STAT 5703

Data Mining I

Final Course Project Report

PLAYERUNKNOWN'S BATTLEGROUNDS Player Statistics

21st Dec, 2018

Submitted By:
Shubham Agarwal(101066272)
Mandeep Singh Bal (101089294)
Gurveen Kaur (101088497)
Vidhi Ashvinbhai Ghiya(101088148)

Submitted To:
Dr. Shirley E. Mills

Table of Contents

1. Introduction

Player Unknown's Battleground also known as PUBG is the online multiplayer survival-shooter royal game. It is developed and published by the PUBG Corporation. It is played by 100 players throughout the world where player fight to the death. Weapons, items and different useful tools are scattered all around the world for the players to search out for. As the time passes, the playable area of the map gets smaller, hence creating more and more conflicts [1][2].

We, here are trying the analyze and explore the behavior of the player to see what kind of strategies player use and using which strategies player often win. We focus on the analysis of PUBG player statistics.

2. Data Collection

We collected the data from the following link:

https://www.kaggle.com/lazyjustin/pubgplayerstats

The dataset we use has 87989 random players and their game statistics. There are 152 features/player. There are different modes of the game like: solo mode, duo mode and the squad mode.

As there are many features, we won't be using all of them. Categorizing the features:

- **Identifiers:** Player-Name, Tracker ID
- **Performance**: Rating, Win Ratio, Top 10 Ratio, No of Wins, kill-Death ratio.
- **Combat:** Kills, Assists, Headshots, Damage Dealt (all these variables per game)
- **Health:** Heals, Revives, Boosts, DBNO (down but not out)
- **Movement:** Vehicle-Distance, Walk-Distance
- **Time:** Time Survived
- **Achievements:** Daily-Kills, Weekly-Kills, Longest-Kill, Max-Kill Streaks

2.1. Flow

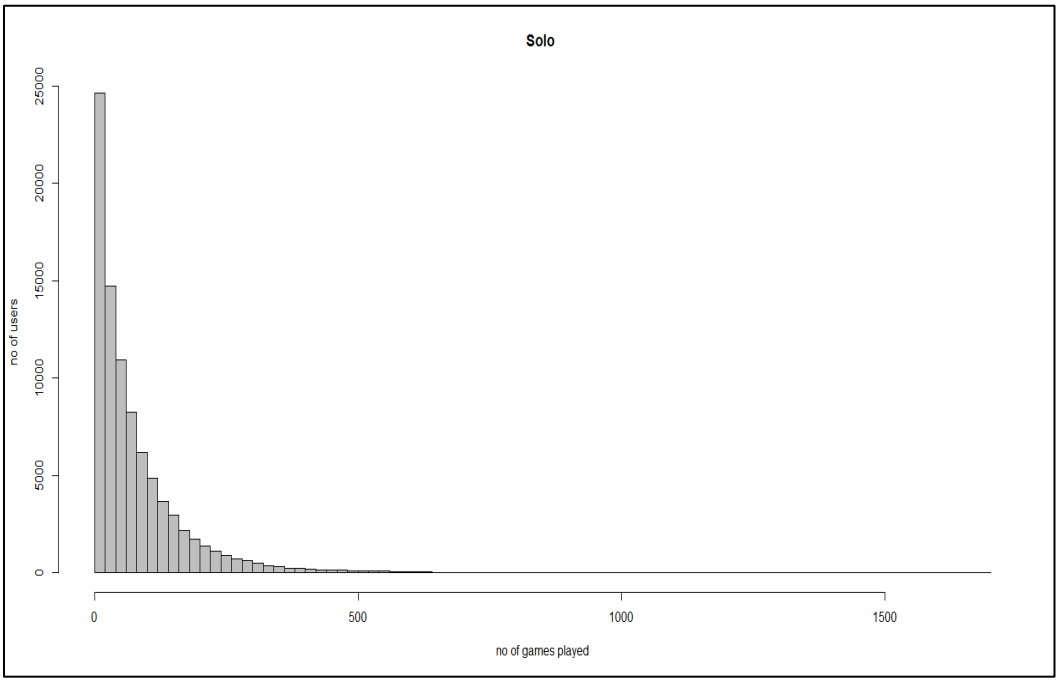
First, we filter-out only feature that are related to their selected mode (solo, duo and squad mode).

Then we have to remove the new players, so we filtered the players who played more than 50 games/round.

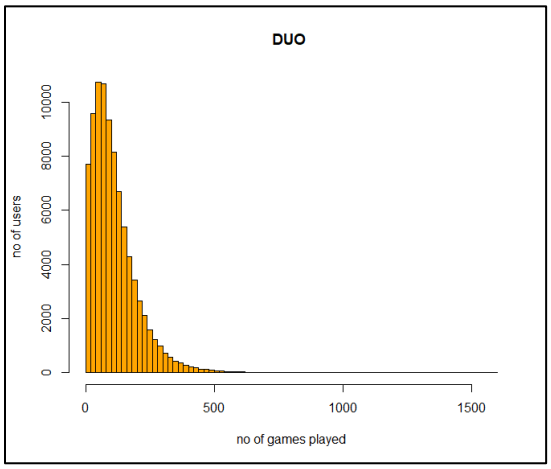
```
> solo_data1 <- subset(solo_data, solo_RoundsPlayed >=50)
> dim(solo_data1)
[1] 43199    52
> duo_data1 <- subset(duo_data, duo_RoundsPlayed >=50)
> dim(duo_data1)
[1] 65701    52
> squad_data1 <- subset(squad_data, squad_RoundsPlayed >=50)
> dim(squad_data1)
[1] 76823    52
```

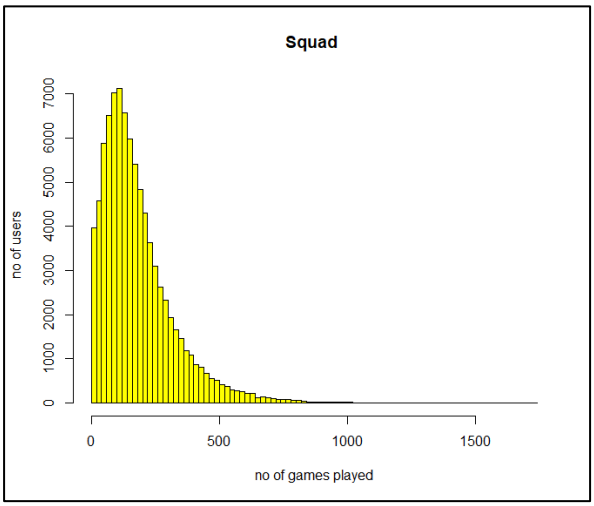
2.2. Data Visualization

After reducing the data according to their playing mode let's check how many games are played in individual mode.



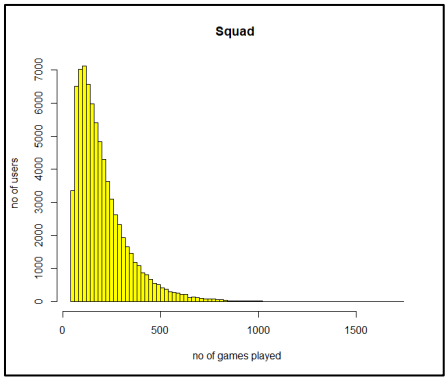
Similarly, we can check for Squad and Duo mode.



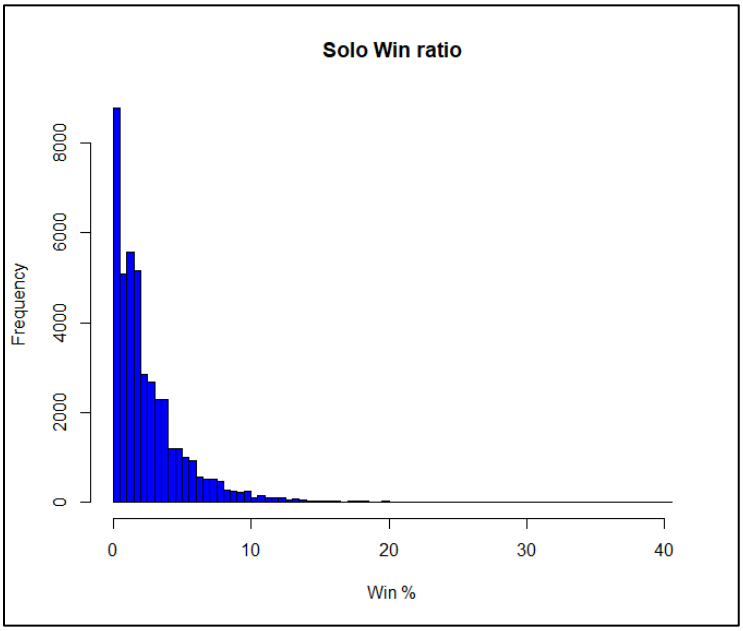


As it can be seen from the histograms that there are many data entries in the dataset where the person has not played that particular mode and also some persons are the beginner or say that they have played very few games in that mode, so we will remove some data from each mode as they would not make impact on the visualization.

So, for our visualization we will consider data for players who have played more than 50 games mode. After removing the players less than 50 games played this is how the histogram looks

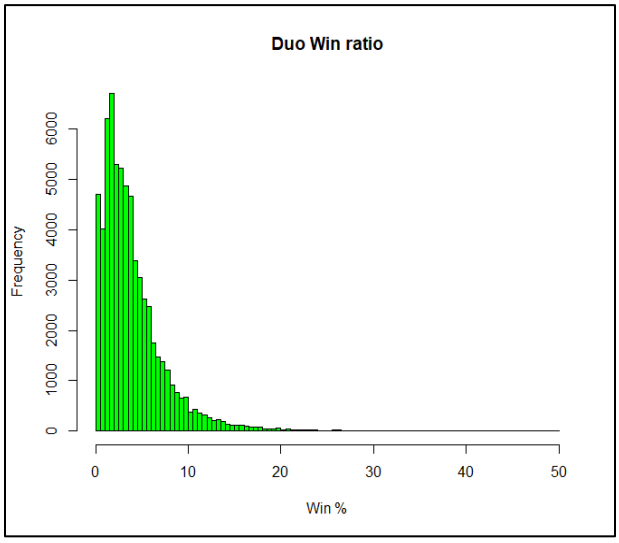


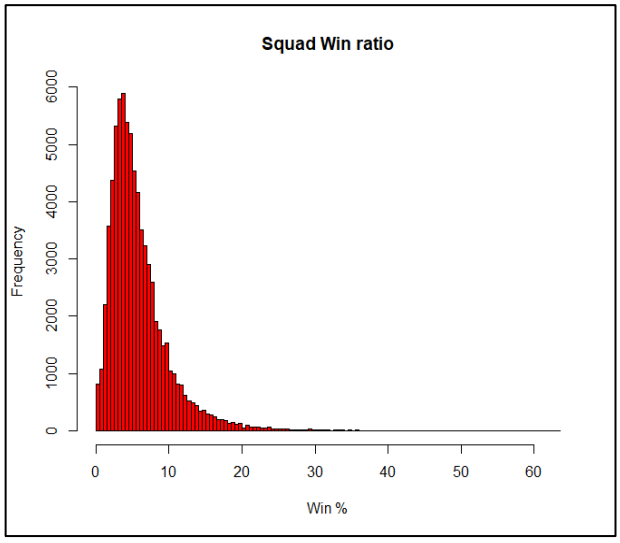
Now we will do the analysis of the dataset for each mode. Firstly, we will analyze the winning ratio for each mode and see how they varies.



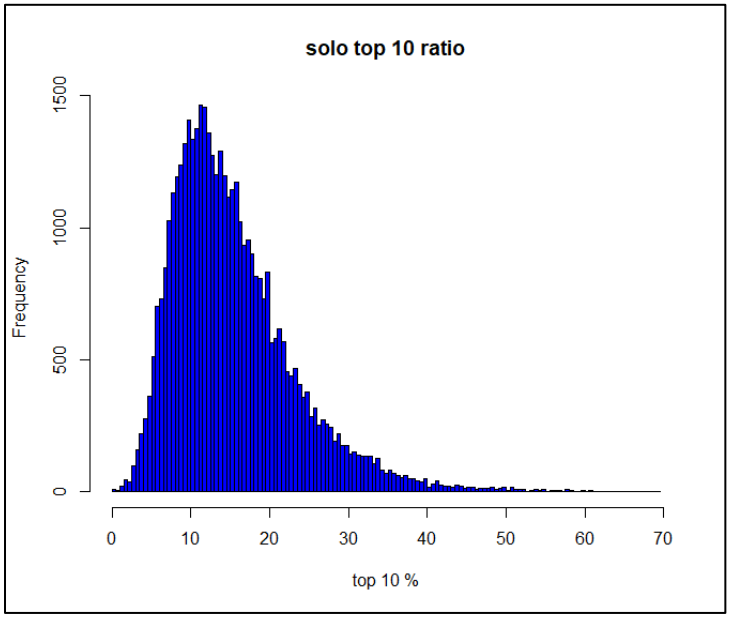
As the histogram displays, we can say that the winning percent when you are playing solo is very less and most of the player have winning percent less than 10% and only few in so much data have winning percent more than 10%. It can also be seen that most of the players in the dataset have 0%-win ratio as there is drastic drop down in more than 0%-win ratio that is almost 4000 frequency gaps. The mean is around 4% which makes the game to be seen more challenging.

Similarly, histograms for Duo and Squad are plotted

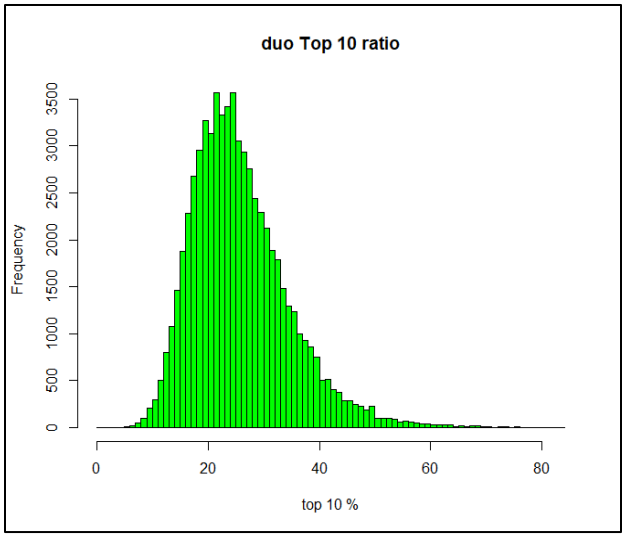


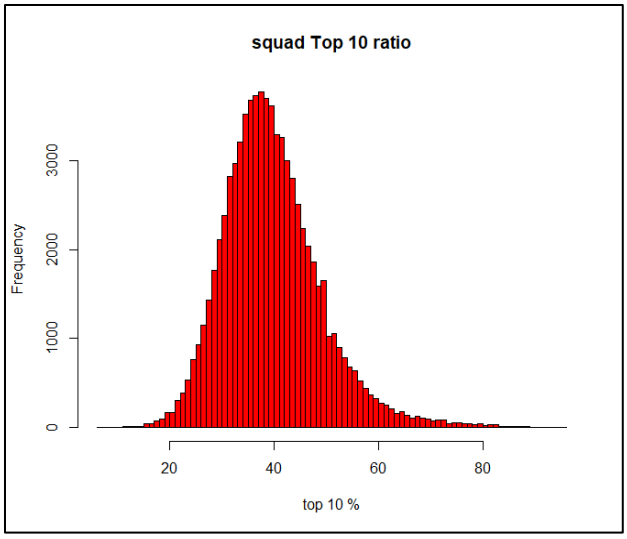


As compared to solo dataset duo and squad have maximum Win Ratio somewhere near 5-8 %. In duo mode still the 0% win ratio is still very high whereas in squad it is very low by which we can say that when a player plays in a team his chance of winning are high compared to other two modes. Also, the maximum win ratio value has increased in both modes and specially in squad mode win ratio percent is gradually spread. So, if you want to win you should prefer playing in a team so that your statistics for winning will be good.

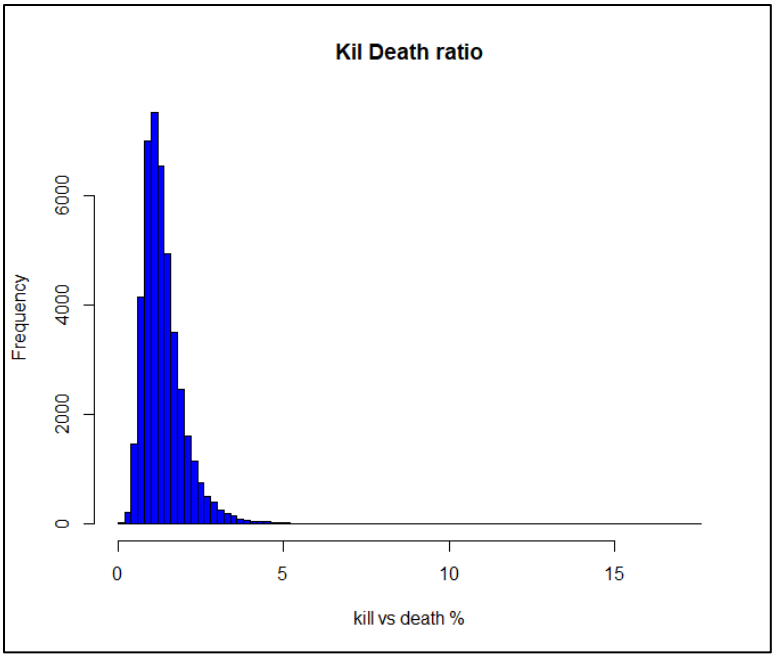


For top 10 ratio of a player finishing in top 10 positions for a particular round counts towards the top 10 ratio. So from the picture above we can visualize that, for solo player percentage of player finishing in top 10 position is maximum between 10-20. For player to be in top 10 for more number of games is difficult or we can say less.

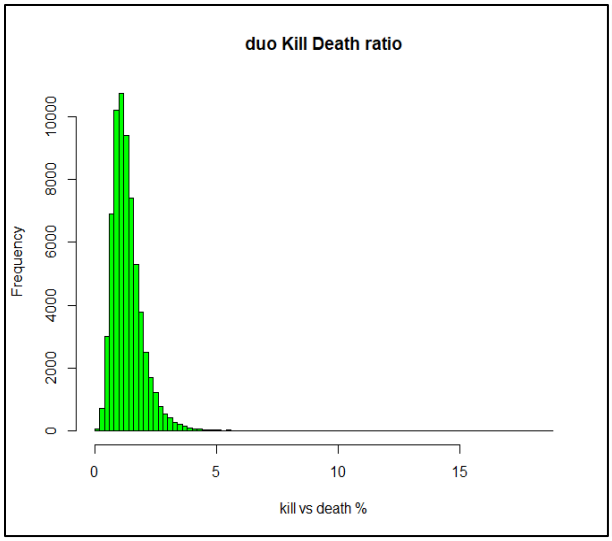


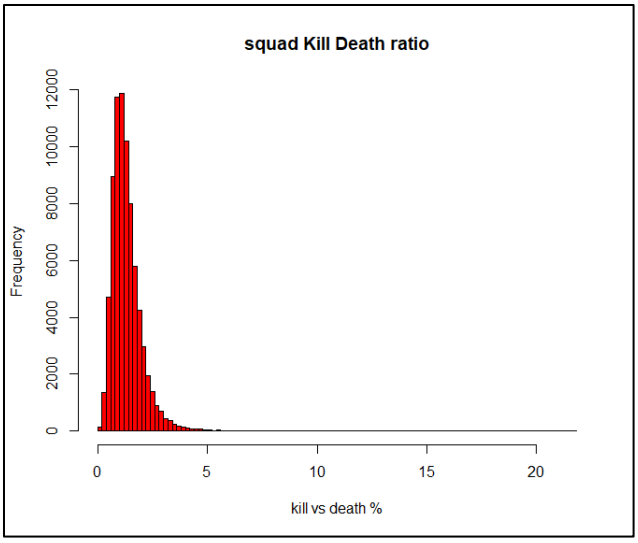


When we compare the top 10 ratios for all 3 types of mode, we can say that when the player is playing in duo and squad mode the maximum probability of player being in top 10 is between 20-40 and 20-60 respectively. So if player wants to be in top 10 for more number of times to make his/her statistics look good he might try playing in squad or duo mode.

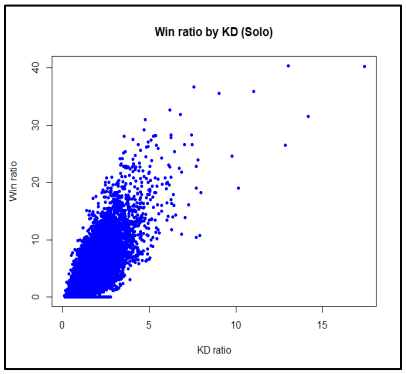


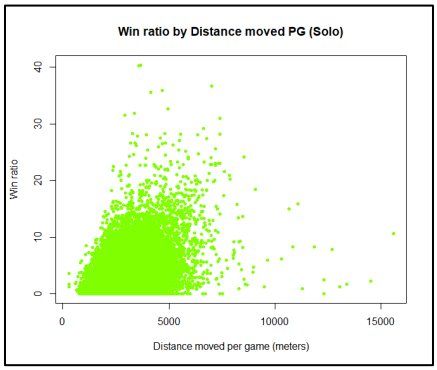
Kill to Death ratio means for the number of games played by player number of people killed to the number of times he died. Form the graph the kill to death ratio is between 0-5 and around 8000 people have the ratio of about 2.5%. Similarly for duo and Squad mode the frequency is around 10000 and 12000 respectively.

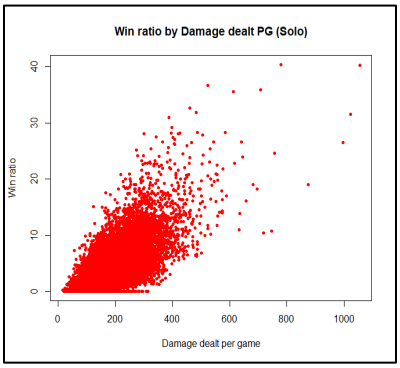




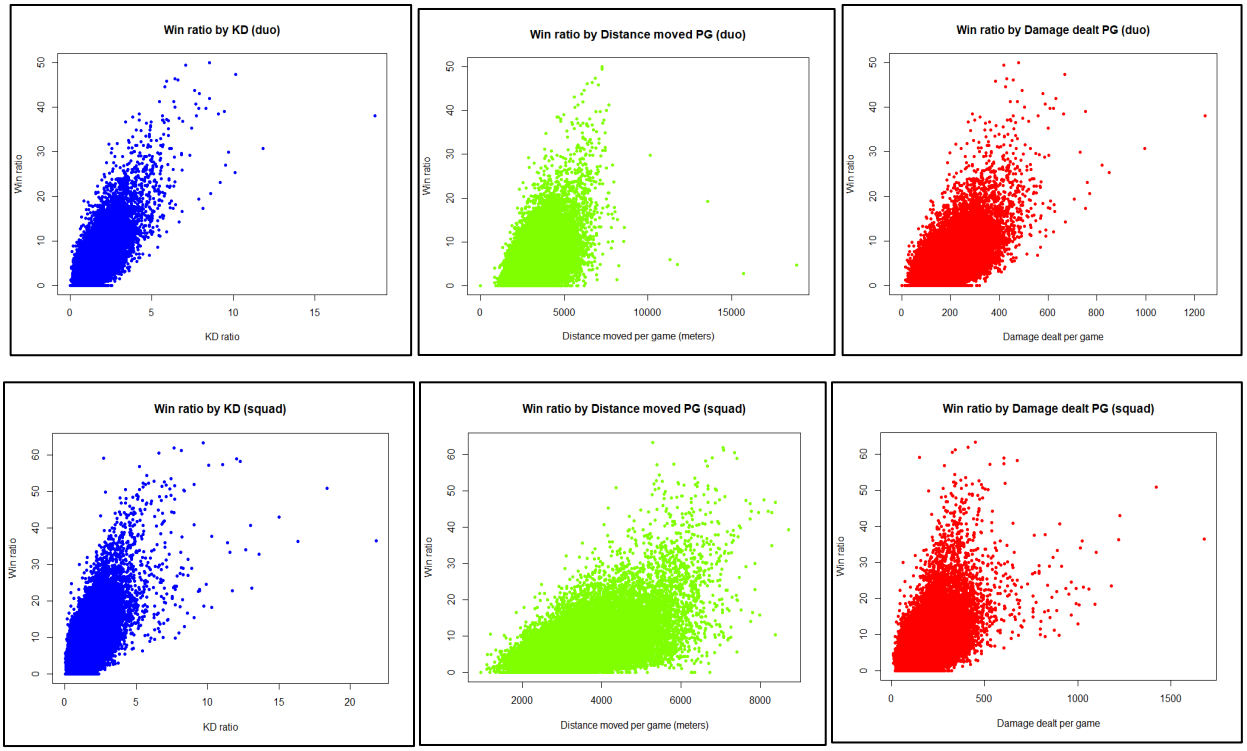
We can see the effect on winning ratio through 3 different variables: Kill to death ratio, Distance moved per game and Damage dealt per game for solo mode.



