TDT4300 - Exercise 3

Jens Waage

March 2020

1 k-Means Clustering

See Jupyter Notebook

2 HAC

(a) HAC generates a nested hierarchical structure of clusters. Its main idea is to keep track of the distances between every cluster, and iteratively joining the two closest clusters.

At the beginning every point is considered a cluster. The distance between every point is calculated and stored in a matrix. Then, the two closest points are joined together into a single cluster, and the matrix is recalculated for this new cluster configuration. This loop continues until only a single cluster remains.

There are different ways to measure the distance between clusters. In a MIN-link, the distance between clusters is considered to be the distance between the two closest points from each cluster. In a MAX-link, the distance is considered to be the distance between the two farthest points from each cluster.

(b) We start by using MIN-link. We follow the steps of the algorithm:

(i) We go through every point and calculate the euclidean distance. We get the following distance matrix:

	A	**B**	**C**	**D**	**E**	**F**
A	0	5.10	4.12	5.00	7.62	10.30
B	5.10	0	1.00	6.40	6.32	8.00
C	4.12	1.00	0	5.66	6.08	8.06
D	5.00	6.40	5.66	0	3.61	6.40
E	7.62	6.32	6.08	3.61	0	2.83
F	10.30	8.00	8.06	6.40	2.83	0

We join the clusters with the smallest distance, which is cluster 2 and 3.

(ii) We calculate the next distance matrix:

	A	**B, C**	**D**	**E**	**F**
A	0	4.12	5.00	7.62	10.30
B, C	4.12	0	5.66	6.08	8.00
D	5.00	5.66	0	3.61	6.40
E	7.62	6.08	3.61	0	2.83
F	10.30	8.00	6.40	2.83	0

We join the clusters with the smallest distance, which is 4 and 5

(iii) We calculate the next distance matrix:

	A	**B, C**	**D**	**E, F**
A	0	4.12	5.00	7.62
B, C	4.12	0	5.66	6.08
D	5.00	5.66	0	3.61
E, F	7.62	6.08	3.61	0

We join the clusters with the smallest distance, which is 3 and 4.

(iv) We calculate the next distance matrix:

	A	**B, C**	**D, E, F**
A	0	4.12	5.00
B, C	4.12	0	5.66
D, E, F	5.00	5.66	0

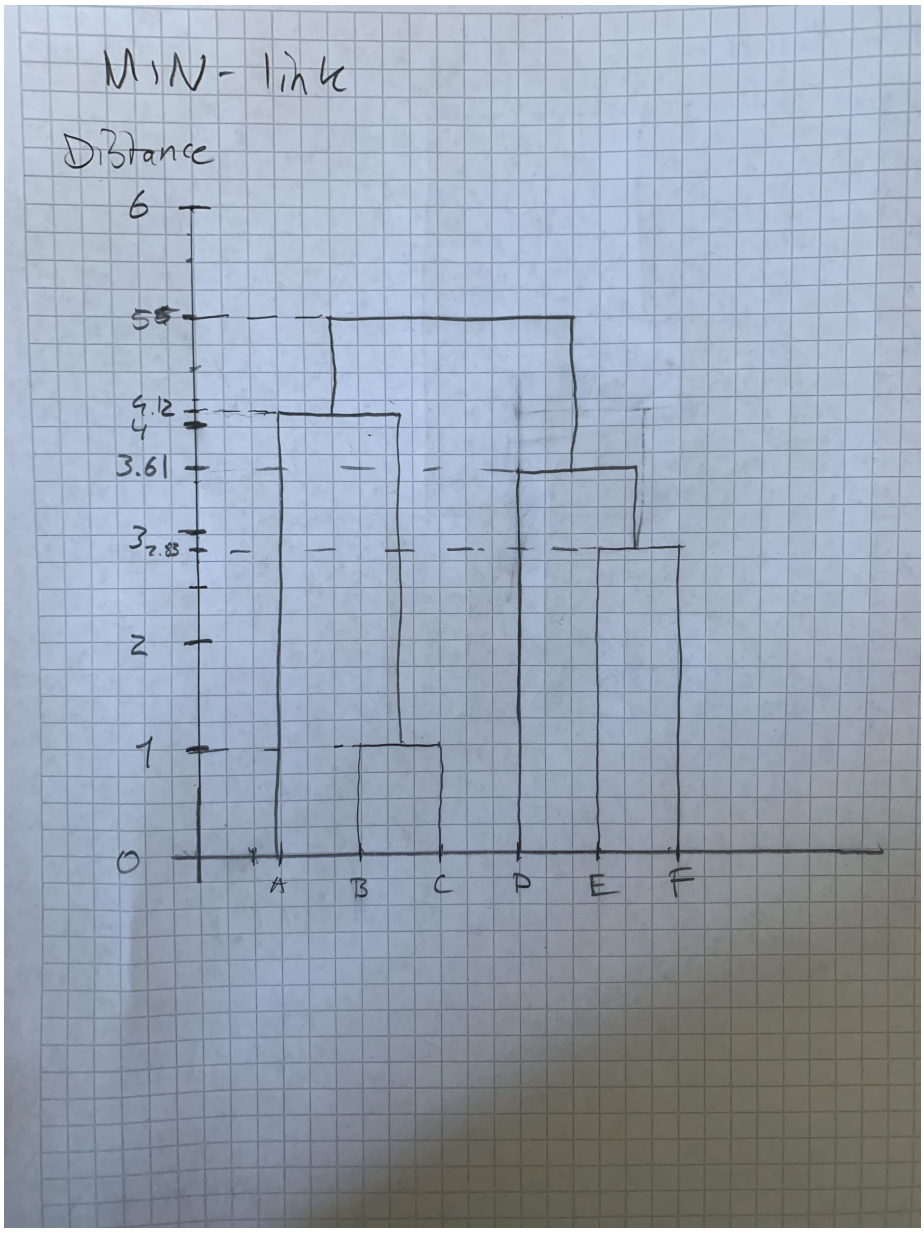
We join the clusters with the smallest distance, which is 1 and 2.

