EDA FINAL PROJECT AKSHAYA RAVICHANDRAN

CODE:

```
dim(gamelog)
```

```
## [1] 168966     21
```

```
length(unique(gamelog$MatchNo))
```

```
## [1] 688
```

We can infer that there are 688 matches and 21 teams belonging to either IPL or T20I League.

DATA CLEANING:

STEP 1: Remove duplicated rows and retain only the distinct/unique rows in the gamelog dataframe.

```
gamelog_distinct <- gamelog %>% distinct()
```

STEP 2: Handling missing data:

We find that 8304 BowlerID, 9787 BatsmanID, 149055 FielderID and 12 NumOutcome are missing. The FielderID's are missing for a reason i.e. there might not have been a fielder present at that point in the game, so they are missing. Therefore, we do not drop the NA's in FielderID. We also retain the missing rows in BowlerID and BatsmanID columns which are missing completely at random, in order to not lose too much data in the process and since we won't be using them much for our analysis. We drop the 12 NumOutcome NA values.

```
head(summary(gamelog_distinct))
head(gamelog_distinct %>% drop_na(NumOutcome))
which(is.na(gamelog_distinct$Outcome))
```

STEP 3: Dropping matches having more than 20 wickets and total_runs greater than 1000.

```
head(gamelog_distinct %>%
group_by(MatchNo) %>%
summarise(total_runs = sum(NumOutcome)) %>%
arrange(desc(total_runs)) )

gamelog_distinct  <-filter(gamelog_distinct,  MatchNo!= 201558)
gamelog_distinct  <-filter(gamelog_distinct,  MatchNo!= 200901)


gamelog_distinct <-gamelog_distinct  %>%  mutate(Wicket_w = Wickets - lag(Wickets))

p<-gamelog_distinct %>% group_by(MatchNo)  %>% summarise(total_wickets= sum(pmax(Wicket_w,0)))
filter(p, total_wickets>20)

gamelog_distinct  <-filter(gamelog_distinct,  MatchNo!= 200921)
```

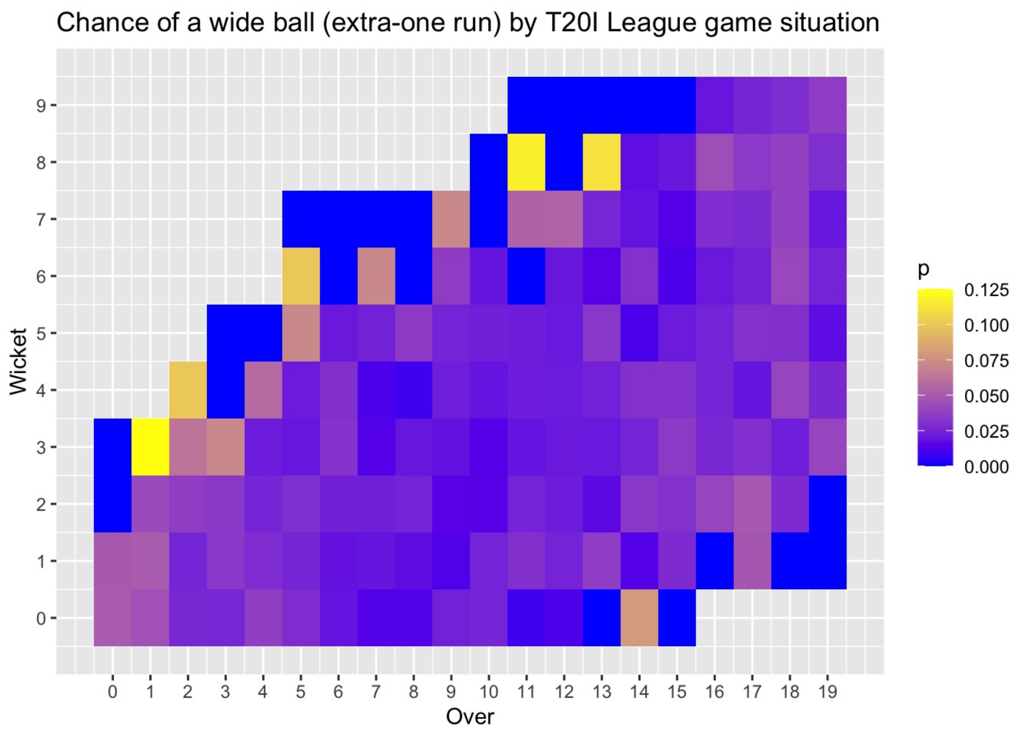
We observe that MatchNo 201558 and 200901 have total_runs >1000 which is quite impossible in a T20 match format, so we drop them. We also observe that MatchNo 200921 has more than 24 wickets so we drop it as well.

QUESTION 1: Make summary statistics

```
filter_gamelog <- function(gamelog_df, over, w) {
  filter(gamelog_df, Over == (over - 1), Wickets == (w - 1), Format == "T20I")
}
```

QUESTION 1: PLOT1 - Heatmap representing the chance of getting a wide ball in the T20I League game situation:

```
p6df <- gamelog_distinct %>% filter(Format == "T20I") %>% group_by(Over, Wickets) %>% summarise(p = length(which(
BallType=="wide")) / n())

ggplot(p6df, aes(x = Over, y = Wickets, fill = p)) + geom_tile() + scale_fill_gradient(low ="blue", high = "yello
w", na.value="white") + labs(title="Chance of a wide ball (extra-one run) by T20I League game situation", x = "Ov
er", y = "Wicket")+scale_x_continuous(breaks= seq(0,19,by=1))+ scale_y_continuous(breaks= seq(0,9,by=1))
```



We can observe that the probability of getting a wide is high in the 1st,11th,13th overs and in the 3rd,8th wickets. The probability of bowling a wide ball is very low in the first and last few wickets (9th) and in the last 5 overs mainly because the best bowlers are usually involved in the last few overs in order to control the runs scored by the team batting in the second inning.

```
#Replacing the -1 in gamelog with 0 to calculate total number of runs
gamelog_distinct_copy <- gamelog_distinct

gamelog_distinct_copy["NumOutcome"][gamelog_distinct_copy["NumOutcome"] == -1] <- 0
```

No of sixes, fours, ones, twos threes and fours obtained in overall by each of the teams over the years:

```
#NOTE FOR THE SQL PART HAS BEEN REFERENCED FROM ONLINE RESOURCES INCLUDING STACK OVERFLOW AND VARIOUS OTHER BLOG
PAGES
library("sqldf", quietly = T)
NumOutcome <- gamelog_distinct_copy %>%
  select(TeamBatting ,NumOutcome) %>%
  filter(NumOutcome %in% c(6,4,3,2,1,5,7))

sixes <- sqldf("select TeamBatting as Team_batting , count(NumOutcome) as no_of_sixes from gamelog_distinct_copy
      where NumOutcome = 6 group by TeamBatting")



fours <- sqldf("select TeamBatting as Team_batting , count(NumOutcome) as no_of_fours from gamelog_distinct_copy
      where NumOutcome = 4 group by TeamBatting")



threes <- sqldf("select TeamBatting as Team_batting , count(NumOutcome) as no_of_threes from gamelog_distinct_cop
y
      where NumOutcome = 3 group by TeamBatting")



twos <- sqldf("select TeamBatting as Team_batting , count(NumOutcome) as no_of_twos from gamelog_distinct_copy
      where NumOutcome = 2 group by TeamBatting")


ones <- sqldf("select  TeamBatting as Team_batting , count(NumOutcome) as no_of_ones from gamelog_distinct_copy
      where NumOutcome = 1 group by TeamBatting")


zeros <- sqldf("select TeamBatting as Team_batting , count(NumOutcome) as no_of_zeros from gamelog_distinct_copy
      where NumOutcome = 0 group by TeamBatting")


sevens <- sqldf("select TeamBatting as Team_batting , count(NumOutcome) as no_of_sevens from gamelog_distinct_cop
y
      where NumOutcome = 7 group by TeamBatting")


fives <- sqldf("select TeamBatting as Team_batting,count(NumOutcome) as no_of_fives from gamelog_distinct_copy
      where NumOutcome = 5 group by TeamBatting")


wickets <- sqldf("select  TeamBatting as Team_batting , count(NumOutcome) as wickets_out from gamelog_distinct_co
py where NumOutcome = -1 group by TeamBatting")




Runs_combined <- sqldf("select sixes.Team_batting ,
                           sixes.no_of_sixes,
                           fours.no_of_fours,
                           threes.no_of_threes,
                           twos.no_of_twos,
                           ones.no_of_ones from sixes
                           inner join fours
                           on sixes.Team_batting = fours.Team_batting
                           inner join threes
                           on sixes.Team_batting = threes.Team_batting
                           inner join twos
                           on sixes.Team_batting = twos.Team_batting
                           inner join ones
                           on sixes.Team_batting = ones.Team_batting ")

head(Runs_combined)
```

```
##   Team_batting no_of_sixes no_of_fours no_of_threes no_of_twos no_of_ones
## 1          AUS         443         877           73        709       3381
## 2           BD         105         377           19        235       1387
## 3          CSK         574        1399           56        920       5279
## 4           DC         215         525           23        357       2168
## 5           DD         358        1039           34        631       3747
## 6          ENG         317         942           55        677       3303
```

```
matches_played <- sqldf("select TeamBatting as Team_batting, TeamBowling as Team_Bowling, MatchNo  from gamelog_d
istinct_copy
       group by MatchNo")

matches_played <- sqldf("select Batsman, sum(NumOutcome) from gamelog_distinct_copy
       group by Batsman")
```

```
##TOP BATSMEN
tb <-  gamelog_distinct_copy %>%
      group_by(Batsman) %>%
      summarise(runs = sum(NumOutcome)) %>%
      arrange(-runs) %>%
      mutate(Batsman = factor(Batsman , levels = Batsman)) %>%
      filter(runs >= 3000)

df2 <- tb %>% slice(-c(2))
head(df2)
```

```
## # A tibble: 6 × 2
##   Batsman         runs
##   <fct>          <dbl>
## 1 CH Gayle        4130
## 2 SK Raina        3499
## 3 V Kohli         3461
## 4 AB de Villiers  3366
## 5 DA Warner       3348
## 6 G Gambhir       3125
```

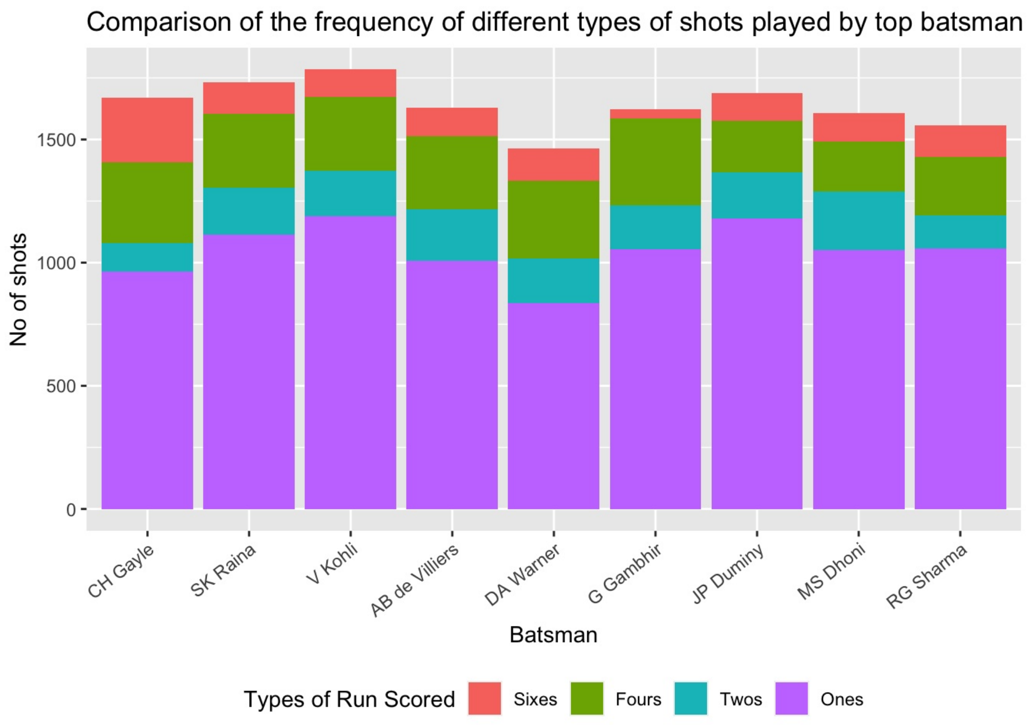
QUESTION 1: PLOT2 - Comparison of the frequency of different types of shots played by the top 7 batsmen who have scored the most number of runs overall in T20 and IPL

```r
#TYPES OF RUNS SCORED (NUMBER OF 4s,6s,3s,2s,and 1s)  ACROSS TOP 10 BATSMAN AND TOTAL RUNS SCORED BY THEM
#NOTE FOR THE SQL PART HAS BEEN REFERENCED FROM ONLINE RESOURCES INCLUDING STACK OVERFLOW AND VARIOUS OTHER BLOG
PAGES
runs_by_best_batsman <- gamelog_distinct_copy %>%
       select(Batsman, NumOutcome) %>%
       filter(NumOutcome %in% c(6,4,3,2,1) & Batsman %in%
           c("CH Gayle",
             "BB McCullum",
             "V Kohli",
             "SK Raina",
             "AB de Villiers",
             "G Gambhir",
             "DA Warner",
             "RG Sharma",
             "JP Duminy",
             "MS Dhoni"))

b_sixes <- sqldf("select Batsman  , count(NumOutcome) as Sixes from runs_by_best_batsman
      where NumOutcome = 6 group by Batsman")


b_fours <- sqldf("select Batsman  , count(NumOutcome) as Fours from runs_by_best_batsman
      where NumOutcome = 4 group by Batsman")


b_threes <- sqldf("select Batsman , count(NumOutcome) as Threes from runs_by_best_batsman
      where NumOutcome = 3 group by Batsman")


b_twos <- sqldf("select Batsman , count(NumOutcome) as Twos from runs_by_best_batsman
      where NumOutcome = 2 group by Batsman")


b_ones <- sqldf("select Batsman  , count(NumOutcome) as Ones from runs_by_best_batsman
      where NumOutcome = 1 group by Batsman")



Runs_Type_Batsmen <- sqldf("select b_sixes.Batsman ,
                           b_sixes.Sixes,
                           b_fours.Fours,
                           b_threes.Threes,
                           b_twos.Twos,
                           b_ones.Ones from b_sixes
                           inner join b_fours
                           on b_sixes.Batsman = b_fours.Batsman
                           inner join b_threes
                           on b_sixes.Batsman = b_threes.Batsman
                           inner join b_twos
                           on b_sixes.Batsman = b_twos.Batsman
                           inner join b_ones
                           on b_sixes.Batsman = b_ones.Batsman
                           "
                           )
merged<- merge(df2,Runs_Type_Batsmen, by ="Batsman")
head(merged)
```

```
##          Batsman runs Sixes Fours Threes Twos Ones
## 1 AB de Villiers 3366   117   296     17  208 1008
## 2       CH Gayle 4130   262   328      6  114  965
## 3      DA Warner 3348   133   315     24  181  836
## 4      G Gambhir 3125    39   351     17  178 1055
## 5      JP Duminy 3098   111   209     12  188 1179
## 6       MS Dhoni 3072   113   203     14  240 1050
```

```r
suppressPackageStartupMessages(library(reshape))
library(reshape, quietly = T)

melt(merged) %>%
              filter(variable %in% c("Sixes","Fours","Twos","Ones","Zeros")) %>%
                ggplot(aes(Batsman, value ,fill = variable))+
                geom_bar(stat ="identity") +
                labs(fill="Types of Run Scored",y="No of shots",x="Batsman",title ="Comparison of the frequency o
f different types of shots played by top batsman ")+
                theme(axis.text.x = element_text(angle = 38, hjust = 1)) +
                theme(legend.position="bottom")
```

