Machine Learning : Unsupervised Learning

PCA, K-means, Hierarchical clustering and Market Basket Analysis

Karnam Yogesh, Yashraj Varma, SriRajitha, Kishan R, Akash Gupta

Preface

The competition among European clubs in Football has always been cut-throat. However today, with the increasing use of Data Analytics, easier access to funding and global reach, the competition has increased dramatically. Traditionally 'smaller' clubs like Liecester City are able to give the big clubs like - Liverpool FC, Manchester City and Chelsea - a run for their money. But the fundamental point remains - they all want to win trophies and it is quite obvious that in order to win trophies, a club needs the best players. We are also aware of the fact that many clubs are quite ambitious but have financial constraints. A rather tricky situation faced by most clubs today is that, even though they are all wanting to attract the best players, the very best of them cost a lot in the transfer market - prices can go up to 100 Million Euros! So this lays the foundation for our analyses. We have been able to gather detailed data for almost 500 player's skills and specifications. Our aim is therefore to use unsupervised learning mechanisms to form clusters of 'similar' players such that any club with financial constraints can remain fairly competitive by buying the best players at the lowest transfer cost. As is explained, we ultimately find that players can be grouped into 4 broad clusters, out of which the best players can be purchased.

The Data

Data regarding players has been scraped from **www.sofifa.com** and made available on Kaggle. The dataset we obtained originally was massive in size, containing information of almost 17000 players with 75 attributes. A key point to note here is that we chose to narrow down our analysis objective to the top 500 players (in terms of overall rating given to them by FIFA). This was a rather intuitive decision since we felt that practically, any club in a major league would really not look to buy a low-rated player from an obscure club. Given the highly competitive nature of the sport, the major clubs would primarily look at the pool of 400 to 500 best players as per FIFA ratings. Due to this practical consideration, we have looked to conduct our analysis on this subset of players. The central idea is this - Everyone would want to buy someone like **Christiano Ronaldo**, however he is priced at almost 100 Million Euros. So the next best thing a club can do is to purchase say - **Morata** - for less than half the price and enjoy similar benefits that a Ronaldo would give them, since these players essentially belong to the **same cluster**. Now there were various problems we faced while preparing this dataset for analyses. We noticed that there were many correlations between measurements, along with missing values and incompatible data types. ***Note that all measurements of player skills are ratings given by FIFA on a scale of 100***.

- **Trim the Dataset rows** :- As mentioned, we chose to analyse the top 500 players, so the players were filtered on the basis of their **overall rating** being above 80.
- **Removing unnecessary columns** :- Columns like **Photo**,**Club logo**,**Player ID** were deleted.
- **Goalkeepers** :- The goalkeepers had a lot of missing values in their fields and we decided not to consider them for our analyses. Only outfield players were taken into consideration. These rows were deleted.
- **Missing values** :- There were some instances where the **market value** of a player was missing. In order to tackle this we actually regressed the top 10 most significant measurements with **market value** and predicted the missing value for a particular player. It was observed that our measurements were not too far off from their actual market value in the real world - a value that we obtained from **www.transfermarkt.com**.
- **Incompatible data types** :- The column containing market value of players in **Million Euros** had symbols which made the data type non-numeric and hence, unfit for analysis. So we applied various functions in **MS Excel** to obtain numeric data types for these.
- **Adjusting for Correlations** :- There were various measurement features that were highly correlated with each other. This was rather interesting since we tackled this issue using a balanced combination of our practical insights and software tools. We intuitively thought that traditionally, measurements like **ball control** and **Dribbling** should be correlated. This initial hypothesis was found to be true since upon plotting a scatterplot, there seemed to be a strong correlation. This was overcome by averaging out these corresponding values to get a holistic measure of **ball control**. A second phase of filtering out correlations and adjusting for them came through using a correlation matrix in **R** and averaging out measurements that had very strong correlations.

Features in our Dataset

After completing all the data cleaning, we were down to the following final features in our dataset :-

- **Age** :- Age of the player.
- **Value in Millions** :- Transfer Market Value of a player.
- **Aggression** :- [out of 100] Level of aggression in playing.
- **Acc** :- [out of 100] Level of acceleration.
- **Balance** :- [out of 100] A measurement of Balance.
- **Dribbling** :- [out of 100] A measurement of dribbling skills.

- **Composure** :- [out of 100] A measurement of Composure.
- **Crossing** :- [out of 100] Abiity to put crosses into the penalty area.
- **Heading** :- [out of 100] Ability to head the ball.
- **Passing** :- [out of 100] A measurement of passing skills.
- **Positioning** :- [out of 100] A measurement of good field positioning.
- **Tackling** :- [out of 100] A measurement of tackling skills.
- **Stamina** :- [out of 100] A measurement of endurance.
- **Strength** :- [out of 100] A measurement of strength.
- **DEF** :- [out of 100] Performance in defensive positions.
- **MIDFWD** :- [out of 100] Performance in midfield and forward positions.

Aim of the Analysis

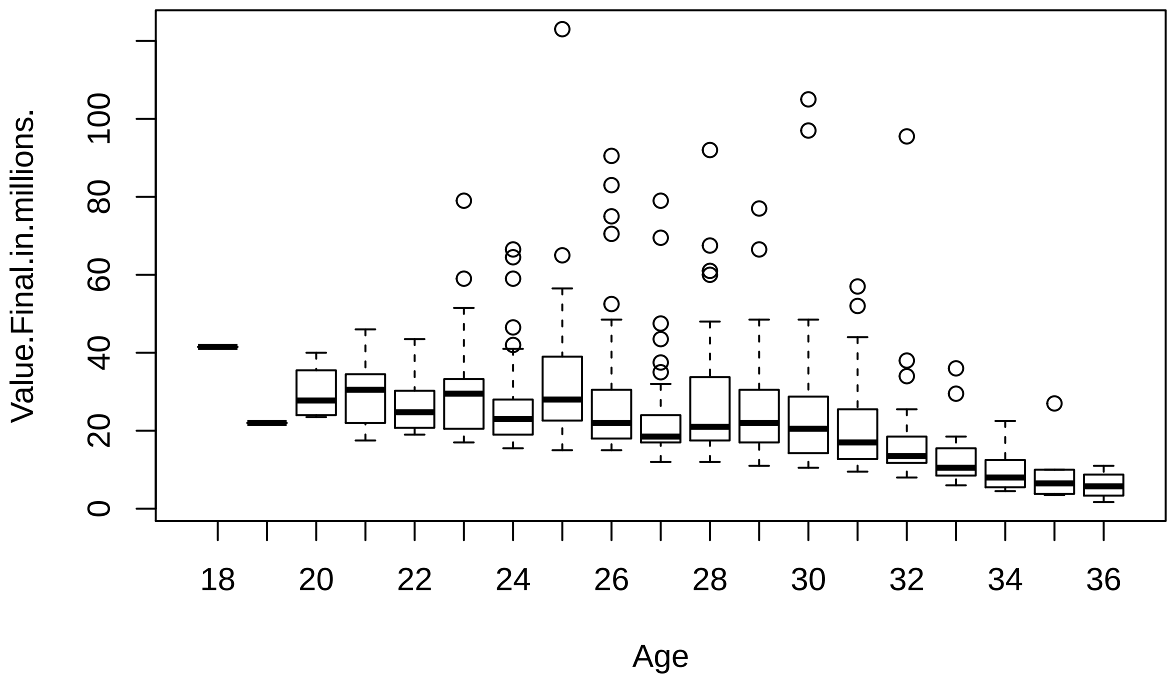
The primary purpose of this analysis is to use **unsupervised** statistical learning methods like **PCA**,**K-means Clustering** and **Hierarchical Clustering** to form easily interpretable visualization of the data and to observe clusters within the data. The purpose of the clusters would then be to suggest clubs with financial constraints to buy good players at a low cost based on cluster characteristics.

