Multivariate Data Analysis

Assignment #8

고려대학교 공과대학

산업경영공학부

2017170857 이우준

[Q1] 데이터 셋 선정하기

분류의 주 목적은, 비슷한 특징의 item들을 묶는 것이다. 이를 적용할 수 있는 여러 예시 중 하나는 바로 생물의 종의 분류이다. 생물의 종은 여러 요소를 통해 결정되지만, 기본적으론 생김새의 유사성을 통해 결정된다. 이번 과제에 생물의 종 분류를 clustering을 통해 구현해보고 싶어, Target Data로 Seed Data를 골랐다. 해당 데이터의 각 Instance는 콩의 생김새의 관한 변수(A (area), P (perimeter), C (compactness) , LK (length of kernel), WK (width of kernel), A_Coef (asymmetry coefficient), LKG (length of kernel groove))와 정답 class인 target으로 이루어져 있다. 앞 7가지의 입력변수들은 콩의 형태에 관한 데이터로 이를 이용해 clustering을 해 생물의 종분류를 간단히 구현해보고 싶었다. 또한 seed의 종류를 분류한 결과가 얼마나 정답 class와 유사한지 purity를 통하여 알아볼 수 있어서 해당 데이터 셋을 선정하였다.

선정한 데이터 셋: Seed_from_UCI, https://www.kaggle.com/dongeorge/seed-from-uci

■ 데이터 전처리

본격적인 Clustering을 진행하기 전, 데이터의 전처리 작업을 진행하였다. Class 정보를 담고 있는target 변수를 따로 seed_class라는 변수에 할당해 주었고, 남은 7가지의 설명변수를 seed_x라는 변수에 할당해주었다.

```
# Load dataset
seed <- read.csv("Seed_Data.csv")
#class 값 제거
seed_class<-seed[,8]
seed_x<-seed[,-8]

#scaling
seed_x_scaled <- scale(seed_x, center = TRUE, scale = TRUE)
```

[Q2] K-Means Clustering (최적의 K 찾기)

```
> summary(seed_clValid_k)
Clustering Methods:
 kmeans
Cluster sizes:
 2 3 4 5
Validation Measures:
                        2       3       4       5

kmeans APN           0.0351  0.0976  0.1641  0.2036
       AD            2.3330  1.9462  1.8844  1.7667
       ADM           0.1632  0.3088  0.4636  0.5015
       FOM           0.6714  0.5707  0.5655  0.5236
       Connectivity 16.8964 39.5476 63.6409 79.9706
       Dunn          0.0870  0.1188  0.0888  0.0814
       Silhouette    0.4658  0.4007  0.3379  0.2881
Optimal Scores:
            Score   Method Clusters
APN          0.0351 kmeans 2
AD           1.7667 kmeans 5
ADM          0.1632 kmeans 2
FOM          0.5236 kmeans 5
Connectivity 16.8964 kmeans 2
Dunn         0.1188 kmeans 3
Silhouette   0.4658 kmeans 2
```

소요시간은 다음과 같다.

사용자	시스템	elapsed
0.17	0.00	0.18

데이터 셋의 크기도 적을 뿐더러, 군집의 개수 또한, 2~5개로 적어 계산시간은 약 0.18초밖에 소요되지 않았다.

	Dunn	Silhouette
2	0.070	0.4658
3	0.1188	0.4007
4	0.0888	0.3379
5	0.0814	0.2881

위의 표를 통해서 군집의 개수가 2부터 10일 때까지 1씩 늘리며 타당성 지표를 확인할 수 있다. Dunn Index는 minimum distance에 대한 maximum diameter로 정의한다. Silhouette Index는 각 군집 내의 객체에 대해 distance를 고려하는 a(i)와 군집에 속하지 않는 객체들의 distance를 고려하는 b(i)를 모두 고려한다. K가 3일때가 K가 2일 때보다 Dunn Index Scored은 약 17배 크고, Silhouette Index의 Score은 0.8배 작으므로, 군집의 개수 K가 3일때가 최적의 군집 수라고 할 수 있다.

[Q3] 최적의 K-Means Clustering (10회 반복)

Q2에서 정한 최적의 군집 수, K가 3일때의 K-Means Clustering을 10회 반복한 결과 각 Cluster별 입력변수별 Center는 아래의 표와 같다.

1회 수행의 center (K=3)			
	Cluster 1	**Cluster 2**	**Cluster 3**
A	-0.209250872	1.223597957	-1.041430964
P	-0.244381675	1.232802856	-1.011132901
C	0.420948265	0.531454099	-1.033961405
LK	-0.340026522	1.217027169	-0.885952811
WK	-0.053245061	1.127693072	-1.118446097
A_Coef	-0.658907422	-0.046680275	0.797560539
LKG	-0.662769115	1.277154818	-0.582060589

2회 수행의 center (K=3)			
	Cluster 1	**Cluster 2**	**Cluster 3**
A	1.25368596	-0.140783093	-1.027796663
P	1.25895795	-0.169637241	-1.004249146
C	0.559128334	0.448534632	-0.962604962
LK	1.234931926	-0.257199869	-0.895545115
WK	1.162075101	0.001643014	-1.082995635
A_Coef	-0.045111567	-0.660340792	0.693148212
LKG	1.289227274	-0.58449646	-0.623319149

