STAT-847-Final-Project

Pranjal Gaur

20/04/2022

Reading Data

Reading the gamelog data and checking dimensions

```
match_data= read.csv("Gamelog T20I Stat 847.csv")
head(match_data)
```

```
##   Format MatchNo TeamBowling TeamBatting Inning Over Ball Bowler BowlerID
## 1   T20I      33         AUS          BD      1    0    1  B Lee       17
## 2   T20I      33         AUS          BD      1    0    2  B Lee       17
## 3   T20I      33         AUS          BD      1    0    3  B Lee       17
## 4   T20I      33         AUS          BD      1    0    4  B Lee       17
## 5   T20I      33         AUS          BD      1    0    4  B Lee       17
## 6   T20I      33         AUS          BD      1    0    5  B Lee       17
##       Batsman BatsmanID Fielder FielderID Outcome NumOutcome BallType
## 1 Tamim Iqbal      1041                NA      no          0      run
## 2 Tamim Iqbal      1041                NA      no          0      run
## 3 Tamim Iqbal      1041                NA      no          0      run
## 4 Tamim Iqbal      1041                NA       1          1     wide
## 5 Tamim Iqbal      1041                NA      no          0      run
## 6 Tamim Iqbal      1041                NA      no          0      run
##   NumBallType  Notes
## 1           0   good
## 2           0  short
## 3           0  short
## 4           2  Tamim
## 5           0 fuller
## 6           0   good
##
## 1                                          good start by Lee   dug in short of a length outside of
## 2                                       short of a length outside off again   this time Tamim gets
## 3                                          short again and aimed at the body   Tamim gets on the ba
## 4 Tamim backs away to whack that over the off side   Lee senses it and thuds it in short, the ball sa
## 5                                           fuller in length and inviting the drive   Tamim flashe
## 6                                                                                 good length aimed at
##   IDflag Wickets
## 1      0       0
## 2      0       0
## 3      0       0
## 4      0       0
## 5      0       0
## 6      0       0
```

```
dim(match_data)
```

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

```
data1=match_data
```

Data Cleaning

Analyzing the data and filtering out match_ids with more than 1 match data, missing leagues, abnormal wicket count(like 24), more than two teams in a game, etc.

```
sort(unique(match_data$MatchNo))
```

```
##   [1]      1      2      3      4      5      6      7      8     10     11
##  [11]     12     13     14     15     16     18     20     23     24     25
##  [21]     28     29     30     31     32     33     34     35     36     37
##  [31]     38     39     40     41     42     43     44     45     46     47
##  [41]     48     49     50     51     52     53     54     55     56     57
##  [51]     69     72     73     76     77     78     79     80     81     82
##  [61]     83     84     85     86     87     88     89     92     93     95
##  [71]     97     99    100    103    104    105    106    107    109    111
##  [81]    112    113    114    115    116    117    118    119    120    121
##  [91]    122    123    124    125    126    127    131    134    144    145
## [101]    146    147    148    149    150    151    154    155    156    157
## [111]    158    159    161    163    164    165    166    167    168    169
## [121]    170    171    172    173    174    175    176    177    178    179
## [131]    180    181    182    183    184    185    186    187    188    189
## [141]    190    191    192    193    194    195    196    197    198    199
## [151]    200    201    202    203    204    205    206    207    208    209
## [161]    210    211    212    213    214    215    216    217    218    219
## [171]    220    221    222    223    226    228    229    241    242    243
## [181]    244    245    246    247    248    255    256    257    258    259
## [191]    260    261    262    263    266    267    269    270    271    272
## [201]    274    275    276    277    278    279    280    281    282    283
## [211]    284    285    286    287    288    289    290    291    292    293
## [221]    294    295    296    297    298    299    300    301    302    303
## [231]    304    305    306    308    312    315    316    317    321    322
## [241]    323    324    325    326    327    328    329    331    333    334
## [251]    336    340    341    343    350    351    352    353    355    356
## [261]    357    358    361    362    363    364    365    378    379    380
## [271]    381    382    383    385    387    388    389    391    392    393
## [281]    395    397    398    399    400    401    402    403    405    406
## [291]    407    408    409    411    412    413    414    415    416    417
## [301]    418    423    428    429    440    442    448    449    450    451
## [311]    452    453    454    455    456    457 200901 200902 200903 200904
## [321] 200905 200906 200908 200909 200910 200911 200912 200914 200915 200916
## [331] 200917 200918 200919 200920 200921 200922 200923 200924 200925 200926
## [341] 200927 200928 200929 200930 200931 200932 200933 200934 200935 200936
## [351] 200937 200938 200939 200940 200941 200942 200943 200944 200945 200946
## [361] 200947 200948 200949 200950 200951 200952 200953 200954 200955 200956
## [371] 200957 200958 200959 200960 200961 200962 201101 201102 201103 201106
```

```
## [381] 201107 201108 201109 201110 201111 201112 201114 201115 201116 201117
## [391] 201118 201119 201122 201123 201124 201125 201127 201128 201129 201130
## [401] 201131 201132 201133 201134 201135 201136 201137 201139 201140 201141
## [411] 201142 201143 201145 201146 201147 201148 201149 201150 201152 201153
## [421] 201154 201155 201156 201157 201158 201159 201160 201161 201162 201163
## [431] 201164 201165 201166 201167 201168 201169 201170 201171 201172 201173
## [441] 201174 201175 201201 201202 201203 201204 201205 201206 201207 201209
## [451] 201210 201211 201212 201213 201214 201215 201216 201217 201218 201219
## [461] 201220 201221 201222 201224 201225 201226 201228 201229 201230 201231
## [471] 201236 201237 201238 201239 201240 201241 201243 201244 201246 201247
## [481] 201248 201249 201250 201251 201253 201254 201255 201256 201257 201258
## [491] 201259 201261 201262 201263 201265 201266 201267 201268 201269 201270
## [501] 201271 201272 201273 201274 201275 201276 201301 201302 201303 201305
## [511] 201306 201307 201308 201309 201310 201311 201312 201314 201316 201317
## [521] 201318 201319 201320 201321 201322 201323 201324 201325 201326 201327
## [531] 201328 201330 201333 201334 201335 201336 201337 201338 201340 201341
## [541] 201342 201343 201344 201345 201346 201347 201348 201349 201351 201352
## [551] 201353 201354 201355 201356 201357 201358 201359 201360 201361 201362
## [561] 201363 201364 201365 201366 201367 201368 201369 201370 201371 201372
## [571] 201401 201402 201403 201404 201405 201406 201407 201408 201409 201410
## [581] 201411 201412 201413 201414 201415 201416 201417 201418 201419 201420
## [591] 201421 201422 201423 201424 201425 201426 201427 201428 201429 201430
## [601] 201431 201432 201433 201434 201435 201436 201437 201438 201439 201440
## [611] 201441 201442 201443 201444 201445 201446 201447 201448 201449 201450
## [621] 201451 201452 201453 201454 201455 201456 201457 201458 201459 201460
## [631] 201501 201502 201503 201504 201505 201506 201507 201508 201509 201510
## [641] 201511 201512 201513 201514 201515 201516 201517 201518 201519 201520
## [651] 201521 201522 201523 201524 201526 201527 201528 201529 201530 201531
## [661] 201532 201533 201534 201535 201536 201537 201538 201539 201540 201541
## [671] 201542 201543 201544 201545 201546 201547 201548 201549 201550 201551
## [681] 201552 201553 201554 201555 201557 201558 201559 201560
```

```
summary(match_data$NumOutcome)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
##   -1.000    0.000    1.000    1.238    1.000 2015.000       16
```

As we see there exists a match with 2015 runs which is an anomaly, so we filter it out.

```
rowsToExclude=which(data1$NumOutcome==2015)
data1=match_data[-rowsToExclude,]
```

