Olympic Run Record

Introduction

The objective of this project was analyzing the data of the Olymic run record to cluster countries based on their run records using different clustering methods and to analyze which one would produce the best result.

K-means clustering

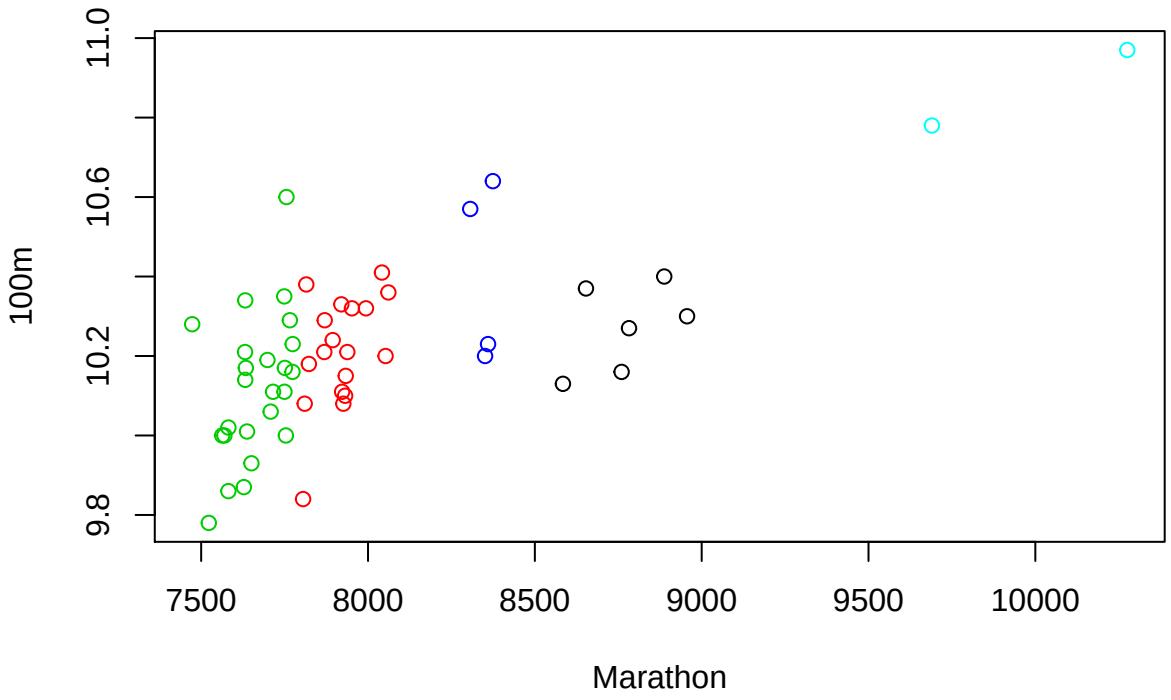
```
# Set random seed.
set.seed(1)

# Explore your data with str() and summary()
str(run_record)
```

```
## 'data.frame':    54 obs. of  8 variables:
##  $ X100m   : num  10.23 9.93 10.15 10.14 10.27 ...
##  $ X200m   : num  20.4 20.1 20.4 20.2 20.3 ...
##  $ X400m   : num  46.2 44.4 45.8 45 45.3 ...
##  $ X800m   : num  106 104 106 104 107 ...
##  $ X1500m  : num  221 212 215 214 222 ...
##  $ X5000m  : num  800 776 796 770 878 ...
##  $ X10000m : num  1659 1652 1663 1612 1829 ...
##  $ marathon: num  7774 7651 7933 7632 8782 ...
```

```
summary(run_record)
```

```
##      X100m           X200m           X400m           X800m      
##  Min.   : 9.78   Min.   :19.32   Min.   :43.18   Min.   :101.4  
##  1st Qu.:10.10   1st Qu.:20.17   1st Qu.:44.91   1st Qu.:103.8  
##  Median :10.20   Median :20.43   Median :45.58   Median :105.6  
##  Mean   :10.22   Mean   :20.54   Mean   :45.83   Mean   :106.1  
##  3rd Qu.:10.32   3rd Qu.:20.84   3rd Qu.:46.32   3rd Qu.:108.0  
##  Max.   :10.97   Max.   :22.46   Max.   :51.40   Max.   :116.4  
##      X1500m          X5000m          X10000m        marathon    
##  Min.   :206.4   Min.   : 759.6   Min.   :1588   Min.   : 7473  
##  1st Qu.:213.0   1st Qu.: 788.9   1st Qu.:1653   1st Qu.: 7701  
##  Median :216.6   Median : 805.2   Median :1675   Median : 7819  
##  Mean   :219.2   Mean   : 817.1   Mean   :1712   Mean   : 8009  
##  3rd Qu.:224.2   3rd Qu.: 834.5   3rd Qu.:1739   3rd Qu.: 8050  
##  Max.   :254.4   Max.   :1002.0   Max.   :2123   Max.   :10276
```

```
# Cluster run_record using k-means
run_km = kmeans(run_record , 5, nstart = 20)

# Plot the 100m as function of the marathon. Color using clusters
plot(run_record$marathon ,run_record$X100m , col = run_km$cluster  , xlab = "Marathon" , ylab = "100m")
```




```
# Calculate Dunn's index
dunn_km = dunn(clusters = run_km$cluster ,Data = run_record)
dunn_km
```

