다변량분석 8번째 과제

Assignment 8

Clustering

학과 산업경영공학부

학번 2016170863

이름 추창욱

[Question 1]

제가 이번 clustering을 하기 위해 선택한 데이터는 모바일 레전드라는 게임 속 캐릭터들에 대한 데이터 셋입니다. 제가 이 데이터 셋을 선택한 이유는 정말 순수하게 재미있을 것 같아서 입니다. 저는 모바일 게임을 매우 좋아합니다. 특히나 롤과 같은 AOS 게임 장르를 매우 좋아하고 실제로 제가 이전에 즐겨 하였던 게임 중 하나가 모바일 레전드라는 게임입니다. 또한 게임을 하면서 한번쯤 들었던 의문은 이러한 장르의 챔피언들은 각자 라인과 각자의 역할이 팀 내에 있는데, 부여되는 능력치를 통해 이를 구분할 수 있을까 아니면 스킬 구성의 덕이 더 클까 였습니다. 제가 고른 데이터 속에는 스킬 구성을 제외한 여러 수치적인 정보들이 들어 있고 이에 따라 과연 태생적인 능력치는 어느 정도로 캐릭터들의 라인을 잘 구분할 수 있을 것인가 에 대한 답을 얻을 수 있을 것 같아 고르게 되었습니다.

데이터셋 링크: https://www.kaggle.com/christina0512/mobile-legends-bang-bang-mlbb-hero-dataset?rvi=1

[K-Means Clustering]

[Question2]

군집 분석을 실행하기 전 몇가지 전 처리 과정을 거쳐 주었습니다. 우선 가장 먼저 해준 것은 타겟 변수에 해당하는 role안의 값을 확인한 후 이 값들을 조금 수정해주었습니다. 원본 데이터에서는 fighter, assassin, mage 등등 string 형태로 직업을 나타내 주고 있는 추후의 작업들을 더 편하게 해주기 위해 'assassin' = 1, 'fighter' = 2, 'mage' = 3, 'marksman' = 4, 'support' = 5, 'tank' = 6으로 치환 해 주었습니다. 이후 타겟에 해당하는 role이라는 칼럼을 hero_class에 넣어주고 hero_x는 그 3~18번 까지의 칼럼을 넣어주었습니다. 원래 총 20개의 칼럼이 들어 있는데 첫번째 칼럼인 캐릭터 고유의 이름과 마지막 칼럼인 캐릭터 출시일은 제외하고 넣어 분석을 진행하였습니다. 이후 스케일링을 해주었습니다.

군집화를 하는 방법 중 K-means Clustering은 미리 정해 놓은 군집의 프로토타입에

각 개체가 얼마나 유사한가를 가지고 군집을 형성하는 기법 중 한가지 입니다. 즉 군집을 미리 정해주어야 하기 때문에, clvalid()의 방법을 kmeans를 넣어준 후 2:10가지의 타당성 지표를 산출하였습니다. Dunn과 Silhouette의 두가지 지표를 이용하였는데 Dunn index는 경우는 clustering의 유효성을 검증하기 위한 값으로 1에 가까울수록 가장 작은 클러스터 간의 거리가 클러스터 내의 가장 먼 거리보다 길다고 할 수 있어 유용한 클러스터 이고 실루엣 지표는 개체별로 적합성을 평가할 수 있고 이 또한 값이 값이 클수록 좋습니다.

```
Validation Measures:
                            2       3       4       5       6       7       8       9      10

kmeans APN             0.0047  0.0697  0.0878  0.1580  0.2010  0.2150  0.2681  0.1996  0.2079
       AD              5.2565  5.1065  4.9272  4.4578  4.3123  4.1165  4.0385  3.8917  3.7933
       ADM             0.0324  1.4349  0.3164  0.6346  1.0340  0.9753  1.0073  1.0617  0.9998
       FOM             0.9971  0.9907  0.9882  0.9408  0.9277  0.9124  0.8995  0.8921  0.9001
       Connectivity    2.9290 38.8325 12.5528 44.5425 59.2270 62.2004 57.6274 83.5925 86.5425
       Dunn            0.6145  0.2195  0.4601  0.2249  0.2845  0.2893  0.2744  0.2894  0.2894
       Silhouette      0.4207  0.1739  0.2763  0.1897  0.1775  0.1928  0.1929  0.1680  0.1740

Optimal Scores:

             Score  Method Clusters
APN          0.0047 kmeans 2
AD           3.7933 kmeans 10
ADM          0.0324 kmeans 2
FOM          0.8921 kmeans 9
Connectivity 2.9290 kmeans 2
Dunn         0.6145 kmeans 2
Silhouette   0.4207 kmeans 2
```