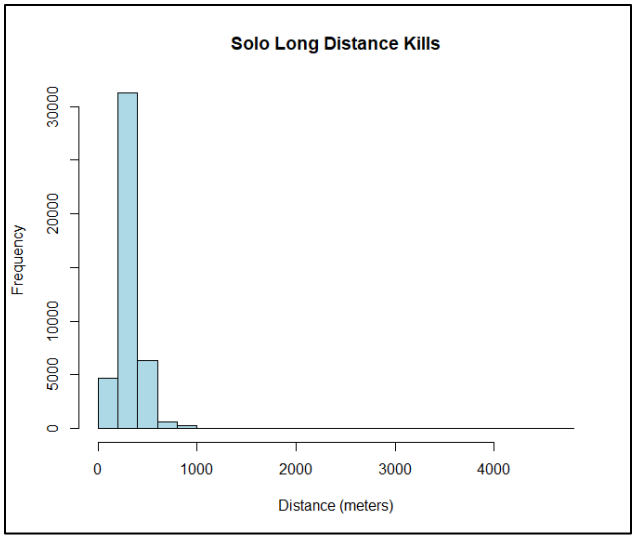


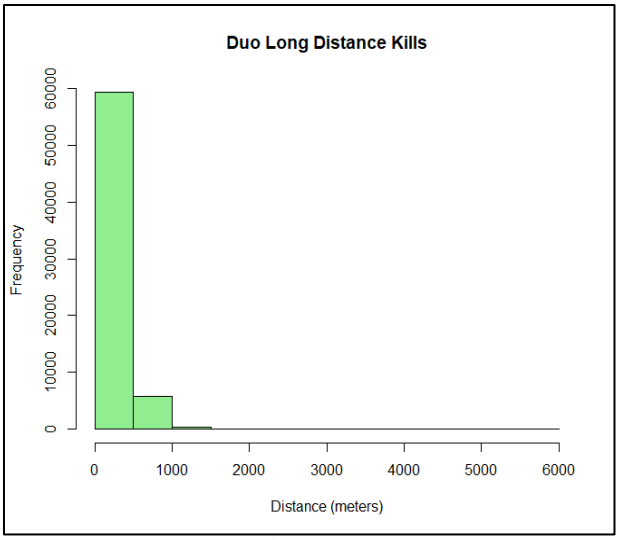


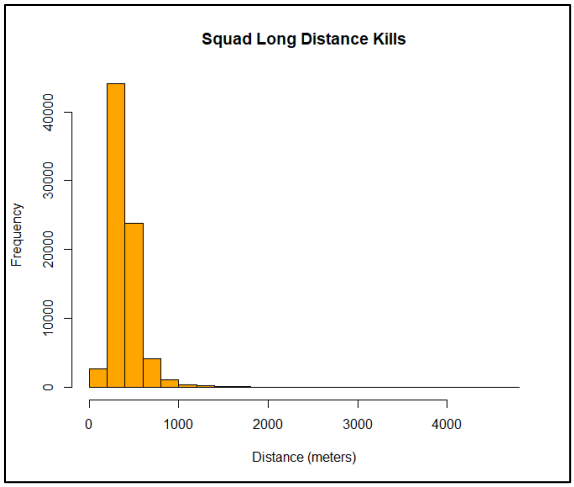
From the picture above, we can see that the winning ratio is more if the kill to death ratio is greater than 5 but most of the players have KD Ratio between 0-5 making the winning ratio below average. For distance attribute, the average winning ratio is between 0-20 where maximum distance moved is around 5000 as the area reduces after certain interval of time and if the player is out of the area then chances of dying increases whereas the average damage given by the particular player is in range of 200-400 and maximum damage is in range of 800-1000 making the winning ratio high.



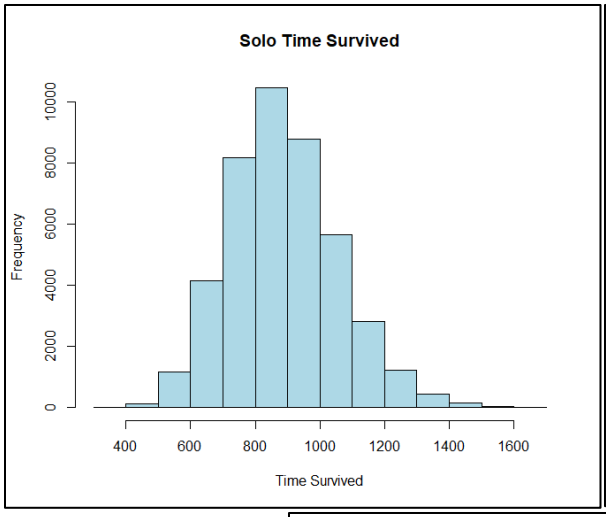
If we compare all 3 variables for all 3 modes, we see that the kill to death ratio has same impact on win ratio in all modes. The distance travelled gives almost same impact as in solo but in squad chances of winning are more if the player travels more distance as he would drop at a very far place collect the weapons and come slowly inside the circle to avoid the team fight in the starting of the game. If the player plays in squad than it is obvious that all team members will give some damage and so the damage will be less even though the win ratio is high.

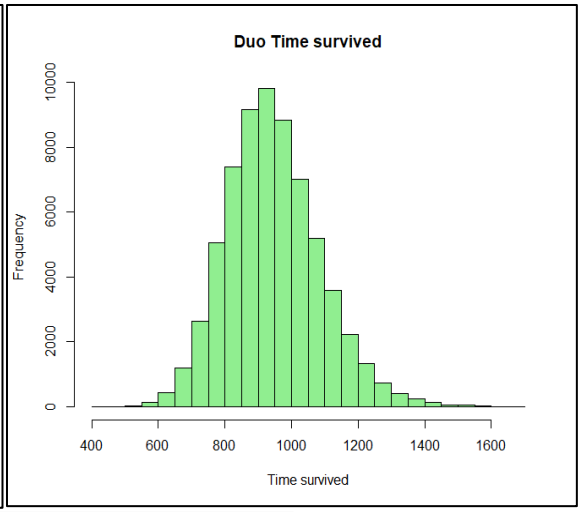


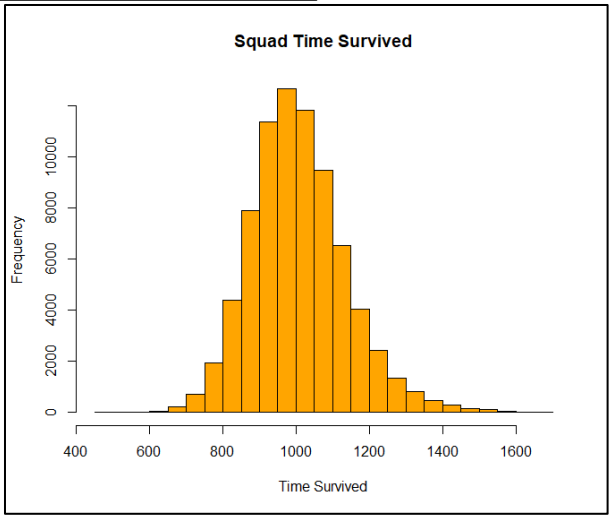




From the above graphs it can be said that most of the players can shoot around 1000 meters. Though for squad mode the longest distance of killing goes to 2000 but the frequency for that is very low. The mean longest distance at which the player killed for all 3 modes is around 300 meters.





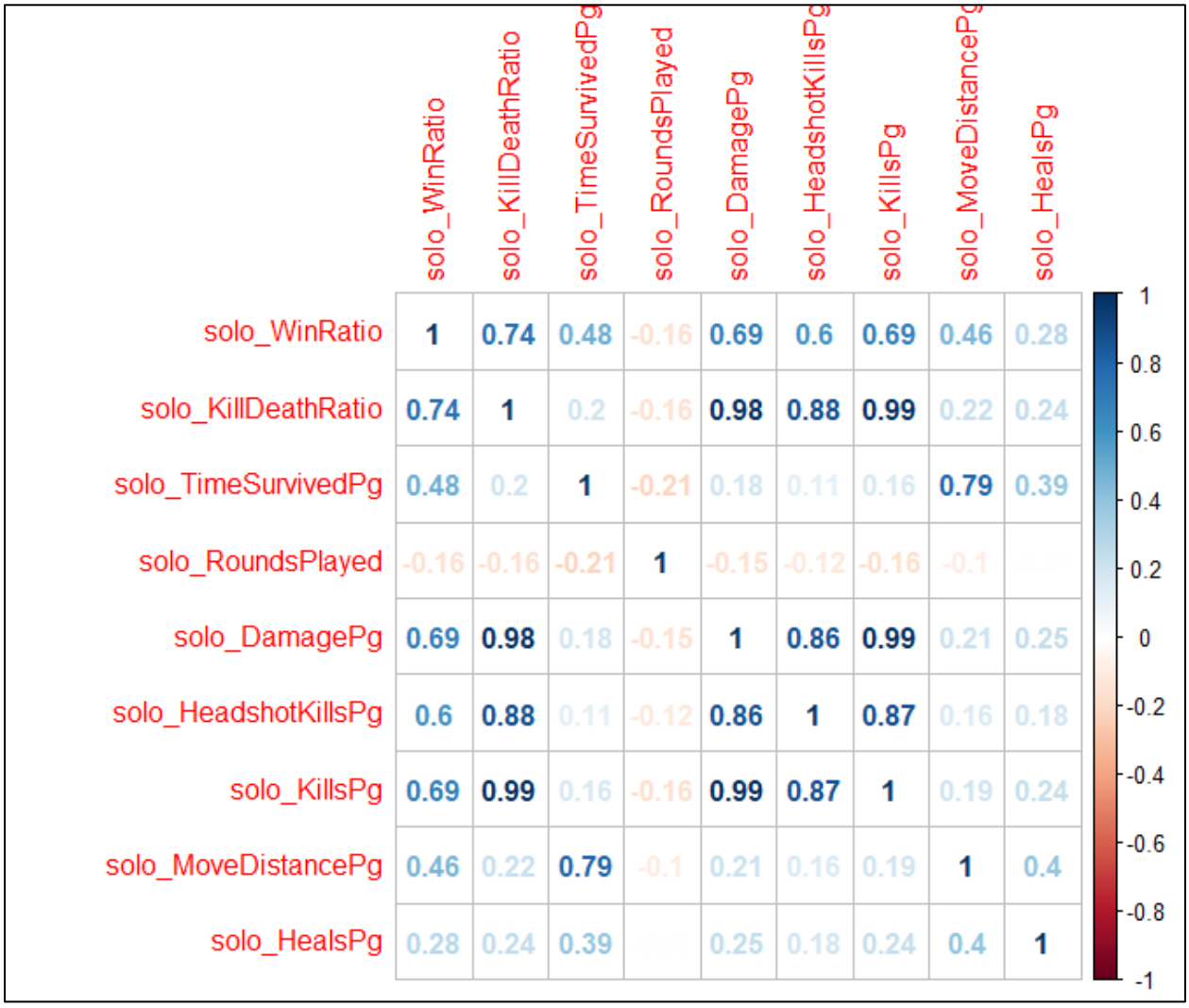


The survival time per game in solo and duo mode is between 800-1100 whereas in squad mode it is between 800-1200. Survival time is more in squad and duo because if the player kills another player, that player doesn't die instead he gets knocked down and can be revived by their teammates. So, if player wants to improve his survival time then he can play in either of squad or duo mode. Mean survival time for solo, duo and squad mode are around 800,900,1000 respectively.

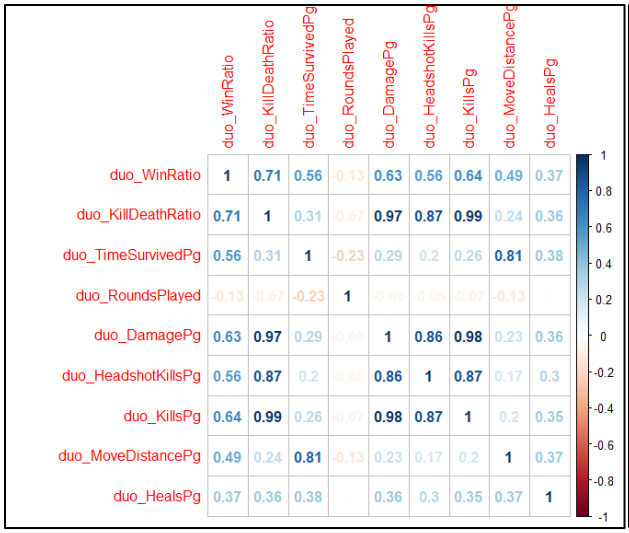
Now we are reducing the data by just including the important features which are necessary for the statistics in all three modes. Hence after reducing, the dimension is:

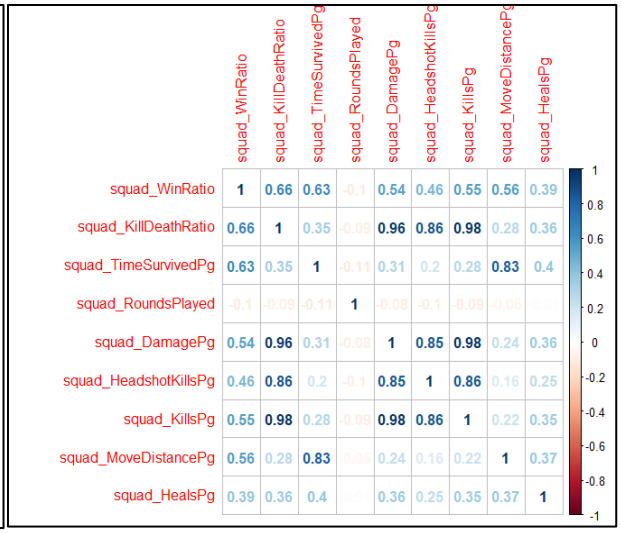
```
> dim(solo_data2)
[1] 43199     9
> dim(duo_data2)
[1] 65701     9
> dim(squad_data2)
[1] 76823     9
>
```

Let's see how different important variables are related to each other using the co-relation matrix.



It can be said by looking at the relation graph that every variable is related to itself with maximum value of 1. Accordingly, if we see the value ranges from 0 to 1 for every comparison and as large the value more closely related are the two variables. As kill death ratio is dependent on kills per game variable and if the kills per game are more the kill to death ratio is more. Similarly, we can compare each variable with other variable and know how closely they are related. Same way co-relation can be measured for squad and duo modes.





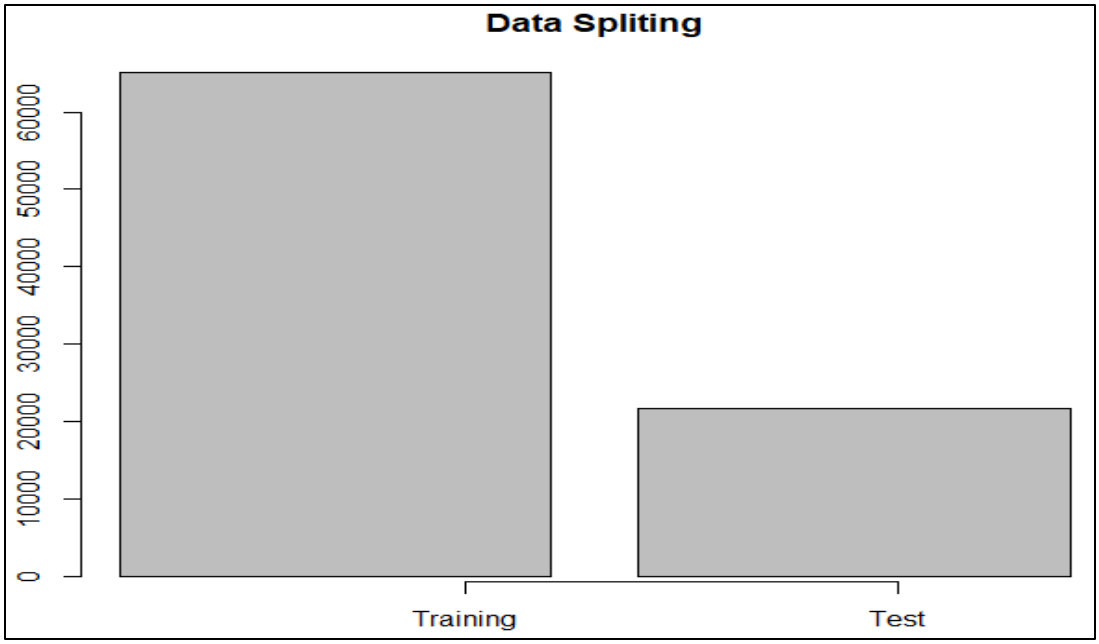
3. Dimension Reduction

Dimension Reduction is the process in statistics, machine learning and information theory in which the random variables are reduced by attaining principal variables. these are then divided into feature selection and feature extraction.

In dimension reduction we are performing below mentioned analysis:

- We have used PCA (Principal Component Analysis) and ICA (Independent Component Analysis) for dimension reduction.
- The first 3 components of PCA for the 3 dataset (raw, train, test) gives us 76% approx. of cumulative proportion and the rest of the components doesn't have a large difference.
- We are performing ICA on the 3 components and comparing the result with PCA.
- We have used the three components achieved by performing PCA and ICA for unsupervised learning on raw dataset (complete dataset), training dataset and test dataset.

Data Split: We are using the full dataset and have divided the dataset into training and test set on basis of tracker id values and after that we added one more column to see that a player prefers playing which of the 3 mode.



3.1. Principal Component Analysis (PCA)

Before performing the PCA we have standardized the raw dataset, training dataset and test dataset and then performed PCA on all of them. For analysis we are considering 3 PCA components from all the 3 dataset (raw, training, test).

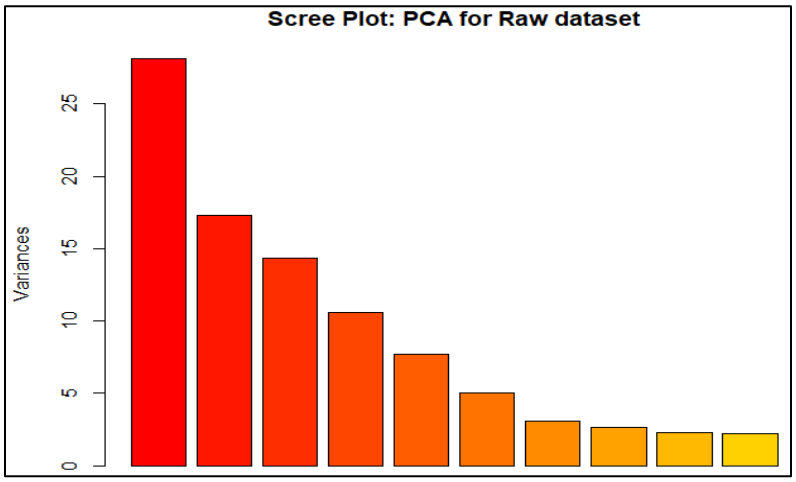
3.1.1. PCA on Standardized Raw Dataset

We have performed PCA on standardized raw dataset using prcomp() function and for analysis we have considered 3 components. We have also plotted Scree plot for standardized raw dataset.

```
> summary(pc.pubg_statistics)
Importance of components:
                          PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9   PC10   PC11
Standard deviation     5.3013 4.1609 3.7864 3.24957 2.77084 2.24048 1.7592 1.63552 1.52279 1.47786 1.3936
Proportion of Variance 0.1925 0.1186 0.0982 0.07233 0.05259 0.03438 0.0212 0.01832 0.01588 0.01496 0.0133
Cumulative Proportion  0.1925 0.3111 0.4093 0.48160 0.53418 0.56856 0.5898 0.60808 0.62397 0.63892 0.6522
                          PC12    PC13    PC14    PC15    PC16    PC17    PC18    PC19    PC20    PC21
Standard deviation     1.34612 1.31037 1.28605 1.25188 1.24732 1.21896 1.13952 1.11722 1.10835 1.06308
Proportion of Variance 0.01241 0.01176 0.01133 0.01073 0.01066 0.01018 0.00889 0.00855 0.00841 0.00774
Cumulative Proportion  0.66464 0.67640 0.68773 0.69846 0.70912 0.71929 0.72819 0.73674 0.74515 0.75289
                          PC22    PC23    PC24    PC25    PC26    PC27    PC28    PC29    PC30    PC31
Standard deviation     1.05868 1.04098 1.02747 1.02348 1.00829 0.99761 0.98485 0.97130 0.95362 0.94306
Proportion of Variance 0.00768 0.00742 0.00723 0.00717 0.00696 0.00682 0.00664 0.00646 0.00623 0.00609
Cumulative Proportion  0.76057 0.76799 0.77522 0.78240 0.78936 0.79618 0.80282 0.80928 0.81551 0.82160
                          PC32    PC33    PC34    PC35    PC36    PC37    PC38    PC39    PC40    PC41    PC42
Standard deviation     0.94016 0.9284 0.91453 0.90025 0.88110 0.87417 0.85995 0.85013 0.84424 0.83802 0.83483
Proportion of Variance 0.00605 0.0059 0.00573 0.00555 0.00532 0.00523 0.00507 0.00495 0.00488 0.00481 0.00477
Cumulative Proportion  0.82766 0.8336 0.83929 0.84484 0.85016 0.85539 0.86045 0.86540 0.87029 0.87510 0.87987
                          PC43    PC44    PC45    PC46   PC47    PC48   PC49   PC50    PC51   PC52    PC53
Standard deviation     0.82189 0.79971 0.79155 0.78219 0.7641 0.76081 0.7452 0.7251 0.71310 0.7048 0.69602
Proportion of Variance 0.00463 0.00438 0.00429 0.00419 0.0040 0.00396 0.0038 0.0036 0.00348 0.0034 0.00332
Cumulative Proportion  0.88450 0.88888 0.89317 0.89736 0.9014 0.90532 0.9091 0.9127 0.91621 0.9196 0.92293
                          PC54    PC55    PC56    PC57    PC58    PC59    PC60    PC61    PC62    PC63
Standard deviation     0.67624 0.66421 0.64366 0.62480 0.62156 0.60658 0.60341 0.59318 0.59001 0.58405
Proportion of Variance 0.00313 0.00302 0.00284 0.00267 0.00265 0.00252 0.00249 0.00241 0.00238 0.00234
Cumulative Proportion  0.92606 0.92909 0.93192 0.93460 0.93724 0.93976 0.94226 0.94467 0.94705 0.94939
                          PC64    PC65    PC66    PC67    PC68    PC69    PC70    PC71    PC72    PC73    PC74
Standard deviation     0.57532 0.56753 0.55059 0.53417 0.52309 0.51895 0.51366 0.50993 0.4982 0.49713 0.47521
Proportion of Variance 0.00227 0.00221 0.00208 0.00195 0.00187 0.00184 0.00181 0.00178 0.0017 0.00169 0.00155
Cumulative Proportion  0.95165 0.95386 0.95594 0.95789 0.95977 0.96161 0.96342 0.96520 0.9669 0.96859 0.97014
                          PC75    PC76    PC77    PC78    PC79    PC80    PC81    PC82    PC83    PC84    PC85
Standard deviation     0.4676 0.45885 0.44944 0.44207 0.42892 0.42412 0.40661 0.39866 0.39756 0.39151 0.38602
Proportion of Variance 0.0015 0.00144 0.00138 0.00134 0.00126 0.00123 0.00113 0.00109 0.00108 0.00105 0.00102
Cumulative Proportion  0.9716 0.97308 0.97446 0.97580 0.97706 0.97829 0.97942 0.98051 0.98160 0.98265 0.98367
```

```
                          PC86    PC87    PC88    PC89    PC90    PC91    PC92    PC93    PC94    PC95    PC96
Standard deviation     0.37180 0.36119 0.35688 0.35152 0.34375 0.33771 0.32681 0.31214 0.30815 0.2963 0.29374
Proportion of Variance 0.00095 0.00089 0.00087 0.00085 0.00081 0.00078 0.00073 0.00067 0.00065 0.0006 0.00059
Cumulative Proportion  0.98461 0.98551 0.98638 0.98723 0.98803 0.98882 0.98955 0.99021 0.99086 0.9915 0.99206
                          PC97    PC98    PC99   PC100   PC101   PC102  PC103   PC104   PC105   PC106   PC107
Standard deviation     0.28025 0.27564 0.2706 0.26616 0.26023 0.24402 0.2416 0.22792 0.22303 0.21859 0.21578
Proportion of Variance 0.00054 0.00052 0.0005 0.00049 0.00046 0.00041 0.0004 0.00036 0.00034 0.00033 0.00032
Cumulative Proportion  0.99259 0.99312 0.9936 0.99410 0.99457 0.99497 0.9954 0.99573 0.99607 0.99640 0.99672
                         PC108   PC109   PC110   PC111   PC112   PC113   PC114   PC115   PC116   PC117   PC118
Standard deviation     0.2096 0.20326 0.19636 0.19082 0.18369 0.17867 0.17692 0.16159 0.15795 0.15180 0.14705
Proportion of Variance 0.0003 0.00028 0.00026 0.00025 0.00023 0.00022 0.00021 0.00018 0.00017 0.00016 0.00015
Cumulative Proportion  0.9970 0.99730 0.99756 0.99781 0.99804 0.99826 0.99848 0.99866 0.99883 0.99899 0.99913
                          PC119   PC120   PC121   PC122   PC123   PC124   PC125   PC126   PC127   PC128
Standard deviation     0.13892 0.13453 0.12972 0.12604 0.11727 0.10365 0.08636 0.07444 0.06704 0.05926
Proportion of Variance 0.00013 0.00012 0.00012 0.00011 0.00009 0.00007 0.00005 0.00004 0.00003 0.00002
Cumulative Proportion  0.99927 0.99939 0.99950 0.99961 0.99971 0.99978 0.99983 0.99987 0.99990 0.99993
                          PC129   PC130   PC131   PC132   PC133   PC134   PC135   PC136    PC137     PC138
Standard deviation     0.05493 0.05054 0.04378 0.03988 0.03219 0.01859 0.01771 0.01186 0.003833 1.944e-07
Proportion of Variance 0.00002 0.00002 0.00001 0.00001 0.00001 0.00000 0.00000 0.00000 0.000000 0.000e+00
Cumulative Proportion  0.99995 0.99996 0.99998 0.99999 0.99999 1.00000 1.00000 1.00000 1.000000 1.000e+00
                          PC139     PC140     PC141     PC142     PC143     PC144     PC145     PC146
Standard deviation     1.75e-07 1.504e-07 1.575e-14 1.375e-14 4.514e-15 3.669e-15 7.385e-16 4.823e-16
Proportion of Variance 0.00e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
Cumulative Proportion  1.00e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
                          PC147     PC148     PC149     PC150     PC151     PC152
Standard deviation     4.823e-16 4.823e-16 4.823e-16 4.823e-16 4.823e-16 1.925e-16
Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
```

```
> plot(pc.pubg_statistics, main="Scree Plot: PCA for Raw dataset", col=heat.colors(15))
> # First  Principal Components
> pc1.pubg_statistics <- data.frame(pc.pubg_statistics$x[,1:3])
> head(pc1.pubg_statistics)
        PC1       PC2        PC3
1 -22.30945 -24.39832  -8.187687
2 -26.36356 -24.60434  -9.447477
3 -24.57998 -26.35365 -12.267364
4 -14.75258 -24.09506   1.090240
5 -19.40421 -21.07874   2.312225
6 -21.30970 -19.42916 -10.200063
```



