Untitled

Problem 7.1

Function helping to make right chouse about number of K

```
wssplot <- function(data, nc=15, seed=1234){
               wss <- (nrow(data)-1)*sum(apply(data,2,var))
               for (i in 2:nc){
                    set.seed(seed)
                    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
                plot(1:nc, wss, type="b", xlab="Number of Clusters",
                     ylab="Within groups sum of squares")}
```

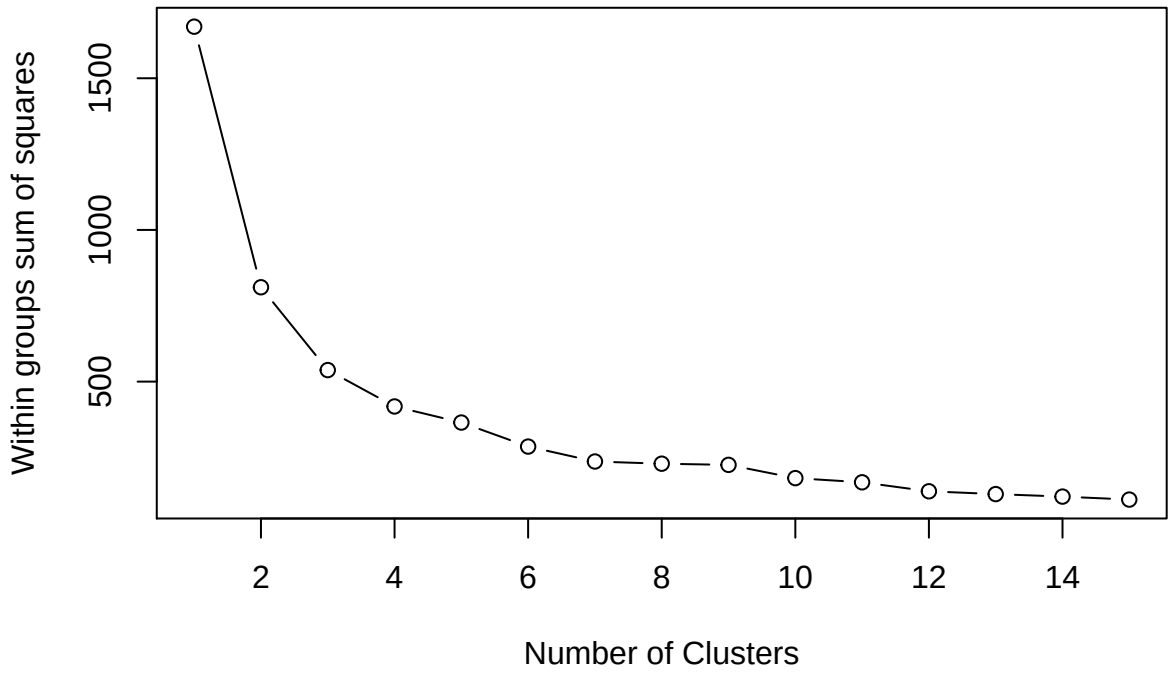
Data that we want to use for clusterization

```
kmeans.data <- data[4:6]
head(kmeans.data)
```

```
##    Gm Age Salary
## 1  74  28   9.50
## 2  77  29   9.25
## 3  83  29   9.00
## 4  93  29   8.75
## 5  81  29   8.75
## 6 108  24   8.00
```

Scaling the data

```
kmeans.data.scaled <- scale(kmeans.data[-1])
wssplot(kmeans.data.scaled)
```



