Thomas Dinh

Oct. 17, 2023

DSE 6211

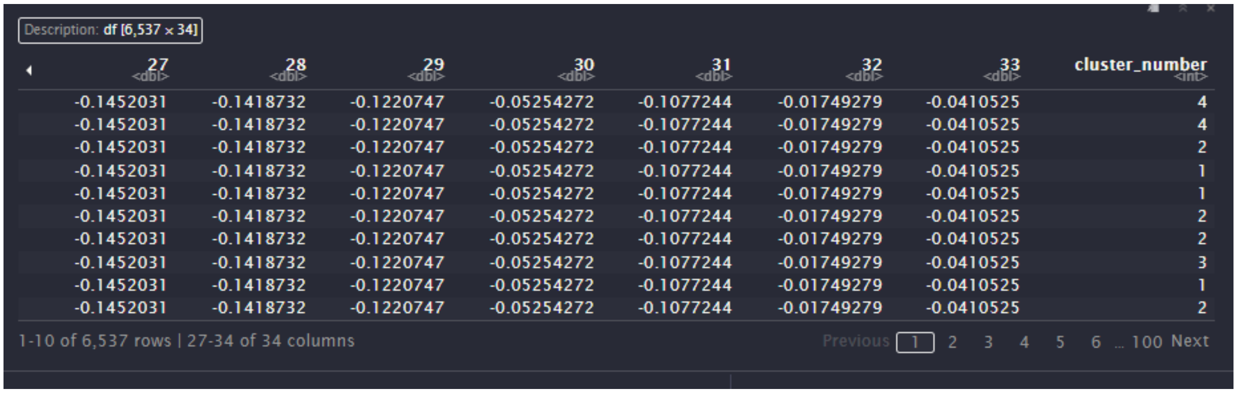
Week 8

Exercises

1) Why is it good practice to center and scale before applying k-means clustering?

As a distance-based clustering algorithm, K-means clustering puts data points in groups according to how far apart they are from one another. Features with bigger sizes will impact the clustering results more if the data is not centered and scaled. This may result in clusters where certain qualities predominate, and other features are disregarded.

2) Print the cluster sizes and centers to the R console. Include a screenshot of the output.

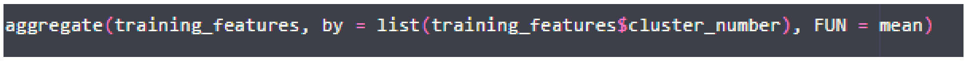


Description: df [6,537 x 34]

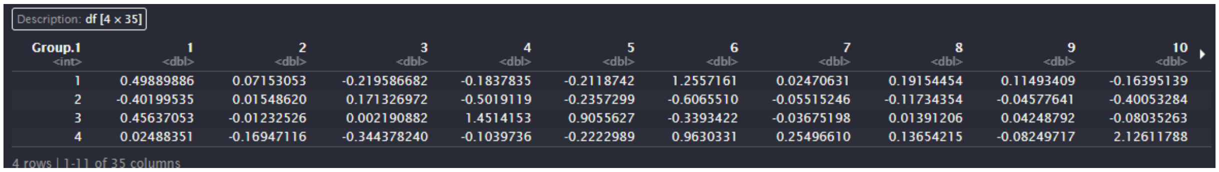
27 <dbl>	28 <dbl>	29 <dbl>	30 <dbl>	31 <dbl>	32 <dbl>	33 <dbl>	cluster_number <int>
-0.1452031	-0.1418732	-0.1220747	-0.05254272	-0.1077244	-0.01749279	-0.0410525	4
-0.1452031	-0.1418732	-0.1220747	-0.05254272	-0.1077244	-0.01749279	-0.0410525	4
-0.1452031	-0.1418732	-0.1220747	-0.05254272	-0.1077244	-0.01749279	-0.0410525	2
-0.1452031	-0.1418732	-0.1220747	-0.05254272	-0.1077244	-0.01749279	-0.0410525	1
-0.1452031	-0.1418732	-0.1220747	-0.05254272	-0.1077244	-0.01749279	-0.0410525	1
-0.1452031	-0.1418732	-0.1220747	-0.05254272	-0.1077244	-0.01749279	-0.0410525	2
-0.1452031	-0.1418732	-0.1220747	-0.05254272	-0.1077244	-0.01749279	-0.0410525	2
-0.1452031	-0.1418732	-0.1220747	-0.05254272	-0.1077244	-0.01749279	-0.0410525	3
-0.1452031	-0.1418732	-0.1220747	-0.05254272	-0.1077244	-0.01749279	-0.0410525	1
-0.1452031	-0.1418732	-0.1220747	-0.05254272	-0.1077244	-0.01749279	-0.0410525	2

1-10 of 6,537 rows | 27-34 of 34 columns

3) Use the aggregate() function to calculate the mean of each variable within each cluster. Include a screenshot of the output.