위의 결과에서 각 군집의 개수 마다 Dunn과 실루엣에 대한 값들이 나온 것을 확인할 수 있었고 가장 높은 값을 가지는 최적의 군집 수는 각각 Dunn index: 2, silhouette: 2가 나왔다는 것을 확인할 수 있었습니다. 군집의 크기에 상관없이 지표 값 들이 나왔다는 것을 확인할 수 있었습니다. 하지만 현재 가장 높은 결과값을 산출해낸 군집의 개수가 2이기 때문에 2를 사용하는 것이 맞긴 하지만 조사를 좀 해본 결과 많은 경우에서 2가 최적의 군집수로 도출되는 경우가 빈번하게 일어나 아예 2를 최적의 군집으로 생각하지 않고 다음으로 높게 나온 결과물을 확인하는 경우가 많다고 합니다. 그렇기에 저는 두번째로 큰 값이 4를 이용하여 다음에 나오는 질문들에 답변하고자 합니다.

```
사용자   시스템 elapsed
  2.71    0.00    2.83
```

이 작업을 하는데 걸린 시간은 2.83초 정도 였다는 것을 확인할 수 있었습니다.

[Question3]

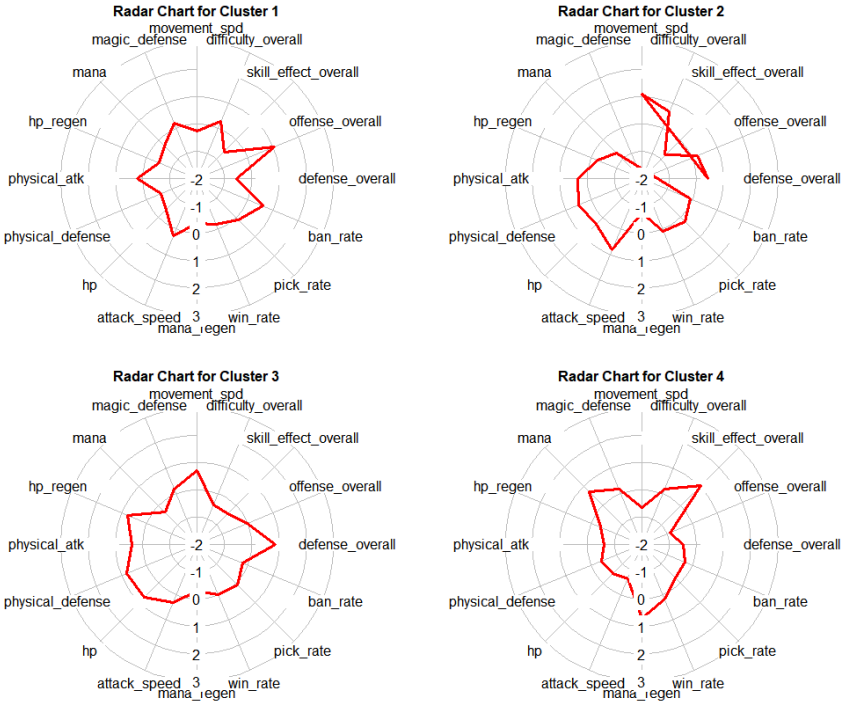
우선 두번째 질문에서 최적의 군집의 수가 2로 나왔다는 것을 확인하였기 때문에, 군집의 특성을 알아보기 위하여 for 반복문을 통해 생성된 군집의 각 군집의 중심과 군집의 개체수 변화를 알아보았습니다.

```
Number of Iteraion: 1
  defense_overall offense_overall skill_effect_overall difficulty_overall movement_spd magic_defense       mana   hp_regen physical_atk physical_defense         hp attack_speed
1       0.4973405       0.2891109           -0.5564484         -0.1072584    0.6543931    -0.1745711 -0.4515592  0.4072537    0.4716353        0.4304943  0.4244557    0.4474410
2      -0.5698693      -0.3312730            0.6375971          0.1229003   -0.7498255     0.2000294  0.5174116 -0.4666448   -0.5404154       -0.4932747 -0.4863555   -0.5126929
  mana_regen   win_rate   pick_rate   ban_rate
1 -0.4357662 -0.1363172  0.05520733  0.1750812
2  0.4993154  0.1561967 -0.06325839 -0.2006138
[1] 55 48
Number of Iteraion: 2
  defense_overall offense_overall skill_effect_overall difficulty_overall movement_spd magic_defense       mana   hp_regen physical_atk physical_defense         hp attack_speed
1      -0.5698693      -0.3312730            0.6375971          0.1229003   -0.7498255     0.2000294  0.5174116 -0.4666448   -0.5404154       -0.4932747 -0.4863555   -0.5126929
2       0.4973405       0.2891109           -0.5564484         -0.1072584    0.6543931    -0.1745711 -0.4515592  0.4072537    0.4716353        0.4304943  0.4244557    0.4474410
  mana_regen   win_rate   pick_rate   ban_rate
1  0.4993154  0.1561967 -0.06325839 -0.2006138
2 -0.4357662 -0.1363172  0.05520733  0.1750812
[1] 48 55
Number of Iteraion: 3
  defense_overall offense_overall skill_effect_overall difficulty_overall movement_spd magic_defense       mana   hp_regen physical_atk physical_defense         hp attack_speed
1      -0.5698693      -0.3312730            0.6375971          0.1229003   -0.7498255     0.2000294  0.5174116 -0.4666448   -0.5404154       -0.4932747 -0.4863555   -0.5126929
2       0.4973405       0.2891109           -0.5564484         -0.1072584    0.6543931    -0.1745711 -0.4515592  0.4072537    0.4716353        0.4304943  0.4244557    0.4474410
  mana_regen   win_rate   pick_rate   ban_rate
1  0.4993154  0.1561967 -0.06325839 -0.2006138
2 -0.4357662 -0.1363172  0.05520733  0.1750812
[1] 48 55
Number of Iteraion: 4
  defense_overall offense_overall skill_effect_overall difficulty_overall movement_spd magic_defense       mana   hp_regen physical_atk physical_defense         hp attack_speed
1       0.4973405       0.2891109           -0.5564484         -0.1072584    0.6543931    -0.1745711 -0.4515592  0.4072537    0.4716353        0.4304943  0.4244557    0.4474410
2      -0.5698693      -0.3312730            0.6375971          0.1229003   -0.7498255     0.2000294  0.5174116 -0.4666448   -0.5404154       -0.4932747 -0.4863555   -0.5126929
  mana_regen   win_rate   pick_rate   ban_rate
1 -0.4357662 -0.1363172  0.05520733  0.1750812
2  0.4993154  0.1561967 -0.06325839 -0.2006138
[1] 55 48
Number of Iteraion: 5
  defense_overall offense_overall skill_effect_overall difficulty_overall movement_spd magic_defense       mana   hp_regen physical_atk physical_defense         hp attack_speed
1      -0.5698693      -0.3312730            0.6375971          0.1229003   -0.7498255     0.2000294  0.5174116 -0.4666448   -0.5404154       -0.4932747 -0.4863555   -0.5126929
