INDIAN
PREMIER
LEAGUE

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

DATASET DESCRIPTION

Column Name	Type	Description
id	int	Unique id alloted to each match
city	char	City where match was played
date	char	Date on which match was played
player_of_match	char	Man of the Match
venue	char	Stadium where match was played
team1	char	First of the two teams competing
team2	char	Second of the two teams competing
toss_winner	char	Team which was the toss
toss_decision	char	Batting or bowling decision made by the toss winning team
winner	char	Team that won the match
result	char	Winning Factor which is either by runs or wickets
result_margin	int	Won By how many runs or wikctes
eliminator	int	Y= if the type of match is eliminator otherwise N

This Dataset Contains Data of IPL Matches starting from 2008 to 2020.

ABOUT THE IPL:

The IPL is the most attended cricket league in the world and rank sixth among all sports leagues. In 2010 the IPL became the first sporting event in the world to be broadcasted live on YOUTUBE . The brand value of IPL was estimated to be US $3.2 billion in 2014. According to BCCI, the 2015 IPL Season contributed $11.5 million to the GDP of the Indian economy.

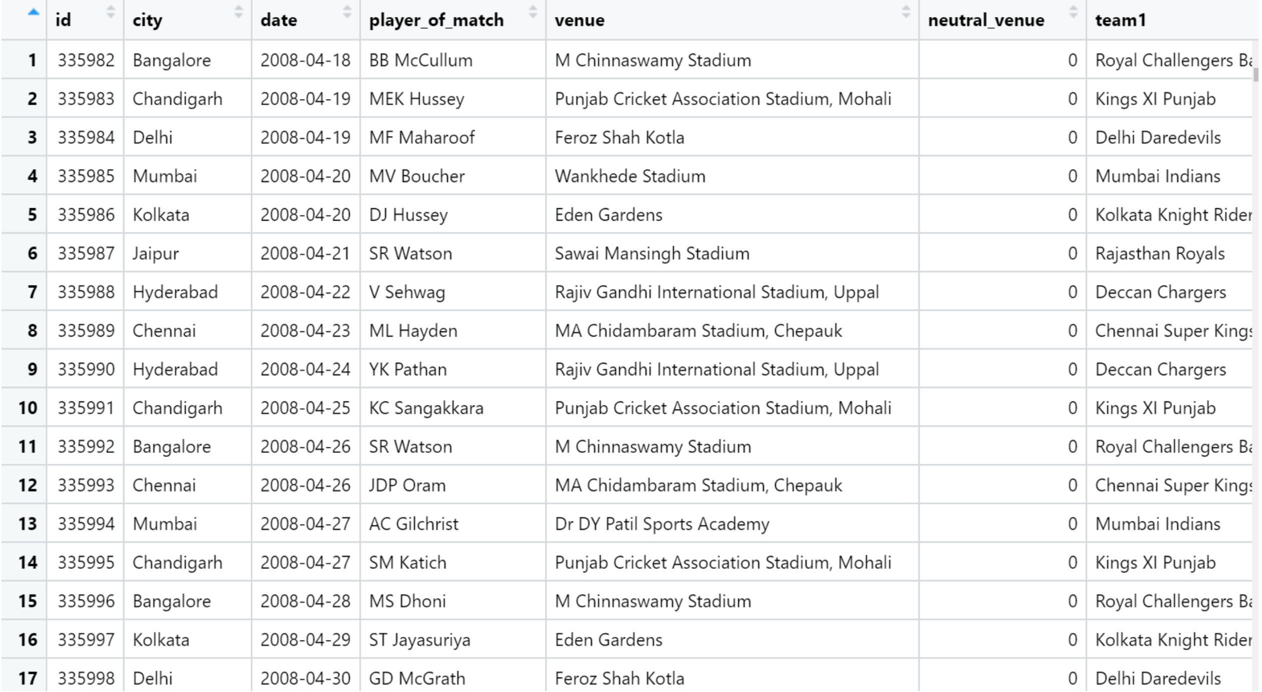
By now you must have become cognizant of the fact that IPL is almost a festival in India and like every other kid in India, we grew up celebrating it every summer since 2008. In this 13 year of IPL, a lot has been changed specially in the field of presentation and broadcasting. Since past 5-6 years, Data analytics has been extensively used by the organizers in providing tons and tons of interesting facts about the game and players during live commentary. With that in mind, we concluded what could be a better dataset than IPL to begin our Data Analytics Interpretation for our academic project.

