K-means Clustering of Premier League Teams 16/17

Clare Clingain

May 1, 2018

Introduction

Sports analysts and fans alike try to decipher what sets England's top teams apart from each other. What makes the elusive top 4/6 so far away from the rest? What about the relegation-threatened teams? Using k-means unsupervised learning, we will attempt to cluster teams based on season-total statistics from the 2016-2017 season to determine whether there are ways in which we can separate the top-performing teams from the rest of the pack. Similarly, we will try to see which teams are squeezing their way into the top 4/6.

Data Preparation

Using publicly available data we cleaned and manipulated, we created a new data set *EPL_Aggregate*, which can be found here, that contains the summary statistics for the 20 Premier League teams from 2000/2001 to 2017/2018 seasons. Twelve different stats were recorded over the course of each season for each team: Yellow Cards, Red Cards, Goals Scored, Goals Against, Fouls Committed, Fouls Against, Corners Won, Corners Conceded, Shots, Shots Conceded, Shots on Target, Shots on Target Conceded. For this analysis, we summed each of the twelve stats for each team to get an idea of the "total" performance of a team over the course of the 2016/2017 season.

Teams for 16/17: Arsenal, Bournemouth, Burnley, Chelsea, Crystal Palace, Everton, Hull, Leicester City, Liverpool, Manchester City, Manchester United, Middlesbrough, Southampton, Stoke, Sunderland, Swansea, Tottenham, Watford, West Bromwich Albion, West Ham United.

```
EPL_Cluster <- EPL_aggregate %>% filter(Season=="16/17") %>% group_by(Team) %>%
  mutate(YellowCardSum = sum(Yellow_Cards))

EPL_Cluster <- EPL_Cluster %>% mutate(RedCardSum = sum(Red_Cards)) %>%
  mutate(GoalsSum = sum(Goals_Scored)) %>%
  mutate(GoalsAgainstSum = sum(Goals_Conceded)) %>%
  mutate(FoulsSum = sum(Fouls_Committed)) %>%
  mutate(FoulsAgainstSum = sum(Fouls_Against)) %>%
  mutate(CornersSum = sum(Corners)) %>%
  mutate(CornersConcededSum = sum(Corners_Conceded)) %>%
  mutate(ShotsSum = sum(Shots_Taken)) %>%
  mutate(ShotsTargetSum = sum(Shots_on_Target)) %>%
  mutate(ShotsCondededSum = sum(Shots_Conceded)) %>%
  mutate(ShotsTargetConSum = sum(Shots_on_Target_Conceded))
EPL_Cluster <- EPL_Cluster %>% filter(Week==38)
EPL_Cluster <- EPL_Cluster[c(1:4, 31:42)]
```

How many clusters?

There are a couple packages out there that do a good job of determining the number of clusters that might exist in a given data set. The Mclust package provides BIC, classification, uncertainty, and density for model-based clustering using an EM algorithm.

```
mcl <- Mclust(EPL_Cluster[c(5:16)])
summary(mcl)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 2 components:
##
##  log.likelihood  n  df       BIC       ICL
##       -824.6405 20 170 -2158.555 -2158.555
##
## Clustering table:
##  1  2
## 15  5
```

The results from Mclust reveal that the best model for our data is a 2-cluster VEV solution. The suggested clustering split has 15 teams in Cluster 1 and 5 teams in Cluster 2. However, this portion of the results from Mclust should be taken with caution. K-means may very well cluster teams in a different way.

```
k.means <- NbClust(EPL_Cluster[c(5:16)],method='kmeans',index='ch')
k.means$Best.nc
```

```
## Number_clusters     Value_Index
##          2.0000         36.5953
```

```
table(k.means$Best.partition)
```

```
##
##  1  2
##  8 12
```

NbClust provides a similar analysis for determining the optimal number of clusters in a data set. The maximum index is selected. We chose to use the Calinski and Harabasz (1974) index with method as k-means. Similar to Mclust, NbClust suggests a 2-cluster solution. Yet the proposed clusters in this solution are more equal in size, with 8 teams in Cluster 1 and 12 teams in Cluster 2.

Comparing the two packages' solutions, we see that Mclust classifies more teams in Cluster 1 than NbClust.

```
table(mcl$classification,k.means$Best.partition)
```

```
##
##      1  2
##   1  3 12
##   2  5  0
```

Given that both packages suggested a two-cluster solution, we will carry out k-means clustering with 2 clusters.

K-means: 2 clusters

The analysis was run on all 12 variables in our data set. A seed was set for reproducibility purposes.

We also used a written function to calculate the *C(g)* for model comparison. The *C(g)* is equivalent to the Calinki & Harabasz (1974) criterion.

$C(g) = \frac{trace(Between)/g-1}{trace(Within)/n-g}$ where g is the number of clusters.

The goal is to maximize *C(g)*.

```
set.seed(2011)
km.2 <- kmeans(EPL_Cluster[c(5:16)], 2, nstart= 100)

c.crit <- function(km.obj) {
  #based on k-means, for convenience due to amt of addl info in the km result object.
  #cd be generalized.
  sizes <- km.obj$size
  n <- sum(sizes)
  g <- length(sizes)
  msW<-sum(km.obj$withinss)/(n-g)
  overall.mean <- apply(km.obj$centers*km.obj$size,2,sum)/sum(km.obj$size)
  msB<-sum(km.obj$size*(t(t(km.obj$centers)-overall.mean))^2)/(g-1)
  list(msB=msB,msW=msW,C.g=msB/msW)
}
c.crit(km.2)$C.g
```

```
## [1] 36.59531
```

The *C(g)* for a 2-cluster solution is 36.595.

