Data Mining for Business Intelligence
IS7036

UNIVERSITY OF Cincinnati

Project Report

Cluster Analysis on FIFA 20

Sayak Chakraborty | Samreen Zehra | Niharika Gupta | Rohit Thakur

University of CINCINNATI

Introduction

FIFA, also known as FIFA Football or FIFA Soccer, is a series of association football video games or football simulator, released annually by Electronic Arts under the EA Sports label. Football video games such as Sensible Soccer, Kick Off and Match Day had been developed since the late 1980s and already competitive in the games market when EA Sports announced a football game as the next addition to their EA Sports label. The Guardian called the series "the slickest, most polished and by far the most popular football game around.

- As of 2011, the FIFA franchise has been localised into 18 languages and available in 51 countries.
- Listed in Guinness World Records as the best-selling sports video game franchise in the world, by 2018, the FIFA series had sold over 260 million copies.
- It is also one of the best-selling video game franchises.

Source: **Wikipedia**

Data Source: **Kaggle FIFA 20 Data Set**

Problem Statement

In professional football, it is not uncommon for a player to drop out of team due to contract transfer, contract expiration or medical reasons. By the means of our analysis, we will be trying to find suitable replacements players based on their skillset, strength and physical attributes. We will be doing that by using Clustering Analysis which is an unsupervised learning method. We will be building two different models using K-means clustering and Hierarchical clustering and in the end compare both the models to see which one performs better.

Data Preparation

Packages Used

We begin by loading the packages that will be required throughout the course of our analysis.

```
library (tidyverse)
library (ggfortify)
library (qdapTools)
library (caret)
library (tidyr)
library (corrplot)
library (ggpubr)
library(ggplot2)
library(PerformanceAnalytics)
library(factoextra)
library(fpc)
```

Importing the Data

The data set was imported into R studio using the below code

```
fifa_data <- read.csv("fifa-20-complete-player-dataset/players_20.csv", stringsAsFactors = FALSE)
```

```
attach(fifa_data)

set.seed(123456)
```

Displaying the Top 10 Values of the Data

```
head (fifa_data, 10)
```

Structure of the data

Next we use the str() function to see the structure of the dataset.

```
str(fifa_data)
```

Observations:

The dataset has 18278 observations and 104 variables. The dataset has both numeric and character variables.

Glimpse of the data

```
glimpse(fifa_data)
```

Data Cleaning - Splitting into 2 groups - Goalkeepers and Outfield Players

Both goalkeepers and outfield players have different set of attributes. It makes more sense to divide the whole dataset into two groups: Outfield players and Goal keepers so that there are no Null values as we can assign only the attributes that are relevant to each group. This will also help us to align our model to our problem statement.

Data Set - Goalkeepers

```
data_gk <- subset(x=fifa_data, subset= player_positions == "GK", select = c(sofifa_id,short_name,age,height_cm,weight_kg,overall,potential,value_eur,wage_eur,player_positions,international_reputation,weak_foot,skill_moves,gk_diving,gk_handling,gk_kicking,gk_reflexes,gk_speed,gk_positioning,attacking_crossing,attacking_finishing,attacking_heading_accuracy,attacking_short_passing,attacking_volleys,skill_dribbling,skill_curve,skill_fk_accuracy,skill_long_passing,skill_ball_control,movement_acceleration,movement_sprint_speed,movement_agility,movement_reactions,movement_balance,power_shot_power,power_jumping,power_stamina,power_strength,power_long_shots,mentality_aggression,mentality_interceptions,mentality_positioning,mentality_vision,mentality_penalties,mentality_composure,defending_marking,defending_standing_tackle,defending_sliding_tackle,goalkeeping_diving,goalkeeping_handling,goalkeeping_kicking,goalkeeping_positioning,goalkeeping_reflexes))
```

Analysing the Goalkeeper Dataset - Head, Dimension and Glimpse

```
head(data_gk)
dim(data_gk)
glimpse(data_gk)
```

Observations:

The goalkeeper dataset has 2036 observations and 53 variables.

Outfield Players Dataset

```
data_outfield <- subset(x=fifa_data, subset= player_positions != "GK", select = c(sofifa_id,short_name,age,height_cm,weight_kg,overall,potential,value_eur,wage_eur,player_positions,international_reputation,weak_foot,skill_moves,pace,shooting,passing,dribbling,defending,physic,attacking_crossing,attacking_finishing,attacking_heading_accuracy,attacking_short_passing,attacking_volleys,skill_dribbling,skill_curve,skill_fk_accuracy,skill_long_passing,skill_ball_control,movement_acceleration,movement_sprint_speed,movement_agility,movement_reactions,movement_balance,power_shot_power,power_jumping,power_stamina,power_strength,power_long_shots,mentality_aggression,mentality_interceptions,mentality_positioning,mentality_vision,mentality_penalties,mentality_composure,defending_marking,defending_standing_tackle,defending_sliding_tackle,goalkeeping_diving,goalkeeping_handling,goalkeeping_kicking,goalkeeping_positioning,goalkeeping_reflexes))
```

Analysing the Outfield Dataset - Head, Dimension and Glimpse

```
head(data_outfield)
dim(data_outfield)
glimpse(data_outfield)
```

Observations:

The Outfield dataset has 16242 observations and 53 variables.

Creating Data Sets for Goalkeepers and Outfield Players by only taking Physical, Strength and Skill Attritubes for Clustering

- Since we are using physical attributes, strength and skillset to find suitable replacement players, we will be filtering the datasets to retain only these variables.
- The resulting datasets have a total of 47 variables each.

```
data_gk_int <- subset(data_gk,select = -c(short_name, player_positions, sofifa_id, value_eu
r, wage_eur, skill_moves))
glimpse (data_gk_int)


data_outfield_int <- subset(data_outfield, select = -c(short_name, player_positions, sofifa_i
d, value_eur, wage_eur, skill_moves))
glimpse(data_outfield_int)
```

Scaling the Dataframes

Next as a part of Data Pre Processing we use the scale() function to basically normalize the datasets. This function places continuous variables on unit scale by subtracting the mean of the variable and dividing the result by the variable's standard deviation (also sometimes called z-scoring or simply scaling). The result is that the values in the transformed variable have the same relationship to one another as in the untransformed variable, but the transformed variable has mean 0 and standard deviation.

```
data_gk_int <- scale(data_gk_int)

data_outfield_int <- scale(data_outfield_int)

set.seed(123456)
```

EDA - Exploratory Data Analysis

Sometimes same player takes over several positions, sometimes only one like a goal keeper. Also notice that names of positions separated by comma and space if several are present. What we can do here is to display for each player it's relative relation to a position. This can be accomplished in few steps.

- We remove white spaces from all cells in player_positions column.
- We separate any string in the cell by comma if there any comma present.
- We transform column into a matrix N x n where N - number of rows in the data (total number of rows present in the table) and n is the total number of unique attributes found in the column.