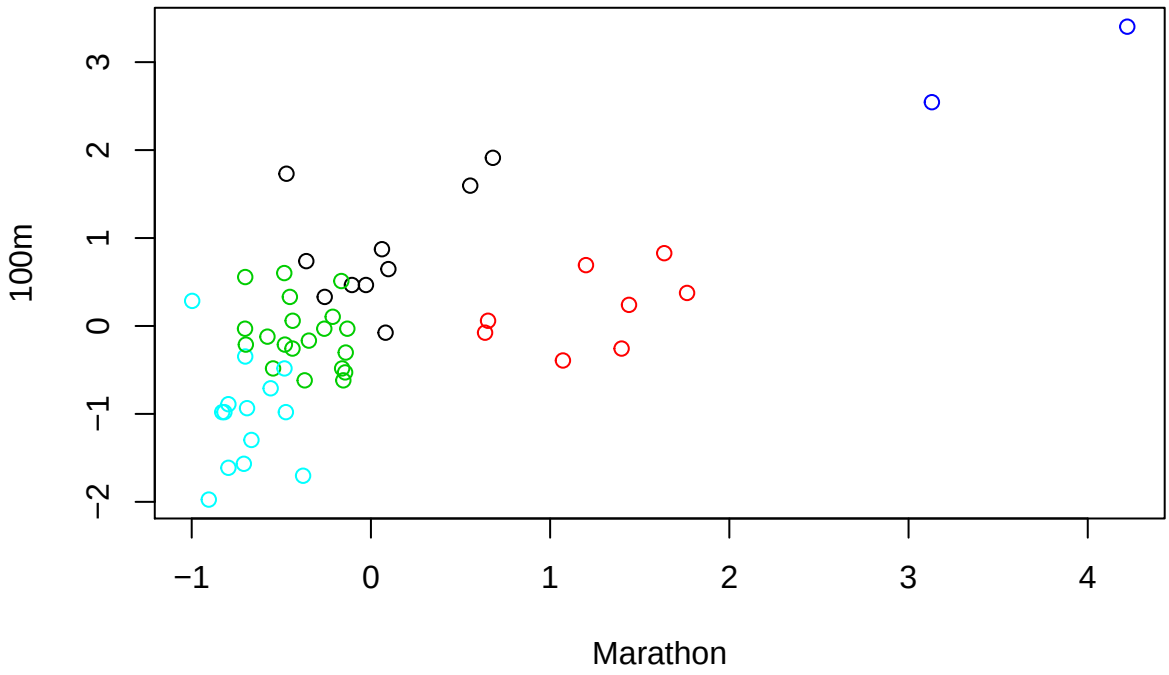
```
## [1] 0.05651773
```

As you can see from the plot the the unstandarized clusters are completely dominated by the marathon records; you can even separate every cluster only based on the marathon records. Moreover Dunn's index seems to be quite low.

K-means clustering on standardized data

In order to get satisfying results, the data needs to be standardized which is done using scale function.

```
# Set random seed.
set.seed(1)

# Standardize run_record, transform to a dataframe
run_record_sc = as.data.frame(scale(run_record))

# Cluster run_record_sc using k-means
run_km_sc = kmeans(run_record_sc , 5, nstart= 20)

# Plot records on 100m as function of the marathon.
plot(run_record_sc$marathon ,run_record_sc$X100m , col = run_km_sc$cluster , xlab = "Marathon" , ylab =
```




```
# Compare the resulting clusters in a table
table(run_km$cluster , run_km_sc$cluster)
```

```
##
##      1  2  3  4  5
##   1  0  6  0  0  0
##   2  7  0 10  0  1
##   3  1  0 10  0 13
##   4  2  2  0  0  0
##   5  0  0  0  2  0
```

```
# Calculate Dunn's index
dunn_km_sc = dunn(clusters = run_km_sc$cluster ,Data = run_record_sc)
dunn_km_sc
```

