Exploratory Data Analysis of cricket dataset and Analyzing Virat and Dhoni's performance in R

Tanvi

May 3, 2018

In this post, we will learn how to analyze sports performance and compare the cricketers in R. In the previous post we gave a brief idea regarding what we will exactly do in this post and how the extraction and cleaning of data are done. If you haven't gone through the previous post then I highly recommend you to do so **the link**. We have taken the data taken from **"ESPN Crickinfo Statsguru"** using the **cricketr** package. We will be analyzing individual performances of Virat Kohli and MS Dhoni and also do a comparison of their performance side by side.

Let us start by displaying the data in both the datasets before proceeding with the EDA in R.

```
head(virat)
```

```
##   Runs Mins BF 4s 6s     SR Pos Dismissal Inns  Opposition        Ground
## 1   12   33 22  1  0  54.54   2       lbw    1 v Sri Lanka      Dambulla
## 2   37   82 67  6  0  55.22   2    caught    2 v Sri Lanka      Dambulla
## 3   25   40 38  4  0  65.78   1   run out    1 v Sri Lanka Colombo (RPS)
## 4   54   87 66  7  0  81.81   1    bowled    1 v Sri Lanka Colombo (RPS)
## 5   31   45 46  3  1  67.39   1       lbw    2 v Sri Lanka Colombo (RPS)
## 6    2    6  2  0  0 100.00   7   not out    1 v Sri Lanka Colombo (RPS)
##    Start Date
## 1 18 Aug 2008
## 2 20 Aug 2008
## 3 24 Aug 2008
## 4 27 Aug 2008
## 5 29 Aug 2008
## 6 14 Sep 2009
```

The above output displays the first few rows in Virat's dataset.

```
head(dhoni)
```

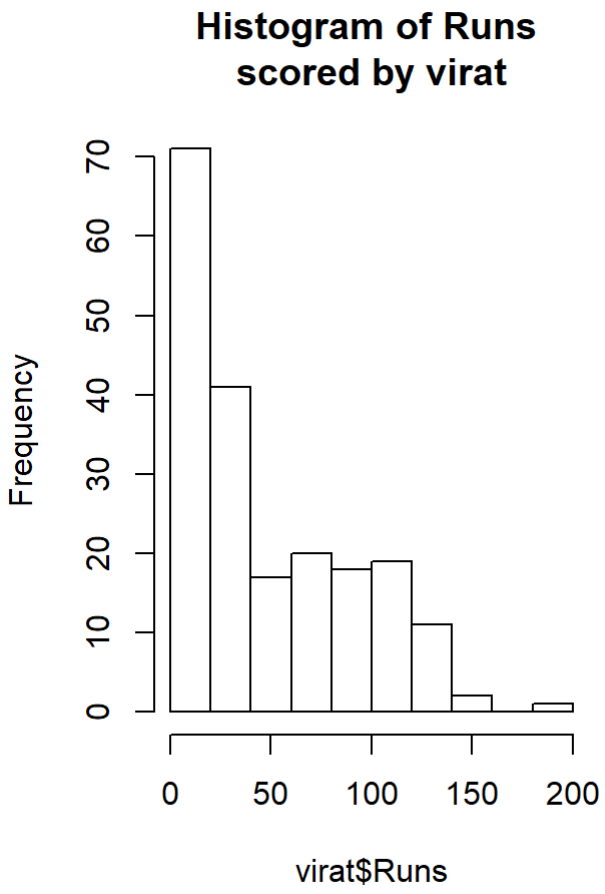
```
##   Runs Mins  BF 4s 6s     SR Pos Dismissal Inns   Opposition        Ground
## 1    0    1   1  0  0   0.00   7   run out    1 v Bangladesh    Chittagong
## 2   12   16  11  2  0 109.09   7    caught    2 v Bangladesh         Dhaka
## 3    7    2   2  0  1 350.00   7   not out    1 v Bangladesh         Dhaka
## 4    3    8   7  0  0  42.85   7    caught    1   v Pakistan         Kochi
## 5  148  155 123 15  4 120.32   3    caught    1   v Pakistan Visakhapatnam
## 6   28   36  24  5  0 116.66   3    caught    2   v Pakistan    Jamshedpur
##    Start Date
## 1 23 Dec 2004
## 2 26 Dec 2004
## 3 27 Dec 2004
## 4  2 Apr 2005
## 5  5 Apr 2005
## 6  9 Apr 2005
```

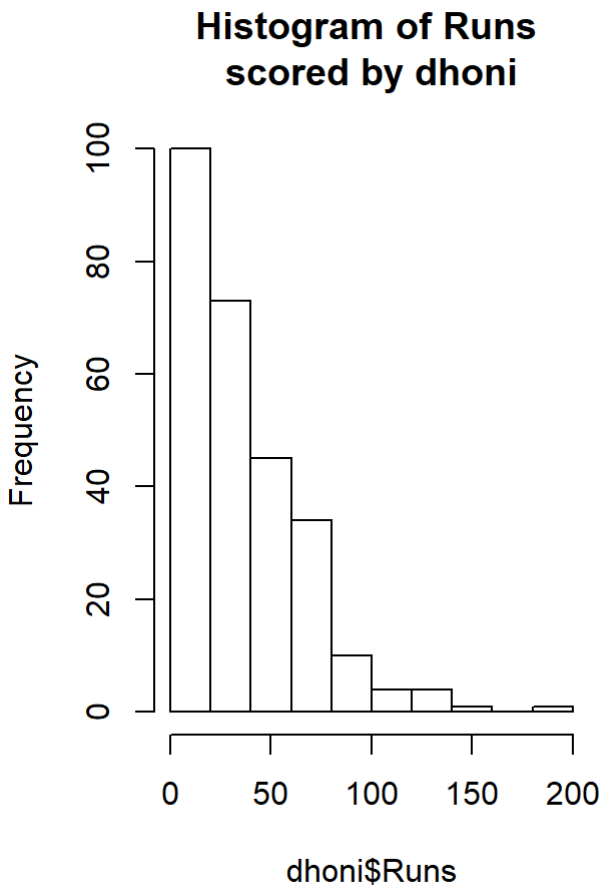
The above output displays the first few rows in Dhoni's dataset. Now, we'll proceed with the EDA.

Histogram of Runs Scored

The Histograms in R can be plotted using the `hist` function.They are used for displaying the frequencies of values of the specified data variable.

```
par(mfrow=c(1,2))
hist(virat$Runs,main="Histogram of Runs \nscored by virat")
hist(dhoni$Runs,main="Histogram of Runs \nscored by dhoni")
```





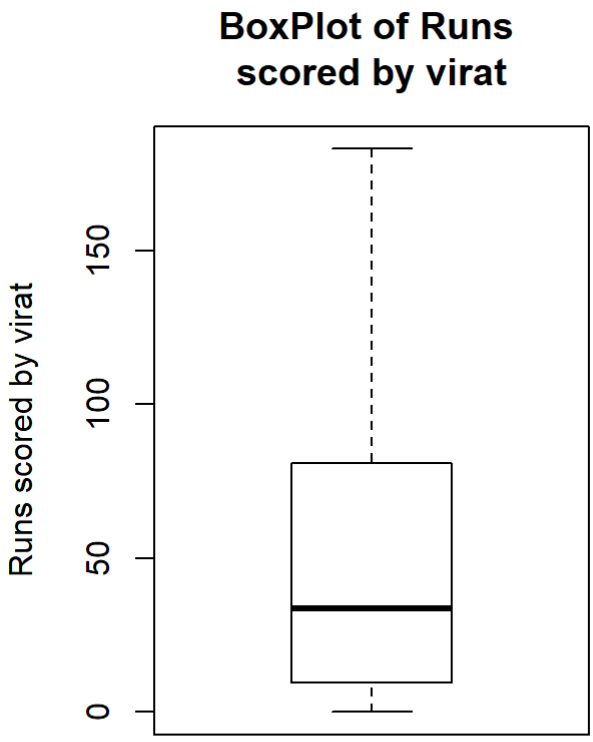
The above histograms tell us the range over which Kohli and Dhoni have scored runs as well its frequency. We can observe the following:

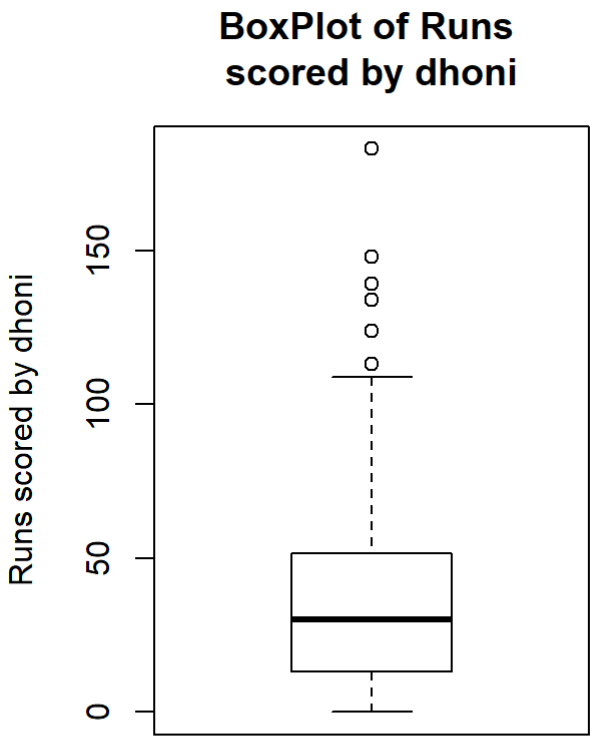
1. Both Virat and Dhoni have mostly scored runs in the range of 0-50.
2. Percentage of centuries scored by Virat is more than that of Dhoni.

Boxplot of Runs Scored

The Box Plot is used for showing the distribution of the data of the specified data variable.It can be drawn in R using `boxplot` function.

```
par(mfrow=c(1,2))
boxplot(virat$Runs,ylab='Runs scored by virat',main="BoxPlot of Runs \nscored by virat")
boxplot(dhoni$Runs,ylab='Runs scored by dhoni',main="BoxPlot of Runs \nscored by dhoni")
```





The above boxplots not only tell us about the range of the runs scored but also emphasize on the median of the runs scored by both. The line drawn is used for representing the median.

- The median score for Virat is 33.5.
- The median score for Dhoni is 30.

Also, the extreme lines in the figures known as whiskers indicate the least and the highest runs scored by both. The least runs scored by both is zero and the maximum is 183 for Kohli and 109 for Dhoni(excluding outliers). Also, the round circles in Dhoni's plot indicate outliers. Outliers are those values that are inconsistent with the range of the given values and therefore have a very different value from the rest.The values of outliers are 148,183,139,124,113,139 and 134.

Adding new columns to the data

Here, we have added 3 new columns - month, year and day. To achieve this, we have converted the character format of Date to Date Format. We have used the `separate` function so that we could extract only the year part of the data. The parameters of this function are explained below:

- `data` indicates the data with which you want to work.
- `col` indicates the column name.In this case, it is the *Start date* column which represents the date of the match.
- `into` indicates that the month has to be separated into the month, day and year part.
- `sep` indicates the separator which is used in the format of date in the dataset.

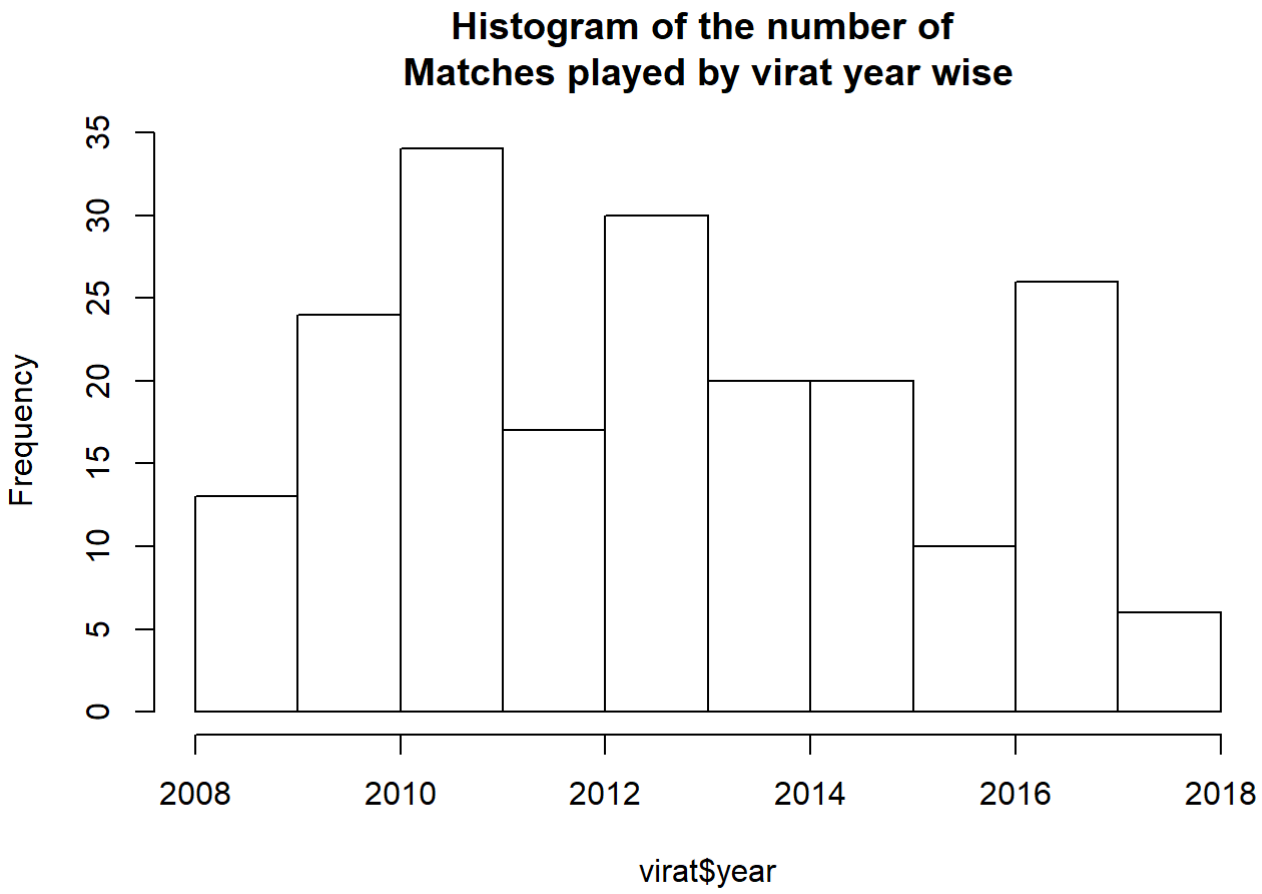
```
virat$`Start Date` <- as.Date(virat$`Start Date`, "%d %b %Y")
virat <- separate(data = virat, col = `Start Date`,
              into = c("year", "month", "day"),
              sep = "-", remove = FALSE)
head(virat)
```

```
##   Runs Mins BF 4s 6s     SR Pos Dismissal Inns  Opposition        Ground
## 1   12   33 22  1  0  54.54   2       lbw    1 v Sri Lanka      Dambulla
## 2   37   82 67  6  0  55.22   2    caught    2 v Sri Lanka      Dambulla
## 3   25   40 38  4  0  65.78   1   run out    1 v Sri Lanka Colombo (RPS)
## 4   54   87 66  7  0  81.81   1    bowled    1 v Sri Lanka Colombo (RPS)
## 5   31   45 46  3  1  67.39   1       lbw    2 v Sri Lanka Colombo (RPS)
## 6    2    6  2  0  0 100.00   7   not out    1 v Sri Lanka Colombo (RPS)
##   Start Date year month day
## 1 2008-08-18 2008    08  18
## 2 2008-08-20 2008    08  20
## 3 2008-08-24 2008    08  24
## 4 2008-08-27 2008    08  27
## 5 2008-08-29 2008    08  29
## 6 2009-09-14 2009    09  14
```