In our case we have 15 attributes, so the table will be N x 15.

```
print ("Players position")
```

```
## [1] "Players position"
```

```
head (player_positions)
```

```
## [1] "RW, CF, ST" "ST, LW"     "LW, CAM"    "GK"         "LW, CF"
## [6] "CAM, CM"
```

```
player_positions <- gsub('\\s+', ",player_positions)#remove white spaces from specified colu
mns
players_role <- mtabulate(strsplit(player_positions, ",")) #one hot encode player position
```

```
print ("Number of entries (rows) and number of (columns) of the matrix")
```

```
## [1] "Number of entries (rows) and number of (columns) of the matrix"
```

```
dim (players_role)
```

```
## [1] 18278    15
```

Frquency of Positions Played

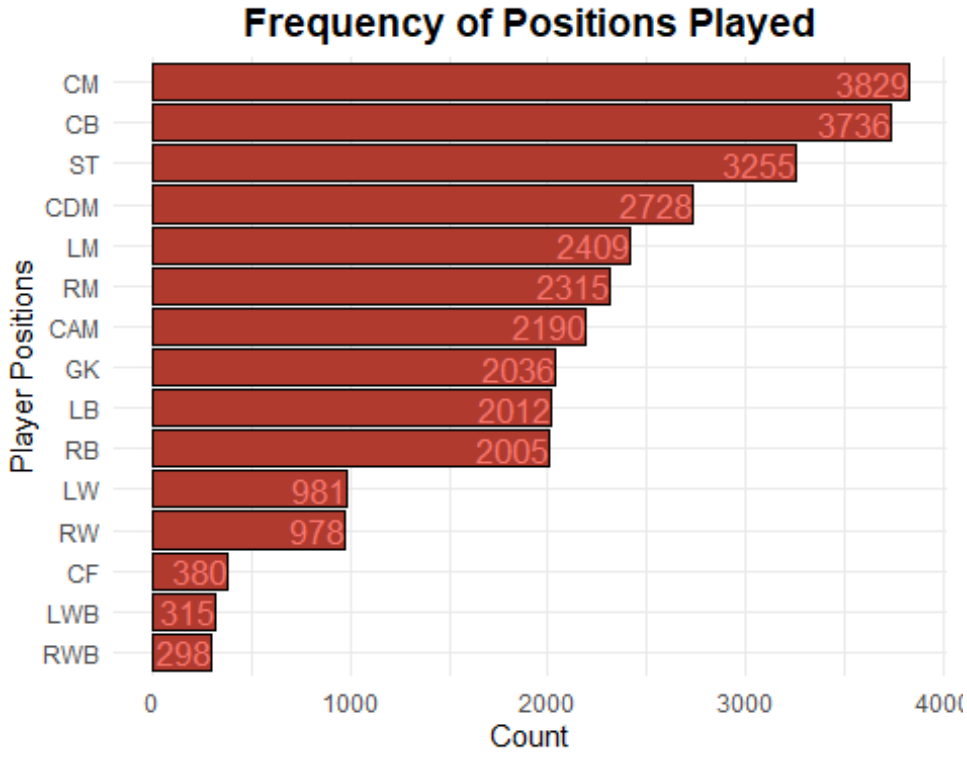
```
role_frequency <- t(sort(apply (players_role, 2, sum))) #count number of players position appearing

role_frequency <- gather (data.frame(role_frequency), Position, Count) #transfer data into format for ggplot

role_frequency$Position <- reorder(role_frequency$Position, role_frequency$Count)#reorder position by ascending count

#barplot of players position frequency

ggplot(role_frequency, aes(x=Position, y = Count, fill = Position)) +
  geom_bar(stat = "identity", color ="black", fill = "#B03A2E") +
  coord_flip() +
  theme_minimal() +
  geom_text(aes(label = Count, hjust = 1, color = "#FDFEFE" , size = 3.5)) +
  ggtitle("Frequency of Positions Played") +
  xlab("Player Positions") +
  ylab("Count") +
  theme(legend.position = "none",
        plot.title = element_text(color = "black", size = 14, face = "bold", hjust = 0.5),
        plot.subtitle = element_text(color = "darkblue", hjust = 0.5),
        axis.title.y = element_text(),
        axis.title.x = element_text(),
        axis.ticks = element_blank())
```



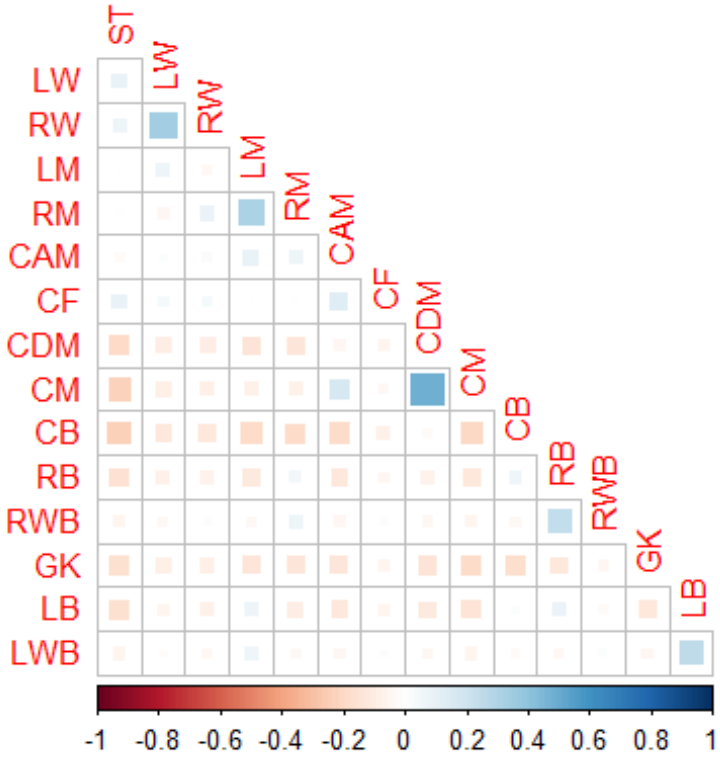
Correlation amongst outfield players

To see correlation between values in the table (players_role) we can compute correlation between them by applying function cor. It requires only numeric columns, make sure you do not have any rows with string inside.