Before we begin with the analysis, lets first load the data and have a quick look at it :

```
ipl <- read.csv("C:/Users/divya/Downloads/IPL DATASET/IPL Matches 2008
-2020.csv")
str(ipl)
```

```
## 'data.frame':    816 obs. of  17 variables:
##  $ id             : int  335982 335983 335984 335985 335986 335987
335988 335989 335990 335991 ...
##  $ city           : chr  "Bangalore" "Chandigarh" "Delhi" "Mumbai"
...
##  $ date           : chr  "2008-04-18" "2008-04-19" "2008-04-19" "20
08-04-20" ...
##  $ player_of_match: chr  "BB McCullum" "MEK Hussey" "MF Maharoof" "
MV Boucher" ...
##  $ venue          : chr  "M Chinnaswamy Stadium" "Punjab Cricket As
sociation Stadium, Mohali" "Feroz Shah Kotla" "Wankhede Stadium" ...
##  $ neutral_venue  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ team1          : chr  "Royal Challengers Bangalore" "Kings XI Pu
njab" "Delhi Daredevils" "Mumbai Indians" ...
##  $ team2          : chr  "Kolkata Knight Riders" "Chennai Super Kin
```

```
gs" "Rajasthan Royals" "Royal Challengers Bangalore" ...
##  $ toss_winner    : chr  "Royal Challengers Bangalore" "Chennai Sup
er Kings" "Rajasthan Royals" "Mumbai Indians" ...
##  $ toss_decision  : chr  "field" "bat" "bat" "bat" ...
##  $ winner         : chr  "Kolkata Knight Riders" "Chennai Super Kin
gs" "Delhi Daredevils" "Royal Challengers Bangalore" ...
##  $ result         : chr  "runs" "runs" "wickets" "wickets" ...
##  $ result_margin  : int  140 33 9 5 5 6 9 6 3 66 ...
##  $ eliminator     : chr  "N" "N" "N" "N" ...
##  $ method         : chr  NA NA NA NA ...
##  $ umpire1        : chr  "Asad Rauf" "MR Benson" "Aleem Dar" "SJ Da
vis" ...
##  $ umpire2        : chr  "RE Koertzen" "SL Shastri" "GA Pratapkumar
" "DJ Harper" ...

attach(ipl)
```



	id	city	date	player_of_match	venue	neutral_venue	team1
1	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium	0	Royal Challengers Ba
2	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab
3	335984	Delhi	2008-04-19	MF Maharoof	Feroz Shah Kotla	0	Delhi Daredevils
4	335985	Mumbai	2008-04-20	MV Boucher	Wankhede Stadium	0	Mumbai Indians
5	335986	Kolkata	2008-04-20	DJ Hussey	Eden Gardens	0	Kolkata Knight Rider
6	335987	Jaipur	2008-04-21	SR Watson	Sawai Mansingh Stadium	0	Rajasthan Royals
7	335988	Hyderabad	2008-04-22	V Sehwag	Rajiv Gandhi International Stadium, Uppal	0	Deccan Chargers
8	335989	Chennai	2008-04-23	ML Hayden	MA Chidambaram Stadium, Chepauk	0	Chennai Super Kings
9	335990	Hyderabad	2008-04-24	YK Pathan	Rajiv Gandhi International Stadium, Uppal	0	Deccan Chargers
10	335991	Chandigarh	2008-04-25	KC Sangakkara	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab
11	335992	Bangalore	2008-04-26	SR Watson	M Chinnaswamy Stadium	0	Royal Challengers Ba
12	335993	Chennai	2008-04-26	JDP Oram	MA Chidambaram Stadium, Chepauk	0	Chennai Super Kings
13	335994	Mumbai	2008-04-27	AC Gilchrist	Dr DY Patil Sports Academy	0	Mumbai Indians
14	335995	Chandigarh	2008-04-27	SM Katich	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab
15	335996	Bangalore	2008-04-28	MS Dhoni	M Chinnaswamy Stadium	0	Royal Challengers Ba
16	335997	Kolkata	2008-04-29	ST Jayasuriya	Eden Gardens	0	Kolkata Knight Rider
17	335998	Delhi	2008-04-30	GD McGrath	Feroz Shah Kotla	0	Delhi Daredevils

We can clearly see that this dataset contains 17 columns which includes id, city, date, player_of_match, venue neutral, venue, team1, team2, toss_winner, toss_decision, winner, result, result_margin, eliminator, method, umpire1 and umpire2.

Data Cleaning

Only thing left before starting the analysis is cleaning the data by deleting unwanted columns and making minor adjustments in the data to be able to write and understand the code clearly.

```
##Removing Irrelevant features
ipl <- select(ipl, -c(method,umpire1,umpire2,neutral_venue))

teams <- list("Kolkata Knight Riders" = "KKR","Chennai Super Kings"= "
CSK","Rajasthan Royals"="RR","Royal Challengers Bangalore"="RCB","Decc
an Chargers"="SRH",
              "Kings XI Punjab" = "KXIP","Delhi Daredevils" = "DC","Mu
mbai Indians"="MI","Kochi Tuskers Kerala"="KTK","Pune Warriors"="PW","
Sunrisers Hyderabad"="SRH",
              "Rising Pune Supergiants" = "RPS","Gujarat Lions"="GL","
Rising Pune Supergiant"="RPS","Delhi Capitals"="DC")

for ( i in seq(1,length(teams),1)){
  ipl <- ipl %>% mutate_all(funs(str_replace(.,names(teams)[i],teams[[
names(teams)[i]]])))
}

## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was gen
erated.

ipl$result_margin <- as.numeric(ipl$result_margin)
```

We have successfully deleted the unwanted columns("umpire1, umpire2, method,neutral_venue"). And also replaced the names of the Teams with their respective abbreviations.