Let's do some clusterization

3 clusters

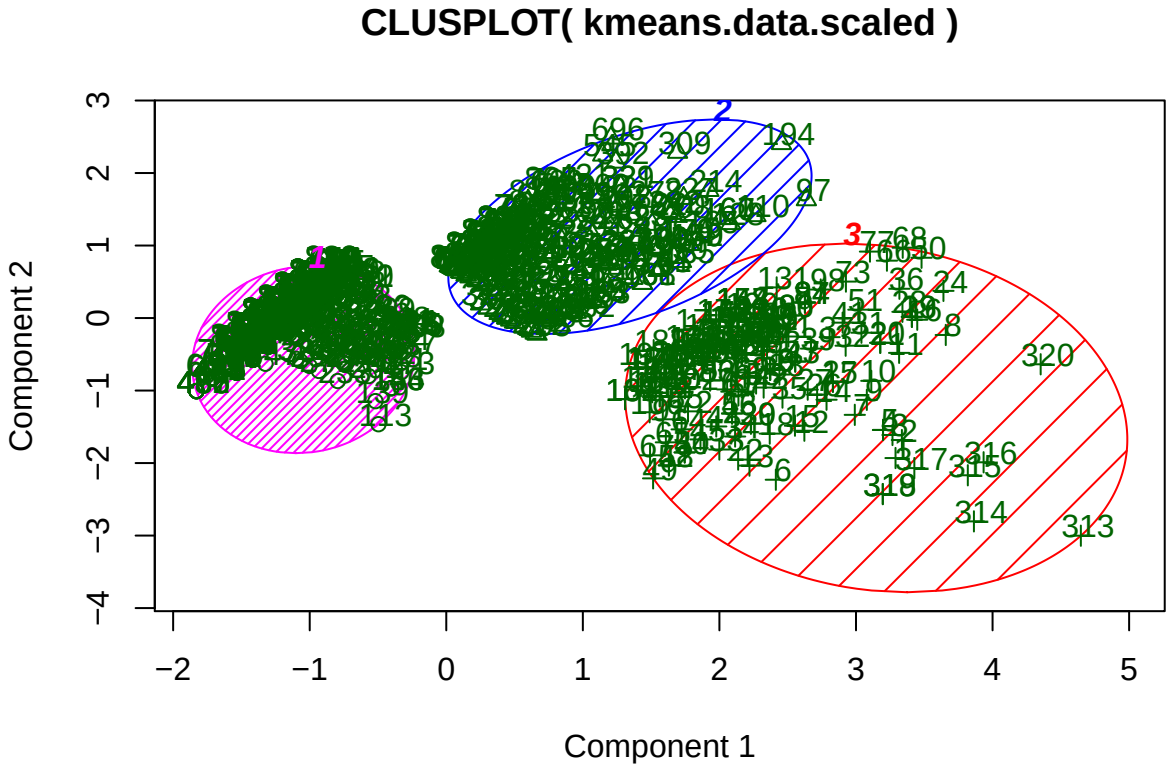
```
# K-Means Cluster Analysis
fit3 <- kmeans(kmeans.data.scaled, 3)
# get cluster means
aggregate(kmeans.data.scaled,by=list(fit3$cluster),FUN=mean)
```

```
##   Group.1        Age     Salary
## 1       1 -0.7153821 -0.5447271
## 2       2  1.0839778 -0.1229349
## 3       3  0.5785685  1.6573821
```

```
# append cluster assignment
kmeans.data.scaled <- data.frame(kmeans.data.scaled, fit3$cluster)
```

Cluster Plot against 1st 2 principal components

```
# vary parameters for most readable graph
library(cluster)
clusplot(kmeans.data.scaled, fit3$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```




```
#Centroid Plot against 1st 2 discriminant functions

#library(fpc)
#plotcluster(kmeans.data, fit$cluster)
```

4 clusters

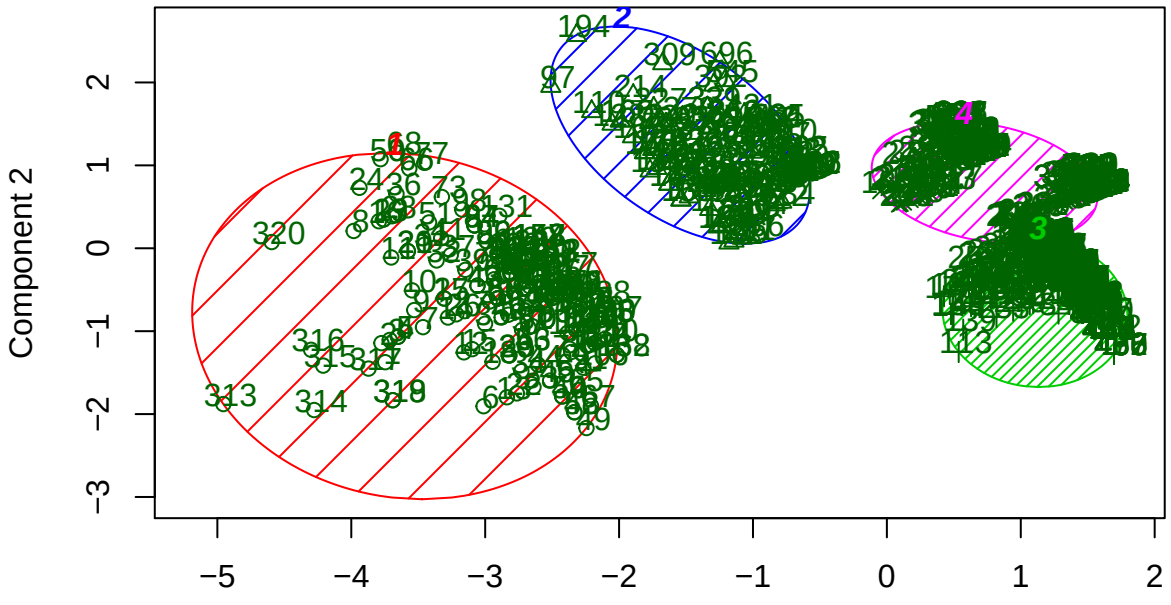
```
# K-Means Cluster Analysis
fit4 <- kmeans(kmeans.data.scaled, 4)
# get cluster means
aggregate(kmeans.data.scaled,by=list(fit4$cluster),FUN=mean)
```