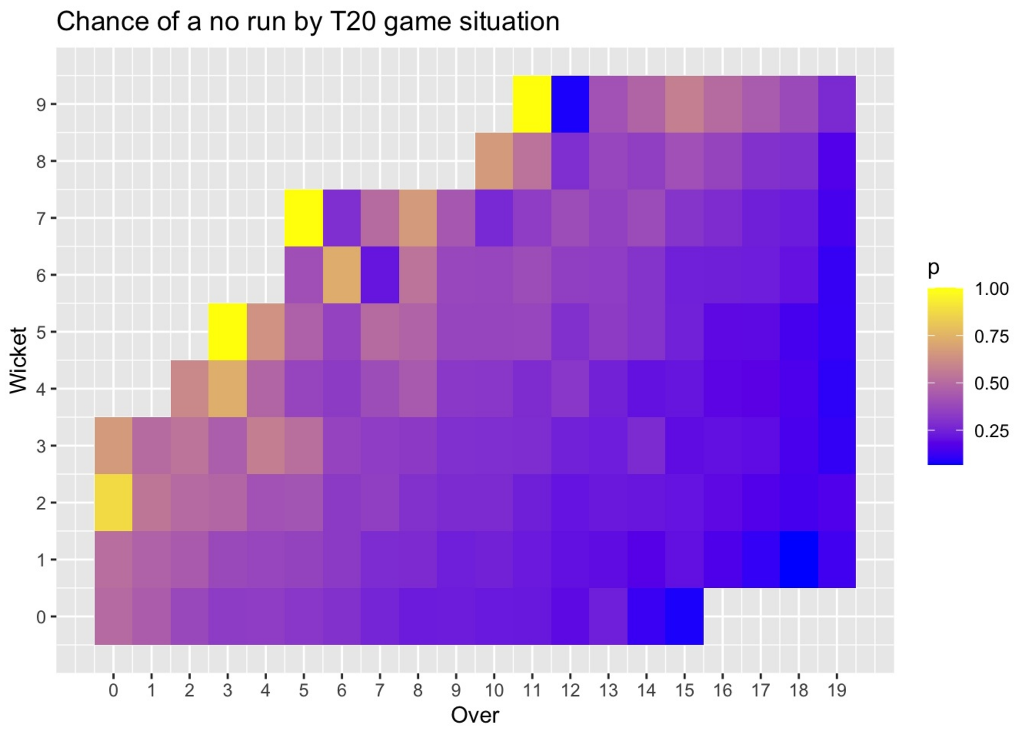
```
## Using Batsman as id variables
```



From the barplot it is clear that Chris Gayle has scored the highest number of sixes overall, followed David Warner and SK Raina. Gautam Gambhir seems to have scored the highest number of fours in T20 cricket, followed by Chris Gayle and David Warner. We can also observe that Chris Gayle followed by SK Raina and Virat Kohil have scored the maximum number of runs overall.

QUESTION 1: PLOT3 - Heatmap representing the chance/ probability of hitting a dot ball

```
# Show the chance of a no run
p0df <- gamelog_distinct %>% filter(Format == "T20I") %>% group_by(Over, Wickets) %>% summarise(p = sum(NumOutcom
e == 0) / n())
ggplot(p0df, aes(x = Over, y = Wickets, fill = p)) + geom_tile() + scale_fill_gradient(low ="blue", high = "yello
w", na.value="white") + labs(title="Chance of a no run by T20 game situation", x = "Over", y = "Wicket")+scale_x_
continuous(breaks= seq(0,19,by=1))+ scale_y_continuous(breaks= seq(0,9,by=1))
```

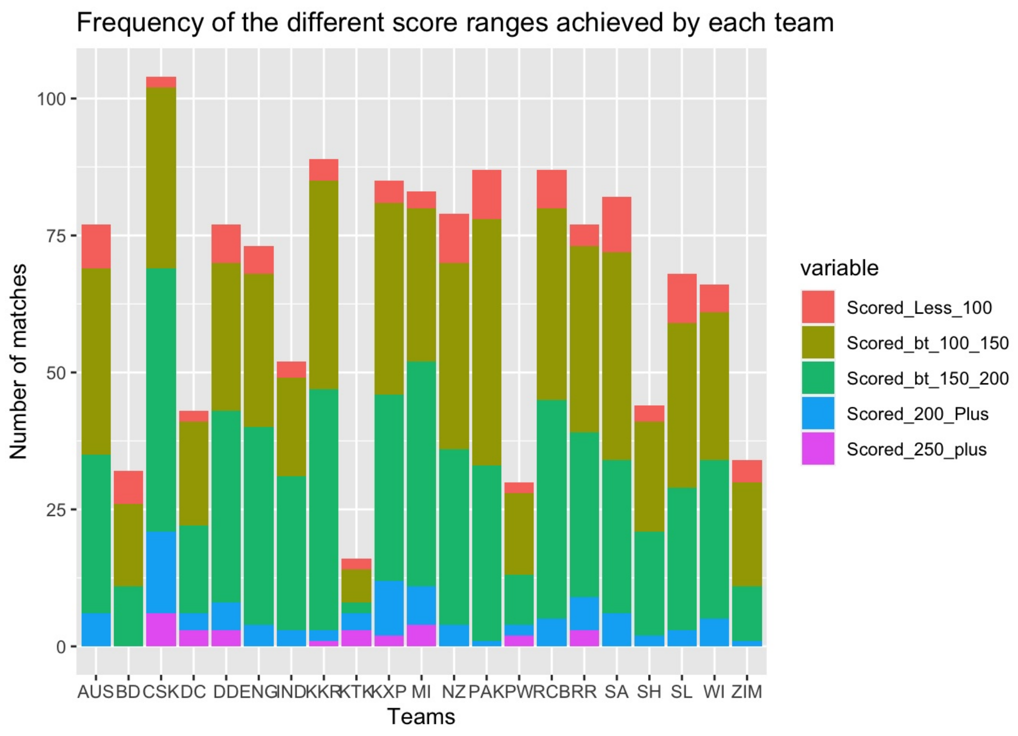


We can see that the probability of hitting a dot ball is high on the 9th,7th,5th,2nd wickets and in the 11th,5th,3rd and 0th over. On the whole the probability of getting a dot ball seems quite high on the first 10 overs and seems to be nearly zero later on. The main reason for this pattern, could be because the bowlers in the first few overs are usually fast bowlers who are good at swinging a dot ball i.e. balls which are hard to hit or may result in a wicket easily. This is usually done in order to control the run rate. On the other hand, the probability of hitting a dot ball seems to very less in the last 4 overs.

```
#NOTE FOR THE SQL PART HAS BEEN REFERENCED FROM ONLINE RESOURCES INCLUDING STACK OVERFLOW AND VARIOUS OTHER BLOG
PAGES
runs_1 <-   gamelog_distinct_copy %>%
       group_by(MatchNo,TeamBatting) %>%
       summarise(runs = sum(NumOutcome)) %>%
       filter(runs >= 200)

dd <- sqldf(" select TeamBatting as Team, count(runs) as Scored_200_Plus from runs_1
        group by  TeamBatting")

runs_2 <-   gamelog_distinct_copy %>%
       group_by(MatchNo,TeamBatting) %>%
       summarise(runs = sum(NumOutcome)) %>%
       filter(runs <= 100)


a <- sqldf(" select TeamBatting as Team, count(runs) as Scored_Less_100 from runs_2
               group by TeamBatting ")


runs_3 <- gamelog_distinct_copy %>%
      group_by(MatchNo,TeamBatting) %>%
      summarise(runs = sum(NumOutcome)) %>%
      filter(runs %in% c(100:150))



runs_4 <- gamelog_distinct_copy %>%
      group_by(MatchNo,TeamBatting) %>%
      summarise(runs = sum(NumOutcome)) %>%
      filter(runs <=150 & runs >=100)



b <- sqldf(" select TeamBatting as Team, count(runs) as Scored_bt_100_150 from runs_3
               group by TeamBatting ")

runs_4 <- gamelog_distinct_copy %>%
      group_by(MatchNo,TeamBatting) %>%
      summarise(runs = sum(NumOutcome)) %>%
      filter(runs %in% c(150:200))

c <- sqldf(" select TeamBatting as Team, count(runs) as Scored_bt_150_200 from runs_4
               group by TeamBatting ")


runs_5 <- gamelog_distinct_copy %>%
      group_by(MatchNo,TeamBatting, TeamBowling) %>%
      summarise(runs = sum(NumOutcome)) %>%
      filter(runs >= 250)



e <- sqldf(" select TeamBatting as Team, count(runs) as Scored_250_plus from runs_5
               group by TeamBatting ")

score_table<- sqldf("select b.Team,
              a.Scored_Less_100 ,
              b.Scored_bt_100_150 ,
              c.Scored_bt_150_200,
              dd.Scored_200_Plus,
              e.Scored_250_plus
              from b
              left join a
              on a.Team = b.Team
              left join c
              on b.Team = c.Team
              left join dd
              on b.Team = dd.Team
              left join e
              on b.Team = e.Team")
```