Lets start with finding out total number of matches played from 2008 to 2020.

How many matches we've got in the dataset?

```
length(ipl$id)

## [1] 816
```

Our dataset says that there have been 816 cricket matches played in the history of IPL till now. Isn't that a huge number?

How many seasons we've got in the dataset?

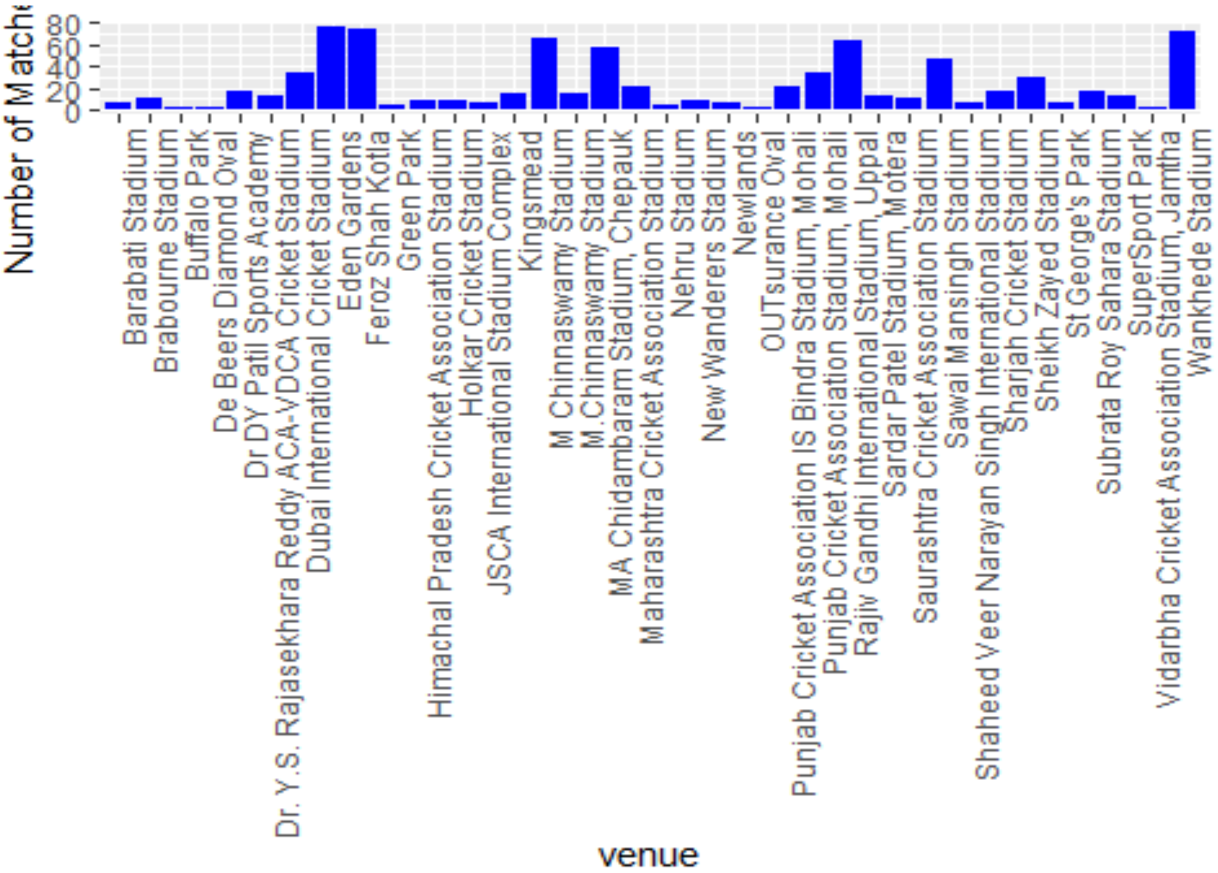
```
ipl <- ipl %>%  mutate(date = as.Date(date, format= "%Y-%m-%d") )

ipl <- ipl %>%
  mutate(
    season = case_when(
      as.numeric(format(ipl$date, "%Y")) == 2008 ~ "1",
      as.numeric(format(ipl$date, "%Y")) == 2009 ~ "2",
      as.numeric(format(ipl$date, "%Y")) == 2010 ~ "3",
      as.numeric(format(ipl$date, "%Y")) == 2011 ~ "4",
      as.numeric(format(ipl$date, "%Y")) == 2012 ~ "5",
      as.numeric(format(ipl$date, "%Y")) == 2013 ~ "6",
      as.numeric(format(ipl$date, "%Y")) == 2014 ~ "7",
      as.numeric(format(ipl$date, "%Y")) == 2015 ~ "8",
      as.numeric(format(ipl$date, "%Y")) == 2016 ~ "9",
      as.numeric(format(ipl$date, "%Y")) == 2017 ~ "10",
      as.numeric(format(ipl$date, "%Y")) == 2018 ~ "11",
      as.numeric(format(ipl$date, "%Y")) == 2019 ~ "12",
      as.numeric(format(ipl$date, "%Y")) == 2020 ~ "13",
      TRUE ~ "other"
    )
  )
```

There are total of 13 seasons with latest been played in 2020.

