Functional trait clustering in wheat seeds

Miguel Ángel Juárez Garzón

Exercise 1: Data Load

```
seeds<-read.csv("seeds.csv")
head(seeds)
```

	area <dbl>	perimeter <dbl>	compactness <dbl>	length <dbl>	wi… <dbl>	asymme… <dbl>	groove_length <dbl>	species_class <int>
1	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	1
2	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
3	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	1
4	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
5	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1
6	14.38	14.21	0.8951	5.386	3.312	2.462	4.956	1

6 rows

K-means Groups

```
seeds_km <- kmeans(seeds[,-8], centers = 3, nstart = 20)
seeds_km
```

```
## K-means clustering with 3 clusters of sizes 77, 61, 72
##
## Cluster means:
##       area perimeter compactness   length    width asymmetry groove_length
## 1 11.96442  13.27481   0.8522000 5.229286 2.872922  4.759740      5.088519
## 2 18.72180  16.29738   0.8850869 6.208934 3.722672  3.603590      6.066098
## 3 14.64847  14.46042   0.8791667 5.563778 3.277903  2.648933      5.192319
##
## Clustering vector:
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 1 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3
##  [38] 2 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 1 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 2 2 2 2 2 2 3 3 3 3 2 3 3 3 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 195.7453 184.1086 207.4648
##  (between_SS / total_SS =  78.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# Confusion matrix
tb <- table(seeds_km$cluster, seeds$species_class)

# Ordering matrix
fst <- which.max(c(tb [1,1],tb [2,1],tb [3,1]))
scd <- which.max(c(tb [1,2],tb [2,2],tb [3,2]))
thd <- which.max(c(tb [1,3],tb [2,3],tb [3,3]))

tb <- tb[c(fst, scd, thd),]
tb
```

```
##
##      1  2  3
##   3 60 10  2
##   2  1 60  0
##   1  9  0 68
```

```
error_rate <- sum(sum(tb-diag(diag(tb))))/sum(colSums(tb))
error_rate
```

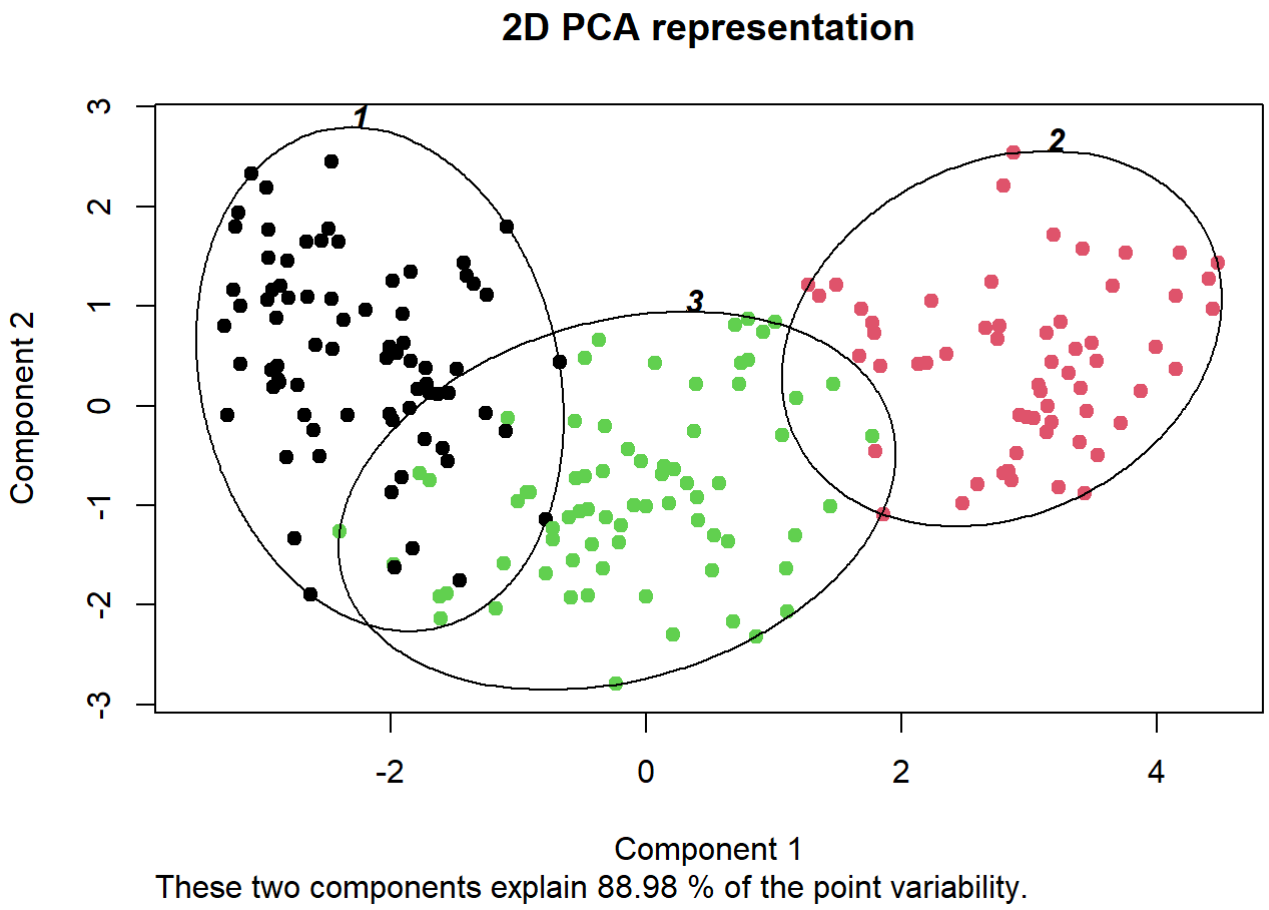
```
## [1] 0.1047619
```

Graphical representation

```
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 4.0.4
```

```
clusplot(seeds[-8], seeds_km$cluster, main="2D PCA representation",
color=FALSE, shade=FALSE, labels=4, lines=0,
col.p = seeds_km$cluster, col.clu = 1, plotchar = FALSE,
pch = 19)
```