(v) We calculate the next distance matrix:

	A, B, C	**D, E, F**
A, B, C	0	5.00
D, E, F	5.00	0

Finally, we join the two remaining clusters into a single cluster, and we're done.

We get the following dendrogram:



We do the same process for MAX-link:

(i) We go through every point and calculate the euclidean distance. We get the following distance matrix:

	A	B	C	D	E	F
A	0	5.10	4.12	5.00	7.62	10.30
B	5.10	0	1.00	6.40	6.32	8.00
C	4.12	1.00	0	5.66	6.08	8.06
D	5.00	6.40	5.66	0	3.61	6.40
E	7.62	6.32	6.08	3.61	0	2.83
F	10.30	8.00	8.06	6.40	2.83	0

We join the clusters with the smallest distance, which is cluster 2 and 3.

(ii) We calculate the next distance matrix:

	A	B, C	D	E	F
A	0	5.10	5.00	7.62	10.30
B, C	5.10	0	6.40	6.32	8.06
D	5.00	6.40	0	3.61	6.40
E	7.62	6.32	3.61	0	2.83
F	10.30	8.06	6.40	2.83	0

We join the clusters with the smallest distance, which is 4 and 5

(iii) We calculate the next distance matrix:

	A	B, C	D	E, F
A	0	5.10	5.00	10.30
B, C	5.10	0	6.40	8.06
D	5.00	6.40	0	6.40
E, F	10.30	8.06	6.40	0

We join the clusters with the smallest distance, which is 1 and 3.

(iv) We calculate the next distance matrix:

	A, D	B, C	E, F
A, D	0	6.40	10.30
B, C	6.40	0	8.06
E, F	10.30	8.06	0

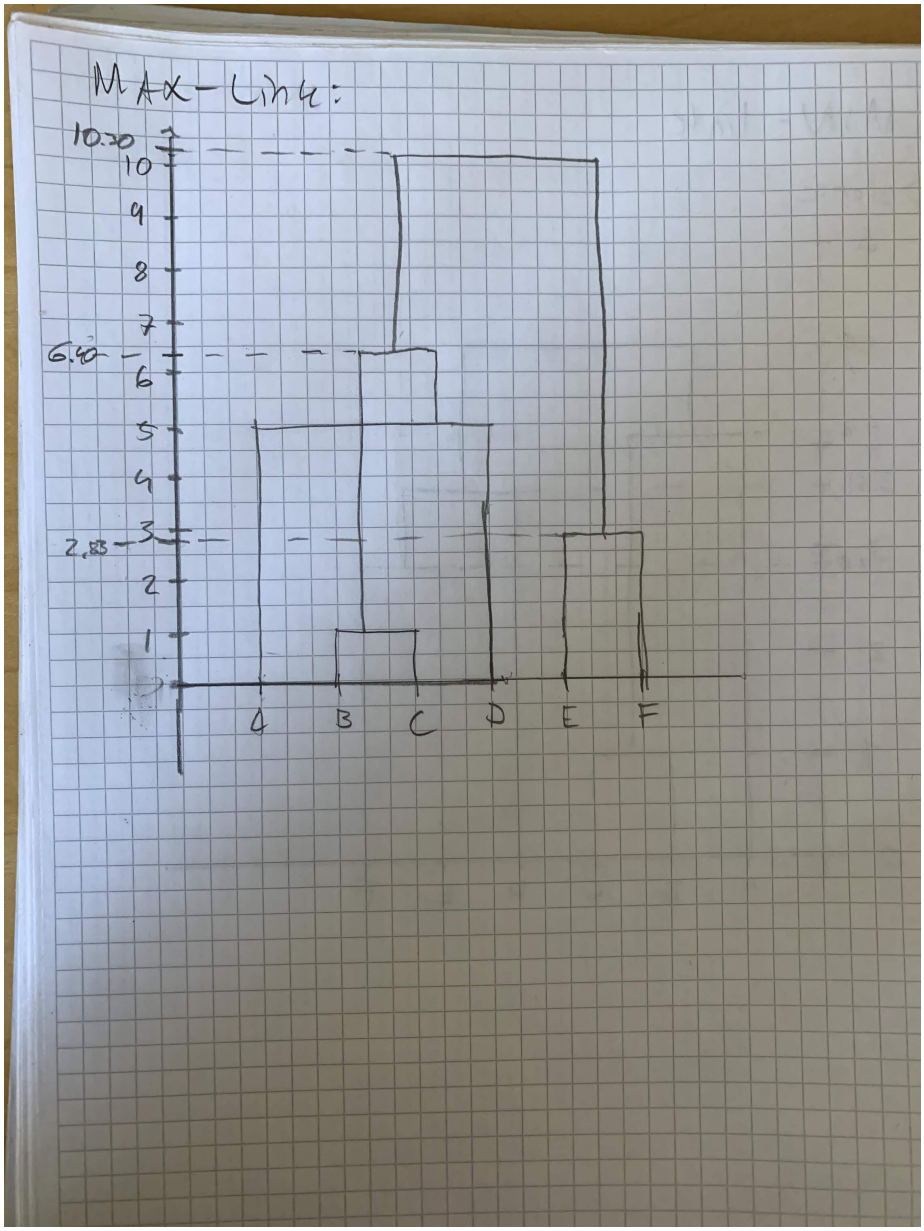
We join the clusters with the smallest distance, which is 1 and 2.

(v) We calculate the next distance matrix:

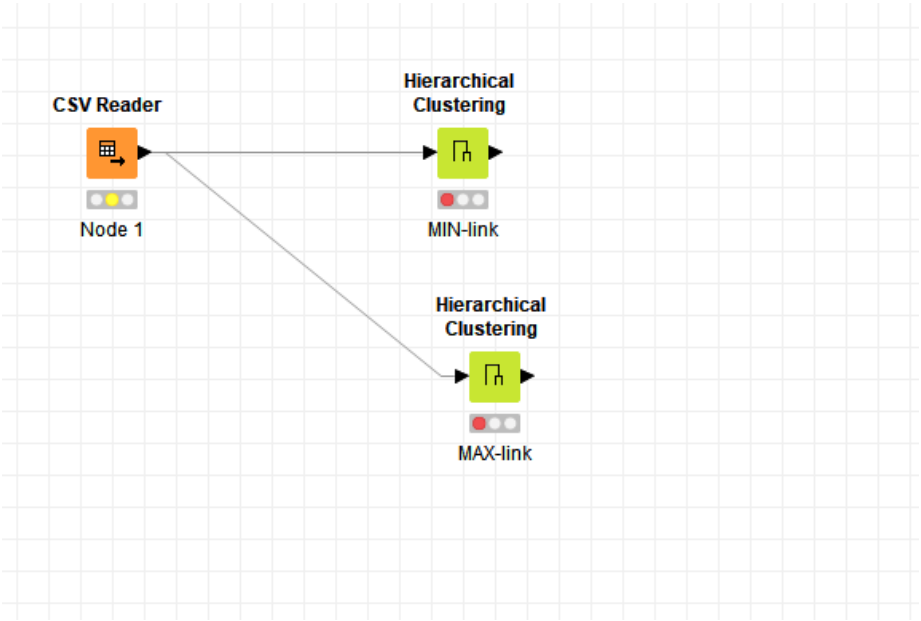
	A, D, B, C	E, F
A, D, B, C	0	10.30
D, E, F	10.30	0

Finally, we join the two remaining clusters into a single cluster, and we're done.