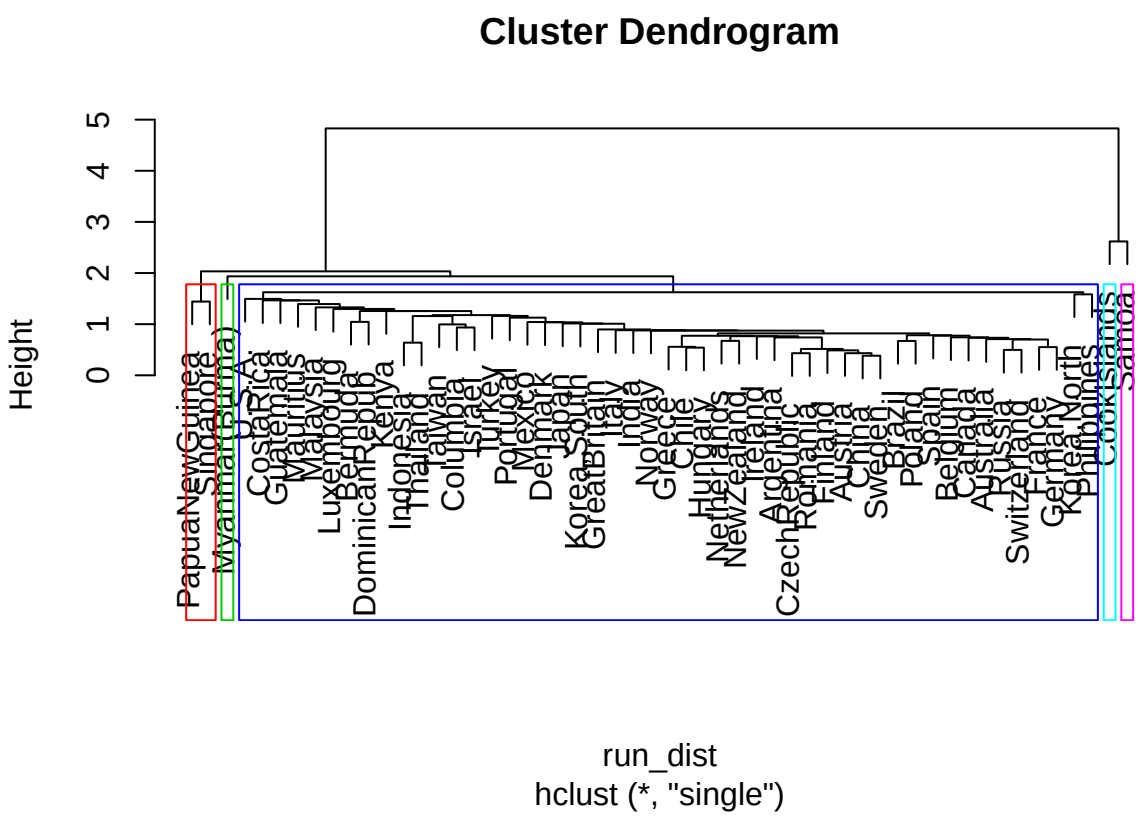
```
## [1] 0.1453556
```

The plot now shows the influence of the 100m records on the resulting clusters! Dunn's index is clear about it, the standardized clusters are more compact and better separated.

Hierarchical clustering with single linkage

```
# Apply dist() to run_record_sc
run_dist = dist(run_record_sc )

# Apply hclust() to run_dist
run_single = hclust(run_dist , method = "single")

# Apply cutree() to run_single.
```

