TDT4300 - Assignment 3

Andreas Amundsen
Magnus Lauritzen Holtet

Table of Contents

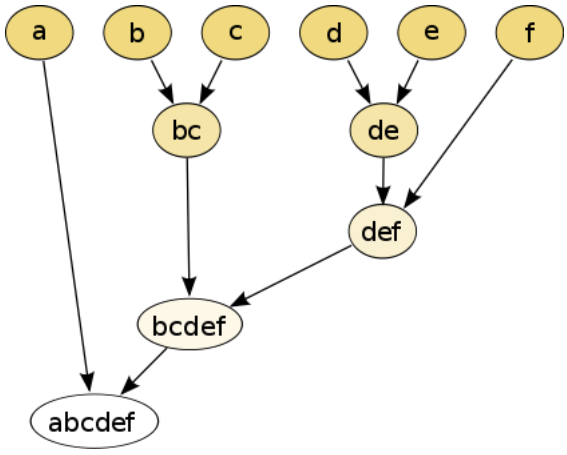
Task 1

Se attached files.

Task 2

a)

Hierarchical clustering works by building clusters step by step. Hierarchical agglomerative clustering (HAC) works by initially starting with each data point as one cluster, and then merging those who are close to each other, ending up with one cluster or K clusters. This works in a bottom up fashion, as shown in the illustration below.



MIN-link finds the elements with the closest distance between them, and is used in the single-linkage clustering

$$\min\{\, d(x, y) : x \in \mathcal{A},\, y \in \mathcal{B} \,\}$$

MAX-link finds the elements with the furthest distance between them, and is used in complete-linkage clustering

$$\max\{\, d(x, y) : x \in \mathcal{A},\, y \in \mathcal{B} \,\}$$

b)

Steps when performing HAC

1) Start by **computing the distance**, using Euclidean distance method, between every pair of nodes that you want to cluster, which will produce a distance matrix. Along the diagonal axis the distance will be zero, since the distance between 1 and 2 is the same as the distance between 2 and 1.
2) Determine the two nodes with the largest/lowest (complete/single linkage) distance between them using the distance matrix, and **merge** them together
3) To calculate the new distance matrix, remove the nodes you just connected, and insert them as one node in the distance matrix.
4) Perform step 2) again

To illustrate the difference between complete and single linkage, let us look at an example. You combine node 3 and 5, which gives node "35". In complete linkage (same as MAX-link), you find the distance between another node, say node 1, to 3 and 5, and then take the maximum between those two. dist(1, 3) = 3 and dist(1, 5) = 11, will give dist(1, "35) = 11.

In single-linkage (MIN-link) the minimum distance between these two will be considered as the distance between node 1 and "35", which gives dist(1, "35") = 3

For MIN-link

Determine which has the smallest value in the distance matrix, and connect the two points in to a shared cluster ((B, C) in the next table)

	A	**B**	**C**	**D**	**E**	**F**
A	0					
B	5,1	0				
C	4,1	1	0			
D	5	6,4	5,7	0		
E	7,6	6,3	6,1	3,6	0	
F	10,3	8	8,1	6,4	2,8	0