2       0.4973405       0.2891109           -0.5564484         -0.1072584    0.6543931    -0.1745711 -0.4515592  0.4072537    0.4716353        0.4304943  0.4244557    0.4474410
  mana_regen   win_rate   pick_rate   ban_rate
1  0.4993154  0.1561967 -0.06325839 -0.2006138
2 -0.4357662 -0.1363172  0.05520733  0.1750812
[1] 48 55
Number of Iteraion: 6
  defense_overall offense_overall skill_effect_overall difficulty_overall movement_spd magic_defense       mana   hp_regen physical_atk physical_defense         hp attack_speed
1       0.4973405       0.2891109           -0.5564484         -0.1072584    0.6543931    -0.1745711 -0.4515592  0.4072537    0.4716353        0.4304943  0.4244557    0.4474410
2      -0.5698693      -0.3312730            0.6375971          0.1229003   -0.7498255     0.2000294  0.5174116 -0.4666448   -0.5404154       -0.4932747 -0.4863555   -0.5126929
  mana_regen   win_rate   pick_rate   ban_rate
1 -0.4357662 -0.1363172  0.05520733  0.1750812
2  0.4993154  0.1561967 -0.06325839 -0.2006138
[1] 55 48
Number of Iteraion: 7
  defense_overall offense_overall skill_effect_overall difficulty_overall movement_spd magic_defense       mana   hp_regen physical_atk physical_defense         hp attack_speed
1       0.4973405       0.2891109           -0.5564484         -0.1072584    0.6543931    -0.1745711 -0.4515592  0.4072537    0.4716353        0.4304943  0.4244557    0.4474410
2      -0.5698693      -0.3312730            0.6375971          0.1229003   -0.7498255     0.2000294  0.5174116 -0.4666448   -0.5404154       -0.4932747 -0.4863555   -0.5126929
  mana_regen   win_rate   pick_rate   ban_rate
1 -0.4357662 -0.1363172  0.05520733  0.1750812
2  0.4993154  0.1561967 -0.06325839 -0.2006138
[1] 55 48
Number of Iteraion: 8
  defense_overall offense_overall skill_effect_overall difficulty_overall movement_spd magic_defense       mana   hp_regen physical_atk physical_defense         hp attack_speed
1      -0.5698693      -0.3312730            0.6375971          0.1229003   -0.7498255     0.2000294  0.5174116 -0.4666448   -0.5404154       -0.4932747 -0.4863555   -0.5126929
2       0.4973405       0.2891109           -0.5564484         -0.1072584    0.6543931    -0.1745711 -0.4515592  0.4072537    0.4716353        0.4304943  0.4244557    0.4474410
  mana_regen   win_rate   pick_rate   ban_rate
1  0.4993154  0.1561967 -0.06325839 -0.2006138
2 -0.4357662 -0.1363172  0.05520733  0.1750812
[1] 48 55
Number of Iteraion: 9
  defense_overall offense_overall skill_effect_overall difficulty_overall movement_spd magic_defense       mana   hp_regen physical_atk physical_defense         hp attack_speed
1      -0.5698693      -0.3312730            0.6375971          0.1229003   -0.7498255     0.2000294  0.5174116 -0.4666448   -0.5404154       -0.4932747 -0.4863555   -0.5126929
2       0.4973405       0.2891109           -0.5564484         -0.1072584    0.6543931    -0.1745711 -0.4515592  0.4072537    0.4716353        0.4304943  0.4244557    0.4474410
  mana_regen   win_rate   pick_rate   ban_rate
1  0.4993154  0.1561967 -0.06325839 -0.2006138
2 -0.4357662 -0.1363172  0.05520733  0.1750812
[1] 48 55
Number of Iteraion: 10
  defense_overall offense_overall skill_effect_overall difficulty_overall movement_spd magic_defense       mana   hp_regen physical_atk physical_defense         hp attack_speed
1       0.4973405       0.2891109           -0.5564484         -0.1072584    0.6543931    -0.1745711 -0.4515592  0.4072537    0.4716353        0.4304943  0.4244557    0.4474410
2      -0.5698693      -0.3312730            0.6375971          0.1229003   -0.7498255     0.2000294  0.5174116 -0.4666448   -0.5404154       -0.4932747 -0.4863555   -0.5126929
  mana_regen   win_rate   pick_rate   ban_rate
1 -0.4357662 -0.1363172  0.05520733  0.1750812
2  0.4993154  0.1561967 -0.06325839 -0.2006138
[1] 55 48
```