3회 수행의 center (K=3)			
	Cluster 1	**Cluster 2**	**Cluster 3**
A	-1.041430964	1.223597957	-0.209250872
P	-1.011132901	1.232802856	-0.244381675

C	-1.033961405	0.531454099	0.420948265
LK	-0.885952811	1.217027169	-0.340026522
WK	-1.118446097	1.127693072	-0.053245061
A_Coef	0.797560539	-0.046680275	-0.658907422
LKG	-0.582060589	1.277154818	-0.662769115

4회 수행의 center (K=3)			
	Cluster 1	Cluster 2	Cluster 3
A	1.25368596	-0.140783093	-1.027796663
P	1.25895795	-0.169637241	-1.004249146
C	0.559128334	0.448534632	-0.962604962
LK	1.234931926	-0.257199869	-0.895545115
WK	1.162075101	0.001643014	-1.082995635
A_Coef	-0.045111567	-0.660340792	0.693148212
LKG	1.289227274	-0.58449646	-0.623319149

5회 수행의 center (K=3)			
	Cluster 1	Cluster 2	Cluster 3
A	-0.209250872	1.223597957	-1.041430964
P	-0.244381675	1.232802856	-1.011132901
C	0.420948265	0.531454099	-1.033961405
LK	-0.340026522	1.217027169	-0.885952811
WK	-0.053245061	1.127693072	-1.118446097
A_Coef	-0.658907422	-0.046680275	0.797560539
LKG	-0.662769115	1.277154818	-0.582060589

6회 수행의 center (K=3)			
	Cluster 1	Cluster 2	Cluster 3
A	1.25368596	-1.027796663	-0.140783093
P	1.25895795	-1.004249146	-0.169637241
C	0.559128334	-0.962604962	0.448534632
LK	1.234931926	-0.895545115	-0.257199869
WK	1.162075101	-1.082995635	0.001643014
A_Coef	-0.045111567	0.693148212	-0.660340792
LKG	1.289227274	-0.623319149	-0.58449646

7회 수행의 center (K=3)			
	Cluster 1	Cluster 2	Cluster 3
A	-1.041430964	-0.209250872	1.223597957
P	-1.011132901	-0.244381675	1.232802856
C	-1.033961405	0.420948265	0.531454099

LK	-0.885952811	-0.340026522	1.217027169
WK	-1.118446097	-0.053245061	1.127693072
A_Coef	0.797560539	-0.658907422	-0.046680275
LKG	-0.582060589	-0.662769115	1.277154818

8회 수행의 center (K=3)			
	Cluster 1	Cluster 2	Cluster 3
A	-0.140783093	-1.027796663	1.25368596
P	-0.169637241	-1.004249146	1.25895795
C	0.448534632	-0.962604962	0.559128334
LK	-0.257199869	-0.895545115	1.234931926
WK	0.001643014	-1.082995635	1.162075101
A_Coef	-0.660340792	0.693148212	-0.045111567
LKG	-0.58449646	-0.623319149	1.289227274

9회 수행의 center (K=3)			
	Cluster 1	Cluster 2	Cluster 3
A	-0.209250872	-1.041430964	1.223597957
P	-0.244381675	-1.011132901	1.232802856
C	0.420948265	-1.033961405	0.531454099
LK	-0.340026522	-0.885952811	1.217027169
WK	-0.053245061	-1.118446097	1.127693072
A_Coef	-0.658907422	0.797560539	-0.046680275
LKG	-0.662769115	-0.582060589	1.277154818

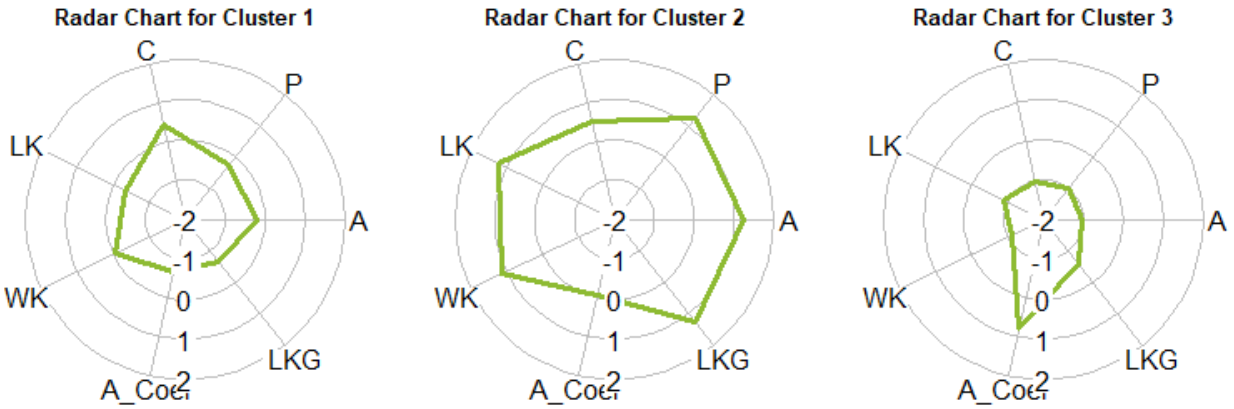
10회 수행의 center (K=3)			
	Cluster 1	Cluster 2	Cluster 3
A	1.25368596	-1.027796663	-0.140783093
P	1.25895795	-1.004249146	-0.169637241
C	0.559128334	-0.962604962	0.448534632
LK	1.234931926	-0.895545115	-0.257199869
WK	1.162075101	-1.082995635	0.001643014
A_Coef	-0.045111567	0.693148212	-0.660340792
LKG	1.289227274	-0.623319149	-0.58449646