QUESTION 1: PLOT 4: Frequency of the different score ranges achieved by each team

```
#TEAM SCORES FREQUENCY ACROSS TEAMS AND HIGHEST SCORE ACHIEVED BY EACH TEAM
melt(score_table) %>%
     filter(variable %in% c("Scored_Less_100","Scored_bt_100_150","Scored_bt_150_200","Scored_200_Plus","Scored_2
50_plus"))%>%
      ggplot(aes(Team,value,fill = variable))+
      geom_bar(stat ="identity") +
      labs(x="Teams",y="Number of matches", title="Frequency of the different score ranges achieved by each team"
)
```



```
      theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

We can observe that CSK, DC, KTK and PW has scored less than 100 only twice in IPL games while SouthAfrica has scored less than 100 10 times followed by NewZealand and Pakistan. KKR in IPL games and the Pakistan team in T20I League have scored between 100-150 the maximum number of times. CSK has scored the highest number of times (48), between 150 and 200 runs.

The Bangaldesh team seems to have never scored more than 200 while CSK in IPL games seems to have scored more than 200 and 250 the highest number of times compared to other teams. KXP also in IPL games seems to be the second best after CSK in the above stated score ranges. It is also important to note that while many teams have scored more than 250 in the IPL league, no team has scored more than 250 in the T20I cricket.

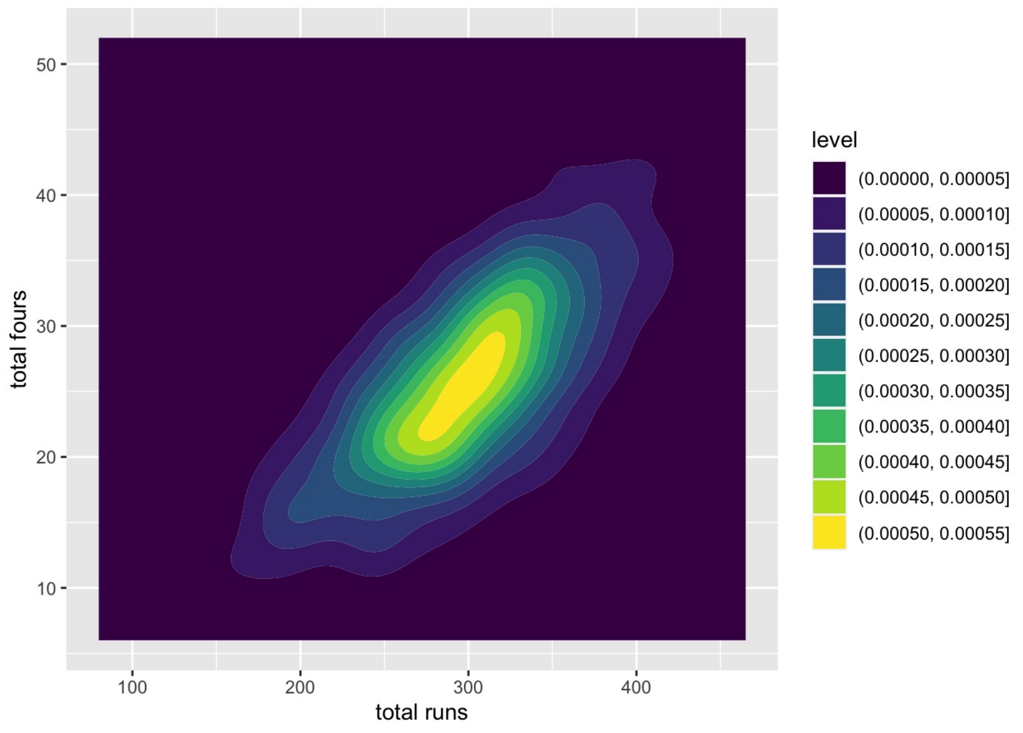
QUESTION 3: Kmeans clustering for total runs, total balls and total fours

```
library(qpcR,quietly=T)
library("dplyr")
```

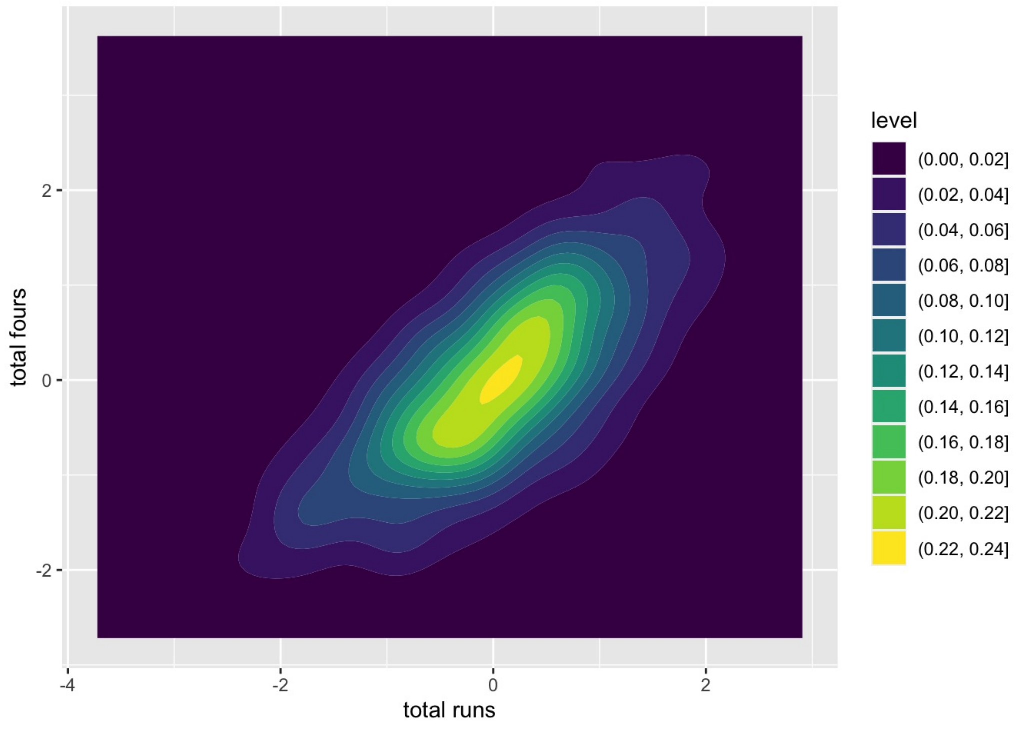
```r
df_kmeans = ddply(gamelog_distinct, "MatchNo", summarize,
                  total_runs = sum( pmax(NumOutcome, 0)),
                  total_sixes =  length(which(NumOutcome == 6)),
                  total_fours =  length(which(NumOutcome == 4)),
                  total_threes = length(which(NumOutcome == 3)),
                  total_twos = length(which(NumOutcome == 2)),
                  total_ones = length(which(NumOutcome == 1)),
                  dot_balls = length(which(NumOutcome == 0)),
                  runs_1stinning = sum(pmax(NumOutcome*1*(Inning == 1), 0)),
                  runs_2ndinning = sum(pmax(NumOutcome*1*(Inning == 2), 0)),
                  total_wickets = length(which(NumOutcome == -1) ),
                  fielder_mentions = length(which(Fielder != "")),
                  balls_until_1st_wicket = length(which(Wickets == 0)),
                  average_wickets_in_during_match = mean(Wickets, na.rm=TRUE),
                  total_balls = sum(pmax(length(BallType), 0))
          )

df_kmeans <-drop_na(df_kmeans)

#FINDING NUMBER OF BALLS IN SECOND INNING:
second_innings <- gamelog_distinct %>% filter( Inning == 2 )
second_inning_balls_df = ddply(second_innings, "MatchNo", summarize,
                  balls_2nd_inning = sum(pmax(length(Ball), 0)))

df_kmeans<-merge(df_kmeans,second_inning_balls_df,by = "MatchNo")


#FINDING NUMBER OF BALLS IN FIRST INNING:
first_innings <- gamelog_distinct %>% filter( Inning == 1 )
first_inning_balls_df = ddply(first_innings, "MatchNo", summarize,
                  balls_1st_inning = sum(pmax(length(Ball), 0)))

df_kmeans<-merge(df_kmeans,first_inning_balls_df,by = "MatchNo")

#Standardizing the columns so that they are equally represented in proportion
df_kmeans$total_runs_scaled = (df_kmeans$total_runs - mean(df_kmeans$total_runs)) / sd(df_kmeans$total_runs)
df_kmeans$total_fours_scaled = (df_kmeans$total_fours - mean(df_kmeans$total_fours)) / sd(df_kmeans$total_fours)
df_kmeans$total_balls_scaled = (df_kmeans$total_balls - mean(df_kmeans$total_balls)) / sd(df_kmeans$total_balls)
```

Density plot showing the number of fours that made up the sum of the total runs scored by both teams

```r
library(ggplot2, quietly = T)
gr2 <- ggplot(df_kmeans, aes(x = total_runs,  y = total_fours)) +
   geom_density2d_filled() +
  xlab("total runs") +
  ylab("total fours")

gr2
```



```
gr2_scaled <- ggplot(df_kmeans, aes(x = total_runs_scaled,  y = total_fours_scaled)) +
   geom_density2d_filled() +
  xlab("total runs") +
  ylab("total fours")

gr2_scaled
```

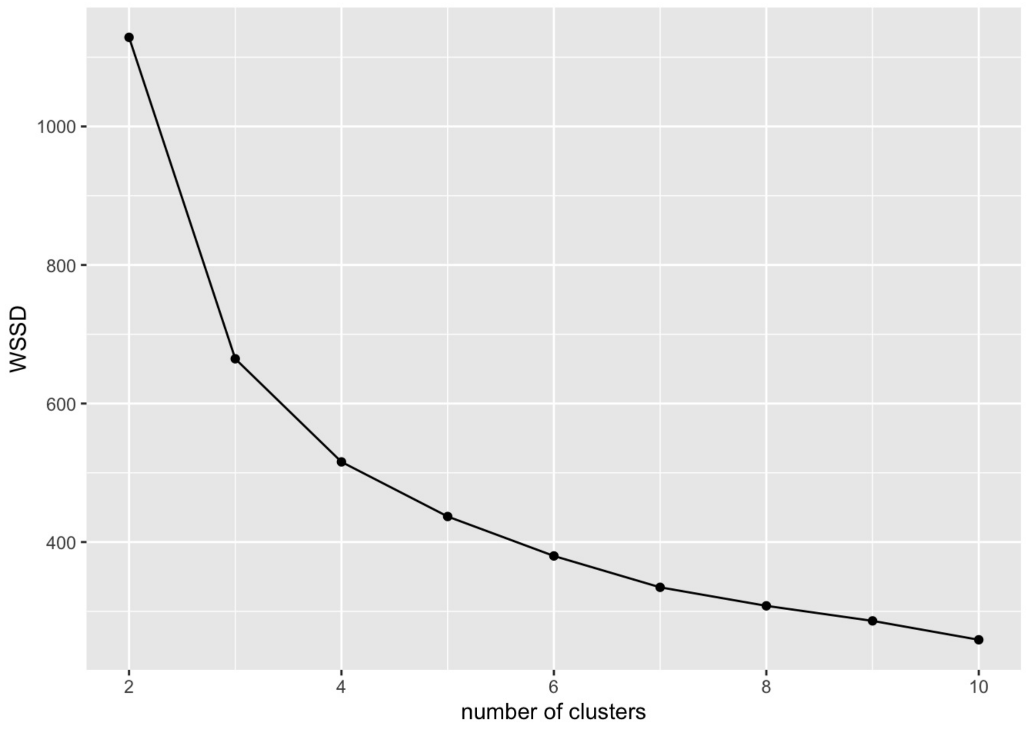


The above contour plot gives a clear picture of locations of high and low shot density clearly. The dominant (yellow) density location reveals that matches where the total runs scored in both innings summed upto 270-320, had approximately 20-23 fours that contributed to the overall total score i.e. 20-25 is the number of fours scored together by both teams in that particular match. In matches which had a total score of 350-400, the number of fours scored seem to be around 30. Those matches which had 150-200 total runs roughly had about 15 fours. Those matches having similar runs and fours are present/ clustered together in one of the 3 clusters.

Determining the optimal number of clusters required:

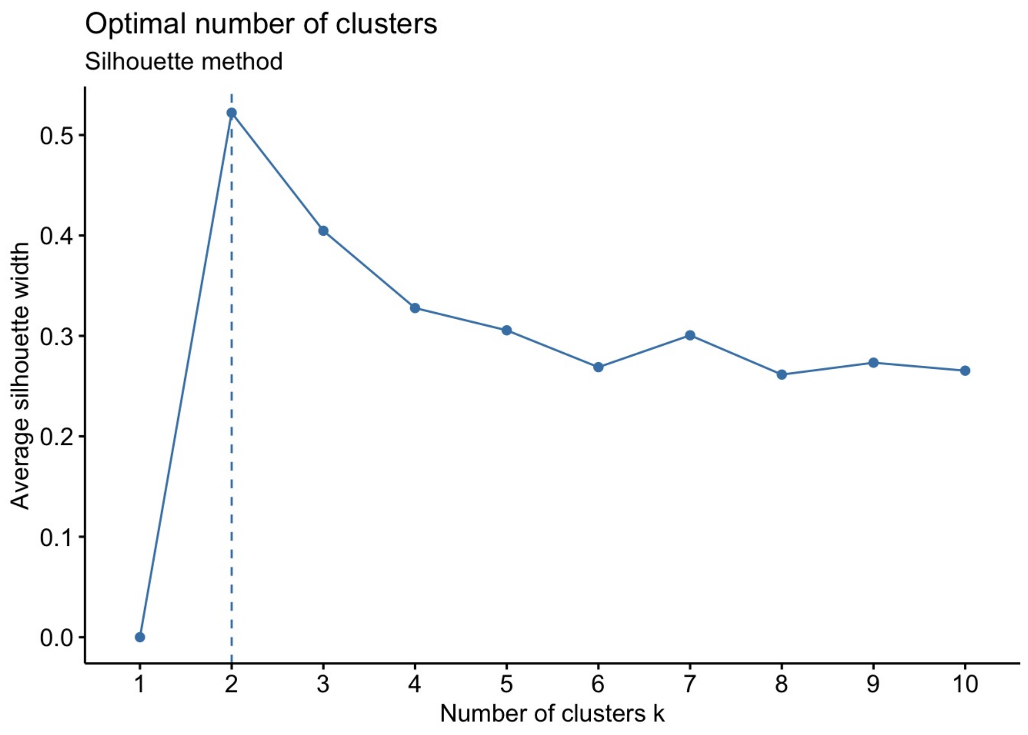
METHOD 1: Elbow Plot (fviz_nbclust() function)