진행한 모든 iteration의 결과는 위와 같이 나왔는데, 군집의 중심 좌표는 변하는 것이 없다는 것을 확인할 수 있었습니다. 군집의 개수는 같은 경우 55/ 48로 나뉘어 지거나 48/55로 나뉘어 지는 것처럼 numbering만 달라 졌고, 군집 자체가 가지고 있는 특징 값들은 일정하다는 것을 알 수 있었습니다. 또한 처음부터 나온 값이 10번을 반복해서 돌렸을 때 돌려도 계속 같은 군집화가 일어났다는 것을 알 수 있었습니다.

다음으로는 위에서 언급한 것과 같이 개인적으로 생각한 최적의 군집 개수인 4를 이용하여 다시 반복문을 돌려 결과를 확인해보았습니다. 이 경우 결과는 따로 첨부 하지 않고 결과에 대해서만 설명을 하도록 하겠습니다. 우선 각 변수들의 중심을 확인해본 결과 2개 때와는 다르게 중심 값들이 다르게 나온다는 것을 확인할 수 있었습니다. 군집의 사이즈가 달라진다는 것을 유추할 수 있었고 확인을 해본 결과 [20 37 28 18], [1 30 35 37], [35 4 39 25], [21 38 13 31], [9 35 27 32], [39 25 35 4], [27 18 24 34]라는 값들이 나왔는데, 아무래도 두번째로 높은 군집 개수를 이용하고 kmeans 특성상 시작점에 따라 결과가 달라질 수 있기에 이렇게 다양한 결과가 나왔을 거라고 생각합니다.

[Question4]



Radar chart는 각 군집의 변수들의 평균을 이용하여 그래프를 그리는 방법입니다. 우선 군집의 개수를 정의하고 k-means clustering을 진행한 후에 각 변수의 평균값을 구하게 됩니다. 이 평균값들의 점을 찍고 선으로 잇게 되는데 k=4로 하였을 때 위와 같은 결과가 나왔습니다. 군집 간의 유사도를 radar chart의 크기와 모양으로 간접적으로 판별할 수 있는데, 모양과 크기가 비슷하면 유사한 cluster라고 할 수 있습니다. 제가 생각하기에 유사할 것 같다고 예상되는 그래프 즉, X는 cluster1, Y는 cluster4가 될 것 같다고 생각합니다. 그 이유는 크기가 매우 비슷하고 디펜스 hp 즉 왼쪽 아래 부분이 비교적 넓지 않다는 점에서 비슷할 것이라고 생각하였고, 완전 상이할 것이라고 생각하는 군집은 cluster2이고 cluster4와 상대적으로 가장 다르다고 생각합니다. 그 이유는 우선 한눈에 보아도 cluster2의 모양이 다른 여느 그래프들과는 확연하게 차이가 있고 cluster 2의 경우는 왼쪽 아래부분에 완전히 치우쳐져 있는 듯한 모습을 보이고 있으나 cluster4의 경우는 상단 부분에 치우쳐져 있다는 것을 확인할 수 있었기 때문입니다.

[Question5]

다음으로는 문제 4에서 선정된 비슷할 것 같은 군집과 완전 다를 것 같은 군집을 골랐었는데, 이들의 관계를 확인하기 위해 통계적 검정을 해보았습니다. 제가 사용한 검정은 변수 별 평균값 차이를 보아야 하기 때문에 paired t-test를 시행하였습니다. 하지만 이 과정을 시행하는 도중 완전히 동일한 값을 가지는 두 군집에 대한 에러가 자주 발생하여 이를 해결하기 위해 같은 값들을 가지는 변수들끼리 비교를 할 때 그에 해당하는 셀에 'None'을 입력하도록 하였습니다. 그 외에 본 문제는 몇가지의 군집만 가지고 비교를 하면 되지만 저는 총 4개로 비교할 조합이 얼마 되지 않기에 전부 수행하여 표로 정리하여 보았습니다. 결과는 다음과 같았습니다.