Exercise 2: Data standardization

```
seeds_std <- scale(seeds[,-8], center = TRUE, scale = TRUE)
seeds_std <- data.frame(seeds_std,seeds$species_class)
head(seeds_std)
```

	area <dbl>	perimeter <dbl>	compactness <dbl>	length <dbl>	width <dbl>	asymmetry <dbl>	groove_le <
1	0.14175904	0.214948819	6.045733e-05	0.30349301	0.1413640	-0.9838010	-0.382
2	0.01116136	0.008204153	4.274938e-01	-0.16822270	0.1969616	-1.7839036	-0.919
3	-0.19160873	-0.359341919	1.438945e+00	-0.76181710	0.2075516	-0.6658882	-1.186
4	-0.34626388	-0.474200066	1.036904e+00	-0.68733567	0.3187467	-0.9585276	-1.227
5	0.44419577	0.329806966	1.371233e+00	0.06650665	0.8032397	-1.5597684	-0.474
6	-0.16067770	-0.267455401	1.019976e+00	-0.54740087	0.1413640	-0.8235144	-0.919

6 rows | 1-8 of 9 columns

Standardized K-means groups

```
seeds_km_std <- kmeans(seeds_std[,-8], centers = 3, nstart = 20)
seeds_km_std
```

```
## K-means clustering with 3 clusters of sizes 72, 71, 67
##
## Cluster means:
##         area  perimeter compactness     length        width   asymmetry
## 1 -1.0277967 -1.0042491  -0.9626050 -0.8955451 -1.082995635  0.69314821
## 2 -0.1407831 -0.1696372   0.4485346 -0.2571999  0.001643014 -0.66034079
## 3  1.2536860  1.2589580   0.5591283  1.2349319  1.162075101 -0.04511157
##   groove_length
## 1    -0.6233191
## 2    -0.5844965
## 3     1.2892273
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 1 2 2 2 2 2 1 3 3 3 3
##  [75] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [112] 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 2 3 3 2 3 2 2 3 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 144.5954 144.4586 139.5542
##  (between_SS / total_SS =  70.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# Confusion matrix
tb_std <- table(seeds_km_std$cluster, seeds_std$seeds.species_class)

fst <- which.max(c(tb_std [1,1],tb_std [2,1],tb_std  [3,1]))
scd <- which.max(c(tb_std  [1,2],tb_std  [2,2],tb_std  [3,2]))
thd <- which.max(c(tb_std  [1,3],tb_std  [2,3],tb_std  [3,3]))

tb_std  <- tb_std [c(fst, scd, thd),]
tb_std
```

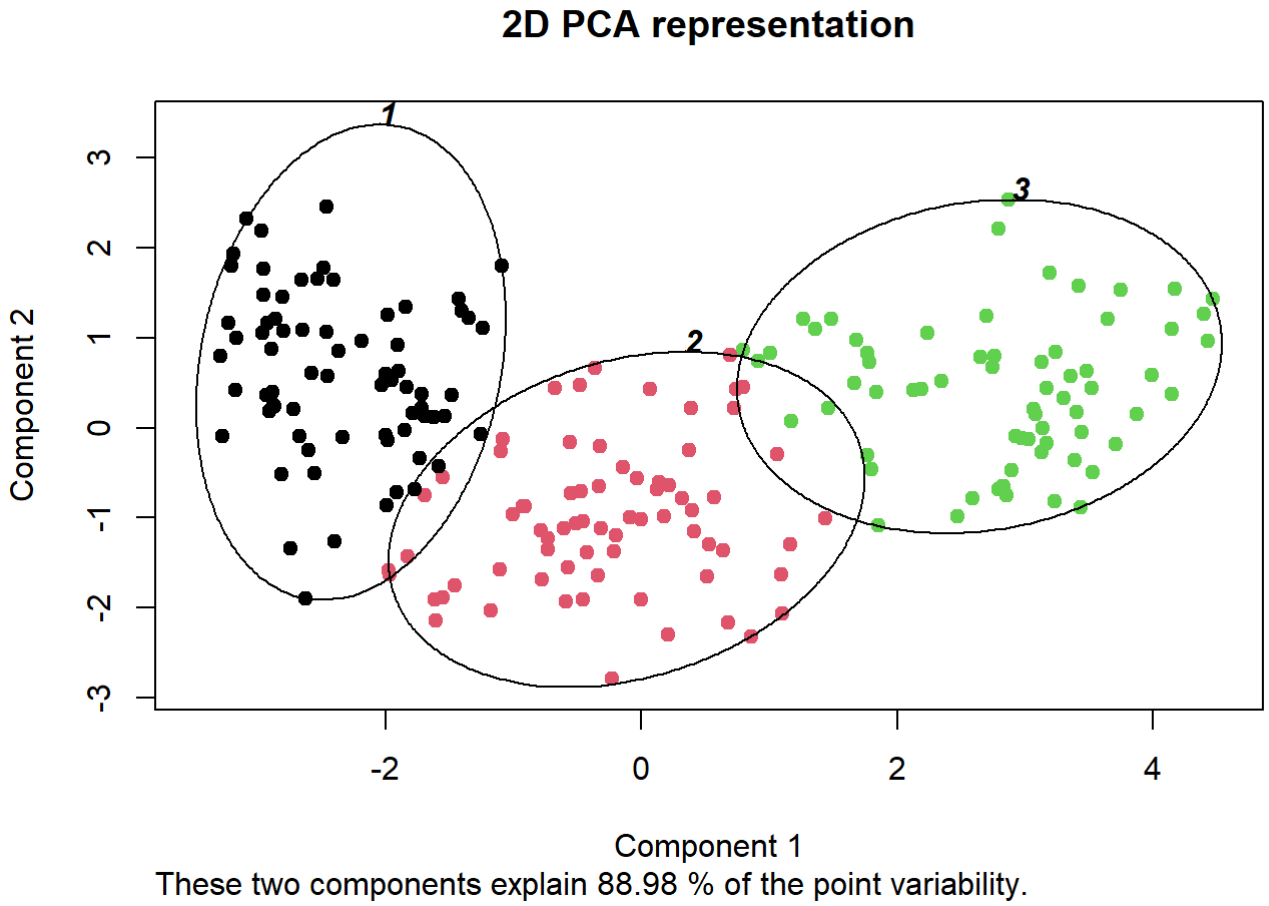
```
##
##      1  2  3
##   2 62  5  4
##   3  2 65  0
##   1  6  0 66
```

```
#Error rate
error_rate <- sum(sum(tb_std-diag(diag(tb_std))))/sum(colSums(tb_std))
error_rate
```

```
## [1] 0.08095238
```