We can plot corrplot using corrplot function of corrplot package. One can specify various arguments of a function. corrplot:: specifies the namespace of a package. It can be helpful when two packages have different function but share same function name.

```
role_cor <- cor(players_role) #correlation between players role

corrplot::corrplot(role_cor, hclust.method = "ward.D", order = "hclust", method = "square",
diag = F, type = "lower" ) #correlation plot
```



Observation:

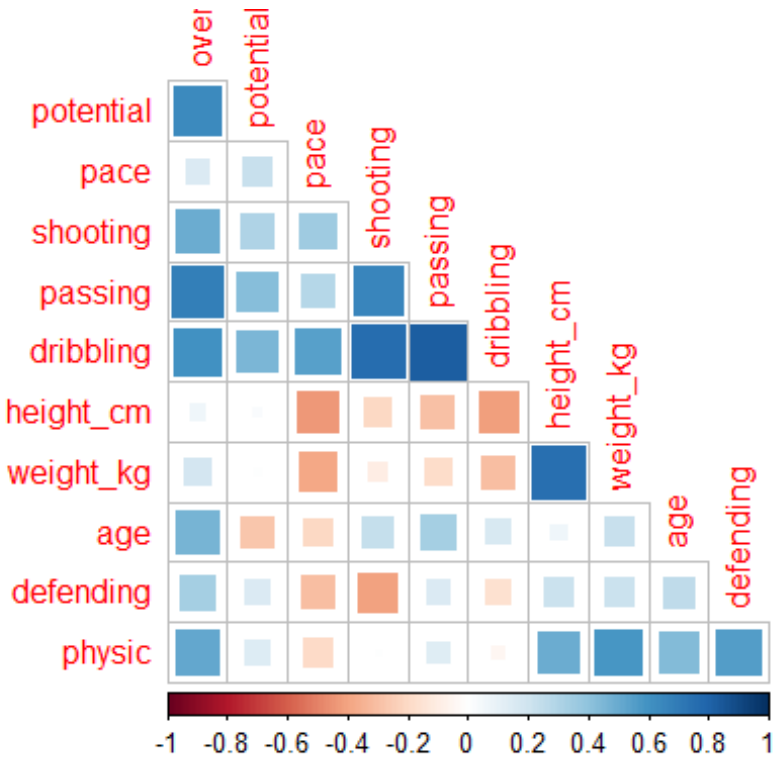
We can see that mid center players often share position of a center defending midfielder and less with center attacking midfielder. Left and right back wingers clearly have positive correlation so as right and left mid-siders.

It seems that central forward position (ST) is negatively correlated with (CM - Central middle, CDM - Central defense middle, CB - Central backfielder, RB - right backfielder) a little less with LB - left backfielder. Clearly players playing in the forward position rarely play in the defense

Correrlation of key statistics from Outfield dataframes:

```
data_outfield_Summary <- subset(x=data_outfield, select = c(age,height_cm,weight_kg, overall, potential,pace,shooting,passing,dribbling,defending,physic))

corr1 <- cor(data_outfield_Summary)
corrplot::corrplot(corr1, hclust.method = "ward.D", order = "hclust", method = "square",diag = F, type = "lower" ) #correlation plot for Outfield Players
```




```
#chart.Correlation(data_outfield_Summary, histogram=TRUE, pch=19)
```

Cluster Analysis

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including pattern recognition, image analysis, information retrieval, bioinformatics, data compression, computer graphics and machine learning.

More Information can be found on **Wikipedia**

K-Means Clustering

K-Means Clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. It is popular for cluster analysis in data mining. k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, Better Euclidean solutions can be found using k-medians and k-medoids.

More Information can be found on **Wikipedia**

We want to view the result so We can use fviz_cluster. It is function can provide a nice graph of the clusters. Usually, we have more than two dimensions (variables) fviz_cluster will perform principal component analysis (PCA) and plot the data points according to the first two principal components that explain the majority of the variance.

Cluster Analysis for Goalkeepers

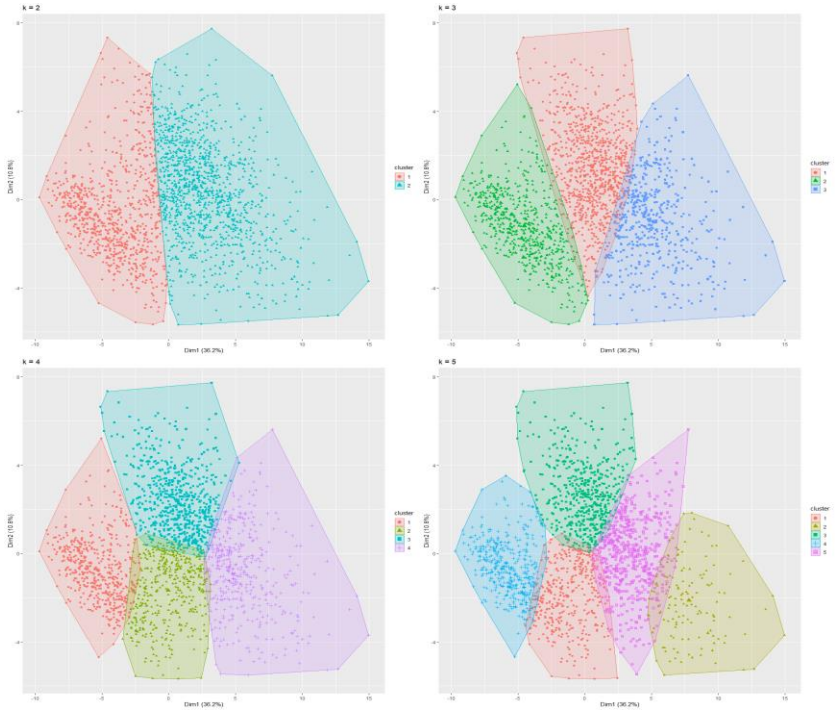
In order to use k-means method for clustering and plot results, we can use k means function in R. It will group the data into a specified number of clusters (centers = 2, 3, 4, 5). The nstart option of this function can allow the algorithm to attempt multiple initial configurations and reports on the best one.

In order to view the results, swe can use fviz_cluster function which can provide a nice graph of the clusters. Usually, we have more than two dimensions (variables) fviz_cluster will perform principal component analysis (PCA) and plot the data points according to the first two principal components that explain the majority of the variance.

```
gk2 <- kmeans(data_gk_int, centers = 2, nstart = 25)
gk3 <- kmeans(data_gk_int, centers = 3, nstart = 25)
gk4 <- kmeans(data_gk_int, centers = 4, nstart = 25)
gk5 <- kmeans(data_gk_int, centers = 5, nstart = 25)

# plots to compare
gkp1 <- fviz_cluster(gk2, geom = "point",  data = data_gk_int) + ggtitle("k = 2")
gkp2 <- fviz_cluster(gk3, geom = "point",  data = data_gk_int) + ggtitle("k = 3")
gkp3 <- fviz_cluster(gk4, geom = "point",  data = data_gk_int) + ggtitle("k = 4")
gkp4 <- fviz_cluster(gk5, geom = "point",  data = data_gk_int) + ggtitle("k = 5")

library(gridExtra)
grid.arrange(gkp1, gkp2, gkp3, gkp4, nrow = 2)
```