10회 반복 결과 1회차 수행의 center와 같은 경우가 5번 (1, 3, 5, 7, 9), 2회차 수행의 center와 같은 경우가 5번 (2, 4, 6, 8, 10회 나왔다. 즉 총 2가지의 군집화 결과가 나왔으며, 각 경우마다 5회, 5회 동일하게 발생하였다.

[Q4] K-Means Clustering (K=3 일 때) Radar Chart 도시와 군집 분석

Q3에서 도출한 cluster은 두가지의 경우가 나왔다. 두가지 cluster는 모두 다 같은 횟수 검출되어 두가지 경우 중 1회차 수행결과로 나온 cluster로 임의로 선정하였다.

첫번째로 도시한 Radar Chart는 Q3에서 구한 1회차의 경우이다.



1회 수행의 center (K=3)			
	Cluster 1	Cluster 2	Cluster 3
A	-0.209250872	1.223597957	-1.041430964
P	-0.244381675	1.232802856	-1.011132901
C	0.420948265	0.531454099	-1.033961405
LK	-0.340026522	1.217027169	-0.885952811
WK	-0.053245061	1.127693072	-1.118446097
A_Coef	-0.658907422	-0.046680275	0.797560539
LKG	-0.662769115	1.277154818	-0.582060589

1번 군집의 경우 C, P, A, LK, WK 변수들의 평균값이 0에 , A_Coef와 LKG 변수의 평균이 -1에 가까운 것을 알 수 있습니다. 2번 군집 같은 경우 A_coef, C를 제외한 모든 변수들의 평균이 1에 가까운 것을 알 수 있습니다. 3번 군집 같은 경우는 A_coef를 제외한 모든 변수들의 평균이 -1에 가깝다는 것을 알 수 있습니다.

이를 토대로 가장 유사할 것 같은 군집은 1번 군집과, 3번 군집, 가장 상이할 것으로 보이는 군집은 2번 군집과 3번 군집이라고 판단하였습니다. 그 이유는 Radar Chart를 보았을 때 Radar Chart가 차지하는 범위의 차가 2번과 3번 군집에서 가장 크게 나타나고, 1번과 3번 군집에서 가장 작게 나타나 이와 같이 판단하였습니다.

[Q5] 유사 군집, 상이 군집 변수 별 평균값 차이에 대한 통계적 검정

유사 군집과 상이 군집들 간의 변수 별 평균값 차이에 대한 통계적 검정을 시행하기 위해 각 군집 별 t-test를 진행하였다. 귀무가설을 채택하는 경우 군집들 간의 평균이 같다는 결론을 내릴 수 있고, 대립가설을 채택하는 경우 양측검정인지, 단측검정 중 greater인지 less인지에 따라 결론이 달라지게 된다. 유의수준은 0.05로 설정하였다

	H_0	H_1
Two-Sided	군집 A의 평균 = 군집 B의 평균	군집 A의 평균 ≠ 군집 B의 평균
Greater		군집 A의 평균 > 군집 B의 평균
Less		군집 A의 평균 < 군집 B의 평균

첫번째로 유사 군집 (1번, 3번 군집)들 간의 평균값 차이에 대한 통계적 검정을 시행하였다.

```
> result1
         two.sided     greater        less
A      1.550631e-29 7.753154e-30 1.000000e+00
P      4.218148e-24 2.109074e-24 1.000000e+00
C      2.932528e-21 1.466264e-21 1.000000e+00
LK     5.598985e-12 2.799493e-12 1.000000e+00
WK     1.387356e-32 6.936778e-33 1.000000e+00
A_Coef 1.386128e-19 1.000000e+00 6.930640e-20
LKG    2.626686e-01 8.686657e-01 1.313343e-01
```

위 결과를 통해 1번과 3번 군집들 간의 각 변수들의 평균값의 관계는 아래의 표와 같이 나타낼 수 있다.

	Cluster 1		Cluster 3
A	Cluster 1의 평균	>	Cluster 3의 평균
P		>	
C		>	
LK		>	
WK		>	
A_Coef		<	
LKG		>	

이를 통해 유사 군집(1번 군집, 3번 군집)의 변수 중에서 유의수준 0.05에서 값의 차이가 나타나는 변수의 비율은 100%인 것을 알 수 있다. 7가지의 변수 모두 1번 군집과 3번 군집의 평균 간의 차이가 있다는 것을 t-test를 통해 알 수 있다.