K-means: 3 clusters

Although the results from Mclust and NbClust are trustworthy, it's safer to run an addtional k-means model to ensure that 2 clusters is the best solution given our data.

```
set.seed(2011)
km.3 <- kmeans(EPL_Cluster[c(5:16)], 3, nstart= 100)
c.crit(km.3)$C.g
```

```
## [1] 27.32056
```

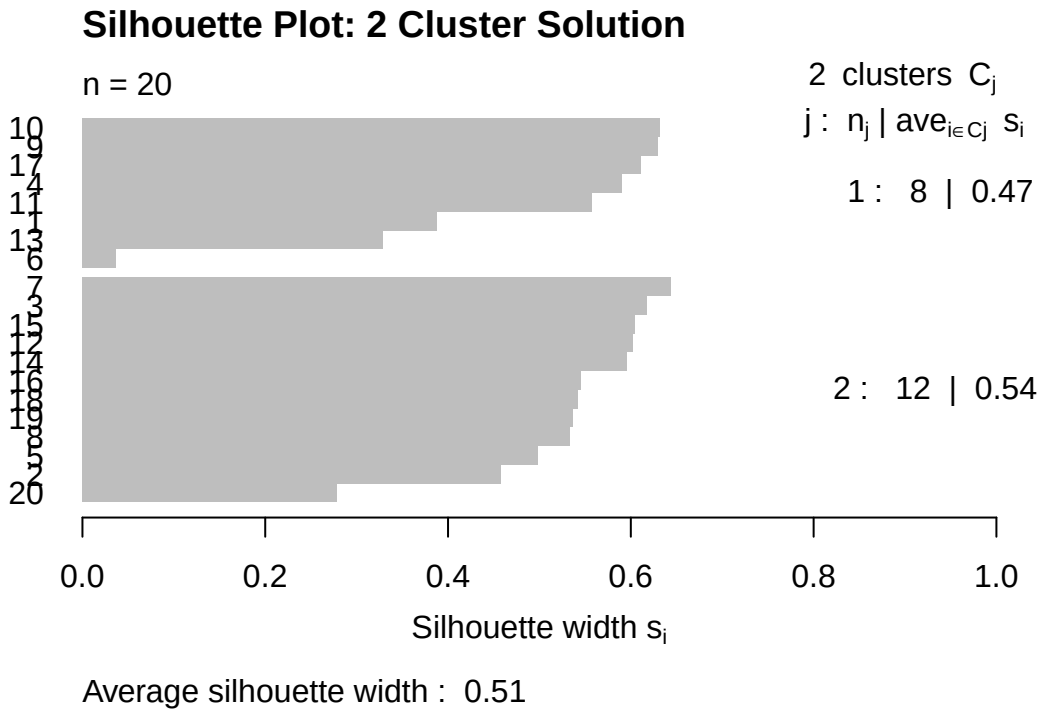
The *C(g)* for a 3-cluster solution is worse than that for a 2-cluster solution, which suggests that we should stick with only 2 clusters.

Exploring the 2-cluster Solution

Silhouette Plot

First, we examine the silhouette plot of the clusters.

```
plot(silhouette(km.2$clust, dist(EPL_Cluster[5:16])), main = "Silhouette Plot: 2 Cluster Solution")
```



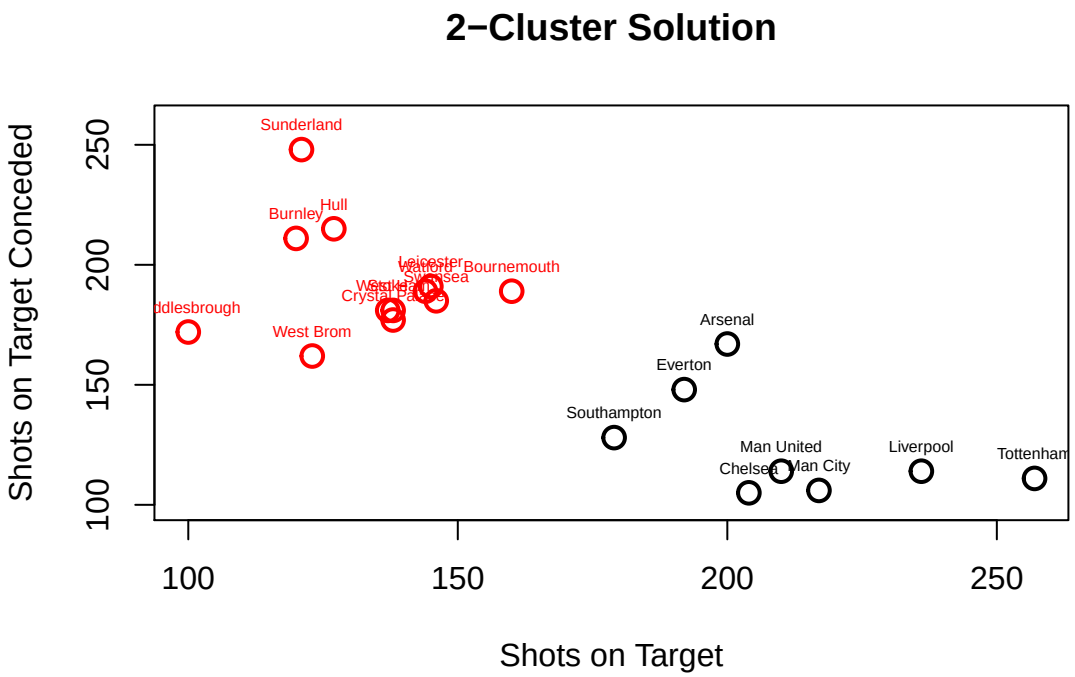
The average silhouette width is 0.51, with Cluster 1 having a width of 0.47 and Cluster 2 having a width of 0.54. These widths suggest that our clusters are separated enough from each other such that the data points are well clustered.