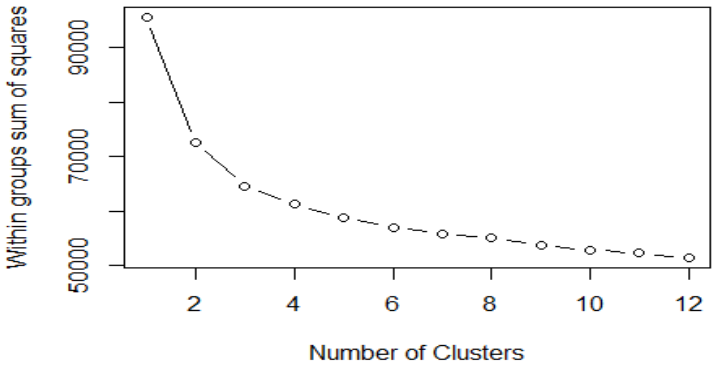


Determine the Number of Clusters

To determine the number of clusters, we use a simple within group sum of squares method. The idea is that, if the plot is an arm, the elbow of the arm is the optimal number of clusters, which is 3 in our case

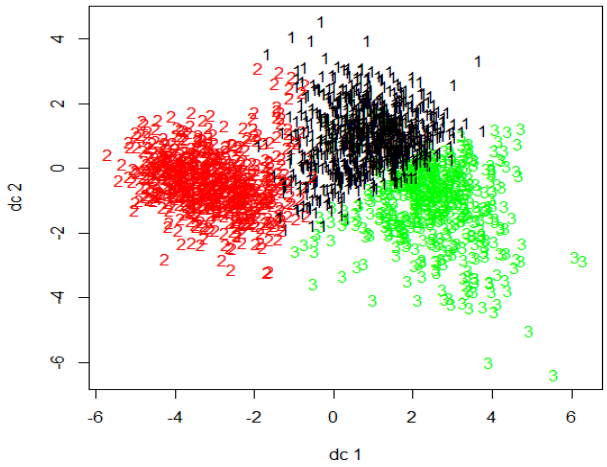
```
# Determine number of clusters
wss <- (nrow(data_gk_int)-1)*sum(apply(data_gk_int,2,var))

for (i in 2:12) wss[i] <- sum(kmeans(data_gk_int,
                                     centers=i)$withinss)
plot(1:12, wss, type="b", xlab="Number of Clusters",ylab="Within groups sum of squares")
```



Plotting the Clusters for Goalkeepers

```
library(fpc)
plotcluster (data_gk_int, gk3$cluster)
```



Checking the Prediction Strength - Goalkeepers

Next we use the prediction strength with a cutoff value of 0.8. This will recommends the largest number of clusters that leads to a prediction strength above 0.8 which is again 3 in our case.

```
prediction.strength(data_gk_int, Gmin=2, Gmax=15, M=10,cutoff=0.8)
```

```
## Prediction strength
## Clustering method:  kmeans
## Maximum number of clusters:  15
## Resampled data sets:  10
## Mean pred.str. for numbers of clusters:  1 0.9465497 0.8432161 0.5933725 0.5177097 0.4
696646 0.4155968 0.337572 0.3162195 0.3031324 0.303777 0.2817019 0.2539696 0.233146 0
.2034054
## Cutoff value:  0.8
## Largest number of clusters better than cutoff:  3
```

Output of the K-Means - Goalkeepers

```
#use the output of kmeans
gk3$centers
```

```
##         age   height_cm   weight_kg     overall  potential
## 1  0.2722008  0.02397076  0.07642374 -0.07399651 -0.3579285
## 2 -0.8056819 -0.12273409 -0.30715291 -0.85725102 -0.2796168
## 3  0.6092431  0.12395932  0.28095721  1.29565980  1.0169297
##   international_reputation   weak_foot   gk_diving gk_handling gk_kicking
## 1              -0.2350217 -0.06479749 -0.09311223  -0.1257872 -0.1234589
## 2              -0.2444925 -0.11650054 -0.79456140  -0.7315079 -0.6834791
## 3               0.7504065  0.27354667  1.24457994   1.2171482  1.1477927
##   gk_reflexes   gk_speed gk_positioning attacking_crossing
## 1 -0.08862617  0.3399101     -0.0388205         0.2646941
## 2 -0.79693478 -0.9586043     -0.8238856        -0.4995433
## 3  1.23981494  0.6963138      1.1877283         0.2069570
##   attacking_finishing attacking_heading_accuracy attacking_short_passing
## 1           0.5003257                 0.2688590             0.09051013
```

```
## 2         -0.9545552             -0.4840682          -0.49828966
## 3          0.4051983              0.1785306           0.51538745
##   attacking_volleys skill_dribbling skill_curve skill_fk_accuracy
## 1         0.5142830       0.3294312  0.2206243         0.2378758
## 2        -0.9583390      -0.7315953 -0.4934196        -0.4716182
## 3         0.3854848       0.4067560  0.2771085         0.2167922
##   skill_long_passing skill_ball_control movement_acceleration
## 1         0.00199558          0.2531884            0.3124755
## 2        -0.39767486         -0.7311943           -0.9113362
## 3         0.53637942          0.5419610            0.6809839
##   movement_sprint_speed movement_agility movement_reactions
## 1             0.3375020       0.1554552         0.06137424
## 2            -0.9280013      -0.6517950        -0.87298190
## 3             0.6590510       0.6081713         1.07599202
##   movement_balance power_shot_power power_jumping power_stamina
## 1        0.2324551       -0.1232354    0.02396498     0.2473939
## 2       -0.5164536       -0.6812621   -0.56030709    -0.7255783
## 3        0.2873176        1.1443847    0.71807279     0.5446529
##   power_strength power_long_shots mentality_aggression
## 1     0.07560664        0.5100448           0.09599788
## 2    -0.49088483       -0.9914499          -0.47999630
## 3     0.53186919        0.4379863           0.48077923
##   mentality_interceptions mentality_positioning mentality_vision
## 1               0.3898286             0.4477818       -0.1791500
## 2              -0.9282583            -0.9424443       -0.2905230
## 3               0.5662330             0.4823089        0.7134243
##   mentality_penalties mentality_composure defending_marking
## 1           0.2483565           0.0386662         0.2111671
## 2          -0.6792757          -0.6779313        -0.5391643
## 3           0.4800728           0.8515986         0.3560557
##   defending_standing_tackle defending_sliding_tackle goalkeeping_diving
## 1                 0.2964850                0.3273071        -0.09311223
## 2                -0.5304807               -0.5343752        -0.79456140
## 3                 0.1923582                0.1427675         1.24457994
##   goalkeeping_handling goalkeeping_kicking goalkeeping_positioning
## 1           -0.1257872          -0.1234589             -0.0388205
## 2           -0.7315079          -0.6834791             -0.8238856
```

```
## 3          1.2171482          1.1477927          1.1877283
##   goalkeeping_reflexes
## 1          -0.08862617
## 2          -0.79693478
## 3           1.23981494
```