Standardized data graph

```
clusplot(seeds_std[-8], seeds_km_std$cluster, main="2D PCA representation",
color=FALSE, shade=FALSE, labels=4, lines=0,
col.p = seeds_km_std$cluster, col.clu = 1, plotchar = FALSE,
pch = 19)
```



We can observe that standardized data exhibits a lower error rate compared to non-standardized data. Additionally, in the graphical representation, we can see that the clustering is improved, and there are overlapping regions between smaller groups.

Validation indexes

Validation non-standardized data

```
library(fpc)
```

```
## Warning: package 'fpc' was built under R version 4.0.4
```

```
# Calculamos las distancias
seeds_dist = dist(seeds[,-8], method = "euclidean")
# Calculamos las medidas de eficiencia
clust_stats <- cluster.stats(d = seeds_dist, seeds$species_class,
seeds_km$cluster)
clust_stats$corrected.rand
```

```
## [1] 0.7166199
```

```
clust_stats$vi
```

```
## [1] 0.6689058
```

Exercise 3: Standardized data validation

```
# Distance calculation
seeds_dist_std = dist(seeds_std[,-8], method = "euclidean")
# Means calculation
clust_stats <- cluster.stats(d = seeds_dist_std, seeds_std$seeds.species_class,
seeds_km_std$cluster)
clust_stats$corrected.rand
```

```
## [1] 0.7732937
```

```
clust_stats$vi
```

```
## [1] 0.5978132
```

Observing the indices for standardized and non-standardized data, we can see that standardized data leads to better clustering, with a Rand index closer to 1 and a smaller VI index.

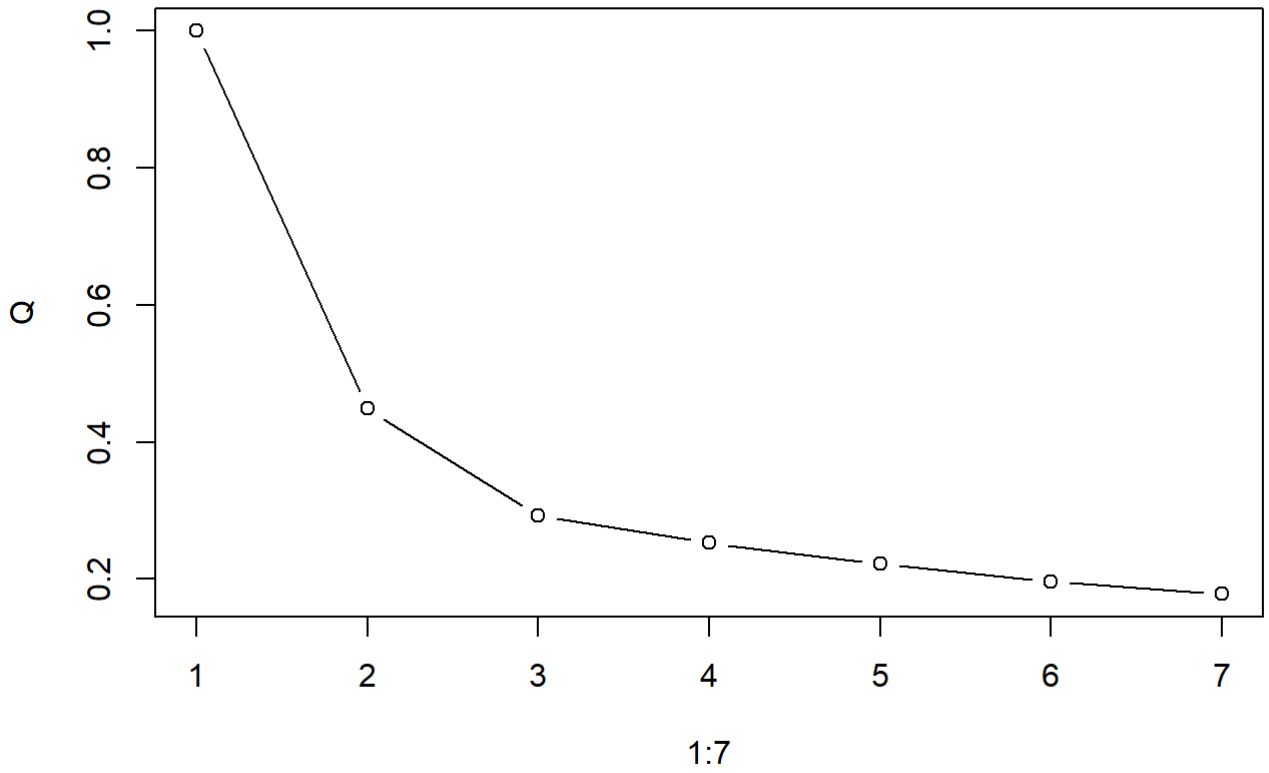
Estimation of the number of clusters

Exercise 4: Elbow plot

```
Q <- c()
for (i in 1:7){
  kres <- kmeans(seeds_std[,-8], centers = i, nstart = 20)

  Q <- c(Q, sum(kres$withinss)/kres$totss)
}

plot(1:7, Q, type = "b")
```



We can observe that according to the elbow plot, it would be best to use a number of clusters of K = 3, as it is the region where the elbow is generated in our graph.