Repeat the step from above

	A	**B, C**	**D**	**E**	**F**
A	0				
B, C	4,1	0			
D	5	5,7	0		
E	7,6	6,1	3,6	0	
F	10,3	8	6,4	2,8	0

Repeat the step from above

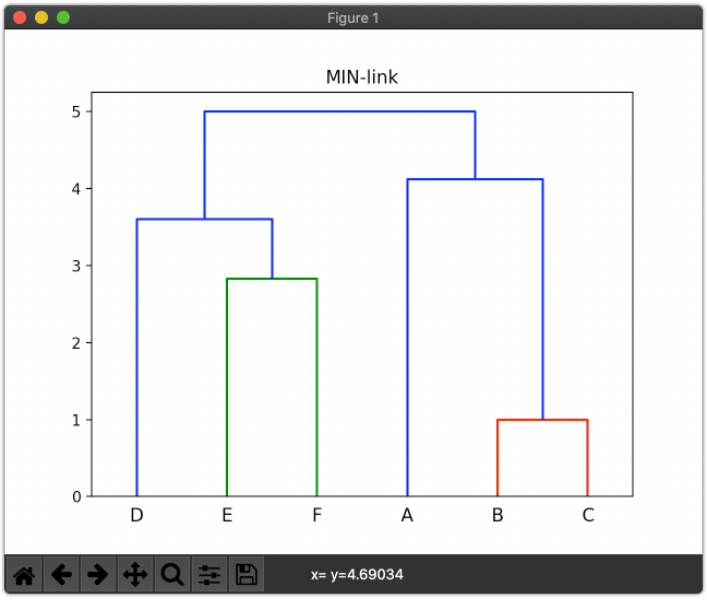
	A	**B, C**	**D**	**E, F**
A	0			
B, C	4,1	0		
D	5	5,7	0	
E, F	7,6	6,1	3,6	0

Repeat the step from above

	A, B, C	**D**	**E, F**
A, B, C	0		
D	5	0	
E, F	6,1	3,6	0

After a few iterations you end up with two clusters, as shown below

	A, B, C	**D, E, F**
A, B, C	0	
D, E, F	5	0



This illustration is created with a python script called dendrogram.py [0], gathered from a fellow student on Piazza.

For MAX-link

Determine the smallest value, but choose the largest distance as the new distance

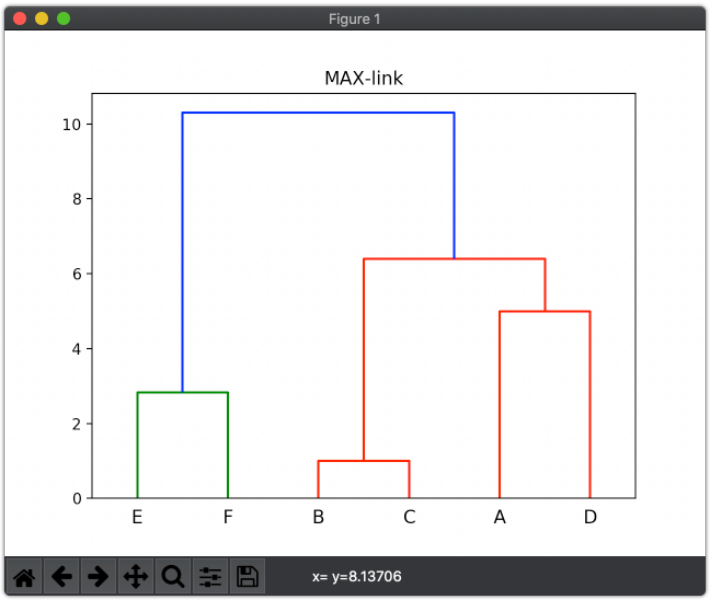
	A	B	C	D	E	F
A	0					
B	5,1	0				
C	4,1	1	0			
D	5	6,4	5,7	0		
E	7,6	6,3	6,1	3,6	0	
F	10,3	8	8,1	6,4	2,8	0

	A	B, C	D	E	F
A	0				
B, C	5,1	0			
D	5	6,4	0		
E	7,6	6,3	3,6	0	
F	10,3	8,1	6,4	2,8	0

	A	B, C	D	E, F
A	0			
B, C	5,1	0		
D	5	6,4	0	
E, F	10,3	8,1	6,4	0

	A, D	B, C	E, F
A, D	0		
B, C	6, 4	0	
E, F	10,3	8,1	0

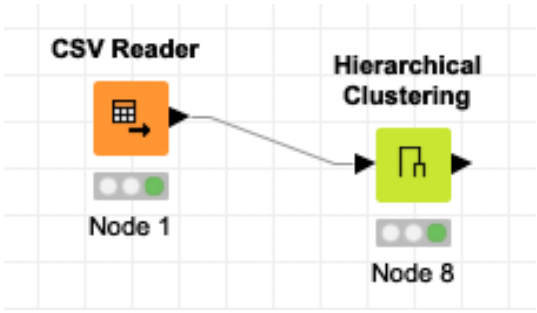
	A, B, C, D	E, F
A, B, C, D	0	
E, F	10,3	0