```
library(factoextra, quietly= T)
library(NbClust, quietly= T)
suppressPackageStartupMessages(library(factoextra))
suppressPackageStartupMessages(library(NbClust))



df_selected = subset(df_kmeans, select = c(total_runs_scaled, total_fours_scaled, total_balls_scaled))

wssd <- rep(NA,9)

for(k in 2:10)
{
   selected_clust <- kmeans(df_selected, centers = k)
   wssd[k-1] <- selected_clust$tot.withinss
}

centers <- 2:10
dat <- data.frame(centers, wssd)

gr3 <- ggplot(dat, aes(x=centers, y=wssd)) +
        geom_line() +
        geom_point() +
        xlab("number of clusters") +
        ylab("WSSD")
plot(gr3)
```



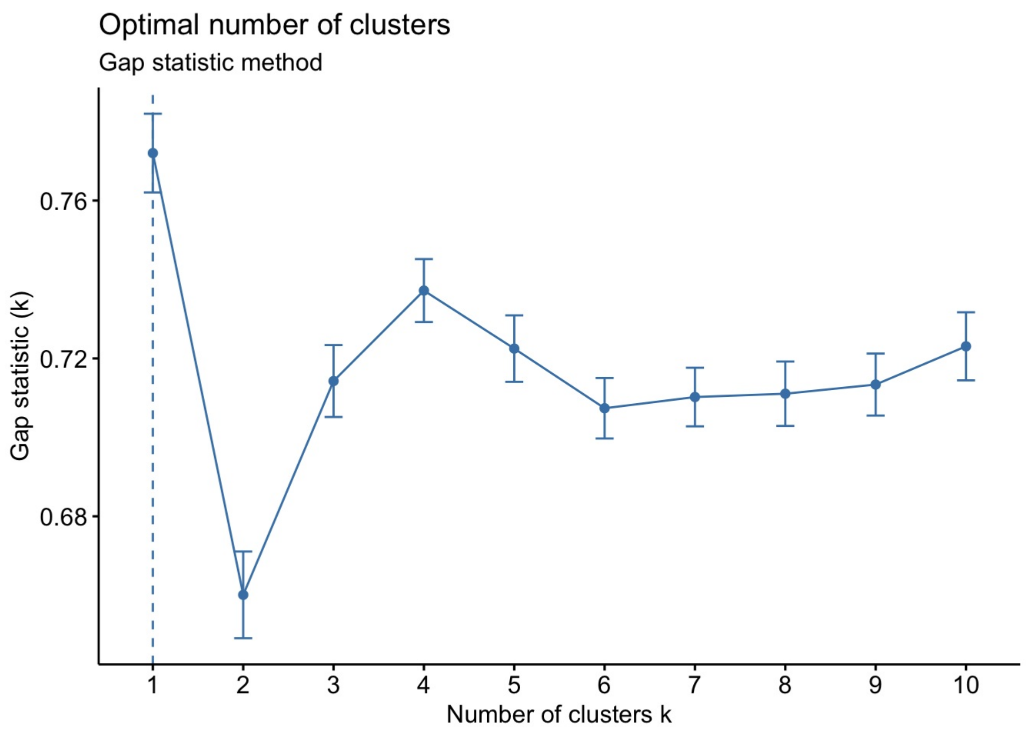
METHOD 2: Average Silhouette Method (fviz_nbclust() function)

```
set.seed(12345)

fviz_nbclust(df_selected, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



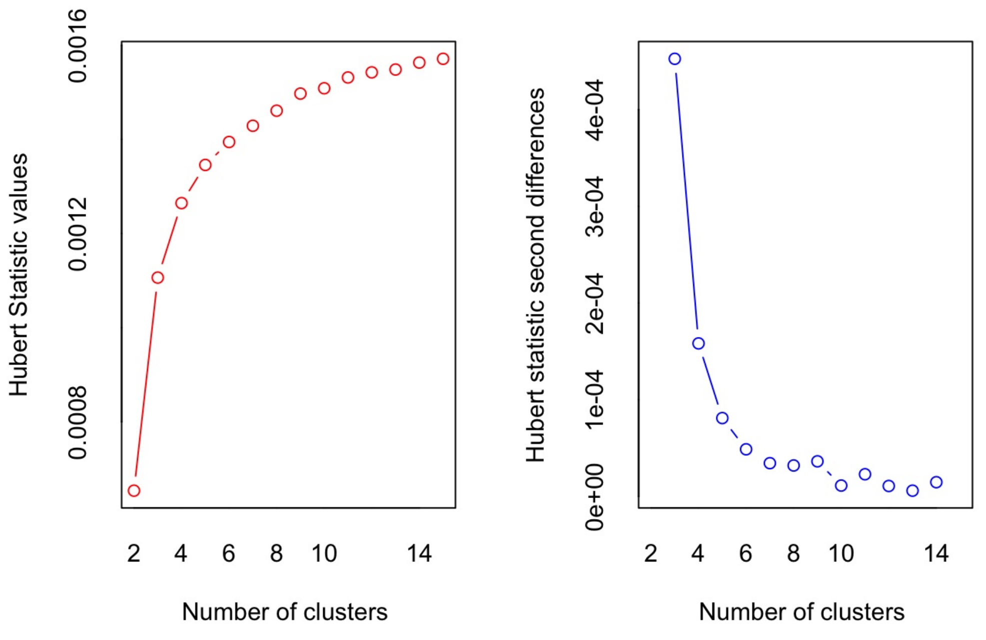
METHOD 3: Gap Statistic Method (fviz_nbclust() function)

```
fviz_nbclust(df_selected, kmeans, nstart = 10,  method = "gap_stat", nboot = 25)+
  labs(subtitle = "Gap statistic method")
```

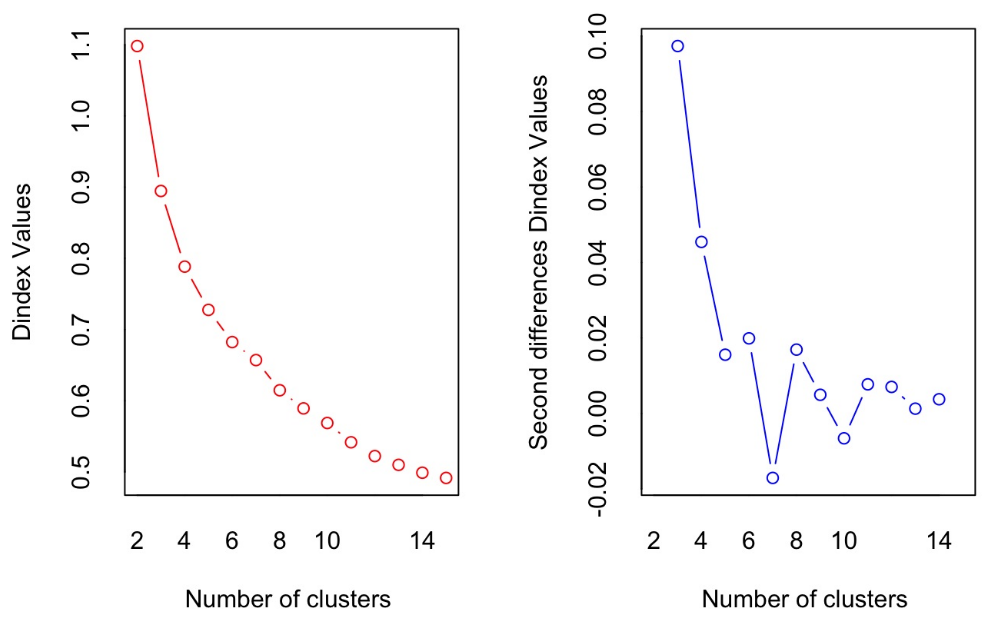


METHOD 4: NbClust() function

```
NbClust(data = df_selected, diss = NULL, distance ="euclidean",
        min.nc = 2, max.nc = 15, method = "kmeans")
```



```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 2 proposed 2 as the best number of clusters
## * 15 proposed 3 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
## * 1 proposed 11 as the best number of clusters
## * 1 proposed 12 as the best number of clusters
## * 1 proposed 15 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