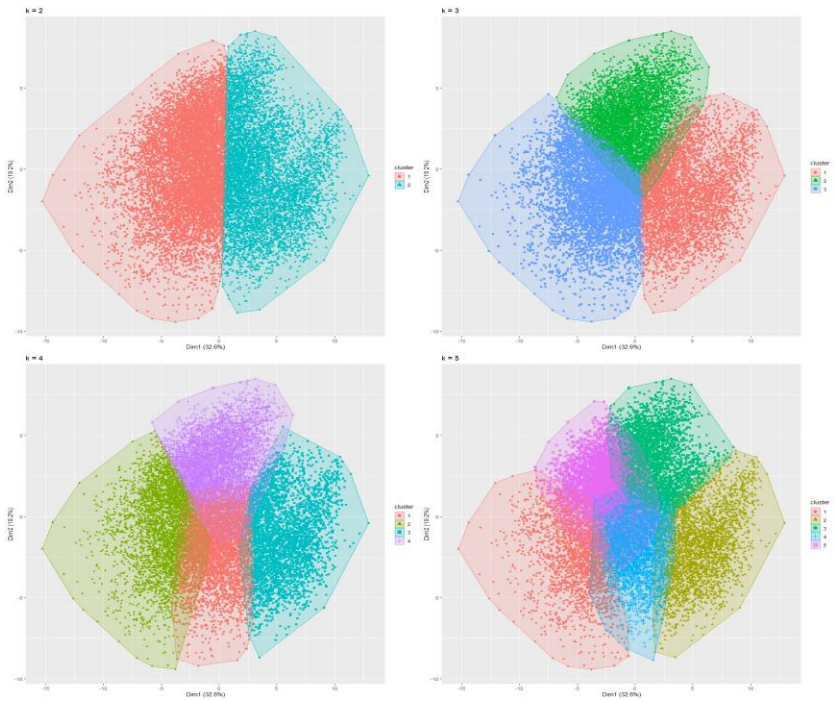
Cluster Analysis for Outfield Players

Next we perform cluster analysis of outfield players using kmeans() function and centers value of 2, 3, 4 and 5,.

```
ok2 <- kmeans(data_outfield_int, centers = 2, nstart = 25)
ok3 <- kmeans(data_outfield_int, centers = 3, nstart = 25)
ok4 <- kmeans(data_outfield_int, centers = 4, nstart = 25)
ok5 <- kmeans(data_outfield_int, centers = 5, nstart = 25)

# plots to compare
op1 <- fviz_cluster(ok2, geom = "point",  data = data_outfield_int) + ggtitle("k = 2")
op2 <- fviz_cluster(ok3, geom = "point",  data = data_outfield_int) + ggtitle("k = 3")
op3 <- fviz_cluster(ok4, geom = "point",  data = data_outfield_int) + ggtitle("k = 4")
op4 <- fviz_cluster(ok5, geom = "point",  data = data_outfield_int) + ggtitle("k = 5")

library(gridExtra)
grid.arrange(op1, op2, op3, op4, nrow = 2)
```

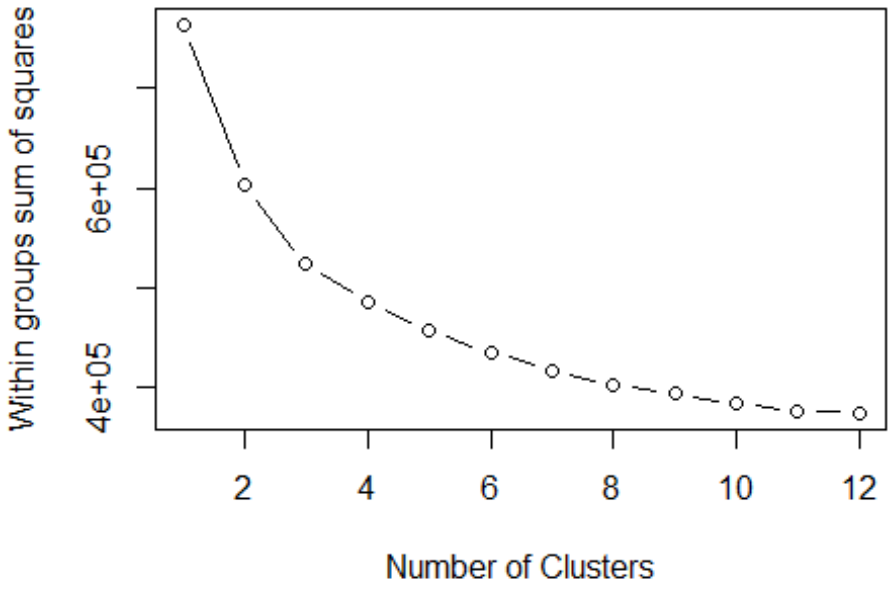


Determine the Number of Clusters - Outfield Players

Using the sum of squares method method, we find the optimal number of clusters to be 3.

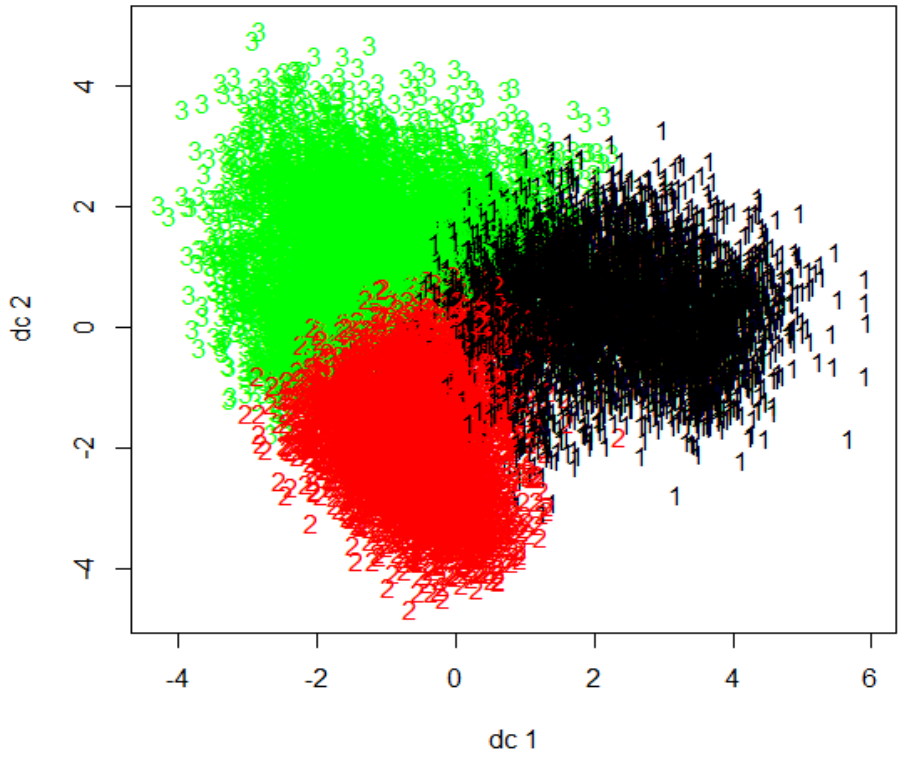
```
# Determine number of clusters
wss_o <- (nrow(data_outfield_int)-1)*sum(apply(data_outfield_int,2,var))

for (i in 2:12) wss_o[i] <- sum(kmeans(data_outfield_int,
                                       centers=i)$withinss)
plot(1:12, wss_o, type="b", xlab="Number of Clusters",ylab="Within groups sum of squares
")
```