3.1.2. PCA on Training Dataset

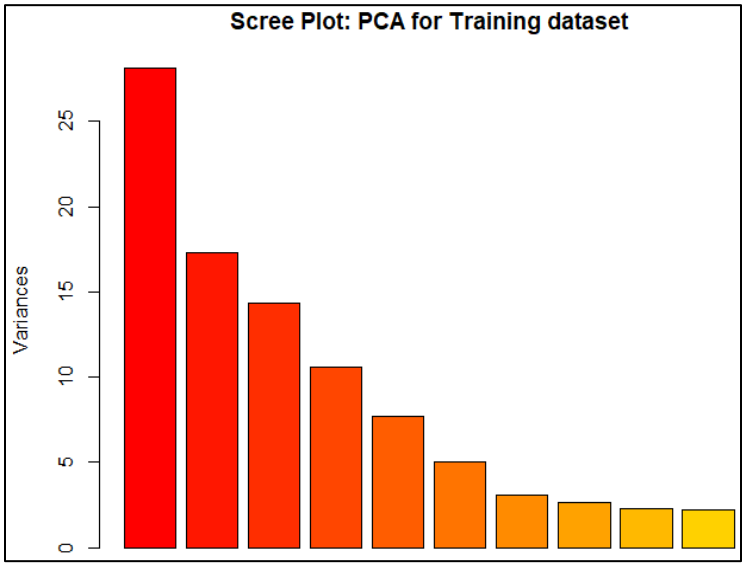
We have performed PCA on training dataset using prcomp() function and for analysis we have considered 3 components. We have also plotted Scree plot for training dataset.

```
> pc.pubg_statistics_train <- prcomp(pubg_statistics_train.std)
> summary(pc.pubg_statistics_train)
Importance of components:
                          PC1    PC2    PC3     PC4    PC5    PC6    PC7     PC8     PC9    PC10    PC11
Standard deviation     5.2547 4.1814 3.8114 3.27732 2.7896 2.2248 1.7635 1.63744 1.52536 1.48778 1.39188
Proportion of Variance 0.1891 0.1197 0.0995 0.07357 0.0533 0.0339 0.0213 0.01836 0.01594 0.01516 0.01327
Cumulative Proportion  0.1891 0.3089 0.4084 0.48194 0.5352 0.5691 0.5905 0.60881 0.62475 0.63991 0.65318
                          PC12    PC13    PC14    PC15    PC16    PC17    PC18   PC19    PC20    PC21    PC22
Standard deviation     1.34653 1.30541 1.28691 1.25843 1.25058 1.22436 1.14239 1.1143 1.10186 1.06381 1.05854
Proportion of Variance 0.01242 0.01167 0.01134 0.01085 0.01071 0.01027 0.00894 0.0085 0.00832 0.00775 0.00767
Cumulative Proportion  0.66560 0.67727 0.68861 0.69946 0.71017 0.72044 0.72938 0.7379 0.74620 0.75395 0.76162
                          PC23    PC24    PC25    PC26    PC27    PC28    PC29    PC30    PC31    PC32
Standard deviation     1.03753 1.02162 1.02057 1.00840 0.99382 0.98325 0.96796 0.95042 0.94695 0.93831
Proportion of Variance 0.00737 0.00715 0.00713 0.00696 0.00676 0.00662 0.00642 0.00619 0.00614 0.00603
Cumulative Proportion  0.76900 0.77614 0.78328 0.79024 0.79701 0.80363 0.81005 0.81623 0.82238 0.82841
                          PC33    PC34    PC35    PC36    PC37    PC38    PC39    PC40   PC41    PC42    PC43
Standard deviation     0.93459 0.91656 0.89804 0.87756 0.87335 0.85830 0.85577 0.84882 0.8369 0.83552 0.82367
Proportion of Variance 0.00598 0.00575 0.00552 0.00527 0.00522 0.00505 0.00502 0.00493 0.0048 0.00478 0.00465
Cumulative Proportion  0.83439 0.84014 0.84567 0.85094 0.85617 0.86121 0.86623 0.87116 0.8760 0.88074 0.88539
                          PC44    PC45    PC46    PC47    PC48    PC49    PC50    PC51    PC52    PC53
Standard deviation     0.79849 0.79014 0.78142 0.76361 0.75760 0.74594 0.72911 0.71373 0.70197 0.69603
Proportion of Variance 0.00437 0.00428 0.00418 0.00399 0.00393 0.00381 0.00364 0.00349 0.00338 0.00332
Cumulative Proportion  0.88975 0.89403 0.89821 0.90221 0.90614 0.90995 0.91359 0.91708 0.92045 0.92377
                          PC54    PC55    PC56    PC57    PC58    PC59    PC60   PC61    PC62    PC63    PC64
Standard deviation     0.67614 0.65996 0.64334 0.62384 0.61981 0.60541 0.60139 0.5915 0.58362 0.58284 0.57225
Proportion of Variance 0.00313 0.00298 0.00283 0.00267 0.00263 0.00251 0.00248 0.0024 0.00233 0.00233 0.00224
Cumulative Proportion  0.92690 0.92989 0.93272 0.93539 0.93802 0.94053 0.94301 0.9454 0.94774 0.95006 0.95231
                          PC65    PC66    PC67    PC68    PC69   PC70    PC71    PC72    PC73    PC74    PC75
Standard deviation     0.56451 0.54662 0.52874 0.52253 0.51803 0.5126 0.50909 0.49646 0.49546 0.47583 0.47011
Proportion of Variance 0.00218 0.00205 0.00191 0.00187 0.00184 0.0018 0.00178 0.00169 0.00168 0.00155 0.00151
Cumulative Proportion  0.95449 0.95653 0.95845 0.96032 0.96216 0.9640 0.96573 0.96742 0.96910 0.97065 0.97217

                          PC65    PC66    PC67    PC68    PC69   PC70    PC71    PC72    PC73    PC74    PC75
Standard deviation     0.56451 0.54662 0.52874 0.52253 0.51803 0.5126 0.50909 0.49646 0.49546 0.47583 0.47011
Proportion of Variance 0.00218 0.00205 0.00191 0.00187 0.00184 0.0018 0.00178 0.00169 0.00168 0.00155 0.00151
Cumulative Proportion  0.95449 0.95653 0.95845 0.96032 0.96216 0.9640 0.96573 0.96742 0.96910 0.97065 0.97217
                          PC76    PC77    PC78    PC79    PC80    PC81    PC82    PC83    PC84    PC85
Standard deviation     0.45834 0.44868 0.44141 0.42389 0.42343 0.40373 0.39621 0.39471 0.38916 0.38502
Proportion of Variance 0.00144 0.00138 0.00133 0.00123 0.00123 0.00112 0.00108 0.00107 0.00104 0.00102
Cumulative Proportion  0.97361 0.97498 0.97632 0.97755 0.97878 0.97989 0.98097 0.98204 0.98307 0.98409
                          PC86    PC87    PC88    PC89    PC90    PC91    PC92    PC93   PC94    PC95    PC96
Standard deviation     0.36889 0.35789 0.35188 0.34761 0.33644 0.32753 0.32463 0.30949 0.2956 0.29360 0.29243
Proportion of Variance 0.00093 0.00088 0.00085 0.00083 0.00078 0.00073 0.00072 0.00066 0.0006 0.00059 0.00059
Cumulative Proportion  0.98502 0.98590 0.98675 0.98757 0.98835 0.98908 0.98981 0.99046 0.9911 0.99165 0.99224
                          PC97    PC98    PC99   PC100   PC101  PC102   PC103   PC104   PC105   PC106   PC107
Standard deviation     0.27985 0.27226 0.26560 0.26382 0.25358 0.2409 0.23182 0.22566 0.22381 0.21654 0.21407
Proportion of Variance 0.00054 0.00051 0.00048 0.00048 0.00044 0.0004 0.00037 0.00035 0.00034 0.00032 0.00031
Cumulative Proportion  0.99277 0.99328 0.99376 0.99424 0.99468 0.9951 0.99545 0.99580 0.99614 0.99646 0.99677
                         PC108   PC109   PC110   PC111   PC112   PC113   PC114   PC115   PC116   PC117
Standard deviation     0.20710 0.20044 0.19527 0.18979 0.18334 0.18235 0.17690 0.15818 0.15412 0.15026
Proportion of Variance 0.00029 0.00028 0.00026 0.00025 0.00023 0.00023 0.00021 0.00017 0.00016 0.00015
Cumulative Proportion  0.99707 0.99734 0.99760 0.99785 0.99808 0.99831 0.99852 0.99869 0.99886 0.99901
                         PC118   PC119   PC120   PC121  PC122   PC123   PC124   PC125   PC126   PC127   PC128
Standard deviation     0.14450 0.13793 0.13421 0.12821 0.1208 0.11546 0.10322 0.08570 0.07424 0.06640 0.05854
Proportion of Variance 0.00014 0.00013 0.00012 0.00011 0.0001 0.00009 0.00007 0.00005 0.00004 0.00003 0.00002
Cumulative Proportion  0.99915 0.99928 0.99941 0.99952 0.9996 0.99971 0.99979 0.99984 0.99987 0.99990 0.99993
                         PC129   PC130   PC131   PC132   PC133   PC134   PC135   PC136    PC137     PC138
Standard deviation     0.05450 0.04994 0.04344 0.03935 0.03149 0.01839 0.01747 0.01189 0.003548 1.968e-07
Proportion of Variance 0.00002 0.00002 0.00001 0.00001 0.00001 0.00000 0.00000 0.00000 0.000000 0.000e+00
Cumulative Proportion  0.99995 0.99996 0.99998 0.99999 0.99999 1.00000 1.00000 1.00000 1.000000 1.000e+00
                          PC139     PC140     PC141     PC142   PC143     PC144     PC145     PC146
Standard deviation     1.751e-07 1.513e-07 1.204e-14 9.753e-15 9.7e-15 8.966e-15 4.007e-15 2.999e-15
Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.0e+00 0.000e+00 0.000e+00 0.000e+00
Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.0e+00 1.000e+00 1.000e+00 1.000e+00

                          PC147     PC148     PC149     PC150     PC151     PC152
Standard deviation     4.711e-16 4.711e-16 4.711e-16 4.711e-16 4.711e-16 3.775e-16
Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
```

```
> plot(pc.pubg_statistics, main="Scree Plot: PCA for Training dataset", col=heat.colors(15))
> # First  Principal Components
> pc1.pubg_statistics_train <- data.frame(pc.pubg_statistics_train$x[,1:3])
> head(pc1.pubg_statistics_train)
        PC1      PC2         PC3
1 -20.93930 23.99096  -7.473021
2 -24.80259 24.23914  -8.751732
3 -23.00496 25.94859 -11.423493
4 -13.96371 23.55767   1.721773
5 -18.50346 20.30399   2.767346
6 -19.88563 19.10147  -9.519350
```



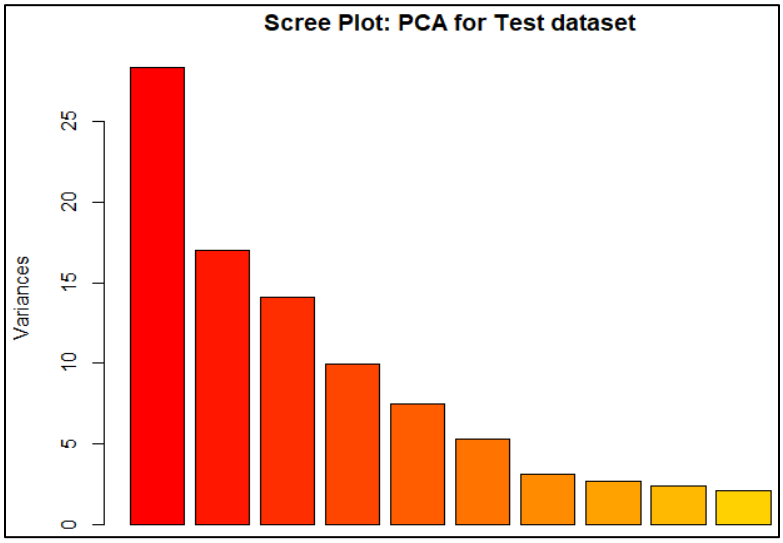
3.1.3. PCA on Test Dataset

We have performed PCA on test dataset using prcomp() function and for analysis we have considered 3 components. We have also plotted Scree plot for test dataset.

```
> pc.pubg_statistics_test <- prcomp(pubg_statistics_test.std[,])
> summary(pc.pubg_statistics_test)
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10    PC11
Standard deviation     5.3251 4.1287 3.75674 3.15513 2.73771 2.30106 1.76601 1.64517 1.53172 1.43363 1.40156
Proportion of Variance 0.1942 0.1168 0.09667 0.06818 0.05134 0.03627 0.02136 0.01854 0.01607 0.01408 0.01345
Cumulative Proportion  0.1942 0.3110 0.40764 0.47583 0.52716 0.56343 0.58479 0.60333 0.61940 0.63348 0.64693
                          PC12    PC13    PC14    PC15    PC16    PC17    PC18    PC19    PC20    PC21
Standard deviation     1.35518 1.33948 1.29610 1.25537 1.22100 1.19413 1.15298 1.14425 1.12755 1.07938
Proportion of Variance 0.01258 0.01229 0.01151 0.01079 0.01021 0.00977 0.00911 0.00897 0.00871 0.00798
Cumulative Proportion  0.65951 0.67180 0.68331 0.69410 0.70431 0.71408 0.72318 0.73215 0.74086 0.74884
                          PC22    PC23   PC24    PC25    PC26    PC27    PC28    PC29    PC30    PC31    PC32
Standard deviation     1.06412 1.06271 1.0464 1.04197 1.01494 1.00611 0.99744 0.99284 0.97624 0.96726 0.94286
Proportion of Variance 0.00776 0.00774 0.0075 0.00744 0.00706 0.00693 0.00681 0.00675 0.00653 0.00641 0.00609
Cumulative Proportion  0.75660 0.76433 0.7718 0.77927 0.78632 0.79326 0.80007 0.80682 0.81335 0.81976 0.82585
                          PC33    PC34    PC35    PC36    PC37    PC38    PC39    PC40    PC41    PC42    PC43
Standard deviation     0.93206 0.9200 0.90048 0.89853 0.87623 0.86049 0.85333 0.83299 0.82759 0.82239 0.81658
Proportion of Variance 0.00595 0.0058 0.00555 0.00553 0.00526 0.00507 0.00499 0.00475 0.00469 0.00463 0.00457
Cumulative Proportion  0.83180 0.8376 0.84315 0.84868 0.85394 0.85901 0.86400 0.86875 0.87344 0.87807 0.88264
                          PC44    PC45    PC46    PC47    PC48    PC49    PC50   PC51    PC52    PC53    PC54
Standard deviation     0.8014 0.79843 0.78707 0.78006 0.75831 0.73207 0.72631 0.7045 0.68922 0.68075 0.67806
Proportion of Variance 0.0044 0.00437 0.00424 0.00417 0.00394 0.00367 0.00361 0.0034 0.00325 0.00317 0.00315
Cumulative Proportion  0.8870 0.89140 0.89565 0.89981 0.90375 0.90742 0.91104 0.9144 0.91769 0.92086 0.92401
                          PC55    PC56    PC57    PC58    PC59    PC60    PC61    PC62    PC63    PC64    PC65
Standard deviation     0.67062 0.6510 0.63446 0.62241 0.62038 0.61044 0.60584 0.59638 0.58915 0.58052 0.57137
Proportion of Variance 0.00308 0.0029 0.00276 0.00265 0.00264 0.00255 0.00251 0.00244 0.00238 0.00231 0.00224
Cumulative Proportion  0.92709 0.9300 0.93275 0.93541 0.93804 0.94060 0.94311 0.94555 0.94792 0.95023 0.95247
                          PC66    PC67    PC68    PC69    PC70    PC71    PC72   PC73    PC74   PC75    PC76
Standard deviation     0.56516 0.5534 0.52835 0.52419 0.52195 0.51516 0.50532 0.4978 0.48456 0.4685 0.45967
Proportion of Variance 0.00219 0.0021 0.00191 0.00188 0.00187 0.00182 0.00175 0.0017 0.00161 0.0015 0.00145
Cumulative Proportion  0.95465 0.9567 0.95866 0.96055 0.96241 0.96423 0.96598 0.9677 0.96928 0.9708 0.97224
                          PC77    PC78    PC79    PC80    PC81    PC82    PC83    PC84    PC85   PC86    PC87
Standard deviation     0.44743 0.44434 0.43452 0.42827 0.41436 0.41181 0.40397 0.39765 0.39084 0.3815 0.37305
Proportion of Variance 0.00137 0.00135 0.00129 0.00126 0.00118 0.00116 0.00112 0.00108 0.00105 0.0010 0.00095
Cumulative Proportion  0.97361 0.97496 0.97625 0.97751 0.97868 0.97985 0.98096 0.98205 0.98309 0.9841 0.98504

                          PC77    PC78    PC79    PC80    PC81    PC82    PC83    PC84    PC85   PC86    PC87
Standard deviation     0.44743 0.44434 0.43452 0.42827 0.41436 0.41181 0.40397 0.39765 0.39084 0.3815 0.37305
Proportion of Variance 0.00137 0.00135 0.00129 0.00126 0.00118 0.00116 0.00112 0.00108 0.00105 0.0010 0.00095
Cumulative Proportion  0.97361 0.97496 0.97625 0.97751 0.97868 0.97985 0.98096 0.98205 0.98309 0.9841 0.98504
                          PC88    PC89    PC90    PC91    PC92    PC93    PC94    PC95    PC96    PC97    PC98
Standard deviation     0.3628 0.35387 0.35036 0.34719 0.33209 0.32694 0.32100 0.30885 0.29850 0.28794 0.28277
Proportion of Variance 0.0009 0.00086 0.00084 0.00083 0.00076 0.00073 0.00071 0.00065 0.00061 0.00057 0.00055
Cumulative Proportion  0.9859 0.98680 0.98764 0.98847 0.98922 0.98996 0.99066 0.99132 0.99193 0.99249 0.99304
                          PC99   PC100   PC101   PC102   PC103   PC104   PC105   PC106   PC107   PC108   PC109
Standard deviation     0.27583 0.27324 0.26060 0.24743 0.23095 0.22809 0.22319 0.21802 0.21431 0.2082 0.20324
Proportion of Variance 0.00052 0.00051 0.00047 0.00042 0.00037 0.00036 0.00034 0.00033 0.00031 0.0003 0.00028
Cumulative Proportion  0.99356 0.99407 0.99454 0.99496 0.99532 0.99568 0.99602 0.99635 0.99666 0.9970 0.99724
                         PC110   PC111   PC112   PC113  PC114   PC115   PC116   PC117   PC118   PC119   PC120
Standard deviation     0.19829 0.19303 0.17910 0.17691 0.1708 0.16290 0.15606 0.15162 0.14957 0.14459 0.13883
Proportion of Variance 0.00027 0.00026 0.00022 0.00021 0.0002 0.00018 0.00017 0.00016 0.00015 0.00014 0.00013
Cumulative Proportion  0.99751 0.99777 0.99798 0.99820 0.9984 0.99858 0.99875 0.99891 0.99906 0.99920 0.99933
                         PC121   PC122  PC123   PC124   PC125   PC126   PC127   PC128   PC129   PC130   PC131
Standard deviation     0.13728 0.13251 0.1224 0.10590 0.08908 0.07383 0.07079 0.06329 0.05724 0.05410 0.04599
Proportion of Variance 0.00013 0.00012 0.0001 0.00008 0.00005 0.00004 0.00003 0.00003 0.00002 0.00002 0.00001
Cumulative Proportion  0.99946 0.99958 0.9997 0.99976 0.99982 0.99985 0.99989 0.99992 0.99994 0.99996 0.99997
                         PC132   PC133   PC134   PC135   PC136   PC137     PC138     PC139     PC140
Standard deviation     0.04264 0.03550 0.01953 0.01874 0.01179 0.004628 1.907e-07 1.766e-07 1.457e-07
Proportion of Variance 0.00001 0.00001 0.00000 0.00000 0.00000 0.000000 0.000e+00 0.000e+00 0.000e+00
Cumulative Proportion  0.99999 0.99999 1.00000 1.00000 1.00000 1.000000 1.000e+00 1.000e+00 1.000e+00
                           PC141     PC142     PC143     PC144     PC145     PC146     PC147     PC148
Standard deviation     3.896e-15 2.52e-15 2.437e-15 1.895e-15 5.798e-16 4.749e-16 4.749e-16 4.749e-16
Proportion of Variance 0.000e+00 0.00e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
Cumulative Proportion  1.000e+00 1.00e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
                           PC149     PC150     PC151     PC152
Standard deviation     4.749e-16 4.749e-16 4.749e-16 2.862e-16
Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00
Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00
```

```
> plot(pc.pubg_statistics_test, main="Scree Plot: PCA for Test dataset", col=heat.colors(15))
> # First  Principal Components
> pc1.pubg_statistics_test <- data.frame(pc.pubg_statistics_test$x[,1:3])
> head(pc1.pubg_statistics_test)
         PC1       PC2        PC3
22 -25.97298  10.66177   5.388195
29 -24.31856  13.17552  17.009275
37 -15.22993  12.82973  -4.200789
38 -24.08380   6.72252   7.132316
61 -17.09884 -21.41766 -24.944482
71 -33.03031  13.32285   9.739827
```



3.2. Independent Component Analysis (ICA)

We have Whitened all the three dataset (raw, test, train) and then performed ICA on it and for analysis purpose we have considered 3 components.

3.2.1. ICA on Raw Dataset

We have performed ICA on raw dataset and for analysis, we have considered following 3 components:

```
> pubg_statistics.ica<-pubg_statistics.white.ica$S
> head(pubg_statistics.ica)
           [,1]       [,2]       [,3]
1  0.2255255398 -0.9877644  0.3495262
2  0.2427503793 -1.4721373  5.4011034
3 -0.0001727991 -1.1252607  2.2726213
4  0.3684572074  0.3772458  1.1656817
5 -0.0953705290 -0.2736759 -0.3653065
6  0.0731648550 -0.9549122 -0.5925547
```

3.2.2. ICA on Training Dataset

We have performed ICA on training dataset and for analysis, we have considered following 3 components:

```
> pubg_statistics_train.ica<-pubg_statistics_train.white.ica$s
> head(pubg_statistics_train.ica)
          [,1]        [,2]        [,3]
1 -0.19318241 -1.13490291 -0.08590744
2 -0.14943837 -1.04338866 -5.61593331
3 -0.08650047 -0.94290386 -2.23629113
4 -0.33770407  0.35911415 -0.99456325
5 -0.07876217 -0.07430485  0.28760091
6 -0.06787546 -0.59777532  0.09106128
```

3.2.3. ICA on Test Dataset

We have performed ICA on raw test and for analysis, we have considered following 3 components:

```
> pubg_statistics_test.ica<-pubg_statistics_test.white.ica$s
> head(pubg_statistics_test.ica)
          [,1]       [,2]        [,3]
22 -2.2814001  0.5665971 -0.14400884
29  0.2059639  0.2043439 -0.23324022
37  0.5574138 -0.2326302 -0.36218029
38 -1.2478622  0.8317376  0.05597061
61  0.2430273  0.1175132 -0.07112680
71  0.7230056 -0.3242281  0.21983750
```

4. Data Reduction

It is a transformation technique in which the data is transformed into a meaningful form. The data could be collected with experiments and then it is ordered, corrected and placed in the simplified form.

In data reduction we are performing below mentioned analysis:

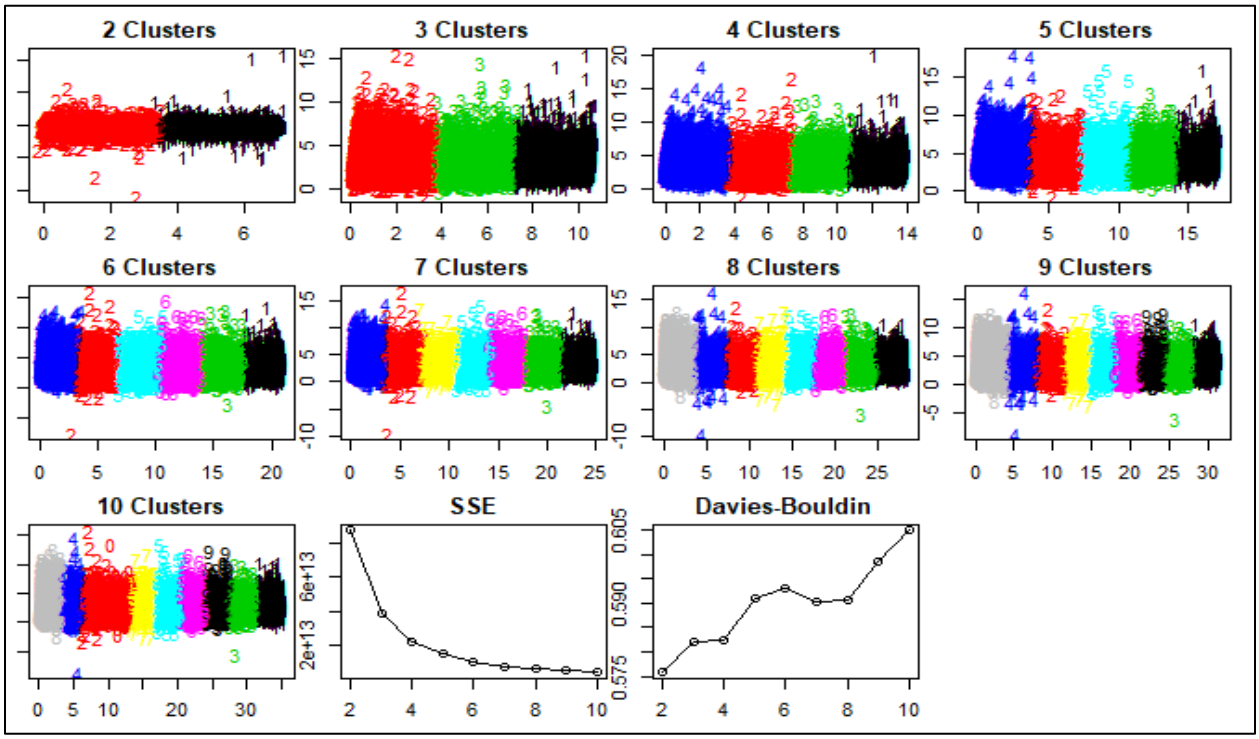
- We have performed clustering on raw dataset, standardized raw dataset and whitened raw dataset. Optimal value for raw, standardized raw and whitened are 2, 8, 10 respectively.
- Then we are creating groups on the basis of optimal value and using comparison value to check which group performed well in which mode.
- Then we are also checking the group members which perform well and poor (as in highest performance mode and lowest performance mode respectively)
- It seems that most of the members work well in squad mode and few of the members don't work well in single mode. This means that in the dataset the players work well as in team.