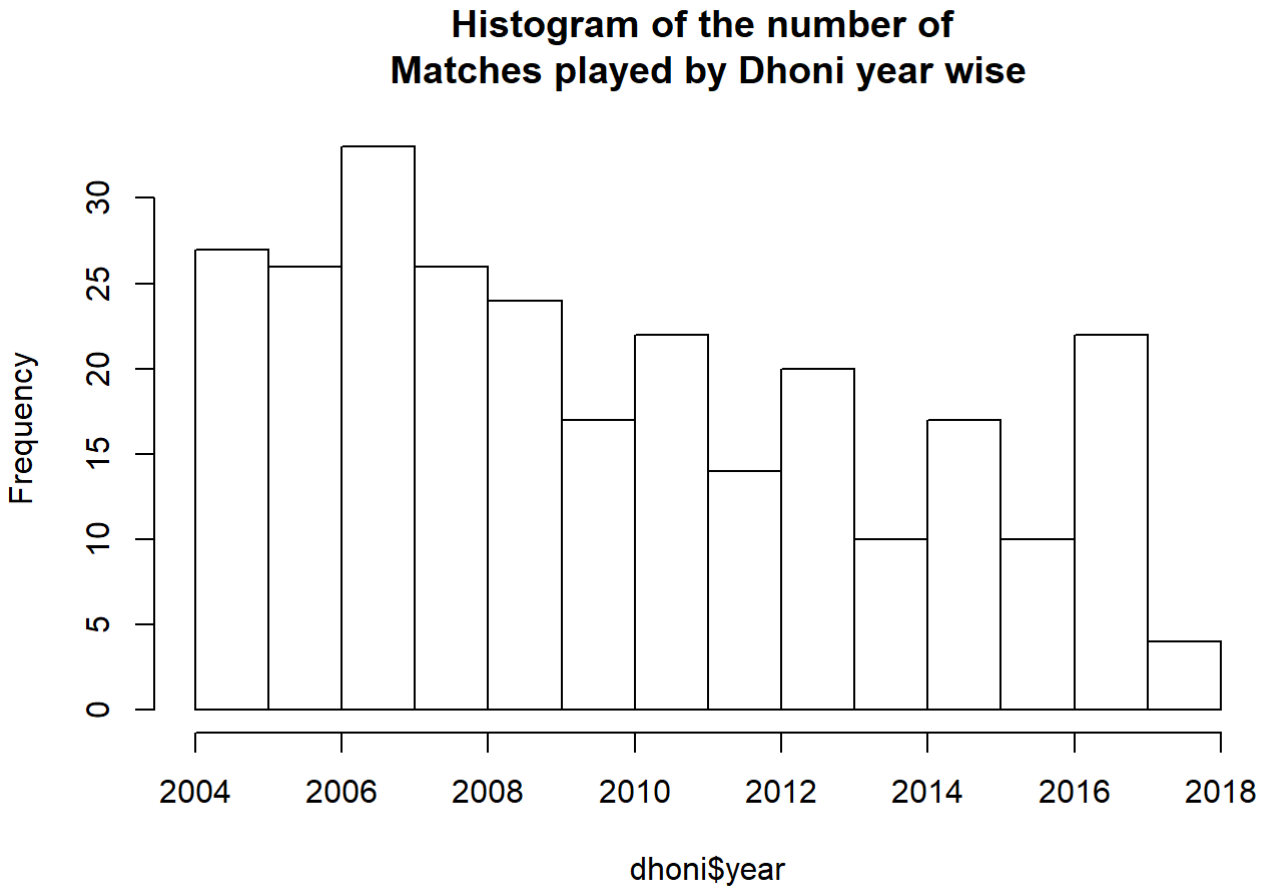
As you can see above the 3 new columns have been separately stored and printed.

Histogram of the number of matches played by the player year wise

```
virat$year <- as.numeric(virat$year)
hist(virat$year,main="Histogram of the number of \nMatches played by virat year wise")
```



From the above histogram, we observe that in 2011 Virat played the maximum number of ODI matches.



Similarly, after performing the above steps, we can observe from the above Histogram that in 2007 Dhoni played the maximum number of ODI matches.

Focussing on specific years i.e-2011 and 2007 and plotting Histograms

The `if-else` function is used for separating the 2011 and 2007 data from the rest for Kohli and Dhoni respectively. The *summary(factor)* has been used to display the frequency of the matches played in 2011 vs the rest for Kohli and 2007 vs rest for Dhoni.The `subset` function has been used to group the 2011 data and store it under a new variable v_yr11 for Kohli.Similarly, dh_yr11 is used for storing the 2007 data for Dhoni.

```
virat$yr11 <- ifelse(virat$year == 2011, "yr11", "other")
summary(factor(virat$yr11))
```

```
## other  yr11
##   166    34
```

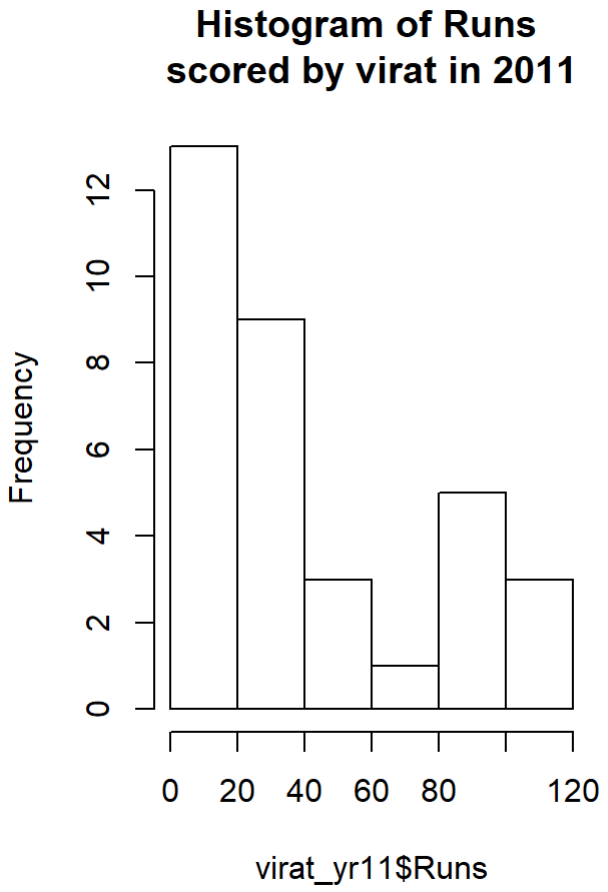
```
virat_yr11 <- subset(virat,year==2011)

dhoni$yr7 <- ifelse(dhoni$year ==2007, "yr7", "other")
summary(factor(dhoni$yr7))
```

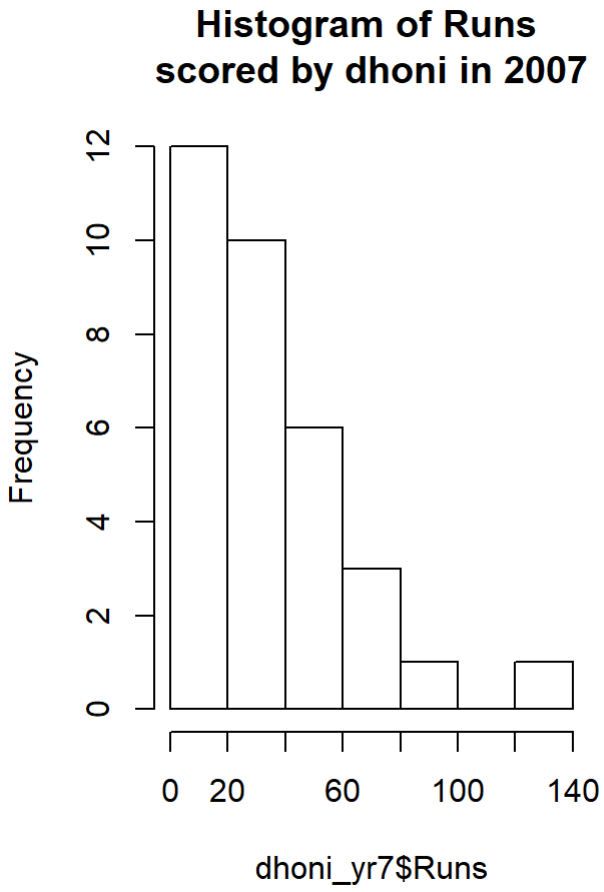
```
## other   yr7
##   239    33
```

```
dhoni_yr7 <- subset(dhoni,year==2007)
```