Similarly, we exclude rows containing NA, considering these to be case of MCAR. Also, we remove rows containing empty fields in Format.

```
data1=data1[!(is.na(data1$MatchNo)),]
data1=data1[!(is.na(data1$NumOutcome)),]
data1=data1[-which(data1$Format ==" "),]
```

Finally, we proceed to find and remove data for matches with abnormal runs or single match id containing duplicate or multiple matches data.

```
match_ids_faulty= c()
match_ids_faulty = c(match_ids_faulty, unique(data1[which(data1$Wickets>9), 2]))

faulty_matches = c(191, 440, 200903, 200906, 200933,200938,200947,200948,201103,
                   201109, 201112,201128,201131,201211,201228,201258
                   ,201273,201275,201536,201554)
match_ids_faulty = c(match_ids_faulty, faulty_matches)
data2=data1[-which(data1$MatchNo %in% match_ids_faulty),]

total_runs_per_match_clean_data = ddply(data2,
                                        .(MatchNo),
                                        summarize,
                                        total_runs = sum(NumOutcome[which(
                                          NumOutcome >=0 & NumOutcome<=7)])
)
summary(total_runs_per_match_clean_data)
```

```
##     MatchNo           total_runs
##  Min.   :     1.0   Min.   : 80.0
##  1st Qu.:   212.8   1st Qu.:262.0
##  Median :200915.5   Median :299.0
##  Mean   :103850.3   Mean   :309.2
##  3rd Qu.:201319.2   3rd Qu.:336.0
##  Max.   :201560.0   Max.   :786.0
```

```
matches_with_redundant_records = total_runs_per_match_clean_data[which(
  total_runs_per_match_clean_data$total_runs > 465),1]
data2=data2[-which(data2$MatchNo %in% matches_with_redundant_records),]

head(data2)
```

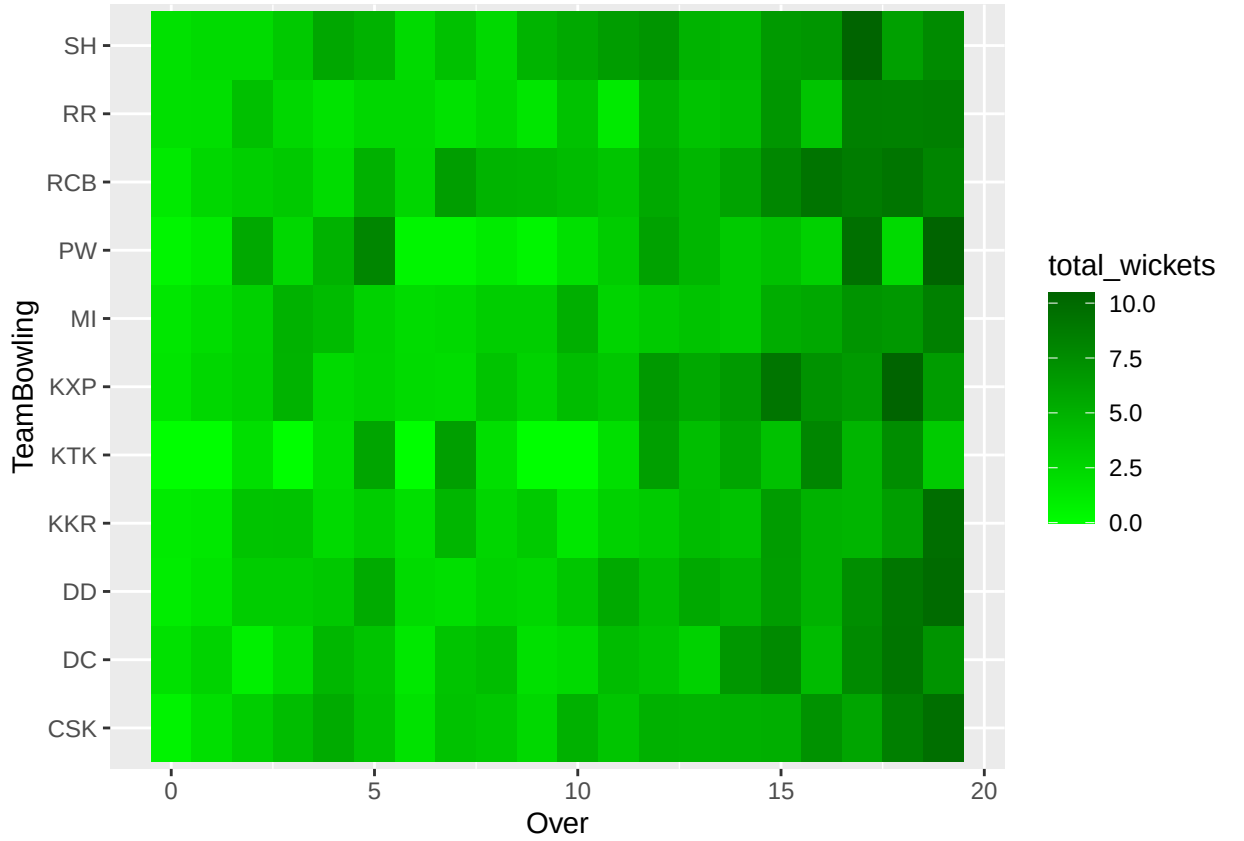
```
##   Format MatchNo TeamBowling TeamBatting Inning Over Ball Bowler BowlerID
## 1   T20I      33         AUS          BD      1    0    1  B Lee       17
## 2   T20I      33         AUS          BD      1    0    2  B Lee       17
## 3   T20I      33         AUS          BD      1    0    3  B Lee       17
## 4   T20I      33         AUS          BD      1    0    4  B Lee       17
## 5   T20I      33         AUS          BD      1    0    4  B Lee       17
## 6   T20I      33         AUS          BD      1    0    5  B Lee       17
##       Batsman BatsmanID Fielder FielderID Outcome NumOutcome BallType
## 1 Tamim Iqbal      1041               NA      no          0      run
## 2 Tamim Iqbal      1041               NA      no          0      run
## 3 Tamim Iqbal      1041               NA      no          0      run
## 4 Tamim Iqbal      1041               NA       1          1     wide
## 5 Tamim Iqbal      1041               NA      no          0      run
## 6 Tamim Iqbal      1041               NA      no          0      run
##   NumBallType  Notes
## 1           0   good
## 2           0  short
## 3           0  short
## 4           2  Tamim
## 5           0 fuller
## 6           0   good
##
```

```
## 1                                                 good start by Lee   dug in short of a length outside of
## 2                                               short of a length outside off again   this time Tamim gets
## 3                                                  short again and aimed at the body   Tamim gets on the ba
## 4 Tamim backs away to whack that over the off side   Lee senses it and thuds it in short, the ball sa
## 5                                                   fuller in length and inviting the drive   Tamim flash
## 6                                                                                    good length aimed at
##   IDflag Wickets
## 1      0       0
## 2      0       0
## 3      0       0
## 4      0       0
## 5      0       0
## 6      0       0
```

Part 1. Answer

For first ggplot, we slice the data to get insights about how teams approach six-hitting through the course of an inning in IPL, we plot the heat map showing percent of total sixes hit in each over by IPL teams.