4.1. Clustering: Raw Dataset

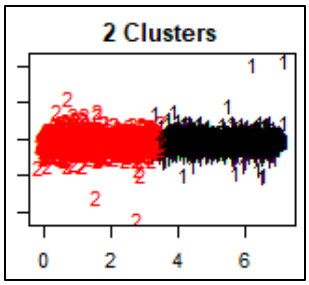
We have performed cluster analysis from range 2 to 10 on Whitened Raw Dataset. For cluster analysis, we have assigned the colors and numbers as per the clusters, as we didn't have any attributes for classification.

On the basis of **DBI value** we choose the **optimal value for Raw Dataset is 2.**

We got the below mentioned results for K=2 to 10.



Below mentioned is the **Optimal Value for Raw dataset i.e. 2**



Below mentioned are the values for the **best seed, wrong classification and centroids** for the cluster:

```
> clus.pubg_statistics<- clustering_euclidean(pubg_statistics[,c(2:4,6,14:18,22,53:54,56,64:68,72,103:104,106,114
:118,122,153)], pubg_statistics, 2)
[1] "Best Seed for the Cluster Size  2 is  2"
[1] "Total Wrong Classification in Cluster Size  2 is  0"
[1] "Centroids for the Cluster Size  2 are :"
  tracker_id solo_KillDeathRatio solo_WinRatio solo_RoundsPlayed solo_DamagePg solo_HeadshotKillsPg
1  164166.11            1.742996      4.471702          76.95231      183.9572            0.3496086
2   53768.78            2.014290      5.675086          80.97410      206.5969            0.4096241
  solo_HealsPg solo_KillsPg solo_MoveDistancePg solo_TimeSurvivedPg duo_KillDeathRatio duo_WinRatio
1     1.366455     1.587832            2725.976            959.9335           1.345147     4.062920
2     1.472207     1.796552            2947.459            994.5693           1.604585     5.295204
```

Below mentioned is the values for the **sum of squares** for the cluster:

```
within cluster sum of squares by cluster:
[1] 4.744808e+13 4.039235e+13
 (between_SS / total_SS =  75.0 %)
```

Then we started the cluster analysis, we divided the clusters into 2 groups on the basis of $comparison value to check people perform good in which mode (1: is for Solo, 2 is for Duo, 3rd is for squad).

As per the below mentioned results it seems the group 2 perform well in mode 3 i.e. in squad and group 1 perform least in mode 1 i.e. solo. We also check the members for group 2 and group 1 for the same:

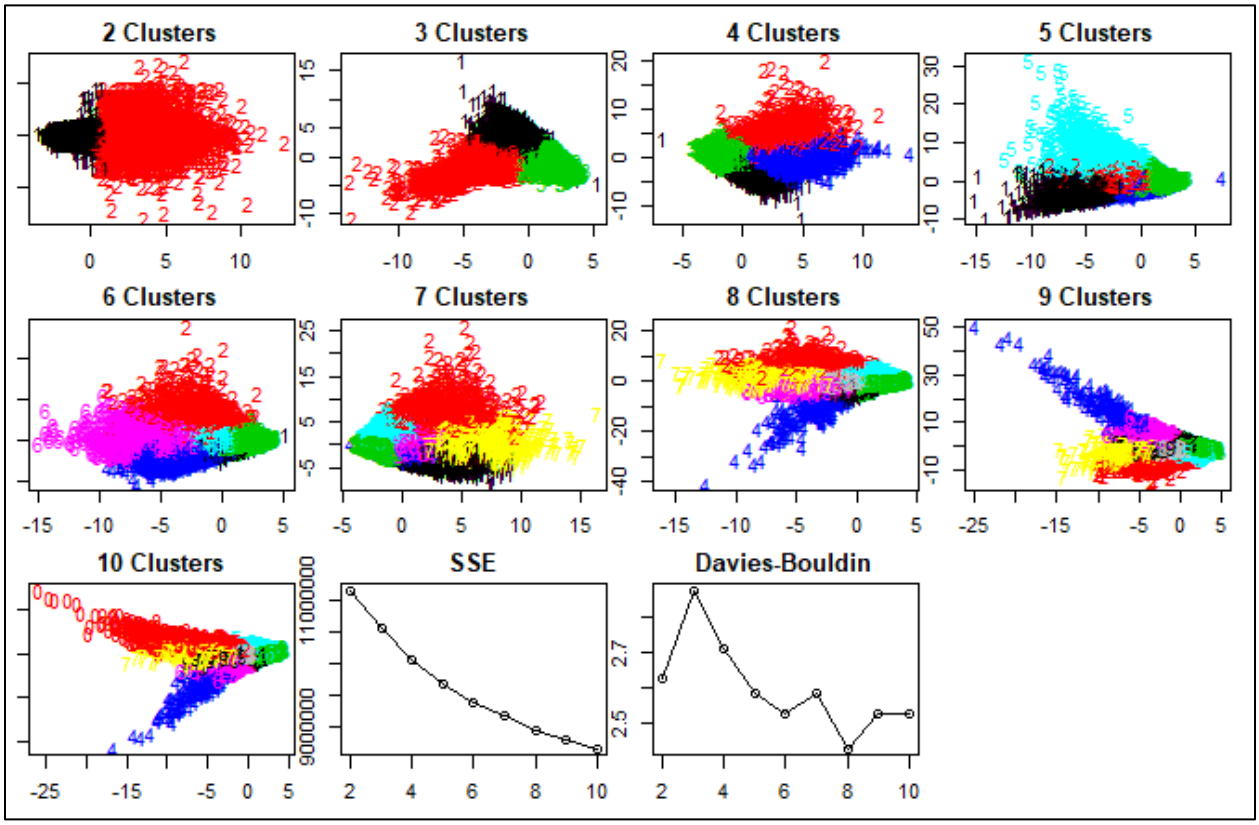
```
> print(clus.pubg_statistics$size)
[1] 44793 41975
> print(paste("Raw: Highest performance mode Group is g", highest.perf.mode.group7, sep=""))
[1] "Raw: Highest performance mode Group is g2"
> print(paste("Raw: Lowest performance mode Group is g", lowest.perf.mode.group7, sep=""))
[1] "Raw: Lowest performance mode Group is g1"
> head(highest.perf.mode.group.raw[order(highest.perf.mode.group.raw$comparison, decreasing=TRUE), ])
      player_name comparison
2       Blackwalk          3
7           Giken          3
9           undor          3
12       Benny-_-          3
50          Pull7          3
54 Nicknameilanda          3
> tail(lowest.perf.mode.group.raw[order(lowest.perf.mode.group.raw$comparison, decreasing=TRUE), ])
       player_name comparison
87872        Copycat          1
87880   RoseFlunder           1
87882        overjoy          1
87888 yaobaidaxiong           1
87892       IssacCOx          1
87897       Neferhor          1
```

4.2. Clustering: Standardized Raw Dataset

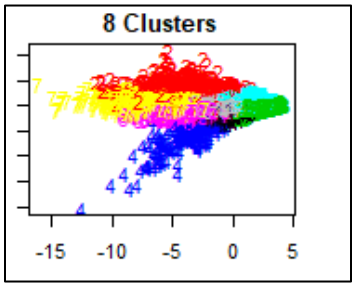
We have performed cluster analysis from range 2 to 10 on Whitened Raw Dataset. For cluster analysis, we have assigned the colors and numbers as per the clusters, as we didn't have any attributes for classification.

On the basis of **DBI value** we choose the **optimal value for Whitened Raw Dataset is 8.**

We got the below mentioned results for K=2 to 10.



Below mentioned is the **Optimal Value for Standardized Raw dataset i.e. 8**



Below mentioned are the values for the **best seed, wrong classification and centroids** for the cluster:

```
> clus.pubg_statistics.std <- clustering_euclidean(pubg_statistics.std, pubg_statistics, 8)
[1] "Best Seed for the Cluster Size  8 is  2"
[1] "Total Wrong Classification in Cluster Size  8 is  0"
[1] "Centroids for the Cluster Size  8 are :"
   tracker_id solo_KillDeathRatio solo_WinRatio solo_TimeSurvived solo_RoundsPlayed   solo_Wins
1 -0.15467141          0.9243253    0.93475344       -0.50552588       -0.54924275  0.02712742
2 -0.28695137         -0.1977693   -0.18072145        3.37907634        3.23978309  2.40105636
3  0.22881552         -0.2804097   -0.25684993       -0.54155887       -0.51828784 -0.44448643
4 -0.46172692          4.6450697    1.85585081       -0.47012074       -0.48674417  0.99283503
5  0.16284266         -0.4042903   -0.31468597        0.63327967        0.63410328 -0.01265482
6 -0.43707291          0.1468874    0.08875113       -0.12348430       -0.12810621  0.07040999
```

Below mentioned is the values for the **sum of squares** for the cluster:

```
Within cluster sum of squares by cluster:
[1] 1700731.8  735809.5 1718113.0  260388.3 1221472.3  967229.9  898300.7 1440643.1
 (between_SS / total_SS =  29.4 %)
```

Then we started the cluster analysis, we divided the clusters into 8 groups on the basis of $comparison value to check people perform good in which mode (1: is for Solo, 2 is for Duo, 3rd is for squad).

As per the below mentioned results it seems the group 2 perform well in mode 3 i.e. in squad and group 1 perform least in mode 1 i.e. solo. We also check the members for group 2 and group 1 for the same:

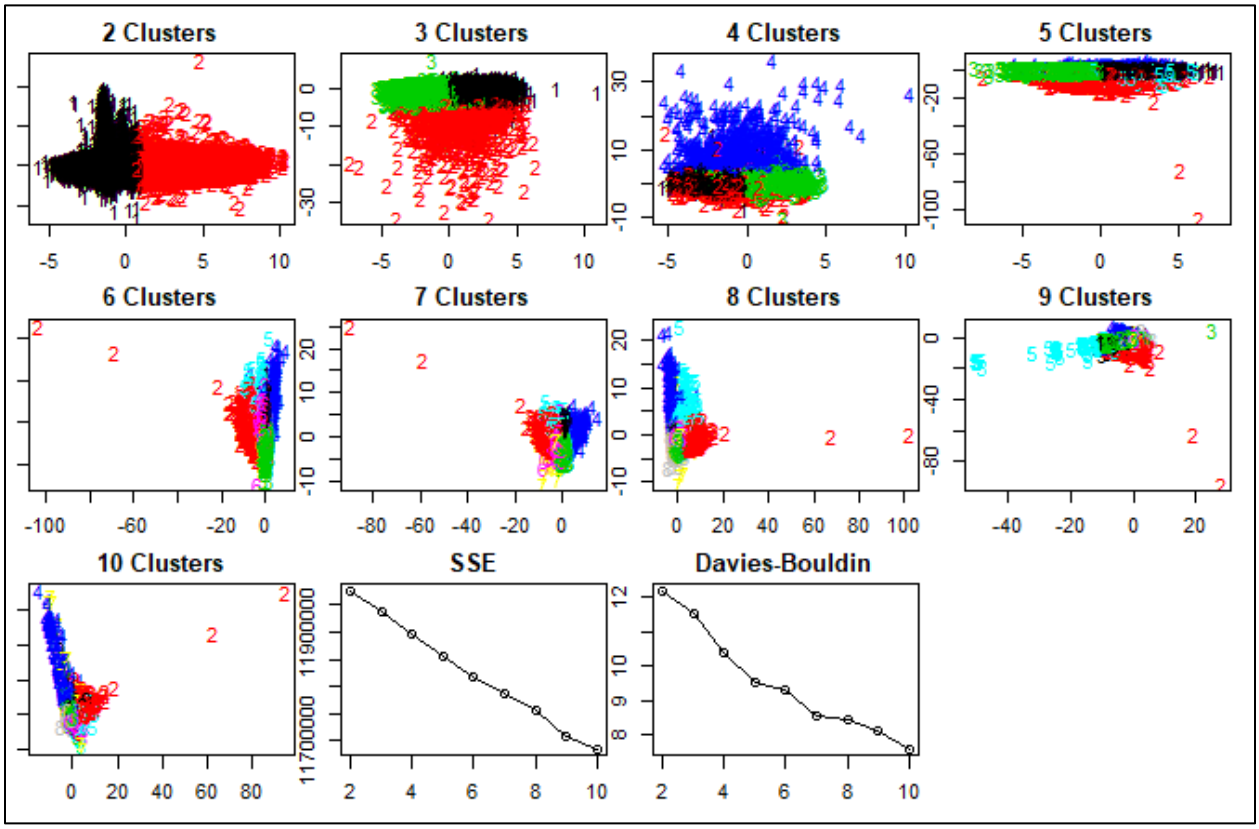
```
> print(clus.pubg_statistics.std$size)
[1] 14038  2969 24754   253 14309  6503  4706 19236
> print(paste("Standard: Highest performance mode Group is g", highest.perf.mode.group, sep=""))
[1] "Standard: Highest performance mode Group is g2"
> print(paste("Standard: Lowest performance mode Group is g", lowest.perf.mode.group, sep=""))
[1] "Standard: Lowest performance mode Group is g1"
> head(highest.perf.mode.group.std[order(highest.perf.mode.group.std$comparison, decreasing=TRUE), ])
       player_name comparison
54  Nicknameilanda          3
67    TreadstoneTW          3
81       AceAcephd          3
89       Autumn_KR          3
122      patbingsu          3
133       PlayerJP          3
> tail(lowest.perf.mode.group.std[order(lowest.perf.mode.group.std$comparison, decreasing=TRUE), ])
         player_name comparison
87795   FelipeThePut          1
87796 RussianOfPeace          1
87862       vPsych0v          1
87863           ZZAL          1
87866     PhantomACE          1
87877       Haiku575          1
```

4.3. Clustering: Whitened Raw Dataset

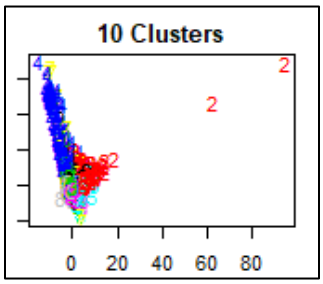
We have performed cluster analysis from range 2 to 10 on Whitened Raw Dataset. For cluster analysis, we have assigned the colors and numbers as per the clusters, as we didn't have any attributes for classification.

On the basis of **DBI value** we choose the **optimal value for Whitened Raw Dataset is 10.**

We got the below mentioned results for K=2 to 10.



Below mentioned is the **Optimal Value for Whitened Raw dataset i.e. 10**



Below mentioned are the values for the **best seed, wrong classification and centroids** for the cluster:

```
> clus.pubg_statistics.white <- clustering_euclidean(pubg_statistics.white, pubg_statistics,10)
[1] "Best Seed for the Cluster Size  10 is  2"
[1] "Total Wrong Classification in Cluster Size  10 is  0"
[1] "Centroids for the Cluster Size  10 are :"
    tracker_id solo_KillDeathRatio solo_WinRatio solo_TimeSurvived solo_RoundsPlayed   solo_Wins
1   0.01245574        -0.024479691   -0.06932000      -0.045282482        0.02943835 -0.04211802
2   0.11448280        -0.015650919   -0.06241414      -0.164191883        0.08912916 -0.01871233
3   0.03782864         0.026049726    0.05370238      -0.076846517       -0.06781813  0.03273736
4  -0.31712435         0.008260687    0.10378754      -0.178688572        0.06880475 -0.02276476
5   0.04197580        -0.081268977   -0.04121367       2.069627060        0.20246076  0.17409609
6  -0.16998683         0.007673673    0.01492878       0.024961104        0.04505882 -0.04874576
```

Below mentioned is the values for the **sum of squares** for the cluster:

```
Within cluster sum of squares by cluster:
 [1] 3229229.8  583585.2 3042885.8  384566.5  724417.4 1305354.0  236115.6  843630.5 1045648.3  288583.6
 (between_SS / total_SS =   3.0 %)
```

Then we started the cluster analysis, we divided the clusters into 10 groups on the basis of $comparison value to check people perform good in which mode (1: is for Solo, 2 is for Duo, 3rd is for squad).

As per the below mentioned results it seems the group 10 perform well in mode 3 i.e. in squad and group 2 perform least in mode 1 i.e. solo. We also check the members for group 10 and group 3 for the same:

```
> print(paste("Whitened: Highest performance mode Group is g", highest.perf.mode.group, sep=""))
[1] "Whitened: Highest performance mode Group is g10"
> print(paste("Whitened: Lowest performance mode Group is g", lowest.perf.mode.group, sep=""))
[1] "Whitened: Lowest performance mode Group is g2"
> head(highest.perf.mode.group.white[order(highest.perf.mode.group.white$comparison, decreasing=TRUE), ])
     player_name comparison
147     denahuen          3
308     Bierbank          3
365     WHITEGOM          3
390       Kemba7          3
443 ikilledhomer          3
593      DragoGo          3
> tail(lowest.perf.mode.group.white[order(lowest.perf.mode.group.white$comparison, decreasing=TRUE), ])
        player_name comparison
87793        Vaas17          1
87800        Scoobs          1
87819 Marijuana_Kid          1
87834  Papsmearicle          1
87889          apsy          1
87890        hierzn          1
```

4.4. Data Reduction: Standardized Dataset Clustering

Based on the results achieved by raw, standardized and whitened dataset it appears that standardized dataset is better in comparison to raw and whitened dataset.

Then we reduced our dataset using standardized dataset.

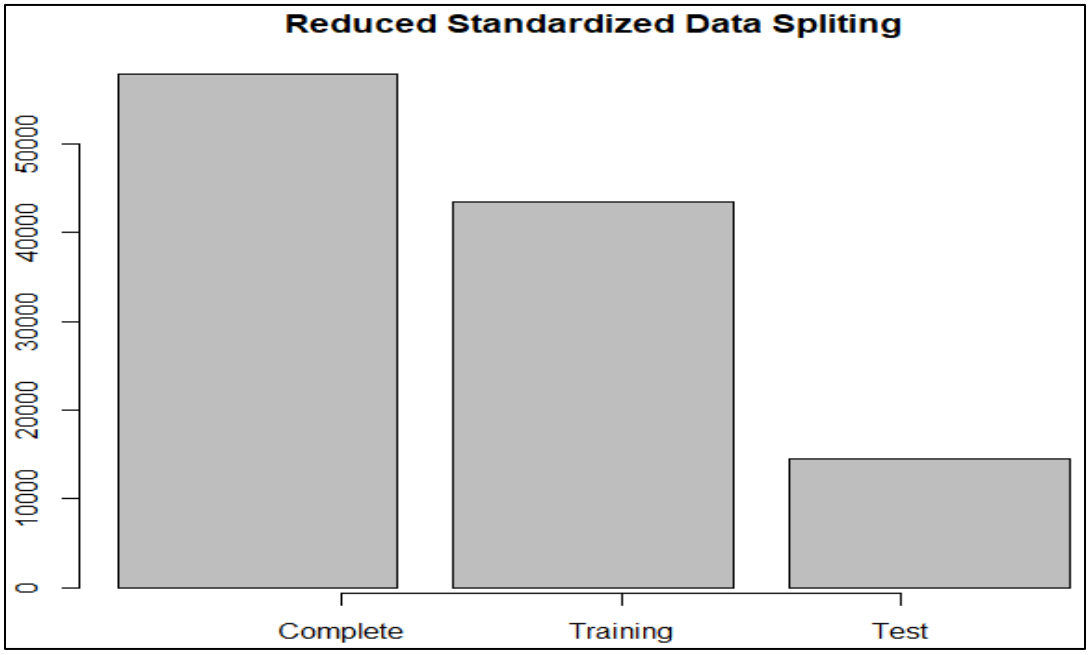
Based on Standardized dataset cluster distribution we took sample of size 2/3 from raw dataset.

Then we reduced the dataset based on standard dataset clustering with proper distribution of reduced dataset.

Then we divided the training and test dataset based on tracker id values.

Now our reduced Training dataset have 43409 rows out of 65081 and reduced Test dataset have 14436 rows out of 21687.

```
> nrow(pubg_statistics.std_reduce[pubg_statistics.std_reduce$tracker_id>164880,])
[1] 14436
> nrow(pubg_statistics.std_reduce[pubg_statistics.std_reduce$tracker_id<=164880,])
[1] 43409
```



5. Unsupervised Learning

It is the branch of machine learning in which the tasks learn from the unclassified, unlabeled and uncategorized data. The tasks try to find the commonalities and react accordingly.

In unsupervised learning we are performing below mentioned analysis:

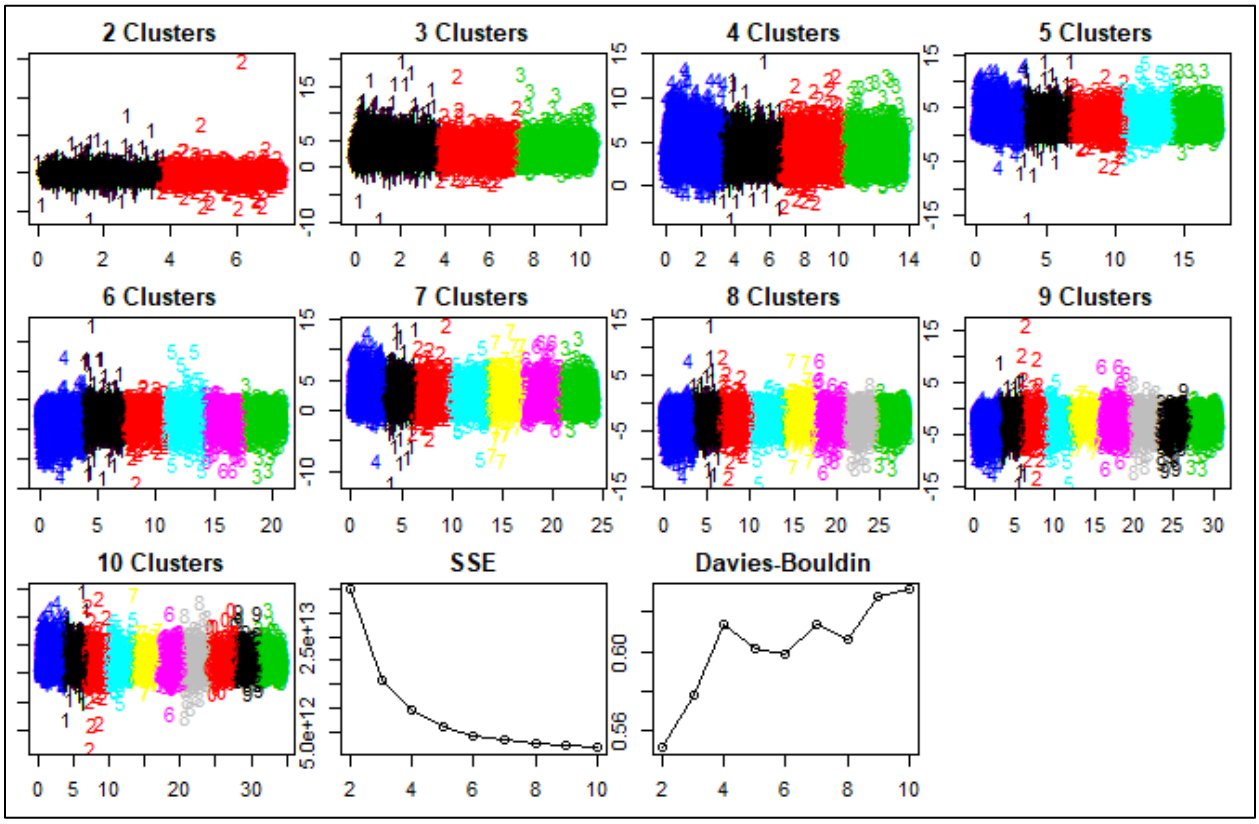
- We have performed unsupervised learning on training dataset, test dataset, PCA dataset, standardized training dataset after PCA, standardized test dataset after PCA, ICA dataset, whitened training dataset after ICA and whitened test dataset after ICA. Optimal value for training dataset, test dataset, PCA dataset, standardized training dataset after PCA, standardized test dataset after PCA, ICA dataset, whitened training dataset after ICA and whitened test dataset after ICA are 2, 2,5,7,5,6,3 respectively.
- Then we are creating groups on the basis of optimal value received and using comparison value to check which group performed well in which mode.
- Then we are also checking the group members which perform well and poor (as in highest performance mode and lowest performance mode respectively)
- It seems that most of the members work well in squad mode and few of the members don't work well in single mode.
- The sum of squares for training dataset, test dataset, PCA dataset, standardized training dataset after PCA, standardized test dataset after PCA, ICA dataset, whitened training dataset after ICA and whitened test dataset after ICA are 76%, 78%, 57%,63%,56.7,48.6% respectively.
- On the basis of analysis performed it seems that PCA dataset gives better results in comparison to ICA dataset.

5.1. Clustering: Training Dataset

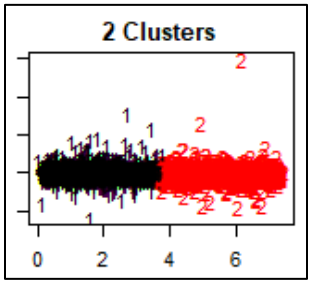
We have performed cluster analysis from range 2 to 10 on Training Dataset. For cluster analysis, we have assigned the colors and numbers as per the clusters, as we didn't have any attributes for classification.

On the basis of **DBI value** we choose the **optimal value for Training Dataset is 2.**

We got the below mentioned results for K=2 to 10.



Below mentioned is the **Optimal Value for Training dataset i.e. 2**



Below mentioned are the values for the **best seed, wrong classification and centroids** for the cluster:

```
> clus.pubg_statistics_train_un <- clustering_euclidean(pubg_statistics_train[,c(2:4,6,14:18,22,53:54,56,64:68,72
,103:104,106,114:118,122,153)], clust.pubg_statistics_train,2)
[1] "Best Seed for the Cluster Size  2 is  2"
[1] "Total Wrong Classification in Cluster Size  2 is  0"
[1] "Centroids for the Cluster Size  2 are :"
  tracker_id solo_KillDeathRatio solo_WinRatio solo_RoundsPlayed solo_DamagePg solo_HeadshotKillsPg
1   40582.92            2.068992      5.885605          81.36831      210.9929             0.4198889
2  124565.82            1.775480      4.616481          79.95424      187.3079             0.3590675
  solo_HealsPg solo_KillsPg solo_MoveDistancePg solo_TimeSurvivedPg duo_KillDeathRatio duo_WinRatio
1     1.492504     1.836585            2994.652           1000.9267           1.651980     5.531505
2     1.386666     1.616891            2762.210            965.3976           1.388501     4.233427
```

Below mentioned is the values for the **sum of squares** for the cluster:

```
Within cluster sum of squares by cluster:
[1] 1.651805e+13 1.845421e+13
 (between_SS / total_SS =  76.6 %)
```

Then we started the cluster analysis, we divided the clusters into 2 groups on the basis of $comparison value to check people perform good in which mode (1: is for Solo, 2 is for Duo, 3rd is for squad).

