Task 2

Task 2a

HAC is mainly used to keep track of the distance between clusters using a nested heirarchical structure. Then joining the closes clusters to create larger clusters defining cluster groups. HAC is done by calculating distance between every point and keep track of this in a matrix. Then the two closest points are merged and and the process is repeted until we have a single cluster. To calculate the distance we can use MIN-link, the smallest distance between two points from each cluster, or MAX-link the furthest distance between two points from each cluster.

Task 2b

MIN-link

We begin by calculating the inital distance

	A	**B**	**C**	**D**	**E**
A	0	4.12310563	4.12310563	**1**	7.21110255
B	4.12310563	0	7.07106781	3.16227766	7
C	4.12310563	7.07106781	0	4.47213595	5.38516481
D	**1**	3.16227766	4.47213595	0	6.70820393
E	7.21110255	7	5.38516481	6.70820393	0

Next we find that cluster A and D are closest and merge these clusters with smallest distance value. The next iteration is calculated...

	A, D	**B**	**C**	**E**
A, D	0	**3.16227766**	4.12310563	6.70820393
B	**3.16227766**	0	7.07106781	7
C	4.12310563	7.07106781	0	5.38516481
E	6.70820393	7	5.38516481	0

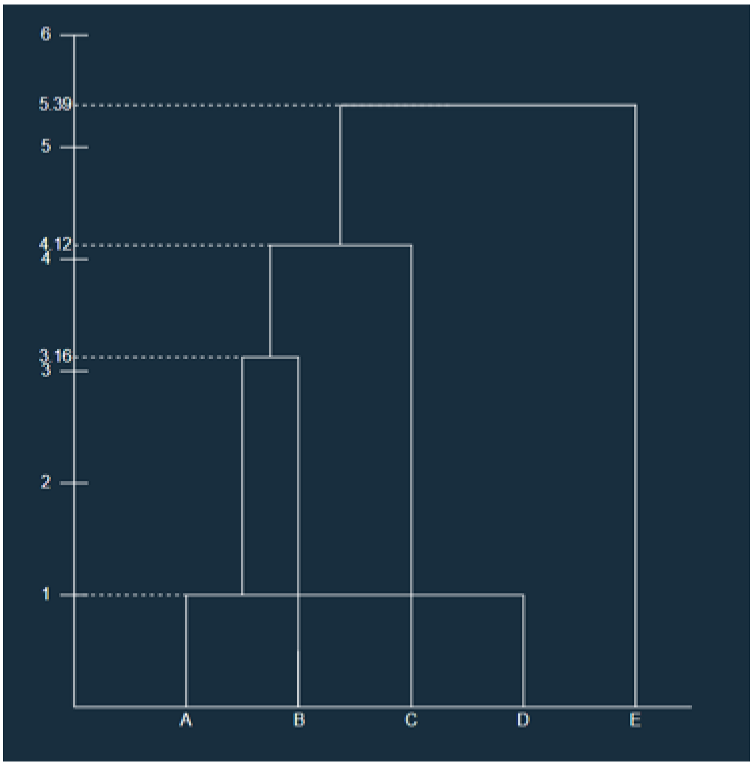
We can then merge cluster AD with B to get

	A, B, D	**C**	**E**
A, B, D	0	**4.12310563**	6.70820393
C	**4.12310563**	0	5.38516481
E	6.70820393	5.38516481	0

Next we merge cluster ABD with C to get the final assosiation matrix

	A, B, C, D	E
A, B, C, D	0	**5.38516481**
E	**5.38516481**	0

We can from this create a dendogram showing the merge steps:



MAX-link

We start by using the same generated table as in MIN

	A	B	C	D	E
A	0	4.12310563	4.12310563	**1**	7.21110255
B	4.12310563	0	7.07106781	3.16227766	7
C	4.12310563	7.07106781	0	4.47213595	5.38516481
D	**1**	3.16227766	4.47213595	0	6.70820393
E	7.21110255	7	5.38516481	6.70820393	0