It is important to note that Everton (Team 6) has a rather small silhouette width. Since it's width is not negative, which would suggest a mismatch, we will continue on with the analysis.

Plotting on Two Dimensions

Next, we will plot the clusters on two dimensions: Shots on Target and Shots on Target Conceded. These two dimensions were chosen since they appear to spearate the data best out of all the pairs plots.

```
#Separate by ShotsTargetSum and ShotsTargetConSum
plot(EPL_Cluster[,c(14,16)],col=km.2$clust,pch=1,cex=1.5, lwd=2,
     xlab = "Shots on Target",ylab = "Shots on Target Conceded",
     main = "2-Cluster Solution",ylim =c(100,260))
text(EPL_Cluster[,c(14,16)],labels=EPL_Cluster$Team,col=km.2$clust, cex=.5,pos=3)
```

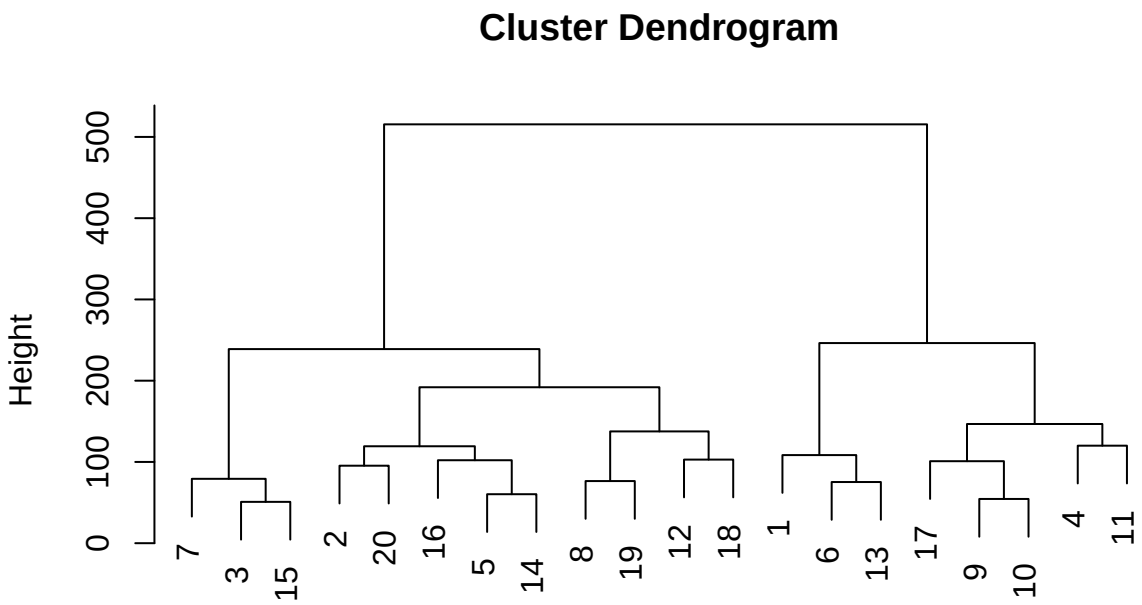


Here we see two pretty well separated clusters. Most noteably, Cluster 2 contains the typical top4/6 teams - Liverpool, Chelsea, Manchester City, Tottenham, Manchester United, Arsenal - but also houses Southampton and Everton.

Comparison to Hierarchical Clustering

For sanity, we compare the two cluster k-means solution with a complete hierarchical clustering solution. The cluster dendrogram shows two clear clusters.

```
plot(hclust(dist(EPL_Cluster[,-(1:4)]),meth='complete'))
```




```
comp.2 <- cutree(hclust(dist(EPL_Cluster[,-(1:4)]),meth='complete'),2)
```

When we compare the k-means results to the hierarchical clustering results, we see they match perfectly.

```
xtabs(~comp.2+km.2$cluster)
```

```
##       km.2$cluster
## comp.2  1  2
##      1  8  0
##      2  0 12
```

For procedure's sake, we also check the Rand Index between the two solutions. We find a perfect match.