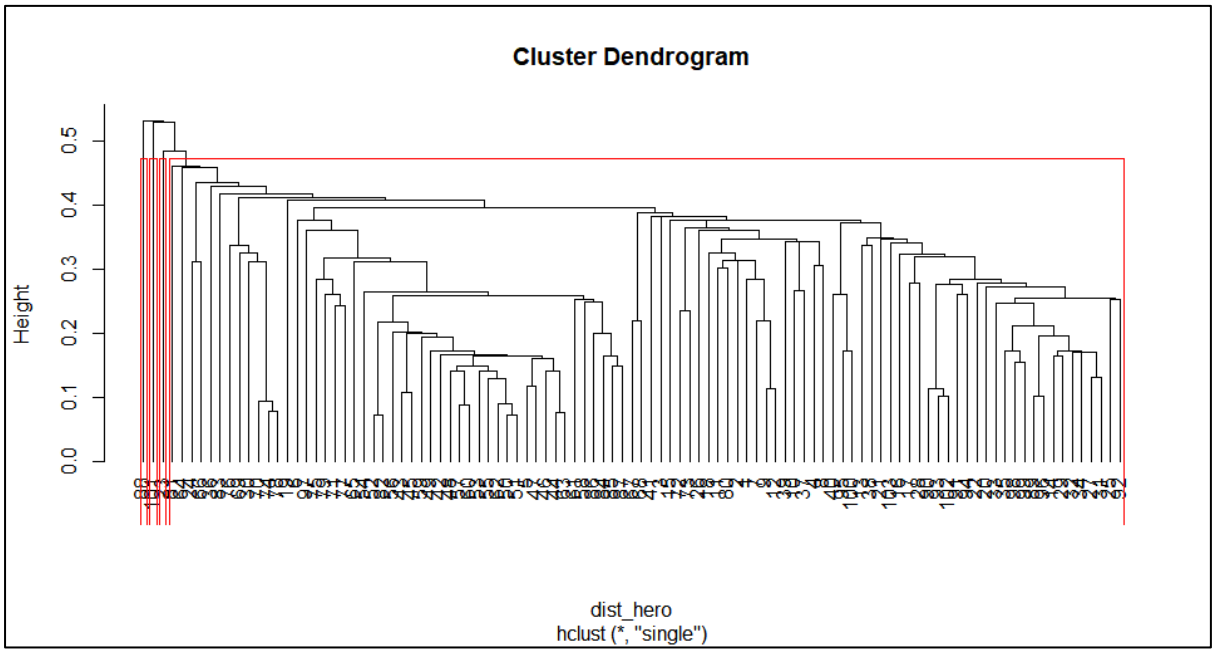
	cluster1vs2	cluster1vs3	cluster1vs4	cluster2vs3	cluster2vs4	cluster3vs4
1	1.88E-15	None	0.02419892	6.56E-11	0.000687805	0.076306671
2	0.054003661	1.32E-14	0.006663467	None	0.019169301	0.860243068
3	1.43E-08	2.33E-15	0.00157481	0.068018103	0.129904871	0.418237116
4	0.000797631	0.547187965	None	None	0.070627628	1
5	1.60E-11	0.000304472	0.035721943	0.026983207	0.220895112	None
6	None	None	None	None	None	None
7	0.002998699	2.59E-09	0.300321532	3.06E-05	0.005455023	1.07E-08
8	1.89E-05	0.31643935	0.720394334	8.45E-05	2.58E-05	None
9	5.12E-06	2.49E-07	0.012326203	0.764889348	0.297422879	0.354665987
10	1.31E-09	0.049672849	0.754344163	0.002976837	0.001710152	0.115697285
11	1.67E-12	0.045607568	0.00436334	0.003470515	6.02E-07	0.636492446
12	None	3.72E-08	0.024714491	0.009361328	0.12605459	0.716326356
13	0.001586078	1.33E-08	0.167979815	0.000430556	0.577106844	None
14	0.536257513	0.000351251	0.135664283	0.010565672	0.237978631	0.800095403
15	0.521720061	0.034024078	0.930245291	0.251408836	0.646328967	0.112817774
16	0.546820144	0.586809847	0.000664231	0.835541116	0.000610826	0.000679369