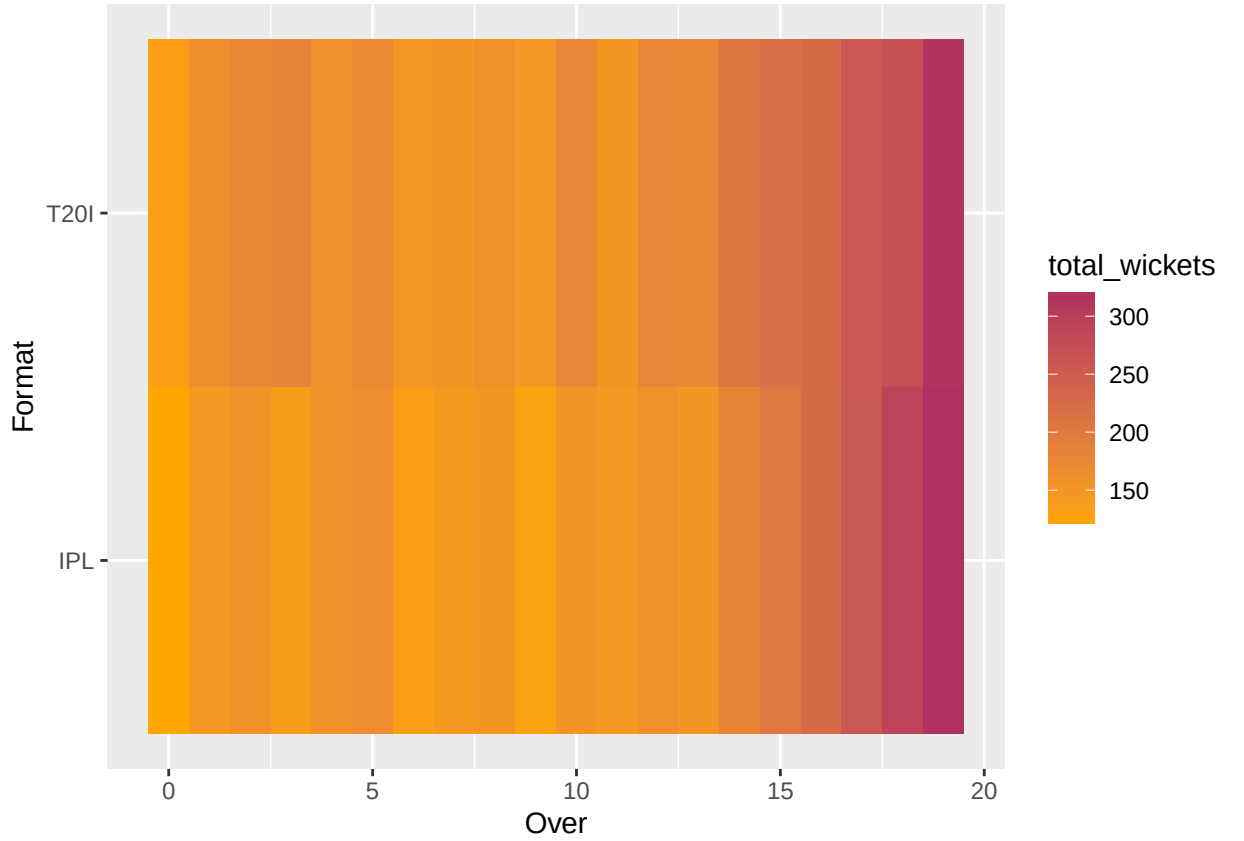
```
# Analysis of percentage of total sixes hit in each over by an IPL team

 #only including IPL teams
df_split_1=ddply(subset(data2, Format == "IPL" & !(TeamBowling %in% c("IND"))),
                 .(TeamBowling,Over),
                 summarize,
                 total_wickets = mean(NumOutcome == 6)*100
)

ggplot(df_split_1,aes(y=TeamBowling,x=Over, fill=total_wickets)
       ,xlab="Over"
       ,ylab="Team") + geom_tile() +scale_fill_gradient(low = "green",
                                                        high = "dark green")
```



As we see above, maximum of sixes are preferred to hit in final 3 overs, which makes sense as teams are not afraid to take more risk towards final 3 overs of the innings.

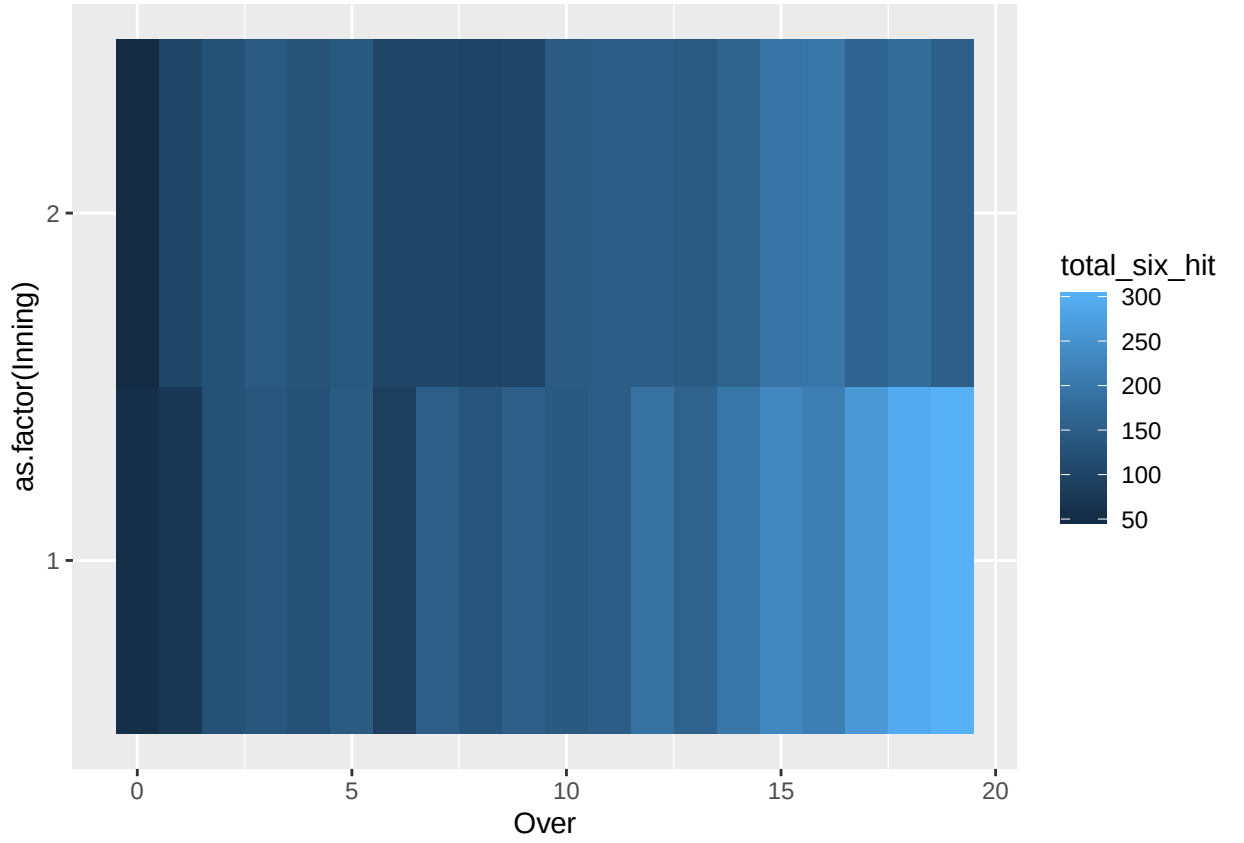
For second ggplot, we slice the data to find total wickets taken in each over across two formats - namely, IPL and T20 Internationals. This analysis again provides insights into regarding how bowler-friendly are the pitches in IPL vs T20I based on how many wickets fall in each over across two leagues.

```
#total wickets taken in each over across format
df_split_2=ddply(data2,
                 .(Over,Format),
                 summarize,
                 total_wickets = length(which(NumOutcome == -1))
)

ggplot(df_split_2,aes(x=Over,y=Format, fill=total_wickets)
       ,xlab="Over"
       ,ylab="Inning") +  geom_tile() + scale_fill_gradient(low = "orange",
                                                            high = "maroon")
```



Further, analyzing the data to find total sixes hit in each over across two innings. This analysis can help to understand the difference in risk approach of batsmen across innings and different overs.

```
#total sixes hit in each over in an inning
df_split_3=ddply(data2,
                 .(Over,Inning),
                 summarize,
                 total_six_hit = length(which(NumOutcome == 6))
                 )

ggplot(df_split_3,aes(x=Over,y=as.factor(Inning), fill=total_six_hit)
       ,xlab="Over"
       ,ylab="Inning") +  geom_tile()
```



Part 2. Answer

Generating highlights of second inning data based on sum of emotional valence and resource usage difference between subsequent balls calculated for each ball to assign a score.