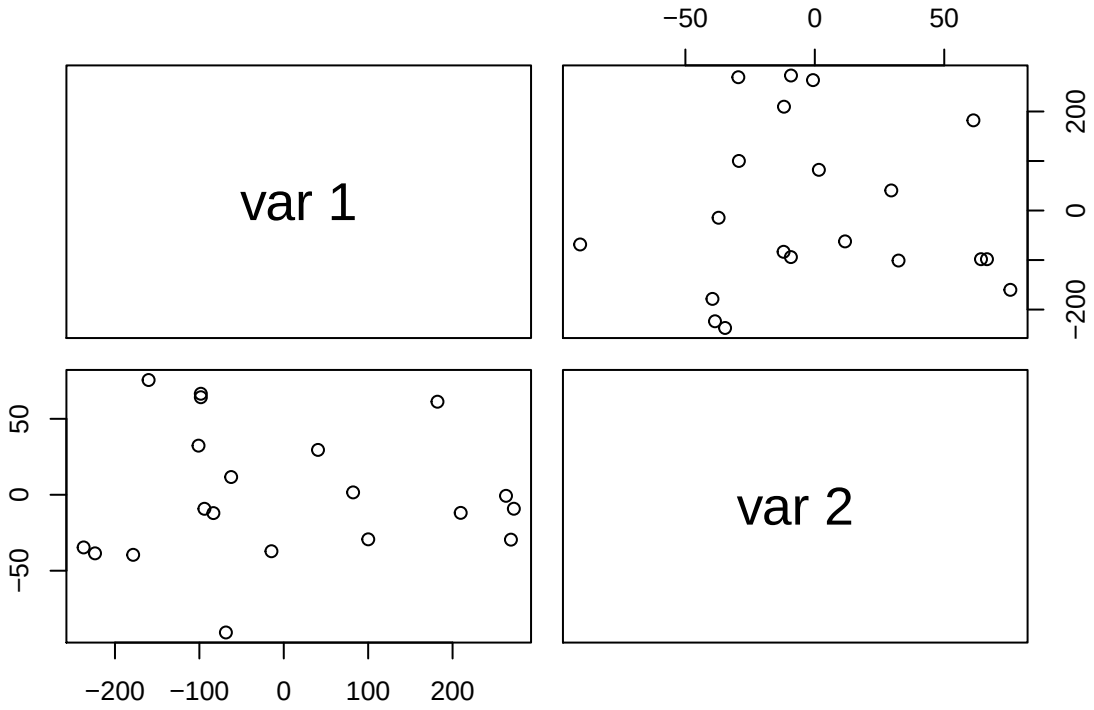
```
RRand(km.2$clust, comp.2)
```

```
##    Rand adjRand  Eindex
##  1.0000  1.0000  0.4525
```

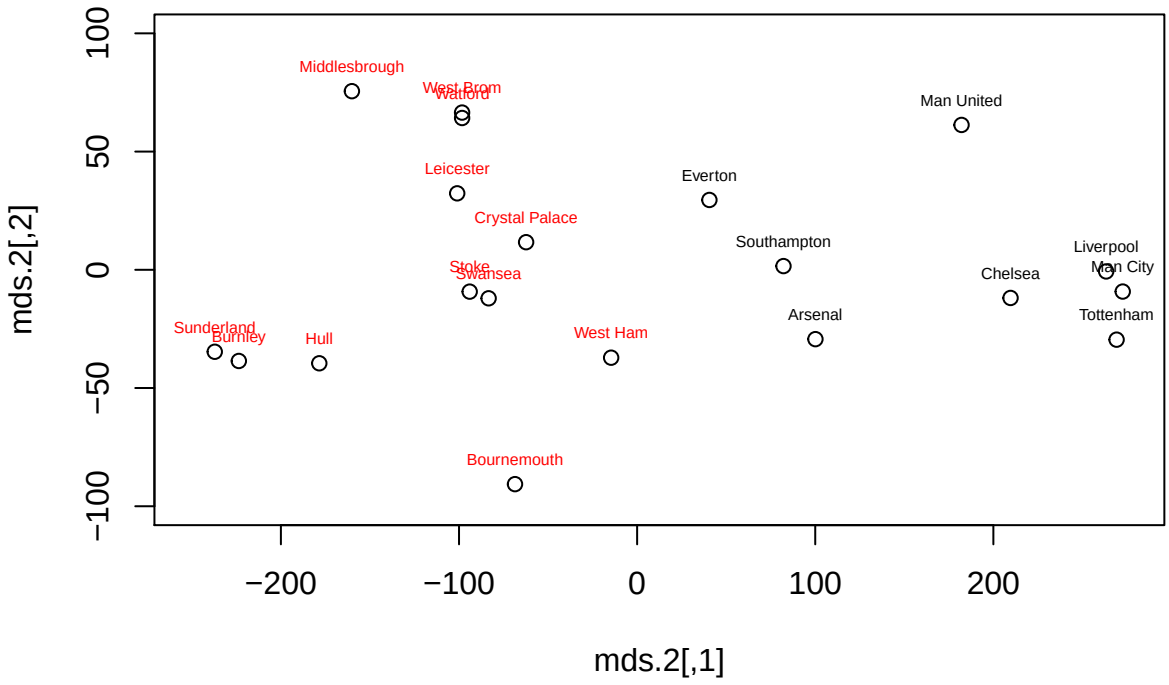
Comparison to Multidimensional Scaling (MDS)

There are 12 possible dimensions which we can use to separate the 20 premier league teams from 16/17. In an effort to reduce the dimensionality, MDS with 2 dimensions was used. We end up with a similar plot to the k-means results, but with less separation between the two clusters.

```
mds.2 <-cmdscale(dist(EPL_Cluster[,-(1:4)]),k=2)
pairs(mds.2)
```




```
plot(mds.2, ylim = c(-100,100), xlim = c(-250,275))
text(mds.2, labels = EPL_Cluster$Team, col = km.2$clust, cex=.5,pos=3)
```



This solution is not as satisfying as the k-means solution, but still suggests that there exists such a distance between the top4/6 teams and the rest. In particular, we see that relegated Sunderland and Hull are the farthest from the top4/6 teams on the first dimension. In a way, we can argue that the first dimension may primarily be attacking. Similarly, Manchester United and Arsenal are some distance away from Liverpool, Manchester City, Tottenham, and Chelsea, all of whom finished in the top 4.