Now we merge A and D but instead of keeping the smallest distance we select the largest distance value

	A, D	B	C	E
A, D	0	**4.12310563**	4.47213595	7.21110255
B	**4.12310563**	0	7.07106781	7

	A, D	**B**	**C**	**E**
C	4.47213595	7.07106781	0	5.38516481
E	7.21110255	7	5.38516481	0

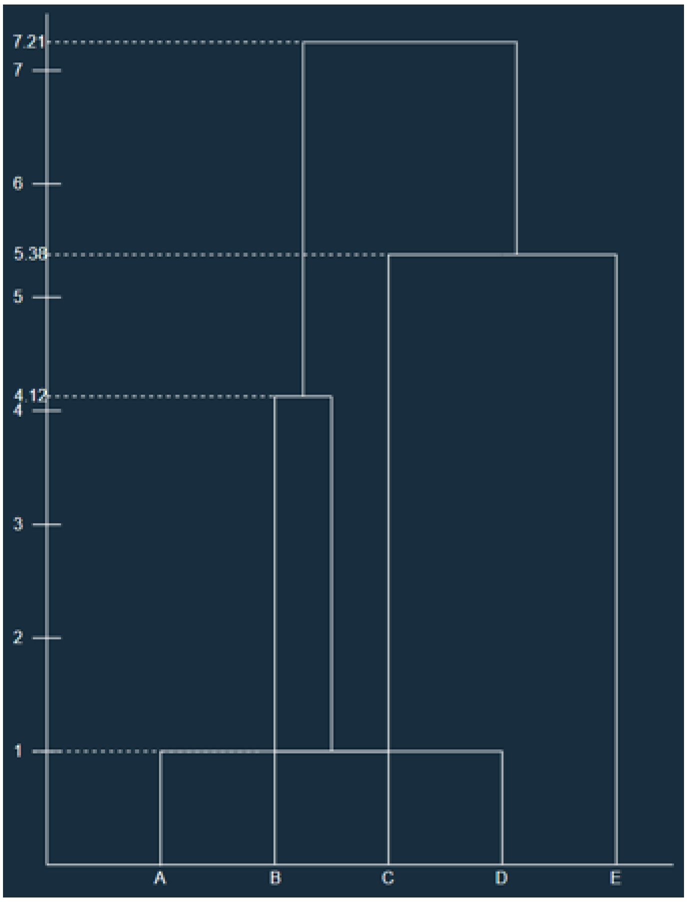
And we continue these iterations till we have 1 cluster

	A, B, D	**C**	**E**
A, B, D	0	7.07106781	7.21110255
C	7.07106781	0	**5.38516481**
E	7.21110255	**5.38516481**	0

Finally we get

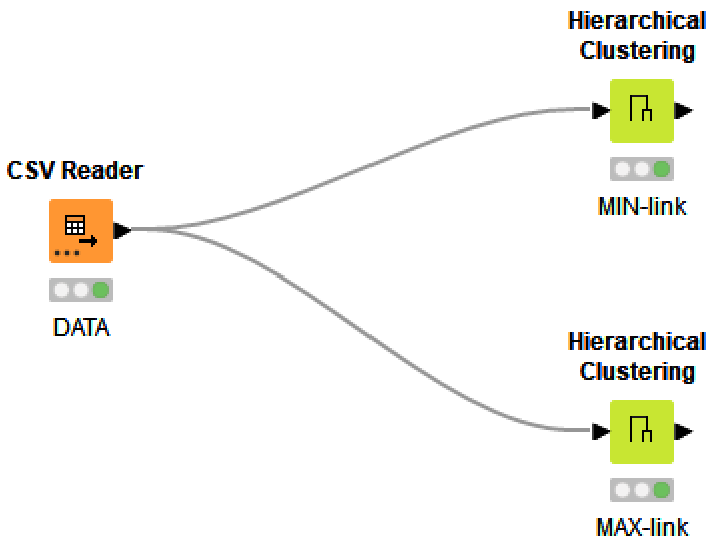
	A, B, D	**C, E**
A, B, D	0	**7.21110255**
C, E	**7.21110255**	0