Exploring the Data

In this section we will view some of the data values and try to find statistical answers to some interesting questions. Firstly we were curious to know if a player's value on average, inceased or decreased as he aged ?

Hence from this plot of Value vs. Age, we can see that value actually declines as a player gets older and tends to peak when a player is in top form at around 25 years of age.

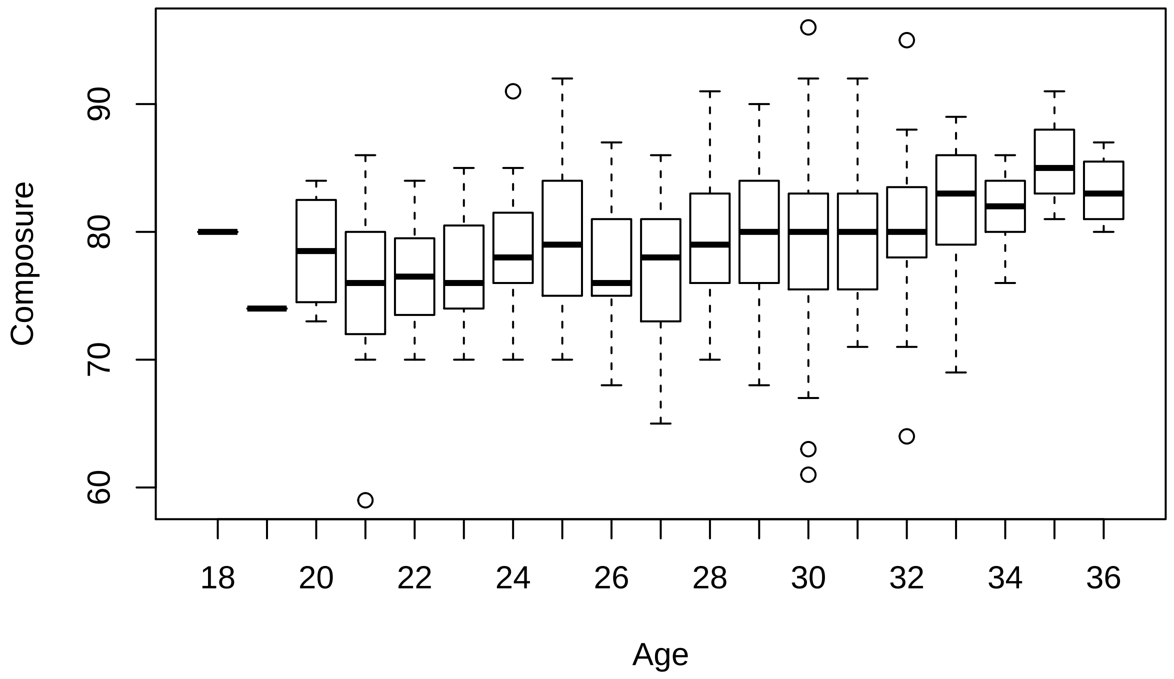
```
boxplot(Value.Final.in.millions.~Age, data = testing)
```



Another searching question is - Do older players tend to be more composed in general ? We find our intuition confirmed with these results.

On average the older players tend to be more composed.

```r
boxplot(Composure~Age,data = testing)
```



Finally, a view of our data :-

```r
head(testing)
```

```
##                 Name Age Value.Final.in.millions. Aggression  Acc Balance
## 1 Cristiano Ronaldo  32                     95.5         63 89.0      63
## 2          L. Messi  30                    105.0         48 91.0      95
## 3            Neymar  25                    123.0         56 95.0      82
## 4   L. Su\xcc\xc1rez  30                     97.0         78 87.0      60
## 5    R. Lewandowski  28                     92.0         80 78.5      80
## 6         E. Hazard  26                     90.5         54 93.0      91
##   Dribbling Composure Crossing Heading Passing Positioning Tackling
## 1      92.0        95     83.0    91.5    80.0          95     27.0
## 2      96.0        96     83.0    69.5    87.5          93     27.0
## 3      95.5        92     78.0    61.5    78.0          90     28.5
## 4      88.5        83     81.5    73.0    73.5          92     41.5
## 5      87.0        87     69.5    84.5    74.0          91     30.5
## 6      92.5        87     81.0    58.0    83.5          85     24.5
##   Stamina Strength  DEF MIDFWD
## 1      92       80 59.9   87.8
## 2      73       59 55.0   88.6
## 3      78       53 56.1   84.6
## 4      89       80 63.3   84.8
## 5      79       84 59.5   82.8
## 6      79       65 57.0   84.7
```

Principal Component Analysis

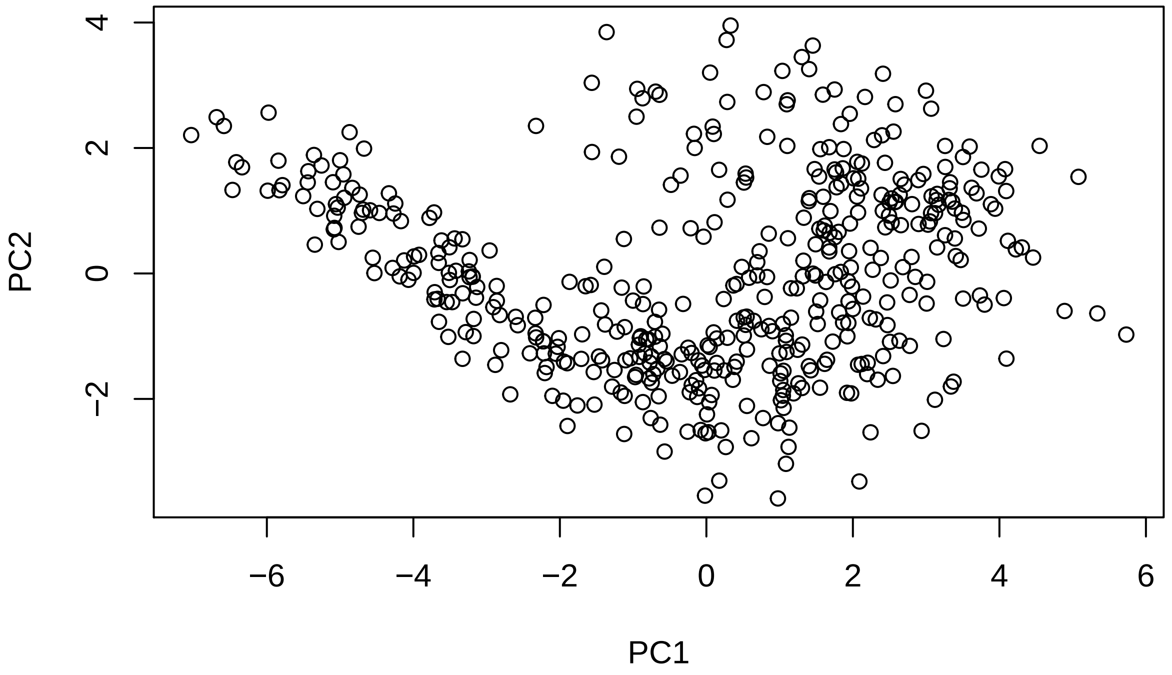
PCA is essentially a way of re-expressing our multi-dimensional complex data into an easily interpretable reduced dimensional form. It primarily does this by changing the basis vectors that express the data. **Principal components** are nothing but the orthogonal direction vectors that capture most amount of variation among the observations in the feature space. The re-expressed observations are known as **scores** and the new basis vectors are known as **loading vectors**. We start with showing you the top 4 principal components with appropriate loading values as per the different features. We can say that these 4 vectors explain most of the variance within our data as we will see further :-

```r
pr.f1 = prcomp(foot, scale = TRUE)
pr.f1$rotation[,1:4]
```