Principal Components Analysis (PCA)

```
EPL_Cluster_std<- EPL_Cluster;
EPL_Cluster_std[,-(1:4)] <- scale(EPL_Cluster[,-(1:4)])
pca<- princomp(EPL_Cluster_std[,-(1:4)])
summary(pca)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
## Standard deviation     2.6145159 1.2796296 1.0337303 0.88650610 0.57293188
## Proportion of Variance 0.5996222 0.1436361 0.0937367 0.06893799 0.02879394
## Cumulative Proportion  0.5996222 0.7432584 0.8369951 0.90593306 0.93472701
##                            Comp.6     Comp.7     Comp.8      Comp.9
## Standard deviation     0.52928603 0.49664217 0.30359967 0.263647164
## Proportion of Variance 0.02457401 0.02163627 0.00808533 0.006097353
## Cumulative Proportion  0.95930102 0.98093728 0.98902261 0.995119966
##                            Comp.10     Comp.11      Comp.12
## Standard deviation     0.189060965 0.112565069 0.0849555289
```

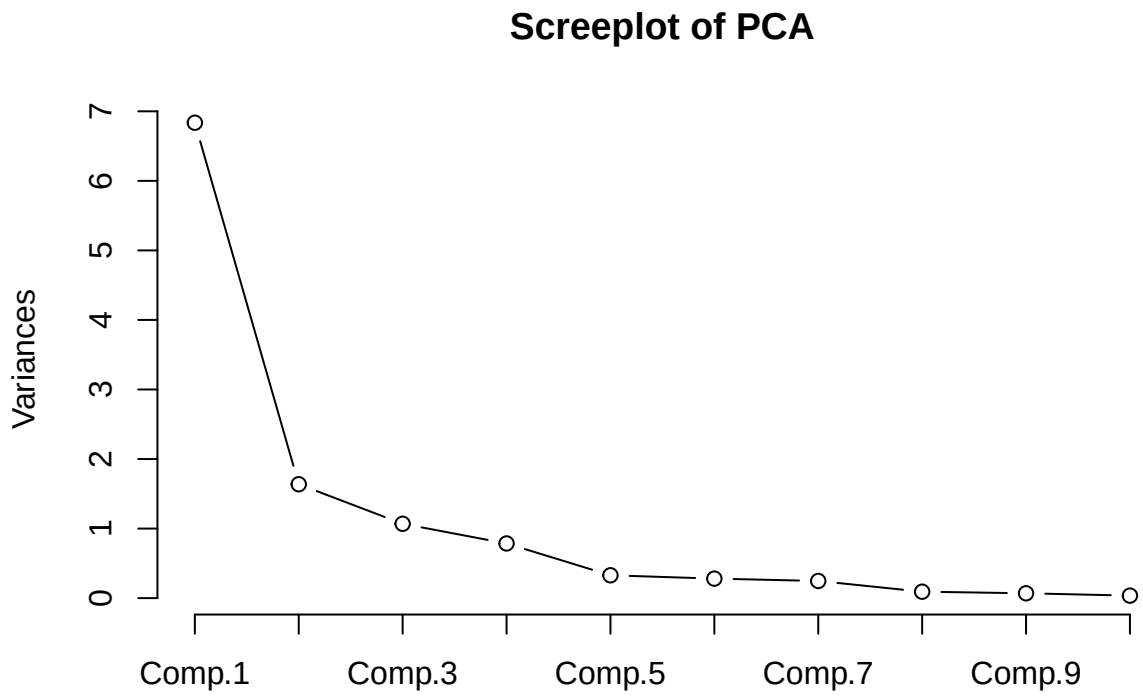
```
## Proportion of Variance 0.003135443 0.001111482 0.0006331089
## Cumulative Proportion  0.998255409 0.999366891 1.0000000000
```

It seems that 3 or 4 components should be enough to explain the variance in our data.

Screeplot

A screeplot was produced to determine an appropriate number of principal components.

```
screeplot(pca,type="l", main = "Screeplot of PCA")
```



The "elbow" appears to be around 3.

Eigenvalues

Next, we examine the eigenvalues to solidify our decision to use 3 principal components.