The dendogram looks like this:

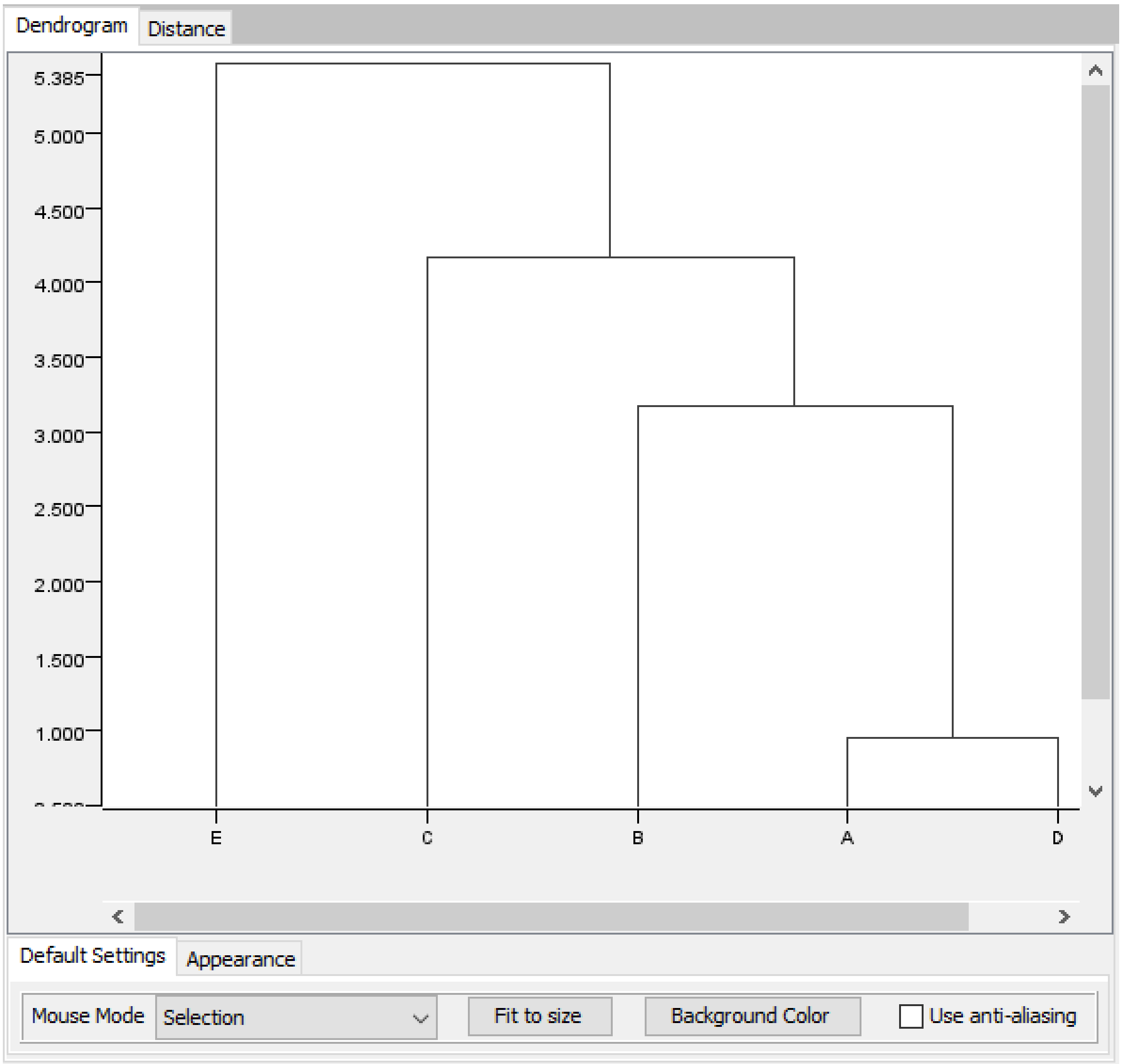


Task 2c