As per the below mentioned results it seems the group 2 perform well in mode 3 i.e. in squad and group 1 perform least in mode 1 i.e. solo. We also check the members for group 2 and group 1 for the same:

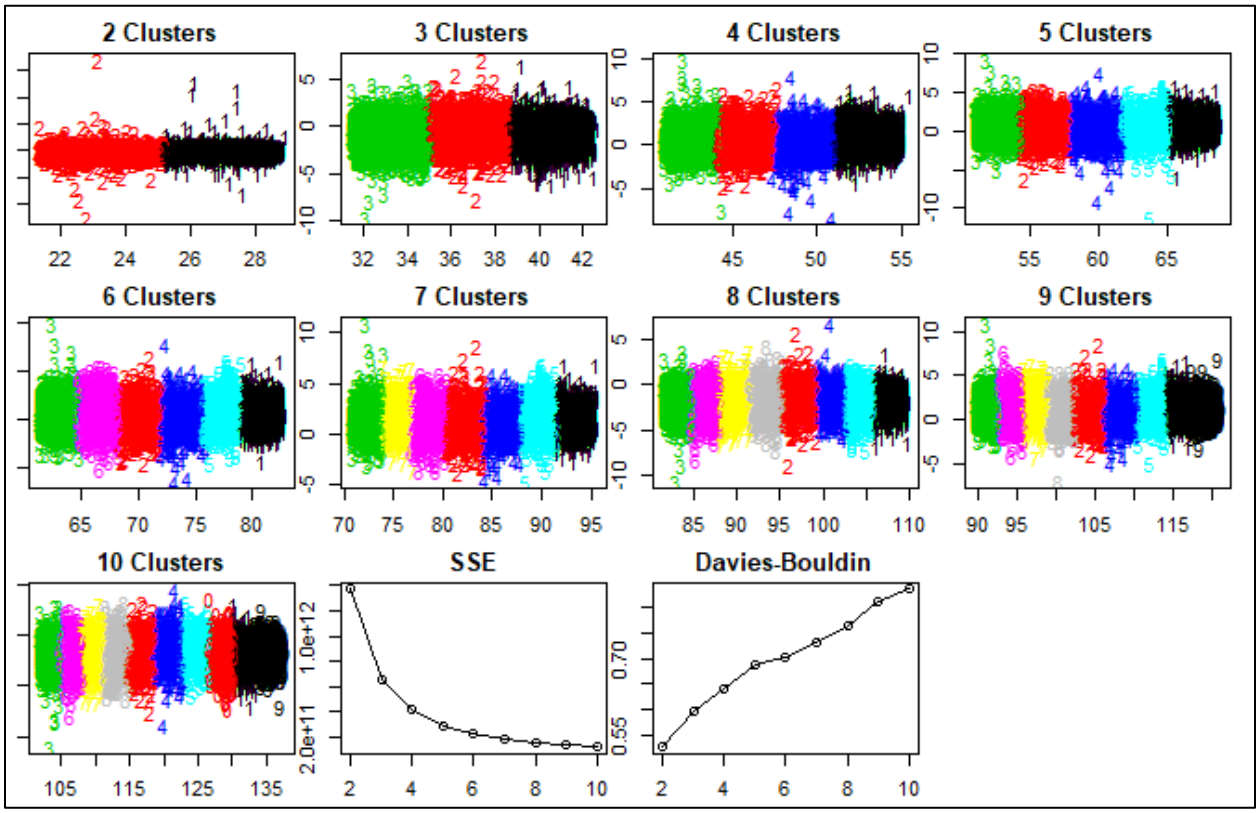
```
> print(clus.pubg_statistics_train$size)
[1] 31954 33127
> print(paste("Raw Training: Highest performance mode Group is g", highest.perf.mode.group1, sep=""))
[1] "Raw Training: Highest performance mode Group is g2"
> print(paste("Raw Training: Lowest performance mode Group is g", lowest.perf.mode.group1, sep=""))
[1] "Raw Training: Lowest performance mode Group is g1"
> head(highest.perf.mode.group_train[order(highest.perf.mode.group_train$comparison, decreasing=TRUE), ])
         player_name comparison
11     PandaTV-Tongk          3
66         Joker666           3
71  Panda-WangShaoye          3
112           Zanpah          3
113      Jiaozhu1627           3
144      Faultlessly           3
> tail(lowest.perf.mode.group_train[order(lowest.perf.mode.group_train$comparison, decreasing=TRUE), ])
          player_name comparison
87884        Nepherius          1
87889             apsy          1
87890           hierzn          1
87893 Vid_TouchesButts          1
87895          KARUKOR          1
87897         Neferhor          1
```

5.2. Clustering: Test Dataset

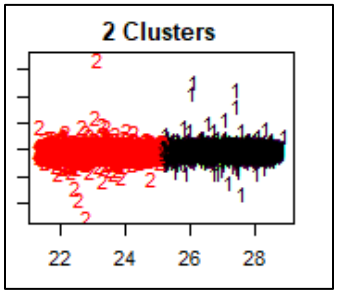
We have performed cluster analysis from range 2 to 10 on Test Dataset. For cluster analysis, we have assigned the colors and numbers as per the clusters, as we didn't have any attributes for classification.

On the basis of **DBI value** we choose the **optimal value for Test Dataset is 2.**

We got the below mentioned results for K=2 to 10.



Below mentioned is the **Optimal Value for Test Dataset i.e.2**



Below mentioned are the values for the **best seed, wrong classification and centroids** for the cluster:

```
> clus.pubg_statistics_test <- clustering_euclidean(pubg_statistics_test[,c(2:4,6,14:18,22,53:54,56,64
104,106,114:118,122,153)], clust.pubg_statistics_test,2)
[1] "Best Seed for the Cluster Size  2 is  2"
[1] "Total Wrong Classification in Cluster Size  2 is  0"
[1] "Centroids for the Cluster Size  2 are :"
  tracker_id solo_KillDeathRatio solo_WinRatio solo_RoundsPlayed solo_DamagePg solo_HeadshotKillsPg
1   208381.3            1.706096      4.427510          71.30986      180.1871            0.3392463
2   178105.3            1.769459      4.563796          75.92677      185.3999            0.3560979
  solo_HealsPg solo_KillsPg solo_MoveDistancePg solo_TimeSurvivedPg duo_KillDeathRatio duo_WinRatio
1     1.345785     1.555859            2685.848            954.9038           1.310981     3.962828
2     1.363105     1.605388            2720.627            961.4706           1.346557     4.083224
```

Below mentioned is the values for the **sum of squares** for the cluster:

```
Within cluster sum of squares by cluster:
[1] 648969638030 725179030508
 (between_SS / total_SS =  78.3 %)
```

Then we started the cluster analysis, we divided the clusters into 2 groups on the basis of $comparison value to check people perform good in which mode (1: is for Solo, 2 is for Duo, 3rd is for squad).

As per the below mentioned results it seems the group 2 perform well in mode 3 i.e. in squad and group 1 perform least in mode 1 i.e. solo. We also check the members for group 2 and group 1 for the same:

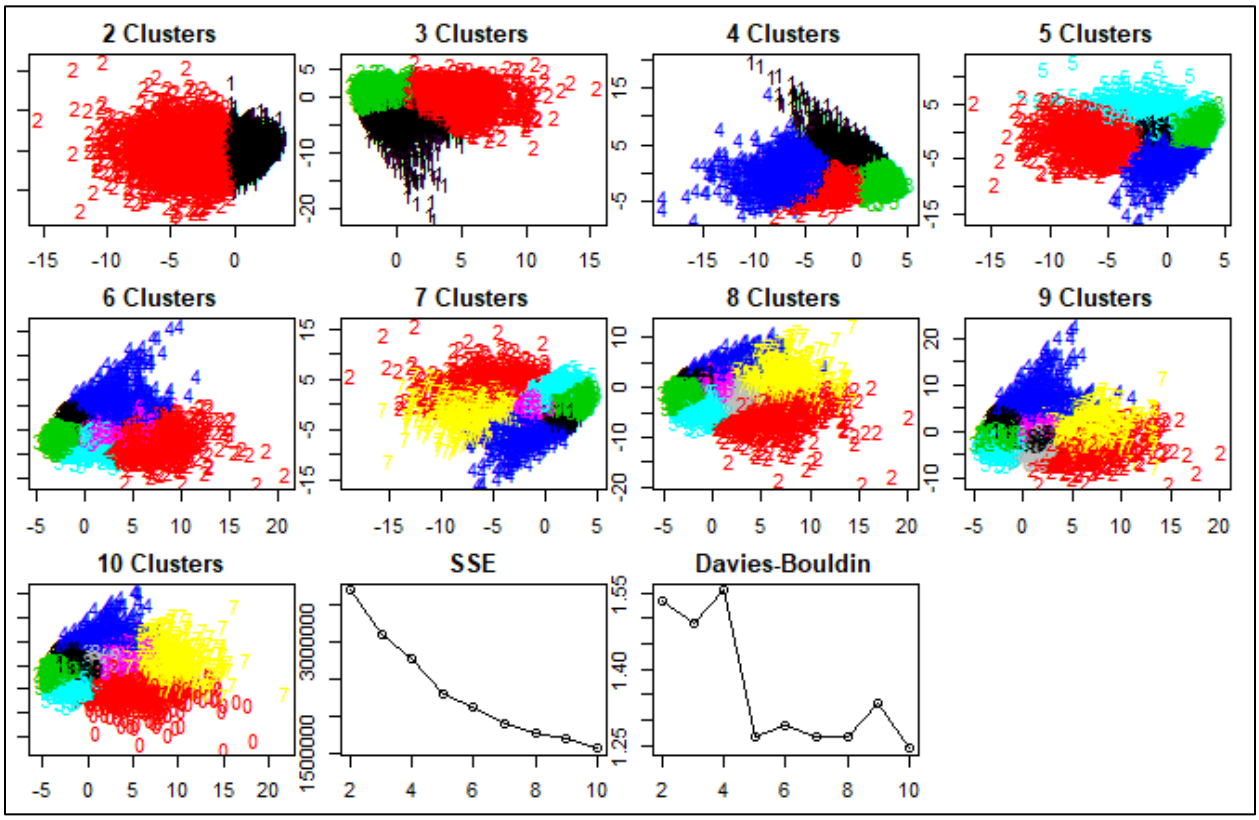
```
> print(clus.pubg_statistics_test$size)
[1] 10721 10966
> print(paste("Raw Test: Highest performance mode Group is g", highest.perf.mode.group2, sep=""))
[1] "Raw Test: Highest performance mode Group is g2"
> print(paste("Raw Test: Lowest performance mode Group is g", lowest.perf.mode.group2, sep=""))
[1] "Raw Test: Lowest performance mode Group is g1"
> head(highest.perf.mode.group_test[order(highest.perf.mode.group_test$comparison, decreasing=TRUE), ])
   player_name comparison
7        Giken          3
12    Benny-_-          3
66    Joker666          3
73      Meluke          3
81   AceAcephd          3
89   Autumn_KR          3
> tail(lowest.perf.mode.group_test[order(lowest.perf.mode.group_test$comparison, decreasing=TRUE), ])
      player_name comparison
87879  Scarsicked          1
87880 RoseFlunder          1
87882     overjoy          1
87889        apsy          1
87894   SaikoMene          1
87897    Neferhor          1
```

5.3. Clustering: Standardized Raw Dataset After PCA

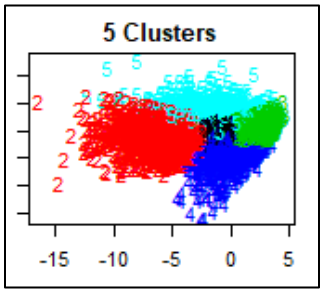
We have performed cluster analysis from range 2 to 10 on Standardized Raw Dataset after PCA. For cluster analysis, we have assigned the colors and numbers as per the clusters, as we didn't have any attributes for classification.

On the basis of **DBI value** we choose the **optimal value for Standardized Raw Dataset after PCA is 5.**

We got the below mentioned results for K=2 to 10.



Below mentioned is the **Optimal Value for Standardized Raw Dataset after PCA i.e. 5**



Below mentioned are the values for the **best seed, wrong classification and centroids** for the cluster:

```
> clus.pc.pubg_statistics <- clustering_euclidean(pc1.pubg_statistics, pubg_statistics,5)
[1] "Best Seed for the Cluster Size  5 is  2"
[1] "Total Wrong Classification in Cluster Size  5 is  0"
[1] "Centroids for the Cluster Size  5 are :"
          PC1        PC2        PC3
1  -1.4606736 -0.4256525 -2.6438014
2 -11.3631497 -2.0887832 -2.6637218
3   4.2175487  0.7333293 -0.2150916
4  -0.2318874 -4.4539348  3.9397581
5  -4.2585366  6.2740011  2.7734361
K-means clustering with 5 clusters of sizes 21282, 6588, 35431, 14019, 9448
```

Below mentioned is the values for the **sum of squares** for the cluster:

```
within cluster sum of squares by cluster:
[1] 368615.3 456225.0 530820.1 472654.0 468847.8
 (between_SS / total_SS =  55.7 %)
```

Then we started the cluster analysis, we divided the clusters into 5 groups on the basis of $comparison value to check people perform good in which mode (1: is for Solo, 2 is for Duo, 3rd is for squad).

As per the below mentioned results it seems the group 5 perform well in mode 3 i.e. in squad and group 1 perform least in mode 1 i.e. solo. We also check the members for group 5 and group 1 for the same:

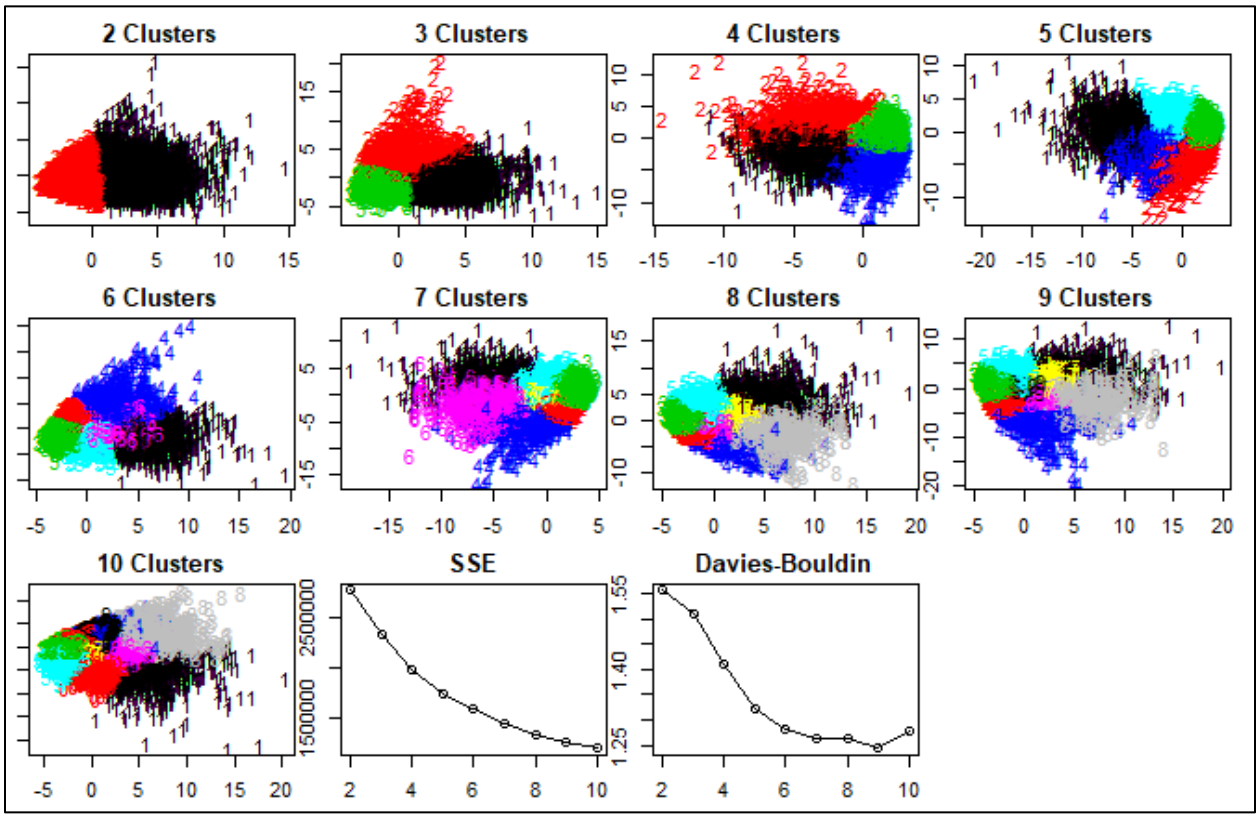
```
> print(clus.pc.pubg_statistics$size)
[1] 21282  6588 35431 14019  9448
> print(paste("PCA on raw dataset: Highest performance mode Group is g", highest.perf.mode.group3, sep=""))
[1] "PCA on raw dataset: Highest performance mode Group is g5"
> print(paste("PCA on raw dataset: Lowest performance mode Group is g", lowest.perf.mode.group3, sep=""))
[1] "PCA on raw dataset: Lowest performance mode Group is g1"
> head(highest.perf.mode.group.pc[order(highest.perf.mode.group.pc$comparison, decreasing=TRUE), ])
      player_name comparison
54  Nicknameilanda          3
81       AceAcephd          3
89       Autumn_KR          3
122      patbingsu          3
133       PlayerJP          3
146         ALDOGG          3
> tail(lowest.perf.mode.group.pc[order(lowest.perf.mode.group.pc$comparison, decreasing=TRUE), ])
        player_name comparison
87840   DirtyDouglas          1
87858  CheckMySteeze          1
87869 Old_Man_Willux          1
87884      Nepherius          1
87889           apsy          1
87892       IssacC0x          1
```

5.4. Clustering: Standardized Training Dataset After PCA

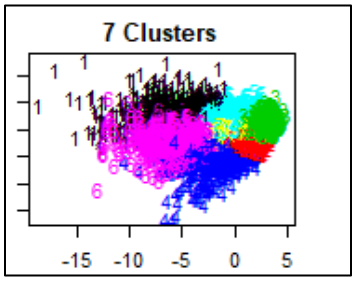
We have performed cluster analysis from range 2 to 10 on Standardized Training Dataset after PCA. For cluster analysis, we have assigned the colors and numbers as per the clusters, as we didn't have any attributes for classification.

On the basis of **DBI value** we choose the **optimal value for Standardized Training Dataset after PCA is 7.**

We got the below mentioned results for K=2 to 10.



Below mentioned is the **Optimal Value for Standardized Training Dataset after PCA i.e. 7**



Below mentioned are the values for the **best seed, wrong classification and centroids** for the cluster:

```
> clus.pc.pubg_statistics_train <- clustering_euclidean(pc1.pubg_statistics_train, pubg_statistics_train,7)
[1] "Best Seed for the Cluster Size  7 is  2"
[1] "Total Wrong Classification in Cluster Size  7 is  0"
[1] "Centroids for the Cluster Size  7 are :"
         PC1        PC2        PC3
1 -11.3859045 -8.5464575  4.2762045
2   2.1905859  2.6245105  2.4719589
3   4.8656820 -0.8985932 -0.9451584
4  -4.6614051  6.3526834  5.6835438
5  -0.9966407 -4.2654697  1.2101847
6  -9.9496482  1.9502215 -4.1680694
7  -1.2416452  1.1336304 -2.9472446
K-means clustering with 7 clusters of sizes 2125, 12438, 18107, 3845, 10893, 4643, 13030
```

Below mentioned is the values for the **sum of squares** for the cluster:

```
Within cluster sum of squares by cluster:
[1] 182607.5 159483.2 215893.2 214720.2 222655.0 234627.2 204023.4
 (between_SS / total_SS =  63.0 %)
```

Then we started the cluster analysis, we divided the clusters into 7 groups on the basis of $comparison value to check people perform good in which mode (1: is for Solo, 2 is for Duo, 3rd is for squad).

As per the below mentioned results it seems the group 1 perform well in mode 3 i.e. in squad and group 7 perform least in mode 1 i.e. solo. We also check the members for group 1 and group 7 for the same:

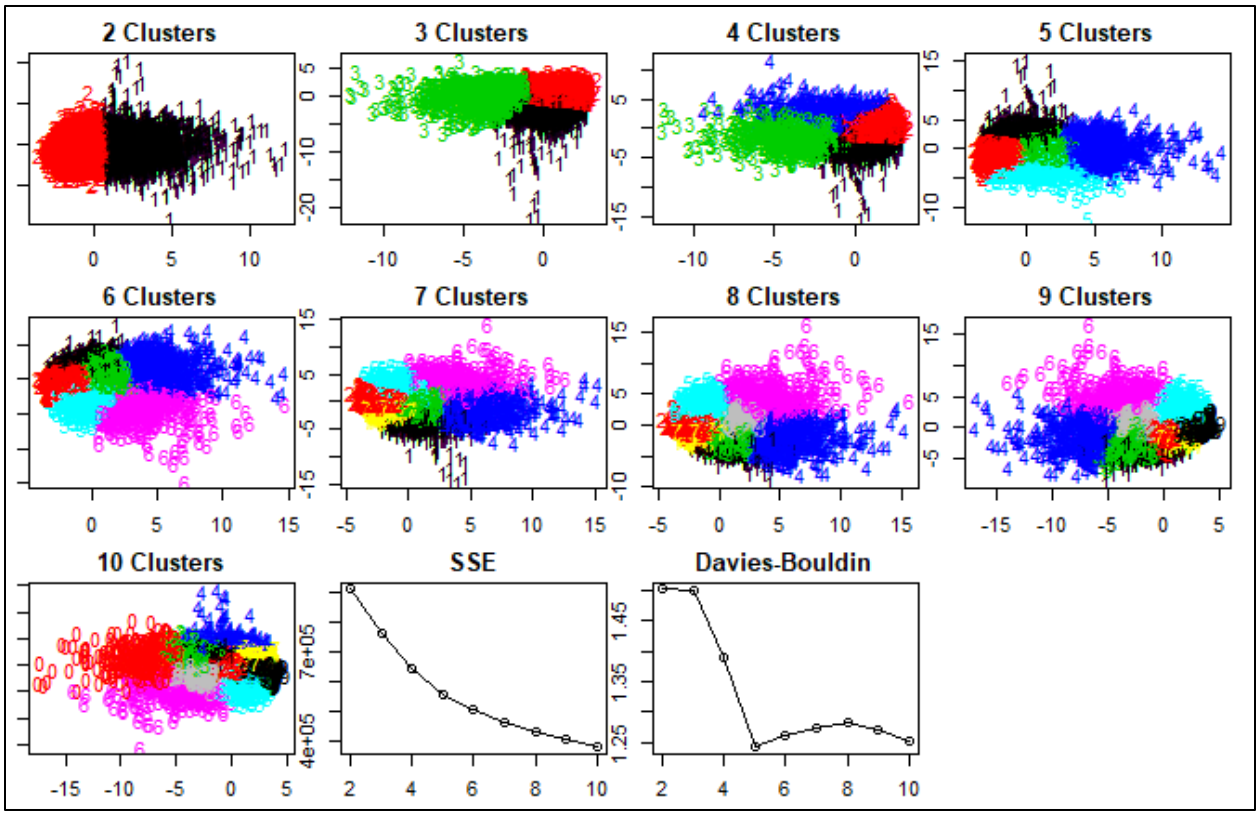
```
> print(clus.pc.pubg_statistics_train$size)
[1]  2125 12438 18107  3845 10893  4643 13030
> print(paste("Train DataSet After PCA: Highest performance mode Group is g", highest.perf.mode.group4, sep=""))
[1] "Train DataSet After PCA: Highest performance mode Group is g1"
> print(paste("Train DataSet After PCA: Lowest performance mode Group is g", lowest.perf.mode.group4, sep=""))
[1] "Train DataSet After PCA: Lowest performance mode Group is g7"
> head(highest.perf.mode.group.pc1[order(highest.perf.mode.group.pc1$comparison, decreasing=TRUE), ])
       player_name comparison
54  Nicknameilanda          3
67    TreadstoneTW          3
81       AceAcephd          3
89       Autumn_KR          3
122      patbingsu          3
133       PlayerJP          3
> tail(lowest.perf.mode.group.pc1[order(lowest.perf.mode.group.pc1$comparison, decreasing=TRUE), ])
       player_name comparison
87825     AreOhBee          1
87832 Handy_Banana          1
87834 Papsmearicle          1
87836     sjyoon97          1
87884    Nepherius          1
87889         apsy          1
```

5.5. Clustering: Standardized Test Dataset After PCA

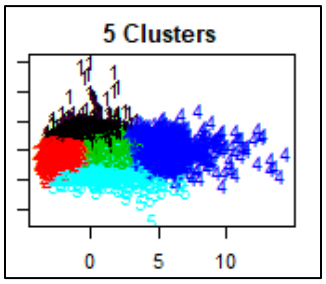
We have performed cluster analysis from range 2 to 10 on Standardized Test Dataset after PCA. For cluster analysis, we have assigned the colors and numbers as per the clusters, as we didn't have any attributes for classification.

On the basis of **DBI value** we choose the **optimal value for Standardized Test Dataset after PCA is 5.**

We got the below mentioned results for K=2 to 10.