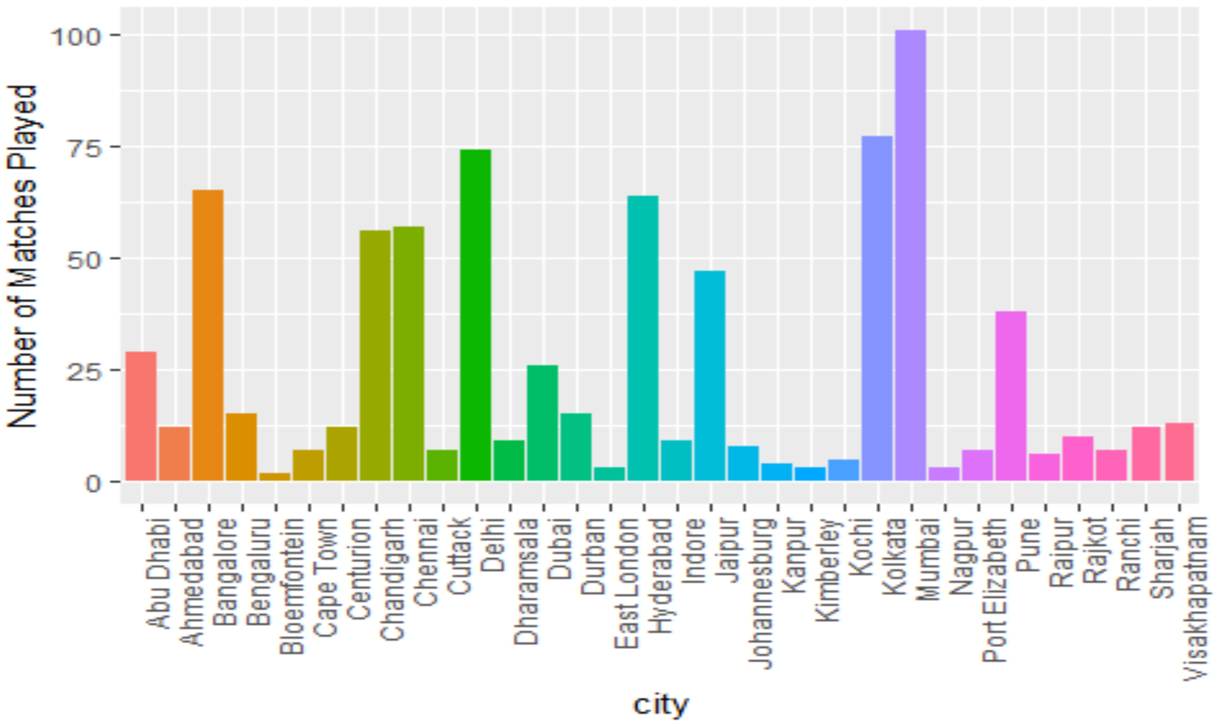
Now Let's find out which stadium hosted the highest number of matches.

Number of matches played in different stadiums.

```
ggplot(ipl,aes(venue, rm.na=T)) + geom_bar(fill="Blue") + theme(axis.t
ext.x = element_text(angle = 90, hjust = 1))+ ylab("Number of Matches
Played")
```



```
ggplot(ipl[which(!is.na(ipl$city)),],aes(city,fill= city,rm.na=T)) +ge
om_bar() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  ylab("Number of Matches Played") +
  guides(fill=FALSE)
```