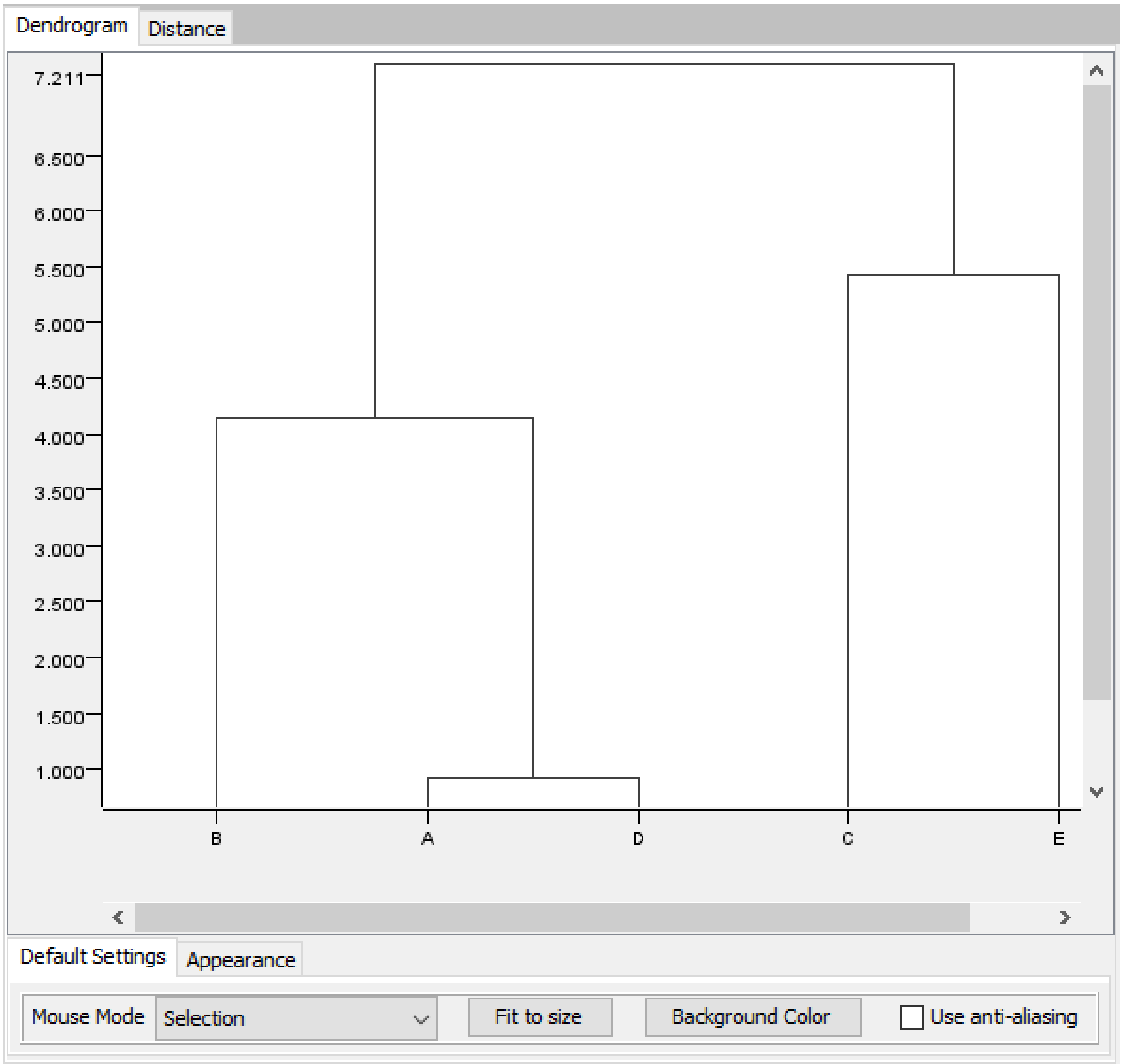
Workflow:



Min-link:



Max-link:

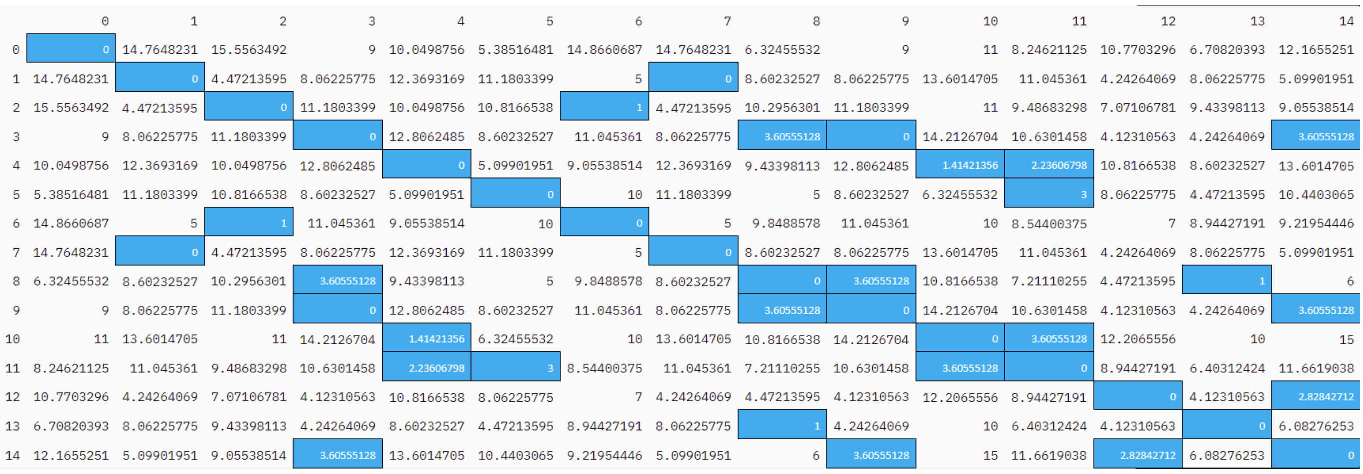


Mouse Mode: Selection

Task 3

Task 3a

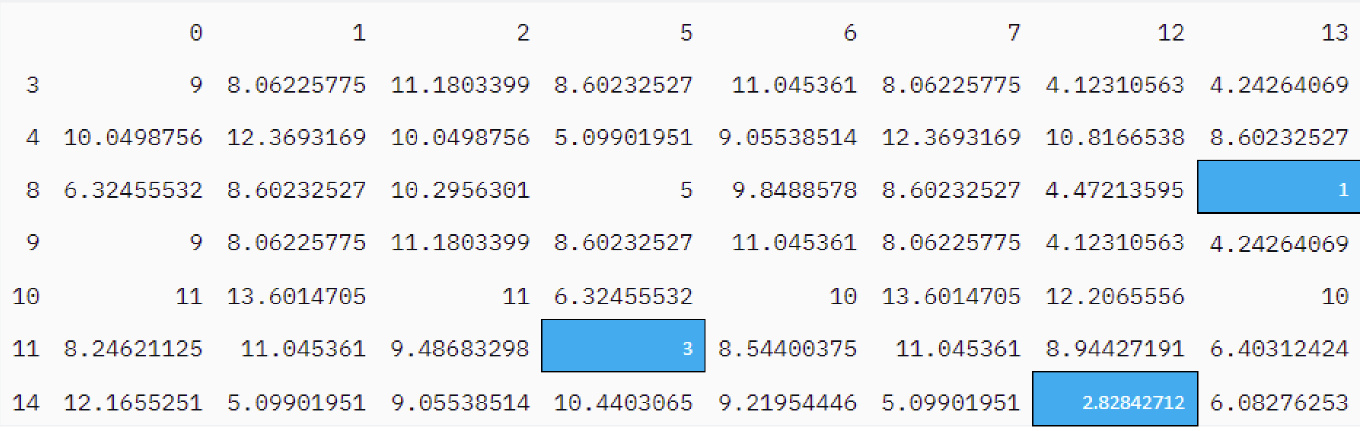
We start by calculating the distance between each point and finding points that are close to 3 other points including self with max distance of 4