다음으로 상이 군집 (2번, 3번 군집)들 간의 평균값 차이에 대한 통계적 검정을 시행하였다.

```
> result2
    two.sided      greater        less
1 7.527780e-60 3.763890e-60 1.000000e+00
2 7.622378e-64 3.811189e-64 1.000000e+00
3 2.407638e-23 1.203819e-23 1.000000e+00
4 1.064757e-51 5.323786e-52 1.000000e+00
5 7.640471e-61 3.820235e-61 1.000000e+00
6 1.649572e-08 1.000000e+00 8.247861e-09
7 3.625867e-53 1.812933e-53 1.000000e+00
```

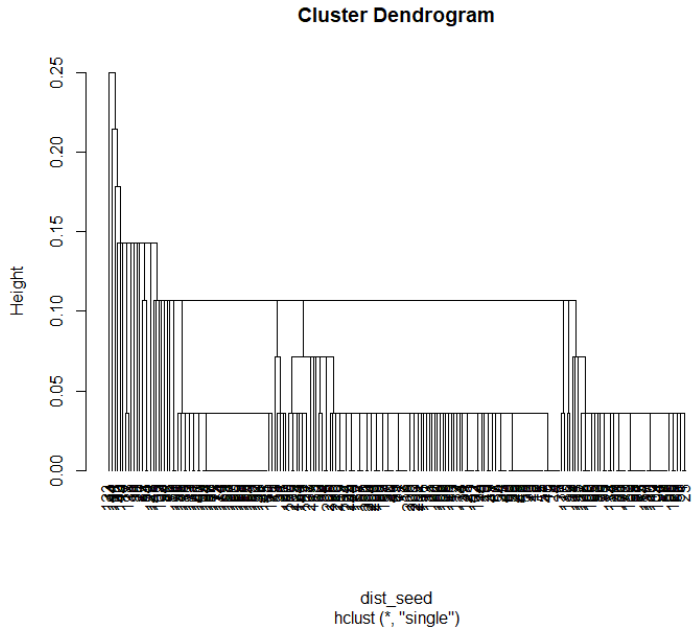
위 결과를 통해 1번과 3번 군집들 간의 각 변수들의 평균값의 관계는 아래의 표와 같이 나타낼 수 있다.

	Cluster 2		Cluster 3
A	Cluster 2의 평균	>	Cluster 3의 평균
P		>	
C		>	
LK		>	
WK		>	
A_Coef		<	
LKG		>	

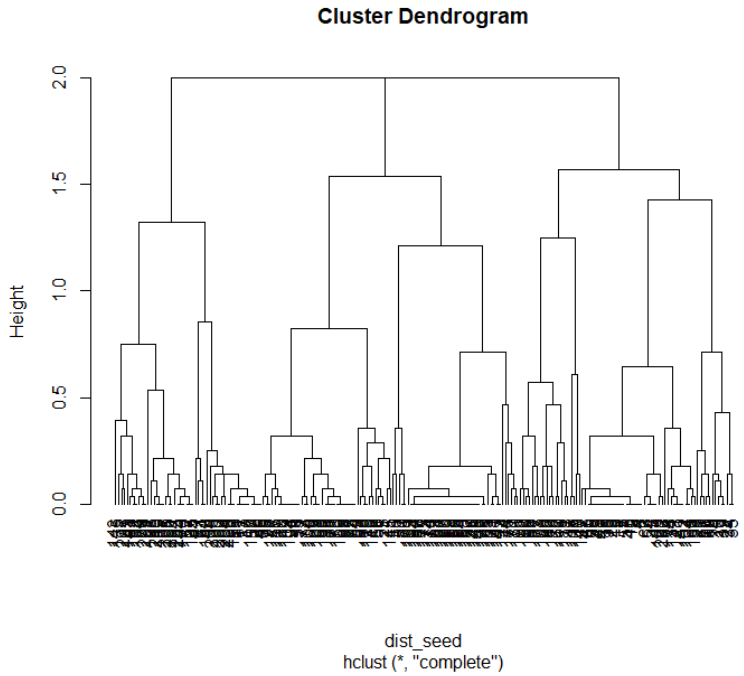
이를 통해 유사 군집(2번 군집, 3번 군집)의 변수 중에서 유의수준 0.05에서 값의 차이가 나타나는 변수의 비율은 100%인 것을 알 수 있다. 7가지의 변수 모두 2번 군집과 3번 군집의 평균 간의 차이가 있다는 것을 t-test를 통해 알 수 있다.

[Q6] Hierarchical Clustering Dendrogram 그리고 비교하기

다음은 hclust() 함수의 method를 single로 지정하여 도시한 dendrogram이다.



다음은 hclust() 함수의 method를 complete 로 지정하여 도시한 dendrogram이다.