`Hist` function is used for plotting the histograms for the above generated variables.

```
par(mfrow=c(1,2))
hist(virat_yr11$Runs,main="Histogram of Runs \nscored by virat in 2011")
hist(dhoni_yr7$Runs,main="Histogram of Runs \nscored by dhoni in 2007")
```





Focussing on the two years we have plotted two Histograms:

1. The first indicates the frequency of the runs scored by Kohli in 2011 and the second indicates the frequency of the runs scored by Dhoni in 2007.
2. Also, the percentage of half-centuries scored by Dhoni is more whereas the percentage of centuries scored by Virat is more.

Barplots for Dismissals of the player

The next variable on which we have focussed is Dismissal which describes how the player was dismissed:

1. bowled
2. caught

3. Hit wicket
4. lbw
5. not out
6. run out
7. stumped
8. `- which means not given

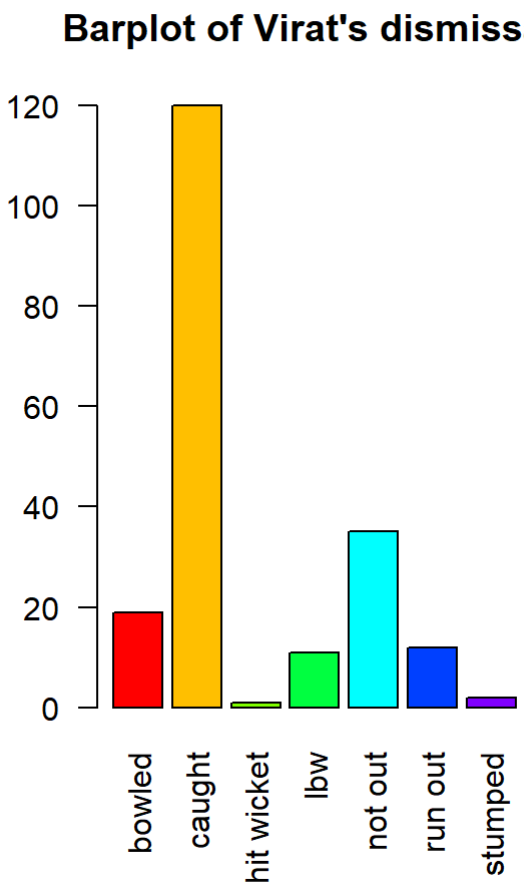
First, a table was made for Virat's and Dhoni's dismissal using `table` function. From the table, names and frequency of each were stored in two different variables.

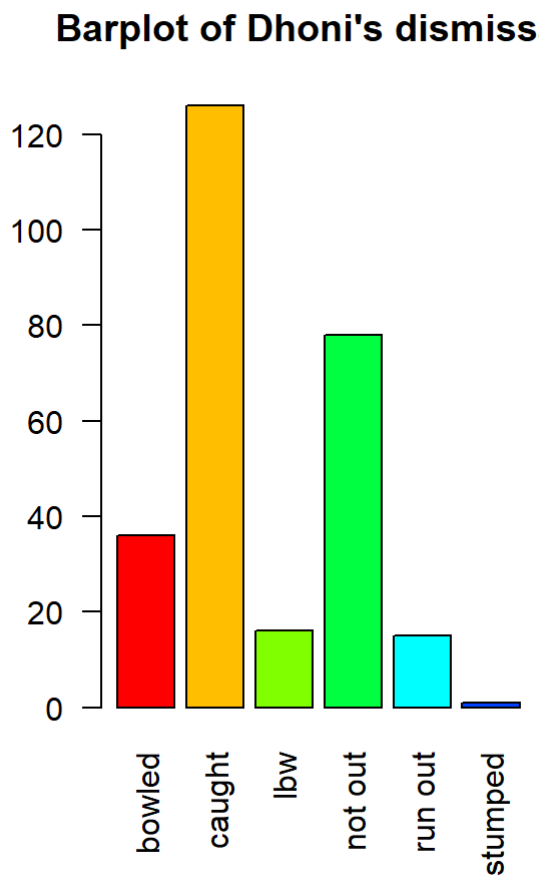
```
name<-names(table(virat$Dismissal))
cnt<-count(virat$Dismissal)
name1<-names(table(dhoni$Dismissal))
cnt1<-count(dhoni$Dismissal)
```

Below, we have used the `barplot` function to plot the barplot.It is used for plotting the frequencies of categorical variables as its height. The parameters used in the function are as follows:

- `height` indicates the length of the bars. In this case, it is the frequency of each type of dismissal.
- `names` indicates the name which needs to be given to each bar. In this case, it is the name of the type of dismissal.
- `col` indicates the color to be given to each bar. In this case, we have given rainbow colors.
- `las` indicates the rotation of the label.
- `main` indicates the title to be given to the barplot.

```
par(mfrow=c(1,2))
barplot(cnt$freq,names=name,col=rainbow(8),las=2,main="Barplot of Virat's dismissal ")
barplot(cnt1$freq,names=name1,col=rainbow(8),las=2,main="Barplot of Dhoni's dismissal ")
```





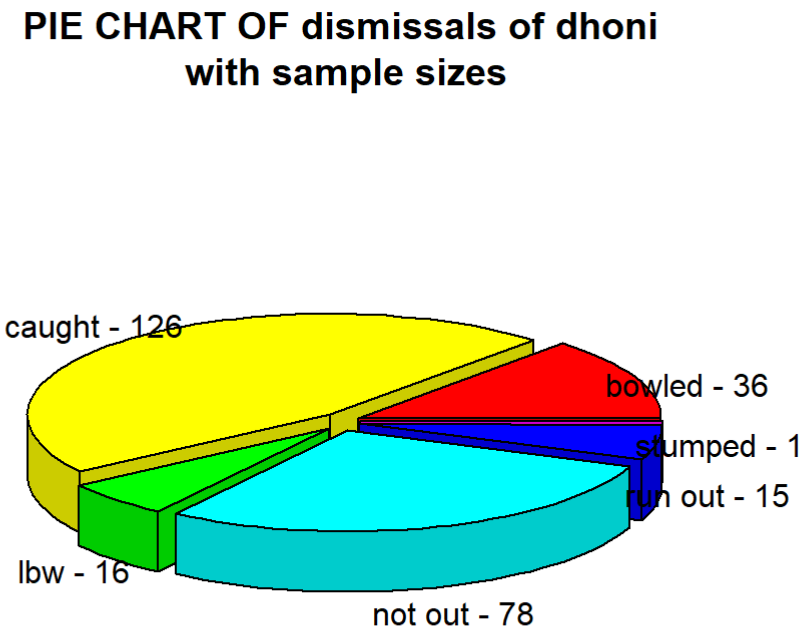
From the above barplots, we observe that both the players were mostly caught out.