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	14.7648231	15.5563492	9	10.0498756	5.38516481	14.8660687	14.7648231	6.32455532	9	11	8.24621125	10.7703296	6.70820393	12.1655251
1	14.7648231	0	4.47213595	8.06225775	12.3693169	11.1803399	5	0	8.60232527	8.06225775	13.6014705	11.045361	4.24264069	8.06225775	5.09901951
2	15.5563492	4.47213595	0	11.1803399	10.0498756	10.8166538	1	4.47213595	10.2956301	11.1803399	11	9.48683298	7.07106781	9.43398113	9.05538514
3	9	8.06225775	11.1803399	0	12.8062485	8.60232527	11.045361	8.06225775	3.60555128	0	14.2126704	10.6301458	4.12310563	4.24264069	3.60555128
4	10.0498756	12.3693169	10.0498756	12.8062485	0	5.09901951	9.05538514	12.3693169	9.43398113	12.8062485	1.41421356	2.23606798	10.8166538	8.60232527	13.6014705
5	5.38516481	11.1803399	10.8166538	8.60232527	5.09901951	0	10	11.1803399	5	8.60232527	6.32455532	3	8.06225775	4.47213595	10.4403065
6	14.8660687	5	1	11.045361	9.05538514	10	0	5	9.8488578	11.045361	10	8.54400375	7	8.94427191	9.21954446
7	14.7648231	0	4.47213595	8.06225775	12.3693169	11.1803399	5	0	8.60232527	8.06225775	13.6014705	11.045361	4.24264069	8.06225775	5.09901951
8	6.32455532	8.60232527	10.2956301	3.60555128	9.43398113	5	9.8488578	8.60232527	0	3.60555128	10.8166538	7.21110255	4.47213595	1	6
9	9	8.06225775	11.1803399	0	12.8062485	8.60232527	11.045361	8.06225775	3.60555128	0	14.2126704	10.6301458	4.12310563	4.24264069	3.60555128
10	11	13.6014705	11	14.2126704	1.41421356	6.32455532	10	13.6014705	10.8166538	14.2126704	0	3.60555128	12.2065556	10	15
11	8.24621125	11.045361	9.48683298	10.6301458	2.23606798	3	8.54400375	11.045361	7.21110255	10.6301458	3.60555128	0	8.94427191	6.40312424	11.6619038
12	10.7703296	4.24264069	7.07106781	4.12310563	10.8166538	8.06225775	7	4.24264069	4.47213595	4.12310563	12.2065556	8.94427191	0	4.12310563	2.82842712
13	6.70820393	8.06225775	9.43398113	4.24264069	8.60232527	4.47213595	8.94427191	8.06225775	1	4.24264069	10	6.40312424	4.12310563	0	6.08276253
14	12.1655251	5.09901951	9.05538514	3.60555128	13.6014705	10.4403065	9.21954446	5.09901951	6	3.60555128	15	11.6619038	2.82842712	6.08276253	0

We then get that the core poinst are [3, 4, 8, 9, 10, 11, 14]

Next we compare the core points with every other point to get border points.