(Please follow comments for code explanation)

```
DLS = read.csv("DLS_T20.csv")[,-1]
result_highlights = data.frame()
match_ids = unique(data2$MatchNo)

for(match in match_ids) #repeat for each match
{
  resource_used_vector = c() #vector that contains resource usage difference for each ball
  resource_left_vector = c() #vector that contains resource left after each ball
  runs_needed_to_keep_up=c() #vector that contains runs needed after each ball

  data_first_inning=subset(data2,MatchNo ==match & Inning==1) #first inning data
  target = by(data_first_inning$NumOutcome>=0 & data_first_inning$NumOutcome<=7,
              data_first_inning$Inning, sum) #target
  data_second_inning=subset(data2,MatchNo == match & Inning==2) #second inning data

  sample_text <- data_second_inning$FullNotes #capturing commentary/FullNotes data

  #performing sentiment analysis by using afinn method
```

```r
afinn_vector=get_sentiment(sample_text, method="afinn", lang="english")

#for each ball bowled in second inning, calculate resource usage and resource left
for(i in 1:nrow(data_second_inning))
{
  resource_left = (1-(data_second_inning[i,7]/6))*DLS[data_second_inning[i,6]+1,
                                                      data_second_inning[i,21]+1]
  +data_second_inning[i,7]/6*DLS[data_second_inning[i,6]+2,
                                 data_second_inning[i,21]+1]
  resource_used = DLS[data_second_inning[i,6]+1,data_second_inning[i,21]+1] -
    resource_left
  runs_needed_to_keep_up = c(runs_needed_to_keep_up,
                             round(resource_used * target, 3))
   # round to 4 digits
  resource_used_vector = c(resource_used_vector, round(resource_used, 4))
  resource_left_vector = c(resource_left_vector, round(resource_left, 4))
}
#initializing vector to contain score for each ball
ball_score=c(as.numeric(afinn_vector[1]))
highlight_balls = data.frame(cbind(data_second_inning,ball_score[1]))
 #renaming column to -> Score
colnames(highlight_balls)[dim(highlight_balls)[2]] = "Score"

for(i in 2:length(afinn_vector)) #for each ball
{
  #calculating emotional valence through sentiment analysis
  emotional_valence = abs(afinn_vector[i])
  #for each ball bowled in second inning, calc difference in resource usage and resource left
  resource_usage_difference = abs(resource_used_vector[i-1] - resource_used_vector[i])
  resource_left_difference = abs(resource_left_vector[i] - resource_left_vector[i-1])

  if(data_second_inning[i,15] == -1 ) #for wicket, assign higher score
    ball_score = c(ball_score, 2.5*(emotional_valence) + 100*resource_usage_difference)

  #for fours (num outcome = 5 included for four on a no-ball or wide), assign higher score
  else if(data_second_inning[i,15] >= 4  & data_second_inning[i,15] < 6)
    ball_score = c(ball_score, 1.5*(emotional_valence) + 100*resource_usage_difference)
  #for sixes (num outcome = 7 included for six on a no-ball or wide), assign higher score
  else if(data_second_inning[i,15] >= 6)
    ball_score = c(ball_score, 2.5*(emotional_valence) + 100*resource_usage_difference)
  else #assign lower score for other balls
    ball_score = c(ball_score, (emotional_valence + resource_usage_difference))
  highlight_balls[nrow(highlight_balls)+1,] = c(data_second_inning[i,], ball_score[i])

}
Final_highlight_balls = highlight_balls
Final_highlight_balls[,22] = highlight_balls[,22]

#sorting in decreasing order of score assigned to each ball
Final_highlight_balls = Final_highlight_balls[order(-Final_highlight_balls$Score),]
#taking top 20 balls with highest score in our highlights
Final_highlight_balls = Final_highlight_balls[1:20,]
#arranging rows in increasing order based on overs and balls
```

```
  Final_highlight_balls = Final_highlight_balls[order(Final_highlight_balls$Over,
                                                      Final_highlight_balls$Ball),]

  result_highlights = rbind(result_highlights,Final_highlight_balls)

}
```

Here, result_highlights is the dataframe containing 20 ball highlights/turning-points of each match provided in the dataset.

As a sample, we see below the 20-ball highlights for the match between Aus and BD.

```
result_highlights[which(result_highlights$MatchNo==161),]
```

```
##     Format MatchNo TeamBowling TeamBatting Inning Over Ball    Bowler BowlerID
## 119   T20I     161          AUS          BD      2    0    3   SW Tait       39
## 122   T20I     161          AUS          BD      2    0    6   SW Tait       39
## 125   T20I     161          AUS          BD      2    1    3 DP Nannes       66
## 138   T20I     161          AUS          BD      2    3    2 DP Nannes       66
## 141   T20I     161          AUS          BD      2    3    5 DP Nannes       66
## 149   T20I     161          AUS          BD      2    4    5   SW Tait       39
## 161   T20I     161          AUS          BD      2    6    5 MJ Clarke        5
## 169   T20I     161          AUS          BD      2    8    1 SPD Smith       46
## 178   T20I     161          AUS          BD      2    9    4 RJ Harris       52
## 180   T20I     161          AUS          BD      2    9    6 RJ Harris       52
## 184   T20I     161          AUS          BD      2   10    4 SPD Smith       46
## 188   T20I     161          AUS          BD      2   11    2 DJ Hussey       14
## 190   T20I     161          AUS          BD      2   11    4 DJ Hussey       14
## 197   T20I     161          AUS          BD      2   12    5 SPD Smith       46
## 206   T20I     161          AUS          BD      2   14    1 RJ Harris       52
## 212   T20I     161          AUS          BD      2   15    1 SPD Smith       46
## 218   T20I     161          AUS          BD      2   16    1 DP Nannes       66
## 222   T20I     161          AUS          BD      2   16    5 DP Nannes       66
## 223   T20I     161          AUS          BD      2   16    6 DP Nannes       66
## 233   T20I     161          AUS          BD      2   18    4 RJ Harris       52
##                Batsman BatsmanID  Fielder FielderID Outcome NumOutcome BallType
## 119       Imrul Kayes      1057                 NA       4          4 leg byes
## 122       Imrul Kayes      1057                 NA     OUT         -1      out
## 125 Mohammad Ashraful      1022     TAIT        39     OUT         -1      out
## 138       Aftab Ahmed      1002                 NA     OUT         -1      out
## 141       Mahmudullah        NA                 NA     OUT         -1      out
## 149   Mushfiqur Rahim      1044 ASHRAFUL      1022    FOUR          4      run
## 161   Shakib Al Hasan      1039                 NA     SIX          6      run
## 169   Shakib Al Hasan      1039                 NA       2          2      run
## 178   Mushfiqur Rahim      1044                 NA     SIX          6      run
## 180   Mushfiqur Rahim      1044                 NA    FOUR          4      run
## 184   Shakib Al Hasan      1039   HUSSEY        54     OUT         -1      out
## 188       Naeem Islam      1049                 NA     SIX          6      run
## 190       Naeem Islam      1049                 NA     OUT         -1      out
## 197     Jahurul Islam      1059                 NA     SIX          6      run
## 206     Shafiul Islam      1050                 NA    FOUR          4      run
## 212     Shafiul Islam      1050                 NA     SIX          6      run
## 218     Jahurul Islam      1059                 NA    FOUR          4      run
```

```
## 222     Jahurul Islam      1059                NA     OUT         -1      out
## 223  Mashrafe Mortaza      1019                NA    FOUR          4      run
## 233  Mashrafe Mortaza      1019                NA     OUT         -1      out
##     NumBallType      Notes
## 119           4       ooh,
## 122           0   expertly
## 125           0     taken!
## 138           0      well,
## 141           0 Bangladesh
## 149           0       good
## 161           0    Smashed
## 169           0    pitched
## 178           0      Shot!
## 180           0    another
## 184           0        Mr.
## 188           0      Naeem
## 190           0    another
## 197           0       they
## 206           0       nice
## 212           0      whoa!
## 218           0       Nice
## 222           0        The
## 223           0   thickish
## 233           0        And
##
## 119
## 122
## 125 taken! Great catch from Tait at third man. Ashraful played it well actually   angling the bat an
## 138                                          well   this is not helping Bangladesh's cause at all. A
## 141                                                Bangladesh are getting outclassed h
## 149
## 161
## 169
## 178
## 180
## 184                    Mr. Cricket earns his side a big wicket! Pitched up and Shakib swings it aw
## 188
## 190
## 197
## 206
## 212
## 218
## 222                                The old fat lady is clearing her throat. You can slog big Di
## 223
## 233
##     IDflag Wickets   Score
## 119      0       0 19.6700
## 122      0       0 26.6700
## 125      0       1 28.0500
## 138      0       2 28.7500
## 141      1       3 25.4900
## 149      0       4 17.5100
## 161      0       4 10.5600
## 169      0       4  5.5088
```