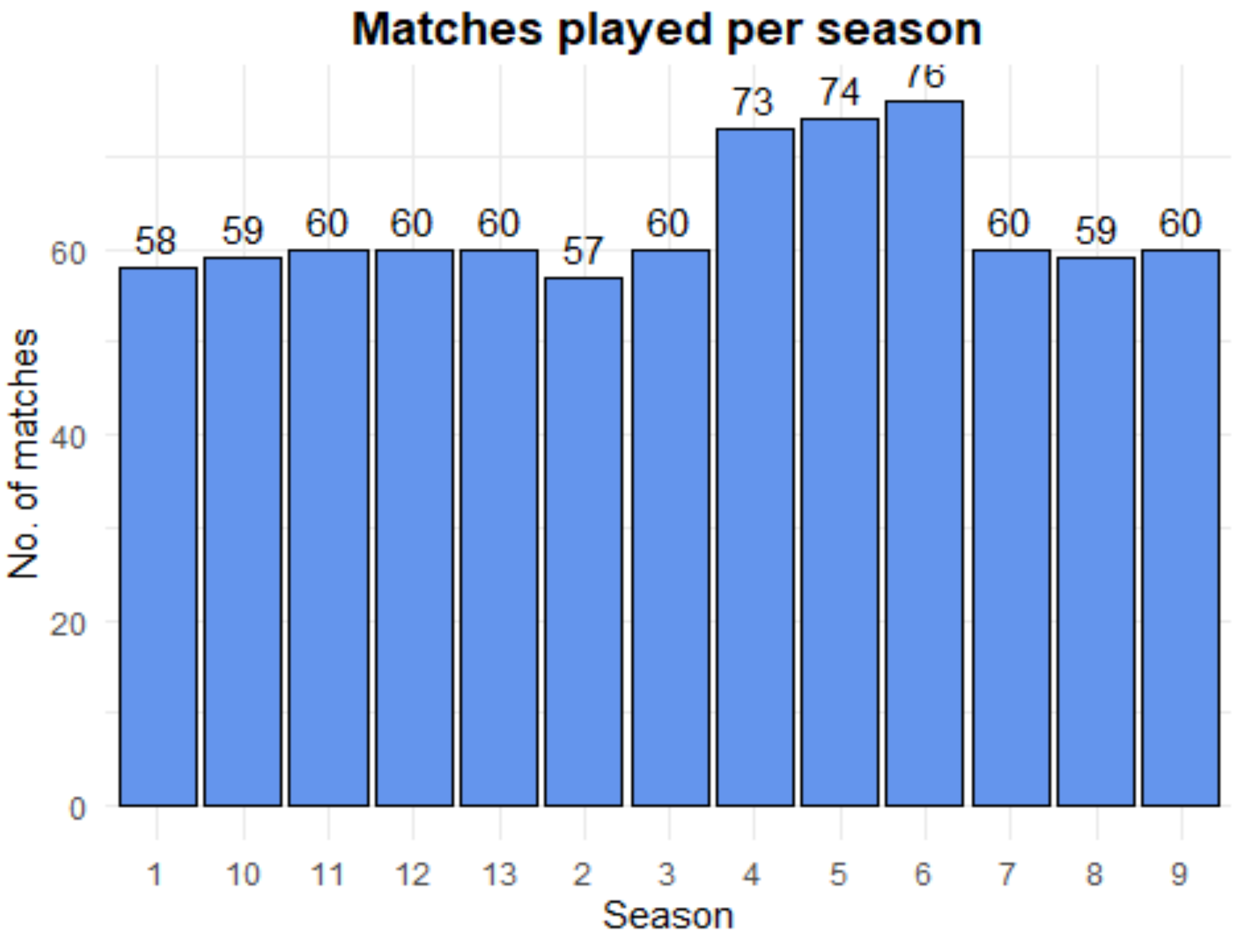
It is pretty evident that Eden Gardens, Feroz Shah Kotla, MChinnaswamy, PCA Mohali and Wankhede Stadium are top 5 stadium to host most number of matched over the years. This is probably because Eliminators and Finals are usually played in these major grounds.

We all know the format of the tournament have changed over the years. So lets see which was the longest season.

Which Season had most number of matches?

```
library(ggplot2)
season_match_count <- ipl %>% group_by(season) %>% summarise(count = n
())
print(season_match_count, row.names = FALSE)

## # A tibble: 13 x 2
##    season count
##   * <chr>  <int>
##   1 1         58
##   2 10        59
##   3 11        60
##   4 12        60
##   5 13        60
##   6 2         57
##   7 3         60
##   8 4         73
##   9 5         74
##  10 6         76
## 11 7          60
## 12 8          59
## 13 9          60
```

```
ggplot(season_match_count,
       aes(x = season,
           y = count)) +
  geom_bar(stat = "identity",fill = "cornflowerblue", color="black") +
  geom_text(aes(label = count),
            vjust=-0.5) +
  labs(x = "Season",
       y = "No. of matches",
       title  = "Matches played per season")+
  theme_minimal() +
  theme(
    plot.title=element_text( hjust=0.5, vjust=0.5, face='bold')
  )
```



```
most_wins <- ipl %>% group_by(season) %>% summarise(count = n()) %>% a
rrange(desc(count)) %>% filter(row_number()==1)
```

```
print(paste0("Season ",most_wins$season, " had most number of matches,
: " ,most_wins$count ))

## [1] "Season 6 had most number of matches, : 76"
```

After viewing the plot, it is evident that the management changed the format in the initial year but from season 7 to season 13 total number of matched are 60.

Note: Seasons 8 and 10 do not have 60 matches probably because some matches might have been called off because of rain or any other reasons.

It will be fun to see which team the match while batting first with the highest margin

Which Team had won by maximum runs?

```
max_run_win <- ipl %>% group_by(result) %>% summarise(count = max(resu
lt_margin)) %>% filter(result == "runs")

team <- ipl %>% filter( result_margin == max_run_win$count) %>% select
(winner,result,result_margin)
team

##   winner result result_margin
## 1     MI   runs           146
```

Which Team had won by maximum wicket?

```
max_wick_win <- ipl %>% group_by(result) %>% summarise(count = max(res
ult_margin)) %>% filter(result == "wickets")

team_wick <- ipl %>% filter( result_margin == max_wick_win$count & res
ult == "wickets" ) %>% distinct(winner,result,result_margin)

team_wick

##   winner  result result_margin
## 1    SRH wickets            10
## 2     DC wickets            10
## 3    RCB wickets            10
## 4     RR wickets            10
## 5     MI wickets            10
## 6    CSK wickets            10
## 7    KKR wickets            10
## 8   KXIP wickets            10
```

Which IPL Team is more successful?

There could be multiple parameters on which we can decide which team is more successful.