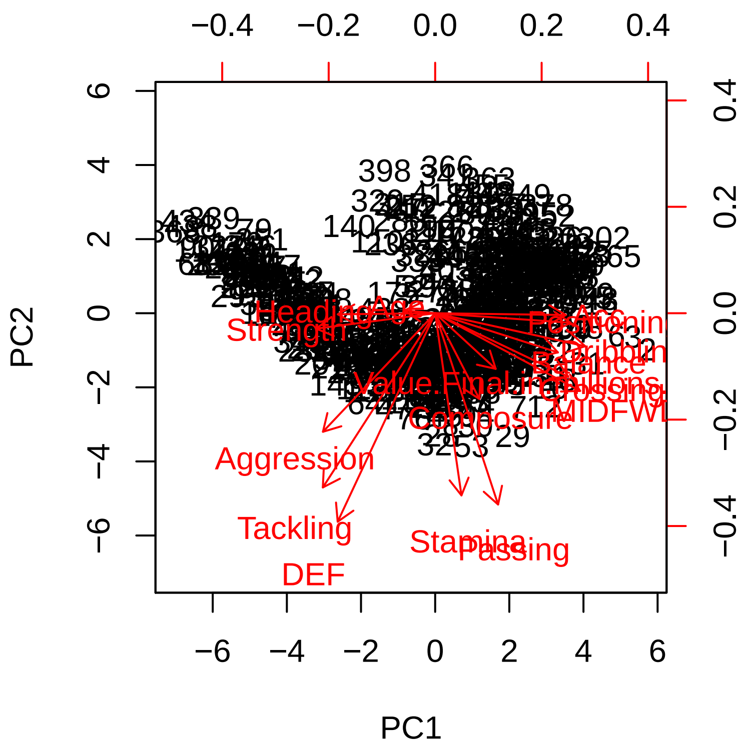
```
##                                  PC1          PC2          PC3         PC4
## Age                      -0.07195747  0.006485509  0.332566575 -0.58618274
## Value.Final.in.millions.  0.14205846 -0.130722438  0.394615492  0.38511192
## Aggression               -0.26348375 -0.278498584  0.077411356  0.09989977
## Acc                       0.30684608 -0.004224272 -0.159665657  0.20925634
## Balance                   0.28826285 -0.092180091 -0.215735967 -0.02447151
## Dribbling                 0.35227296 -0.077561230  0.063355743  0.01145627
## Composure                    0.10388050 -0.201394780  0.586542002 -0.16586270
## Crossing                     0.31314990 -0.149443157  0.009111425 -0.12340894
## Heading                     -0.22854903 -0.002529688  0.302302605  0.32777898
## Passing                      0.14777828 -0.449487310 -0.038806963 -0.31093020
## Positioning                  0.31825262 -0.022569174  0.190312185  0.08740460
## Tackling                    -0.26371470 -0.409348378 -0.178387644 -0.09382768
## Stamina                      0.06172359 -0.428056053 -0.134064264  0.38815646
## Strength                    -0.27929662 -0.036222629  0.306536460  0.18788554
## DEF                         -0.22888513 -0.489785080 -0.135608535 -0.06278418
## MIDFWD                       0.34045490 -0.182565213  0.127517541  0.01928762
```

Now we can present the plots of the principal components and see how the observations are spread out in the reduced dimensional space by taking the **top two principal components** :-

```
plot(pr.f1$x[,1:2],xlab="PC1",ylab="PC2")
```