```
##   Group.1        Age     Salary fit3.cluster
## 1       1  0.5785685  1.6573821     3.000000
## 2       2  1.5014941  0.0695924     2.000000
## 3       3 -0.8646274 -0.5612934     1.000000
## 4       4  0.2795692 -0.4223518     1.535714
```

```
# append cluster assignment
kmeans.data.scaled <- data.frame(kmeans.data.scaled, fit4$cluster)
```

Cluster Plot against 1st 2 principal components

```
# vary parameters for most readable graph
clusplot(kmeans.data.scaled, fit4$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```



7 clusters

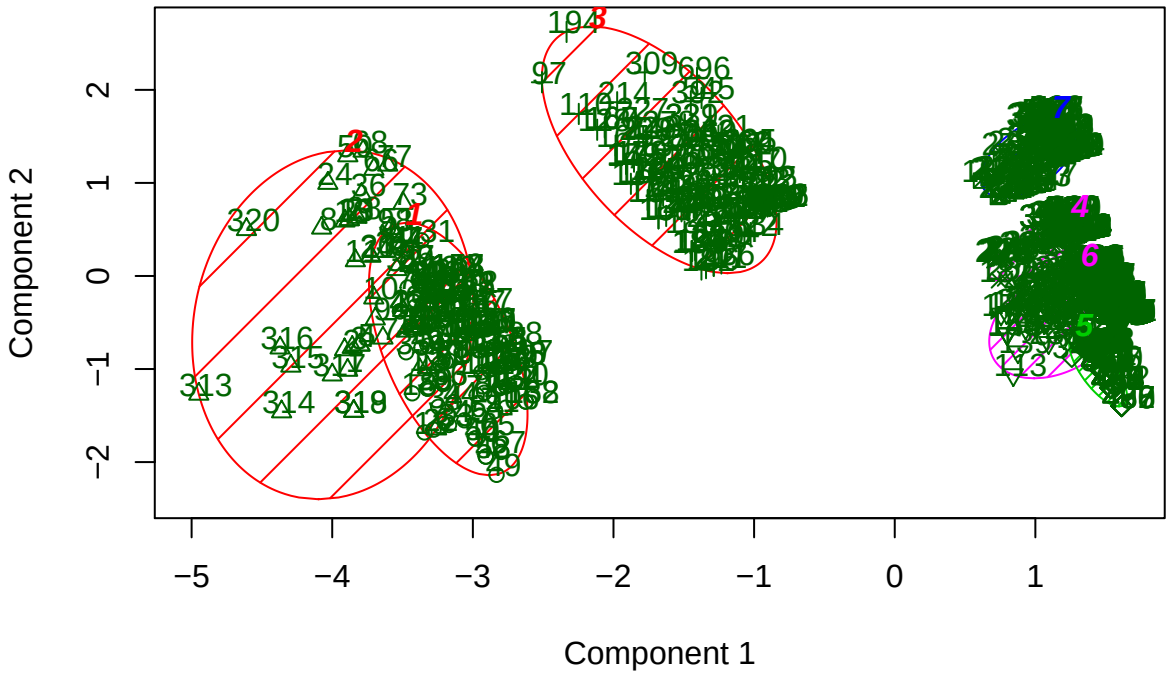
```
# K-Means Cluster Analysis
fit7 <- kmeans(kmeans.data.scaled, 7)
# get cluster means
aggregate(kmeans.data.scaled,by=list(fit7$cluster),FUN=mean)
```

```
##   Group.1          Age     Salary fit3.cluster fit4.cluster
## 1       1  0.359074937  1.2872872            3            1
## 2       2  1.168762228  2.6525262            3            1
## 3       3  1.501494095  0.0695924            2            2
## 4       4  0.009796648 -0.4642320            1            4
## 5       5 -1.332381760 -0.6510886            1            3
## 6       6 -0.650539781 -0.5201948            1            3
## 7       7  0.513372139 -0.3860555            2            4
```

```
# append cluster assignment
kmeans.data.scaled <- data.frame(kmeans.data.scaled, fit7$cluster)
```