Pie3D for Dismissals

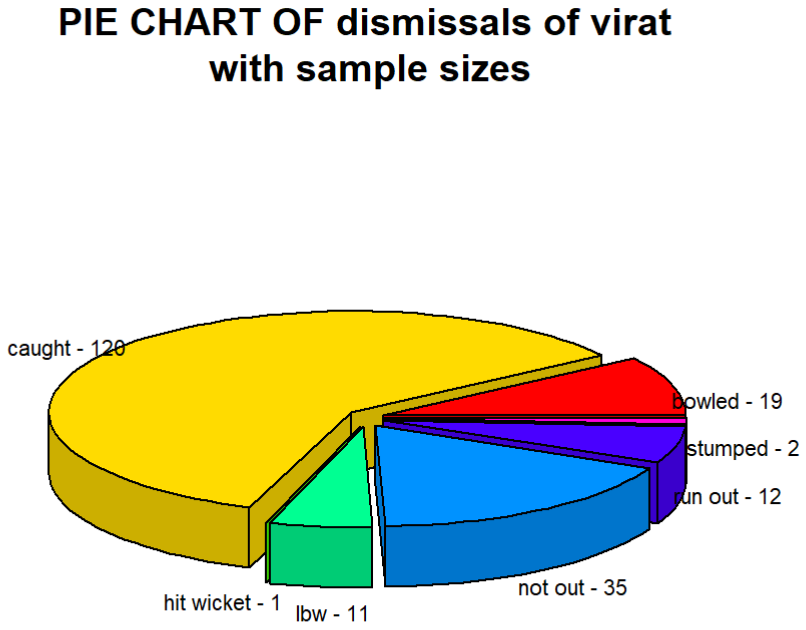
The pie3D function has been used for drawing a 3d pie chart. The parameters are explained below:

- `x` indicates the data to be used.Here we are taking the values of each from the table we created using the table function stored in d1.
- `labels` indicates the names to be given to each slice formed in the pie chart. In this case, we have given the factor values which we had used above while dealing with Dismissal variable.
- `explode` indicates the amount by which you want to explode the pie.
- `main` indicates the title to be given to the pie chart.
- `labelcex` indicates the font size for the labels.

```
d1<-table(dhoni$Dismissal)
labs1<- paste(names(d1),"-",d1)
d<-table(virat$Dismissal)
labs<- paste(names(d),"-",d)
pie3D(d1, labels = labs1, explode = 0.06,main="PIE CHART OF dismissals of dhoni\n with s
ample sizes",labelcex = 1)
```




```
pie3D(d, labels = labs, explode = 0.06,main="PIE CHART OF dismissals of virat\n with sam
ple sizes",labelcex=0.7)
```



The above pie charts also show the frequency of the type of dismissals for both the players. From here also we can conclude that both of them were mostly dismissed because they were caught out.

Applying tapply

The `tapply` function is used for calculating a measure for a group of values.The parameters can be explained below:

- `vector` is the variable for which the measure needs to be calculated.
- `index` is the variable under which the data needs to be grouped.
- `measure or function` is the measure or the function which needs to be applied to the values passed as the first parameter.

```
tapply(virat$Runs,virat$Inns,mean)
```

```
##        1        2
## 43.86207 51.07965
```

The above output gives the mean of the runs scored in Innings 1 and 2 separately for Virat.

```
tapply(dhoni$Runs,dhoni$Inns,mean)
```

```
##        1        2
## 40.30714 32.75758
```

The above output gives the mean of the runs scored in Innings 1 and 2 separately for Dhoni. We can observe that Kohli had a higher average in second innings whereas Dhoni had a higher average in the first innings.

Applying Linear regression between Runs Scored and Balls Faced

After converting the variables to numeric, all the *not available* entries have been replaced with zero to avoid errors in calculations. Then, we applied the `lm` function to generate a linear relationship between runs scored and balls faced by Virat.

```
virat$Runs <- as.numeric(virat$Runs)
virat$Runs[is.na(virat$Runs)] <- 0
virat$BF <- as.numeric(virat$BF)
virat$BF[is.na(virat$BF)] <- 0
fit<-lm(virat$Runs ~ virat$BF)
fit
```

```
##
## Call:
## lm(formula = virat$Runs ~ virat$BF)
##
## Coefficients:
## (Intercept)     virat$BF
##      -6.397        1.044
```

Above the values of the intercept and the coefficient(slope) for balls faced is displayed.

Similarly, after converting the variables to numeric, all the *not available* entries have been replaced with zero to avoid errors in calculations. We again applied the `lm` function to generate a linear relationship between runs scored and balls faced by Dhoni.

```
dhoni$Runs <- as.numeric(dhoni$Runs)
dhoni$Runs[is.na(dhoni$Runs)] <- 0
dhoni$BF <- as.numeric(dhoni$BF)
dhoni$BF[is.na(dhoni$BF)] <- 0
fit1<-lm(dhoni$Runs ~ dhoni$BF)
fit1
```

```
##
## Call:
## lm(formula = dhoni$Runs ~ dhoni$BF)
##
## Coefficients:
## (Intercept)     dhoni$BF
##     -0.6009       0.8986
```

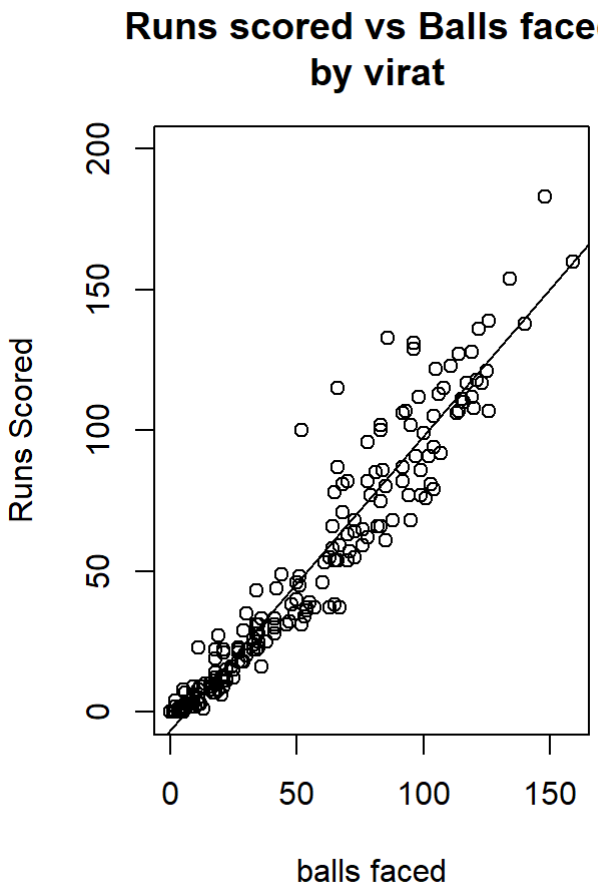
Above the values of the intercept and the coefficient(slope) for balls faced was displayed.

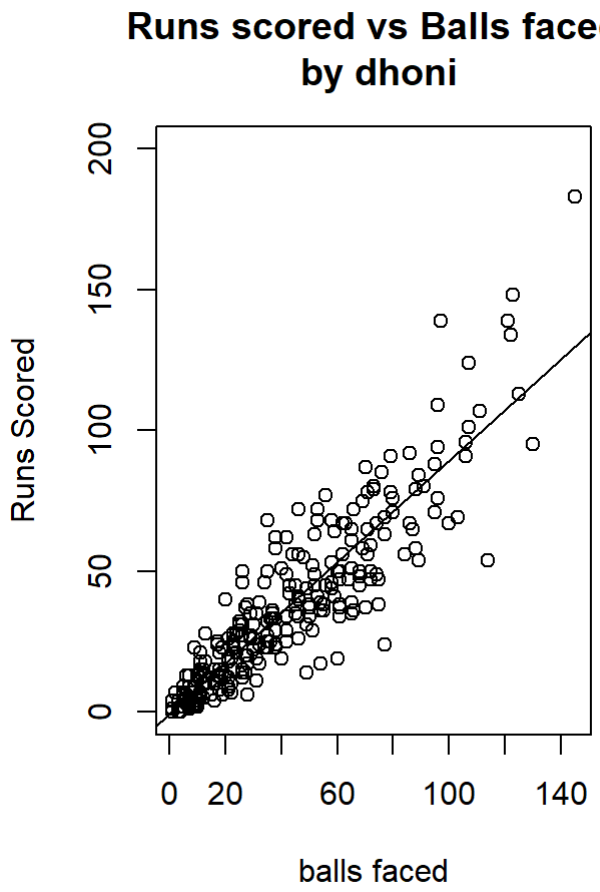
Then plots are drawn using `plot` function and give the relationship between the runs scored and the balls faced.The *plot* function's parameters can be explained below:

- The first indicates the variable of which the values need to be plotted.
- `lwd` indicates the line width.
- `ylim` indicates the limit you want to sepcify for y axis.
- `main` indicates the title to be specified for the plot.
- `xlab` indicates the label to be given for the x axis.
- `ylab` indicates the label to be given for the y axis.