BIC based estimation

```
library("mclust")
```

```
## Warning: package 'mclust' was built under R version 4.0.4
```

```
## Package 'mclust' version 5.4.7
## Type 'citation("mclust")' for citing this R package in publications.
```

```
seeds_clust <- mclustBIC(as.matrix(seeds_std[,-8]), G=1:15)
seeds_clust
```

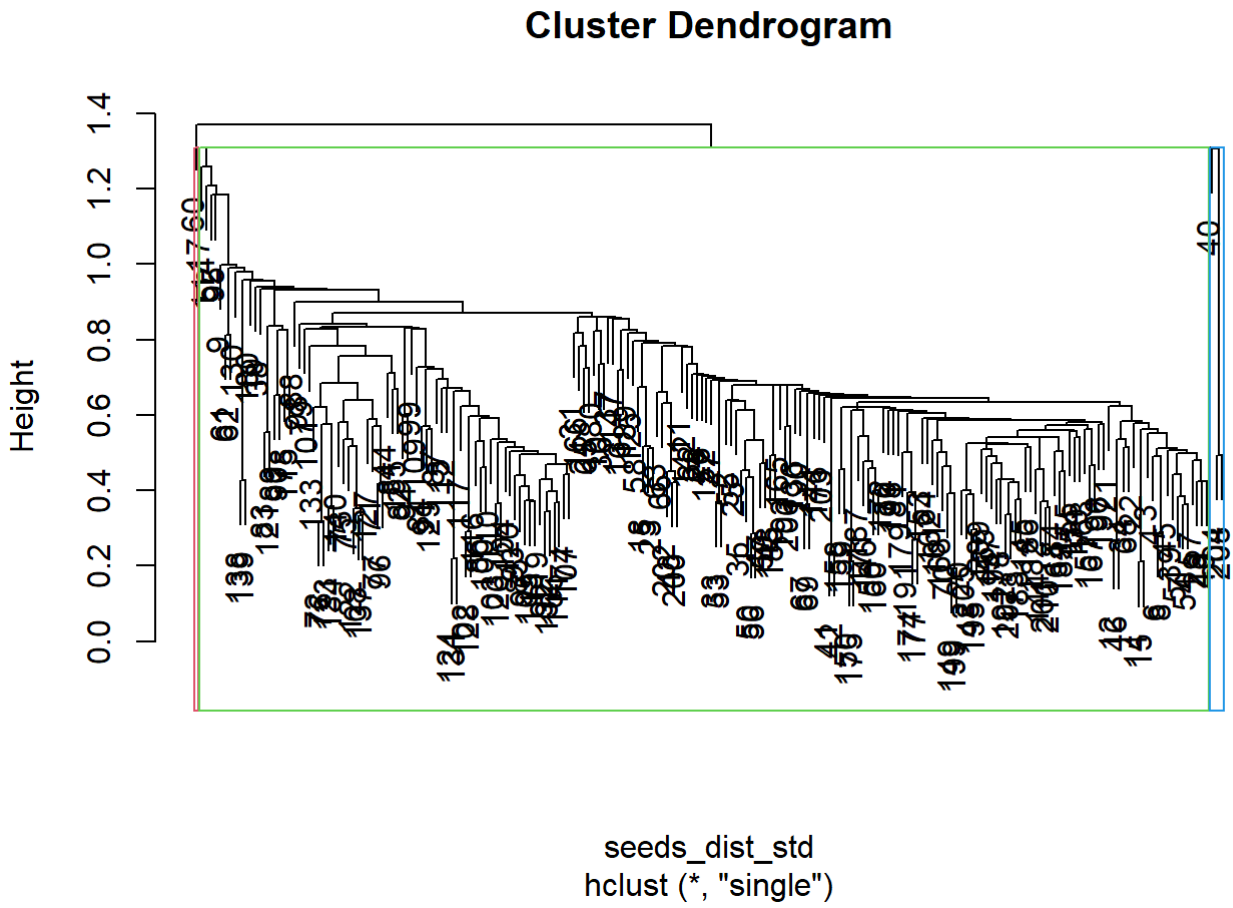
```
## Bayesian Information Criterion (BIC):
##          EII       VII       EEI       VEI       EVI       VVI       EEE
## 1  -4207.439 -4207.439 -4239.522 -4239.522 -4239.522 -4239.522 -655.0563
## 2  -3341.146 -3317.526 -3093.535 -3092.954 -3085.009 -3085.705 -549.0782
## 3  -2934.649 -2942.454 -2560.571 -2567.698 -2563.989 -2570.416 -452.8676
## 4  -2901.331 -2865.454 -2369.840 -2415.592 -2471.739 -2422.290 -422.7040
## 5  -2818.872 -2831.942 -2214.689 -2213.732 -2284.578 -2279.709 -420.6622
## 6  -2731.207 -2790.655 -2160.301 -2205.421 -2249.258 -2239.673 -314.1551
## 7  -2713.226 -2738.029 -2172.620 -2131.242 -2280.968 -2258.361 -323.1411
## 8  -2713.582 -2729.679 -2156.815 -2097.146 -2293.171 -2220.858 -336.3693
## 9  -2648.766 -2642.725 -1967.198 -1986.881 -2181.201 -2152.338 -234.4494
## 10 -2607.912 -2591.223 -1926.588 -1938.212 -2157.135 -2176.742 -233.2278
## 11 -2618.992 -2599.619 -1889.106 -1933.261 -2196.154 -2178.956 -204.3423
## 12 -2609.613 -2601.041 -1862.152 -1909.336 -2224.222 -2188.579 -202.3357
## 13 -2569.697 -2610.020 -1860.532 -1973.373 -2173.932 -2188.876 -189.8296
## 14 -2584.156 -2589.774 -1864.672 -1880.470 -2191.065 -2219.711 -230.3442
## 15 -2578.034 -2587.493 -1807.488 -1887.766 -2209.016 -2274.941 -294.6001
##          VEE       EVE       VVE        EEV        VEV        EVV        VVV
## 1  -655.0563 -655.0563 -655.0563 -655.05629 -655.05629 -655.05629 -655.05629
## 2  -547.5151 -419.4098 -412.2689 -120.67100 -160.45379 -142.53307 -169.13516
## 3  -451.3637 -344.7995 -331.6507   48.81966   66.00606   25.43750   40.09185
## 4  -413.4475 -300.0975 -343.8112  179.98388    3.16270   17.27806  -30.01995
## 5  -328.3832 -352.5601 -317.5233  -68.70442  -85.43907 -132.17808 -119.84940
## 6  -310.2890 -377.7045 -384.6290  -14.77903  -83.00226 -110.48030 -169.21284
## 7  -355.7992        NA        NA -138.62133  -88.05384         NA         NA
## 8  -389.7339        NA        NA -256.39338 -206.40674         NA         NA
## 9  -240.8010        NA        NA -229.57178 -208.64111         NA         NA
## 10 -248.6622        NA        NA -342.91543 -312.34104         NA         NA
## 11 -247.8138        NA        NA -443.13639 -401.88123         NA         NA
## 12 -287.0793        NA        NA -582.39274 -541.09421         NA         NA
## 13 -349.0127        NA        NA -694.74947 -550.53645         NA         NA
## 14        NA        NA        NA -711.36525 -783.54474         NA         NA
## 15        NA        NA        NA -730.96638 -812.19489         NA         NA
##
## Top 3 models based on the BIC criterion:
##     EEV,4     VEV,3     EEV,3
## 179.98388  66.00606  48.81966
```