Plotting the Clusters for Outfield Players

```
library(fpc)
plotcluster(data_outfield_int, ok3$cluster)
```



Checking the Prediction Strength - Outfield Players

Using prediction strength function with a cut-off value of 0.8, we get the largest number of clusters to be 3.

```
prediction.strength(data_outfield_int, Gmin=2, Gmax=15, M=10,cutoff=0.8)
```

```
## Prediction strength
## Clustering method:  kmeans
## Maximum number of clusters:  15
## Resampled data sets:  10
## Mean pred.str. for numbers of clusters:  1 0.9557498 0.9313265 0.7640775 0.5269017 0.6
28195 0.4823138 0.6050884 0.4981196 0.4508408 0.3644348 0.3822084 0.334616 0.339305 0
.2876533
## Cutoff value:  0.8
## Largest number of clusters better than cutoff:  3
```

Output of the K-Means - Outfield Players

```
#use the output of kmeans
ok3$centers
```

```
##          age  height_cm   weight_kg    overall  potential
## 1 -0.08568948  0.5346909  0.41946219 -0.3798785 -0.3049445
## 2 -0.42969900 -0.3353175 -0.37083635 -0.5326929 -0.2225367
## 3  0.45351852 -0.1524315 -0.02391674  0.7925551  0.4541936
##   international_reputation   weak_foot       pace   shooting    passing
## 1              -0.1971186 -0.37270635 -0.6383253 -1.1528294 -0.8197510
## 2              -0.2618142  0.04421257  0.3437641  0.3439705 -0.2548250
## 3               0.3982977  0.27446737  0.2321643  0.6650429  0.9161684
##    dribbling  defending     physic attacking_crossing attacking_finishing
## 1 -1.0279982  0.6420237  0.3122345        -0.7007539         -1.1229190
## 2  0.1232610 -1.0483196 -0.7949969        -0.1831150          0.4803348
## 3  0.7558693  0.3900735  0.4428148         0.7523592          0.5188326
##   attacking_heading_accuracy attacking_short_passing attacking_volleys
## 1                  0.2615464            -0.5857615       -1.0113119
## 2                 -0.5048306            -0.3728760        0.2651309
## 3                  0.2279329             0.8239977        0.6159031
##   skill_dribbling skill_curve skill_fk_accuracy skill_long_passing
```

```
## 1      -1.0238832 -0.94549696      -0.80054354       -0.4196781
## 2       0.1833426  0.01184035      -0.08322047       -0.4657584
## 3       0.6990784  0.78532239       0.74768830        0.7666449
##   skill_ball_control movement_acceleration movement_sprint_speed
## 1        -0.89994868           -0.6525584           -0.5904491
## 2        -0.05558207            0.3472162            0.3219244
## 3         0.80682699            0.2410804            0.2112510
##   movement_agility movement_reactions movement_balance power_shot_power
## 1       -0.8136287        -0.3707109       -0.6576659      -0.92999730
## 2        0.2986811        -0.5433700        0.2830223       0.08213972
## 3        0.4197332         0.7943154        0.3023568       0.70987977
##   power_jumping power_stamina power_strength power_long_shots
## 1     0.1684715    -0.1782467      0.4084593       -1.0598994
## 2    -0.3729318    -0.5034700     -0.5813399        0.1597833
## 3     0.1892038     0.5969024      0.1721837        0.7503042
##   mentality_aggression mentality_interceptions mentality_positioning
## 1            0.3103383             0.5856178           -1.0830421
## 2           -0.8551033            -1.0325543            0.2811159
## 3            0.4977601             0.4235577            0.6620911
##   mentality_vision mentality_penalties mentality_composure
## 1      -0.97396117         -0.9072036        -0.5247257
## 2       0.01122459          0.2717303        -0.3940798
## 3       0.80982746          0.5224170         0.7914436
##   defending_marking defending_standing_tackle defending_sliding_tackle
## 1         0.5870453              0.6466842              0.6604607
## 2        -0.9742263             -1.0114443             -0.9716489
## 3         0.3705854              0.3534209              0.3065035
##   goalkeeping_diving goalkeeping_handling goalkeeping_kicking
## 1        -0.01851878         -0.01831813        -0.04045735
## 2        -0.04997539         -0.05347249        -0.04340598
## 3         0.05994464          0.06287972         0.07257961
##   goalkeeping_positioning goalkeeping_reflexes
## 1             -0.02397761          -0.03111060
## 2             -0.05132719          -0.03430055
## 3              0.06573921           0.05663056
```

Hierarchical Clustering

In data mining and statistics, hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a hierarchy of clusters.

Strategies for hierarchical clustering generally fall into two types:

1. Agglomerative: This is a "bottom-up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
2. Divisive: This is a "top-down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In general, the merges and splits are determined in a greedy manner. The results of hierarchical clustering[2] are usually presented in a dendrogram.

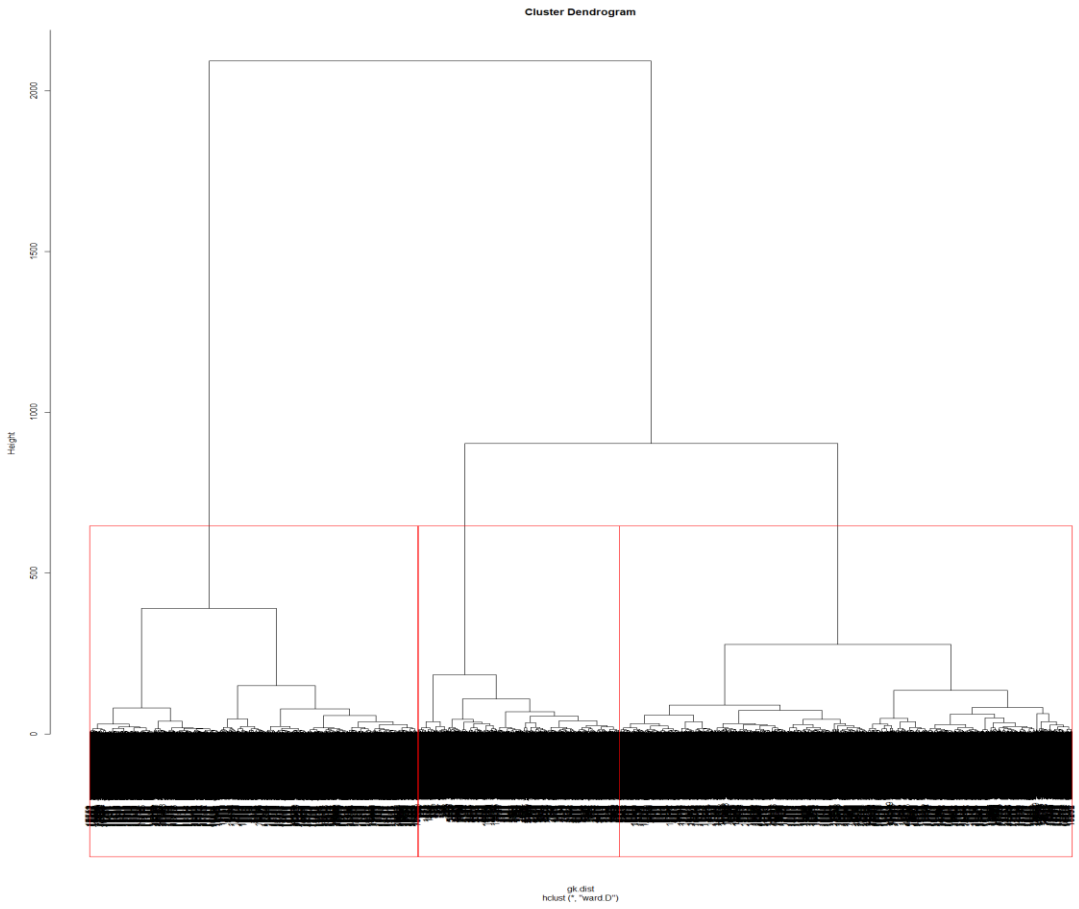
More Information can be found on **Wikipedia**