The `abline` function is used for drawing a straight line passing through the maximum number of data points in both the plots.

```
par(mfrow=c(1,2))
plot(virat$Runs ~ virat$BF,
     lwd = 1,ylim = c(0, 200),
     main = "Runs scored vs Balls faced\n by virat",
     xlab = "balls faced", ylab = "Runs Scored")
abline(fit)
plot(dhoni$Runs ~ dhoni$BF,
     lwd = 1,ylim = c(0, 200),
     main = "Runs scored vs Balls faced\n by dhoni",
     xlab = "balls faced", ylab = "Runs Scored")
abline(fit1)
```





We can summarise the following from above:

1. Virat's plot has a higher slope. This means that for every extra ball faced by Virat we can expect the runs to increase by an average of 1.044.
2. Dhoni's plot has less slope. This means that for every extra ball faced by Dhoni we can expect the runs to increase by an average of 0.8986.

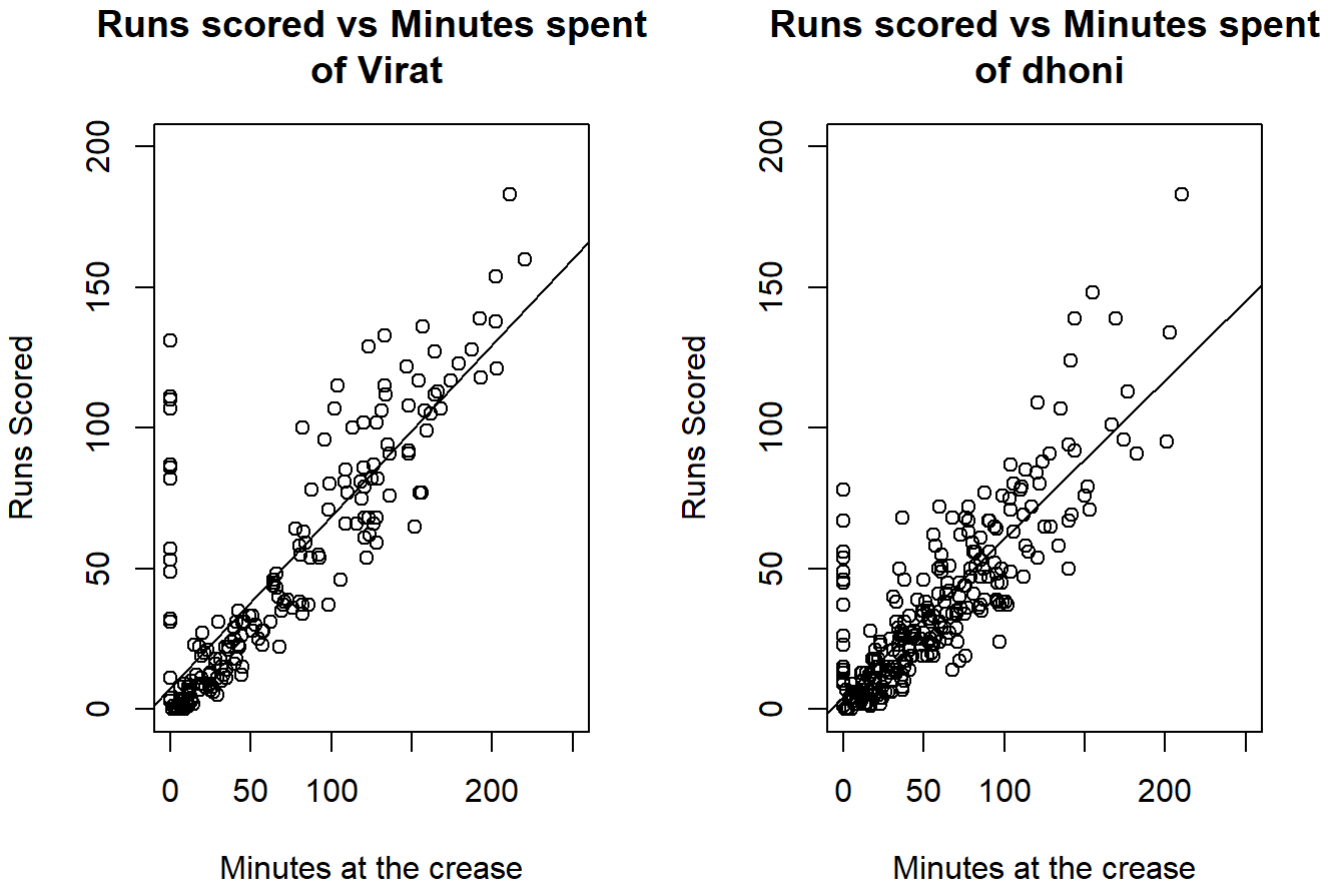
Applying Linear regression between Runs Scored and Minutes Spent at the Crease

Here we will show the relationship between the runs scored and minutes spent at the crease. After converting the variables to numeric, all the *not available* entries have been replaced with zero to avoid errors in calculations. We again applied the `lm` function to generate a linear relationship between runs scored and minutes spent at the crease by Virat as well as Dhoni.

```
virat$Mins <- as.numeric(virat$Mins)
virat$Mins[is.na(virat$Mins)] <- 0
dhoni$Mins <- as.numeric(dhoni$Mins)
dhoni$Mins[is.na(dhoni$Mins)] <- 0
fit<-lm(virat$Runs ~ virat$Mins)
fit1<-lm(dhoni$Runs ~ dhoni$Mins)
```

Similarly, the below plots have been made using `plot` and `abline` functions.The plots give the relationship between the runs scored and the minutes spent at the crease.

```
par(mfrow=c(1,2))
plot(virat$Runs ~ virat$Mins,
     lwd = 1,xlim=c(0,250),ylim = c(0, 200),
     main = "Runs scored vs Minutes spent\n of Virat",
     xlab = "Minutes at the crease", ylab = "Runs Scored")
abline(fit)
plot(dhoni$Runs ~ dhoni$Mins,
     lwd = 1,xlim=c(0,250),ylim = c(0, 200),
     main = "Runs scored vs Minutes spent\n of dhoni",
     xlab = "Minutes at the crease", ylab = "Runs Scored")
abline(fit1)
```