In this case, the best model will be the one that has the highest BIC. From our observation, it appears that the best model is the EEV model with a number of clusters K = 4.

Hierarchical methods

Single linkage

```
# Group assignment
seeds_single = hclust(seeds_dist_std, method = "single")
# Cut dendogram in k=3 groups.
groups_single = cutree(seeds_single, k = 3)
# Representamos el dendograma
plot(seeds_single)
# Plot the groups in dendogram
rect.hclust(seeds_single, k = 3, border=2:4)
```




```
# Comparisson with original classifier
seeds_single_stats <- cluster.stats(d = seeds_dist_std,
seeds$species_class, groups_single)
seeds_single_stats$corrected.rand
```

```
## [1] 3.981971e-06
```

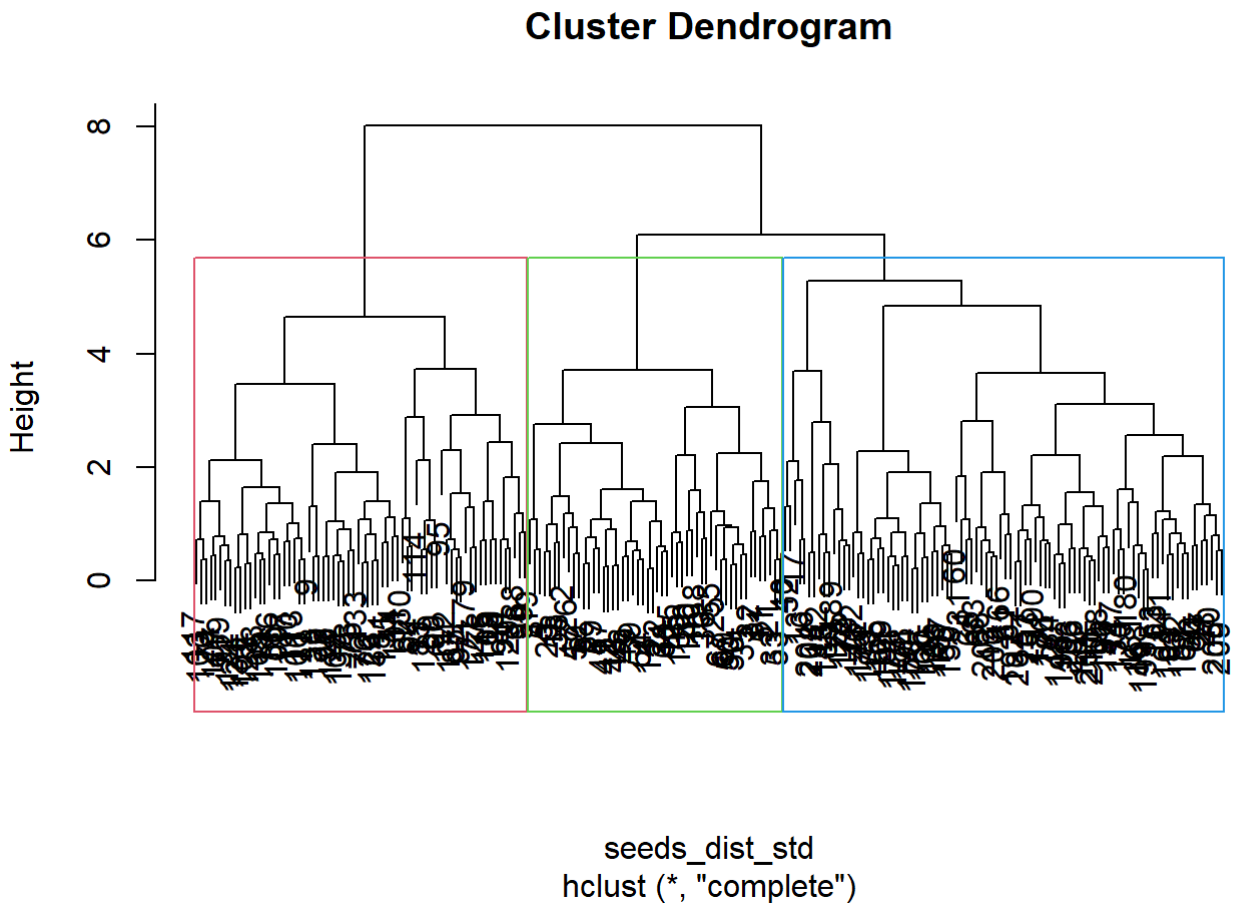
```
single_tb <- table(groups_single,seeds$species_class,dnn=list("single","actual"))
single_tb
```

```
##       actual
## single  1  2  3
##      1 68 70 68
##      2  1  0  2
##      3  1  0  0
```

As we can observe, we obtain a very small Rand index, and most of the instances are clustered into a single group. This indicates that the complete linkage method of clustering is not performing a correct clustering, as we have a significant number of errors.

