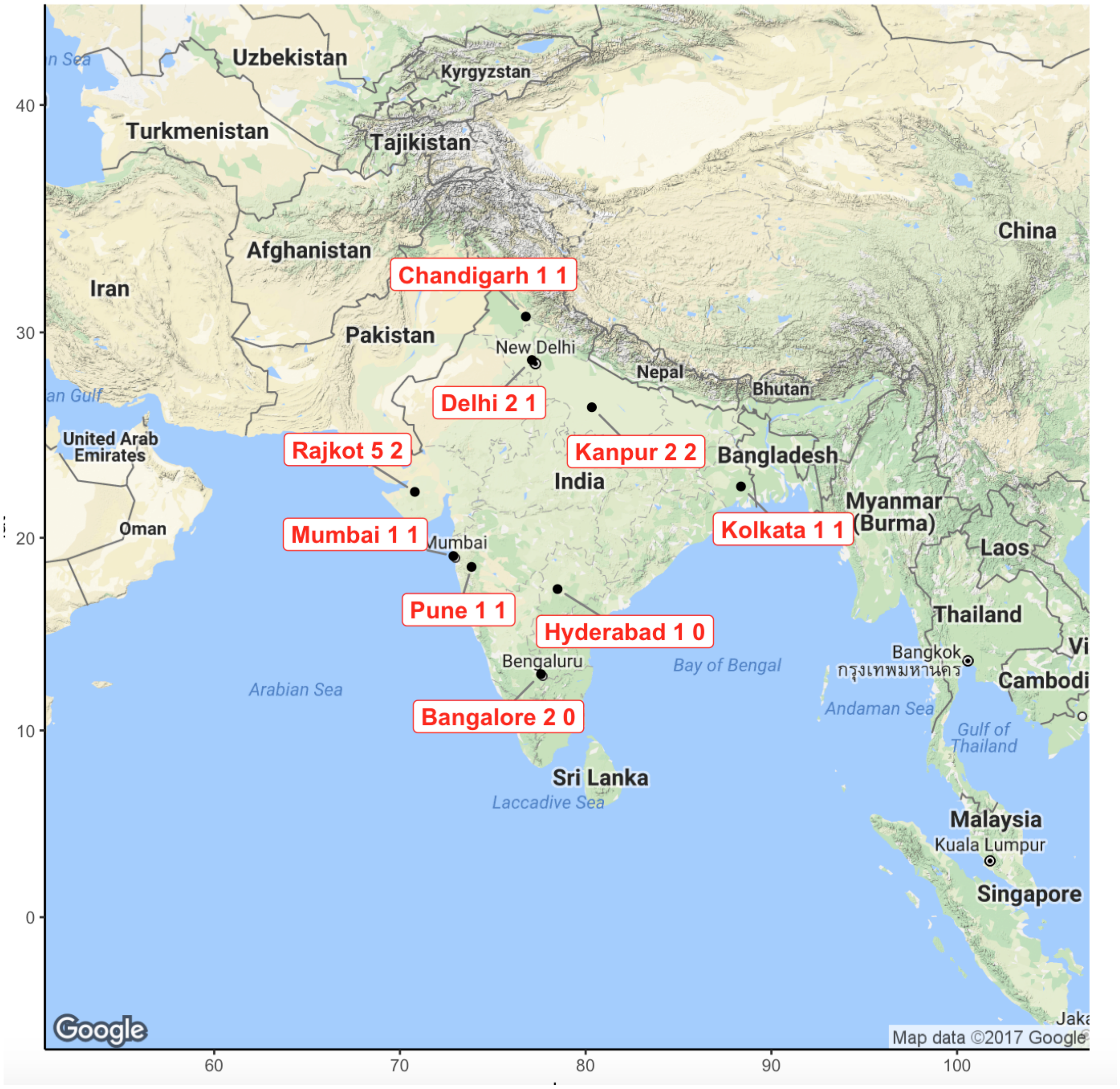
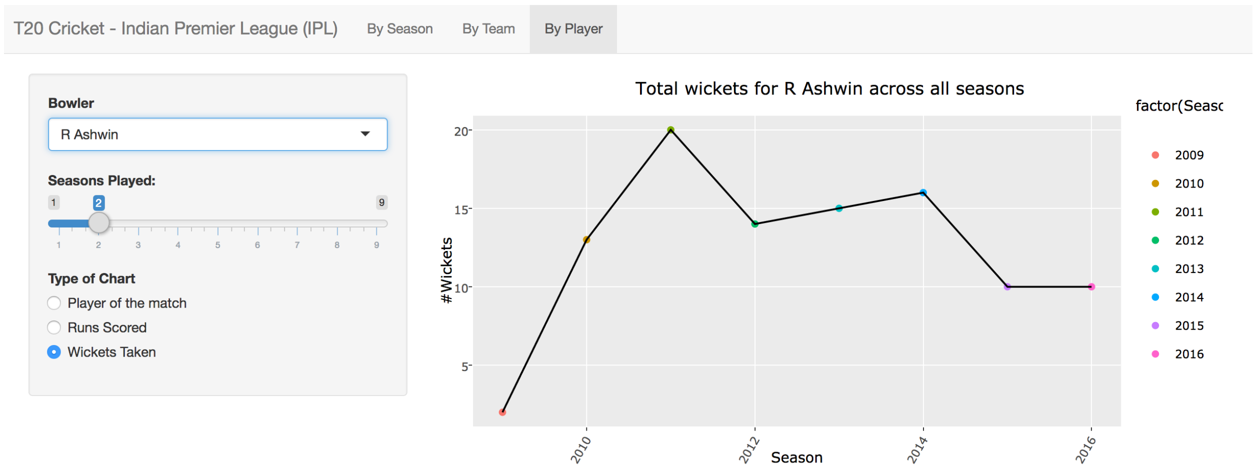