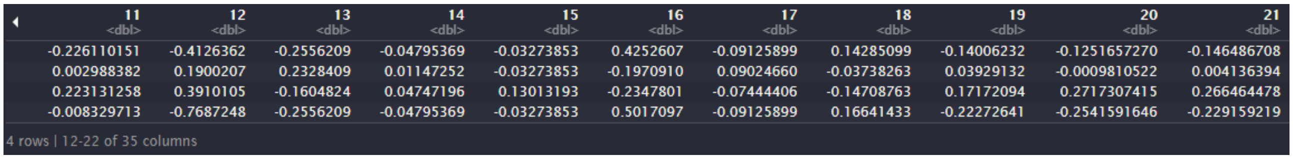
```
aggregate(training_features, by = list(training_features$cluster_number), FUN = mean)
```



Description: df [4 x 35]

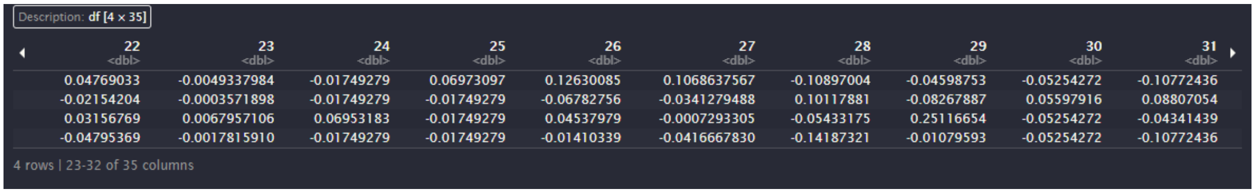
Group.1 <int>	1 <dbl>	2 <dbl>	3 <dbl>	4 <dbl>	5 <dbl>	6 <dbl>	7 <dbl>	8 <dbl>	9 <dbl>	10 <dbl>
1	0.49889886	0.07153053	-0.219586682	-0.1837835	-0.2118742	1.2557161	0.02470631	0.19154454	0.11493409	-0.16395139
2	-0.40199535	0.01548620	0.171326972	-0.5019119	-0.2357299	-0.6065510	-0.05515246	-0.11734354	-0.04577641	-0.40053284
3	0.45637053	-0.01232526	0.002190882	1.4514153	0.9055627	-0.3393422	-0.03675198	0.01391206	0.04248792	-0.08035263
4	0.02488351	-0.16947116	-0.344378240	-0.1039736	-0.2222989	0.9630331	0.25496610	0.13654215	-0.08249717	2.12611788

4 rows | 1-11 of 35 columns



11	12	13	14	15	16	17	18	19	20	21
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
-0.226110151	-0.4126362	-0.2556209	-0.04795369	-0.03273853	0.4252607	-0.09125899	0.14285099	-0.14006232	-0.1251657270	-0.146486708
0.002988382	0.1900207	0.2328409	0.01147252	-0.03273853	-0.1970910	0.09024660	-0.03738263	0.03929132	-0.0009810522	0.004136394
0.223131258	0.3910105	-0.1604824	0.04747196	0.13013193	-0.2347801	-0.07444406	-0.14708763	0.17172094	0.2717307415	0.266464478
-0.008329713	-0.7687248	-0.2556209	-0.04795369	-0.03273853	0.5017097	-0.09125899	0.16641433	-0.22272641	-0.2541591646	-0.229159219

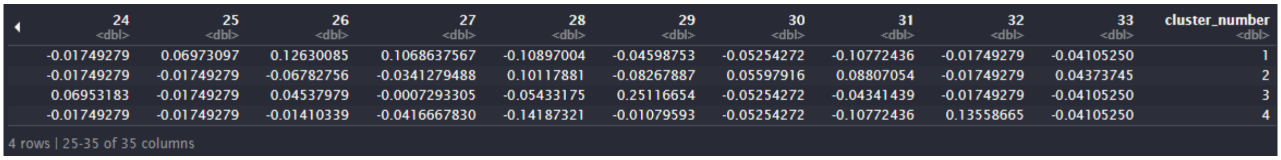
4 rows | 12-22 of 35 columns



Description: df [4 × 35]

22	23	24	25	26	27	28	29	30	31
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
0.04769033	-0.0049337984	-0.01749279	0.06973097	0.12630085	0.1068637567	-0.10897004	-0.04598753	-0.05254272	-0.10772436
-0.02154204	-0.0003571898	-0.01749279	-0.01749279	-0.06782756	-0.0341279488	0.10117881	-0.08267887	0.05597916	0.08807054
0.03156769	0.0067957106	0.06953183	-0.01749279	0.04537979	-0.0007293305	-0.05433175	0.25116654	-0.05254272	-0.04341439
-0.04795369	-0.0017815910	-0.01749279	-0.01749279	-0.01410339	-0.0416667830	-0.14187321	-0.01079593	-0.05254272	-0.10772436