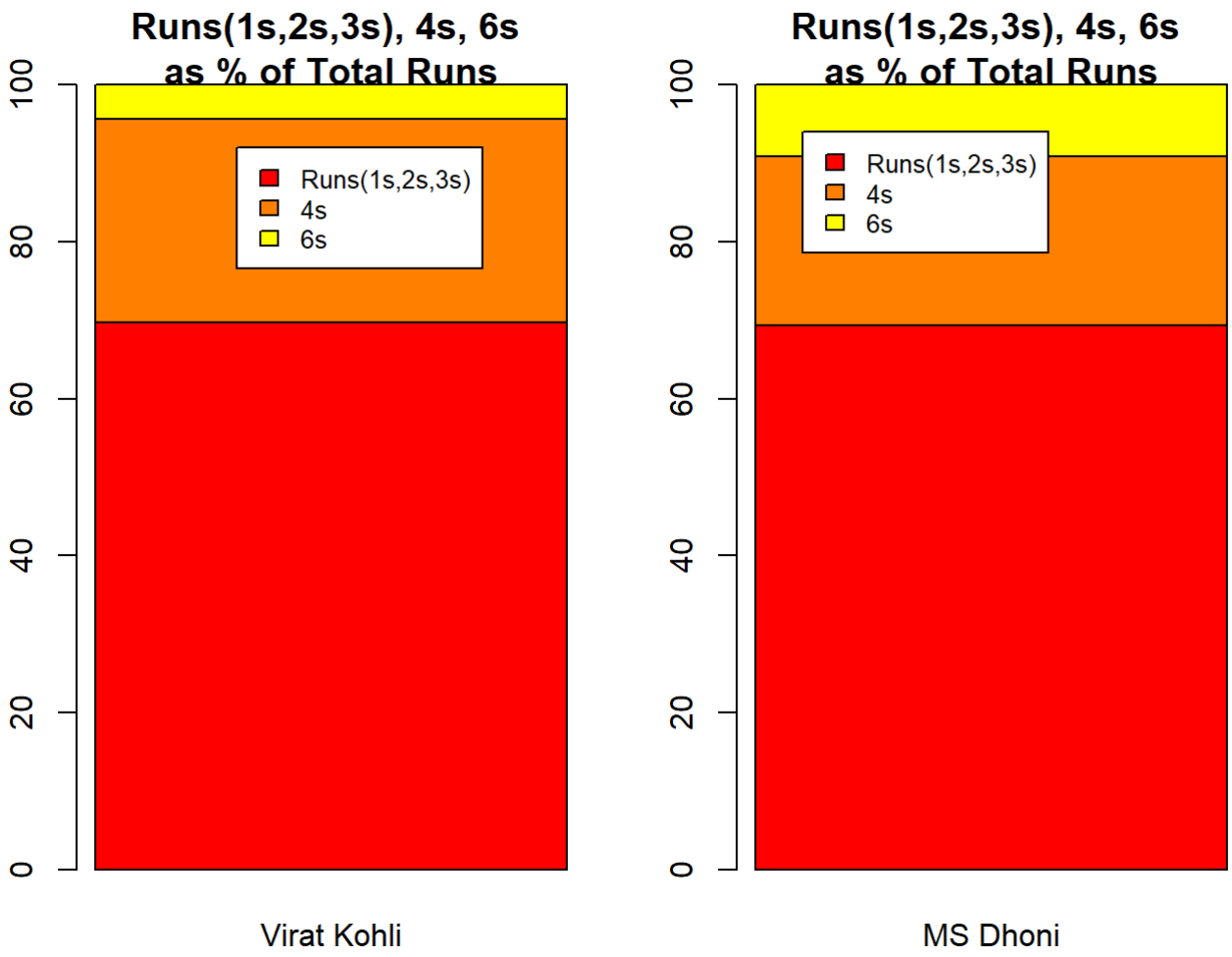
We can summarise the following:

1. Virat's plot has a higher slope. This means that for every extra minute spent at the crease by Virat we can expect the runs to increase by an average of 0.6112.
2. Dhoni's plot has less slope. This means that for every extra minute spent at the crease by Dhoni we can expect the runs to increase by an average of 0.5638.

Barplots for the percentage of 4s,6s, and others in the total runs scored

The `batsman4s6s()` is a function already available in cricketr package.You need to pass the file name and the player name as the parameters of the function. This gives the stacked plots of the percentage of 4s,6s, and others in the total runs scored by both.

```
batsman4s6s("kohliOd","Virat Kohli")
batsman4s6s("dhoniOd","MS Dhoni")
```

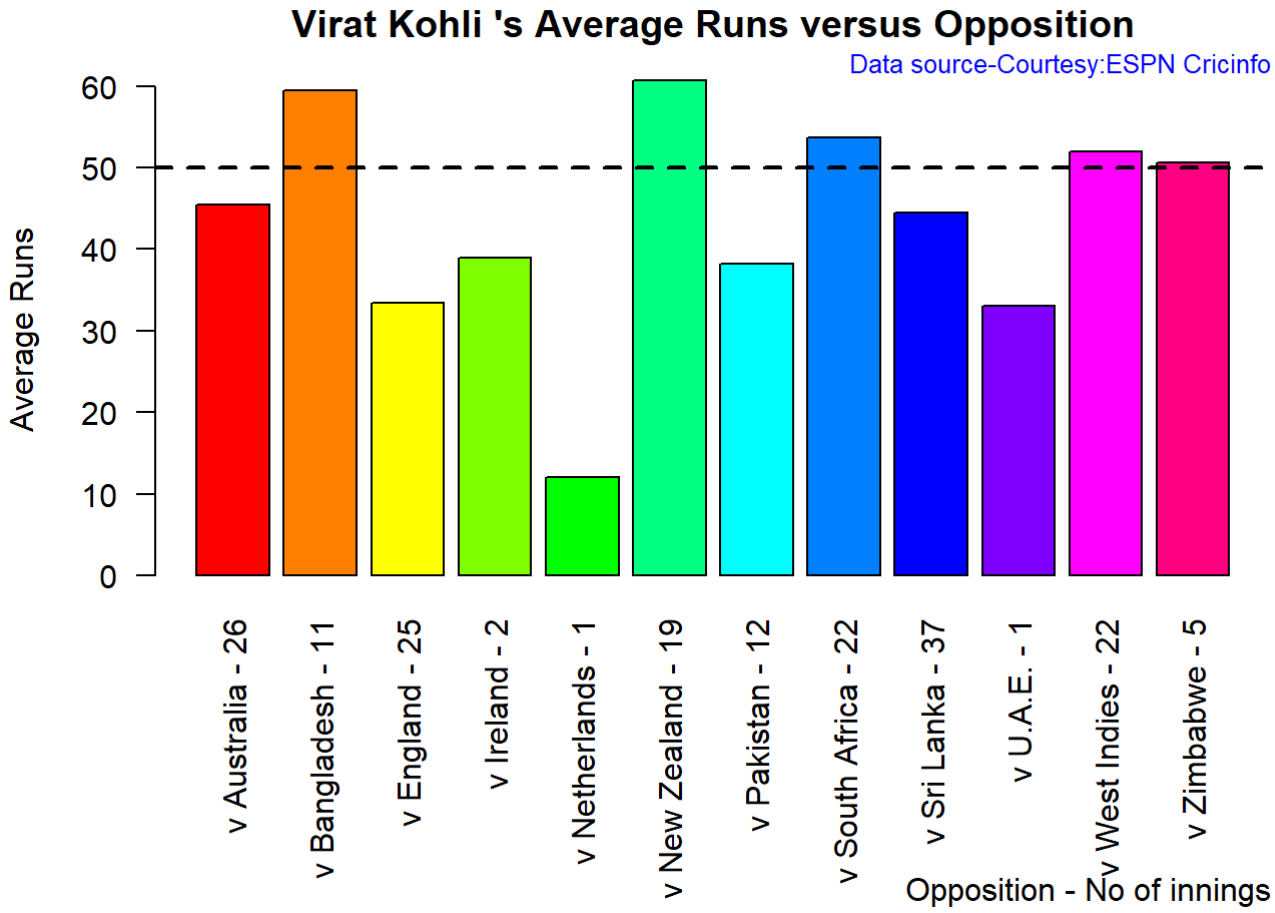


We can observe that percentage of 1s,2s and 3s scored by both Virat and Dhoni is almost same. The percentage of 4s scored is more for Virat whereas the percentage of 6s scored is more for Dhoni.