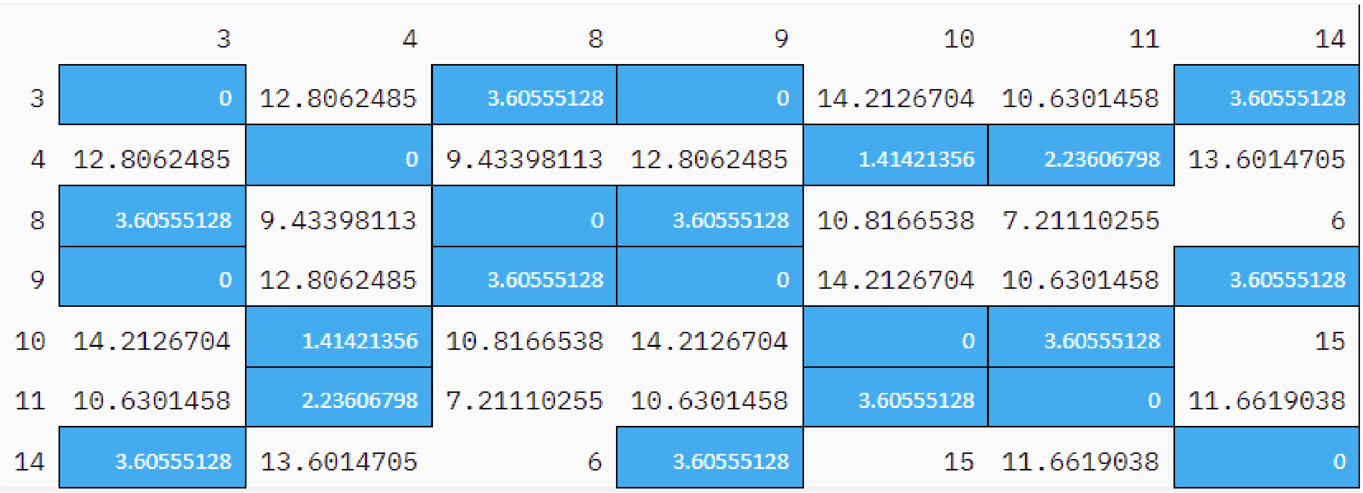


	0	1	2	5	6	7	12	13
3	9	8.06225775	11.1803399	8.60232527	11.045361	8.06225775	4.12310563	4.24264069
4	10.0498756	12.3693169	10.0498756	5.09901951	9.05538514	12.3693169	10.8166538	8.60232527
8	6.32455532	8.60232527	10.2956301	5	9.8488578	8.60232527	4.47213595	1
9	9	8.06225775	11.1803399	8.60232527	11.045361	8.06225775	4.12310563	4.24264069
10	11	13.6014705	11	6.32455532	10	13.6014705	12.2065556	10
11	8.24621125	11.045361	9.48683298	3	8.54400375	11.045361	8.94427191	6.40312424
14	12.1655251	5.09901951	9.05538514	10.4403065	9.21954446	5.09901951	2.82842712	6.08276253

Then we find that [5, 12, 13] are border points

That also concludes that [0, 1, 2, 6, 7] are noise points

We can then find which of the core points are close to each other to merge them into one cluster



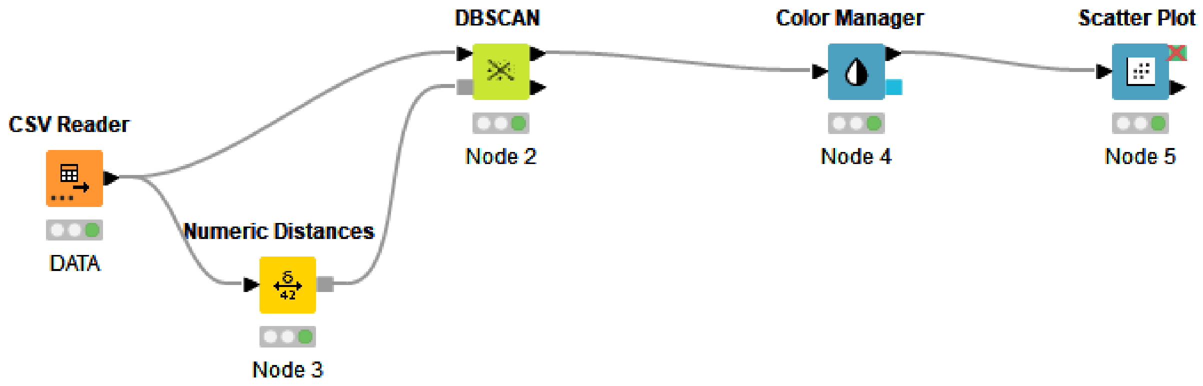
	3	4	8	9	10	11	14
3	0	12.8062485	3.60555128	0	14.2126704	10.6301458	3.60555128
4	12.8062485	0	9.43398113	12.8062485	1.41421356	2.23606798	13.6014705
8	3.60555128	9.43398113	0	3.60555128	10.8166538	7.21110255	6
9	0	12.8062485	3.60555128	0	14.2126704	10.6301458	3.60555128
10	14.2126704	1.41421356	10.8166538	14.2126704	0	3.60555128	15
11	10.6301458	2.23606798	7.21110255	10.6301458	3.60555128	0	11.6619038
14	3.60555128	13.6014705	6	3.60555128	15	11.6619038	0