Exercise 5: Complete linkage

```
seeds_complete = hclust(seeds_dist_std, method = "complete")
# Cut dendogram in K=3 groups.
groups_complete = cutree(seeds_complete, k = 3)
# Plot dendogram
plot(seeds_complete)
# Plot groups
rect.hclust(seeds_complete, k = 3, border=2:4)
```




```
# Compare with initial classifier
seeds_complete_stats <- cluster.stats(d = seeds_dist_std,
seeds$species_class, groups_complete)
seeds_complete_stats$corrected.rand
```

```
## [1] 0.6862626
```

```
complete_tb <- table(groups_complete,seeds$species_class,dnn=list("Complete","actua
l"))
complete_tb
```

```
##         actual
## Complete  1  2  3
##        1 48  4  0
##        2  2 66  0
##        3 20  0 70
```

In this case, we can observe that the Rand index is much closer to the desired value of 1. This indicates that the complete linkage method is performing a more reliable clustering and making fewer errors compared to the single linkage method.

Exercise 6: Single vs Complete

```
#Error rate single linkage
error_rate <- sum(sum(single_tb-diag(diag(single_tb))))/sum(colSums(single_tb))
error_rate
```

```
## [1] 0.6761905
```

```
#Error rate complete linkage
error_rate <- sum(sum(complete_tb-diag(diag(complete_tb))))/sum(colSums(complete_tb))
error_rate
```

```
## [1] 0.1238095
```

In addition to the previously calculated Rand indices, we can calculate the error rates of these methods and observe that the nearest neighbors method has a very high error rate. This is because it is clustering almost all instances into a single group. On the other hand, the furthest neighbors method is much more accurate, with an error rate of around 12%. Therefore, we can conclude that the furthest neighbors method is a better approach for clustering our dataset.

Exercise 7: K-means vs Farthest Neighbors

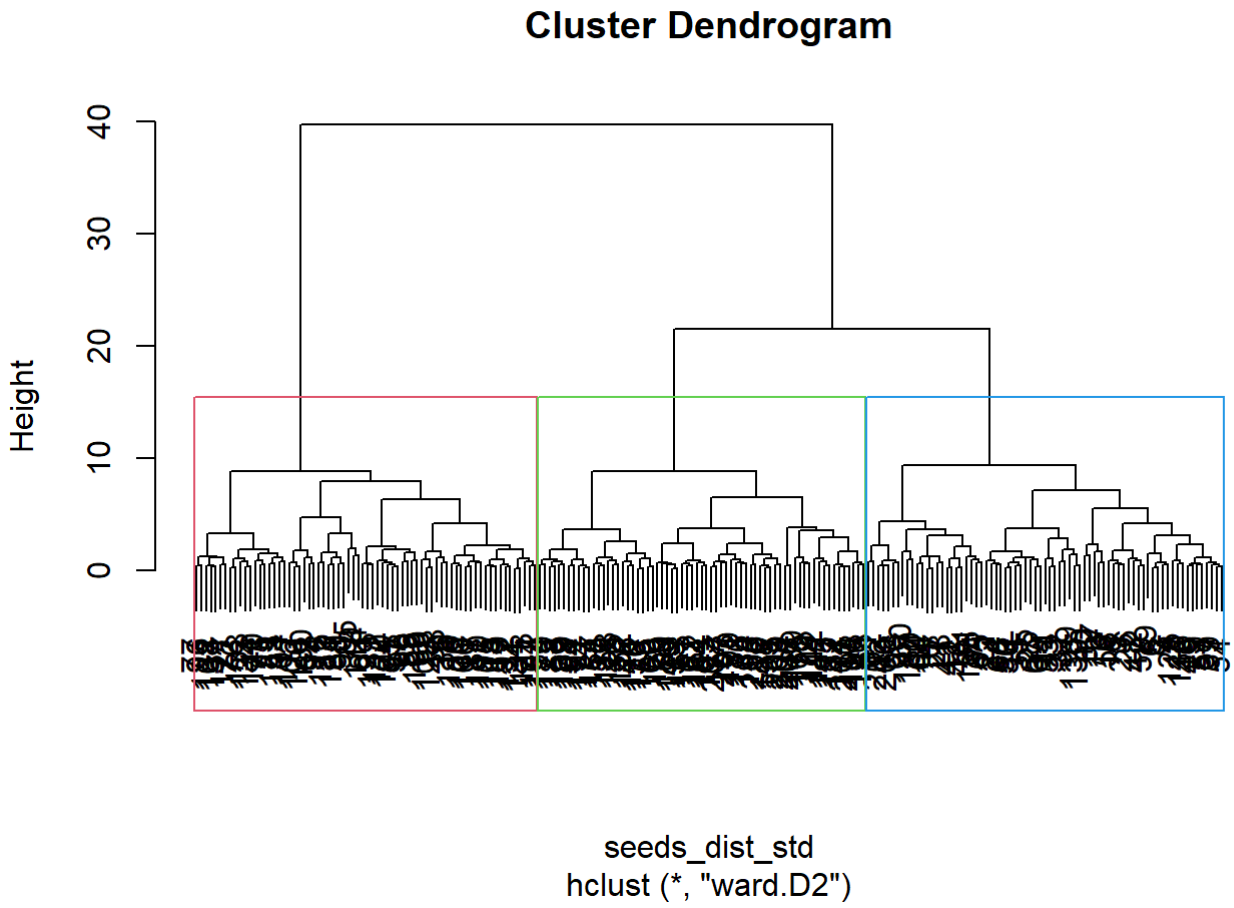
```
#Error rate k-means standardized
error_rate <- sum(sum(tb_std-diag(diag(tb_std))))/sum(colSums(tb_std))
error_rate
```

```
## [1] 0.08095238
```

In this case, if we examine the Rand indices and error rates of these methods, we can observe that the K-means method produces better clustering results than the furthest neighbors method. The Rand index for the K-means method is 0.77, while for the furthest neighbors method, it is 0.68, indicating that K-means performs better. Additionally, the error rates also reflect the superiority of the K-means method, with an error rate of around 8%, whereas the furthest neighbors method has an error rate of 12%.