Cluster Plot against 1st 2 principal components

```
# vary parameters for most readable graph
clusplot(kmeans.data.scaled, fit7$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```



comparing 2 cluster solutions 3-means and 4-means

```
library(fpc)
d <- dist(kmeans.data.scaled)
cluster.stats(d, fit3$cluster, fit4$cluster)
```

```
## $n
## [1] 836
##
## $cluster.number
## [1] 3
##
## $cluster.size
## [1] 457 213 166
##
## $min.cluster.size
## [1] 166
##
## $noisen
## [1] 0
##
## $diameter
## [1] 3.217215 5.639870 5.162372
##
## $average.distance
## [1] 1.229784 2.733434 1.476078
##
## $median.distance
```

```
## [1] 1.202514 2.082944 1.360096
##
## $separation
## [1] 1.788345 1.765984 1.765984
##
## $average.toother
## [1] 4.445117 3.714490 5.530665
##
## $separation.matrix
##          [,1]     [,2]     [,3]
## [1,] 0.000000 1.788345 4.461521
## [2,] 1.788345 0.000000 1.765984
## [3,] 4.461521 1.765984 0.000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]
## [1,] 0.000000 3.326944 5.879882
## [2,] 3.326944 0.000000 4.781407
## [3,] 5.879882 4.781407 0.000000
##
## $average.between
## [1] 4.50213
##
## $average.within
## [1] 1.495483
##
## $n.between
## [1] 208561
##
## $n.within
## [1] 140469
##
## $max.diameter
## [1] 5.63987
##
## $min.separation
## [1] 1.765984
##
## $within.cluster.ss
## [1] 1937.909
##
## $clus.avg.silwidths
##          1          2          3
## 0.62709171 0.05796035 0.69278774
##
## $avg.silwidth
## [1] 0.4951307
##
## $g2
## NULL
##
## $g3
## NULL
##
```