```
## $All.index
##         KL       CH Hartigan     CCC    Scott   Marriot    TrCovW    TraceW
## 2   0.3213 450.4794 446.8392 -9.9780  565.845 110007509 76247.870 1128.6046
## 3   4.9730 604.9718 184.0769 -3.4355 1373.877  70306402 30174.869  664.5941
## 4   2.0412 579.9473 126.0189 -2.1334 1733.965  71331750 24176.428  515.9611
## 5   7.6850 551.5136  85.3351 -1.5254 2107.312  62308082 17950.637  430.8574
## 6   0.3831 516.5720  61.2284 -1.5599 2283.825  68155548 12772.856  379.9569
## 7   0.2797 481.3658  86.9873 -2.4069 2483.774  67941489 10787.890  346.5902
## 8  12.3066 480.7888  54.0342 -0.9895 2689.935  64366028  9752.543  304.8319
## 9   0.5085 462.5681  41.7185 -0.9739 2840.695  64413540  8032.986  280.8922
## 10  0.4043 442.2060  52.2836 -1.3170 2977.611  64249896  6472.460  263.5244
## 11  2.9590 435.4479  34.0731 -0.8421 3135.375  60804379  5295.946  243.3894
## 12  0.3719 419.6692  25.0170 -1.1466 3216.290  63793329  5387.475  230.9201
## 13  2.5654 401.4194  28.1888 -1.7677 3300.098  65706133  4615.641  222.1006
## 14  0.3759 388.6963  19.9917 -2.0674 3378.107  67484539  4100.218  212.5740
## 15  1.0124 373.2549  37.1106 -2.6840 3450.083  69253422  3717.175  206.0157
##    Friedman  Rubin Cindex     DB Silhouette   Duda Pseudot2   Beale Ratkowsky
## 2    3.7186 1.7039 0.2002 1.1914     0.3958 0.6237 225.0271  1.0227    0.4464
## 3    7.1108 2.8935 0.2226 0.9034     0.4047 1.0357 -15.9585 -0.0585    0.4668
## 4    8.9964 3.7270 0.2141 1.0222     0.3277 0.9350  14.9427  0.1177    0.4277
## 5   11.8848 4.4632 0.2329 1.0580     0.3204 1.4059 -81.7043 -0.4889    0.3938
## 6   12.7881 5.0611 0.2029 1.0440     0.3213 1.4909 -99.1118 -0.5578    0.3657
## 7   14.7092 5.5483 0.2289 1.1236     0.2790 1.2011 -44.3691 -0.2835    0.3422
## 8   16.4332 6.3084 0.2118 1.0609     0.2911 1.3771 -55.8587 -0.4624    0.3243
## 9   17.9428 6.8460 0.2043 1.1479     0.2717 1.7967 -81.1460 -0.7463    0.3080
## 10  19.5758 7.2972 0.1940 1.1369     0.2816 1.2435 -21.5417 -0.3277    0.2938
## 11  21.5601 7.9009 0.1834 1.0848     0.2819 3.0899 -48.0222 -1.1176    0.2818
## 12  22.4330 8.3276 0.1772 1.1047     0.2680 1.7733 -25.7292 -0.7177    0.2708
## 13  23.6564 8.6582 0.1867 1.1212     0.2613 1.8443 -41.6578 -0.7542    0.2608
## 14  24.6641 9.0463 0.1898 1.1280     0.2476 2.4519 -46.1883 -0.9894    0.2521
## 15  25.8555 9.3342 0.1863 1.1048     0.2594 3.1047 -57.6224 -1.1129    0.2440
##        Ball Ptbiserial    Frey McClain   Dunn Hubert SDindex Dindex   SDbw
## 2  564.3023     0.4261 -0.0503  0.4852 0.0094 0.0007  4.4936 1.0981 1.5999
## 3  221.5314     0.5211  1.1159  0.5715 0.0159 0.0011  3.7232 0.8946 0.9908
## 4  128.9903     0.4823  0.5498  0.8989 0.0212 0.0013  3.6366 0.7883 0.6973
## 5   86.1715     0.4757  0.7568  1.0303 0.0158 0.0013  3.4730 0.7276 0.5658
## 6   63.3261     0.4546  9.9592  1.2248 0.0184 0.0014  3.5915 0.6824 0.5573
## 7   49.5129     0.4085  0.3277  1.5686 0.0275 0.0014  4.3098 0.6571 0.5490
## 8   38.1040     0.4034  1.6890  1.6738 0.0183 0.0015  4.0159 0.6147 0.4152
## 9   31.2102     0.3680  0.0077  2.0733 0.0251 0.0015  4.4317 0.5893 0.4097
## 10  26.3524     0.3734  0.8549  2.0330 0.0210 0.0015  4.7469 0.5688 0.3525
## 11  22.1263     0.3465  0.8572  2.4085 0.0171 0.0015  4.6763 0.5417 0.3478
## 12  19.2433     0.3287  0.6577  2.7019 0.0171 0.0015  4.8955 0.5223 0.2744
## 13  17.0847     0.3178  1.1190  2.9039 0.0227 0.0015  5.3285 0.5100 0.2991
## 14  15.1839     0.3054 -0.0117  3.1581 0.0125 0.0016  5.9524 0.4989 0.2575
## 15  13.7344     0.3069  0.5259  3.1222 0.0256 0.0016  5.1487 0.4916 0.2192
##
## $All.CriticalValues
##    CritValue_Duda CritValue_PseudoT2 Fvalue_Beale
## 2          0.6431           206.9947       0.3819
## 3          0.6651           233.1283       1.0000
## 4          0.6271           127.8344       0.9497
## 5          0.6254           169.4860       1.0000
## 6          0.6307           176.2135       1.0000
## 7          0.6219           161.1242       1.0000
## 8          0.5889           142.3826       1.0000
## 9          0.5512           149.0067       1.0000
## 10         0.4996           110.1614       1.0000
## 11         0.4105           101.9584       1.0000
## 12         0.3869            93.4940       1.0000
## 13         0.3932           140.4261       1.0000
## 14         0.4868            82.2235       1.0000
## 15         0.3733           142.6946       1.0000
##
## $Best.nc
##                      KL       CH Hartigan     CCC    Scott  Marriot TrCovW
## Number_clusters  8.0000   3.0000   3.0000 11.0000   3.0000        3      3
## Value_Index     12.3066 604.9718 262.7623 -0.8421 808.0315 40726455  46073
##                   TraceW Friedman   Rubin  Cindex     DB Silhouette   Duda
## Number_clusters   3.0000   3.0000  3.0000 12.0000 3.0000     3.0000 3.0000
## Value_Index     315.3774   3.3922 -0.3561  0.1772 0.9034     0.4047 1.0357
##                 PseudoT2  Beale Ratkowsky     Ball PtBiserial Frey McClain
## Number_clusters   3.0000 2.0000    3.0000   3.0000     3.0000    1  2.0000
## Value_Index     -15.9585 1.0227    0.4668 342.7709     0.5211   NA  0.4852
##                   Dunn Hubert SDindex Dindex    SDbw
## Number_clusters 7.0000      0   5.000      0 15.0000
## Value_Index     0.0275      0   3.473      0  0.2192
##
## $Best.partition
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   1   1   2   1   2   2   2   1   3   1   2   1   2   2   2   2   1   2   2
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   1   1   1   1   2   1   1   1   1   2   1   1   2   3   1   1   1   2   1   2
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   2   3   1   3   2   2   1   1   3   1   1   1   1   3   1   2   1   1   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   1   1   1   1   2   1   1   2   1   2   2   1   2   1   1   2   2   3   1   3
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   1   2   1   1   1   1   3   1   1   1   1   1   2   2   2   3   1   1   1
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   2   1   1   2   1   1   2   1   2   3   3   3   1   1   1   1   2   1   1   1
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   1   1   1   2   1   1   2   1   1   1   1   3   1   1   2   2   1   3   2   2
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
##   1   1   1   1   1   1   1   2   1   1   1   1   2   2   2   1   1   1   1   1
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   1   1   1   2   1   1   1   1   1   1   2   1   2   2   1   1   1   1   3   1
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
##   1   1   1   2   1   1   3   1   1   1   3   2   3   1   3   2   3   1   2   1
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
##   2   2   2   1   1   1   1   1   1   1   2   1   1   1   2   1   3   2   1   3
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
##   2   2   1   2   2   1   1   2   2   2   1   1   1   1   2   2   2   2   2   1
## 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
##   1   1   2   1   2   2   2   2   2   1   1   3   1   2   1   2   1   2   2   1
## 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280
##   2   1   2   1   3   2   1   1   2   3   2   1   2   1   2   2   1   2   2   1
## 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
##   1   1   1   2   1   2   1   1   2   1   1   1   1   1   1   2   2   1   2   2
## 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
##   2   2   1   1   1   1   1   2   2   2   2   1   1   2   1   1   3   1   2   2
## 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
##   1   2   1   1   2   1   1   1   1   1   1   1   2   2   1   1   2   2   1   2
## 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
##   1   1   1   1   1   2   1   1   1   1   1   1   2   1   1   2   1   2   1   1
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380
##   2   1   1   2   2   2   2   1   2   2   2   1   2   2   2   2   2   1   1   1
## 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400
##   1   2   1   1   1   2   2   2   2   1   2   3   2   1   2   2   2   3   1   3
## 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420
##   1   2   1   1   1   2   3   2   2   2   2   1   1   1   3   1   1   1   1   1
## 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440
##   2   1   1   1   1   2   1   2   2   1   1   2   1   1   2   1   2   1   2   2
## 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460
##   1   2   1   1   2   2   2   2   2   1   2   1   2   1   2   2   2   1   1   2
## 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480
##   2   1   1   2   2   2   1   2   1   1   1   3   2   2   2   1   2   1   2   2
## 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500
##   1   1   2   3   1   1   1   1   1   2   2   1   2   2   2   2   2   2   2   1
## 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520
##   1   1   2   2   1   3   1   2   2   2   2   1   2   1   1   1   1   2   2   1
## 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540
##   2   2   1   3   2   1   1   1   2   1   1   2   2   1   1   1   1   2   1   3
## 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560
##   1   1   1   1   2   2   1   2   1   1   1   2   2   2   2   1   2   3   1   2
## 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580
##   2   2   1   2   2   1   2   1   1   1   2   1   1   2   2   2   2   2   1   2
## 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600
##   1   2   1   2   2   2   2   2   1   2   2   2   2   2   2   2   1   2   1   2
## 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620
##   2   2   2   3   1   2   2   1   2   1   3   1   1   2   2   1   2   3   2   2
## 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640
##   1   2   2   2   2   2   1   2   2   2   1   2   1   3   2   3   1   2   3   1
## 641 642
##   1   2
```

Therefore,

1.in elbow plot we find no of optimal clusters = 5

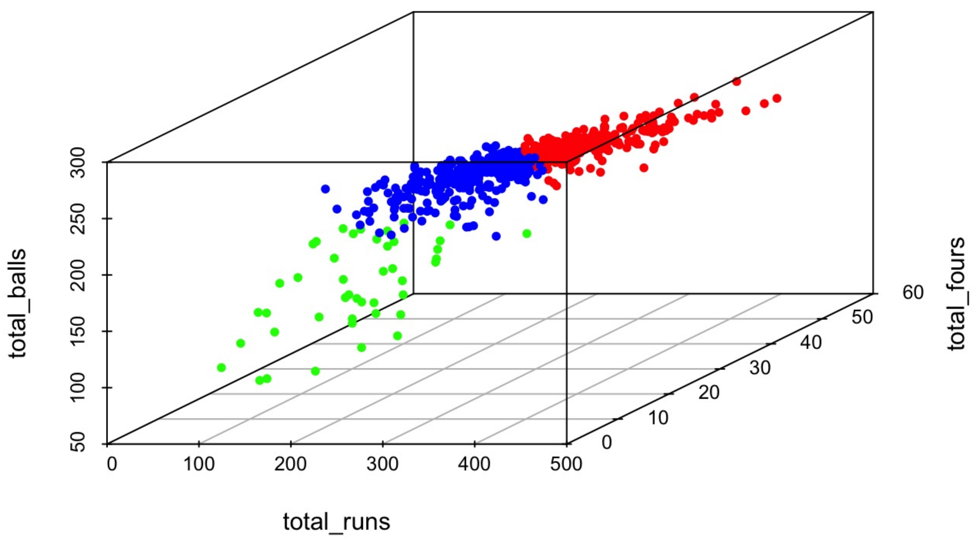
2.in Average Silhouette Method we find no of optimal clusters = 3

3.in Gap Statistic Method we find no of optimal clusters = 1

Verifying above results using NbClust() method, we can conclude that 5 clusters are sufficient and optimal.

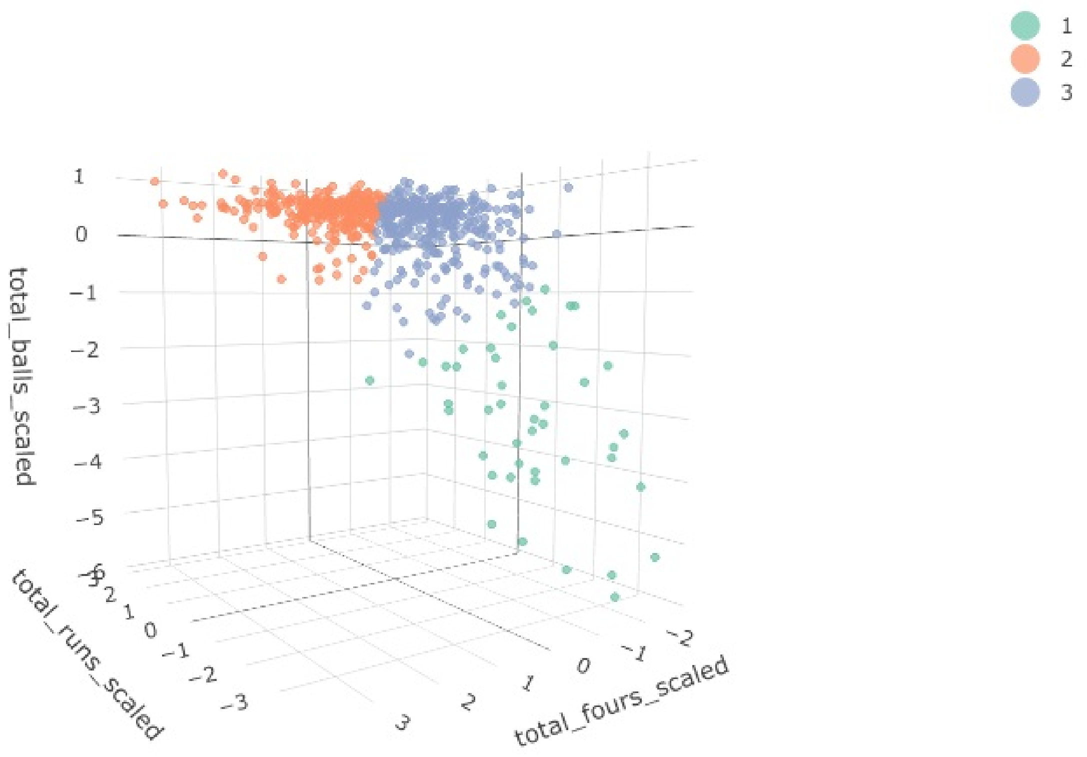
By using majority rule, we can conclude that the optimal number of clusters required =3

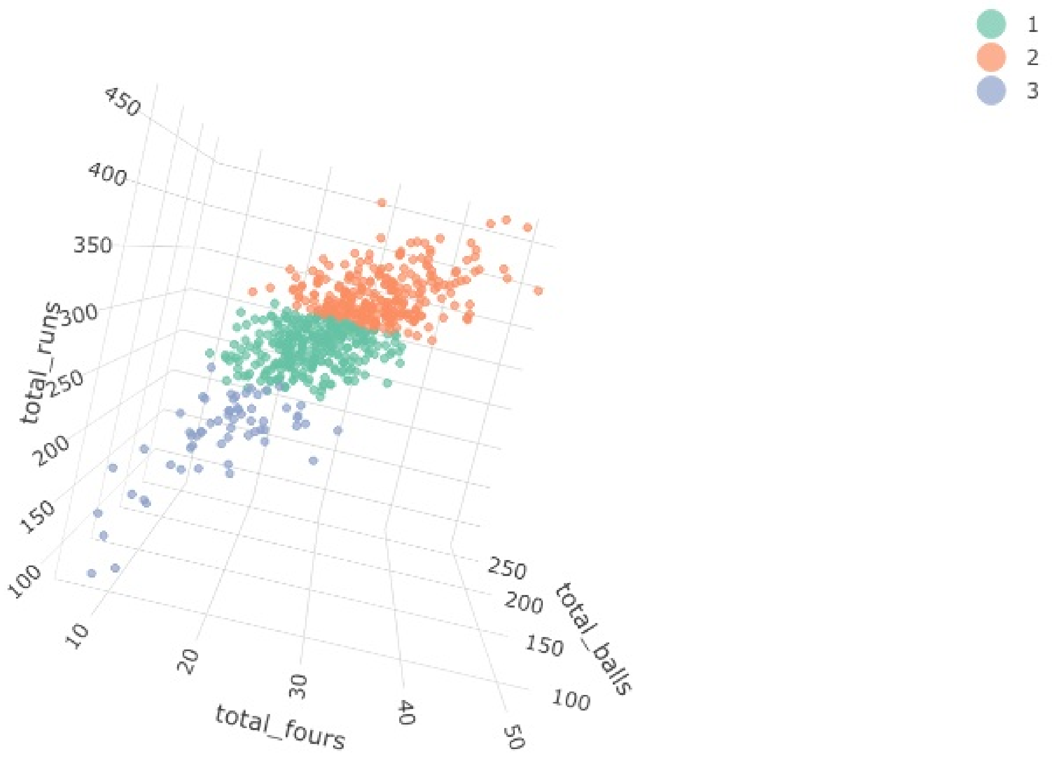
```r
#RGB - RED(x-axis), GREEN(y-axis), BLUE(z-axis)
library(scatterplot3d, quietly = T)
df = df_kmeans[c(18,19,20)]
df$cluster = factor(kmeans(df,3)$cluster)
scatterplot3d(df_kmeans[c(2,4,15)], pch=20, color=rainbow(3)[df$cluster])
```