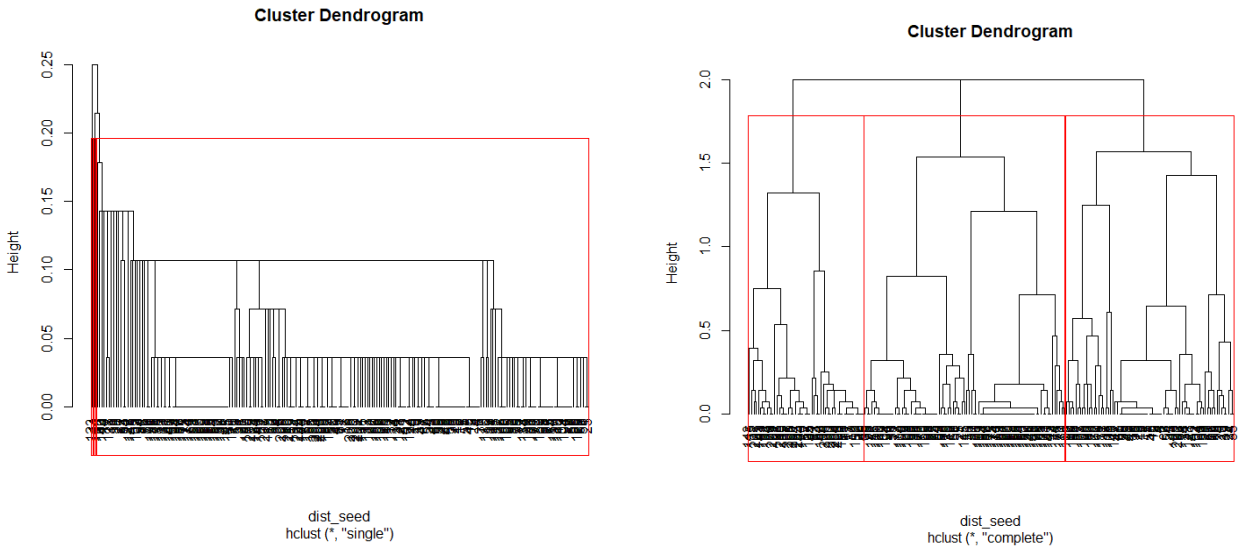


두 dendrogram을 비교해 볼 때, method를 complete로 지정하여 도시한 dendrogram이 single로 지정했

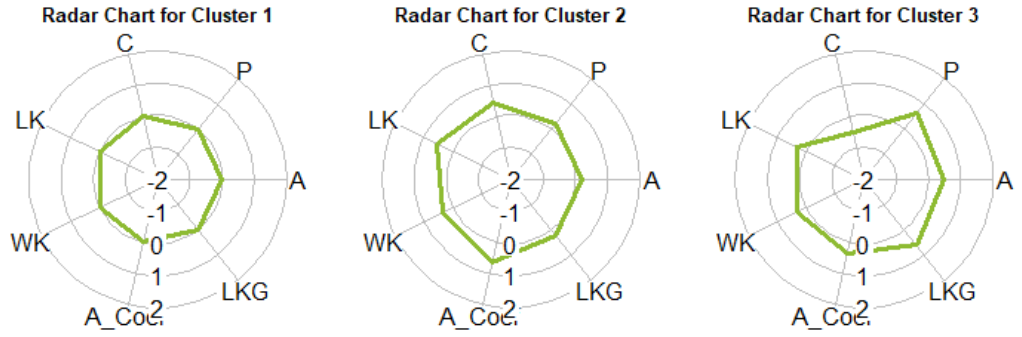
을 때 보다 상위 cluster 간의 거리가 더 명확하게 떨어져 있는 것을 볼 수 있다. 또한 전체적인 개형을 보았을 때 complete method로 진행한 hierarchical clustering의 결과가 더 cluster의 분리가 잘 된 것을 dendrogram의 개형을 통해 알 수 있다.

[Q7] Hierarchical Clustering, Radar Chart 도시

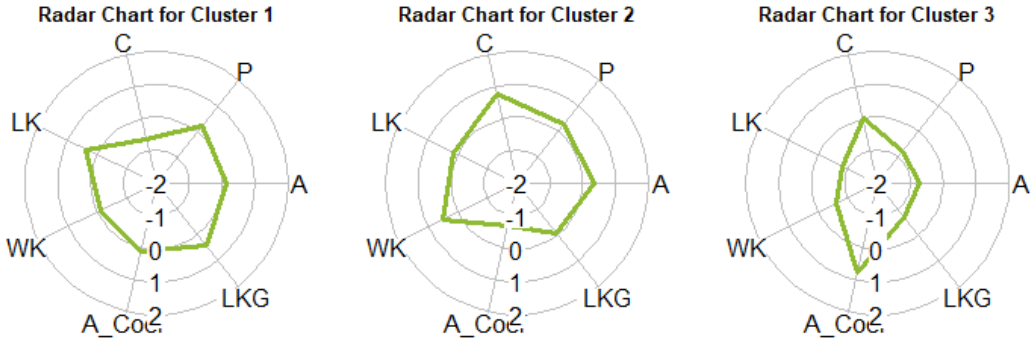
각 Linkage로부터 생성된 Dendrogram으로부터 선정한 최적 개수의 군집을 적용하면 다음과 같다.