```
## $pearsongamma
## [1] 0.7198141
##
## $dunn
## [1] 0.3131249
##
## $dunn2
## [1] 1.21713
##
## $entropy
## [1] 0.9995311
##
## $wb.ratio
## [1] 0.3321723
##
## $ch
## [1] 932.4685
##
## $cwidegap
## [1] 1.799613 4.475266 1.054805
##
## $widestgap
## [1] 4.475266
##
## $sindex
## [1] 1.829292
##
## $corrected.rand
## [1] 0.704604
##
## $vi
## [1] 0.5621104
```

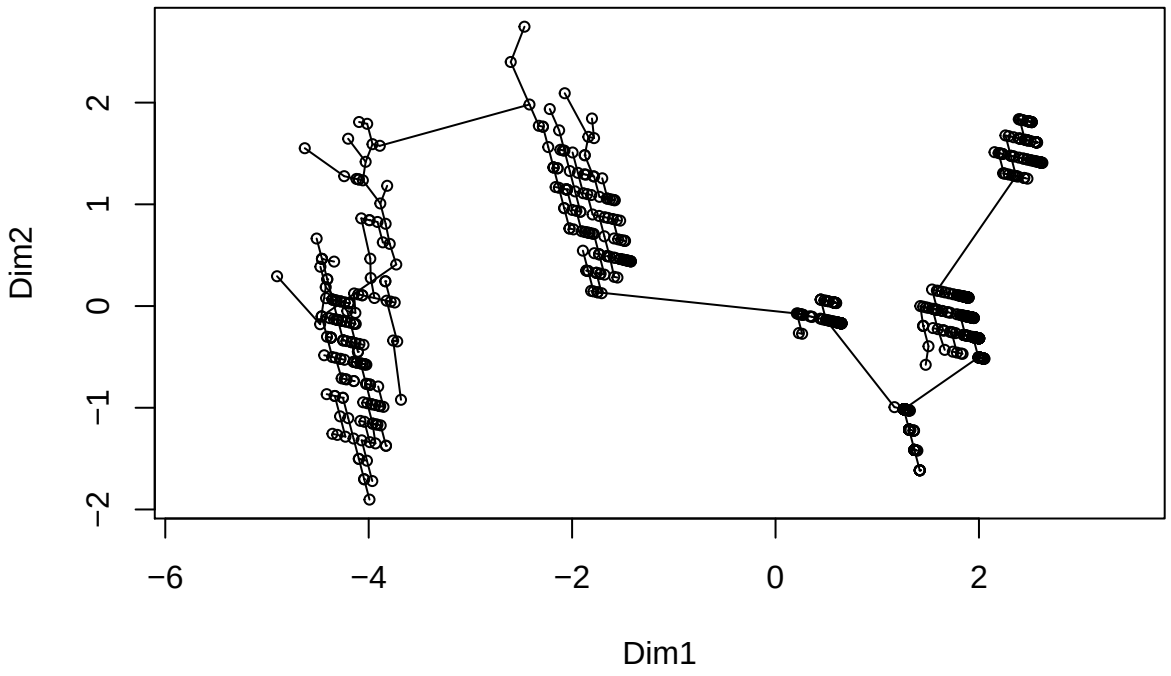
The correct choice of k is often ambiguous, with interpretations depending on the shape and scale of the distribution of points in a data set and the desired clustering resolution of the user. In addition, increasing k without penalty will always reduce the amount of error in the resulting clustering, to the extreme case of zero error if each data point is considered its own cluster (i.e., when k equals the number of data points, n). Intuitively then, the optimal choice of k will strike a balance between maximum compression of the data using a single cluster, and maximum accuracy by assigning each data point to its own cluster.

So, in our case we see that the total within-cluster sum of squares is redusing as k is getting bigger (from 165 for k=3 to 125 for k=7).

Problem 7.2

Build a minimum spanning tree connecting all points and delete 6 maximum edges step by step.

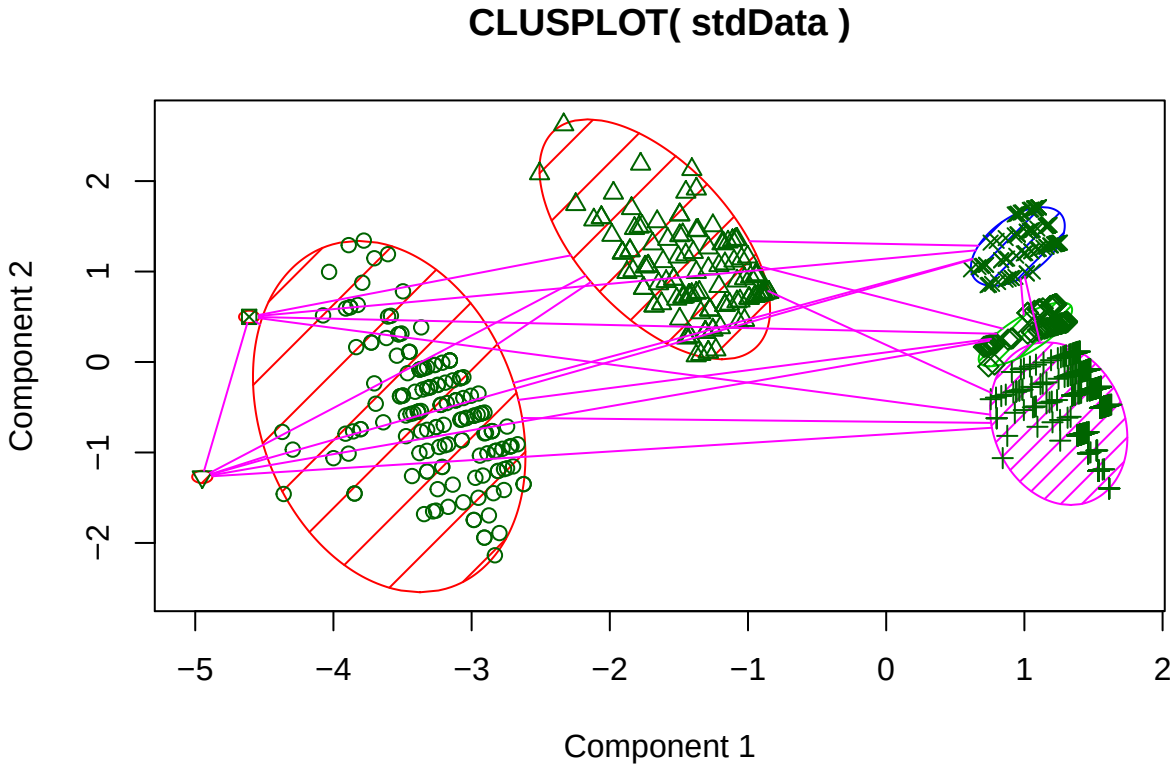
```
stdData <- kmeans.data.scaled
dis <- dist(stdData) #dissimilarity data
tr <- spantree(dis, toolong = 0) #build an MST
plot(tr, cmdscale(dis)) #plot an MST
```