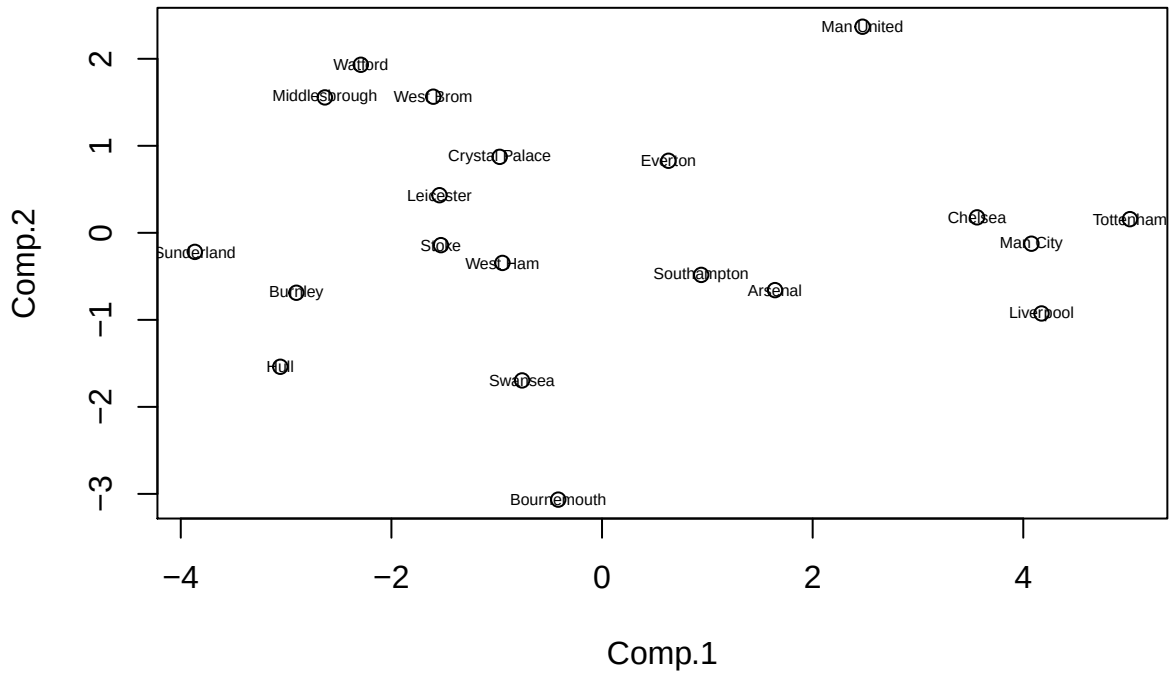
```
eigen(cor(EPL_Cluster_std[,-(1:4)]))$values
```

```
##  [1] 7.195466892 1.723633640 1.124840388 0.827255857 0.345527302
##  [6] 0.294888111 0.259635210 0.097023956 0.073168239 0.037625314
## [11] 0.013337784 0.007597307
```

Only the first three components have an eigenvalue greater than 1, so we will use them in subsequent analysis.

Visualize the principal components

```
plot(pca$scores[,1:2],col=1)
text(pca$scores[,1:2],col=1, labels=EPL_Cluster$Team,cex=.5)
```




```
#3D plot
plot3d(pca$scores[,1:3], cex=1.5)
text3d(pca$scores[,1:3], text=EPL_Cluster$Team, cex=.5)
```

Defining the components

```
pca$loadings
```

```
##
## Loadings:
##                   Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
## YellowCardSum      -0.126  0.612  0.296         0.571  0.191 -0.167
## RedCardSum         -0.128 -0.110  0.770  0.438 -0.162         0.289
## GoalsSum            0.342                      0.486 -0.176 -0.255 -0.331
## GoalsAgainstSum    -0.311 -0.277  0.121        -0.180  0.299 -0.618  0.118
## FoulsSum           -0.108  0.669               -0.499 -0.335 -0.246
## FoulsAgainstSum     0.118 -0.128  0.514 -0.832
## CornersSum          0.340 -0.101         0.190        -0.209 -0.492  0.546
## CornersConcededSum -0.336                      0.299 -0.559  0.180  0.530
## ShotsSum            0.357         0.142  0.147        -0.178  0.119
```

```
## ShotsTargetSum      0.358                0.113 -0.159 -0.369        -0.226
## ShotsCondededSum   -0.358 -0.127                     -0.324        -0.225
## ShotsTargetConSum  -0.342 -0.189                0.111 -0.309 -0.286 -0.432
##                    Comp.9 Comp.10 Comp.11 Comp.12
## YellowCardSum      -0.172 -0.163   0.103   0.231
## RedCardSum          0.140  0.225
## GoalsSum            0.498  0.217  -0.319  -0.173
## GoalsAgainstSum     0.337 -0.385  -0.101   0.126
## FoulsSum            0.199         -0.147  -0.194
## FoulsAgainstSum
## CornersSum         -0.389  0.316
## CornersConcededSum  0.325 -0.189   0.101
## ShotsSum           -0.233 -0.733  -0.344  -0.270
## ShotsTargetSum      0.156 -0.191   0.456   0.609
## ShotsCondededSum   -0.320         -0.589   0.490
## ShotsTargetConSum  -0.329          0.413  -0.419
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.083  0.083  0.083  0.083  0.083  0.083  0.083  0.083
## Cumulative Var  0.083  0.167  0.250  0.333  0.417  0.500  0.583  0.667
##                Comp.9 Comp.10 Comp.11 Comp.12
## SS loadings     1.000   1.000   1.000   1.000
## Proportion Var  0.083   0.083   0.083   0.083
## Cumulative Var  0.750   0.833   0.917   1.000
```

3 components:

Component 1: Attack versus Defense

-Main contributors: Shots Target, Shots Target Conceded, Shots, Shots Conceded, Goals, Goals Conceded, Corners, Corners Conceded

Component 2: Agressive play (cannot distinguish if aggressive play is attacking or defending)