```
df = df_kmeans[c(18,19,20)]
df$cluster = factor(kmeans(df,3)$cluster)

library(plotly, quietly = T)
library(dplyr)
p <- plot_ly(df, x=~total_runs_scaled, y=~total_fours_scaled,
z=~total_balls_scaled, color=~cluster) %>%
     add_markers(size=1.5)
p
```

The 3d plots below show how the five clusters look in the 3-d space. They have been plotted for both the standardized and unstandardized variables for better understanding.





CLASSIFICATION OF THE CLUSTERS INTO MEANINGFUL ARCHETYPES:

```
set.seed(12345) # For reproducibility
df_selected = subset(df_kmeans, select = c(total_runs_scaled, total_fours_scaled, total_balls_scaled))
selected_clust_3 <- kmeans(df_selected, centers = 3)
selected_clust_3$centers
```

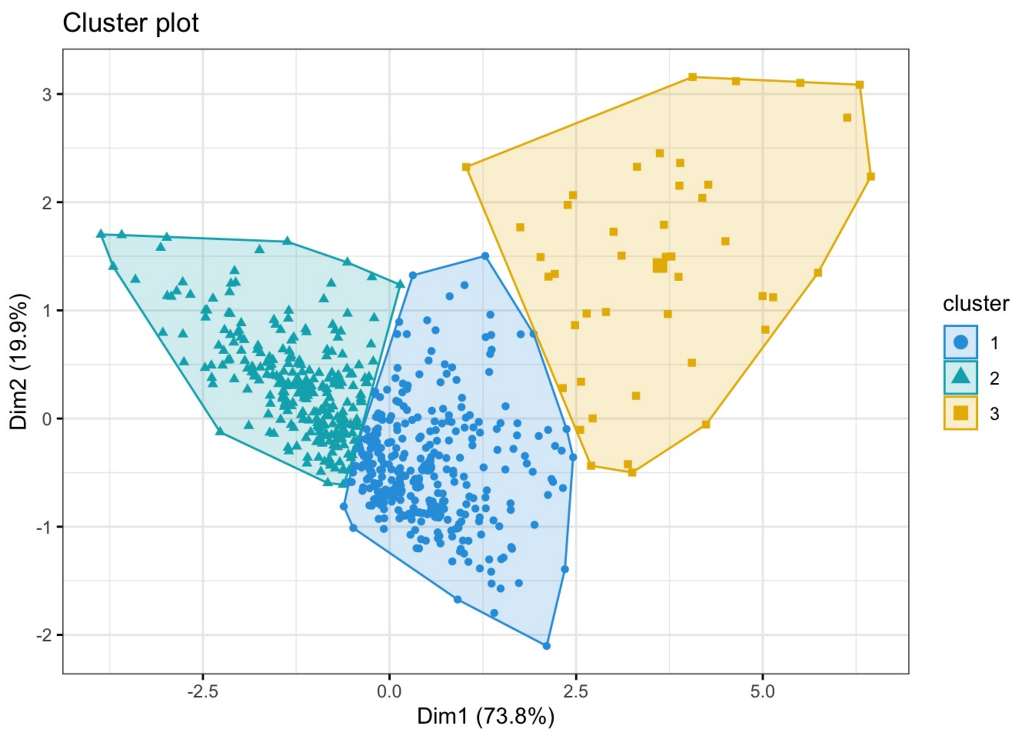
```
##   total_runs_scaled total_fours_scaled total_balls_scaled
## 1         0.8336459          0.8908261          0.3774195
## 2        -0.4127019         -0.5368112          0.1049075
## 3        -1.9540710         -1.3674105         -3.0794403
```

We can classify the 3 clusters as follows:

1. CLUSTER 1: BATSMAN DOMINATED MATCHES (LONGEST RUNNING AND MOST EXCITING MATCHES) - These matches had the most number of balls bowled, hence “longest running matches” duration wise. They also had the highest number of runs overall as well as fours. This means that the batsmen who played in these matches were extremely good and it was difficult for the bowlers to take their wickets (make them out) or make them score lesser number of runs.
2. CLUSTER 2: LOW RISK TAKING MATCHES BECAUSE OF BATSMEN PLAYING AT THE BACK FRONT - This cluster corresponds to matches having a low number of runs and fours, but having a good amount of balls bowled. This indicates that the batsman were playing at the back front and as a result were not able to score well and the bowlers weren’t able to take a wicket that easily as well since the batsman did not risk hitting big runs.
3. CLUSTER 3: BOWLER DOMINATED MATCHES - The total runs, fours and balls are all negative i.e. lesser than those present on an average on most matches. This indicates that the bowlers must have been so good that the batters couldn’t score many runs and were eventually out easily.

Visualising the 3 clusters, Total runs vs Total fours:

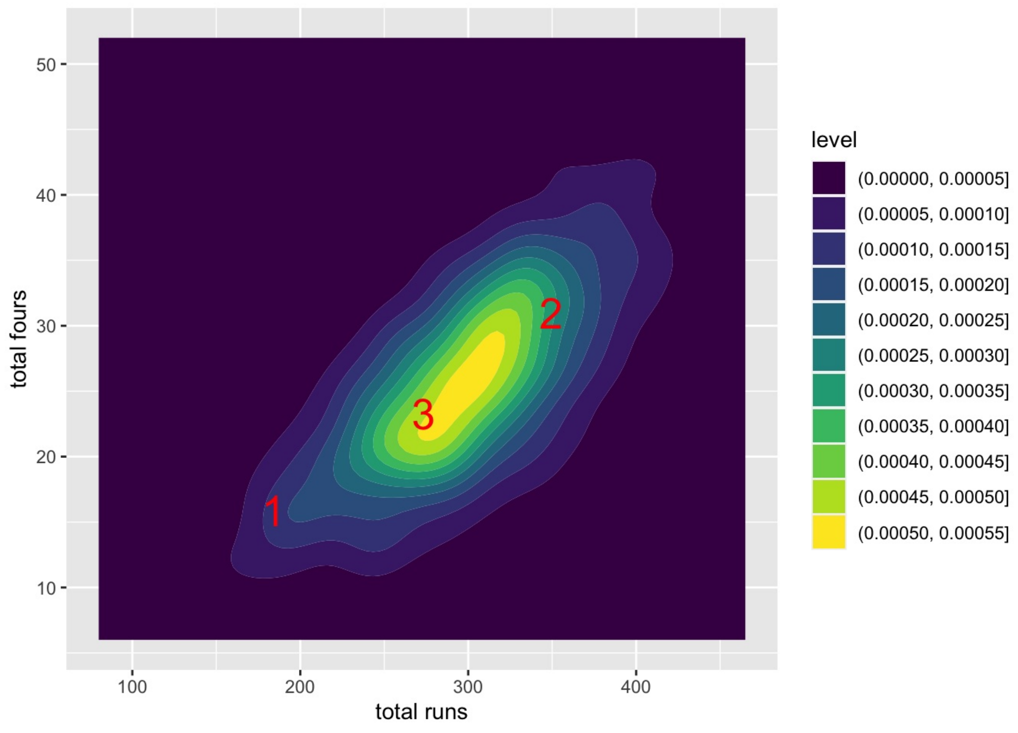
```
df_selected = subset(df_kmeans, select = c(total_runs_scaled, total_fours_scaled, total_balls_scaled))
selected_clust_3 <- kmeans(df_selected, centers = 3, nstart =30 )
fviz_cluster(selected_clust_3, data = df_selected,
             palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw())
```



To visualize the scaled data, we need to understand that those matches (clusters) which have total runs and total fours lesser than the average have a negative value on the scale while those having number of total runs and fours higher than the average have a value greater than 0.

Plotting the cluster centres:

```
df_selected =  subset(df_kmeans, select = c(total_runs, total_fours, total_balls))
selected_clust_3 <- kmeans(df_selected, centers = 3, nstart =30 )
shot_centers <- as.data.frame(selected_clust_3$centers)
gr4 <- gr2 + geom_point(data=shot_centers, aes(x=total_runs, y=total_fours),
inherit.aes = FALSE, col="Red", size = 7, pch=as.character(1:3))
plot(gr4)
```



### Mean squared distance:

Computing the mean squared distance to get an idea about the relative amounts of diffusion and sizes of each of the 3 clusters for quality assurance, given that kmeans clustering is unsupervised.

```
msd <- sqrt(selected_clust_3$withinss / selected_clust_3$size)
print(msd)
```

```
## [1] 53.52250 32.42552 28.71516
```

Relative sizes of each of the 5 clusters:

```
print(selected_clust_3$size)
```

```
## [1]  58 260 324
```

CREATING CLOSE GAMES ARCHETYPE FOR KMEANS CLUSTERING:

```r
#difference in runs
f4 <- gamelog_distinct   %>% group_by(MatchNo, Inning)  %>% filter(Inning ==1) %>% summarise(runs_1stinning = sum
(pmax(NumOutcome,0)))
f5 <- gamelog_distinct   %>% group_by(MatchNo, Inning)  %>% filter(Inning ==2) %>% summarise(runs_2ndinning = sum
(pmax(NumOutcome,0)))
gamelog_distinct <-merge(gamelog_distinct,f4, by = c("MatchNo"))
gamelog_distinct <- merge(gamelog_distinct,f5, by = c("MatchNo") )
gamelog_distinct$difference_in_runs= abs(gamelog_distinct$runs_1stinning.y - gamelog_distinct$runs_2ndinning.y)
gamelog_distinct$difference_in_runs = scale(gamelog_distinct$difference_in_runs)
gamelog_distinct$difference_in_wickets= abs(gamelog_distinct$first_inning_wickets - gamelog_distinct$second_innin
g_wickets)

#difference in wickets
f1<- gamelog_distinct %>% group_by(MatchNo) %>% filter(Inning==1)  %>% summarise(first_inning_wickets = sum(pmax(
Wicket_w,0)))
f2<- gamelog_distinct %>% group_by(MatchNo) %>% filter(Inning==2)  %>% summarise(second_inning_wickets =  sum(pma
x(Wicket_w,0)))
f3 <- merge(f1,f2, by = "MatchNo")
f3$difference_in_wickets <- abs(f3$first_inning_wickets - f3$second_inning_wickets)
gamelog_distinct<-merge(gamelog_distinct,f3, by ="MatchNo")
gamelog_distinct$difference_in_wickets <- scale(gamelog_distinct$difference_in_wickets)

#ADDING DIFFERENCE IN RUNS BETWEEN GAMES IN 2 INNINGS AND DIFFERENCE IN WICKETS
df_kmeans = ddply(gamelog_distinct, "MatchNo", dplyr::summarize,
                  total_runs = sum( pmax(NumOutcome, 0)),
                  total_sixes =  length(which(NumOutcome == 6)),
                  total_fours =  length(which(NumOutcome == 4)),
                  total_wickets = length(which(NumOutcome == -1) ),
                  fielder_mentions = length(which(Fielder != "")),
                  total_balls = length(Ball),
 balls_until_1st_wicket = length(which(Wickets == 0)),
average_wickets_in_during_match = mean(Wickets, na.rm=TRUE),
difference_in_runs = difference_in_runs,
difference_in_wickets = difference_in_wickets
)
```

We can plot the difference in wickets and difference in runs found to find out the close games in T20 cricket, though it hasn't been plotted here in order to keep the report short.

QUESTION 4: Create your own Duckworth Lewis table for the gamelog dataset:

We use only the first innings data to compute the DLS table as usually the DLS table is used to calculate the target score that the second team needs to achieve in order to win a limited overs match which stopped abruptly due to bad weather conditions or other circumstances.

Over2 denotes the number of overs remaining after each subsequent ball within an over. The runs variable represent the sum of runs obtained for each ball bowled. The cum_runs variables is used as a tally variable which computes the score of the second team at that point of time.

We have used optim() which is a generalized optimizer to find the minimum of the objective function. We take a 1 $cross$ 5 vector of zeros as our initial parameter. The loss function here acts as the fn parameter passed into optim() which initially takes in the input values from par and outputs the objective function. The maximum number of iterations is set to 200 and the method used is Nelder-Mead.