1) Team that won most number of matches

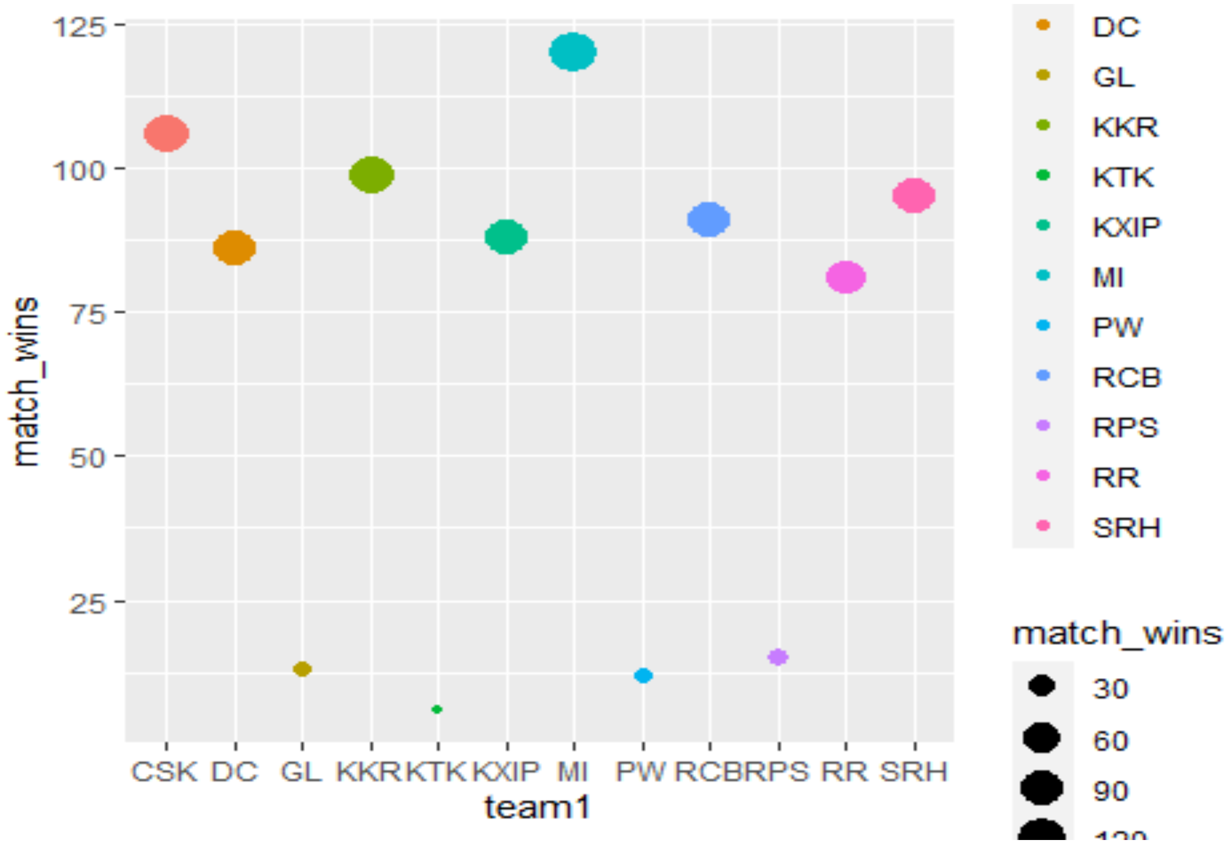
```
successful_team <- ipl %>% group_by(winner) %>% summarise(wins = n())
%>% arrange(desc(wins)) %>% top_n(n=1)

## Selecting by wins

successful_team

## # A tibble: 1 x 2
##   winner  wins
##   <chr>  <int>
## 1 MI       120
```

2)Match Summary of all teams

Overview of matches played, toss won, wins, and losses for each team in the IPL history.

```
team1 <- ipl %>% group_by(team1) %>% summarise(count = n()) %>% arrang
e(team1)

team2 <- ipl %>% group_by(team2) %>% summarise(count = n()) %>% arrang
e(team2)

toss_win <- ipl %>% group_by(toss_winner) %>% summarise(count = n()) %
>% arrange(toss_winner)

match_win <- ipl %>% group_by(winner) %>% summarise(count = n()) %>% a
rrange(winner)

team_summary <- sqldf("select a.team1, (a.count + b.count) as total_ma
tches, c.count as toss_wins, d.count as match_wins,(a.count + b.count)
- d.count as match_loss from team1 a inner join team2 b on a.team1=b.t
eam2 inner join toss_win c on  a.team1 = c.toss_winner inner join matc
h_win d on a.team1=d.winner" )

print(team_summary)

##    team1 total_matches toss_wins match_wins match_loss
## 1    CSK           178        97        106         72
## 2     DC           194       100         86        108
## 3     GL            30        15         13         17
## 4    KKR           192        98         99         93
## 5    KTK            14         8          6          8
## 6   KXIP           190        85         88        102
## 7     MI           203       106        120         83
## 8     PW            46        20         12         34
## 9    RCB           195        87         91        104
## 10    RPS              30         13          15          15
## 11     RR             161         87          81          80
## 12    SRH             199        100          95         104
```

```
ggplot(team_summary, aes(x = team1, y = match_wins  , color = team1 ,
size = match_wins)) +
  geom_point()
```