-Main contributors: Yellow Card, Fouls Committed

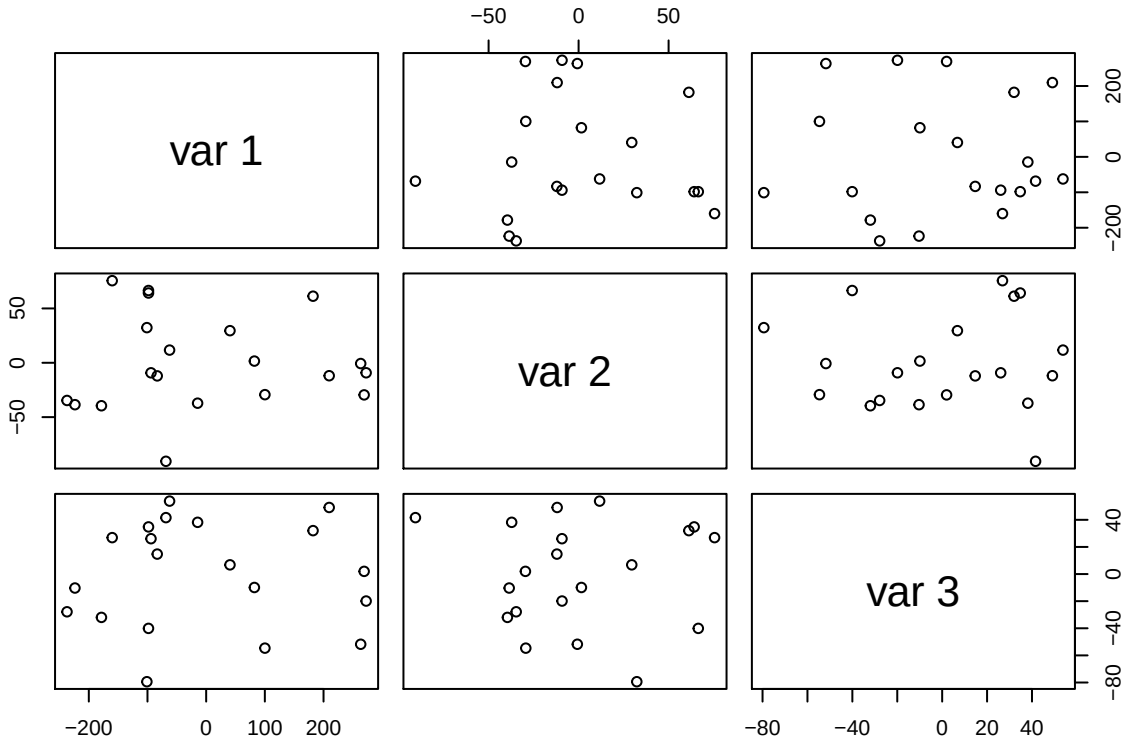
Component 3: Mutual Agression

-Main contributors: Red Card, Fouls Against

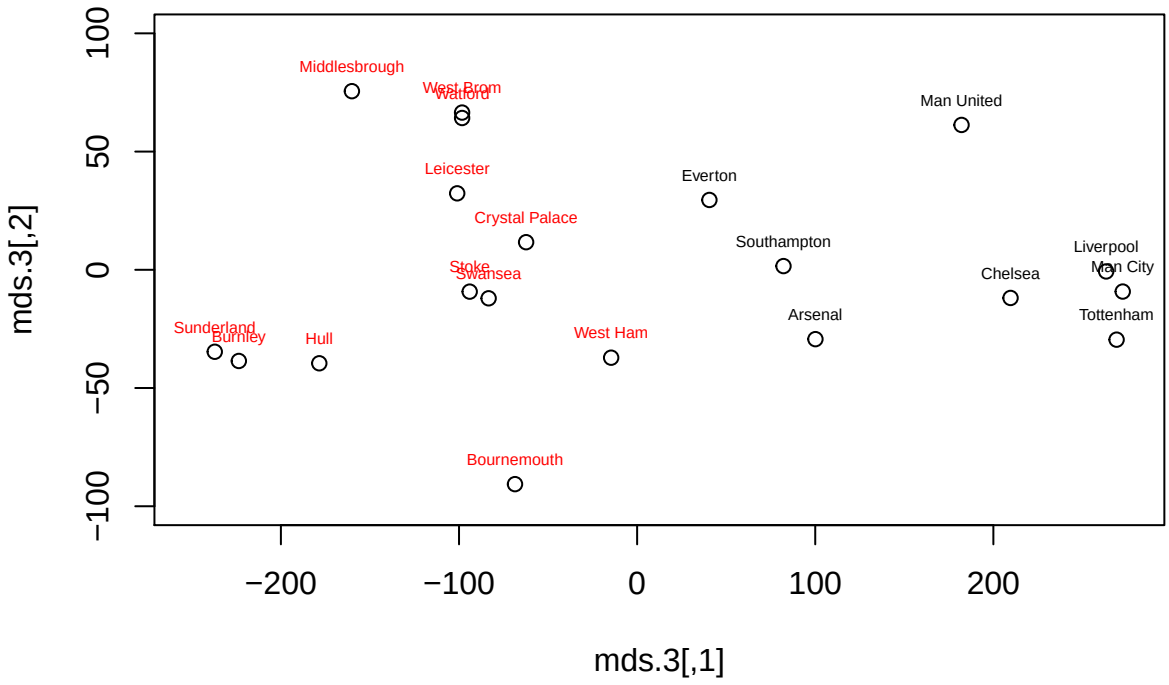
Revisiting MDS

The first three components have eigen values greater than 1, and also explain $\sim 83.70\%$ of the variance, so we conduct MDS with three dimensions.

```
mds.3 <-cmdscale(dist(EPL_Cluster[,-(1:4)]),k=3)
pairs(mds.3)
```




```
plot(mds.3, ylim = c(-100,100), xlim = c(-250,275))
text(mds.3, labels = EPL_Cluster$Team, col = km.2$clust, cex=.5,pos=3)
```



Clustering on PCA

Although there is controversy in clustering on PCA, we want to see if it better separates the teams.