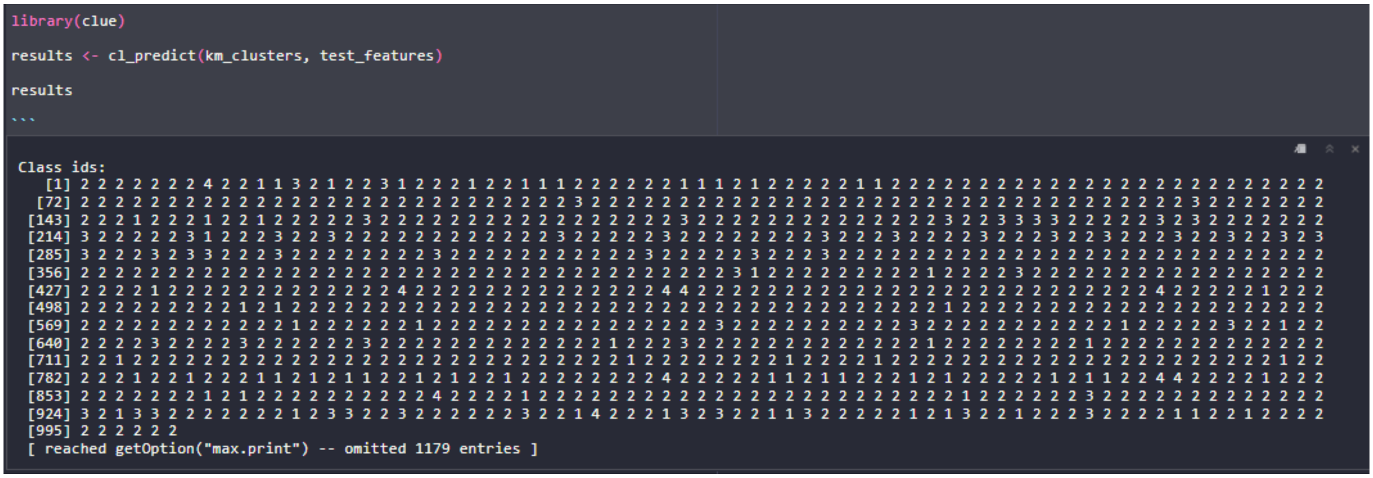
4 rows | 23-32 of 35 columns



24	25	26	27	28	29	30	31	32	33	cluster_number
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
-0.01749279	0.06973097	0.12630085	0.1068637567	-0.10897004	-0.04598753	-0.05254272	-0.10772436	-0.01749279	-0.04105250	1
-0.01749279	-0.01749279	-0.06782756	-0.0341279488	0.10117881	-0.08267887	0.05597916	0.08807054	-0.01749279	0.04373745	2
0.06953183	-0.01749279	0.04537979	-0.0007293305	-0.05433175	0.25116654	-0.05254272	-0.04341439	-0.01749279	-0.04105250	3
-0.01749279	-0.01749279	-0.01410339	-0.0416667830	-0.14187321	-0.01079593	-0.05254272	-0.10772436	0.13558665	-0.04105250	4

4 rows | 25-35 of 35 columns

4) Assign the observations in test_features to the clusters found using the sample of the observations in training_features above. Include a screenshot of the output. Hint: a function in the 'clue' R library can be used for this.



~~~
library(clue)

results <- cl_predict(km_clusters, test_features)

results
```
~~~

```
Class ids:
   [1] 2 2 2 2 2 2 2 4 2 2 1 1 3 2 1 2 2 3 1 2 2 2 1 2 2 1 1 1 2 2 2 2 2 2 1 1 1 2 1 2 2 2 2 2 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
  [72] 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 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 3 2 2 2 2 2 2 2
 [143] 2 2 2 1 2 2 2 1 2 2 1 2 2 2 2 2 3 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 2 2 2 2 3 2 2 3 3 3 3 2 2 2 2 2 3 2 3 2 2 2 2 2 2 2
 [214] 3 2 2 2 2 2 3 1 2 2 2 3 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 3 2 2 2 2 2 2 2 2 3 2 2 2 3 2 2 2 2 3 2 2 2 3 2 2 3 2 2 2 3 2 2 3 2 2 3 2 3
 [285] 3 2 2 2 3 2 3 3 2 2 2 3 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 3 2 2 2 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
 [356] 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 3 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [427] 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 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 4 2 2 2 2 2 1 2 2 2
 [498] 2 2 2 2 2 2 2 2 2 1 2 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 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [569] 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 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 3 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 3 2 2 1 2 2
 [640] 2 2 2 2 3 2 2 2 2 3 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2
 [711] 2 2 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 1 2 2 2 2 2 2 2 2 1 2 2 2 2 1 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
 [782] 2 2 2 1 2 2 1 2 2 2 1 1 2 1 2 1 1 2 2 1 2 1 2 2 1 2 2 2 2 2 2 2 2 4 2 2 2 2 2 1 1 2 1 1 2 2 2 1 2 1 2 2 2 2 2 1 2 1 1 2 2 4 4 2 2 2 2 1 2 2 2
 [853] 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 1 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 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [924] 3 2 1 3 3 2 2 2 2 2 2 2 1 2 3 3 2 2 3 2 2 2 2 2 2 3 2 2 1 4 2 2 2 1 3 2 3 2 2 1 1 3 2 2 2 2 2 1 2 1 3 2 2 1 2 2 2 3 2 2 2 2 1 1 2 2 1 2 2 2 2
 [995] 2 2 2 2 2 2
 [ reached getOption("max.print") -- omitted 1179 entries ]
```