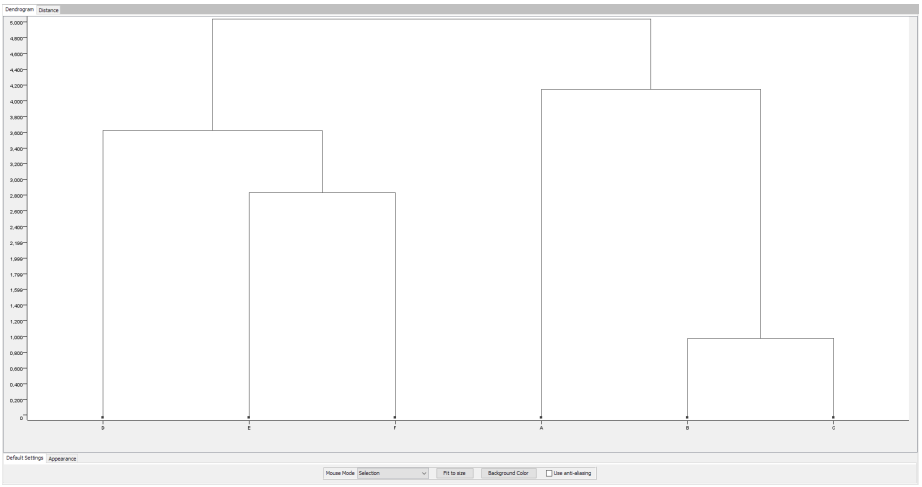
We get the following dendrogram:



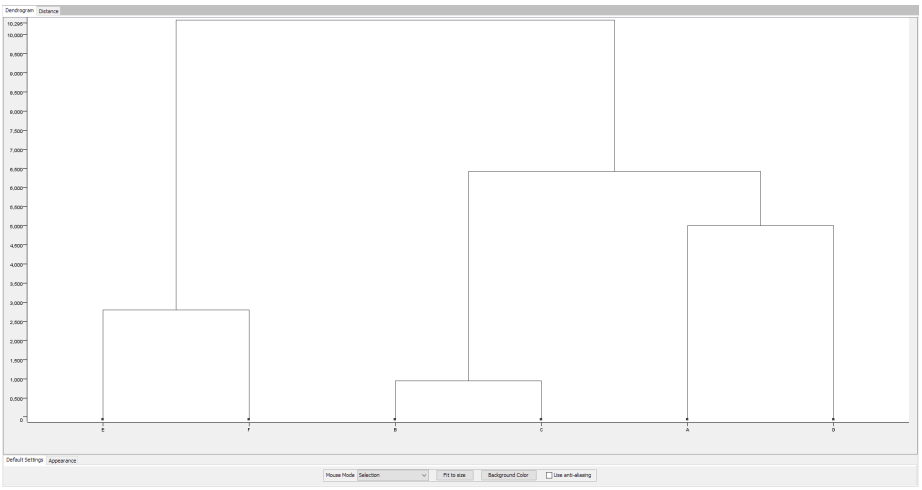
(c) The following workflow was created:



Using min-link gave the following dendrogram:



Using max-link gave the following dendrogram:



These are the same as the ones achieved in the previous tasks.