Below mentioned is the **Optimal Value for Standardized Test Dataset after PCA i.e. 5**



Below mentioned are the values for the **best seed, wrong classification and centroids** for the cluster:

```
> clus.pc.pubg_statistics_test <- clustering_euclidean(pc1.pubg_statistics_test, pubg_statistics_test,5)
[1] "Best Seed for the Cluster Size  5 is  2"
[1] "Total Wrong Classification in Cluster Size  5 is  0"
[1] "Centroids for the Cluster Size  5 are :"
          PC1        PC2        PC3
1   0.623919  4.5514355 -3.8781234
2   4.127236 -0.6245417  0.7304576
3  -2.005670  0.8507236  2.5824939
4 -12.306400  1.1292984  1.5271236
5  -2.459479 -5.7171445 -2.7823149
K-means clustering with 5 clusters of sizes 3506, 8592, 5005, 1659, 2925
```

Below mentioned is the values for the **sum of squares** for the cluster:

```
Within cluster sum of squares by cluster:
[1] 119701.85 123862.63  87629.02 116019.22 112122.78
 (between_SS / total_SS =  56.7 %)
```

Then we started the cluster analysis, we divided the clusters into 5 groups on the basis of $comparison value to check people perform good in which mode (1: is for Solo, 2 is for Duo, 3rd is for squad).

As per the below mentioned results it seems the group 5 perform well in mode 3 i.e. in squad and group 3 perform least in mode 1 i.e. solo. We also check the members for group 5 and group 3 for the same:

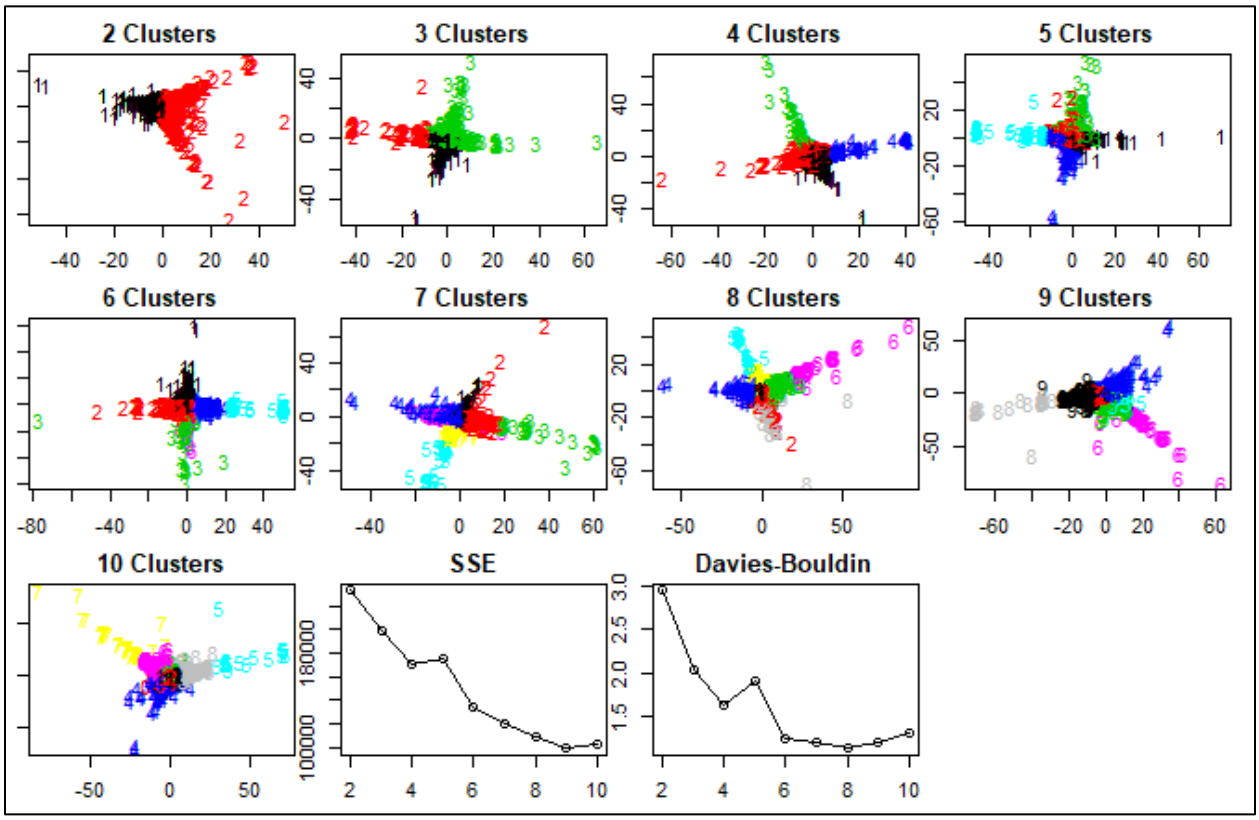
```
> print(clus.pc.pubg_statistics_test$size)
[1] 3506 8592 5005 1659 2925
> print(paste("Test dataset after PCA: Highest performance mode Group is g", highest.perf.mode.group5, sep=""))
[1] "Test dataset after PCA: Highest performance mode Group is g5"
> print(paste("Test dataset after PCA: Lowest performance mode Group is g", lowest.perf.mode.group5, sep=""))
[1] "Test dataset after PCA: Lowest performance mode Group is g3"
> head(highest.perf.mode.group.pc2[order(highest.perf.mode.group.pc2$comparison, decreasing=TRUE), ])
    player_name comparison
433 AkdongPadaK          3
718     HanHo89          3
780      d2ongh          3
882  NALGOSIPDA          3
949      zirnan          3
985       sPray          3
> tail(lowest.perf.mode.group.pc2[order(lowest.perf.mode.group.pc2$comparison, decreasing=TRUE), ])
         player_name comparison
87593    AndySharper          1
87683       MrTruong          1
87705 elitescavenger          1
87733       Deathlly          1
87817    Hihi_Hehe-1          1
87871           Kibb          1
```

5.6. Clustering: Whitened Raw Dataset After ICA

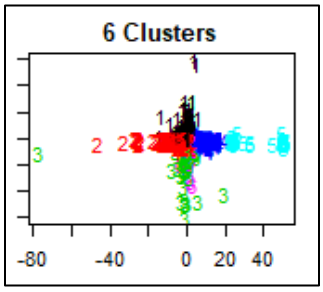
We have performed cluster analysis from range 2 to 10 on Whitened Raw Dataset after ICA. For cluster analysis, we have assigned the colors and numbers as per the clusters, as we didn't have any attributes for classification.

On the basis of **DBI value** we choose the **optimal value for Whitened Raw Dataset after ICA is 6.**

We got the below mentioned results for K=2 to 10.



Below mentioned is the **Optimal Value for Whitened Raw Dataset after ICA i.e. 6**



Below mentioned are the values for the **best seed, wrong classification and centroids** for the cluster:

```
> clus.pubg_statistics.ica <- clustering_euclidean(pubg_statistics.ica, pubg_statistics,6)
[1] "Best Seed for the Cluster Size  6 is  2"
[1] "Total Wrong Classification in Cluster Size  6 is  0"
[1] "Centroids for the Cluster Size  6 are :"
          [,1]        [,2]        [,3]
1   0.04148071 -0.30607044 -0.38688349
2   0.09993009  0.33295121  0.43995999
3  -0.07595058 -0.49211770 11.58851225
4  -3.84077379  0.04436038  0.06914563
5 -22.98276404  0.21611763  0.05499158
6  -0.16043833 14.88917818 -3.89759562
K-means clustering with 6 clusters of sizes 47339, 37952, 175, 1113, 63, 126
```

Below mentioned is the values for the **sum of squares** for the cluster:

```
Within cluster sum of squares by cluster:
[1] 41260.410 53789.489 14789.767  6221.438  4319.308 13288.489
 (between_SS / total_SS =  48.6 %)
```

Then we started the cluster analysis, we divided the clusters into 6 groups on the basis of $comparison value to check people perform good in which mode (1: is for Solo, 2 is for Duo, 3rd is for squad).

As per the below mentioned results it seems the group 6 perform well in mode 3 i.e. in squad and group 4 perform least in mode 1 i.e. solo. We also check the members for group 6 and group 4 for the same:

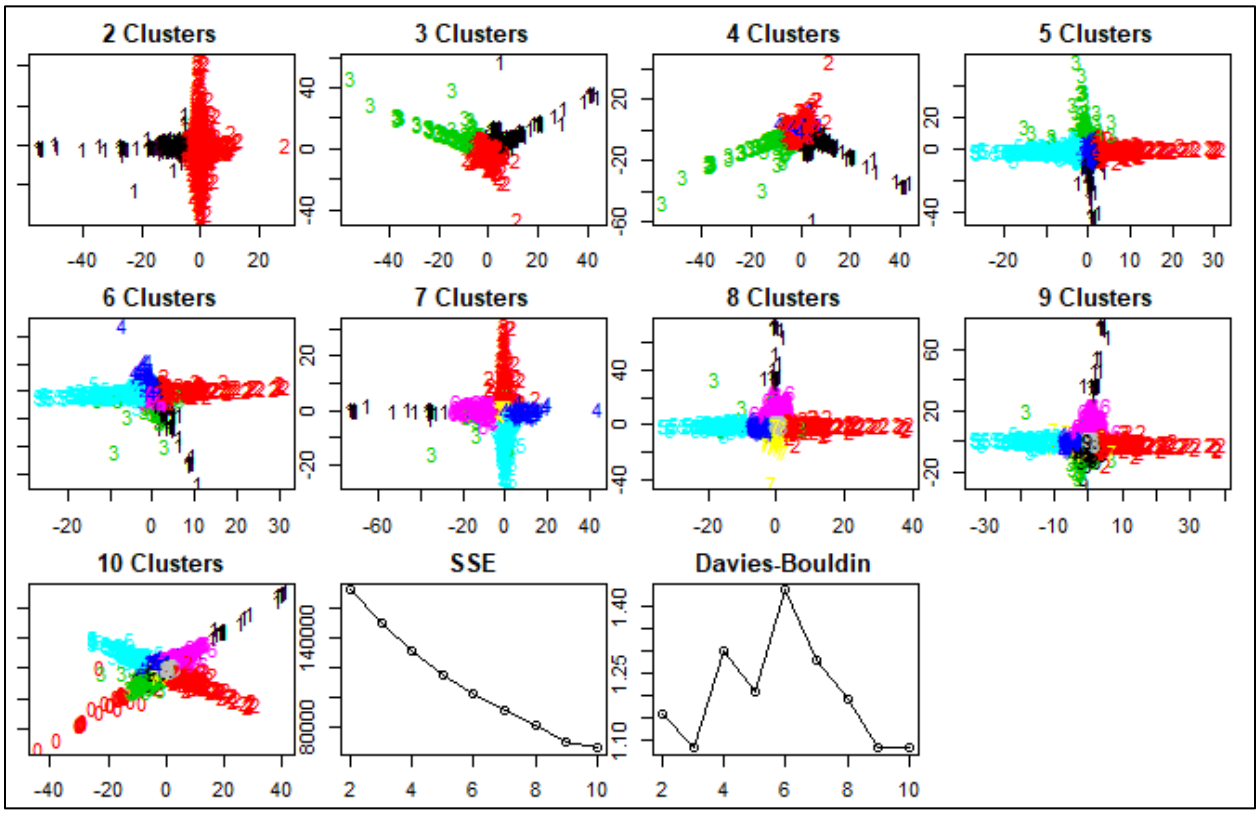
```
> print(clus.pubg_statistics.ica$size)
[1] 47339 37952   175  1113    63   126
> print(paste("ICA on raw dataset: Highest performance mode Group is g", highest.perf.mode.group8, sep=""))
[1] "ICA on raw dataset: Highest performance mode Group is g6"
> print(paste("ICA on raw dataset: Lowest performance mode Group is g", lowest.perf.mode.group8, sep=""))
[1] "ICA on raw dataset: Lowest performance mode Group is g4"
> head(highest.perf.mode.group.ica[order(highest.perf.mode.group.ica$comparison, decreasing=TRUE), ])
       player_name comparison
3298        OKIUER          3
3416        Utter_          3
14433      Digg1tS          3
36628 IllegalAsian          3
42396       OKUDERA          3
51537       Flex004          3
> tail(lowest.perf.mode.group.ica[order(lowest.perf.mode.group.ica$comparison, decreasing=TRUE), ])
         player_name comparison
86899     ParanoiiiD          1
87124        TimGomm          1
87429   KovuTheHusky          1
87437 ScrooogeMcDuck          1
87566       MrrRobin          1
87702         Souusa          1
```

5.7. Clustering: Whitened Training Dataset After ICA

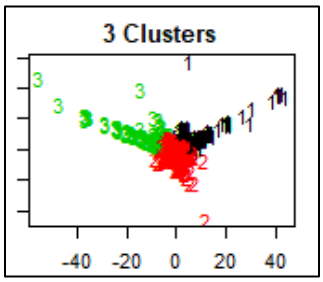
We have performed cluster analysis from range 2 to 10 on Whitened Training Dataset after ICA. For cluster analysis, we have assigned the colors and numbers as per the clusters, as we didn't have any attributes for classification.

On the basis of **DBI value** we choose the **optimal value for Whitened Training Dataset after ICA is 3.**

We got the below mentioned results for K=2 to 10.



Below mentioned is the **Optimal Value for Standardized Raw Dataset after PCA i.e. 3**



Below mentioned are the values for the **best seed, wrong classification and centroids** for the cluster:

```
> clus.pubg_statistics_train.ica <- clustering_euclidean(pubg_statistics_train.ica, pubg_statistics_train,3)
[1] "Best Seed for the Cluster Size  3 is  2"
[1] "Total Wrong Classification in Cluster Size  3 is  0"
[1] "Centroids for the Cluster Size  3 are :"
          [,1]        [,2]        [,3]
1  0.1304419203  0.52717135 -7.52671048
2 -0.0005129521 -0.03301709  0.03860205
3 -0.1019069240 11.47360446  2.42184954
K-means clustering with 3 clusters of sizes 385, 64528, 168
```

Below mentioned is the values for the **sum of squares** for the cluster:

```
Within cluster sum of squares by cluster:
[1]  18589.87 118657.92  12801.13
 (between_SS / total_SS =  23.1 %)
```

Then we started the cluster analysis, we divided the clusters into 3 groups on the basis of $comparison value to check people perform good in which mode (1: is for Solo, 2 is for Duo, 3rd is for squad).

As per the below mentioned results it seems the group 3 perform well in mode 3 i.e. in squad and group 2 perform least in mode 1 i.e. solo. We also check the members for group 3 and group 2 for the same:

```
> print(clus.pubg_statistics_train.ica$size)
[1]   385 64528   168
> print(paste("Train DataSet After ICA: Highest performance mode Group is g", highest.perf.mode.group9, sep=""))
[1] "Train DataSet After ICA: Highest performance mode Group is g3"
> print(paste("Train DataSet After ICA: Lowest performance mode Group is g", lowest.perf.mode.group9, sep=""))
[1] "Train DataSet After ICA: Lowest performance mode Group is g2"
> head(highest.perf.mode.group.ica1[order(highest.perf.mode.group.ica1$comparison, decreasing=TRUE), ])
          player_name comparison
10612         MESTREAK          3
14433          Digg1tS          3
18719        SugahFree          3
24558         S572HAHA          3
27365    Daddypanda661          3
35515 How_old_fxxx_you          3
> tail(lowest.perf.mode.group.ica1[order(lowest.perf.mode.group.ica1$comparison, decreasing=TRUE), ])
          player_name comparison
87890           hierzn          1
87892         IssacC0x          1
87893 Vid_TouchesButts          1
87894        SaikoMene          1
87895          KARUKOR          1
87897         Neferhor          1
```

6. Supervised Learning

It is the branch of machine learning in which the task has the input-output pairs. in this, the algorithm uses the training data to find the inferred function which is then used for mapping new examples.

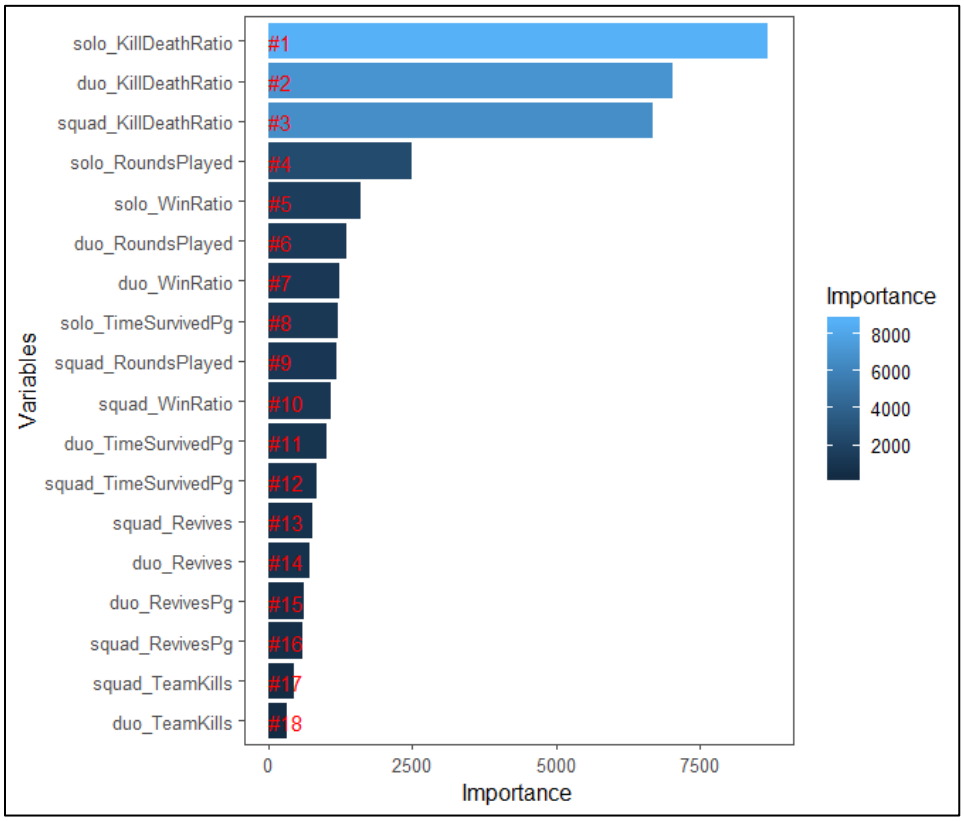
For supervised learning we are going to use Random forest to do the prediction for our test set.

6.1. Random Forest

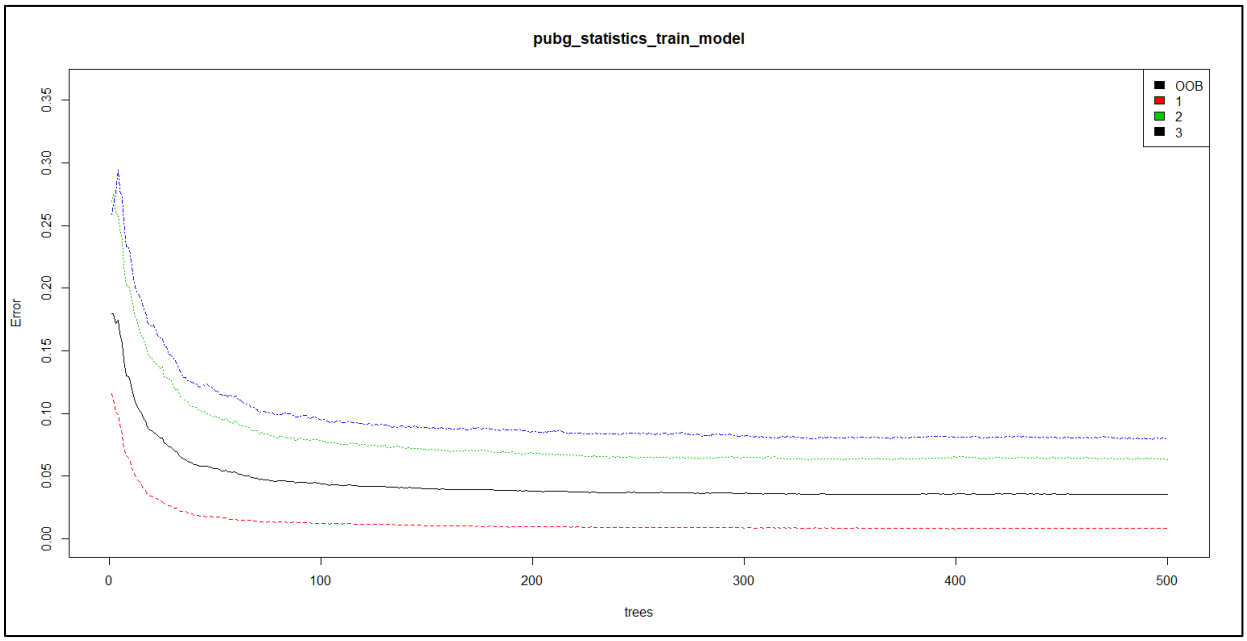
In this section we will be using the random forest package to build a model that will become the prediction stage for our test set. For random forest we are using the full dataset and have divided the dataset into train and test set on basis of tracker id values and after that we added one more column to see that a player prefers playing which of the 3 mode.

For creating model, model we are using random forest algorithm and for our formula we used some important variables that has impact on players statistics from the train_set and set Comparison as the main factor so that everything is calculated on the basis of which mode does it belong to.

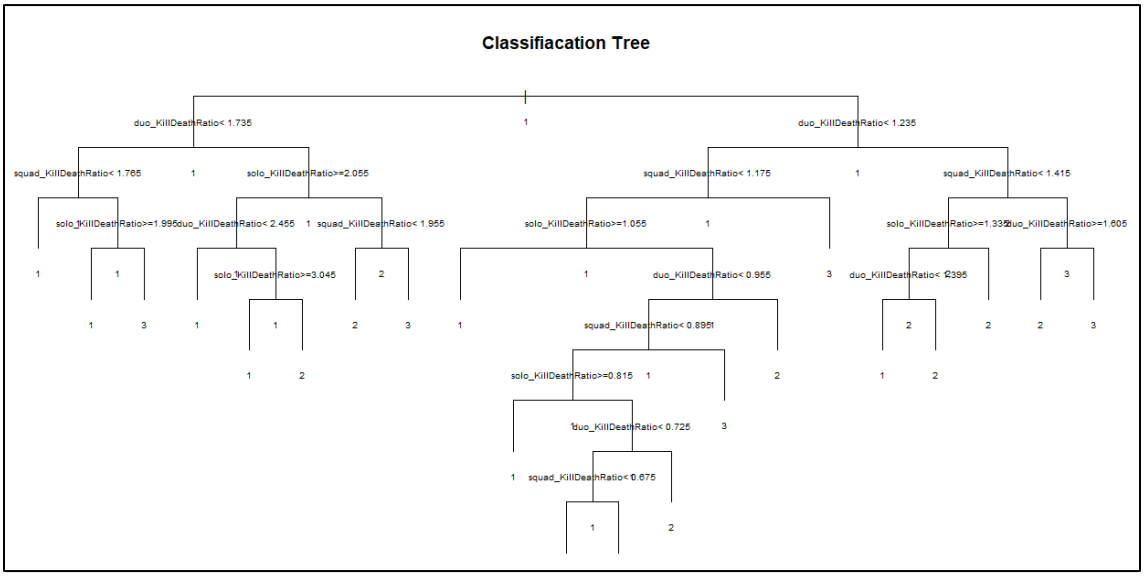
By using random forest function we generated some importance value for each variable and according to their importance we gave them rank and have plotted them using barplot function of gplot.



From the graph we can see that kill death ratio is the most important in all 3 modes and out most of the other variables are highly dependent on these variables. Then we plot the model error graph that consists of overall error rate and error rate for which mode the player prefers more.



In the graph we can see that the black line represents the overall error rate and as we can see that the overall error rate is going below 10% which is good prediction for now. Red, green and blue line shows the error rate for all 3 modes. Using Rpart we can see the classification tree using which any person can come to conclusion to which stage they stand if they are a player.



In the tree above it shows that if the kill death ratio is low then it will direct towards one side otherwise towards other side and as a person will go on it will finally reach a value that will show which type of game that player prefers most.

Contribution

Shubham and Gurveen were the primary programmers. Shubham wrote data analysis, Gurveen performed the dimension reduction using ICA and unsupervised learning. Mandeep and Vidhi are the secondary programmers. Mandeep wrote the dimension reduction using PCA and data reduction, Vidhi wrote random forest clustering.

References

[1] “PlayerUnknown's Battlegrounds.” Wikipedia, Wikimedia Foundation, 8 Dec. 2018, en.wikipedia.org/wiki/PlayerUnknown's_Battlegrounds.

[2] “PUBG Data Analysis.” PUBG Data Analysis, pubganalysis.wordpress.com/.