Two-Cluster Solution

Once again, we start with a two-cluster solution.

```
set.seed(2011)
km.2pca <- kmeans(pca$scores, 2, nstart= 100)
c.crit(km.2pca)$C.g
```

```
## [1] 15.86811
```

The *C(g)* is 15.868.

Three-Cluster Solution

For comparison, we run a three-cluster k-means model.

```
set.seed(2011)
km.3pca <- kmeans(pca$scores, 3, nstart= 100)
c.crit(km.3pca)$C.g
```

```
## [1] 11.16683
```

Once again, three cluster solution is not an improvement, $C(g) = 11.167$. We will stick with our two clusters.

Exploring the Two-Cluster Solution

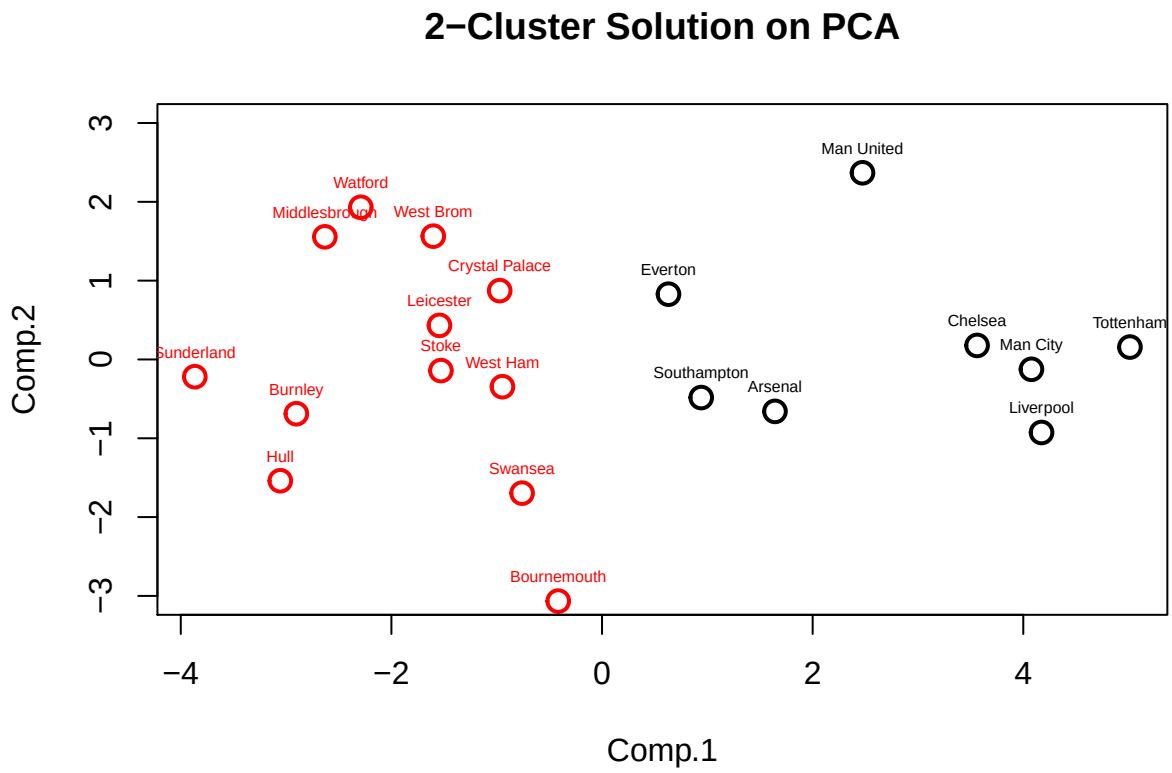
There are no differences in the teams assigned to each cluster across the two solutions.

```
xtabs(~km.2$cluster+km.2pca$cluster)
```

```
##             km.2pca$cluster
## km.2$cluster  1  2
##            1  8  0
##            2  0 12
```

Below is a two-dimensional visualization of the two-cluster solution using PCAs. We see a similar separation to the k-means solution on the raw data in that the top 4/6 teams score higher on Component 1.

```
plot(pca$scores[,1:2],col=km.2pca$clust,pch=1,cex=1.5, lwd=2,
   main = "2-Cluster Solution on PCA", ylim = c(-3,3))
text(pca$scores[,1:2],labels=EPL_Cluster$Team,col=km.2pca$clust, cex=.5,pos=3)
```



3D Plot

Since 3D plots are cool, here are the 3 principal components colored by their respective k-means clusters.

```r
plot3d(pca$scores)
text3d(pca$scores, text = EPL_Cluster$Team, col = km.2pca$clust, cex=.5,pos=3)
```