Ward Method

```
seeds_ward = hclust(seeds_dist_std, method = "ward.D2")
# Cut dendogram in K=3 groups.
groups_ward  = cutree(seeds_ward , k = 3)
# Plot dendogram
plot(seeds_ward )
# Plot groups
rect.hclust(seeds_ward , k = 3, border=2:4)
```




```
# Compare to initial classifier
seeds_ward_stats <- cluster.stats(d = seeds_dist_std,
seeds$species_class, groups_ward)
seeds_ward_stats$corrected.rand
```

```
## [1] 0.7969983
```

```
ward_tb <- table(groups_ward,seeds$species_class,dnn=list("Ward","actual"))
ward_tb
```

```
##     actual
## Ward  1  2  3
##    1 64  4  5
##    2  4 66  0
##    3  2  0 65
```

```
error_rate <- sum(sum(ward_tb-diag(diag(ward_tb))))/sum(colSums(ward_tb))
error_rate
```

```
## [1] 0.07142857
```

For the Ward method, we observe the highest adjusted Rand index, which is closest to 1, as well as the lowest error rate among all the methods. Therefore, we can conclude that this method provides the best clustering of our data and its clustering closely resembles the known grouping.

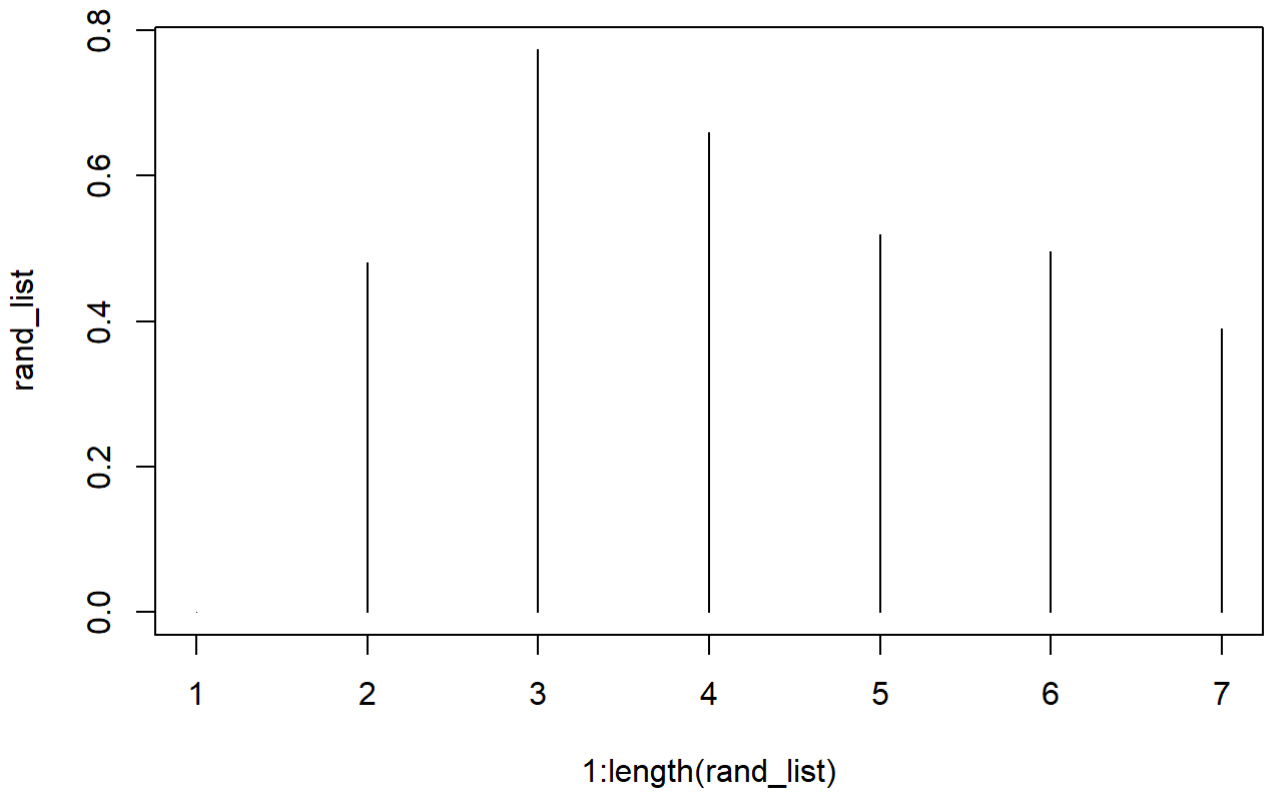
Extra excercises

Exercise 1: Indices calculations

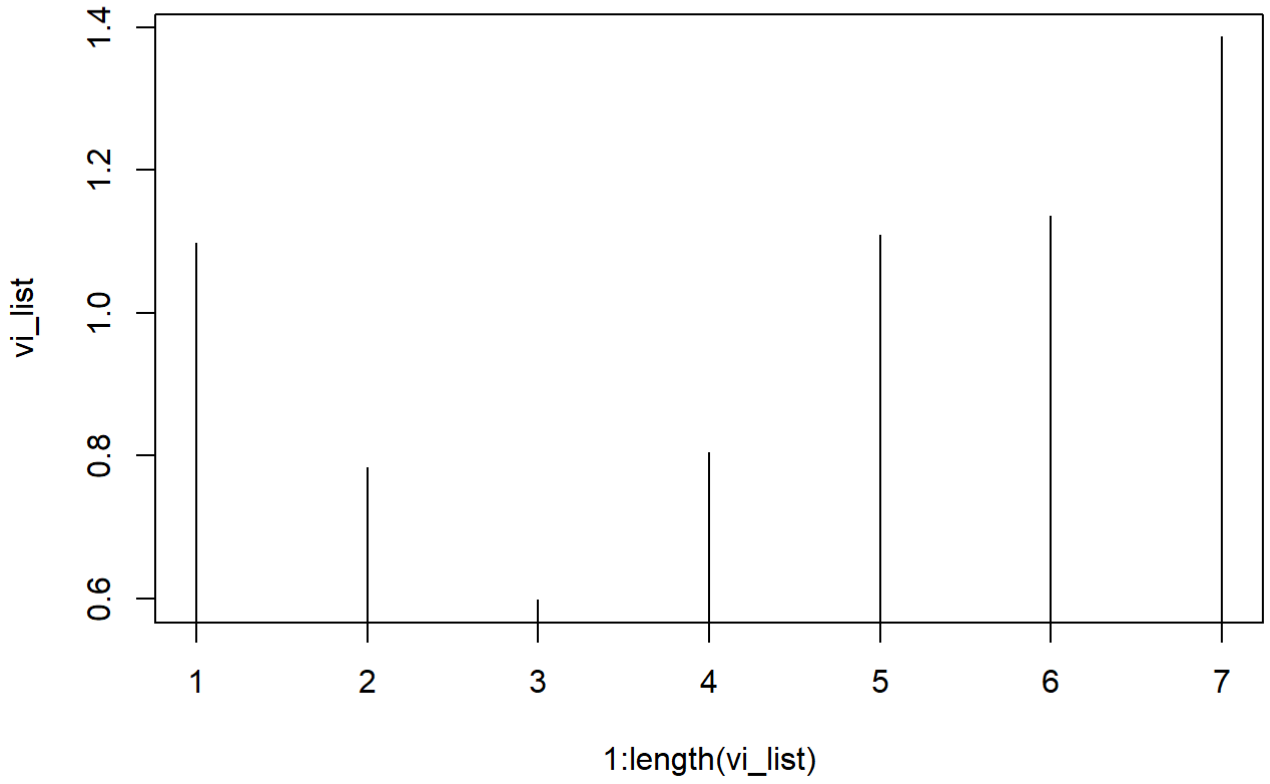
```
rand_list <- list()
vi_list <- list()
for (i in 1:7) {
  seeds_km_std <- kmeans(seeds_std[,-8], centers = i, nstart = 20)

  # Calculate distances
  seeds_dist_std = dist(seeds_std[,-8], method = "euclidean")
  # Calculate means
  clust_stats <- cluster.stats(d = seeds_dist_std, seeds_std$seeds.species_class,
  seeds_km_std$cluster)
  rand_list[i] <- clust_stats$corrected.rand
  vi_list[i] <- clust_stats$vi
}

plot(1:length(rand_list), rand_list, type = "h")
```




```
plot(1:length(vi_list), vi_list, type = "h")
```



We can observe that the optimal values of these indices, a Rand index close to 1 and a lower VI index, are obtained when the number of clusters is K=3. This aligns with the prior knowledge obtained from the elbow plot.

Exercise 2: Compactivity elimination

```
#New data set
seeds_noc <- seeds_std[,-3]
seeds_noc
```

area <dbl>	perimeter <dbl>	length <dbl>	width <dbl>	asymmetry <dbl>	groove_length <dbl>
0.141759037	0.2149488188	0.303493006	0.141364035	-0.983800962	-0.38266305
0.011161356	0.0082041534	-0.168222697	0.196961591	-1.783903583	-0.91981560