Appendix

Code for PUBG_Player_Statistics dataset:

```
# "drive", AND "path.up" SHOULD BE THE ONLY PARTS THAT REQUIRE YOUR PROFESSOR
# OR TA TO BE ABLE TO RUN YOUR CODE
drive="D:"
path.upto <- paste("School", "STAT5703-Data Mining", sep="/" )
code.dir <- paste(drive, path.upto, Student.info, "Code", sep="/")
data.dir <- paste(drive, path.upto, Student.info, "Data", sep="/")
work.dir <- paste(drive, path.upto, Student.info, "Work", sep="/")
setwd(work.dir)

library(ggplot2)
library(DescTools)
library(aod)
library(ROCR)
library(pROC)
library(dplyr)
library(readxl)
library(MASS)
library(Rcpp)
library(sand)
library(igraph)
library(ppcor)
library(corrplot)



#Read file and checking file contents
```

```
pubg_stat <- paste(data.dir,"PUBG_Player_Statistics.csv", sep="/")

pubg_statistics <- read.csv(pubg_stat, na.strings="")

summary(pubg_statistics)

head(pubg_statistics)

dim(pubg_statistics)

sum(is.na(pubg_statistics))

sapply(pubg_statistics, function(x) sum(is.na(x)))


#separating out solo,duo and squad data in different variables


solo_data <-pubg_statistics[,1:52]

duo_data <-pubg_statistics[,c(1:2,53:102)]

squad_data<- pubg_statistics[,c(1:2,103:152)]




hist(solo_data$solo_RoundsPlayed, breaks=100, col="grey", main = "Solo ", xlab = "no of games played",
ylab = "no of users")

Desc(solo_data1$solo_RoundsPlayed)


hist(duo_data$duo_RoundsPlayed, breaks=100, col="orange", main = "DUO ", xlab = "no of games
played", ylab = "no of users")

Desc(solo_data1$solo_RoundsPlayed)


hist(squad_data$squad_RoundsPlayed, breaks=100, col="yellow", main = "Squad ", xlab = "no of games
played", ylab = "no of users")

Desc(solo_data1$solo_RoundsPlayed)



#subset datasets according to rounds played by them
```

```
solo_data1 <- subset(solo_data, solo_RoundsPlayed >=50)

dim(solo_data1)


duo_data1 <- subset(duo_data, duo_RoundsPlayed >=50)

dim(duo_data1)


squad_data1 <- subset(squad_data, squad_RoundsPlayed >=50)

dim(squad_data1)




hist(solo_data1$solo_RoundsPlayed, breaks=100, col="grey", main = "Solo ", xlab = "no of games played", ylab = "no of users")

hist(duo_data1$duo_RoundsPlayed, breaks=100, col="orange", main = "Duo ", xlab = "no of games played", ylab = "no of users")

hist(squad_data1$squad_RoundsPlayed, breaks=100, col="yellow", main = "Squad ", xlab = "no of games played", ylab = "no of users")


#Analysis of Win Ratio in all 3 modes



hist(solo_data1$solo_WinRatio, breaks=100, col="blue", main = "Solo Win ratio", xlab = "Win %")

Desc(solo_data1$solo_WinRatio)


summary(solo_data1)



hist(duo_data1$duo_WinRatio, breaks=100, col="green", main = "Duo Win ratio", xlab = "Win %")

Desc(duo_data1$duo_WinRatio)
```

```
summary(duo_data1)



hist(squad_data1$squad_WinRatio, breaks=100, col="red", main = "Squad Win ratio", xlab = "Win %")

Desc(squad_data1$squad_WinRatio)

summary(squad_data1)



#Analysis of top 10 ratio

hist(solo_data1$solo_Top10Ratio, breaks=100, col="blue", main = "solo top 10 ratio", xlab = "top 10 %")

Desc(solo_data1$solo_Top10Ratio)

summary(solo_data1)



hist(duo_data1$duo_Top10Ratio, breaks=100, col="green", main = "duo Top 10 ratio", xlab = "top 10
%")

Desc(duo_data1$duo_Top10Ratio)

summary(duo_data1)



hist(squad_data1$squad_Top10Ratio, breaks=100, col="red", main = "squad Top 10 ratio", xlab = "top
10 %")

Desc(squad_data1$squad_Top10Ratio)
```

```
summary(squad_data1)



#Analysis of Kill to Death Ratio



hist(solo_data1$solo_KillDeathRatio, breaks=100, col="blue", main = "solo Kil Death ratio", xlab = "kill vs death %")

Desc(solo_data1$solo_KillDeathRatio)


summary(solo_data1)



hist(duo_data1$duo_KillDeathRatio, breaks=100, col="green", main = "duo Kill Death ratio", xlab = "kill vs death %")

Desc(duo_data1$duo_KillDeathRatio)


summary(duo_data1)



hist(squad_data1$squad_KillDeathRatio, breaks=100, col="red", main = "squad Kill Death ratio", xlab = "kill vs death %")

Desc(squad_data1$squad_KillDeathRatio)


summary(squad_data1)



#comparing various variables with the WinRatio to see how they affect the winning ratio for solo mode
```

```
plot(solo_data1$solo_KillDeathRatio, solo_data1$solo_WinRatio, xlab = "KD ratio",
    ylab = "Win ratio", main = "Win ratio by KD (Solo)", col = "blue", pch = 20)
Desc(solo_data1$solo_WinRatio ~ solo_data1$solo_KillDeathRatio)

plot(solo_data1$solo_MoveDistancePg, solo_data1$solo_WinRatio, xlab = "Distance moved per game
(meters)",
    ylab = "Win ratio", main = "Win ratio by Distance moved PG (Solo)", col = "chartreuse1", pch = 20)
Desc(solo_data1$solo_WinRatio ~ solo_data1$solo_MoveDistancePg)

plot(solo_data1$solo_DamagePg, solo_data1$solo_WinRatio, xlab = "Damage dealt per game",
    ylab = "Win ratio", main = "Win ratio by Damage dealt PG (Solo)", col = "red", pch = 20)
Desc(solo_data1$solo_WinRatio ~ solo_data1$solo_DamagePg)



#comparing various variables with the WinRatio to see how they affect the winning ratio for duo mode

plot(duo_data1$duo_KillDeathRatio, duo_data1$duo_WinRatio, xlab = "KD ratio",
    ylab = "Win ratio", main = "Win ratio by KD (duo)", col = "blue", pch = 20)
Desc(duo_data1$duo_WinRatio ~ duo_data1$duo_KillDeathRatio)

plot(duo_data1$duo_MoveDistancePg, duo_data1$duo_WinRatio, xlab = "Distance moved per game
(meters)",
    ylab = "Win ratio", main = "Win ratio by Distance moved PG (duo)", col = "chartreuse1", pch = 20)
Desc(duo_data1$duo_WinRatio ~ duo_data1$duo_MoveDistancePg)

plot(duo_data1$duo_DamagePg, duo_data1$duo_WinRatio, xlab = "Damage dealt per game",
    ylab = "Win ratio", main = "Win ratio by Damage dealt PG (duo)", col = "red", pch = 20)
Desc(duo_data1$duo_WinRatio ~ duo_data1$duo_DamagePg)
```

```
#comparing various variables with the WinRatio to see how they affect the winning ratio for squad mode

plot(squad_data1$squad_KillDeathRatio, squad_data1$squad_WinRatio, xlab = "KD ratio",
    ylab = "Win ratio", main = "Win ratio by KD (squad)", col = "blue", pch = 20)
Desc(squad_data1$squad_WinRatio ~ squad_data1$squad_KillDeathRatio)

plot(squad_data1$squad_MoveDistancePg, squad_data1$squad_WinRatio, xlab = "Distance moved per game (meters)",
    ylab = "Win ratio", main = "Win ratio by Distance moved PG (squad)", col = "chartreuse1", pch = 20)
Desc(squad_data1$squad_WinRatio ~ squad_data1$squad_MoveDistancePg)

plot(squad_data1$squad_DamagePg, squad_data1$squad_WinRatio, xlab = "Damage dealt per game",
    ylab = "Win ratio", main = "Win ratio by Damage dealt PG (squad)", col = "red", pch = 20)
Desc(squad_data1$squad_WinRatio ~ squad_data1$squad_DamagePg)



#lets visualize the longest kill for all 3 modes

hist(solo_data1$solo_LongestKill, xlab = 'Distance (meters)',
    col = "lightblue", main = "Solo Long Distance Kills")

hist(duo_data1$duo_LongestKill, xlab = 'Distance (meters)',
    col = "lightgreen", main = "Duo Long Distance Kills")

hist(squad_data1$squad_LongestKill, xlab = 'Distance (meters)',
    col = "orange", main = "Squad Long Distance Kills")
```

```
summary(solo_data1$solo_LongestKill)
summary(duo_data1$duo_LongestKill)
summary(squad_data1$squad_LongestKill)

#lets visualize time survived in all 3 modes


hist(solo_data1$solo_TimeSurvivedPg, xlab = 'Time Survived',
    col = "lightblue", main = "Solo Time Survived")

hist(duo_data1$duo_TimeSurvivedPg, xlab = 'Time survived',
    col = "lightgreen", main = "Duo Time survived")

hist(squad_data1$squad_TimeSurvivedPg, xlab = 'Time Survived',
    col = "orange", main = "Squad Time Survived")

summary(solo_data1$solo_TimeSurvivedPg)
summary(duo_data1$duo_TimeSurvivedPg)
summary(squad_data1$squad_TimeSurvivedPg)


#shortening dataset by including only important data from each model

solo_data2<- solo_data1[,c("solo_WinRatio" , "solo_KillDeathRatio" , "solo_TimeSurvivedPg" ,
"solo_RoundsPlayed" , "solo_DamagePg" ,
                    "solo_HeadshotKillsPg", "solo_KillsPg" , "solo_MoveDistancePg" , "solo_HealsPg" )]
```

```
duo_data2<- duo_data1[,c("duo_WinRatio" , "duo_KillDeathRatio" , "duo_TimeSurvivedPg" ,
"duo_RoundsPlayed" , "duo_DamagePg" ,
                         "duo_HeadshotKillsPg", "duo_KillsPg" , "duo_MoveDistancePg" , "duo_HealsPg" )]


squad_data2<- squad_data1[,c("squad_WinRatio" , "squad_KillDeathRatio" , "squad_TimeSurvivedPg" ,
"squad_RoundsPlayed" , "squad_DamagePg" ,
                           "squad_HeadshotKillsPg", "squad_KillsPg" , "squad_MoveDistancePg" , "squad_HealsPg"
)]



dim(solo_data2)

dim(duo_data2)

dim(squad_data2)


#lets see how closely are these parameters related to each other


solo_data_relation <- cor(solo_data2)

duo_data_relation <- cor(duo_data2)

squad_data_relation <- cor(squad_data2)


corrplot(solo_data_relation, method = "number")

corrplot(duo_data_relation, method = "number")

corrplot(squad_data_relation, method = "number")


pubg_statistics <- na.omit(pubg_statistics)



#_____________________________________________

#         Training and Test Dataset

#_____________________________________________
```

```
get.train <- function (data.sz, train.sz)
{
  set.seed(123)
  # Take subsets of data for training/test samples
  # Return the indices
  train.ind <- sample(data.sz, train.sz)
  test.ind <- (data.sz) %w/o% train.ind
  list(train=train.ind, test=test.ind)
}

pubg_statistics_train <- subset(pubg_statistics, pubg_statistics$tracker_id<=164880)
pubg_statistics_test <- subset(pubg_statistics, pubg_statistics$tracker_id>164880)
dim(pubg_statistics_test)
dim(pubg_statistics_train)

#let's see in which mode a particular person performs well

summary(pubg_statistics$tracker_id)

pubg_statistics$comparison[pubg_statistics$solo_KillDeathRatio>pubg_statistics$duo_KillDeathRatio &
pubg_statistics$solo_KillDeathRatio > pubg_statistics$squad_KillDeathRatio] <- '1'
pubg_statistics$comparison[pubg_statistics$duo_KillDeathRatio>pubg_statistics$solo_KillDeathRatio &
pubg_statistics$duo_KillDeathRatio > pubg_statistics$squad_KillDeathRatio] <- '2'
pubg_statistics$comparison[pubg_statistics$squad_KillDeathRatio>pubg_statistics$duo_KillDeathRatio
& pubg_statistics$squad_KillDeathRatio > pubg_statistics$solo_KillDeathRatio] <- '3'
pubg_statistics$comparison<-as.numeric(as.character(pubg_statistics$comparison))
str(pubg_statistics$comparison)
is.numeric(pubg_statistics$comparison)
```

```
pubg_statistics_train$comparison[pubg_statistics_train$solo_KillDeathRatio>pubg_statistics_train$duo
_KillDeathRatio & pubg_statistics_train$solo_KillDeathRatio >
pubg_statistics_train$squad_KillDeathRatio] <- '1'

pubg_statistics_train$comparison[pubg_statistics_train$duo_KillDeathRatio>pubg_statistics_train$solo
_KillDeathRatio & pubg_statistics_train$duo_KillDeathRatio >
pubg_statistics_train$squad_KillDeathRatio] <- '2'

pubg_statistics_train$comparison[pubg_statistics_train$squad_KillDeathRatio>pubg_statistics_train$du
o_KillDeathRatio & pubg_statistics_train$squad_KillDeathRatio >
pubg_statistics_train$solo_KillDeathRatio] <- '3'

pubg_statistics_train$comparison<-as.numeric(as.character(pubg_statistics_train$comparison))

str(pubg_statistics_train$comparison)

is.numeric(pubg_statistics_train$comparison)


pubg_statistics_test$comparison[pubg_statistics_test$solo_KillDeathRatio>pubg_statistics_test$duo_Kil
lDeathRatio & pubg_statistics_test$solo_KillDeathRatio > pubg_statistics_test$squad_KillDeathRatio] <-
'1'

pubg_statistics_test$comparison[pubg_statistics_test$duo_KillDeathRatio>pubg_statistics_test$solo_Kil
lDeathRatio & pubg_statistics_test$duo_KillDeathRatio > pubg_statistics_test$squad_KillDeathRatio] <-
'2'

pubg_statistics_test$comparison[pubg_statistics_test$squad_KillDeathRatio>pubg_statistics_test$duo_
KillDeathRatio & pubg_statistics_test$squad_KillDeathRatio > pubg_statistics_test$solo_KillDeathRatio]
<- '3'

pubg_statistics_test$comparison<-as.numeric(as.character(pubg_statistics_test$comparison))

str(pubg_statistics_test$comparison)

is.numeric(pubg_statistics_test$comparison)


head(pubg_statistics$comparison)

pubg_statistics$comparison

head(pubg_statistics_train$comparison)

pubg_statistics_train$comparison

head(pubg_statistics_test$comparison)

pubg_statistics_test$comparison
```

```
pubg_statistics <- na.omit(pubg_statistics)
summary(pubg_statistics)
pubg_statistics$comparison

pubg_statistics_train <- na.omit(pubg_statistics_train)
pubg_statistics_train$comparison

pubg_statistics_test <- na.omit(pubg_statistics_test)
pubg_statistics_test$comparison

out = c (nrow(pubg_statistics_train), nrow(pubg_statistics_test))
x.names=c("Training","Test")
barplot(out, main="Data Spliting",xaxt="n")
axis(1,at = 1:2,labels=x.names)

#_____________________________________________
#         Standardise Dataset
#_____________________________________________

f.data.std <- function(data) {
  data <- as.matrix(data)
  bar <- apply(data, 2, mean)
  s <- apply(data, 2, sd)
  t((t(data) - bar)/s)
}
pubg_statistics.std <- f.data.std(pubg_statistics[-1])
head(pubg_statistics.std)
summary(pubg_statistics.std)
```

```
pubg_statistics_train.std <- f.data.std(pubg_statistics_train[-1])
head(pubg_statistics_train.std)
summary(pubg_statistics_train.std)



pubg_statistics_test.std <- f.data.std(pubg_statistics_test[-1])
head(pubg_statistics_test.std)
summary(pubg_statistics_test.std)



pubg_statistics.std[is.nan(pubg_statistics.std)] <- 0
pubg_statistics_train.std[is.nan(pubg_statistics_train.std)] <- 0
pubg_statistics_test.std[is.nan(pubg_statistics_test.std)] <- 0

#_____________________________________________
#       Dimension Reduction
#_____________________________________________

#_____________________________________________
#       PCA
#_____________________________________________

#Using prcomp to get principal component vectors

library(stats)
pc.pubg_statistics <- prcomp(pubg_statistics.std)
```

```
summary(pc.pubg_statistics)
plot(pc.pubg_statistics, main="Scree Plot: PCA for Raw dataset", col=heat.colors(15))
# First  Principal Components
pc1.pubg_statistics <- data.frame(pc.pubg_statistics$x[,1:3])
head(pc1.pubg_statistics)

pc.pubg_statistics_train <- prcomp(pubg_statistics_train.std)
summary(pc.pubg_statistics_train)
plot(pc.pubg_statistics, main="Scree Plot: PCA for Training dataset", col=heat.colors(15))
# First  Principal Components
pc1.pubg_statistics_train <- data.frame(pc.pubg_statistics_train$x[,1:3])
head(pc1.pubg_statistics_train)

pc.pubg_statistics_test <- prcomp(pubg_statistics_test.std[,])
summary(pc.pubg_statistics_test)
plot(pc.pubg_statistics_test, main="Scree Plot: PCA for Test dataset", col=heat.colors(15))
# First  Principal Components
pc1.pubg_statistics_test <- data.frame(pc.pubg_statistics_test$x[,1:3])
head(pc1.pubg_statistics_test)
#_____________________________________________
#        Whitened Dataset Center and Sphere
#_____________________________________________

Sphere.Data <- function(data) {
  data <- as.matrix(data)
  data <- t(t(data) - apply(data, 2, mean))
  data.svd <- svd(var(data))
  sphere.mat <- t(data.svd$v %*% (t(data.svd$u) * (1/sqrt(data.svd$d))))
  return(data %*% sphere.mat)
```

```
}

pubg_statistics.white <- Sphere.Data(pubg_statistics[-1])
colnames(pubg_statistics.white) <- colnames(pubg_statistics.std)
apply(pubg_statistics.white, 2, mean)
apply(pubg_statistics.white, 2, sd)
head(pubg_statistics.white)

pubg_statistics_train.white <- Sphere.Data(pubg_statistics_train[-1])
colnames(pubg_statistics_train.white) <- colnames(pubg_statistics_train.std)
apply(pubg_statistics_train.white, 2, mean)
apply(pubg_statistics_train.white, 2, sd)
head(pubg_statistics_train.white)

pubg_statistics_test.white <- Sphere.Data(pubg_statistics_test[-1])
colnames(pubg_statistics_test.white) <- colnames(pubg_statistics_test.std)
apply(pubg_statistics_test.white, 2, mean)
apply(pubg_statistics_test.white, 2, sd)
head(pubg_statistics_test.white)

#_____________________________________________
#       Dimension Reduction
#_____________________________________________

#_____________________________________________
#       ICA
#_____________________________________________
library(fastICA)
```

```
pubg_statistics.white.ica <- fastICA(pubg_statistics.white[,-1], 3, alg.typ = "parallel", fun = "logcosh",
alpha = 1,method = "R",
                                     row.norm = FALSE, maxit = 200, tol = 0.0001, verbose = TRUE)

pubg_statistics.ica<-pubg_statistics.white.ica$S

head(pubg_statistics.ica)

pubg_statistics_train.white.ica <- fastICA(pubg_statistics_train.white[,-1], 3, alg.typ = "parallel", fun =
"logcosh", alpha = 1,method = "R",
                                     row.norm = FALSE, maxit = 200, tol = 0.0001, verbose = TRUE)

pubg_statistics_train.ica<-pubg_statistics_train.white.ica$S

head(pubg_statistics_train.ica)

pubg_statistics_test.white.ica <- fastICA(pubg_statistics_test.white[,-1], 3, alg.typ = "parallel", fun =
"logcosh", alpha = 1,method = "R",
                                     row.norm = FALSE, maxit = 200, tol = 0.0001, verbose = TRUE)

pubg_statistics_test.ica<-pubg_statistics_test.white.ica$S

head(pubg_statistics_test.ica)




#_____________________________________________

#       Data Reduction

#_____________________________________________


#*****************************************

#       Davies Bouldin Function

#*****************************************
```

```
Davies.Bouldin <- function(A, SS, m) {
 # A  - the centres of the clusters
 # SS - the within sum of squares
 # m  - the sizes of the clusters
 N <- nrow(A)   # number of clusters
 # intercluster distance
 S <- sqrt(SS/m)
 # Get the distances between centres
 M <- as.matrix(dist(A))
 # Get the ratio of intercluster/centre.dist
 R <- matrix(0, N, N)
 for (i in 1:(N-1)) {
  for (j in (i+1):N) {
   R[i,j] <- (S[i] + S[j])/M[i,j]
   R[j,i] <- R[i,j]
  }
 }
 return(mean(apply(R, 1, max)))
}

#****************************************
#error_cal Function to calculate wrong classifications in cluster
#****************************************

error_cal <- function(tbl,cluster_size)
{
 wrong_data <- 0
 for(clust in 1:cluster_size)
```

```
  {
    wrong_data <- wrong_data + (sum(tbl[,clust])-max(tbl[,clust]))
  }
  return(wrong_data)
}

#Function for Euclidean Clustering
clustering_euclidean <- function(data_set,data_set.orig, limit)
{

  oldpar <- par(mfrow = c(4,4))
  par(mar=c(2,1,2,1))

  errs <- rep(0, 7)
  DBI <- rep(0, 7)
  library(cluster)
  library(fpc)
  library(flexclust)
  library(stats)
  for (i in limit)
  {
    min_error <- 250
    min_error_km <- 0
    best.seed <- 0
    #Loop for Seed
    for (j in 2:10)
    {

      set.seed(j)
```

```
  #Clustering Using K means
  KM <- kmeans((data_set[,]), i, 25)
  ct.km <- table(KM$cluster, KM$cluster)

  #Calculating toal wrong data for each seed
  error <- error_cal(ct.km,i)
  if(min_error > error)
  {
   #Storing Error count and Kmeans output and best seed for min error
   min_error <- error
   min_error_km <-KM
   best.seed <- j
  }

}
print(paste("Best Seed for the Cluster Size " , i ,"is " , best.seed))
print(paste("Total Wrong Classification in Cluster Size " , i ,"is " , min_error))
print(paste("Centroids for the Cluster Size " , i ,"are :"))
print(min_error_km$centers)
print(min_error_km)

#Plotting the CLuster
plotcluster(data_set, col=min_error_km$cluster,min_error_km$cluster, main=paste(i,"Clusters"))

if(length(limit) > 1)
{
  #CLuster Analysis
  errs[i-1] <- sum(min_error_km$withinss)
  DBI[i-1] <- Davies.Bouldin(min_error_km$centers, min_error_km$withinss, min_error_km$size)
```

```
  }

 }
 if(length(limit) > 1)
 {
  plot(2:10, errs, main = "SSE")
  lines(2:10, errs)
  #
  plot(2:10, DBI, main = "Davies-Bouldin")
  lines(2:10, DBI)
  #
 }
 else
 {
  return(min_error_km)
 }
 return(errs)
}

cluster_range <- 2:10

library (flexclust)

#*****************************************
#        Clustering: Raw DataSet pubg_statistics
#*****************************************
```

```
clust.pubg_statistics <-
clustering_euclidean(pubg_statistics[,c(2:4,6,14:18,22,53:54,56,64:68,72,103:104,106,114:118,122,153)
], pubg_statistics,cluster_range)


#Optimal Cluster Size is 2


clus.pubg_statistics<-
clustering_euclidean(pubg_statistics[,c(2:4,6,14:18,22,53:54,56,64:68,72,103:104,106,114:118,122,153)
], pubg_statistics, 2)


group1<-pubg_statistics[clus.pubg_statistics$cluster == 1,]$comparison

group2<-pubg_statistics[clus.pubg_statistics$cluster == 2,]$comparison


#which is highest performance mode group

highest.perf.mode.group7 <- which.max(c(mean(group1),mean(group2)))

#which is lowest performance mode group

lowest.perf.mode.group7 <- which.min(c(mean(group2),mean(group2)))



highest.perf.mode.group.raw <- pubg_statistics[clus.pubg_statistics$cluster ==
highest.perf.mode.group7, c(1,153)]

lowest.perf.mode.group.raw <- pubg_statistics[clus.pubg_statistics$cluster ==
lowest.perf.mode.group7, c(1,153)]


print(clus.pubg_statistics$size)


print(paste("Raw: Highest performance mode Group is g", highest.perf.mode.group7, sep=""))


print(paste("Raw: Lowest performance mode Group is g", lowest.perf.mode.group7, sep=""))
```

```
head(highest.perf.mode.group.raw[order(highest.perf.mode.group.raw$comparison, decreasing=TRUE),
])


tail(lowest.perf.mode.group.raw[order(lowest.perf.mode.group.raw$comparison, decreasing=TRUE), ])


#*****************************************
#       Clustering: Standard DataSet pubg_statistics.std
#*****************************************
clust.pubg_statistics.std <- clustering_euclidean(pubg_statistics.std, pubg_statistics,cluster_range)


#Optimal Cluster Size is 8


clus.pubg_statistics.std <- clustering_euclidean(pubg_statistics.std, pubg_statistics, 8)


group1.std<-pubg_statistics[clus.pubg_statistics.std$cluster == 1,]$comparison
group2.std<-pubg_statistics[clus.pubg_statistics.std$cluster == 2,]$comparison
group3.std<-pubg_statistics[clus.pubg_statistics.std$cluster == 3,]$comparison
group4.std<-pubg_statistics[clus.pubg_statistics.std$cluster == 4,]$comparison
group5.std<-pubg_statistics[clus.pubg_statistics.std$cluster == 5,]$comparison
group6.std<-pubg_statistics[clus.pubg_statistics.std$cluster == 6,]$comparison
group7.std<-pubg_statistics[clus.pubg_statistics.std$cluster == 7,]$comparison
group8.std<-pubg_statistics[clus.pubg_statistics.std$cluster == 8,]$comparison


#which is highest performance mode group
highest.perf.mode.group <- which.max(c(mean(group1.std),mean(group2.std),
mean(group3.std),mean(group4.std),mean(group5.std),mean(group6.std), mean(group7.std),
mean(group8.std)))
#which is lowest performance mode group
```

```
lowest.perf.mode.group <- which.min(c(mean(group1.std),mean(group2.std),
mean(group3.std),mean(group4.std),mean(group5.std),mean(group6.std), mean(group7.std),
mean(group8.std)))



highest.perf.mode.group.std <- pubg_statistics[clus.pubg_statistics.std$cluster ==
highest.perf.mode.group, c(1,153)]

lowest.perf.mode.group.std <- pubg_statistics[clus.pubg_statistics.std$cluster ==
lowest.perf.mode.group, c(1,153)]


print(clus.pubg_statistics.std$size)


print(paste("Standard: Highest performance mode Group is g", highest.perf.mode.group, sep=""))


print(paste("Standard: Lowest performance mode Group is g", lowest.perf.mode.group, sep=""))


head(highest.perf.mode.group.std[order(highest.perf.mode.group.std$comparison, decreasing=TRUE),
])


tail(lowest.perf.mode.group.std[order(lowest.perf.mode.group.std$comparison, decreasing=TRUE), ])


#*****************************************
#       Clustering: Whitened DataSet pubg_statistics.white
#*****************************************
cluster_range <- 2:10


library (flexclust)


#Clustering on Whitened Raw DataSet pubg_statistics

clust.pubg_statistics.white <- clustering_euclidean(pubg_statistics.white, pubg_statistics,cluster_range)