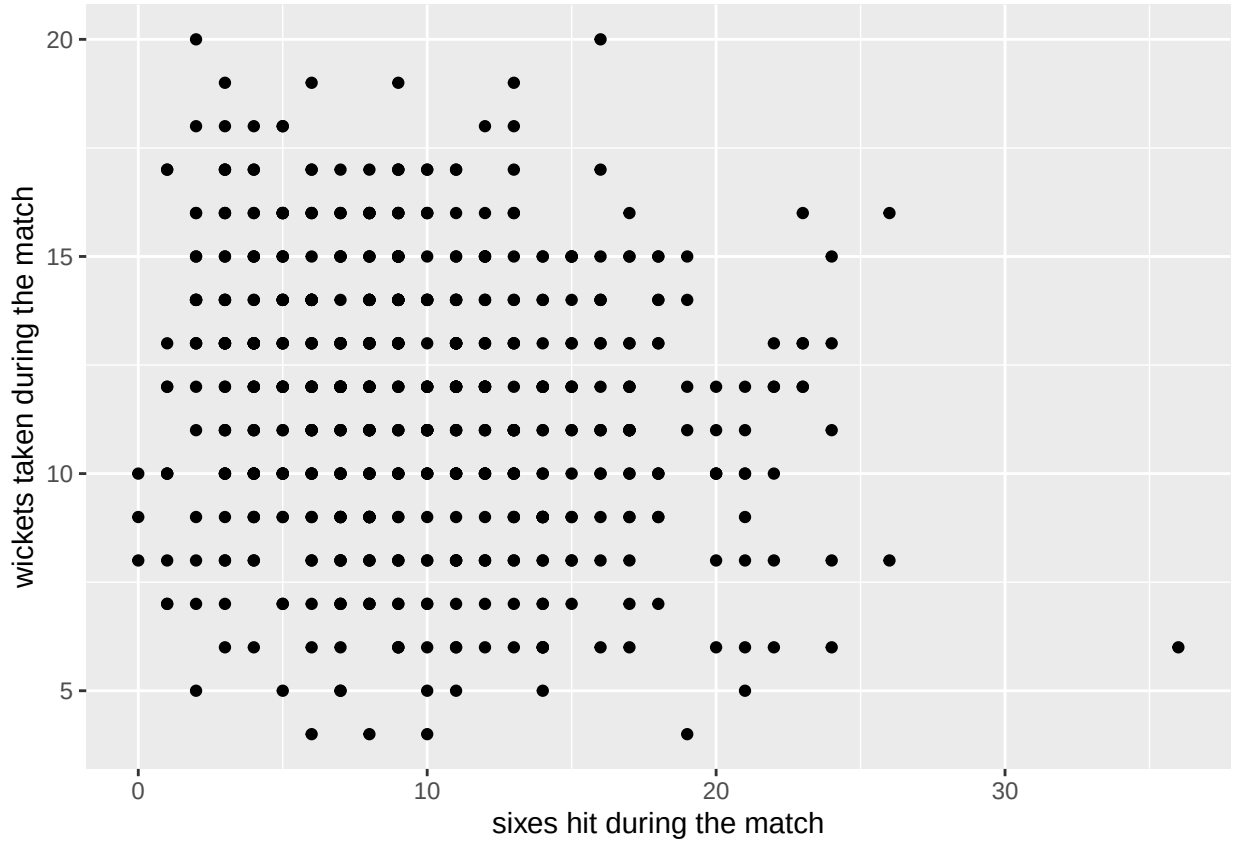
```
## 178      0       4 18.9500
## 180      0       4 13.4500
## 184      0       4 15.8300
## 188      0       5  9.6400
## 190      0       5 12.1300
## 197      0       6  8.3600
## 206      0       7 27.3000
## 212      0       7 23.4200
## 218      0       8 25.6300
## 222      0       8 12.4600
## 223      0       9 11.8300
## 233      0       9  6.0800
```

Part 3. Answer

Here, we are trying to analyze cluster of matches based on total wickets taken and total sixes hit during the match. We try to visualize the data using contour plot and scatter plot. We find that 10 sixes and around 12 wickets per match are the dominant figures in our data as seen in our plots below:

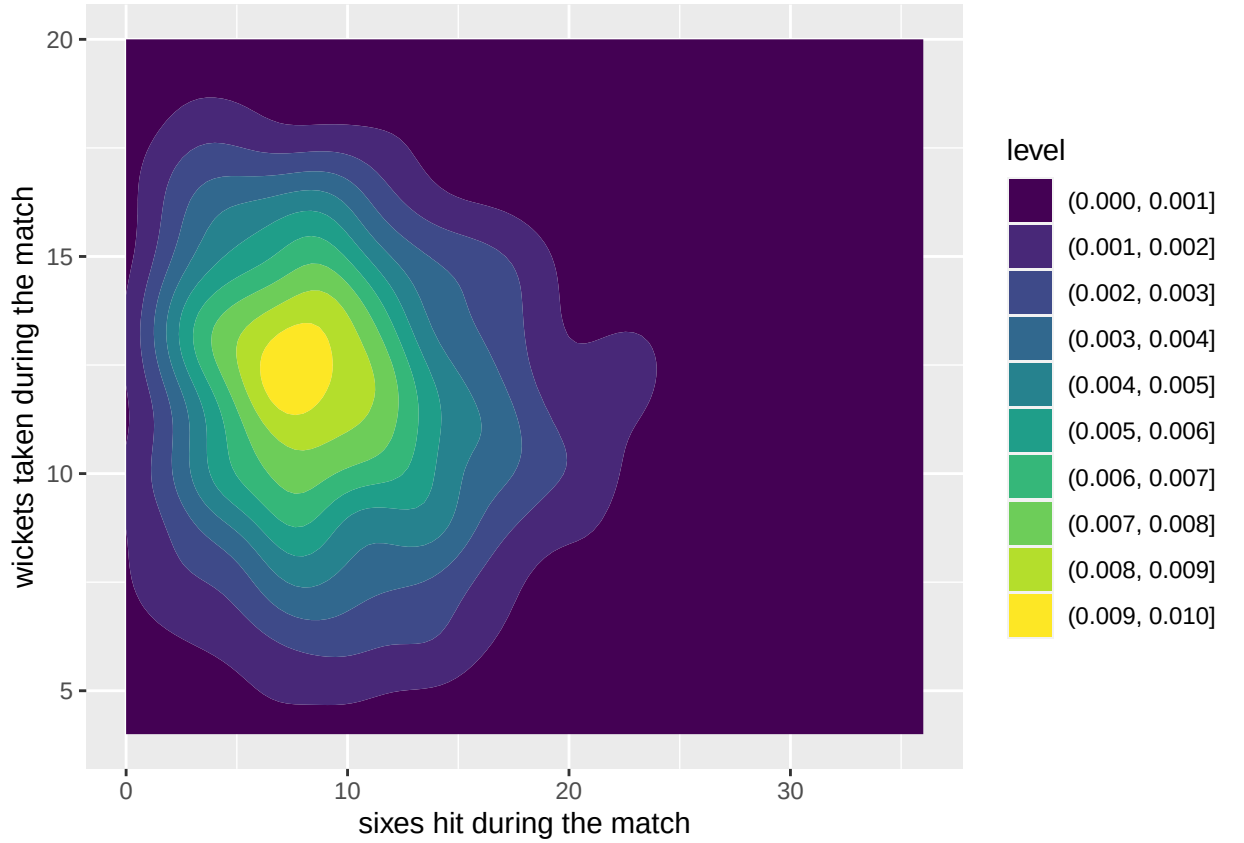
Scatter Plot

```
cluster_df = data2
df_matches = ddply(cluster_df, "MatchNo", summarize,
                   total_wickets = length(which(NumOutcome == -1)),
                   total_six = length(which(NumOutcome == 6 | NumOutcome ==7)),
                   total_four = length(which(NumOutcome == 4 | NumOutcome ==5)),
                   total_runs = sum(pmax(NumOutcome, 0, na.rm=TRUE))
)

gr1 <- ggplot(df_matches, aes(x = total_six,  y = total_wickets)) +
  geom_point() +
  ylab("wickets taken during the match") +
  xlab("sixes hit during the match")
plot(gr1)
```