Barplots for average runs scored against each opposition

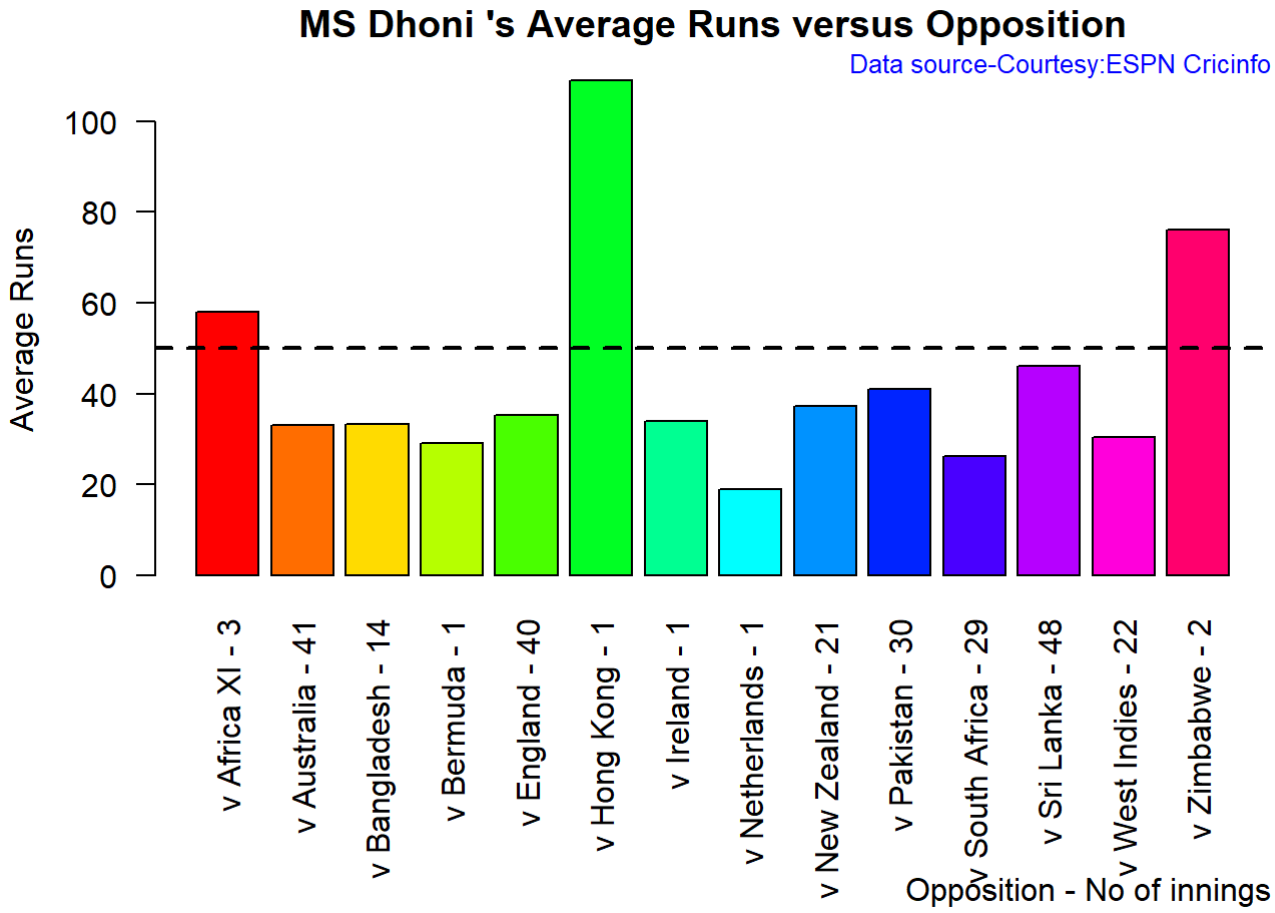
`batsmanAvgRunsOpposition` function is already available in cricketr package.The above plot shows the average runs scored by Kohli against each opposition available in the dataset. You need to pass the file name and the player name as the parameters of the function.

```
batsmanAvgRunsOpposition("kohliOd","Virat Kohli")
```



From above plot, we can observe that Virat had a high average against Bangladesh and New Zealand.

```
batsmanAvgRunsOpposition("dhoniOd","MS Dhoni")
```



From above plot, we can observe that Dhoni had a high average against Hong Kong.

Line Plot for Strike Rates

Here we are dealing with the strike rates(SR) of both the players. After converting the variables to numeric, all the *not available* entries have been replaced with zero to avoid errors in calculations.

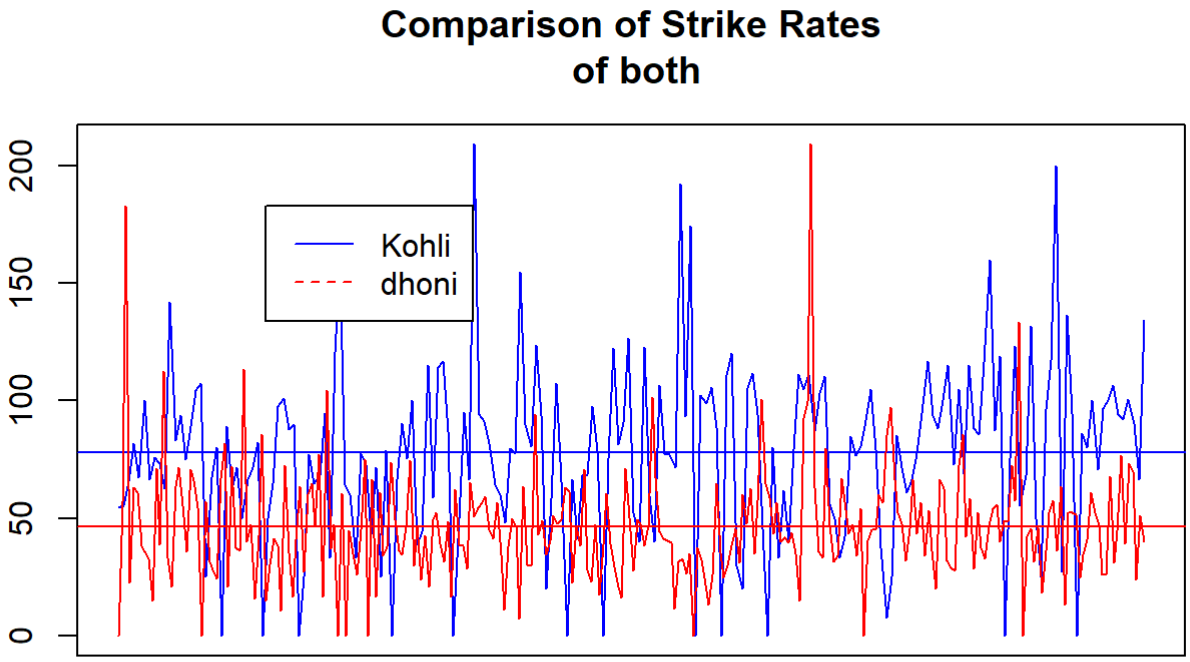
```
virat$SR <- as.numeric(virat$SR)
virat$SR[is.na(virat$SR)] <- 0
dhoni$SR <- as.numeric(dhoni$SR)
dhoni$SR[is.na(dhoni$SR)] <- 0
```

The `plot` function has been used to plot lines.The parameters can be explained below:

- The first indicates the variable of which the values need to be plotted.
- `lwd` indicates the line width.
- `type` indicates the type of the plot.Here we have given *l* which indicates a line plot.
- `xaxt='n'` has been given to remove the default x-axis values.
- `col` indicates the color of the line.
- `ann=FALSE` indicates that the default title for x and y-axis should not appear.
- `axes=false` in the next plot function is used to indicate that the axes should not be drawn again as it is already drawn on the previous plot.

Finally, we have shown the comparison of strike rates of both the players in a single plot. We have also drawn a line to specify the mean strike rate of both the players.

```
par(oma=c(0,0,2,0))
plot(virat$SR,type="l",xaxt="n",col="blue",ann=FALSE)
abline(h=mean(virat$SR),col="blue")
title("Comparison of Strike Rates\n of both")
par(new=TRUE)
plot(dhoni$SR,type="l", ann=FALSE, axes=FALSE,col='red')
abline(h=mean(dhoni$SR),col="red")
legend(40,350,legend=c("Kohli","dhoni"),col=c("blue", "red"),lty=1:2)
```



We observe that mean strike rate of Kohli is higher than that of Dhoni for the given data.

Hope this post was informative as well as interesting. Please feel free to comment or ask for more details in the comment section. In the next and the final post of this series, we will make use of Machine Learning techniques for complex analysis.

Thanks,

Tanvi (https://www.linkedin.com/in/tanvipareek)