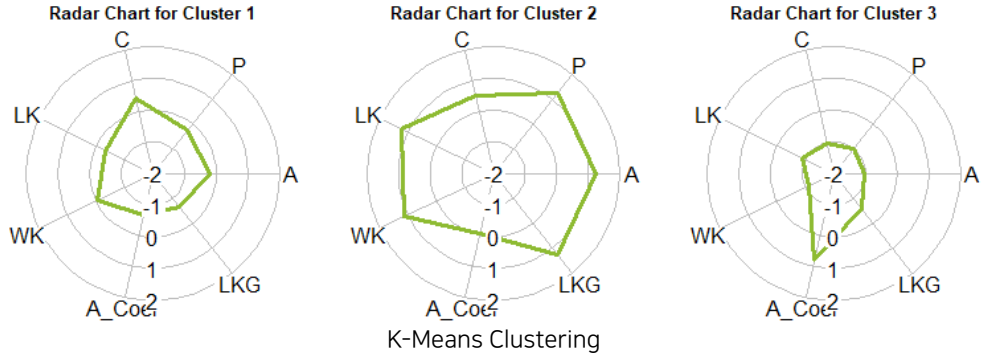
이를 바탕으로 Radar Chart를 도사하면 아래 그림과 같은 결과를 구할 수 있다.



Method = "single"



Method = "complete"



K-Means Clustering

K-means clustering으로 구한 Radar chart가 위와 같다는 것을 바탕으로 개형 비교를 해보면 method가 complete인 것으로 진행한 계층적 군집화 모형이 K – means clustering과 유사하다고 판단하였다. Cluster 3의 좁은 Radar Chart개형이 single로 진행한 계층적 군집화 모형에서는 찾아볼 수 없지만, complete로 진행한 모형에서는 찾아볼 수 있기 때문이다.

[Q8] Hierarchical Clustering, K-Means Clustering 유사도 정량적 측정 아이디어.

Single Linkage와 Complete Linkage 중 K-means clustering과 유사한지 정량적으로 측정할 수 있는 방법은 다음과 같다. 각 구현 모델의 cluster별 centroid 값들의 유사도 (Cosine Similarity)를 합하여 해당 유사도가 높을수록 더 유사한 Clustering이라고 정량적으로 판단할 수 있을 것이다. 이를 구현한 방법은 아래와 같다.

첫째, K-Means Clustering, Hierarchical Clustering의 cluster 별 centroid를 matrix 형태로 정리하였다.

```
hc_summary_complete=t(hc_summary_complete)
hc_summary_single=t(hc_summary_single)
rownames(kmc_centroid)<-c("cluster 1", "cluster 2", "cluster 3")
kmc_centroid
hc_summary_complete
hc_summary_single
```

```
> kmc_centroid
                A          P          C         LK          WK        A_Coef       LKG
cluster 1 -0.2092509 -0.2443817  0.4209483 -0.3400265 -0.05324506 -0.65890742 -0.6627691
cluster 2  1.2235980  1.2328029  0.5314541  1.2170272  1.12769307 -0.04668027  1.2771548
cluster 3 -1.0414310 -1.0111329 -1.0339614 -0.8859528 -1.11844610  0.79756054 -0.5820606
> hc_summary_complete
                A          P          C         LK          WK        A_Coef       LKG
cluster 1  0.1130007  0.2055313 -0.6611551  0.3630574 -0.1035756  0.1293753  0.45166806
cluster 2  0.3562518  0.2921503  0.7688375  0.1292702  0.5206823 -0.6908500 -0.05137439
cluster 3 -0.7167489 -0.7841639  0.0279071 -0.8204543 -0.5799745  0.7835280 -0.71089581
> hc_summary_single
                A          P          C         LK          WK        A_Coef       LKG
cluster 1 -0.0032632 -0.0041283  0.00024357 -0.0041443 -0.0033830 -0.0051051 -0.0040581
cluster 2  0.2242417  0.2302632  0.46558189  0.5607924  0.3637542  0.6722717  0.2440149
cluster 3  0.4545061  0.6284381 -0.51624513  0.3012359  0.3399267  0.3896087  0.6000819
```

다음 이러한 matrix들을 두가지 clustering 기법을 선택 후 각 clustering 기법으로 구한 3가지의 cluster 별 cosine 유사도들을 3X3 matrix로 나타낼 수 있는 함수를 정의하였다.

```
#K-Mean Clustering 와 Hierarchical Clustering 의 cluster 별 코사인 유사도 matrix 구하는 함수
custom_simliarity<-function(a,b){
  resultmat<-matrix(NA,nrow(a),nrow(b))
  for (i in 1:nrow(a)){
    for (j in 1:nrow(b)){
      tmpmat<-rbind(a[i,],b[j,])
```

```
      resultmat[i,j]<-as.matrix(dist(tmpmat, method = "cosine"))[1,2]
    }
  }
  return(resultmat)
}
```

이를 이용한 결과는 아래와 같다.

```
> kmc_X_complete<-custom_simliarity(kmc_centroid,hc_summary_complete)
> kmc_X_single<-custom_simliarity(kmc_centroid,hc_summary_single)
> kmc_X_complete
          [,1]      [,2]      [,3]
[1,] 0.1807986 0.5790103 0.6964496
[2,] 0.6349018 0.4486730 0.1155420
[3,] 0.9967770 0.1406076 0.1258472
> kmc_X_single
          [,1]      [,2]      [,3]
[1,] 0.1588236 0.4339728 0.1265579
[2,] 0.1746141 0.3049420 0.2683284
[3,] 0.4717616 0.4818170 0.5975894
```

kmc_X_complete		Hierarchical Clustering (complete)		
		1	2	3
K-Means Clustering	1	0.1807986	0.5790103	0.6964496
	2	0.6349018	0.4486730	0.1155420
	3	0.9967770	0.1406076	0.1258472

kmc_X_single		Hierarchical Clustering (single)		
		1	2	3
K-Means Clustering	1	0.1588236	0.4339728	0.1265579
	2	0.1746141	0.3049420	0.2683284
	3	0.4717616	0.4818170	0.5975894