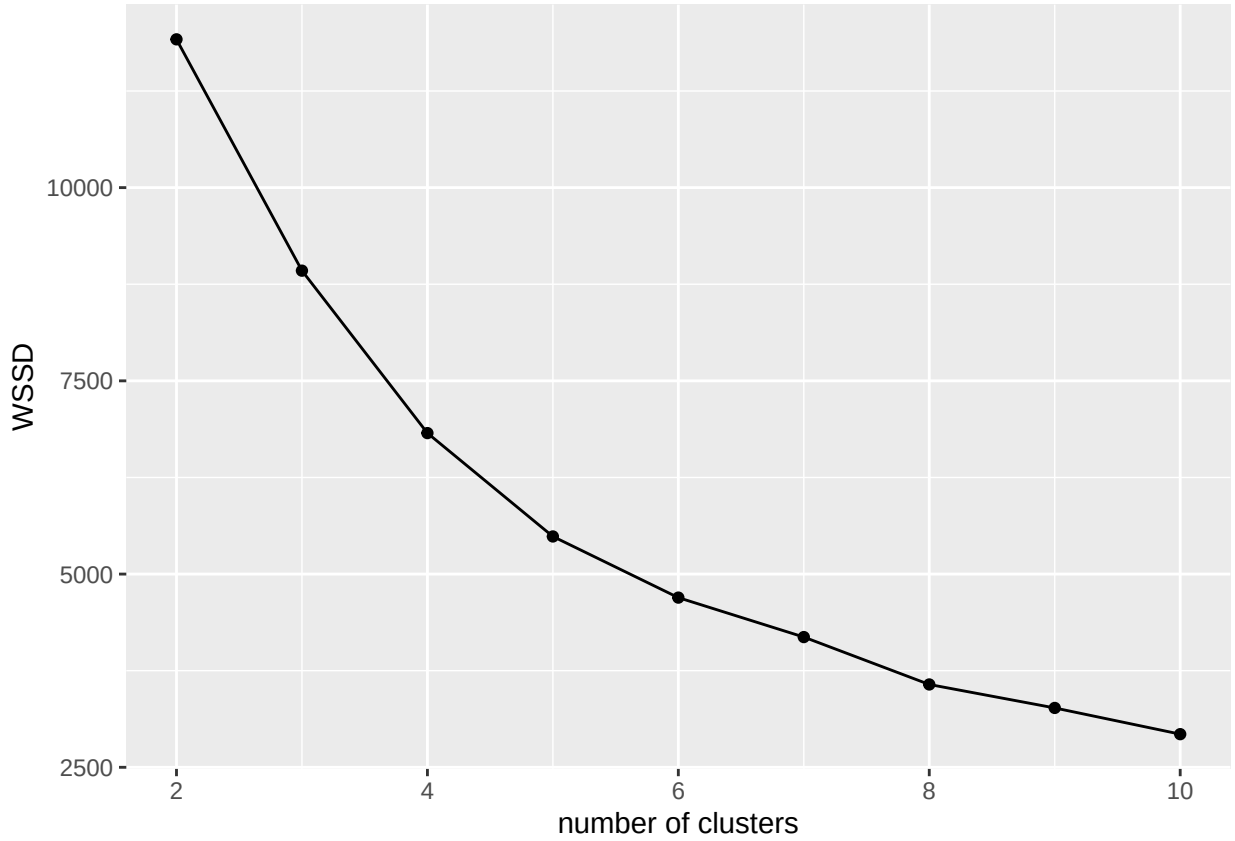
2D Contour Plot

```
gr2 <- ggplot(df_matches, aes(x = total_six,  y = total_wickets)) +
  geom_density_2d_filled() +
  ylab("wickets taken during the match") +
  xlab("sixes hit during the match")
plot(gr2)
```



Now, we proceed with K-means clustering. We look at within sum of squares distance before deciding on the number of clusters.

```
df_shot_risk = subset(df_matches, select = c(total_six, total_wickets))
wssd <- rep(NA,9)
for(k in 2:10) {
  shot_clust <- kmeans(df_shot_risk, centers = k)
  wssd[k-1] <- shot_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)
```



From the elbow plot, either 5 or 6 clusters provides an ideal balance between parsimony and goodness-of-fit. We'll try 5 clusters for our initial analysis. This is because this is the point where diminishing returns are no longer worth the additional cost.

```
# We can take 5 clusters in our case as can be seen from elbow chart

shot_clust_5 <- kmeans(df_shot_risk, centers = 5)
shot_clust_5$centers
```

```
##   total_six total_wickets
## 1  5.433333      9.025000
## 2 19.183908     11.022989
## 3  4.628571     14.164286
## 4 12.064516      9.064516
## 5 10.379085     14.065359
```

The relative amounts of diffusion and the sizes of each of these clusters is as below:

```
msd <- sqrt(shot_clust_5$withinss / shot_clust_5$size)
msd
```

```
## [1] 2.831948 4.268316 2.582782 2.742979 2.668332
```

```
shot_clust_5$size
```

```
## [1] 120  87 140 124 153
```