-0.191608729	-0.3593419185	-0.761817099	0.207551602	-0.665888201	-1.18635720
-0.346263878	-0.4742000660	-0.687335672	0.318746714	-0.958527563	-1.22705057
0.444195774	0.3298069663	0.066506648	0.803239702	-1.559768435	-0.47422315
-0.160677699	-0.2674554005	-0.547400870	0.141364035	-0.823514403	-0.91981560
-0.054137485	-0.0530535253	-0.147909581	0.001046394	-0.075953850	-0.38469772
-0.253470789	-0.3516847087	-0.470662431	0.114889009	-0.665223111	-0.83029017
0.612598048	0.6896958284	0.958026757	0.546431943	-1.104182155	0.95411430

area <dbl>	perimeter <dbl>	length <dbl>	width <dbl>	asymmetry <dbl>	groove_length <dbl>
0.547299207	0.5288944219	0.576591571	0.652332050	-1.151403506	0.25418826

1-10 of 210 rows | 1-6 of 7 columns Previous **1** 2 3 4 5 6 … 21 Next

```
#Groups calculation
seeds_noc_km <- kmeans(seeds_noc[,-7], centers = 3, nstart = 20)
seeds_km_std <- kmeans(seeds_std[,-8], centers = 3, nstart = 20)

tb_noc <- table(seeds_noc_km$cluster, seeds_km_std$cluster)

fst <- which.max(c(tb_noc [1,1],tb_noc [2,1],tb_noc  [3,1]))
scd <- which.max(c(tb_noc  [1,2],tb_noc  [2,2],tb_noc  [3,2]))
thd <- which.max(c(tb_noc  [1,3],tb_noc  [2,3],tb_noc  [3,3]))

#Confusion Matrix
tb_noc  <- tb_noc [c(fst, scd, thd),]
tb_noc
```

```
##
##      1  2  3
##   2 67  1  0
##   1  0 66  3
##   3  0  4 69
```

```
#Error rate
error_rate <- sum(sum(ward_tb-diag(diag(ward_tb))))/sum(colSums(ward_tb))
error_rate
```

```
## [1] 0.07142857
```

As we can observe, the elimination of the "compactness" attribute does not result in a significant change in the obtained clustering. There is only a 7% difference in the clustering of instances, which could be attributed to the method itself rather than the data.

```
# Distance Calculation
seeds_dist_noc = dist(seeds_noc[,-7], method = "euclidean")
# Rand index
clust_stats <- cluster.stats(d = seeds_dist_noc, seeds_noc$seeds.species_class,
seeds_noc_km$cluster)
clust_stats$corrected.rand
```

```
## [1] 0.7510454
```

Additionally, the Rand index for the clustering without the "compactness" attribute is very similar to the one obtained using this attribute. However, it is slightly lower, suggesting that the "compactness" attribute may provide some additional information, although the contribution seems to be minimal.