Hierarchical Clustering Analysis for Goalkeepers

```
#Wards Method or Hierarchical clustering
#Calculate the distance matrix
gk.dist = dist(data_gk_int)
#Obtain clusters using the Wards method
gk.hclust = hclust(gk.dist, method = "ward")

plot(gk.hclust)

rect.hclust(hclust(gk.dist, method = "ward"), h = 500)
```



Sayak Chakraborty | Samreen Zehra | Niharika Gupta | Rohit Thakur

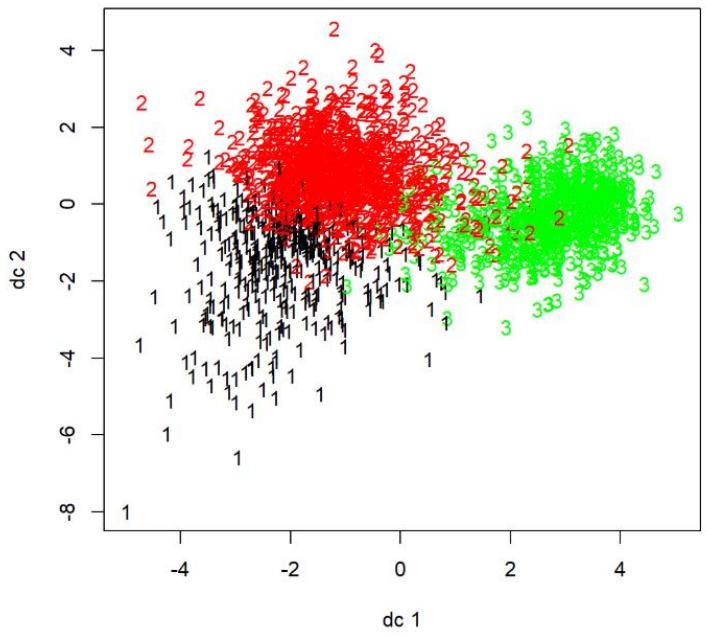
Views the items in the selected cluster - Goalkeepers

```
#Cut dendrogram at the 3 clusters level and obtain cluster membership
gk.3clust = cutree(gk.hclust,k=3)

#See exactly which item are in third group
data_gk_int[gk.3clust==3,]
```

Plotting the hclust() for Goalkeepers (k=3)

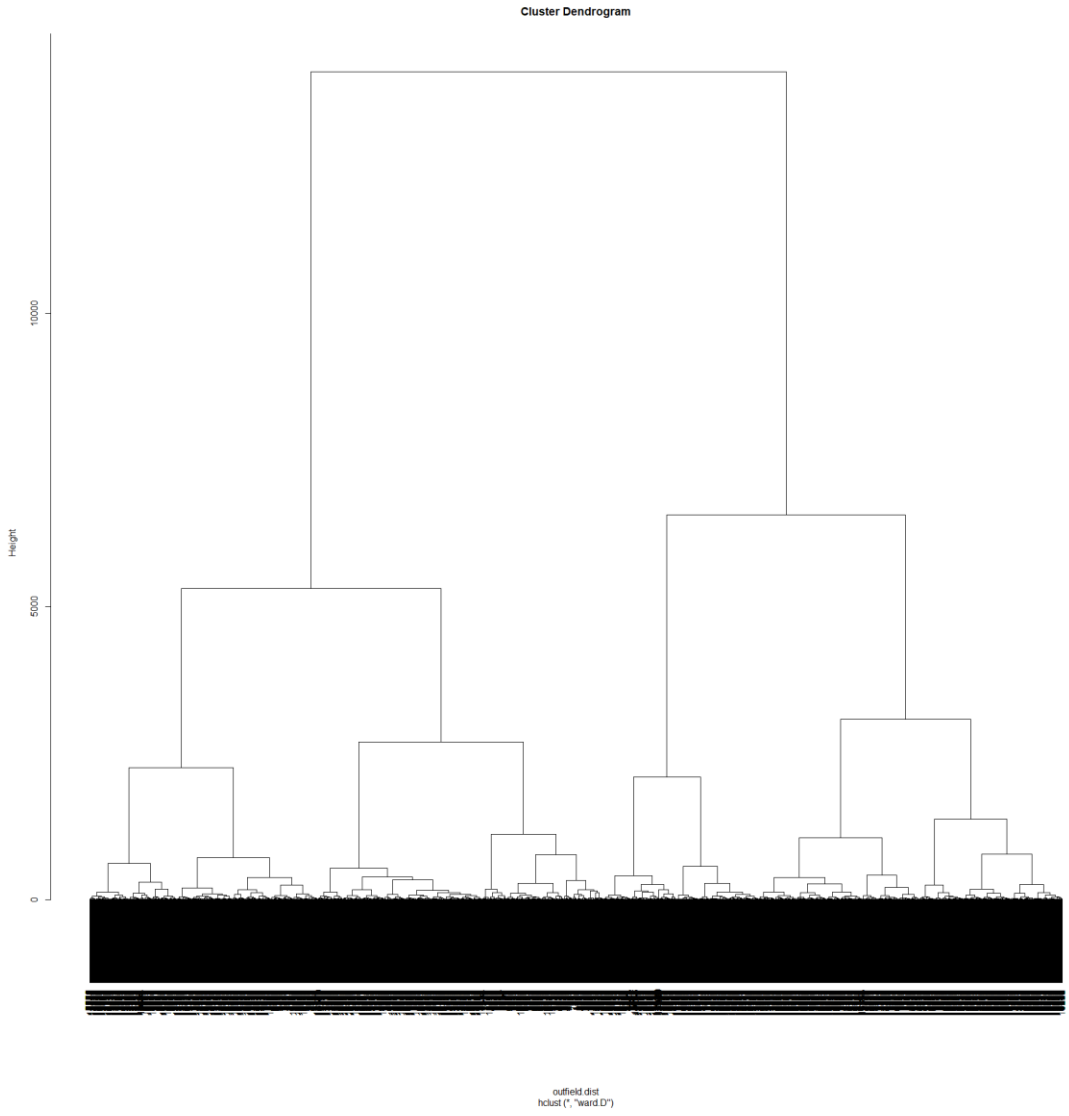
```
#get cluster means for raw data
#Centroid Plot against 1st 2 discriminant functions
#Load the fpc library needed for plotcluster function
library(fpc)
#plotcluster(ZooFood, fit$cluster)
plotcluster(data_gk_int, gk.3clust)
```