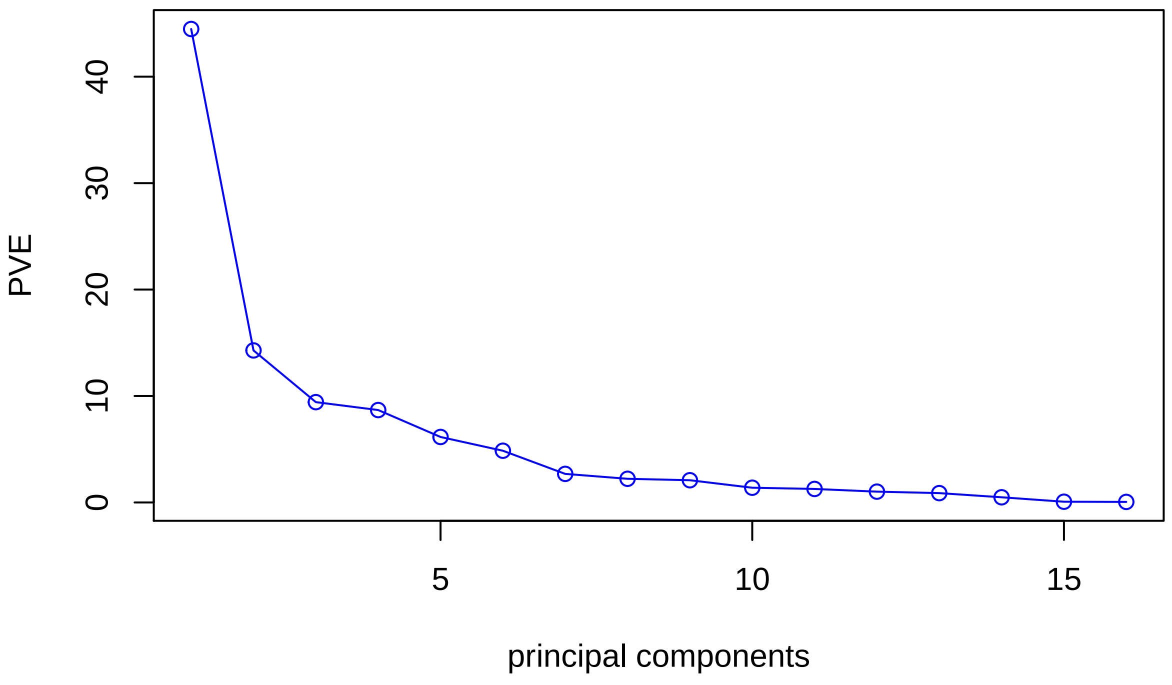


```
biplot(pr.f1, scale = 0)
```

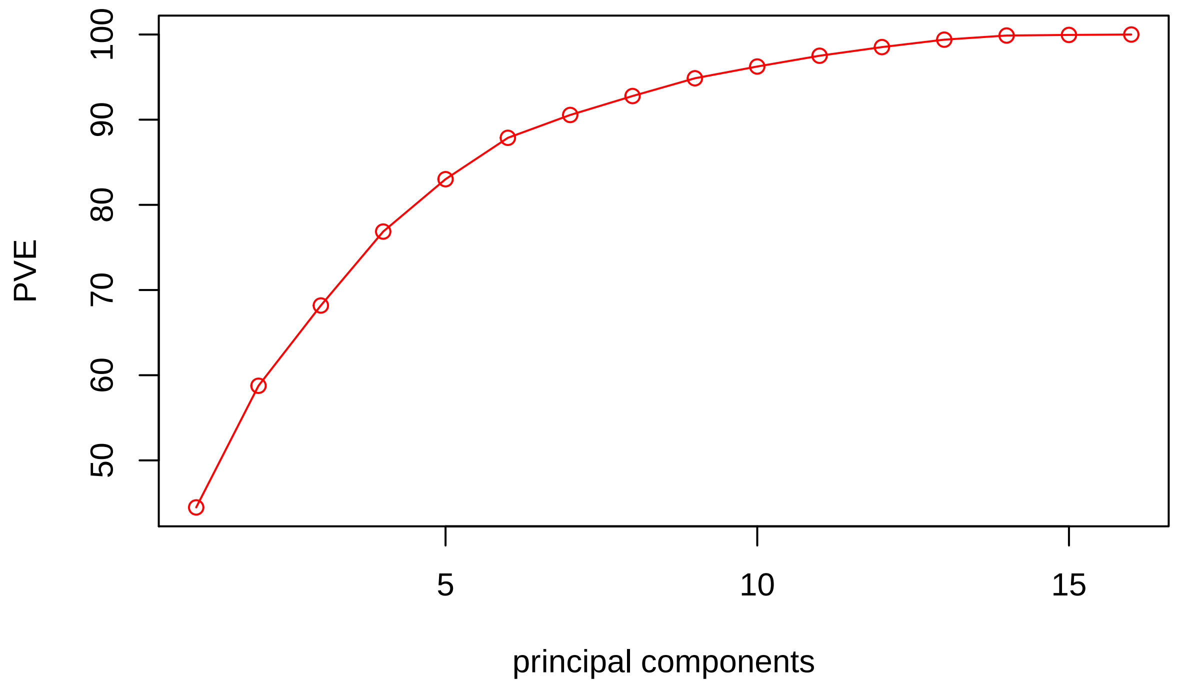


Finally we would like to see the **scree plots** telling us exactly how much **proportion of variance** is explained by each principal component.

```
pve = 100*pr.f1$sdev^2/sum(pr.f1$sdev^2)
plot(pve, type = "o", ylab="PVE",xlab="principal components",col='blue')
```



```
plot(cumsum(pve), type = "o", ylab="PVE",xlab="principal components",col='red')
```



```r
summary(pr.f1)$importance[3,]
```

```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
## 0.44480 0.58763 0.68186 0.76866 0.83016 0.87872 0.90555 0.92777 0.94862
##    PC10    PC11    PC12    PC13    PC14    PC15    PC16
## 0.96244 0.97510 0.98523 0.99398 0.99882 0.99950 1.00000
```

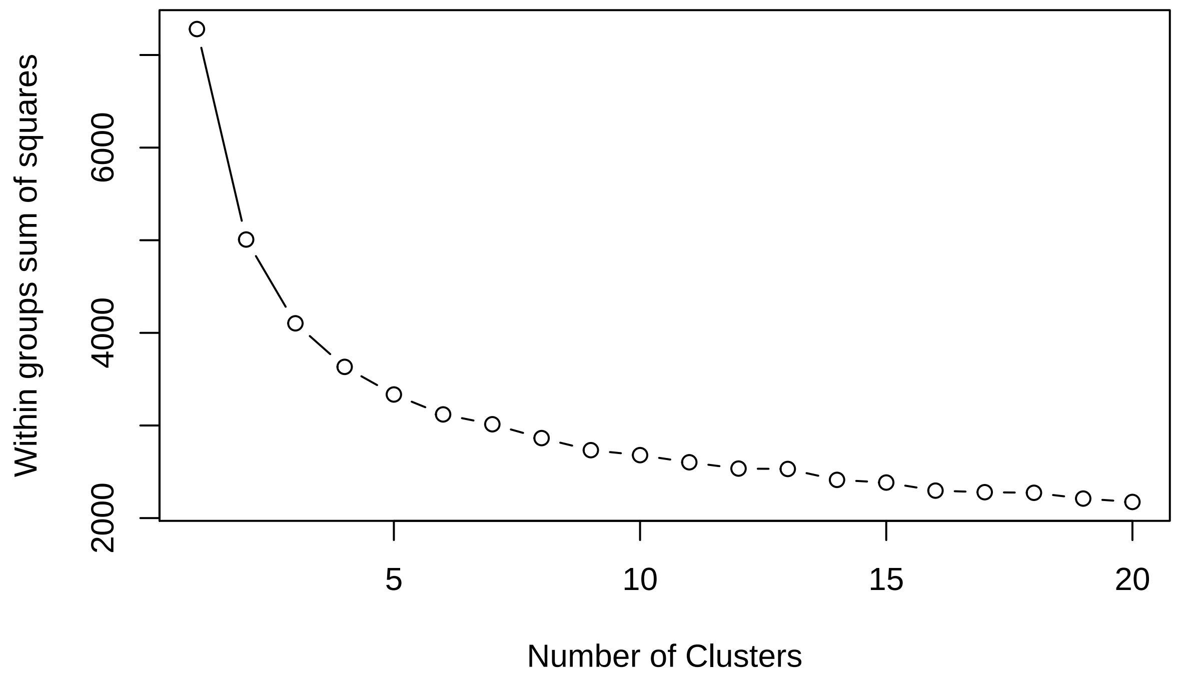
It is clear from the cumulative values that the first 7 principal components cumulatively explain almost 90 % of the variation in our data.

The PCA plot tells us that 4 broad clusters can be observed in the reduced dimensional re-expression of the data. Towards the extreme left side we can see that the observations clustered together have a high loading value for **Strength** and **Heading**. The bottom left cluster of observations tend to show a high loading for features like **Aggression** and **Tackling**. The bottom right cluster of points relate more to **Composure** and **Passing**. Finally the right most cluster relates to features like **Dribbling**, **MIDFWD** and **Positioning**. We can effectively use these groupings to form a team of players with different characteristics.

K-Means Clustering

This is a method by which clusters are formed in the data through an algorithm that initially assigns random pre-specified clusters to observations and then uses distance between the **centroid** and observations to further classify them. The Centroid is nothing but the vector of feature **means** and acts as the reference point from where distance of observations is judged. At first, we will use a scree plot to understand the optimal number of clusters giving us the least **within cluster sum of squares** value. It is given below :-

```r
wss <- (nrow(sd.data)-1)*sum(apply(sd.data,2,var))
for (i in 2:20) wss[i] <- sum(kmeans(sd.data, centers=i)$withinss)
plot(1:20, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")
```



From this figure we gather that the elbow is around 4 clusters.