Where we finally get 2 clusters

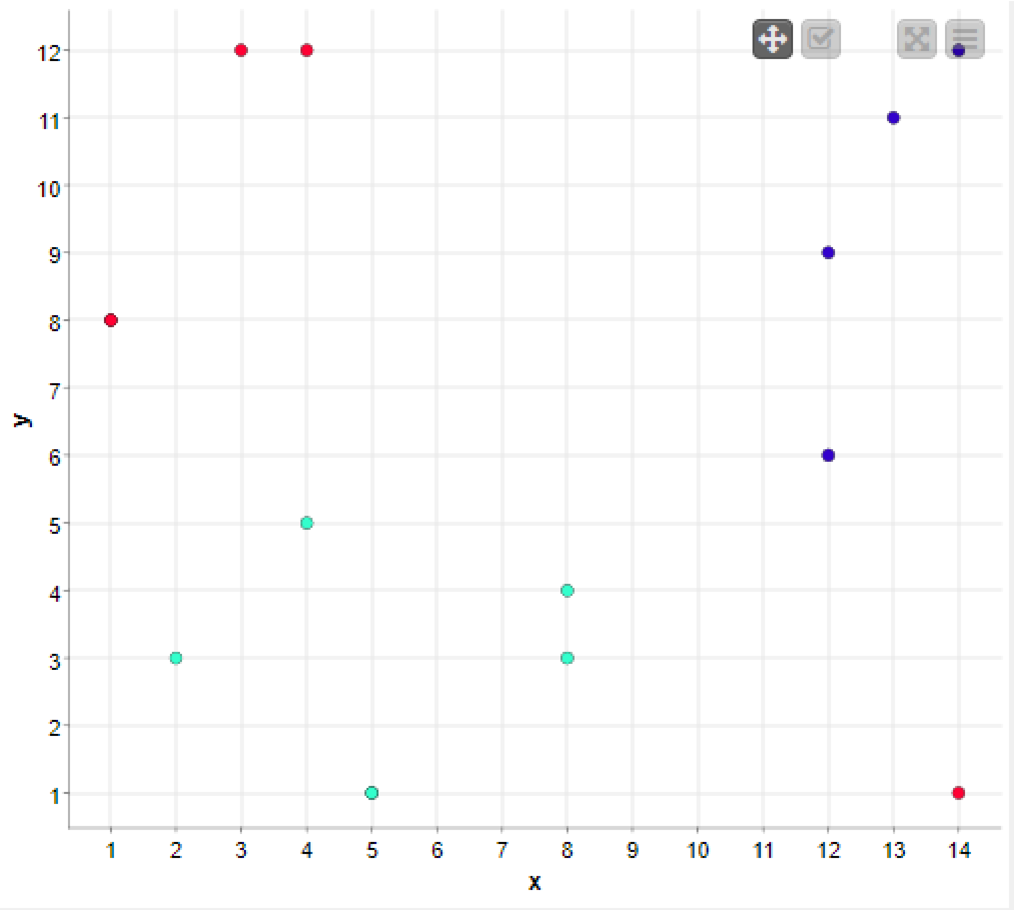
Cluster 1	Cluster 2
3	4
8	10
9	11
14	5
12	
13	

Task 2b

Workflow:



Plot:



Here we can see cluster 1 in light blue, cluster 2 in purple and the noise as red.