```

```
#Optimal Cluster Size is 10

clus.pubg_statistics.white <- clustering_euclidean(pubg_statistics.white, pubg_statistics,10)


group1.white<-pubg_statistics[clus.pubg_statistics.white$cluster == 1,]$comparison

group2.white<-pubg_statistics[clus.pubg_statistics.white$cluster == 2,]$comparison

group3.white<-pubg_statistics[clus.pubg_statistics.white$cluster == 3,]$comparison

group4.white<-pubg_statistics[clus.pubg_statistics.white$cluster == 4,]$comparison

group5.white<-pubg_statistics[clus.pubg_statistics.white$cluster == 5,]$comparison

group6.white<-pubg_statistics[clus.pubg_statistics.white$cluster == 6,]$comparison

group7.white<-pubg_statistics[clus.pubg_statistics.white$cluster == 7,]$comparison

group8.white<-pubg_statistics[clus.pubg_statistics.white$cluster == 8,]$comparison

group9.white<-pubg_statistics[clus.pubg_statistics.white$cluster == 9,]$comparison

group10.white<-pubg_statistics[clus.pubg_statistics.white$cluster == 10,]$comparison


#which is highest performance mode group

highest.perf.mode.group <- which.max(c(mean(group1.white),mean(group2.white),
mean(group3.white),mean(group4.white),mean(group5.white),mean(group6.white),
mean(group7.white), mean(group8.white),mean(group9.white),mean(group10.white)))

#which is lowest performance mode group

lowest.perf.mode.group <- which.min(c(mean(group1.white),mean(group2.white),
mean(group3.white),mean(group4.white),mean(group5.white),mean(group6.white),
mean(group7.white), mean(group8.white),mean(group9.white),mean(group10.white)))



highest.perf.mode.group.white <- pubg_statistics[clus.pubg_statistics.white$cluster ==
highest.perf.mode.group, c(1,153)]

lowest.perf.mode.group.white <- pubg_statistics[clus.pubg_statistics.white$cluster ==
lowest.perf.mode.group, c(1,153)]


print(clus.pubg_statistics.white$size)
```

```
print(paste("Whitened: Highest performance mode Group is g", highest.perf.mode.group, sep=""))

print(paste("Whitened: Lowest performance mode Group is g", lowest.perf.mode.group, sep=""))

head(highest.perf.mode.group.white[order(highest.perf.mode.group.white$comparison,
decreasing=TRUE), ])

tail(lowest.perf.mode.group.white[order(lowest.perf.mode.group.white$comparison,
decreasing=TRUE), ])



#*****************************************
#         Data Reduction on Standardized dataset
#*****************************************
## Function Set the indices for the  sets
get.subset <- function (data, size)
{
  set.seed(123)
  data_subset <- sample(data, size)
}

pubg_statistics.std.new <- pubg_statistics
pubg_statistics.std.new$cluster <- clus.pubg_statistics.std$cluster

median1<-median(pubg_statistics.std.new$comparison[pubg_statistics.std.new$cluster == 1])
median2<-median(pubg_statistics.std.new$comparison[pubg_statistics.std.new$cluster == 2])
median3<-median(pubg_statistics.std.new$comparison[pubg_statistics.std.new$cluster == 3])
median4<-median(pubg_statistics.std.new$comparison[pubg_statistics.std.new$cluster == 4])
```

```
median5<-median(pubg_statistics.std.new$comparison[pubg_statistics.std.new$cluster == 5])

median6<-median(pubg_statistics.std.new$comparison[pubg_statistics.std.new$cluster == 6])

median7<-median(pubg_statistics.std.new$comparison[pubg_statistics.std.new$cluster == 7])

median8<-median(pubg_statistics.std.new$comparison[pubg_statistics.std.new$cluster == 8])

print(paste("Median Values-> Cluster1 = ", median1,", Cluster2 = ",median2,", Cluster3 =
",median3,",Cluster4 = ",median4,",Cluster5 = ",median3,",Cluster6 = ",median6,",Cluster7 =
",median7,",Cluster8 = ",median8, sep=""))


pub1.std.ind <- which(pubg_statistics.std.new$cluster == 1)

pub2.std.ind <- which(pubg_statistics.std.new$cluster == 2)

pub3.std.ind <- which(pubg_statistics.std.new$cluster == 3)

pub4.std.ind <- which(pubg_statistics.std.new$cluster == 4)

pub5.std.ind <- which(pubg_statistics.std.new$cluster == 5)

pub6.std.ind <- which(pubg_statistics.std.new$cluster == 6)

pub7.std.ind <- which(pubg_statistics.std.new$cluster == 7)

pub8.std.ind <- which(pubg_statistics.std.new$cluster == 8)


pub1.std.size <- round((2*length(pub1.std.ind))/3)

pub2.std.size <- round((2*length(pub2.std.ind))/3)

pub3.std.size <- round((2*length(pub3.std.ind))/3)

pub4.std.size <- round((2*length(pub4.std.ind))/3)

pub5.std.size <- round((2*length(pub5.std.ind))/3)

pub6.std.size <- round((2*length(pub6.std.ind))/3)

pub7.std.size <- round((2*length(pub7.std.ind))/3)

pub8.std.size <- round((2*length(pub8.std.ind))/3)



pub1.std <- get.subset(pub1.std.ind,pub1.std.size)

pub2.std <- get.subset(pub2.std.ind,pub2.std.size)
```

```
pub3.std <- get.subset(pub3.std.ind,pub3.std.size)

pub4.std <- get.subset(pub4.std.ind,pub4.std.size)

pub5.std <- get.subset(pub5.std.ind,pub5.std.size)

pub6.std <- get.subset(pub6.std.ind,pub6.std.size)

pub7.std <- get.subset(pub7.std.ind,pub7.std.size)

pub8.std <- get.subset(pub8.std.ind,pub8.std.size)



pubg_statistics.std_reduce_ind <-
c(pub1.std,pub2.std,pub3.std,pub4.std,pub5.std,pub6.std,pub7.std,pub8.std)


pubg_statistics.std_reduce <- pubg_statistics[pubg_statistics.std_reduce_ind,]


head(pubg_statistics.std_reduce)


#Dividing the reduced standardised dataset into training and test on the basis of tracker_id and its
proper 1/3 test and 2/3 train

nrow(pubg_statistics.std_reduce[pubg_statistics.std_reduce$tracker_id>164880,])

nrow(pubg_statistics.std_reduce[pubg_statistics.std_reduce$tracker_id<=164880,])



#Train Dataset with tracker_id<=164880 and total rows are 43409 (earlier it was 65081)

pubg_statistics.std_reduce_train <-
pubg_statistics.std_reduce[pubg_statistics.std_reduce$tracker_id<=164880,]


#Train Dataset with tracker_id<=164880 and total rows are 14436 (earlier it was 21687)

pubg_statistics.std_reduce_test <-
pubg_statistics.std_reduce[pubg_statistics.std_reduce$tracker_id>164880,]
```

```
out.std_reduce = c(nrow(pubg_statistics.std_reduce),nrow(pubg_statistics.std_reduce_train),
nrow(pubg_statistics.std_reduce_test))

x.names=c("Complete","Training","Test")

barplot(out.std_reduce,main="Reduced Standardized Data Spliting",xaxt="n",width=c(1,1,1))

axis(1,at = 1:3,labels=x.names)



#_____________________________________________

#        Unsupervised Learning

#_____________________________________________


#*****************************************

#        Clustering: Training DataSet pubg_statistics_train

#*****************************************

cluster_range <- 2:10

pubg_statistics_train[0,]


#Clustering on Raw Train DataSet pubg_statistics_train

clust.pubg_statistics_train <-
clustering_euclidean(pubg_statistics_train[,c(2:4,6,14:18,22,53:54,56,64:68,72,103:104,106,114:118,12
2,153)],pubg_statistics_train,cluster_range)


#Optimal Cluster Size is 2

clus.pubg_statistics_train <-
clustering_euclidean(pubg_statistics_train[,c(2:4,6,14:18,22,53:54,56,64:68,72,103:104,106,114:118,12
2,153)], clust.pubg_statistics_train,2)


group1_train<-pubg_statistics[clus.pubg_statistics_train$cluster == 1,]$comparison

group2_train<-pubg_statistics[clus.pubg_statistics_train$cluster == 2,]$comparison
```

```
#which is highest performance mode group

highest.perf.mode.group1 <- which.max(c(mean(group1_train),mean(group2_train)))

#which is lowest performance mode group

lowest.perf.mode.group1 <- which.min(c(mean(group1_train),mean(group2_train)))



highest.perf.mode.group_train <- pubg_statistics[clus.pubg_statistics_train$cluster ==
highest.perf.mode.group1, c(1,153)]

lowest.perf.mode.group_train <- pubg_statistics[clus.pubg_statistics_train$cluster ==
lowest.perf.mode.group1, c(1,153)]


print(clus.pubg_statistics_train$size)


print(paste("Raw Training: Highest performance mode Group is g", highest.perf.mode.group1, sep=""))


print(paste("Raw Training: Lowest performance mode Group is g", lowest.perf.mode.group1, sep=""))


head(highest.perf.mode.group_train[order(highest.perf.mode.group_train$comparison,
decreasing=TRUE), ])


tail(lowest.perf.mode.group_train[order(lowest.perf.mode.group_train$comparison, decreasing=TRUE),
])

#*****************************************

#       Clustering: Test DataSet pubg_statistics_test

#*****************************************

cluster_range <- 2:10


#Clustering on Raw test DataSet pubg_statistics_test

clust.pubg_statistics_test <-
clustering_euclidean(pubg_statistics_test[,c(2:4,6,14:18,22,53:54,56,64:68,72,103:104,106,114:118,122,
153)],pubg_statistics_test,cluster_range)
```

```
#Optimal Cluster Size is 2

clus.pubg_statistics_test <-
clustering_euclidean(pubg_statistics_test[,c(2:4,6,14:18,22,53:54,56,64:68,72,103:104,106,114:118,122,
153)], clust.pubg_statistics_test,2)


group1_test<-pubg_statistics[clus.pubg_statistics_test$cluster == 1,]$comparison

group2_test<-pubg_statistics[clus.pubg_statistics_test$cluster == 2,]$comparison


#which is highest performance mode group

highest.perf.mode.group2 <- which.max(c(mean(group1_test),mean(group2_test)))

#which is lowest performance mode group

lowest.perf.mode.group2 <- which.min(c(mean(group1_test),mean(group2_test)))



highest.perf.mode.group_test <- pubg_statistics[clus.pubg_statistics_test$cluster ==
highest.perf.mode.group2, c(1,153)]

lowest.perf.mode.group_test <- pubg_statistics[clus.pubg_statistics_test$cluster ==
lowest.perf.mode.group2, c(1,153)]


print(clus.pubg_statistics_test$size)

print(paste("Raw Test: Highest performance mode Group is g", highest.perf.mode.group2, sep=""))


print(paste("Raw Test: Lowest performance mode Group is g", lowest.perf.mode.group2, sep=""))


head(highest.perf.mode.group_test[order(highest.perf.mode.group_test$comparison,
decreasing=TRUE), ])


tail(lowest.perf.mode.group_test[order(lowest.perf.mode.group_test$comparison, decreasing=TRUE), ])
```

```
#*****************************************
#        Clustering: PCA DataSet pc.pubg_statistics
#*****************************************
cluster_range <- 2:10
clust.pc.pubg_statistics <- clustering_euclidean(pc1.pubg_statistics,pubg_statistics,cluster_range)

#Optimal Cluster Size is 5
clus.pc.pubg_statistics <- clustering_euclidean(pc1.pubg_statistics, pubg_statistics,5)

group1.pc<-pubg_statistics[clus.pc.pubg_statistics$cluster == 1,]$comparison
group2.pc<-pubg_statistics[clus.pc.pubg_statistics$cluster == 2,]$comparison
group3.pc<-pubg_statistics[clus.pc.pubg_statistics$cluster == 3,]$comparison
group4.pc<-pubg_statistics[clus.pc.pubg_statistics$cluster == 4,]$comparison
group5.pc<-pubg_statistics[clus.pc.pubg_statistics$cluster == 5,]$comparison
#which is highest performance mode group
highest.perf.mode.group3 <- which.max(c(mean(group1.pc),mean(group2.pc),
mean(group3.pc),mean(group4.pc),mean(group5.pc)))
#which is lowest performance mode group
lowest.perf.mode.group3 <- which.min(c(mean(group1.pc),mean(group2.pc),
mean(group3.pc),mean(group4.pc),mean(group5.pc)))

highest.perf.mode.group.pc <- pubg_statistics[clus.pc.pubg_statistics$cluster ==
highest.perf.mode.group3, c(1,153)]
lowest.perf.mode.group.pc <- pubg_statistics[clus.pc.pubg_statistics$cluster ==
lowest.perf.mode.group3, c(1,153)]

print(clus.pc.pubg_statistics$size)

print(paste("PCA on raw dataset: Highest performance mode Group is g", highest.perf.mode.group3,
sep=""))
```

```
print(paste("PCA on raw dataset: Lowest performance mode Group is g", lowest.perf.mode.group3, sep=""))


head(highest.perf.mode.group.pc[order(highest.perf.mode.group.pc$comparison, decreasing=TRUE), ])


tail(lowest.perf.mode.group.pc[order(lowest.perf.mode.group.pc$comparison, decreasing=TRUE), ])


#*****************************************

#       Clustering: Train DataSet After PCA pc1.pubg_statistics_train

#*****************************************

cluster_range <- 2:10

clust.pc.pubg_statistics_train <- clustering_euclidean(pc1.pubg_statistics_train,pubg_statistics_train,cluster_range)


#Optimal Cluster Size is 7

clus.pc.pubg_statistics_train <- clustering_euclidean(pc1.pubg_statistics_train, pubg_statistics_train,7)


group1.train.pc<-pubg_statistics_train[clus.pc.pubg_statistics_train$cluster == 1,]$comparison

group2.train.pc<-pubg_statistics_train[clus.pc.pubg_statistics_train$cluster == 2,]$comparison

group3.train.pc<-pubg_statistics_train[clus.pc.pubg_statistics_train$cluster == 3,]$comparison

group4.train.pc<-pubg_statistics_train[clus.pc.pubg_statistics_train$cluster == 4,]$comparison

group5.train.pc<-pubg_statistics_train[clus.pc.pubg_statistics_train$cluster == 5,]$comparison

group6.train.pc<-pubg_statistics_train[clus.pc.pubg_statistics_train$cluster == 6,]$comparison

group7.train.pc<-pubg_statistics_train[clus.pc.pubg_statistics_train$cluster == 7,]$comparison

#which is highest performance mode group

highest.perf.mode.group4 <- which.max(c(mean(group1.train.pc),mean(group2.train.pc), mean(group3.train.pc),mean(group4.train.pc),mean(group5.train.pc),mean(group6.train.pc),mean(group7.train.pc)))

#which is lowest performance mode group
```

```
lowest.perf.mode.group4 <- which.min(c(mean(group1.train.pc),mean(group2.train.pc),
mean(group3.train.pc),mean(group4.train.pc),mean(group5.train.pc),mean(group6.train.pc),mean(grou
p7.train.pc)))


highest.perf.mode.group.pc1 <- pubg_statistics_train[clus.pc.pubg_statistics_train$cluster ==
highest.perf.mode.group4, c(1,153)]

lowest.perf.mode.group.pc1 <- pubg_statistics_train[clus.pc.pubg_statistics_train$cluster ==
lowest.perf.mode.group4, c(1,153)]


print(clus.pc.pubg_statistics_train$size)


print(paste("Train DataSet After PCA: Highest performance mode Group is g",
highest.perf.mode.group4, sep=""))


print(paste("Train DataSet After PCA: Lowest performance mode Group is g", lowest.perf.mode.group4,
sep=""))


head(highest.perf.mode.group.pc1[order(highest.perf.mode.group.pc1$comparison, decreasing=TRUE),
])


tail(lowest.perf.mode.group.pc1[order(lowest.perf.mode.group.pc1$comparison, decreasing=TRUE), ])


#****************************************

#       Clustering: Test DataSet After PCA pc1.pubg_statistics_test

#****************************************

cluster_range <- 2:10

clust.pc.pubg_statistics_test <-
clustering_euclidean(pc1.pubg_statistics_test,pubg_statistics_test,cluster_range)


#Optimal Cluster Size is 5

clus.pc.pubg_statistics_test <- clustering_euclidean(pc1.pubg_statistics_test, pubg_statistics_test,5)
```

```
group1.test.pc<-pubg_statistics_test[clus.pc.pubg_statistics_test$cluster == 1,]$comparison

group2.test.pc<-pubg_statistics_test[clus.pc.pubg_statistics_test$cluster == 2,]$comparison

group3.test.pc<-pubg_statistics_test[clus.pc.pubg_statistics_test$cluster == 3,]$comparison

group4.test.pc<-pubg_statistics_test[clus.pc.pubg_statistics_test$cluster == 4,]$comparison

group5.test.pc<-pubg_statistics_test[clus.pc.pubg_statistics_test$cluster == 5,]$comparison


#which is highest performance mode group

highest.perf.mode.group5 <- which.max(c(mean(group1.test.pc),mean(group2.test.pc),
mean(group3.test.pc),mean(group4.test.pc),mean(group5.test.pc)))

#which is lowest performance mode group

lowest.perf.mode.group5 <- which.min(c(mean(group1.test.pc),mean(group2.test.pc),
mean(group3.test.pc),mean(group4.test.pc),mean(group5.test.pc)))


highest.perf.mode.group.pc2 <- pubg_statistics_test[clus.pc.pubg_statistics_test$cluster ==
highest.perf.mode.group5, c(1,153)]

lowest.perf.mode.group.pc2 <- pubg_statistics_test[clus.pc.pubg_statistics_test$cluster ==
lowest.perf.mode.group5, c(1,153)]


print(clus.pc.pubg_statistics_test$size)

print(paste("Test dataset after PCA: Highest performance mode Group is g", highest.perf.mode.group5,
sep=""))

print(paste("Test dataset after PCA: Lowest performance mode Group is g", lowest.perf.mode.group5,
sep=""))


head(highest.perf.mode.group.pc2[order(highest.perf.mode.group.pc2$comparison, decreasing=TRUE),
])

tail(lowest.perf.mode.group.pc2[order(lowest.perf.mode.group.pc2$comparison, decreasing=TRUE), ])


#*****************************************

#       Clustering: ICA DataSet pubg_statistics.ica

#*****************************************
```

```
cluster_range <- 2:10

clust.pubg_statistics.ica <- clustering_euclidean(pubg_statistics.ica,pubg_statistics,cluster_range)


#Optimal Cluster Size is 6

clus.pubg_statistics.ica <- clustering_euclidean(pubg_statistics.ica, pubg_statistics,6)


group1.ica<-pubg_statistics[clus.pubg_statistics.ica$cluster == 1,]$comparison

group2.ica<-pubg_statistics[clus.pubg_statistics.ica$cluster == 2,]$comparison

group3.ica<-pubg_statistics[clus.pubg_statistics.ica$cluster == 3,]$comparison

group4.ica<-pubg_statistics[clus.pubg_statistics.ica$cluster == 4,]$comparison

group5.ica<-pubg_statistics[clus.pubg_statistics.ica$cluster == 5,]$comparison

group6.ica<-pubg_statistics[clus.pubg_statistics.ica$cluster == 6,]$comparison

#which is highest performance mode group

highest.perf.mode.group8 <- which.max(c(mean(group1.ica),mean(group2.ica),
mean(group3.ica),mean(group4.ica),mean(group5.ica),mean(group6.ica)))

#which is lowest performance mode group

lowest.perf.mode.group8 <- which.min(c(mean(group1.ica),mean(group2.ica),
mean(group3.ica),mean(group4.ica),mean(group5.ica),mean(group6.ica)))


highest.perf.mode.group.ica <- pubg_statistics[clus.pubg_statistics.ica$cluster ==
highest.perf.mode.group8, c(1,153)]

lowest.perf.mode.group.ica <- pubg_statistics[clus.pubg_statistics.ica$cluster ==
lowest.perf.mode.group8, c(1,153)]


print(clus.pubg_statistics.ica$size)


print(paste("ICA on raw dataset: Highest performance mode Group is g", highest.perf.mode.group8,
sep=""))


print(paste("ICA on raw dataset: Lowest performance mode Group is g", lowest.perf.mode.group8,
sep=""))
```

```
head(highest.perf.mode.group.ica[order(highest.perf.mode.group.ica$comparison, decreasing=TRUE), ])

tail(lowest.perf.mode.group.ica[order(lowest.perf.mode.group.ica$comparison, decreasing=TRUE), ])

#*****************************************
#       Clustering: Train DataSet After ICA pubg_statistics_train.ica
#*****************************************
cluster_range <- 2:10
clust.pubg_statistics_train.ica <-
clustering_euclidean(pubg_statistics_train.ica,pubg_statistics_train,cluster_range)

#Optimal Cluster Size is 3
clus.pubg_statistics_train.ica <- clustering_euclidean(pubg_statistics_train.ica, pubg_statistics_train,3)

group1.train.ica<-pubg_statistics_train[clus.pubg_statistics_train.ica$cluster == 1,]$comparison
group2.train.ica<-pubg_statistics_train[clus.pubg_statistics_train.ica$cluster == 2,]$comparison
group3.train.ica<-pubg_statistics_train[clus.pubg_statistics_train.ica$cluster == 3,]$comparison

#which is highest performance mode group
highest.perf.mode.group9 <- which.max(c(mean(group1.train.ica),mean(group2.train.ica),
mean(group3.train.ica)))
#which is lowest performance mode group
lowest.perf.mode.group9 <- which.min(c(mean(group1.train.ica),mean(group2.train.ica),
mean(group3.train.ica)))

highest.perf.mode.group.ica1 <- pubg_statistics_train[clus.pubg_statistics_train.ica$cluster ==
highest.perf.mode.group9, c(1,153)]
lowest.perf.mode.group.ica1 <- pubg_statistics_train[clus.pubg_statistics_train.ica$cluster ==
lowest.perf.mode.group9, c(1,153)]
```

```
print(clus.pubg_statistics_train.ica$size)


print(paste("Train DataSet After ICA: Highest performance mode Group is g", highest.perf.mode.group9, sep=""))


print(paste("Train DataSet After ICA: Lowest performance mode Group is g", lowest.perf.mode.group9, sep=""))


head(highest.perf.mode.group.ica1[order(highest.perf.mode.group.ica1$comparison, decreasing=TRUE), ])


tail(lowest.perf.mode.group.ica1[order(lowest.perf.mode.group.ica1$comparison, decreasing=TRUE), ])


#_____________________________________________

#         Supervised Learning

#_____________________________________________

pubg_statistics_test$comparison<-NA


dim(pubg_statistics_test)


pubg_solution <-subset(pubg_statistics, pubg_statistics$tracker_id>164880)


pubg_solution$comparison[pubg_solution$solo_KillDeathRatio>pubg_solution$duo_KillDeathRatio & pubg_solution$solo_KillDeathRatio > pubg_solution$squad_KillDeathRatio] <- '1'


pubg_solution$comparison[pubg_solution$duo_KillDeathRatio>pubg_solution$solo_KillDeathRatio & pubg_solution$duo_KillDeathRatio > pubg_solution$squad_KillDeathRatio] <- '2'


pubg_solution$comparison[pubg_solution$squad_KillDeathRatio>pubg_solution$duo_KillDeathRatio & pubg_solution$squad_KillDeathRatio > pubg_solution$solo_KillDeathRatio] <- '3'
```

```
head(pubg_statistics_train$comparison)
str(pubg_statistics_train$comparison)
head(pubg_solution$comparison)
pubg_solution$comparison

pubg_statistics_train <- na.omit(pubg_statistics_train)
pubg_statistics_train$comparison
which(is.na(pubg_statistics_train$comparison))

pubg_solution <- na.omit(pubg_solution)
pubg_solution$comparison
which(is.na(pubg_solution$comparison))
dim(pubg_solution)
library(dplyr)
library(readr)
library(ggthemes)
library(randomForest)
library(rpart)
library(caret)
library(rpart.plot)
library(MASS)
library(DAAG)
library(tree)
set.seed(754)

pubg_statistics_train_model<- randomForest(factor(comparison)~
solo_WinRatio+solo_KillDeathRatio+solo_TimeSurvivedPg+solo_RoundsPlayed+duo_WinRatio+
```

```
duo_KillDeathRatio+duo_TimeSurvivedPg+duo_RoundsPlayed+duo_RevivesPg+duo_Revives+duo_Team
Kills+

squad_WinRatio+squad_KillDeathRatio+squad_TimeSurvivedPg+squad_RoundsPlayed+squad_RevivesPg
+squad_Revives+squad_TeamKills, data = pubg_statistics_train)


pubg_statistics_train_model

str(pubg_statistics_train_model)


#plotting the graph for important variables

importance(pubg_statistics_train_model)

importance_model<- importance(pubg_statistics_train_model)

varImportance <- data.frame(Variables = row.names(importance_model),

                    Importance = round(importance_model[ ,'MeanDecreaseGini'],2))

rankImportance <- varImportance %>%

  mutate(Rank = paste0('#',dense_rank(desc(Importance))))

ggplot(rankImportance, aes(x = reorder(Variables, Importance),

                     y = Importance, fill = Importance)) +

  geom_bar(stat='identity') +

  geom_text(aes(x = Variables, y = 0.5, label = Rank),

          hjust=0, vjust=0.55, size = 4, colour = 'red') +

  labs(x = 'Variables') +

  coord_flip() +

  theme_few()




#providing prediction to test set

pubg_prediction <- predict(pubg_statistics_train_model, data=pubg_statistics_test)
```

```
mean(pubg_prediction==pubg_solution$comparison)


summary(pubg_statistics_test)
dim(pubg_statistics_test)
pubg_prediction <- pubg_prediction[1:21974]
pubg_statistics_test<- pubg_statistics_test
pubg_statistics_test$comparison<- pubg_prediction
pubg_statistics_test$comparison



which(is.na(pubg_statistics_test$comparison))
pubg_statistics_test$comparison

dim(pubg_solution)
pubg_solution <- pubg_solution[1:42614,]
pubg_prediction <-pubg_prediction[1:21687]
Actual.Values <- pubg_solution$comparison
Predicted.Values<-as.numeric(pubg_prediction)
table(Predicted.Values,Actual.Values)
confusion(Predicted.Values, Actual.Values)

#plotting the error rate graph
print(pubg_statistics_train_model)
plot(pubg_statistics_train_model, ylim=c(0,0.36))
legend('topright', colnames(pubg_statistics_train_model$err.rate),col = 1:3,fill = 1:3)
```

```
randomForest_plot <-
rpart(factor(comparison)~solo_WinRatio+solo_KillDeathRatio+solo_TimeSurvivedPg+duo_WinRatio+

duo_KillDeathRatio+duo_TimeSurvivedPg+duo_RoundsPlayed+duo_RevivesPg+duo_Revives+duo_Team
Kills+

squad_WinRatio+squad_KillDeathRatio+squad_TimeSurvivedPg+squad_RoundsPlayed+squad_RevivesPg
+squad_Revives+squad_TeamKills,data = pubg_statistics_train,method="class")



plot(randomForest_plot,uniform = TRUE,main="Classifiacation Tree")

text(randomForest_plot,use.n = FALSE, all = TRUE, cex=0.6)
```