3 DBSCAN

(a) We start by calculating all the euclidean distances between the points, giving the following matrix:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0,00	14,76	12,08	2,00	10,77	13,00	11,40	13,04	2,00	7,00	8,94	14,21	13,60	3,16	8,60	3,61	11,70	18,38
2	14,76	0,00	8,94	13,04	9,49	2,24	10,77	2,00	13,93	9,22	9,06	5,66	3,00	12,65	6,32	11,18	3,61	6,00
3	12,08	8,94	0,00	11,40	1,41	9,22	2,00	7,21	10,30	11,18	3,16	4,00	10,63	8,94	6,32	9,22	9,22	8,25
4	2,00	13,04	11,40	0,00	10,20	11,18	11,05	11,40	2,83	5,00	8,25	13,04	11,70	3,16	7,07	2,24	9,85	17,03
5	10,77	9,49	1,41	10,20	0,00	9,43	1,41	7,62	8,94	10,44	2,00	5,10	10,82	7,62	5,83	8,06	9,22	9,49
6	13,00	2,24	9,22	11,18	9,43	0,00	10,82	2,24	12,37	7,07	8,54	6,71	1,41	11,18	5,00	9,49	1,41	8,06
7	11,40	10,77	2,00	11,05	1,41	10,82	0,00	8,94	9,49	11,70	3,16	6,00	12,21	8,25	7,21	9,00	10,63	10,20
8	13,04	2,00	7,21	11,40	7,62	2,24	8,94	0,00	12,08	8,06	7,07	4,47	3,61	10,77	4,47	9,43	3,00	6,32
9	2,00	13,93	10,30	2,83	8,94	12,37	9,49	12,08	0,00	7,28	7,21	12,73	13,15	1,41	7,62	3,00	11,18	17,03
10	7,00	9,22	11,18	5,00	10,44	7,07	11,70	8,06	7,28	0,00	8,54	11,18	7,21	6,71	5,00	4,47	5,66	14,32
11	8,94	9,06	3,16	8,25	2,00	8,54	3,16	7,07	7,21	8,54	0,00	5,83	9,85	5,83	4,24	6,08	8,06	10,30
12	14,21	5,66	4,00	13,04	5,10	6,71	6,00	4,47	12,73	11,18	5,83	0,00	8,06	11,31	6,32	10,82	7,28	4,47
13	13,60	3,00	10,63	11,70	10,82	1,41	12,21	3,61	13,15	7,21	9,85	8,06	0,00	12,04	6,08	10,20	2,00	9,00
14	3,16	12,65	8,94	3,16	7,62	11,18	8,25	10,77	1,41	6,71	5,83	11,31	12,04	0,00	6,32	2,24	10,05	15,62
15	8,60	6,32	6,32	7,07	5,83	5,00	7,21	4,47	7,62	5,00	4,24	6,32	6,08	6,32	0,00	5,00	4,12	10,00
16	3,61	11,18	9,22	2,24	8,06	9,49	9,00	9,43	3,00	4,47	6,08	10,82	10,20	2,24	5,00	0,00	8,25	14,87
17	11,70	3,61	9,22	9,85	9,22	1,41	10,63	3,00	11,18	5,66	8,06	7,28	2,00	10,05	4,12	8,25	0,00	9,22
18	18,38	6,00	8,25	17,03	9,49	8,06	10,20	6,32	17,03	14,32	10,30	4,47	9,00	15,62	10,00	14,87	9,22	0,00

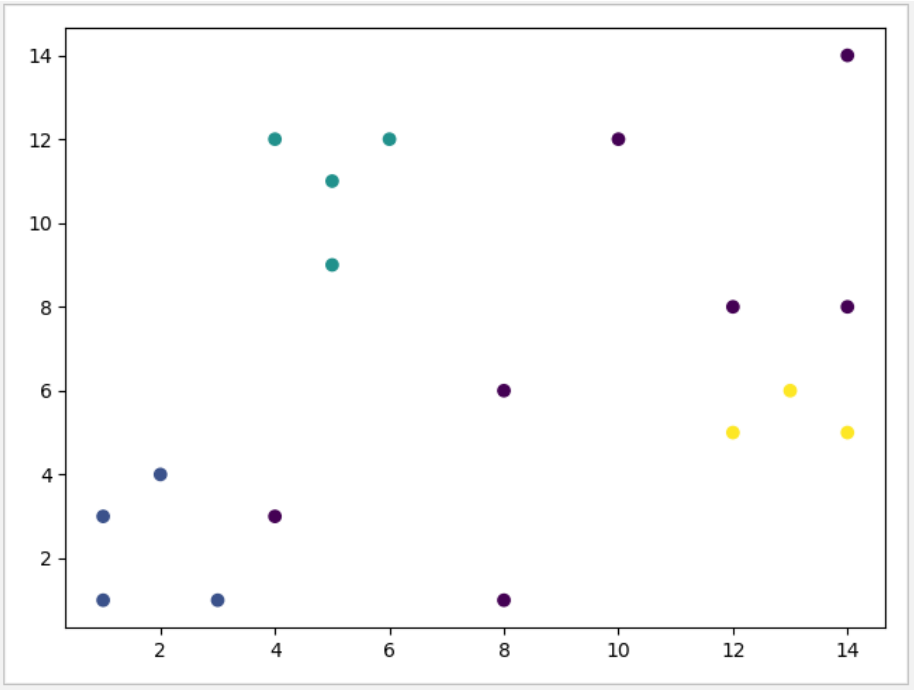
- We classify all the core points by counting how many have at least 3 points within a distance of 2.00. We get the following result: [1 3 5 6 7 9 13 17]
- Next, we find all the points that are within 2.00 of a core point. These are classified as border points. We get the following result: [4 11 14]