```r
first_innings <- gamelog_distinct %>% filter( Inning == 1 )
first_innings[is.na(first_innings)] = 0
first_innings$Over2 = first_innings$Over + first_innings$Ball/6
first_innings$runs<- pmax(first_innings$NumOutcome, 0)
#runs is NRuns
library("dplyr")
first_innings <- first_innings %>% group_by(MatchNo) %>%  mutate(cum_runs=cumsum(runs))
#cumsum is NRunsTally

first_innings$Over2square <- (first_innings$Over2)**2
first_innings<-first_innings %>% group_by(MatchNo)  %>%  mutate(prop= cum_runs/sum(runs))

loss_function = function(x, prop)
{
  A = x[1]
  B = x[2]
  C = x[3]
  D = x[4]
  E = x[5]
  #G = x[6]

 prop_smooth = A*first_innings$Over2 + B*(first_innings$Wickets) + C*(first_innings$Over2square) + D*(log(first_i
nnings$Over2square +1))+E
 error = sum( (prop - prop_smooth)^2)
  return(error)
}



best_params = optim(par=c(0,0,0,0,0), loss_function, prop = first_innings$prop, control= list(trace=1, maxit=200)
)$par
A = best_params[1]
B = best_params[2]
C = best_params[3]
D = best_params[4]
E = best_params[5]


print(A)
print(B)
print(C)
print(D)
print(E)

#CREATING DLT TABLE:

newDLT = matrix(NA, nrow=20, ncol=10)

# Compute the matrix row by row, where each row is an over
for(overcount in 0:20)
{
  # Apply the example formula. 1 - (formula) because resource = 1 - proportion.
  newDLT[overcount,] = 1 -( A*overcount + B*(0:9) + C*overcount**2 + D*log(overcount**2+1)+E)
}

#rename the columns according to the original DLS table
colnames(newDLT) = c("X0","X1","X2","X3","X4","X5","X6","X7","X8","X9")
newDLT = round((newDLT - min(newDLT))/(max(newDLT)-min(newDLT)),3)
head(newDLT)
```

Final DLS table obtained:

```r
for(loopcount in 1:10)
{

temp = rbind(newDLT[2:20, ], rep(0, 10))
newDLT = pmax(temp, newDLT)
}
rownames(newDLT) = c("20","19","18","17","16","15","14","13","12","11","10","9","8","7","6","5","4","3","2","1")
newDLT
```

```
##       X0    X1    X2    X3    X4    X5    X6    X7    X8    X9
## 20 1.000 0.974 0.948 0.922 0.896 0.870 0.844 0.818 0.792 0.766
## 19 0.956 0.930 0.904 0.878 0.852 0.826 0.800 0.775 0.749 0.723
## 18 0.916 0.890 0.864 0.838 0.812 0.786 0.760 0.734 0.708 0.682
## 17 0.877 0.852 0.826 0.800 0.774 0.748 0.722 0.696 0.670 0.644
## 16 0.840 0.814 0.788 0.762 0.736 0.710 0.684 0.658 0.632 0.607
## 15 0.803 0.777 0.751 0.725 0.699 0.673 0.647 0.622 0.596 0.570
## 14 0.766 0.740 0.714 0.689 0.663 0.637 0.611 0.585 0.559 0.533
## 13 0.729 0.703 0.677 0.651 0.625 0.599 0.573 0.548 0.522 0.496
## 12 0.692 0.666 0.640 0.614 0.588 0.562 0.536 0.510 0.484 0.458
## 11 0.654 0.628 0.602 0.576 0.550 0.524 0.498 0.472 0.446 0.420
## 10 0.615 0.589 0.563 0.537 0.511 0.485 0.459 0.433 0.407 0.381
## 9  0.576 0.550 0.524 0.498 0.472 0.446 0.420 0.394 0.368 0.342
## 8  0.536 0.510 0.484 0.458 0.432 0.406 0.380 0.354 0.328 0.302
## 7  0.495 0.469 0.443 0.417 0.391 0.365 0.339 0.313 0.287 0.261
## 6  0.454 0.428 0.402 0.376 0.350 0.324 0.298 0.272 0.246 0.220
## 5  0.411 0.385 0.359 0.333 0.307 0.281 0.256 0.230 0.204 0.178
## 4  0.368 0.342 0.316 0.290 0.264 0.238 0.212 0.186 0.160 0.134
## 3  0.324 0.298 0.272 0.246 0.220 0.194 0.168 0.142 0.116 0.091
## 2  0.279 0.253 0.227 0.201 0.176 0.150 0.124 0.098 0.072 0.046
## 1  0.234 0.208 0.182 0.156 0.130 0.104 0.078 0.052 0.026 0.000
```

We can observe that the DLS table obtained for our dataset is correct as it starts from 1 and ends in 0 and the trend seems to be decreasing across the overs from top to bottom. Similarly, the values seem to be decreasing across the 10 wickets from 0 to 9. The only difference is that the original DLS tabel has a row of 0’s attached at the end, which we have omitted in this case, therefore we get a 20 $cross$ 10 matrix.

EDAFINALPROJECTAKSHAYA

CODE:

.

QUESTION 2:

STEP 1: Computing valence scores by performing sentiment analysis on FullNotes

1.Using nrc method :

```
library(dplyr)
library(stringr)
library(syuzhet)
library(sentimentr)
second_innings <- gamelog_distinct %>% filter(Inning ==2)
first_innings <- gamelog_distinct %>% filter(Inning ==1)
s_v <- get_sentences(second_innings$FullNotes)
s_v<-unlist(as.character(s_v))

nrc_data <- get_nrc_sentiment(s_v)

angry_items <- which(nrc_data$anger > 0)
head(angry_items,1)
```

```
## [1] 16
```

```
joy_items <- which(nrc_data$joy > 0)
head(s_v[joy_items],1)
```

```
## [1] "good length   drifting into Gilchrist, no room to cut, played back to the bowler"
```

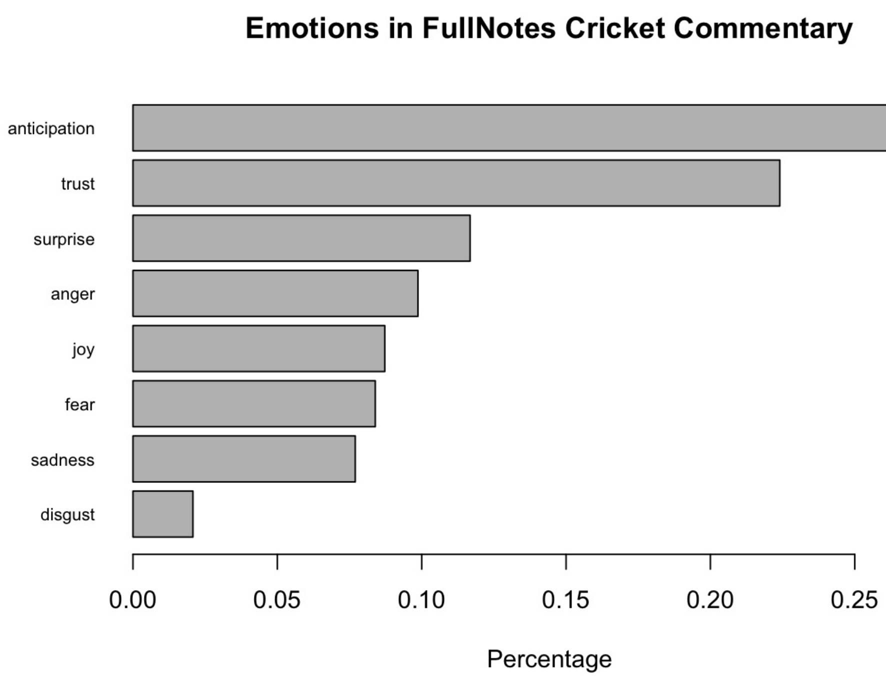
```
anticipation_items <- which(nrc_data$anticipation > 0)
head(s_v[anticipation_items])
```

```
## [1] "good length   drifting into Gilchrist, no room to cut, played back to the bowler"
## [2] "c(\"that was a Jaffa!\", \"good length on the corridor outside off   shapes away just a touch, Gilchrist
sticks his bat out and the ball just beats the bat\")"
## [3] "good length   on the stumps, pushed back to Rasel"
## [4] "full and driven past Rasel   long-off stops it, Ashraful goes on the defensive with this field"
## [5] "good length on the stumps   pushed back to Rasel"
## [6] "c(\"wow!\", \"pitches just outside off on a good length   Hayden gets on his knee and sweeps over deep sq
uare leg\")"
```

```
# head(pander::pandoc.table(nrc_data[, 1:8], split.table = Inf))
# head(pander::pandoc.table(nrc_data[, 9:10]))
valence <- (nrc_data[, 9]*-1) + nrc_data[, 10]
head(valence,10)
```

```
##  [1]  1  1  1 -1  1  0  0  1  1 -1
```

```
barplot(
  sort(colSums(prop.table(nrc_data[, 1:8]))),
  horiz = TRUE,
  cex.names = 0.7,
  las = 1,
  main = "Emotions in FullNotes Cricket Commentary", xlab="Percentage"
  )
```



2. Using affin vector and nrc vector:

```
afinn_vector <- get_sentiment(s_v, method = "afinn")
head(afinn_vector)
```

```
## [1]  0  1  3  0  0 -2
```

```
nrc_vector <- get_sentiment(s_v, method = "nrc", lang = "english")
head(nrc_vector)
```

```
## [1]  1  1  1 -1  1  0
```

```
sum(nrc_vector)
```

```
## [1] 23154
```

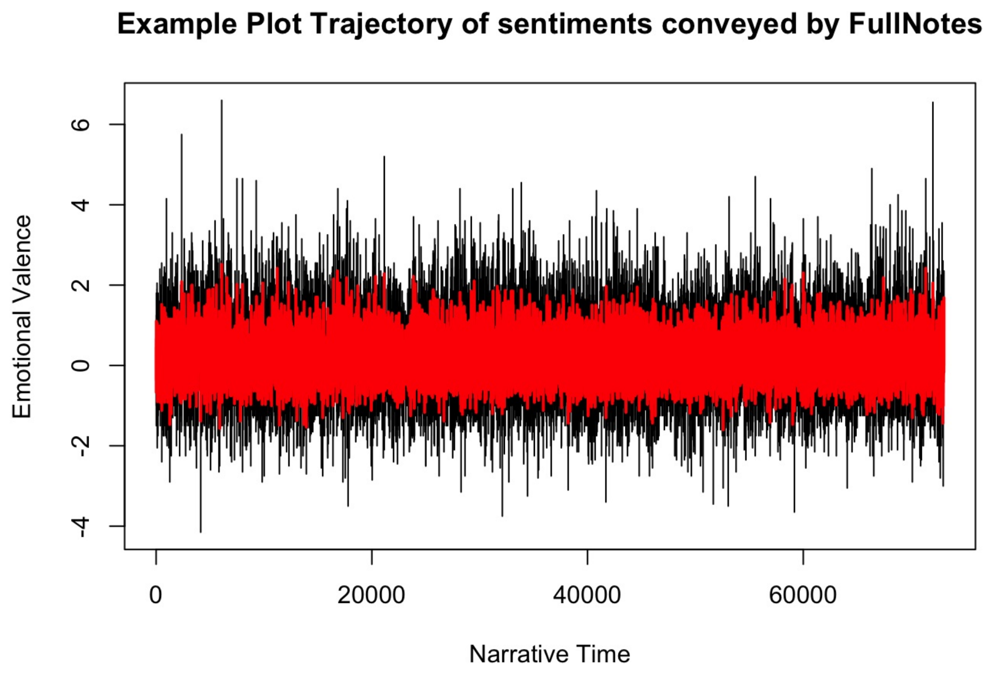
```
mean(nrc_vector)
```

```
## [1] 0.316987
```

```
summary(nrc_vector)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -6.000   0.000   0.000   0.317   1.000   7.000
```

```
library(zoo)

s_v_sentiment <- get_sentiment(s_v)
plot(
  s_v_sentiment,
  type="l",
  main="Example Plot Trajectory of sentiments conveyed by FullNotes",
  xlab = "Narrative Time",
  ylab= "Emotional Valence"
  )
lines(zoo :: rollmean(s_v_sentiment,k=3,fill=NA),col="red",lwd=2)
```



We use the valence scores classifying the comments in FullNote as positive and negative by using the nrc_data and get_nrc_sentiment function i.e. Method 1.

The mean turns out to be around 0.3 indicating that most of the comments made by the commentators have been neutral in the gamelog dataset.

We can also note that most of the comments in FullNotes have been classified to have an anticipatory tone, probably due to the usual uncertainity involved in games like cricket, which is usually expressed in commentaries.

STEP 2: Computing resources used and the excitement score:

Excitement score has been computed as follows:

$$excitementscore \;=\; exp(|runs\ scored\ per\ ball| - (resources\ used)\cdot(target\ runs\ scored\ by\ the\ first\ team) + |valence| - |difference\ in\ required\ run\ rate\ and\ current\ run\ rate|)$$