3)Team that won the winning title maximum number of times

List the winner of each season

```
season_final <- ipl %>% group_by(season) %>% summarise(finale = max(da
te)) %>% arrange(season)

season_winner <- sqldf("select a.season, a.winner from ipl a inner joi
n season_final b on a.date=b.finale")

print(season_winner)

##    season winner
## 1       1     RR
## 2       2    SRH
## 3       3    CSK
## 4       4    CSK
## 5       5    KKR
## 6       6     MI
## 7       7    KKR
## 8       8     MI
## 9       9    SRH
## 10     10     MI
## 11     11    CSK
## 12     12     MI
## 13     13     MI
```

Top 5 most successful teams in IPL

```
top_teams <- sqldf("select winner, count(winner) as title_count from season_winner group by winner order by 2 desc limit 5")
top_teams
```

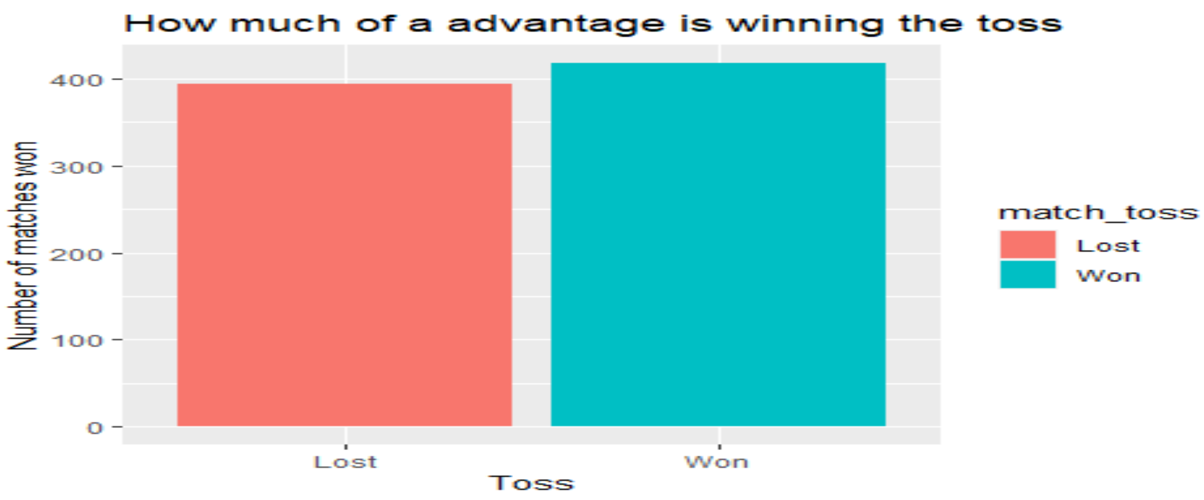
```
##   winner title_count
## 1     MI           5
## 2    CSK           3
## 3    SRH           2
## 4    KKR           2
## 5     RR           1
```

Does winning the toss has any advantage?

```
toss_stats <- ipl %>% filter( toss_winner == winner) %>% group_by(toss_winner) %>% summarise(count = n()) %>% summarise(total = sum(count))

ipl %>% select(id) %>% summarise(count = n()) %>% mutate( winning_prob = (toss_stats$total / count) * 100 )
```

```
##   count winning_prob
## 1   816     51.22549
```

```
ipl_stat <- ipl %>% select(toss_winner,winner)
ipl_stat$match_toss<-ifelse(as.character(ipl$toss_winner)==as.character(ipl$winner),"Won","Lost")

ggplot(ipl_stat[which(!is.na(ipl_stat$match_toss)),],aes(match_toss, fill = match_toss))+
  geom_bar()+ xlab("Toss") +ylab("Number of matches won")+ ggtitle("How much of a advantage is winning the toss")
```



In 816 matches over the period of 13 years, 51.2% of the times the team who won the toss also won the match. This clearly says that winning the toss does not have much effect on winning chances.

RANDOM FOREST FOR PREDICTING WINNER

Now that we have gave gained quite a useful insight from the dataset, let pivot to some more exciting stuff.

Let's try to predict the winning chances of a team from the given set of data we have.