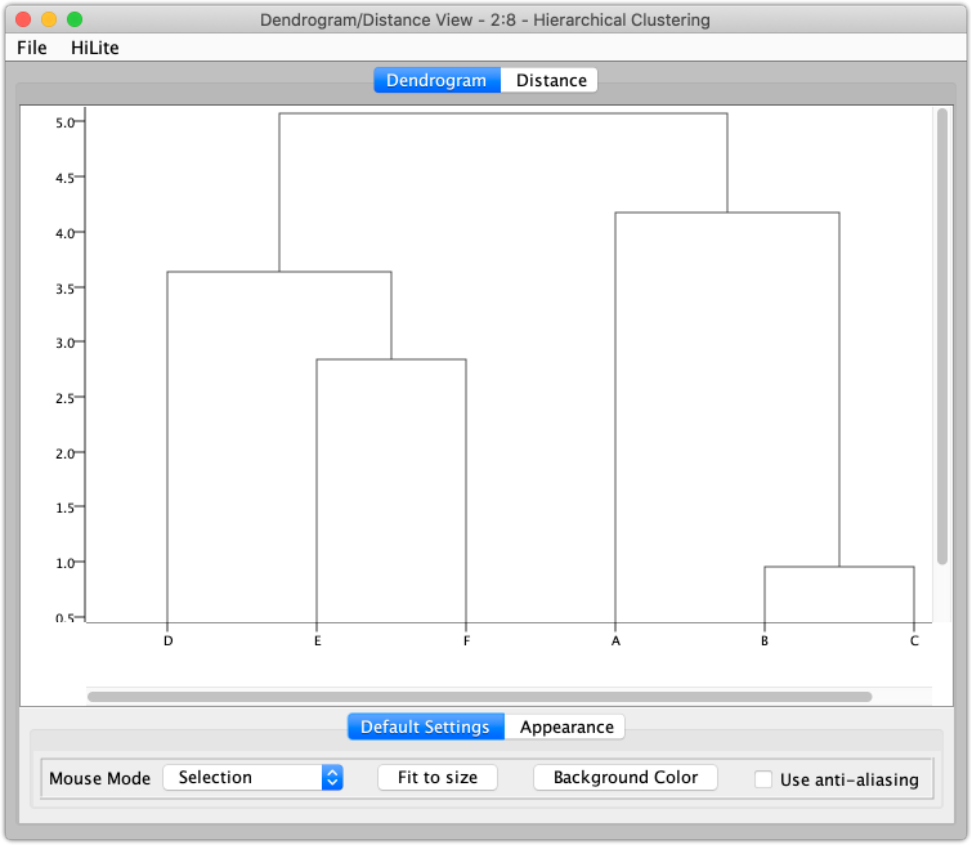
This illustration is created with a python script called dendrogram.py [0], gathered from a fellow student on Piazza.

c)