```
for(i in 1:6) #loop for finding 6 maximum edges in MST
{
  m <- which.max(tr$dist)
  tr$dist[m] <- NA
}
m <- which.max(tr$dist)
tres <- tr$dist[m] + 0.00001 #set a treshold
j <- distconnected(dis, toolong = tres) #find connected components in MST after deleting 6 edges
```

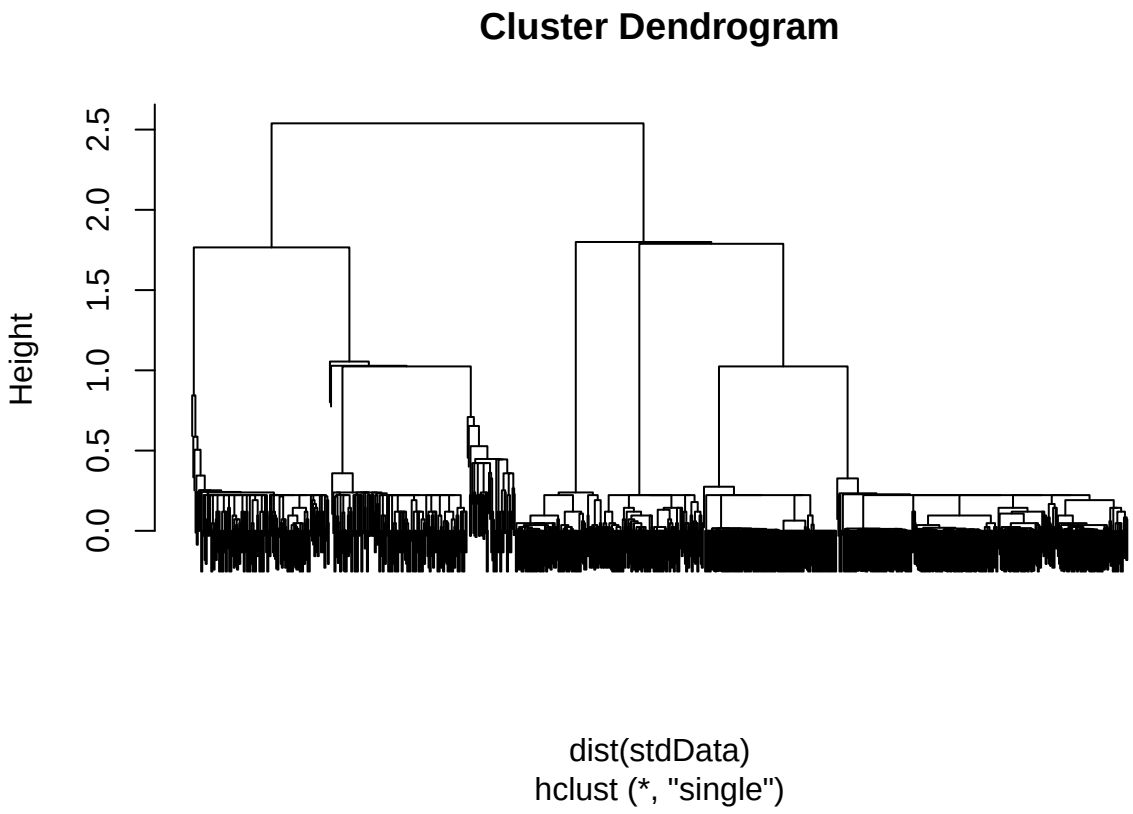
```
## Connectivity of distance matrix with threshold dissimilarity 1.024483
## Data are disconnected: 7 groups
## Groups sizes
##   1   2   3   4   5   6   7
## 164 123 379  90  78   1   1
```