So we will now specify a K value as 4 and run K-means on standardized data :-

```r
set.seed(2)
km.out = kmeans(sd.data,4,nstart=30)
km.clusters = km.out$cluster
km.clusters[1:10]
```

```
##  1  2  3  4  5  6  7  8  9 10
##  4  1  1  4  4  1  1  4  3  1
```

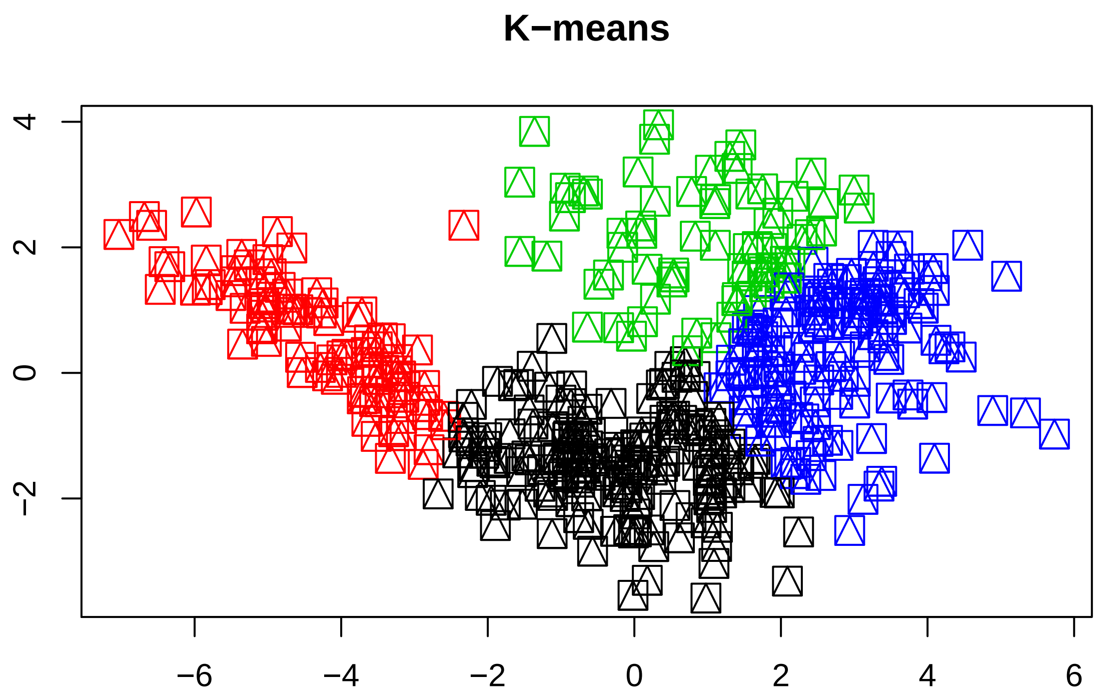
```r
table(km.clusters)
```

```
## km.clusters
##   1   2   3   4
## 140  88 158  70
```

The above code output indicates that the cluster that each serial number [player] belongs to. For example the player indexed by 1 (Ronaldo) belongs to cluster 4. The table indicates that the first cluster has 140 observations in it.

Now we will see how the **K-means** algorithm performs with observations as specified by the PCA space. What this essentially means is that we will attempt to apply the K-means algorithm on the observation space formed by the two most prominent principal components. We can notice from the plot below that **4 distinct clusters** are formed.

```
km.pc = kmeans(pr.f1$x[,1:2], 4, nstart = 30)
plot(pr.f1$x[,1:2], col = (km.pc$cluster), main = "K-means", xlab = "",ylab = "",pch=14,
```



```
km.pc$cluster[1:10]
```

```
##  1  2  3  4  5  6  7  8  9 10
##  4  4  4  4  4  4  1  3  1  4
```

Note: The cluster names in each iteration of the function may be different. For example observations corresponding to cluster (1) in one function might correspond to cluster (4) in some other. We will clear this apparently misleading point in the consistency comparisons section below.

K-means using non-standardized data is given below :-

```
set.seed(2)
km.non = kmeans(foot,4,nstart=30)
km.clon = km.non$cluster
km.clon[1:10]
```

```
##  1  2  3  4  5  6  7  8  9 10
##  4  1  1  4  4  1  3  4  3  1
```

Finally, we present **K-means** using **correlation distance** between observations. The correlation distance is basically a measure of the degree of correlation between observations. Sometimes it is observed that observations in the feature space might be quite close to each other in terms of Euclidean distance, but might actually have very different feature characteristics. In order to account for this, we use correlation distance between observations as a measure of dissimilarity.

```r
cord = as.dist(1-cor(t(foot)))
km.cord = kmeans(cord,4,nstart = 30)
km.cord$cluster[1:10]
```

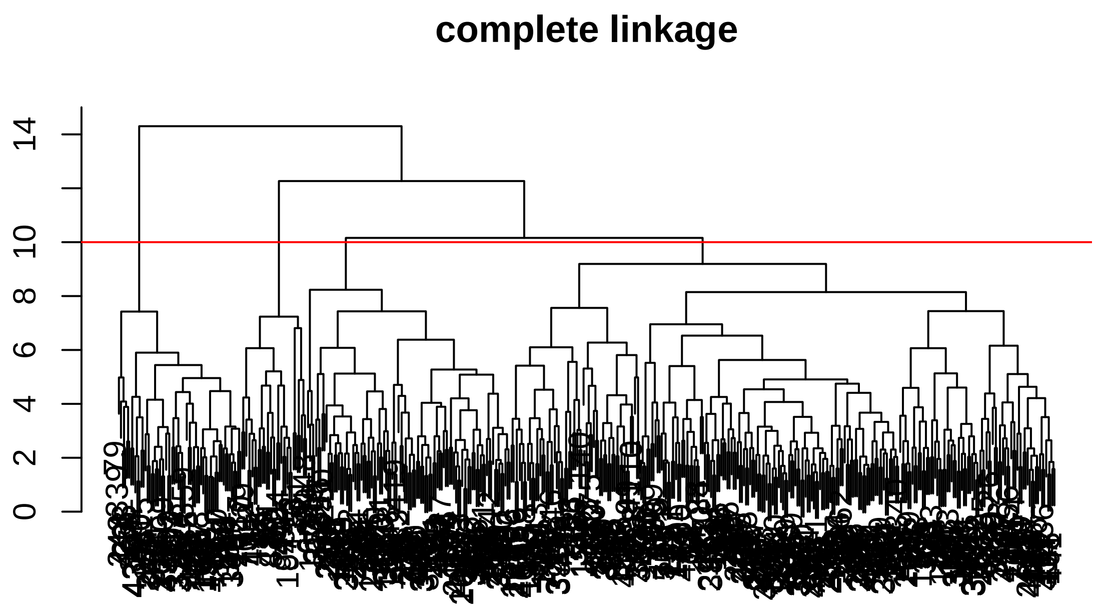
```
##  1  2  3  4  5  6  7  8  9 10
##  3  3  3  3  3  3  3  3  4  3
```

An interesting observation from the **K-means** outcome is that the clustering procedure gave somewhat consistent results for **standardized** and **non-standardized** data as we can even observe from the first 10 observations. K-means using PCA gave slightly different results but was largely in line with the previous two methods. However, we notice that the method using **correlation distance** as opposed to **Euclidean** turned out to be quite different. We are however inclined to believe that the results from standardized Euclidean measurements are better since they seem to be consistent with other approaches as well.