Part 4. Answer

Using loss function, we come up with a new DLS table based on the dataset provided. Care has been taken to ensure it follows the following attributes: 1. ranges 0 to 1 2. monotonic in wickets and in overs 3. has some non-linearity 4. only based on overs and wicket

```
DLT = read.csv("DLS_T20.csv")[,-1]
dls_df = data.frame(subset(data2, Inning==1)) #First inning data filtered

dls_df$over2 = dls_df[,6] + dls_df[,7]/6 #reading overs column
dls_df$Nruns = pmax(dls_df[,15], 0) #reading runs column

dls_df_result = data.frame()

#function to calculate proportion of target achieved in each ball
func_calc_prop = function(match)
{
  df_temp = data.frame()
  df_temp = data.frame(subset(dls_df, MatchNo==match))
  target = sum(df_temp$Nruns) #target

  #using cumulative sum to calc prop of target achieved after each ball
  for(i in 1:nrow(df_temp))
  {
    df_temp$cum = cumsum(df_temp$Nruns)
    df_temp$prop[i] = df_temp$cum[i]/target
  }
  return(rbind(dls_df_result,df_temp))
}

for(match in match_ids) #calling above function
  dls_df_result = func_calc_prop(match)

head(dls_df_result)
```

```
##   Format MatchNo TeamBowling TeamBatting Inning Over Ball Bowler BowlerID
## 1   T20I      33         AUS          BD      1    0    1  B Lee       17
## 2   T20I      33         AUS          BD      1    0    2  B Lee       17
## 3   T20I      33         AUS          BD      1    0    3  B Lee       17
## 4   T20I      33         AUS          BD      1    0    4  B Lee       17
## 5   T20I      33         AUS          BD      1    0    4  B Lee       17
## 6   T20I      33         AUS          BD      1    0    5  B Lee       17
##       Batsman BatsmanID Fielder FielderID Outcome NumOutcome BallType
## 1 Tamim Iqbal      1041                NA      no          0      run
## 2 Tamim Iqbal      1041                NA      no          0      run
## 3 Tamim Iqbal      1041                NA      no          0      run
## 4 Tamim Iqbal      1041                NA       1          1     wide
## 5 Tamim Iqbal      1041                NA      no          0      run
```

```
## 6 Tamim Iqbal      1041                NA      no          0      run
##   NumBallType  Notes
## 1           0   good
## 2           0  short
## 3           0  short
## 4           2  Tamim
## 5           0 fuller
## 6           0   good
##
## 1                                          good start by Lee   dug in short of a length outside of
## 2                                       short of a length outside off again   this time Tamim gets
## 3                                          short again and aimed at the body   Tamim gets on the ba
## 4 Tamim backs away to whack that over the off side   Lee senses it and thuds it in short, the ball sa
## 5                                           fuller in length and inviting the drive   Tamim flash
## 6                                                                                 good length aimed at
##   IDflag Wickets     over2 Nruns cum        prop
## 1      0       0 0.1666667     0   0 0.000000000
## 2      0       0 0.3333333     0   0 0.000000000
## 3      0       0 0.5000000     0   0 0.000000000
## 4      0       0 0.6666667     1   1 0.008264463
## 5      0       0 0.6666667     0   1 0.008264463
## 6      0       0 0.8333333     0   1 0.008264463
```

```
#preparing optimization

over2 = dls_df_result$over2
wicket = dls_df_result$Wickets
prop = dls_df_result$prop

#loss function
loss_function = function(x, prop) {
  A = x[1]
  B = x[2]
  C = x[3]
  D = x[4]
  prop_smooth = A*over2^2 + B*(10-wicket)^2 + C*over2^3 + D*over2*(10-wicket)
  error = sum((prop - prop_smooth)^2)
  return(error)
}


best_params = optim(par=c(0,0,0,0), loss_function, prop=prop)$par
A = best_params[1]
B = best_params[2]
C = best_params[3]
D = best_params[4]
#Make a matrix. 20 rows for 20 overs, 10 columns for the 0-9 wickets taken # NA for cell values to star
# we know if we missed something because it will still be NA
newDLT = matrix(NA, nrow=20, ncol=10)
# Compute the matrix row by row, where each row is an over
for(overcount in 1:20) {
  # Apply the example formula. 1 - (formula) because resource = 1 - proportion.
  newDLT[overcount,] = 1 - (A*overcount^2 + B*(0:9)^2 + C*overcount^3 + D*overcount*(0:9))
  }
```

```r
# Compare
newDLT
```

```
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
##  [1,] 0.99434816 0.99366386 0.99112472 0.98673074 0.98048192 0.97237825
##  [2,] 0.97809172 0.97765054 0.97535453 0.97120367 0.96519797 0.95733743
##  [3,] 0.95227930 0.95208126 0.95002836 0.94612063 0.94035806 0.93274064
##  [4,] 0.91795955 0.91800462 0.91619486 0.91253025 0.90701079 0.89963650
##  [5,] 0.87618107 0.87646927 0.87490263 0.87148114 0.86620482 0.85907365
##  [6,] 0.82799251 0.82852383 0.82720031 0.82402195 0.81898875 0.81210070
##  [7,] 0.77444248 0.77521693 0.77413653 0.77120130 0.76641122 0.75976630
##  [8,] 0.71657962 0.71759719 0.71675992 0.71406781 0.70952085 0.70311906
##  [9,] 0.65545254 0.65671324 0.65611910 0.65367011 0.64936628 0.64320761
## [10,] 0.59210989 0.59361371 0.59326269 0.59105683 0.58699613 0.58108058
## [11,] 0.52760029 0.52934723 0.52923934 0.52727660 0.52345902 0.51778659
## [12,] 0.46297236 0.46496243 0.46509766 0.46337804 0.45980358 0.45437428
## [13,] 0.39927473 0.40150792 0.40188627 0.40040978 0.39707845 0.39189228
## [14,] 0.33755603 0.34003235 0.34065382 0.33942046 0.33633225 0.33138920
## [15,] 0.27886488 0.28158432 0.28244893 0.28145868 0.27861360 0.27391367
## [16,] 0.22424992 0.22721249 0.22832021 0.22757309 0.22497114 0.22051433
## [17,] 0.17475977 0.17796546 0.17931631 0.17881232 0.17645348 0.17223980
## [18,] 0.13144305 0.13489187 0.13648584 0.13622498 0.13410926 0.13013871
## [19,] 0.09534841 0.09904035 0.10087745 0.10085970 0.09898711 0.09525968
## [20,] 0.06752445 0.07145952 0.07353974 0.07376512 0.07213566 0.06865135
##             [,7]       [,8]       [,9]      [,10]
##  [1,] 0.96241974 0.95060639 0.93693819 0.92141516
##  [2,] 0.94762204 0.93605182 0.92262675 0.90734683
##  [3,] 0.92326838 0.91194127 0.89875933 0.88372254
##  [4,] 0.89040736 0.87932339 0.86638456 0.85159090
##  [5,] 0.85008764 0.83924678 0.82655108 0.81200054
##  [6,] 0.80335782 0.79276009 0.78030751 0.76600010
##  [7,] 0.75126653 0.74091193 0.72870248 0.71463819
##  [8,] 0.69486242 0.68475094 0.67278461 0.65896345
##  [9,] 0.63519409 0.62532574 0.61360254 0.60002449
## [10,] 0.57331019 0.56368495 0.55220488 0.53886996
## [11,] 0.51025933 0.50087722 0.48964027 0.47654847
## [12,] 0.44709014 0.43795116 0.42695733 0.41410866
## [13,] 0.38485126 0.37595540 0.36520470 0.35259915
## [14,] 0.32459130 0.31593857 0.30543099 0.29306857
## [15,] 0.26735890 0.25894929 0.24868484 0.23656554
## [16,] 0.21420269 0.20603620 0.19601487 0.18413870
## [17,] 0.16617128 0.15824792 0.14846971 0.13683667
## [18,] 0.12431332 0.11663308 0.10709800 0.09570807
## [19,] 0.08967741 0.08224030 0.07294834 0.06180154
## [20,] 0.06331220 0.05611821 0.04706938 0.03616571
```