```
clusplot(stdData, j, color=TRUE, shade=TRUE) #plot clusters
```

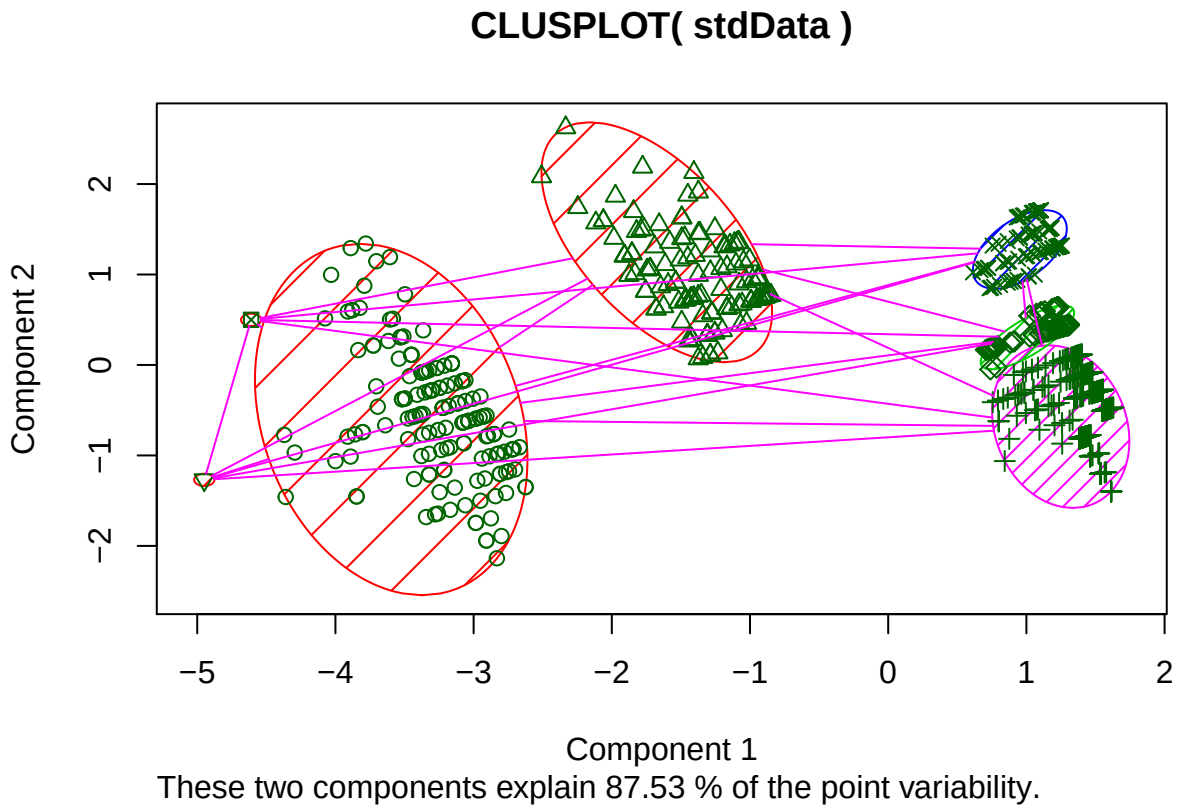


As we see, MST doesn't work good for our data. It devides dataset in 1 big cluster and 6 small (1-3 players). Make sure that it is true by using the single linkage method (which is closely related to the minimal spanning tree).

```
hc <- hclust(dist(stdData), method = "single")
plot(hc, labels = FALSE)
```




```
memb <- cutree(hc, k = 7)
clusplot(stdData, memb, color=TRUE, shade=TRUE)
```



We get the same results (1 big cluster with 828 players, 1 cluster with 3 plauers and 5 clusters with 1 player). This may be partly because MST (with a single linkage) finds elongated structures in data. However, hockey features don't have any geometric meaning. So, in our case K-means works much better (although not perfect) which makes convex clusters of players by grouping them around some centroids.