We will be implementing the supervised machine learning algorithm **Random forest** for this purpose.

```
ipl <- ipl  %>% mutate( city = ifelse( is.na(city)  & venue == "Sharja
h Cricket Stadium","Sharjah",ifelse(is.na(city) &  venue == "Dubai Int
ernational Cricket Stadium","Dubai",city)))

ipl <- ipl  %>% mutate(winner= replace_na(winner,"Draw"))
sum(is.na(matches))

## Warning in is.na(matches): is.na() applied to non-(list or vector)
of type
## 'closure'

## [1] 0

matches <- ipl %>% select(team1,team2,city,toss_decision,toss_winner,v
enue,winner)
matches$toss_decision <- as.numeric(as.factor(matches$toss_decision))
matches$city <- as.numeric(as.factor(matches$city))
matches$venue <- as.numeric(as.factor(matches$venue))
matches$winner <- as.factor(matches$winner)
```

```
set.seed(123)
train_idx <- sample(nrow(matches), .70*nrow(matches))

matches_train <- matches[train_idx,]
matches_test <- matches[-train_idx,]


##To begin the algorithm, we will install package named “randomForest”

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine

rf <- randomForest(winner ~ team1 + team2 + venue + toss_winner + city
+ toss_decision ,data=matches_train)
rf

##
## Call:
##  randomForest(formula = winner ~ team1 + team2 + venue + toss_winne
r +      city + toss_decision, data = matches_train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 50.26%
## Confusion matrix:
##      CSK DC Draw GL KKR KTK KXIP MI PW RCB RPS RR SRH class.error
## CSK   35  4    0  0   1   0    2  5  0   3   0  5   3   0.3965517
## DC     6 20    0  0   7   0    3  8  1   2   1  3   5   0.6428571
## Draw   0  1    0  0   0   0    0  0  0   1   0  2   0   1.0000000
## GL     0  2    0  1   0   0    0  1  0   2   0  0   0   0.8333333
## KKR    3  6    0  0  44   0    4  4  1   2   0  4   5   0.3972603
## KTK    0  0    0  0   3   0    1  1  0   0   0  0   0   1.0000000
## KXIP   5  6    0  0   4   0   24 10  0   5   0  3   6   0.6190476
## MI     7 11    0  0   9   0    5 47  0   2   1  3   3   0.4659091
## PW     2  2    0  0   1   0    1  1  2   0   0  1   1   0.8181818
## RCB    5  3    0  1   1   1    2  6  0  33   1  5   7   0.4923077
## RPS    0  0    0  0   0   0    1  3  0   3   3  0   3   0.7692308
## RR     4  2    1  0   6   0    4  3  0   2   0 35   4   0.4262295
## SRH    1  5    0  0   5   0    7  0  0   6   0  4  40   0.4117647
```

```
summary(rf)
```

```
##                  Length Class  Mode
## call                   3   -none- call
## type                   1   -none- character
## predicted            571   factor numeric
## err.rate            7000   -none- numeric
## confusion            182   -none- numeric
## votes               7423   matrix numeric
## oob.times            571   -none- numeric
## classes               13   -none- character
## importance             6   -none- numeric
## importanceSD           0   -none- NULL
## localImportance        0   -none- NULL
## proximity              0   -none- NULL
## ntree                  1   -none- numeric
## mtry                   1   -none- numeric
## forest                14   -none- list
## y                    571   factor numeric
## test                   0   -none- NULL
## inbag                  0   -none- NULL
## terms                  3   terms  call
```

```
pred = predict(rf, matches_test, type ="response")
cm = table(matches_test$winner, pred)
cm
```

```
##       pred
##        CSK DC Draw GL KKR KTK KXIP MI PW RCB RPS RR SRH
##   CSK   33  3    0  0   2   0    4  1  2   3   0  0   0
##   DC     1 11    0  0   2   0    1  2  0   2   0  4   7
##   Draw   0  0    0  0   0   0    0  0  0   0   0  0   0
##   GL     0  4    0  1   0   0    2  0  0   0   0  0   0
##   KKR    1  3    0  0  15   0    0  1  0   2   0  0   4
##   KTK    0  1    0  0   0   0    0  0  0   0   0  0   0
##   KXIP   0  4    0  0   4   0   10  0  0   2   0  1   4
##   MI     4  1    0  0   3   0    2 16  0   0   0  1   5
##   PW     0  0    0  0   0   0    1  0  0   0   0  0   0
##   RCB    1  3    0  0   2   0    4  2  0  13   0  1   0
##   RPS    0  1    0  0   0   0    0  0  0   1   0  0   0
##   RR      2  1    0  0   0   0    1  3  0   2   0  9   2
##   SRH     1  0    0  0   2   0    3  2  0   1   0  2  16
```

```
mean(matches_test$winner == pred)
```

```
## [1] 0.5061224
```

Our model is performing moderately in the task of predicting chances of winning a game in the tournament with an accuracy of 50.6%.

From the confusion matrix attached above, we can take the instance of CSK, the model was able to accurately predict 33 times out of the total 45 CSK matches present in the test dataset.

Appendix

Classification is the method of predicting the class of a given input data point. Classification problems fall under the Supervised learning method.

One such classification method is **randomForest**.

randomForest function is a supervised classification and regression algorithm. As the name suggests, this algorithm randomly creates a forest with several trees. It can also be used in unsupervised mode for assessing proximities among data points.

Generally, the more trees in the forest the more robust the forest looks like. Similarly, in the random forest classifier, the higher the number of trees in the forest, greater is the accuracy of the results. Random forest builds multiple decision trees (called the forest) and glues them together to get a more accurate and stable prediction. It builds the forest which is a collection of Decision Trees, trained with the bagging method. based on the idea of bagging, which is used to reduce the variation in the predictions by combining the result of multiple Decision trees on different samples of the data set. This is what we have shown in the above use case.