- Finally, the remained of the points are noise points. We get the following result: [2 8 10 12 15 16 18]

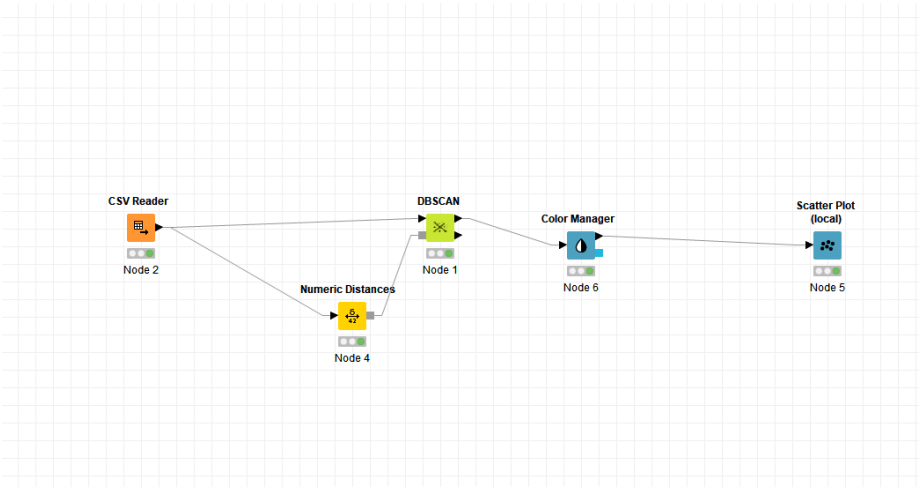
Now we use our core points given above, and put an edge between every node that is within 2.00 of each other. Then, all nodes that are in the same chain are put into the same cluster. We get the following clusters:

Point	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Cluster	1	None	2	1	2	3	2	None	1	None	2	None	3	1	None	None	4	None

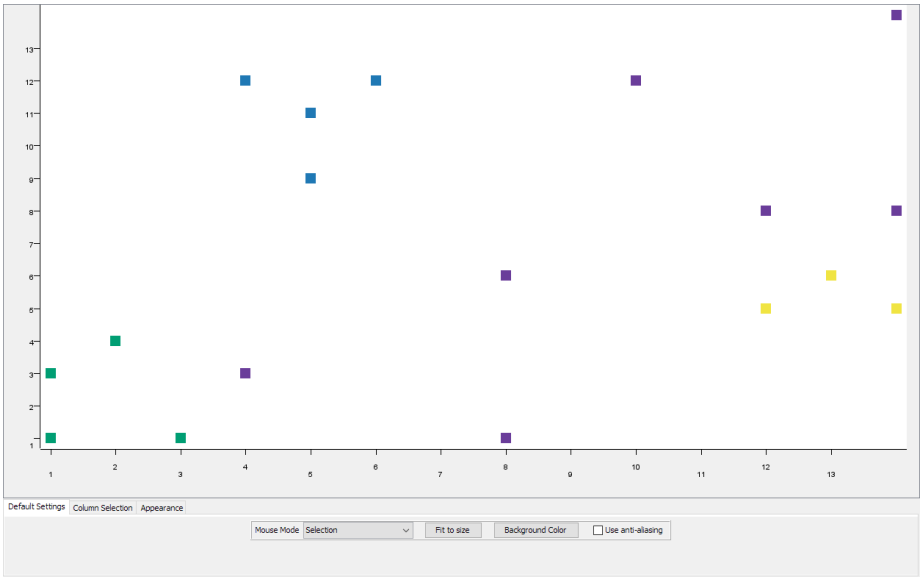
Illustrated with a scatter plot, where each cluster has it's own color:



(b) The following workflow was created:



This provided the following scatter plot with clusters:



We can see that the two clusters are equal.