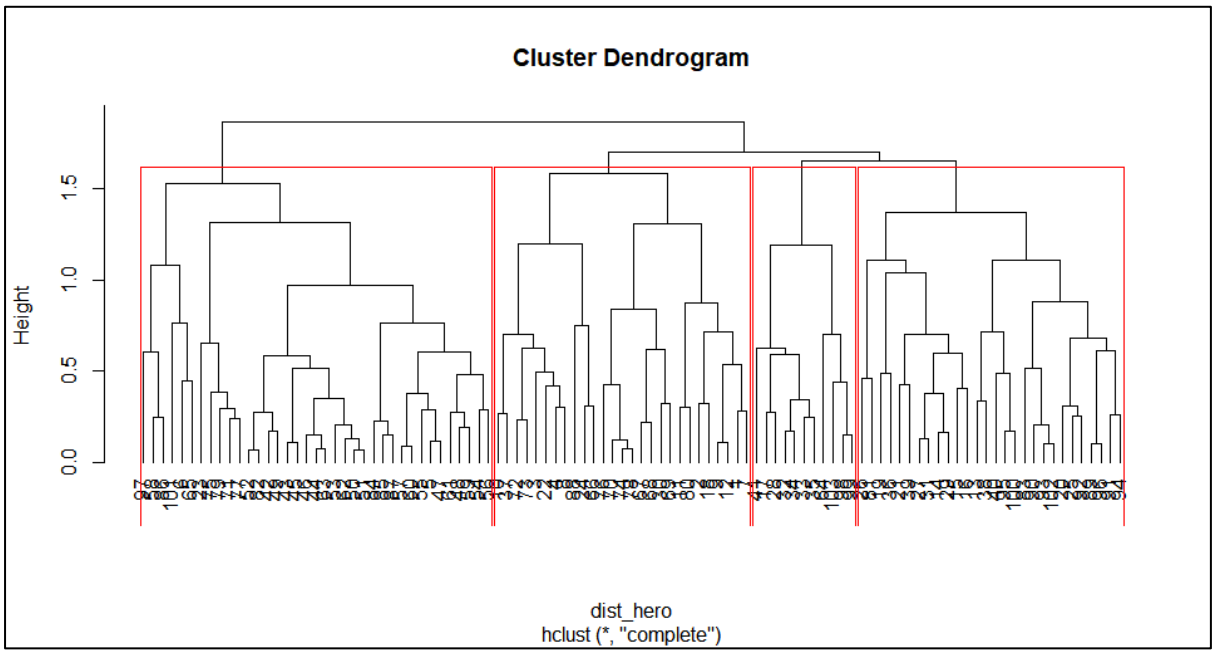
우선 왼쪽의 숫자는 1~16까지의 변수를 의미하고 칼럼에 노란 부분은 제가 비슷할 것이라 예상했던 두가 변수들에 관한 파트이고 파란색은 다를 것이라고 생각했던 군집 간의 비교를 보여주는 것입니다. 또한 초록색으로 쳐 놓은 부분은 p-value가 0.05보다 작은 값들을 쳐 놓은 것입니다(군집 A,B, 군집 X,Y에 관해서만). 사실 압도적으로 제가 비슷한 군집 들이라고 생각한 조합의 변수들 중에서 유의수준이 0.05보다 작은 값이 더 많을 거라고 생각하였는데 그렇지 못하다는 것을 확인할 수 있었습니다. 하지만 아까 위에서 말했다 시피 변수들 사이에서 아예 같은 값들을 가지는 것끼리 비교를 할 때는 (에러를 일으키기 때문에) 직접 None을 입력하였는데 이 또한 아예 0이라고 한다면 비슷한 군집들에 대한 변수 유의수준 중 0.05보다 더 작은 값이 많아지게 되는 것 같습니다.

[계층적 군집화]

[Question6]

계층적 군지화를 하기 위해서는 군집화가 되기 위해 모든 개체들 간의 거리와 유사도가 계산되어야 합니다. 거리를 계산 하기 위해서 dist 함수를 이용하여 각 해들의 거리를 판단하고 거리를 계산하기 전에 각 개체 간의 correlation을 구하는데 이는 클수록 거리가 가깝다는 것을 의미합니다. 문제에서 제시된 대로 single과 complete 방법으로 덴드로그램을 그려보았습니다. 또한 군집의 크기를 예측해보기 위해서 k=4라고 설정한 후에 그 크기를 비교해보기 위한 작업을 거쳤습니다. K의 크기를 굳이 4로 한 이유는 이전에 수행한 k-means에서 최적의 군집 개수를 4로 하였었기 때문입니다. 결과는 다음과 같이 나왔습니다.





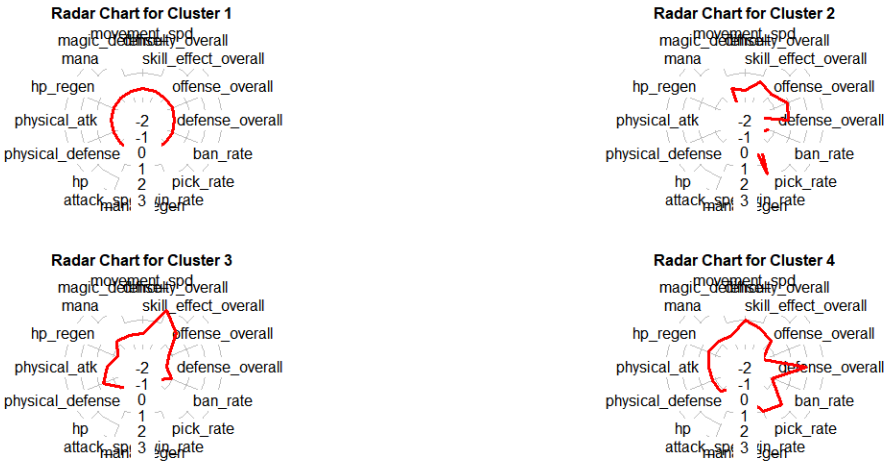
위의 결과에서도 바로 알 수 있다시피 single로 하였을 때 군집의 크기가 매우 비 대칭이라는 것을 알 수 있습니다. 이는 빨간색 박스의 크기를 보고 알 수 가 있는데, single 방법과는 상반 되도록 complete 방법을 보았을 때는 그나마 균등한 크기의 cluster 크기가 생성되었다는 것을 확인할 수 있었습니다.