Smoothing out values to get our final DLS Table.

```r
#smoothing out values
for(loopcount in 1:10)
{

  temp = rbind(newDLT[2:20,], rep(0, 10))
```

```
  newDLT = pmax(temp, newDLT)
}

range = max(newDLT) - min(newDLT)
newDLT2 = (newDLT - min(newDLT)) / range
newDLT2
```

```
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
##  [1,] 1.00000000 0.99928584 0.99663588 0.99205014 0.98552860 0.97707127
##  [2,] 0.98303409 0.98257366 0.98017744 0.97584543 0.96957762 0.96137403
##  [3,] 0.95609515 0.95588846 0.95374598 0.94966770 0.94365363 0.93570377
##  [4,] 0.92027759 0.92032464 0.91843589 0.91461134 0.90885101 0.90115488
##  [5,] 0.87667580 0.87697658 0.87534156 0.87177075 0.86626415 0.85882176
##  [6,] 0.82638416 0.82693867 0.82555739 0.82224032 0.81698746 0.80979880
##  [7,] 0.77049707 0.77130532 0.77017777 0.76711444 0.76211531 0.75518038
##  [8,] 0.71010892 0.71117091 0.71029710 0.70748749 0.70274210 0.69606091
##  [9,] 0.64631411 0.64762983 0.64700975 0.64445388 0.63996222 0.63353477
## [10,] 0.58020702 0.58177647 0.58141013 0.57910800 0.57487008 0.56869636
## [11,] 0.51288205 0.51470524 0.51459263 0.51254424 0.50856005 0.50264006
## [12,] 0.44543359 0.44751051 0.44765164 0.44585698 0.44212652 0.43646028
## [13,] 0.37895603 0.38128669 0.38168155 0.38014063 0.37666391 0.37125139
## [14,] 0.31454377 0.31712816 0.31777676 0.31648957 0.31326658 0.30810780
## [15,] 0.25329119 0.25612932 0.25703165 0.25599819 0.25302894 0.24812390
## [16,] 0.19629269 0.19938455 0.20054062 0.19976090 0.19704538 0.19239408
## [17,] 0.14464266 0.14798826 0.14939806 0.14887208 0.14641029 0.14201272
## [18,] 0.09943550 0.10303483 0.10469837 0.10442611 0.10221807 0.09807423
## [19,] 0.06176559 0.06561865 0.06753593 0.06751741 0.06556310 0.06167299
## [20,] 0.03272732 0.03683412 0.03900513 0.03924035 0.03753977 0.03390340
##             [,7]       [,8]       [,9]      [,10]
##  [1,] 0.96667814 0.95434923 0.94008452 0.92388402
##  [2,] 0.95123464 0.93915946 0.92514848 0.90920172
##  [3,] 0.92581811 0.91399667 0.90023943 0.88454640
##  [4,] 0.89152296 0.87995525 0.86645175 0.85101245
##  [5,] 0.84944358 0.83812960 0.82487983 0.80969427
##  [6,] 0.80067435 0.78961411 0.77661807 0.76168624
##  [7,] 0.74630967 0.73550316 0.72276086 0.70808277
##  [8,] 0.68744393 0.67689116 0.66440259 0.64997824
##  [9,] 0.62517153 0.61487249 0.60263766 0.58846704
## [10,] 0.56058685 0.55054155 0.53856045 0.52464356
## [11,] 0.49478429 0.48499272 0.47326536 0.45960221
## [12,] 0.42885824 0.41932040 0.40784678 0.39443736
## [13,] 0.36390309 0.35461899 0.34339910 0.33024342
## [14,] 0.30101323 0.29198287 0.28101671 0.26811477
## [15,] 0.24128306 0.23250644 0.22179401 0.20914580
## [16,] 0.18580697 0.17728408 0.16682539 0.15443092
## [17,] 0.13567935 0.12741020 0.11720524 0.10506450
## [18,] 0.09199460 0.08397917 0.07402796 0.06214095
## [19,] 0.05584710 0.04808541 0.03838792 0.02675465
## [20,] 0.02833124 0.02082329 0.01137954 0.00000000
```

Final DLT table values, rounding off to 3 decimal places

```
# Final results of our DLT table values
round(newDLT2,3)
```

```
##        [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
##  [1,] 1.000 0.999 0.997 0.992 0.986 0.977 0.967 0.954 0.940 0.924
##  [2,] 0.983 0.983 0.980 0.976 0.970 0.961 0.951 0.939 0.925 0.909
##  [3,] 0.956 0.956 0.954 0.950 0.944 0.936 0.926 0.914 0.900 0.885
##  [4,] 0.920 0.920 0.918 0.915 0.909 0.901 0.892 0.880 0.866 0.851
##  [5,] 0.877 0.877 0.875 0.872 0.866 0.859 0.849 0.838 0.825 0.810
##  [6,] 0.826 0.827 0.826 0.822 0.817 0.810 0.801 0.790 0.777 0.762
##  [7,] 0.770 0.771 0.770 0.767 0.762 0.755 0.746 0.736 0.723 0.708
##  [8,] 0.710 0.711 0.710 0.707 0.703 0.696 0.687 0.677 0.664 0.650
##  [9,] 0.646 0.648 0.647 0.644 0.640 0.634 0.625 0.615 0.603 0.588
## [10,] 0.580 0.582 0.581 0.579 0.575 0.569 0.561 0.551 0.539 0.525
## [11,] 0.513 0.515 0.515 0.513 0.509 0.503 0.495 0.485 0.473 0.460
## [12,] 0.445 0.448 0.448 0.446 0.442 0.436 0.429 0.419 0.408 0.394
## [13,] 0.379 0.381 0.382 0.380 0.377 0.371 0.364 0.355 0.343 0.330
## [14,] 0.315 0.317 0.318 0.316 0.313 0.308 0.301 0.292 0.281 0.268
## [15,] 0.253 0.256 0.257 0.256 0.253 0.248 0.241 0.233 0.222 0.209
## [16,] 0.196 0.199 0.201 0.200 0.197 0.192 0.186 0.177 0.167 0.154
## [17,] 0.145 0.148 0.149 0.149 0.146 0.142 0.136 0.127 0.117 0.105
## [18,] 0.099 0.103 0.105 0.104 0.102 0.098 0.092 0.084 0.074 0.062
## [19,] 0.062 0.066 0.068 0.068 0.066 0.062 0.056 0.048 0.038 0.027
## [20,] 0.033 0.037 0.039 0.039 0.038 0.034 0.028 0.021 0.011 0.000
```

As we can see above, our values are pretty close to the values in the original DLT table which is present below.

```
#Actual DLT
DLT
```

```
##       X0    X1    X2    X3    X4    X5    X6    X7    X8    X9
## 1  1.000 0.968 0.926 0.867 0.788 0.682 0.544 0.375 0.210 0.083
## 2  0.961 0.933 0.892 0.839 0.767 0.666 0.535 0.373 0.210 0.083
## 3  0.922 0.896 0.859 0.811 0.742 0.650 0.527 0.369 0.210 0.083
## 4  0.882 0.857 0.825 0.779 0.717 0.633 0.516 0.366 0.210 0.083
## 5  0.841 0.818 0.790 0.747 0.691 0.613 0.504 0.362 0.208 0.083
## 6  0.799 0.779 0.753 0.716 0.664 0.592 0.491 0.357 0.208 0.083
## 7  0.754 0.737 0.714 0.680 0.634 0.569 0.477 0.352 0.208 0.083
## 8  0.710 0.694 0.673 0.645 0.604 0.544 0.461 0.345 0.207 0.083
## 9  0.664 0.650 0.633 0.606 0.571 0.519 0.443 0.336 0.205 0.083
## 10 0.617 0.604 0.590 0.567 0.537 0.491 0.424 0.327 0.203 0.083
## 11 0.567 0.558 0.544 0.527 0.500 0.461 0.403 0.316 0.201 0.083
## 12 0.518 0.511 0.498 0.484 0.461 0.428 0.378 0.302 0.198 0.083
## 13 0.466 0.459 0.451 0.438 0.420 0.394 0.352 0.286 0.193 0.083
## 14 0.413 0.408 0.401 0.392 0.378 0.355 0.322 0.269 0.186 0.083
## 15 0.359 0.355 0.350 0.343 0.332 0.314 0.290 0.246 0.178 0.081
## 16 0.304 0.030 0.297 0.292 0.284 0.272 0.253 0.221 0.166 0.081
## 17 0.246 0.244 0.242 0.239 0.233 0.224 0.212 0.189 0.148 0.080
## 18 0.187 0.186 0.184 0.182 0.180 0.175 0.168 0.154 0.127 0.074
## 19 0.127 0.125 0.125 0.124 0.124 0.120 0.117 0.110 0.097 0.065
## 20 0.064 0.064 0.064 0.064 0.064 0.062 0.062 0.060 0.057 0.044
## 21 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
```