```
DLS = read.csv("Downloads/DLS_T20.csv")[,-1]
DLS[16,2] =0.3
DLS_matrix <- data.matrix(DLS)
second_innings <- gamelog_distinct %>% filter(Inning ==2)
over <- c(1:20)
wickets <- c(0:9)
resources_left_1= matrix(0, 21, 10)
resources_used_1= matrix(0, 21, 10)

for(i  in over)
{
for(j in wickets )
{
resources_left_1[i,j+1] = 5/6*(DLS_matrix[i,j+1]) + 1/6*(DLS_matrix[i+1,j+1])
resources_used_1[i,j+1] <- DLS_matrix[i,j+1] - resources_left_1[i,j+1]
}
}

colnames(resources_used_1) = c("0","1","2","3","4","5","6","7","8","9")
library(reshape)
resources_used_1 <- melt(resources_used_1)
colnames(resources_used_1) = c("Over","Wickets","resources_used_value")
resources_used_1$Ball <-1

over <- c(1:20)
wickets <- c(0:9)

resources_left_2= matrix(0, 21, 10)
resources_used_2= matrix(0, 21, 10)


for(i  in over)
{
for(j in wickets )
{
resources_left_2[i,j+1] = 4/6*(DLS_matrix[i,j+1]) + 2/6*(DLS_matrix[i+1,j+1])
resources_used_2[i,j+1] <- DLS_matrix[i,j+1] - resources_left_2[i,j+1]
}
}

colnames(resources_used_2) = c("0","1","2","3","4","5","6","7","8","9")

library(reshape)
resources_used_2 <- melt(resources_used_2)
colnames(resources_used_2) = c("Over","Wickets","resources_used_value")
resources_used_2$Ball <-2


over <- c(1:20)
wickets <- c(0:9)
resources_left_3= matrix(0, 21, 10)
resources_used_3= matrix(0, 21, 10)


for(i  in over)
{
for(j in wickets )
{
resources_left_3[i,j+1] = 3/6*(DLS_matrix[i,j+1]) + 3/6*(DLS_matrix[i+1,j+1])
resources_used_3[i,j+1] <- DLS_matrix[i,j+1] - resources_left_3[i,j+1]
}
}

colnames(resources_used_3) = c("0","1","2","3","4","5","6","7","8","9")

library(reshape)
resources_used_3 <- melt(resources_used_3)
colnames(resources_used_3) = c("Over","Wickets","resources_used_value")
resources_used_3$Ball <-3

over <- c(1:20)
wickets <- c(0:9)
resources_left_4= matrix(0, 21, 10)
resources_used_4= matrix(0, 21, 10)


for(i  in over)
{
for(j in wickets )
{
resources_left_4[i,j+1] = 2/6*(DLS_matrix[i,j+1]) + 4/6*(DLS_matrix[i+1,j+1])
resources_used_4[i,j+1] <- DLS_matrix[i,j+1] - resources_left_4[i,j+1]
}
}

colnames(resources_used_4) = c("0","1","2","3","4","5","6","7","8","9")

library(reshape)

resources_used_4 <- melt(resources_used_4)


colnames(resources_used_4) = c("Over","Wickets","resources_used_value")

resources_used_4$Ball <-4

over <- c(1:20)
wickets <- c(0:9)

resources_left_5= matrix(0, 21, 10)
resources_used_5= matrix(0, 21, 10)

for(i  in over)
{
for(j in wickets )
{
resources_left_5[i,j+1] = 1/6*(DLS_matrix[i,j+1]) + 5/6*(DLS_matrix[i+1,j+1])
resources_used_5[i,j+1] <- DLS_matrix[i,j+1] - resources_left_4[i,j+1]
}
}

colnames(resources_used_5) = c("0","1","2","3","4","5","6","7","8","9")
library(reshape)

resources_used_5 <- melt(resources_used_5)
```

```
colnames(resources_used_5) = c("Over","Wickets","resources_used_value")

resources_used_5$Ball <-5


over <- c(1:20)
wickets <- c(0:9)
resources_left_6= matrix(0, 21, 10)
resources_used_6= matrix(0, 21, 10)

for(i  in over)
{
for(j in wickets )
{
resources_left_6[i,j+1] = 0*(DLS_matrix[i,j+1]) + 6/6*(DLS_matrix[i+1,j+1])
resources_used_6[i,j+1] <- DLS_matrix[i,j+1] - resources_left_6[i,j+1]
}
}

colnames(resources_used_6) = c("0","1","2","3","4","5","6","7","8","9")
library(reshape)

resources_used_6 <- melt(resources_used_6)
colnames(resources_used_6) = c("Over","Wickets","resources_used_value")

resources_used_6$Ball <-6

#resources left merged df contains the resources used computed on a ball by ball basis
resources_left_merged <- bind_rows(resources_used_1,resources_used_2,resources_used_3,resources_used_4,resources_
used_5,resources_used_6)


#Finding out the target runs for the second team
Targets <- first_innings %>% dplyr::select(MatchNo,NumOutcome)  %>% group_by(MatchNo) %>% summarise(total_runs =
sum(pmax(NumOutcome,0)))
Targets$Target_runs <- Targets$total_runs +1


#Adding the targets column to second_innings data
second_innings <- merge (second_innings, Targets, by = "MatchNo")

#Finding runs that the second team falls behind from the target
second_innings <- second_innings %>% group_by(MatchNo) %>%  mutate(cum_runs=cumsum(pmax(NumOutcome,0)))
second_innings$runs_short <- second_innings$Target_runs - second_innings$cum_runs


#Computing overs remaining and the required run rate by the second team to beat the target runs set by the first
team
second_innings$Overs_used = second_innings$Over + second_innings$Ball/6
second_innings$Overs_remaining <- 20- second_innings$Overs_used
second_innings$required_run_rate <- second_innings$runs_short / second_innings$Overs_remaining


#Current run rate of the team playing in the second innings
second_innings$current_run_rate <- second_innings$cum_runs / second_innings$Overs_used

#How much the second team falls behind or ahead can be computed by finding out the difference between current run
rate and required run rate -> this is indicated by lag_lead
second_innings$lag_lead <- second_innings$current_run_rate - second_innings$required_run_rate
second_innings$valence <- valence

#Merging the resources used value with the seocnd inning df
second_innings<-merge(second_innings, resources_left_merged, by=c("Over","Wickets","Ball"))

#filtering infinite values in laglead
second_innings <- second_innings %>% filter( lag_lead > -Inf )
second_innings <- second_innings %>% filter( lag_lead < Inf )

#scale each of the variables used in computing the excitement score before using it for computation so that each
variable is considered equally and equally contribute towards computation of the excitement score
second_innings$NumOutcome = scale(second_innings$NumOutcome)
second_innings$resources_used_value = scale(second_innings$resources_used_value)
second_innings$Target_runs = scale(second_innings$Target_runs)
second_innings$valence = scale(second_innings$valence)
second_innings$lag_lead = scale(second_innings$lag_lead)

#computing the excitement score
second_innings$excitement_score_computed <-exp(abs(pmax(second_innings$NumOutcome,0))-second_innings$resources_us
ed_value*second_innings$Target_runs+abs(second_innings$valence)-abs(second_innings$lag_lead))
second_innings$excitement_score_computed = scale(second_innings$excitement_score_computed )
h<-head(second_innings[order(second_innings$excitement_score_computed, decreasing = TRUE), ],50)
# write.csv(h, "excitement_score.csv")
```

Results have been attached in a csv file called excitement_scores.csv due to space issues. We can find out that we have got a diverse set of 20 balls and varying ball types(i.e, it has sixes, fours, outs, ones, legbyes, byes, runs, no ball etc) from the scaled excitement scores obtained above.

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

WHICH BALLS TO INCLUDE IN THE HIGHLIGHT REEL:

```
excitement_scores <- read.csv("Documents/excitement_score.csv")
head(excitement_scores, 1)
```

```
##      X Over Wickets Ball MatchNo Format TeamBowling TeamBatting Inning
## 1 8455   11       1    6     334   T20I         PAK          SA      2
##         Bowler BowlerID      Batsman BatsmanID Fielder FielderID Outcome
## 1 Abdur Rehman     5041 F du Plessis      6046                NA     SIX
##   NumOutcome BallType NumBallType Notes
## 1   2.984994      run           0   now
##
## 1 now Faf goes all the way! It was flighted on off   Faf made room and launched this full delivery s
##   IDflag Wicket_w total_runs Target_runs cum_runs runs_short Overs_used
## 1      0        0         98   -2.080416       75         24         12
##   Overs_remaining required_run_rate current_run_rate  lag_lead  valence
## 1               8                 3             6.25 0.4866153 2.899088
##   resources_used_value excitement_score_computed
## 1             2.069577                  176.4786
```

Listing the top 20 exciting balls (indicated by Ball) and the match no, over, the ball number in that over (a over has 6 balls) and bowler who bowled the ball.

```
head(excitement_scores[c("MatchNo","Over","Ball","Bowler","Outcome","excitement_score_computed")],25)
```

```
##    MatchNo Over Ball             Bowler Outcome excitement_score_computed
## 1      334   11    6       Abdur Rehman     SIX                 176.47865
## 2   201266   14    6          VY Mahesh     SIX                  74.21211
## 3   201142   12    6           R Bhatia     SIX                  54.26801
## 4      117   14    6 Mehrab Hossain jnr    FOUR                  39.71832
## 5   201147    3    1             Harris     SIX                  33.18180
## 6      323   14    6           H Davids    FOUR                  31.03107
## 7      151   19    5         SL Malinga     SIX                  30.86198
## 8      189   10    1          JP Duminy     SIX                  30.20935
## 9      131    4    6   Shahadat Hossain    FOUR                  28.58536
## 10  201134   15    6         KA Pollard       1                  28.50597
## 11  201460    8    2    Karanveer Singh     SIX                  28.40006
## 12      41    3    6       CRD Fernando       4                  27.74958
## 13     306   11    2    Mohammad Hafeez     SIX                  24.79209
## 14  200946   17    6         PJ Sangwan     SIX                  24.62696
## 15  201540    9    3            Maxwell     SIX                  24.52754
## 16     392    1    1      Sohail Tanvir    FOUR                  24.27891
## 17  201526    4    6     NM Coulter-Nile     SIX                  21.15639
## 18  201301   11    6           R Bhatia     OUT                  20.35016
## 19     328    8    2        Fawad Ahmed    FOUR                  20.20520
## 20      52   10    6    Harbhajan Singh       1                  19.89369
## 21  201134    9    6         KA Pollard     SIX                  19.50160
## 22     351   16    1         SL Malinga    FOUR                  18.37375
## 23     176   15    5      Shahid Afridi     SIX                  17.36703
## 24  201414   10    6          YS Chahal      no                  17.11379
## 25     414   16    2          KJ Abbott     SIX                  16.41500
```

Listing out the comment for which the sentiment score was generated followed by the sentiment scores obtained using nrc sentiment function (valence variable).

FullNotes, Valence displayed below for the 20 balls in the highlight reels:

1.now Faf goes all the way! It was flighted on off Faf made room and launched this full delivery straight down the ground for maximum, 2.89908828553307

2.smashed over backward square-leg for a six! That was a quicker one angling in and Gilchrist has picked it up and slammed it into the fence for a biggie 22 from that Mahesh over, -3.57670946534003

3.Jaidev Unadkat takes an absolute blinder but he's tripped over the ropes in the process. Slower one and Dhawan slog-swept it and Unadkat, standing on the edge of the ropes at the deep midwicket boundary timed his jump to perfection and plucked it out with his left hand but couldn't hold his balance and fell over. Brilliant effort nevertheless, 6.13698716096961

4.poor ball short on leg stump, begging to be hit, pulls it away with disdain behind square for a boundary, just 18 more needed, -3.57670946534003

5.clobber! finally after eons, Gayle gets one in his slot, full, around off stump, Gayle gets front leg out of the way, and hammers it over long off, there is a man there, he is just closer to the action, can't do anything else, 2.89908828553307

6.it was a mighty decent over till now but Sanga reaches for this full delivery around off and lifts it over the bowler, 3.97838791067858

7.Six Six, Six! What a blow, Nathan McCullum's handed New Zealand a thrilling win, lands on a length outside off, just perfect for McCullum who strikes it powerfully and cleanly over the long-off boundary to seal victory for New Zealand, 6.13698716096961

8.Duminy is into the attack he bowled dreadfully in the first game… and starts very badly here, dragging down a long-hop that Chibhabha slams over the midwicket fence, -3.57670946534003

9.McCullum sits back in his crease this time and launches a wide half volley over backward point for four. In fact his bat looked OK with that last stroke…, 2.49740984019451

10.the keeper hits direct at the batsman's end after the batsman bottom edges the ball over his stumps but the single has already been stolen,-2.49740984019451

11.that's full and wide and Yusuf has a lot of room to swing freely and launch it over extra cover, with the turn, -3.97838791067858

12.another uncharacteristic fumble from Sangakkara. A nasty bounce and a slower ball as well, but hardly an excuse,-3.57670946534003

13.gets a low full-toss and Rory takes full advantage by clobbering this over extra cover. Into the crowd again,2.89908828553307

14.that was awful from Sangwan hard to criticise an 18-year-old, but that was a full toss on the pads, was asking to be hit out of the park, Irfan swings it over the square-leg boundary, -1.418110215049

15.encore! This is into the second tier of the stands behind long-off. Gets down the track to make his own length and hammers it through the line. All Maxi can do is offer an apology to his captain. Hundred-run stand brings out the usual Kohli-Gayle handshake and blow-it-up, 5.0576875358241

16.this is beautifully played! The ball was angling across the right hander who just shows it the way past the first slip fielder.. Fraught with risk… but well played, -4.65600909048554

17.pitched up and outside off width on offer now. Gayle slices it over extra-cover for a flat six. Looks like he is in the mood, 1.81978866038755

18.the lack of runs has forced Ojha to go for the bit hit he steps out and sends it a long way in the air but not long enough, he picks pout Brett Lee at long-off, no problem taking that one, Delhi losing their way here, -5.73530871563106

19.full and on the stumps reverses the hands and gets just a tickle on the ball to send it fine past the keeper, Ahmed was interested as it deflected away off the pad but Rob Bailey had spotted the bat involved beforehand, 2.89908828553307

20.stretches forward and defends the fuller delivery to the off side, 1.81978866038755

The balls present in rows 2-20 (19 balls) in the excitement_score.csv file can be considered to be a part of the highlight reel, as they have the highest excitement scores as can be observed from the results in the csv file.

The 20th ball is the ball present in the 25th row which is a no ball in order to introduce some diversity in type of balls in the highlight reel.

The highlight reel selected here includes different types of balls including sixes, fours, outs/balls which resulted in a wickets, ones, no balls, byes etc.