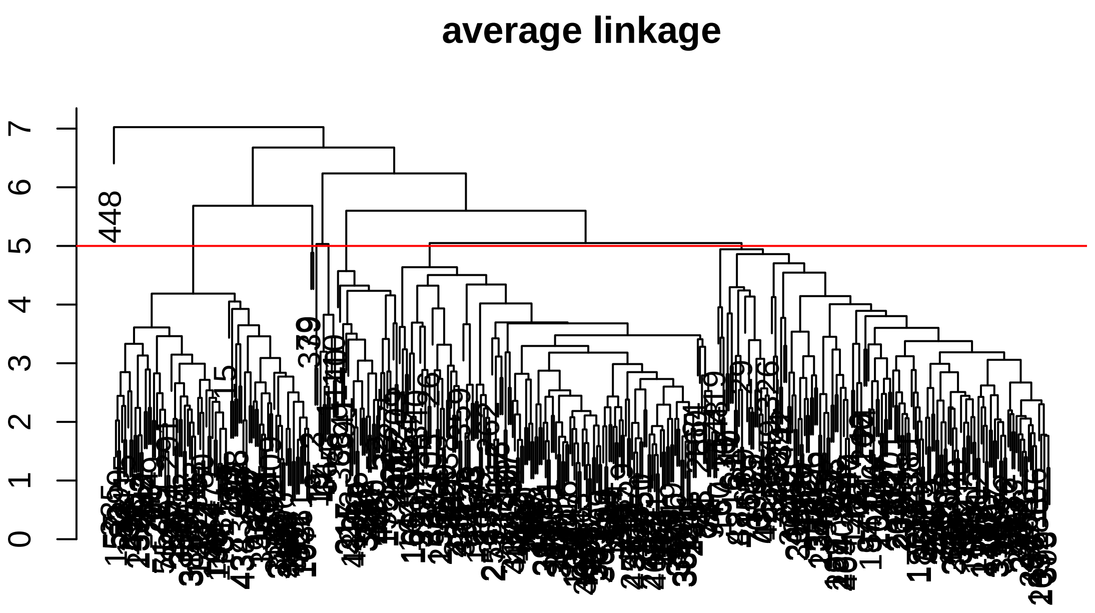
Hierarchical Clustering

This is a data clustering method where clusters are formed sequentially and in an inverted tree like fashion. The resulting cluster diagram is called a **Dendrogram** and the **cut** on the dendrogram determines the number of clusters in the data. Throughout our analyses further on, we will be sticking with the **complete** linkage function, however for the purpose of demonstration, even **average** linkage dendrogram is shown. There is another measure of linkage termed as the **single** method which is usually discouraged in such practical uses, so we have omitted its representation here.

```r
plot(hclust(data.dist), main = "complete linkage", xlab="",sub="",ylab="")
abline(h=10, col="red")
```



```
plot(hclust(data.dist, method="average"), main = "average linkage",
     xlab="",sub="",ylab="")
abline(h=5, col="red")
```



Dendrograms for Complete and Average linkages are shown above. The red line indicates the cut in the dendrograms that ends up forming the 4 clusters of our data.

Some of the results from this **H-clustering** using standardized data is shown :-

```
hc.out = hclust(dist(sd.data))
hc.clusters = cutree(hc.out, 4)
hc.clusters[1:10]
```

```
##  1  2  3  4  5  6  7  8  9 10
##  1  1  1  1  1  1  1  2  2  1
```

```
table(hc.clusters)
```

```
## hc.clusters
##   1   2   3   4
##  32 265  60  99
```

The above code output indicates that the cluster that each serial number [player] belongs to. For example the player indexed by 1 (Ronaldo) belongs to cluster 1. Also the table indicates that the cluster indexed by 1 has 32 observations in it and so forth.

Now we will use **H-Clustering** using the PCA space - the feature space formed by the top two principal loading vectors :-

```r
hc.pca2 = hclust(dist(pr.f1$x[,1:2]), method = "complete")
hc.pclust2 = cutree(hc.pca2, 4)
hc.pclust2[1:10]
```

```
##  1  2  3  4  5  6  7  8  9 10
##  1  1  1  2  2  1  2  3  2  1
```

H-Clustering using non-standardized data is shown along with the associated heatmap of various clusters and features associated with them :-

```r
dis2 = dist(foot)
hc.non = hclust(dist(foot))
hc.clon = cutree(hc.out, 4)
hc.clon[1:10]
```

```
##  1  2  3  4  5  6  7  8  9 10
##  1  1  1  1  1  1  1  2  2  1
```

```r
dend <- as.dendrogram(hc.non)
library(dendextend)
```

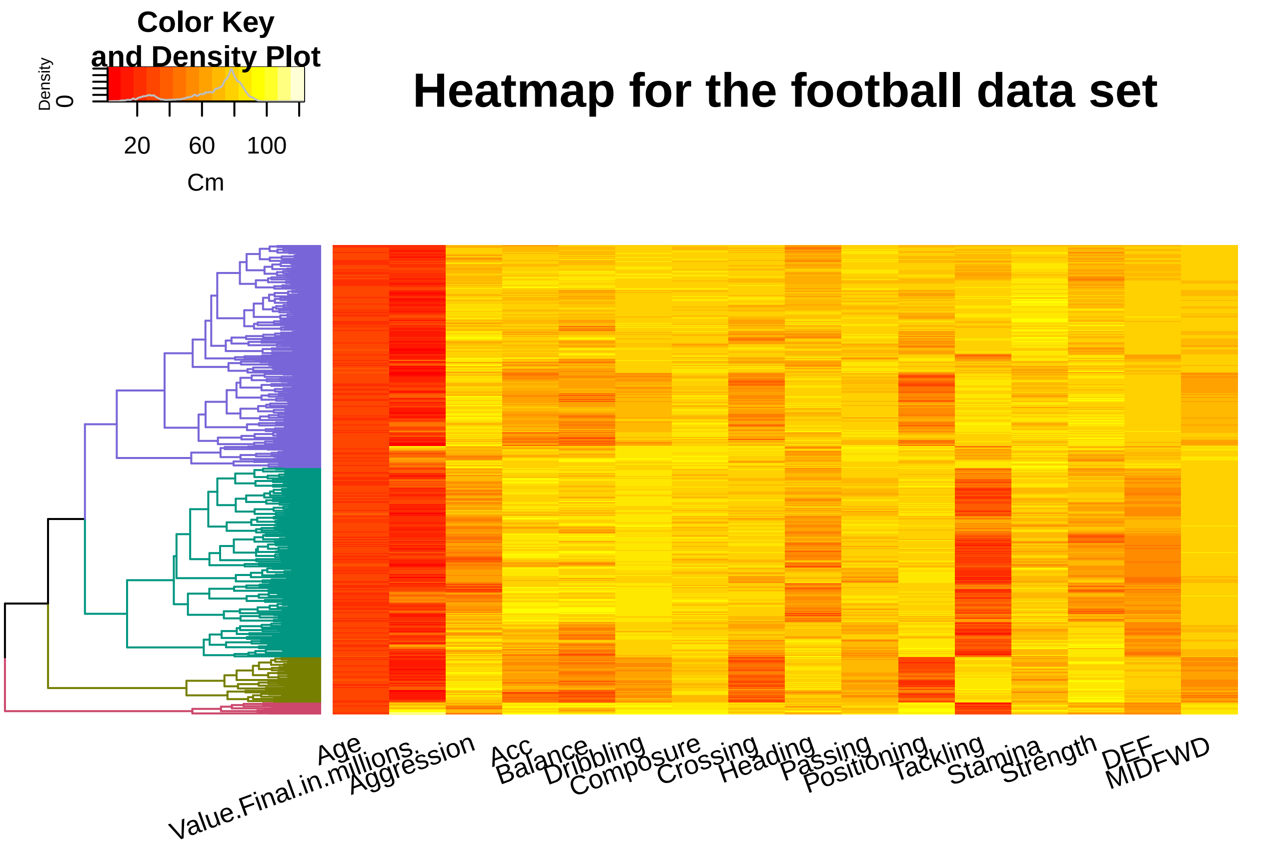
```
##
## ---------------------
## Welcome to dendextend version 1.13.4
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendext
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```

```r
dend <- color_branches(dend, k=4)
gplots::heatmap.2(as.matrix(foot),
          main = "Heatmap for the football data set",
          srtCol = 20,
          dendrogram = "row",
          Rowv = dend,
          Colv = "NA", # this to make sure the columns are not ordered
          trace="none",
          margins =c(5,0.1),
          key.xlab = "Cm",
          denscol = "grey",
          density.info = "density",