Figure 3: Locations of Gujarat Lions


T20 Cricket - Indian Premier League (IPL)
By Season
By Team
By Player

Bowler

R Ashwin

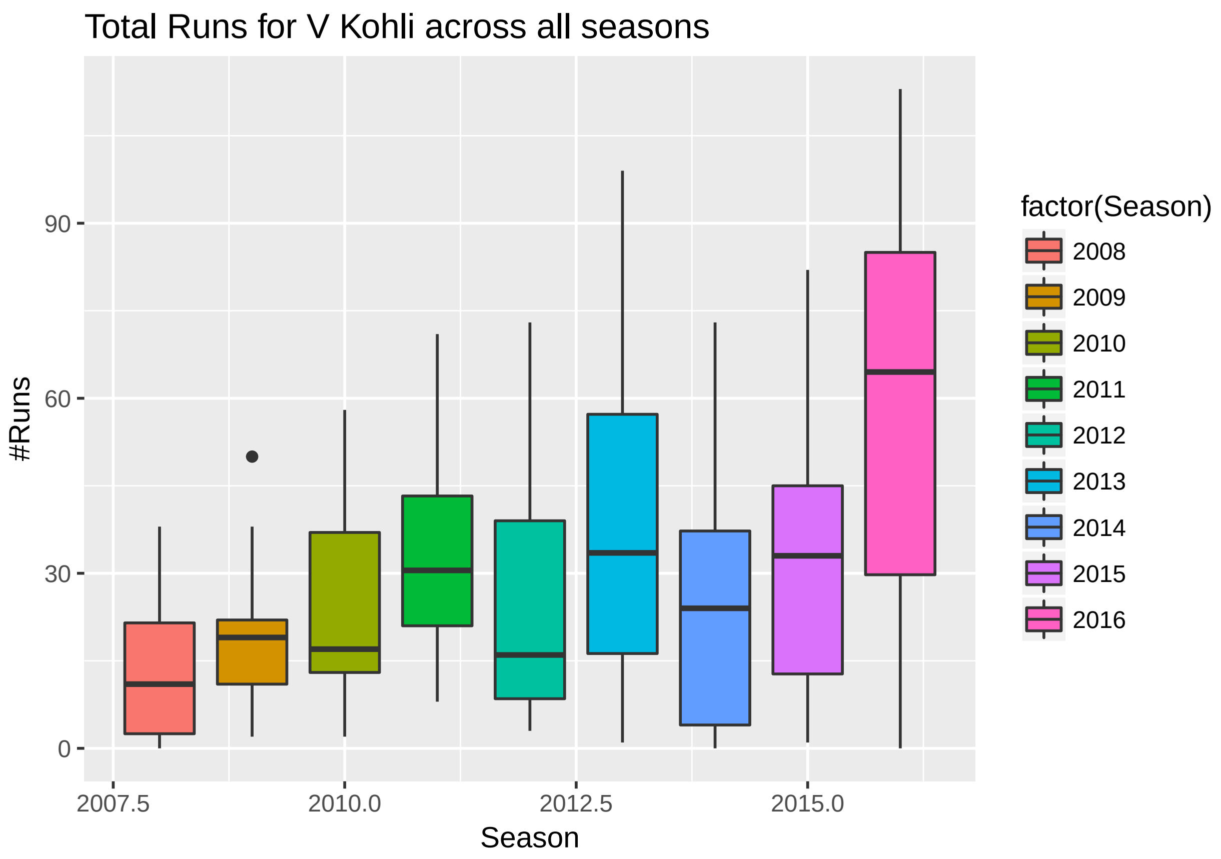
Seasons Played: 2

Type of Chart

- Player of the match
- Runs Scored
- [✓ Wickets Taken]

Total wickets for R Ashwin across all seasons
#Wickets
Season
factor(Seaso
2009
2010
2011
2012
2013
2014
2015
2016

Figure 4: Player analysis



Player of match awards

This plot compares the player of the match award by a player across the seasons. The information is displayed as a histogram. This plot can be used to check the impact of the player in a particular season.

Total Wickets

This plot compares the number of wickets taken by a bowler across seasons. This plot can be used to check the performance of the bowler by checking his bowling performance across the seasons. The user can also hover over the points and get the information about the number of wickets in a particular season.

Future Work

We carried out a very basic analysis of the IPL dataset. We can investigate this dataset by comparing different seasons and teams in a single plot in order to know more about their performance. The analysis of this dataset can be used by the team owners to select players in future for their teams.