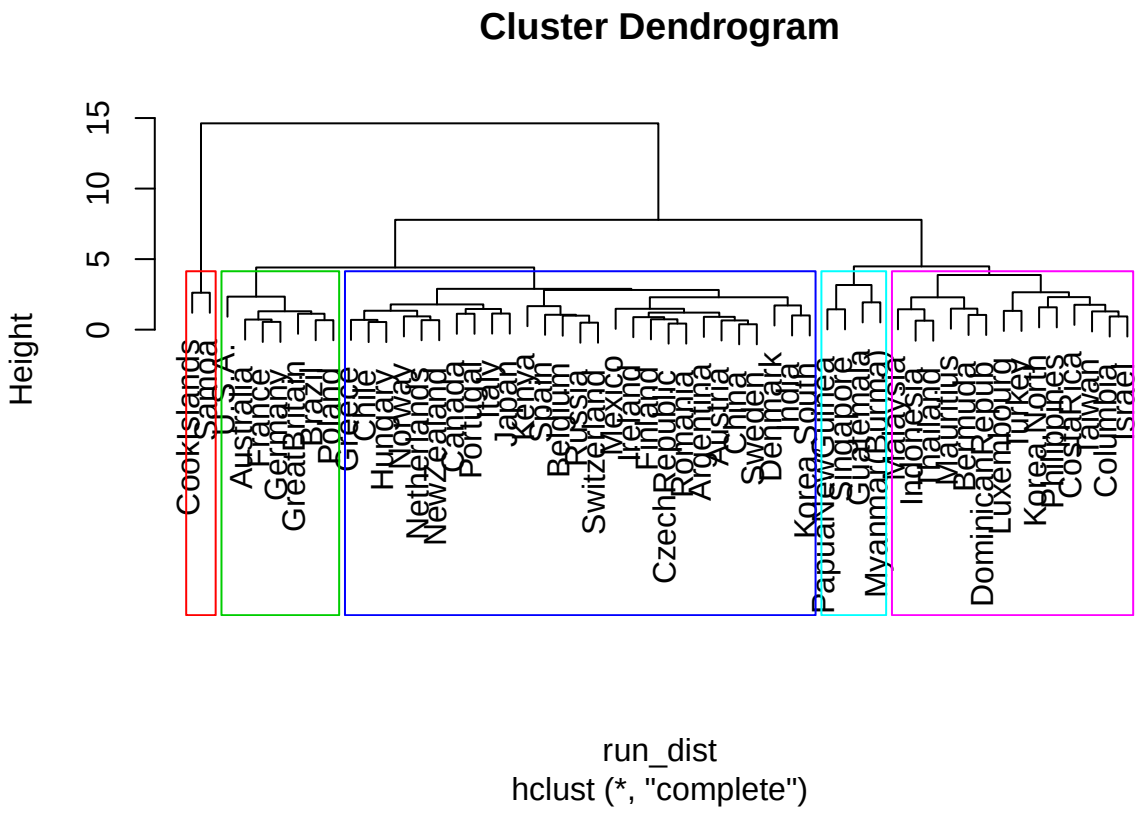
```
memb_single = cutree(run_single , k =5)

# Apply plot() on run_single to draw the dendrogram
plot(run_single )

# Apply rect.hclust() on run_single to draw the boxes
rect.hclust(run_single ,k= 5 ,border = 2:6)
```



As you can see from the plot that there are two islands Samoa and Cook's Islands, who are not known for their sports performances, have both been placed in their own groups. The single-linkage method appears to be placing each outlier in its own cluster.

Hierarchical clustering with complete linkage

```
# Apply hclust() to run_dist
run_complete = hclust(run_dist , method = "complete")

# Apply cutree() to run_complete
memb_complete = cutree(run_complete , k =5)

# Apply plot() on run_complete to draw the dendrogram
plot(run_complete)

# Apply rect.hclust() on run_complete to draw the boxes
rect.hclust(run_complete ,k =5, border = 2:6)
```




```
# table() the clusters memb_single and memb_complete.
table(memb_single , memb_complete )
```

```
##             memb_complete
## memb_single  1  2  3  4  5
##           1 27  7 14  0  1
##           2  0  0  0  1  0
##           3  0  0  0  0  1
##           4  0  0  0  0  2
##           5  0  0  0  1  0
```

Compare the two plots. The five clusters differ significantly from the single-linkage clusters. That one big cluster is now split up into 4 medium sized clusters.

Comparing Dunn's Index

Compare the Dunn's index of the three clustering methods to see which will give the best results.

```
# Set random seed.
set.seed(100)

# Dunn's index for k-means
dunn_km = dunn(cluster = run_km_sc$cluster , Data = run_record_sc)

# Dunn's index for single-linkage
dunn_single = dunn(cluster = memb_single , Data = run_record_sc)
```

```r
# Dunn's index for complete-linkage
dunn_complete = dunn(cluster = memb_complete, Data = run_record_sc)

# Compare k-means with single-linkage
table(run_km_sc$cluster,memb_single)
```

```
##    memb_single
##      1  2  3  4  5
##   1  9  0  1  0  0
##   2  6  0  0  2  0
##   3 20  0  0  0  0
##   4  0  1  0  0  1
##   5 14  0  0  0  0
```

```r
# Compare k-means with complete-linkage
table(run_km_sc$cluster ,memb_complete )
```

```
##    memb_complete
##      1  2  3  4  5
##   1  0  0  8  0  2
##   2  0  0  6  0  2
##   3 20  0  0  0  0
##   4  0  0  0  2  0
##   5  7  7  0  0  0
```

In conclusion the single-linkage method returned the highest ratio of minimal intercluster-distance to maximal cluster diameter thereby giving the most accurate result out of the three methods.