[Question7]

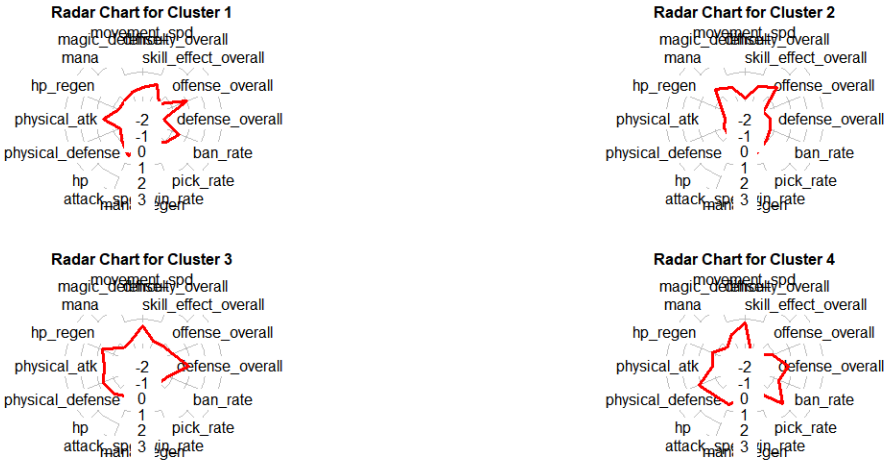
이전에 두번째 문제에서 제가 선정하였던 최적 개수의 군집은 4였습니다. 물론 2라는 결과가 나왔었지만 저는 나름의 이유로 4로 진행하였습니다. 그렇기에 7번 문제를 수행하기 우선 비슷한 방법으로 최적의 결과가 몇이 나오는지 확인하기 위해서 코드를 돌려본 결과, 2번 문제와 같이 2라는 결론이 나왔습니다. 하지만 저는 2를 사용하지 않을거

기 때문에 다음으로 높다고 생각되는 최적의 개수를 찾아본 결과 dunn index는 0.4가 넘어가고 실루엣 스코어가 두번째로 큰 값을 찾아보니 다시 한번 4가 나오게 되었습니다. 결론적으로 최적개수 군집을 4로하여 이 변수 값들을 이용하여 Radar chart를 만들어보았습니다. 각각 single과 complete에 대해서 만들었는데 그 결과는 다음과 같았습니다.

<single>



<complete>



일전에 만들었던 radar차트와 비교하여 보면 complete가 single보다 kmeans radar차트와 더 유사한 것처럼 보였습니다. 그 이유는 우선 형태적인 면을 보았을 때 single의 경우는 원이 형성이 되거나 어느 한쪽이 완전히 찌그러지거나 하는 모양이 많이 있었습니다. 물론 kmeans에도 그런 형태를 가진 차트가 있긴 하지만 그건 중간이 끊어진 듯한 모양을 가진 차트 정도가 있습니다. 하지만 complete를 살펴보면 cluster2는 keans의 cluster4와 거의 비슷한 모양을 가지며 kmeans의 cluster1,3과 비슷한 모양을 가진 cluster들이 complete 속에 있다는 것을 확실하게 확인해볼 수 있었습니다.

다시한번 말하자면 결론적으로 complete linkage가 K-means clustering과 제일 유사한 형태를 가지고 있다는 것을 확인해 볼 수 있었습니다.

[Question8]

숫자로 비교를 하기위해 제가 선택한 방법은 군집을 각각 k-means clustering과 계층적 군집화를 single과 complete로 구축하였을 때 실제 class와 혼동행렬처럼 만들어 이 수치들이 얼마나 비슷하게 설정되어 있는지를 확인해 보는 것이었습니다. 일전에 저는 role이라는 변수 안에 있는 값들을 숫자로 바꿔주었는데 이에 따라 총 여섯가지의 캐릭터 직업에 어떻게 구분하여 군집을 할당해주었는지를 비교해 볼 수 있었습니다. 우선 세가지를 비교하기 위한 테이블은 아래와 같이 나왔습니다.