        )
```



This heat map tells us - what intensities of various features are associated with different clusers. For example we can see that high values for Tackling are associated with the cluster represented in Green.

Finally, **H-clustering** using **correlation distance** is computed :-

```
hc.cord = hclust(cord, method="complete")
hc.coclustr = cutree(hc.cord, 4)
hc.coclustr[1:10]
```

```
##  1  2  3  4  5  6  7  8  9 10
##  1  1  1  1  1  1  1  1  2  1
```

Even with **H-clustering** we notice, like the K-means, that the clusters formed are somewhat similar in the standardized and non-standarized form. However, now that we have ascertained the kind of clustering happening in hierarchical and k-means clustering, we will get a more concrete idea of the consistency in clustering when we compare the clustering across K-means and Hierarchical.

Consistency Comparisons

We will now compare the various cluster assignments among **K-means** and **H-cluster** outputs so as to determine consistency in clustering the observations. We would also attempt to select the pair of clustering models that have the least amount of relative misclassifications. We start with a tabular comparison of clusters with standardized data :-

```
table(km.clusters,hc.clusters)
```

```
##              hc.clusters
## km.clusters   1   2   3   4
##           1  25  22   0  93
##           2   0  28  60   0
##           3   3 155   0   0
##           4   4  60   0   6
```

To give you a sense of this table, we can see that cluster 4 of hierarchical corresponds to cluster 1 of k-means. This is because majority of the observations (93) are common in them. Although misclassifications exist, largely the clusters are consistent.

A tabular comparison on clustering using the PCA vector space :-

```
table(km.pc$cluster,hc.pclust2)
```

```
##    hc.pclust2
##       1   2   3   4
##   1   0 162   0   2
##   2   0   0   1  89
##   3   0   7  63   0
##   4  60  71   1   0
```

A tabular comparison on clustering using non-standardized data :-

```
table(km.clon,hc.clon)
```

```
##          hc.clon
## km.clon   1   2   3   4
##       1  17  16   0  90
##       2   0  31  60   0
##       3   7 155   0   5
##       4   8  63   0   4
```

A tabular comparison on clustering using correlation distance as a dissimilarity measure :-

```r
table(km.cord$cluster,hc.coclustr)
```

```
##    hc.coclustr
##       1   2   3   4
##   1   1   0  86  86
##   2   0 121   5  20
##   3  17   0   7  17
##   4   0  96   0   0
```

We can see from the above consistency comparison tables that when we compare the clusters obtained from K-means and Hierarchical using **correlation distance** as a dissimilarity measure, there is more misclassification across the clustering methods. On the other hand, standardized and non-standardized data clusters show the least cross-cluster misclassification even compared with that of the PCA space clustering. In the correlation clustering, misclassifications are observed in **2** clusters whereas the other methods show a certain degree of misclassification only in **1** cluster. We would finally settle with the **standardized** data results however, since they seem to be consistent with the Principal component analysis plots. Hence, we can conclude that there is the presence of 4 clusters in our data - 4 different kinds of players with various characteristics.

Inference

After a comprehensive analysis of various types of clustering methods applied using different parametric specifications with regard to **Linkage**, **Dissimilarity measures** and **standard/non-standard data**, we have found that for the most part, the Clusters seem to be largely consistent. This further implies that it is entirely possible that our models estimate the true clustering observed among different types of players. As an example, we notice that in one of the clustering functions, **Christiano Ronaldo** is assigned cluster **1** and so is **Adrien Silva**. What we can gather from this observation is that if there is an upcoming football club who wanted someone with the skill set of Ronaldo but couldn't possibly afford him, they could instead invest in a 'similar' player as per our clusters and buy that player at a much lower cost, whilst enjoying the benefits of a player with a skill set comparable to Ronaldo. ***Our heatmap can be particularly handy in this situation. As an example, if we want someone with high values in TACKLING we would notice the intense red regions of the heatmap corresponding to the appropriate cluster and select the best player with the least transfer value.***

Market Basket Analysis

This is a method used to figure out purchasing patterns of customers. Through this, it is possible to identify combinations of products that most frequently occur in retail transactions. The joint probability of a subset of items or **itemset** occuring simultaneously in a transaction order is termed as **Support**. A Bayes' theorem interpretation can be applied as - Given the fact someone bought [X,Y], what is the probability that they will also buy [Z] - This is essentially the concept of **confidence** of an **association rule**. Traditionally the **Apriori** algorithm is used for the purpose of this type of analysis.

Dataset for Market Basket Analysis

For the purpose of demonstration of this concept, we will be using the **Online Retail** dataset obtained from **UCI machine learning repository**. The dataset primarily contains transaction data for an online UK based firm. Following are the important features that we will manipulate and focus on :-

- **InvoiceNo** :- 6 digit invoice number attached to each transaction.
- **Description** :- Product item name.
- **InvoiceDate** :- Recorded date and time of the transaction.

Procedure for Data preparation

Before using the **Apriori** function on the dataset to analyse **association rules**, the data has to be cleaned and converted into a transactional data type. The following procedure was adopted to prepare the data :-

Using **complete.cases** function we filter out the missing values. We only take those rows that do not have missing values.

```r
#read excel into R dataframe
rdat1 <- read_excel('/Users/Akashgupta/Downloads/Online Retail.xlsx')
rdat1 <- rdat1[complete.cases(rdat1), ]     #rows with no missing vals
```

Time and **Date** are extracted from the **InvoiceDate** variable and stored in separate variables so that the products could be grouped together using just the data.

```r
rdat1$Date <- as.Date(rdat1$InvoiceDate)  ## stores date in new var
T_time<- format(rdat1$InvoiceDate,"%H:%M:%S") ## stores time in new var
InvoiceNo <- as.numeric(as.character(rdat1$InvoiceNo))
```

```
## Warning: NAs introduced by coercion
```

A glimpse of the data is provided :-