Hierarchical Clustering Analysis for Outfield Players

```
#Wards Method or Hierarchical clustering
#Calculate the distance matrix
outfield.dist = dist(data_outfield_int)
#Obtain clusters using the Wards method
outfield.hclust = hclust(outfield.dist, method = "ward")

plot(outfield.hclust)
```



Sayak Chakraborty | Samreen Zehra | Niharika Gupta | Rohit Thakur

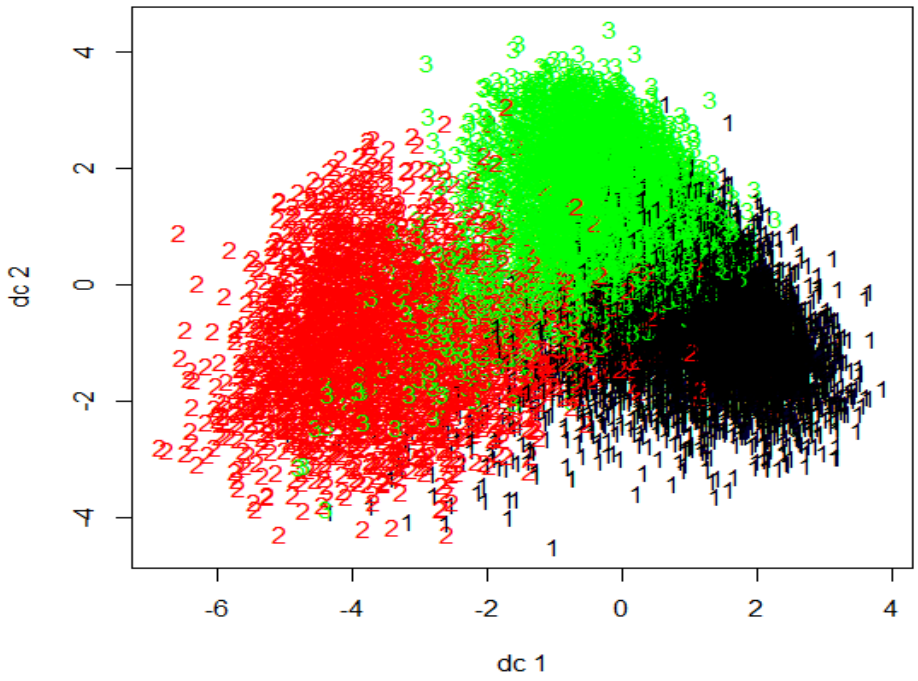
Views the items in the selected cluster - Outfield Players

```
#Cut dendrogram at the 3 clusters level and obtain cluster membership
outfield.3clust = cutree(outfield.hclust,k=3)

#See exactly which item are in third group
data_outfield_int[outfield.3clust==3,]
```

Plotting the hclust() for Outfield Players (k=3)

```
#get cluster means for raw data
#Centroid Plot against 1st 2 discriminant functions
#Load the fpc library needed for plotcluster function
library(fpc)
#plotcluster(ZooFood, fit$cluster)
plotcluster(data_outfield_int, outfield.3clust)
```


dc 2
dc 1
4
2
0
-2
-4
-6
-4
-2
0
2
4

Comparison

Hierarchical Clustering vs K-Means Clustering

Hierarchical clustering builds clusters within clusters and does not require a pre-specified number of clusters like k means. Now we discuss this difference in details. The most important difference is the hierarchy. Actually, there are two different approaches that fall under this name: top-down and bottom-up.

In top-down hierarchical clustering, we divide the data into 2 clusters (using k-means with k=2k=2, for example). Then, for each cluster, we can repeat this process, until all the clusters are too small or too similar for further clustering to make sense, or until we reach a preset number of clusters.

In bottom-up hierarchical clustering, we start with each data item having its own cluster. We then look for the two items that are most similar and combine them in a larger cluster. We keep repeating until all the clusters we have left are too dissimilar to be gathered together, or until we reach a preset number of clusters.

In k-means clustering, we try to identify the best way to divide the data into k sets simultaneously. A good approach is to take k items from the data set as initial cluster representatives, assign all items to the cluster whose representative is closest, and then calculate the cluster mean as a new representative; until it converges (all clusters stay the same)

Reference: **StepupAnalytics.com**

Comparing kmeans() and hclust()

Goalkeepers

```
# Apply cutree() to gk.hclust : gk.cut
gk.cut <- cutree(gk.hclust , k = 3)

# Compare methods
table(gk3$cluster, gk.cut)
```

```
##    gk.cut
##       1   2   3
##   1  27 800  49
##   2   0  55 613
##   3 390  83  19
```

Outfield Players

```
# Apply cutree() to gk.hclust : gk.cut
outfield.cut <- cutree(outfield.hclust  , k = 3)

# Compare methods
table(ok3$cluster, outfield.cut)
```

```
##    outfield.cut
##       1    2    3
##   1  372 2479 2158
##   2 2552    1 2729
##   3 5652  108  191
```

Conclusion

Summary of The Problem

In professional football, it is not uncommon for a player to drop out of team due to contract transfer, contract expiration or medical reasons. By the means of our analysis, we tried to find suitable replacements players based on their skillset, strength and physical attributes. We performed Clustering Analysis which, an unsupervised learning method and built two different models using K-means clustering and Hierarchical clustering and in the end compared both the models to see which one performs better.

Methodology

The methodology involved cleaning the dataset and then splitting it in 2 parts - and Outfield Players. And then by only taking the physical attributes into consideration , performing clustering analysis - K mean Cluster Analysis and Hierarchical Analysis on both of the datasets separately to cluster similar physical type of player together and making it easier for the team management to find a suitable replacement in case of contract transfer, contract expiration or medical reasons.

Implication of the Analysis

This will be useful for team management as it will narrow down their search for a suitable replacement.