	kmc_cluster			
real_class	1	2	3	4
1	0	2	9	1
2	11	1	15	2
3	0	23	0	0
4	0	6	10	0
5	2	6	0	0
6	14	0	0	1

	hero_hlc_S			
real_class	1	2	3	4
1	12	0	0	0
2	28	1	0	0
3	23	0	0	0
4	16	0	0	0
5	7	0	1	0
6	14	0	0	1

	hero_hlc_C			
real_class	1	2	3	4
1	10	2	0	0
2	5	1	16	7
3	0	22	0	1
4	11	5	0	0
5	1	5	2	0
6	0	2	10	3

우선 제가 이전에 radar 차트를 이용하여 계층적 방법의 어느 방법이 k-means와 더 가까울지 예상을 해볼 때 complete라고 했었는데, 현재 위의 테이블을 통해 그게 맞다는 것을 확인할 수 있었습니다. 그 이유는 우선 single같은 경우는 하나의 군집에 거의 모든 값들이 몰려 있는 것에 비해 KMC와 HLC_C는 골고루 퍼져 있고 모두 같은 곳에 분포가 KMC의 3번과 HLC_C의 1번, KMC 2번과 HLC_C 2번이 매우 유사한 형태를 띄고 있다는 것을 알 수 있습니다. 이로써 Complete linkage가 더 유사하다는 것을 다시 한번 더 확인할 수 있었습니다.

[DBSCAN]

[Question9]

DBSCAN은 밀도 군집 분석으로 kmeans clustering의 단점을 보완해줄 수 잇는데 K-means는 모든 객체들을 cluster에 할당해야 하여 noise와 outlier에 민감한데 이런 단점을 보완해줄 수 있습니다. 밀도를 기반으로 clustering 하는 방법이기 때문에 밀도의 기준이 되는 반경인 eps와 반경내에 들어가야 하는 최소 점들의 수를 모수로 정의할 수 있습니다. 사실 처음에 eps를 소수점 단위로 하여 돌려 보았었는데 너무 반경이 너무 작은 탓인지 이상한 결과(거의 0이 나옴)가 나와서 좀더 키워 코드를 돌려보았습니다. 이때의 후보군은 다음과 같이 설정하였습니다.

eps	0.5, 1, 2, 3, 4
minPts	10, 20, 30, 40, 50

이렇게 하여 코드를 돌려본 결과, 50%의 경우에서 군집이 형성되지 않았다는 것을 확인할 수 있었습니다. 그럼에도 총 4번의 경우에서 1개의 군집이 형성되었고, 1가지의 경우에서 2개의 cluster가 생겼습니다. 비록 제가 2번 문제에서 4를 선정하여 문제들에 대해 풀었으나, 사실상 코드 결과로만 본다면 2가 optimal 개수이기 때문에 DBSCAN으로 찾는 최적의 개수를 찾았다고 볼 수 있을 것 같습니다.

[Question10]

우선 9번 문제에서 나온 결과물을 가져오자면 다음과 같다는 것을 확인할 수 있습니다.

```
Available fields: cluster, eps, minPts
DBSCAN clustering for 103 objects.
Parameters: eps = 3, minPts = 10
The clustering contains 2 cluster(s) and 54 noise points.

 0  1  2
54 38 11
```

총 두개의 군집이 생성되었는데 하나는 38개가 2번째 군집에는 11개가 들어가 있는 것을 볼 수 있고, 0이라는 부분을 보면 총 54개의 노이즈 포인트가 들어가 있다는 것을 확인할 수 있었습니다.

[Question11]

결과적으로 말하자면 제가 생각하기에 이 데이터셋에 가장 적합한 군집화 알고리즘은 계층적 군집화에서 complete linkage를 사용할 때라고 생각합니다.그렇게 생각하는 이유는 몇가지가 있는데, 우선 DBSCAN을 보자면 거의 절반 이상을 노이즈로서 취급해버립니다. 하지만 이 데이터셋을 게임 캐릭터들에 관한 데이터 로서 많은 캐릭터들을 노이즈로 취급한다는 것은 그만큼 유사성을 잘 파악하지 못하고 군집을 제대로 하지 못하는 것이라고 생각합니다. 다음으로는 K-means clustering입니다. K-means clustering에서 저는 가장 적합한 개수로 4를 선정하였었는데 사실 이때 나온 dunn index의 값과 실루엣 값이 계층적 군집화의 dunnindex와 실루엣 값과 같지만 radar chart를 보면 알 수 있듯이 k-means에서의 2번째 클러스터에서 끊어져 있는 듯한 모양을 보이고 있습니다. 반면에 계층적 군집화의 complete는 radar 차트가 매우 이쁘게 잘 그려져 있고 추가적으로 k를 4로 하였을 때 덴드로그램을 보면 알 수 있듯이 균등하게 나뉘어 지는 데 게임 특성상 밸런스가 중요하기 때문에 캐릭터들의 능력치와 그 외 요소들이 비슷비슷한데 아주 약간씩만 차별성을 주어야 하므로 균등하게 나뉘는 것이 맞다고 생각했는데 그 모습을 정말 잘 보여주고 있다고 생각하였습니다. 결론적으로 계층적 군집화를 할 때 complete linkage를 이용하는 것이 가장 이 데이터셋과 잘 맞는 방법이라고 생각합니다.