3X3개의 유사도 중 각 Hierarchical Clustering의 cluster가 K-means Clustering의 cluster와 어떻게 1대1로 매치되는지 알아보기 위하여 permutation 함수를 통하여 6가지의 경우의 cosine 유사도의 합을 계산해, 가장 높은 값을 가지는 조합을 알아보았다.

```
permute<-permutations(3,3,1:3)
bestclustermatch<-function(c){
  maxtmp=0
  pertmp=NA
  for (i in 1:6){
    sumtmp<-c[permute[i,1],1]+c[permute[i,2],2]+c[permute[i,3],3]
    if (sumtmp>=maxtmp){
      maxtmp<-sumtmp
      pertmp<-permute[i,]
    }
  }
  pertmp[4]=maxtmp
  return(pertmp)
}
```

```
bestclustermatch(kmc_X_complete)
bestclustermatch(kmc_X_single)
> bestclustermatch(kmc_X_complete)
[1] 3.0000 2.0000 1.0000 2.1419
> bestclustermatch(kmc_X_single)
[1] 2.000000 1.000000 3.000000 1.206176
```

이를 통해 cluster 별 유사한 cluster은 다음과 같다는 것을 알 수 있다.

kmc_X_complete		Hierarchical Clustering (complete)		
		1	2	3
K-Means Clustering	1	0.1807986	0.5790103	0.6964496
	2	0.6349018	0.4486730	0.1155420
	3	0.9967770	0.1406076	0.1258472
총 유사도의 합		2.1419		

kmc_X_single		Hierarchical Clustering (single)		
		1	2	3
K-Means Clustering	1	0.1588236	0.4339728	0.1265579
	2	0.1746141	0.3049420	0.2683284
	3	0.4717616	0.4818170	0.5975894
총 유사도의 합		1.206176		

위의 결과로 Complete Linkage를 사용한 군집 결과물이 Single Linkage 보다 K-means Clustering과 유사한 결과물이 나온다고 할 수 있다.

[Q9 & Q10] DBSCAN 최적 개수(3개)의 군집이 되도록 하는 eps, minPts의 조합 찾기 & Noise 찾기

```
ep <- seq(0,3,0.1)
mP <- seq(10,30,1)
DBSCAN_result_matrix<-data.frame(NA,(length(ep)*length(mP)),4)
DBSCAN_result_matrix
n<-1
for (i in 1:length(ep)) {
  for (j in 1:length(mP)){
    DBSCAN_seed = dbscan(seed_x_scaled, eps = ep[i], minPts = mP[j])
    num_cluster<-length(unique(DBSCAN_seed$cluster))
    num_noise<-length(which(DBSCAN_seed$cluster==0))
    if (num_noise!=0){
      num_cluster<-num_cluster-1
    }
    DBSCAN_result_matrix[n,c(1:4)]<-c(DBSCAN_seed$eps, DBSCAN_seed$minPts, num_cluster,
num_noise)
    DBSCAN_result_matrix[c(1:5),]
    n<-n+1
  }
}
colnames(DBSCAN_result_matrix)<-c("eps","minPoints","Number_of_Cluster", "Number_of_Noise")
DBSCAN_result_matrix
DBSCAN_result_matrix_only3<-DBSCAN_result_matrix[DBSCAN_result_matrix[,3]==3,]
DBSCAN_result_matrix_only3<-DBSCAN_result_matrix_only3[c(order(DBSCAN_result_matrix_only3[,4])),]
DBSCAN_result_matrix_only3
```

DBSCAN을 진행하기 위해 eps가 가능한 값을 1부터 3까지 0.1씩 증가하는 소수들로 설정하였고, minPts의 경우 10~30까지의 사이에 존재하는 자연수 값들로 설정하였다. 위의 결과로 Cluster의 개수가 3개가 나온 경우는 아래와 같다.

```
DBSCAN_result_matrix_only3
    eps minPoints Number_of_Cluster Number_of_Noise
238 1.1       16                3             39
239 1.1       17                3             42
213 1.0       12                3             54
214 1.0       13                3             59
190 0.9       10                3             76
243 1.1       21                3             78
272 1.2       29                3             79
216 1.0       15                3             83
244 1.1       22                3             83
191 0.9       11                3             86
245 1.1       23                3             89
192 0.9       12                3             90
217 1.0       16                3             92
193 0.9       13                3             96
218 1.0       17                3             96
246 1.1       24                3            101
219 1.0       18                3            103
194 0.9       14                3            104
220 1.0       19                3            106
169 0.8       10                3            118
221 1.0       20                3            120
195 0.9       15                3            130
196 0.9       16                3            156
197 0.9       17                3            162
171 0.8       12                3            166
```

총 26개의 조합 중 Noise의 개수가 가장 적으면서 클러스터의 개수가 3개로 나오는 eps와 minPoints의 값은 각각 1.1과 16인 것을 알 수 있다. 이때의 Noise로 판별된 객체의 수는 총 39개이다.