```r
glimpse(rdat1)
```

```
## Observations: 406,829
## Variables: 9
## $ InvoiceNo   <chr> "536365", "536365", "536365", "536365", "536365", ...
## $ StockCode   <chr> "85123A", "71053", "84406B", "84029G", "84029E", "...
## $ Description <chr> "WHITE HANGING HEART T-LIGHT HOLDER", "WHITE METAL...
## $ Quantity    <dbl> 6, 6, 8, 6, 6, 2, 6, 6, 6, 32, 6, 6, 8, 6, 6, 3, 2...
## $ InvoiceDate <dttm> 2010-12-01 08:26:00, 2010-12-01 08:26:00, 2010-12...
## $ UnitPrice   <dbl> 2.55, 3.39, 2.75, 3.39, 3.39, 7.65, 4.25, 1.85, 1....
## $ CustomerID  <dbl> 17850, 17850, 17850, 17850, 17850, 17850, 17850, 1...
## $ Country     <chr> "United Kingdom", "United Kingdom", "United Kingdo...
## $ Date        <date> 2010-12-01, 2010-12-01, 2010-12-01, 2010-12-01, 2...
```

Data is then grouped by **InvoiceNo** and **Date** such that comma separated lists of product names are created in each row.

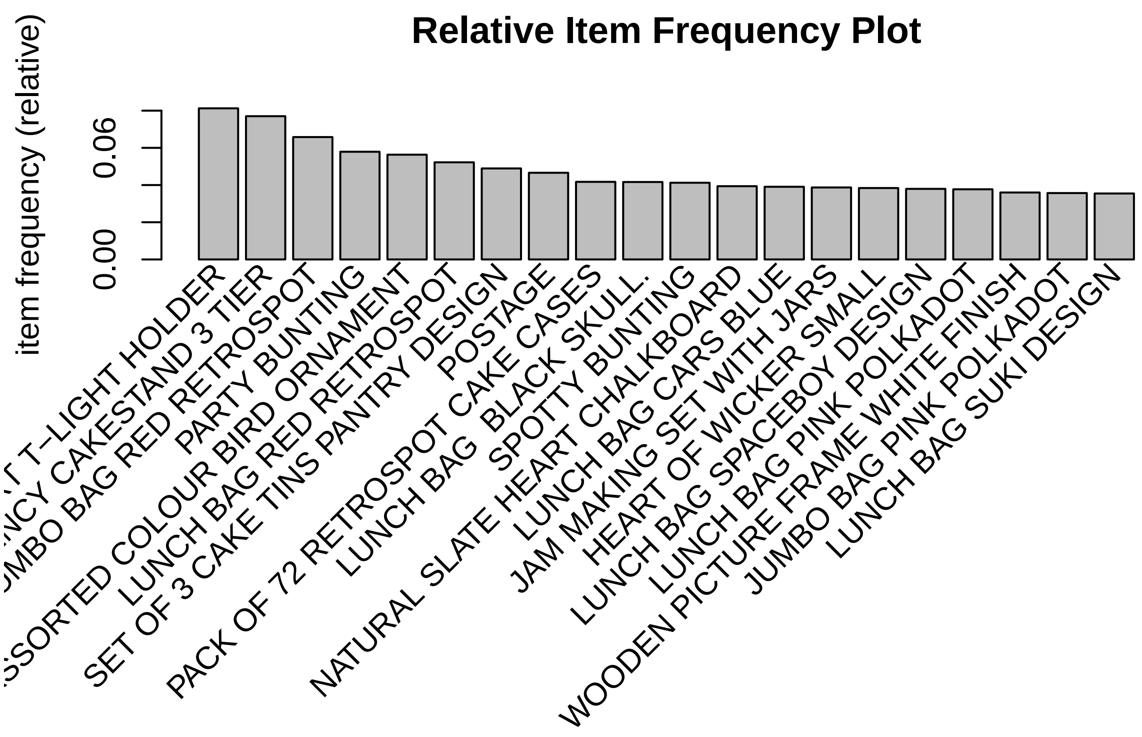
```r
# group data and separate product names by comma
transactionData <- ddply(rdat1,c("InvoiceNo","Date"),
                         function(df1)paste(df1$Description,
                                            collapse = ","))
#set column InvoiceNo of dataframe transactionData
transactionData$InvoiceNo <- NULL
#set column Date of dataframe transactionData
transactionData$Date <- NULL
#Rename column to items
colnames(transactionData) <- c("items")
```

The transformed data is re-read in the software in the **transactional database** format such that it is compatible for the **Apriori** functions.

```r
write.csv(transactionData,"/Users/Akashgupta/Downloads/mba1.csv", quote = FALSE, row.nam
```

Finally, before we begin showing you the relative frequency plot of various items is presented :

```r
itemFrequencyPlot(transaction_new,topN=20,type="relative",main="Relative Item Frequency
```



The above relative frequency plot tells us information like - LUNCH BAG SUKI DESIGN has been bought least number of times in transactions compared to other products.

Analysing the Association Rules

Now we will use the apriori function to mine association rules with a support of **0.002**, confidence of **0.8** and a maximum length of **10** products in a transaction. The summary tells us that upon going through multiple combinations of association rules, the function has found **4272** rules that fulfil the criterion of being above the specified thresholds for **support** and **confidence**. The results are shown :-

```
summary(association.rules)
```

```
## set of 4272 rules
##
## rule length distribution (lhs + rhs):sizes
##    2    3    4    5    6    7    8
##   89  448 1086 1225 1100  316    8
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   4.000   5.000   4.885   6.000   8.000
##
## summary of quality measures:
##     support           confidence          lift              count
##  Min.   :0.002028   Min.   :0.8000   Min.   :  9.933   Min.   : 45.00
##  1st Qu.:0.002163   1st Qu.:0.8305   1st Qu.: 21.730   1st Qu.: 48.00
##  Median :0.002388   Median :0.8644   Median : 25.505   Median : 53.00
##  Mean   :0.002740   Mean   :0.8716   Mean   : 47.176   Mean   : 60.79
##  3rd Qu.:0.002839   3rd Qu.:0.9038   3rd Qu.: 69.347   3rd Qu.: 63.00
##  Max.   :0.015997   Max.   :1.0000   Max.   :443.820   Max.   :355.00
##
## mining info:
##             data ntransactions support confidence
##  transaction_new         22191   0.002        0.8
```

Some of the **association rules** discovered are shown below :-

```
inspect(association.rules[1:10])
```

```
##      lhs                                   rhs
## [1]  {BLACK TEA}                        => {SUGAR JARS}                   0.00
## [2]  {BLACK TEA}                        => {COFFEE}                       0.00
## [3]  {FRENCH BLUE METAL DOOR SIGN 0}    => {FRENCH BLUE METAL DOOR SIGN 2}     0.00
## [4]  {FRENCH BLUE METAL DOOR SIGN 0}    => {FRENCH BLUE METAL DOOR SIGN 1}     0.00
## [5]  {WHITE TEA}                        => {SUGAR JARS}                   0.00
## [6]  {WHITE TEA}                        => {COFFEE}                       0.00
## [7]  {VINTAGE RED ENAMEL TRIM PLATE}      => {VINTAGE RED TRIM ENAMEL BOWL}      0.00
## [8]  {GREEN 3 PIECE POLKADOT CUTLERY SET} => {RED 3 PIECE RETROSPOT CUTLERY SET} 0.00
## [9]  {NURSERY A}                          => {B}                                 0.00
## [10] {B}                                  => {NURSERY A}                         0.00
```

Lastly, we will specifically look for the associations that might interest us. For example, we might want to find - what caused someone to buy the product COFFEE. We can analyse this specific association rule below :-

```
inspect(head(coffee.association.rules))
```

```
##     lhs                        rhs      support     confidence lift
## [1] {BLACK TEA}             => {COFFEE} 0.002072912 1          69.34687
## [2] {WHITE TEA}             => {COFFEE} 0.002929115 1          69.34687
## [3] {SUGAR JARS}            => {COFFEE} 0.004190888 1          69.34687
## [4] {SUGAR}                 => {COFFEE} 0.010409626 1          69.34687
## [5] {SET 3 RETROSPOT TEA}   => {COFFEE} 0.010409626 1          69.34687
## [6] {BLACK TEA,SUGAR JARS} => {COFFEE} 0.002072912 1          69.34687
##     count
## [1]  46
## [2]  65
## [3]  93
## [4] 231
## [5] 231
## [6]  46
```

This can be interpreted as - It is seen that customers who tend to purchase items like **SUGAR**, **WHITE TEA** and **SUGAR JARS**, are absolutely certain to buy **COFFEE** as well. The online store can use this information by targeting coffee advertisements to such people or they could even attempt to club 'coffee' as a complimentary product with certain other products, at a discount so as to increase sales. In a similar fashion, many association rules can be exploited in order to craft effective marketing campaigns.

A Final Word

Using unsupervised learning methods like - PCA, K-means and Hierarchical clustering - our analysis on the FIFA dataset primarily involved reduced dimension data visualization and observing clusters in the data using various parameters for Linkage and dissimilarity measures. We also looked at Market Basket Analysis which is essentially a way to observe association rules among itemsets, using the Apriori algorithm.