[Q11] 데이터셋에 가장 적합한 군집화 알고리즘은 무엇인가.

이번 과제의 분석 데이터셋으로 고른 Seed Data는 정답 class 값이 존재하는 데이터 셋이다. 각 기법으로 구한 clustering의 결과와, 정답 class간의 entropy 계산을 통해 Entropy가 가장 작은 clustering이 이 데이터 셋에 가장 적합한 군집화 알고리즘이라고 할 수 있다.

이를 위해 지금까지 구한 K-Means Clustering, Hierarchical Clustering, DBSCAN의 군집을 나타내보았다.

```
> #K-MEANS
> data_k3$cluster
  [1] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1
 [65] 1 1 1 1 1 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2
[129] 2 2 2 2 2 2 2 1 2 2 1 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3
3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3
[193] 3 3 3 1 3 1 3 1 3 1 3 3 3 1 3 3 3 3
> #Hierarchical Clustering
> mycl_complete
  [1] 1 2 2 2 2 2 2 2 1 2 1 2 3 2 2 3 3 2 2 3 1 2 2 2 1 2 3 2 2 1 2 2 1 2 2 2 2 2 2 3 2 2 2 3 2 2
2 2 2 1 3 3 1 2 1 1 2 2 2 2 3 3 3 3
 [65] 2 2 2 1 2 1 1 1 2 2 1 1 1 1 1 2 1 2 2 1 1 1 2 1 1 2 1 2 1 3 1 1 1 1 1 1 2 2 2 1 1 2 2 1 1 1
2 1 2 1 1 1 2 2 2 1 2 1 1 2 2 2 1 2
[129] 1 3 2 2 1 1 3 2 1 1 1 2 1 3 3 3 1 1 1 3 1 1 3 1 1 1 1 1 3 1 1 1 1 3 1 3 3 2 1 1 1 1 1 3 1 1
1 3 1 1 3 1 3 3 3 3 1 1 1 1 3 3 1 3
[193] 3 1 3 3 3 3 1 3 3 2 3 3 3 3 3 3 3 3
> DBSCAN_seed$cluster
  [1] 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1
1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 0 1
```

```
 [65] 1 0 1 1 1 1 2 2 2 3 2 2 2 0 0 2 2 0 0 3 3 3 3 3 0 0 3 3 3 0 0 2 3 3 3 3 2 3 3 3 3 3 2 3 3
3 3 3 0 0 3 3 3 3 3 0 2 0 3 0 3 3 3
[129] 3 0 3 3 2 2 2 1 2 0 0 2 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1
[193] 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 1 1
```

이에 대응되는 original class는 아래와 같다.

```
#Original Class
> seed_class
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [64] 0 0 0 0 0 0 0 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 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
[127] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[190] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Original Class를 통해 1번부터 70번까지의 데이터는 0번 cluster (숫자는 의미가 없다), 71번부터 140번까지 1번 cluster, 141번부터 210번까지 2번클러스터 총 세가지로 나타나는 것을 알 수 있다. 이에 각 구간별로 cluster의 Entropy를 구하여 더해주었다. 이에 앞서 Entropy를 구하는 함수를 정의하였다.

```
#entropy 함수 정의
entropy <- function(target) {
  freq <- table(target)/length(target)
  # vectorize
  vec <- as.data.frame(freq)[,2]
  #drop 0 to avoid NaN resulting from log2
  vec<-vec[vec>0]
  #compute entropy
  -sum(vec * log2(vec))
}
```

위 함수를 통해 각 Clustering 기법 별 entropy를 구하면 결과는 아래와 같다.

```
#Entropy
KMC_entropy<-
entropy(data_k3$cluster[1:70])+entropy(data_k3$cluster[71:140])+entropy(data_k3$cluster[141:210])
HC_entropy<-
entropy(mycl_complete[1:70])+entropy(mycl_complete[71:140])+entropy(mycl_complete[141:210])
DB_entropy<-
entropy(DBSCAN_seed$cluster[1:70])+entropy(DBSCAN_seed$cluster[71:140])+entropy(DBSCAN_seed$clust
er[141:210])
KMC_entropy
HC_entropy
DB_entropy
> KMC_entropy
[1] 1.164824
> HC_entropy
[1] 3.668022
> DB_entropy
[1] 2.838945
```

K-Means Clustering 기법을 통해 구현한 Clustering의 Entropy 값은 1.164824, Hierarchical Clustering을 통해 구현한 Clustering의 Entropy 값은 3.668022, DBSCAN을 이용해 구한 Clustering의 Entropy 값은 2.838945가 나왔다. Entropy의 값이 낮을수록 purity가 높으므로 세개의 기법 중 K-Means Clustering 기법으로 구한 cluster들이 가장 purity가 높다고 할 수 있다. 이를 통해 이 데이터셋에 가장 적합한 군집화 알고리즘은 K- Means Clustering이라고 할 수 있다.