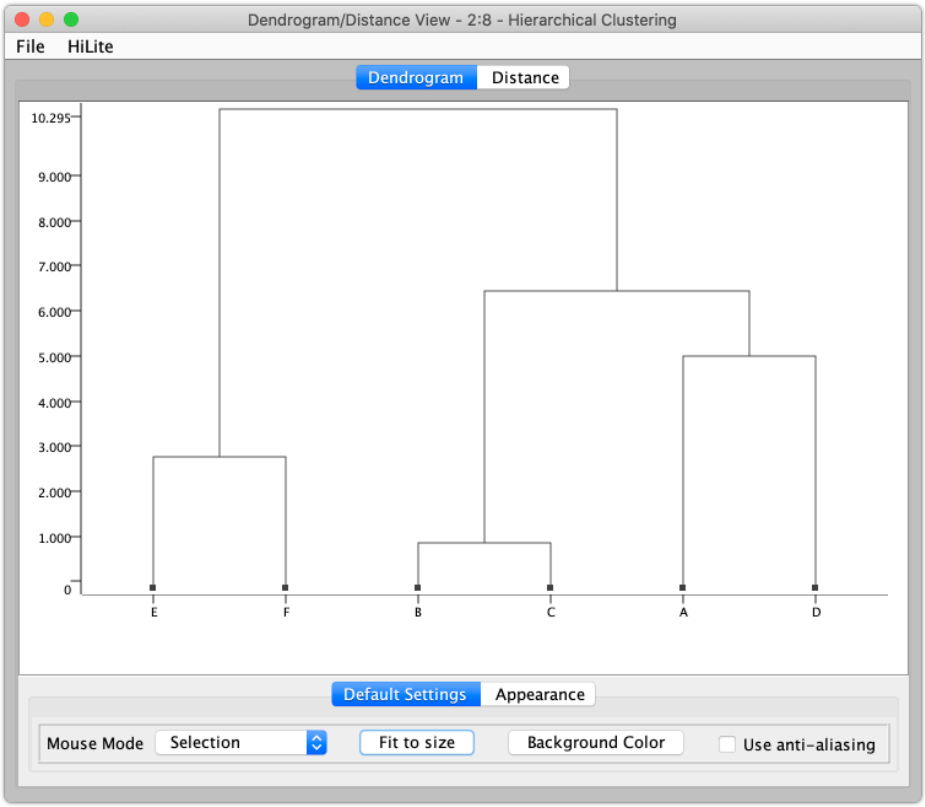
Workflow



For MIN-link



For MAX-link

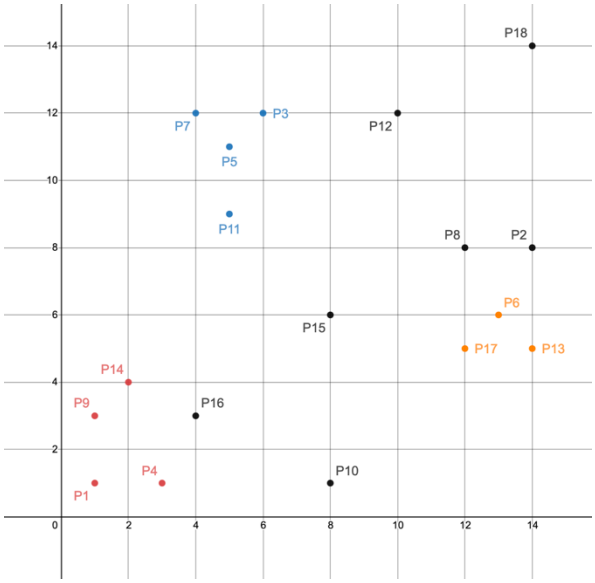


Task 3

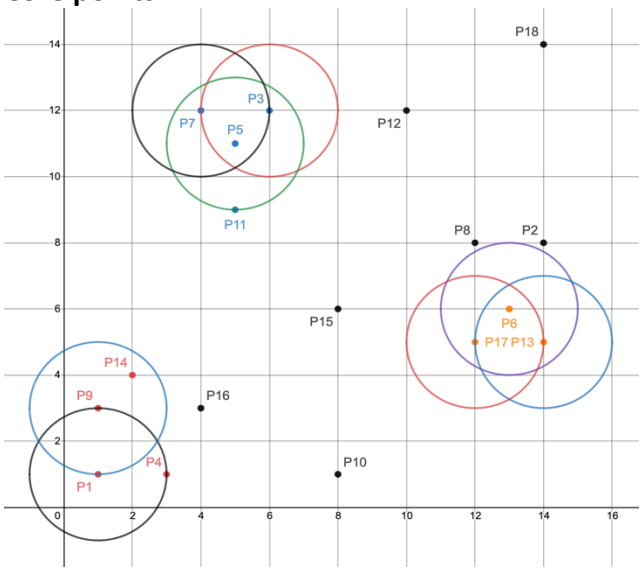
a)

Interactive plot can be found here: https://www.desmos.com/calculator/r9neriz6lt

Data points



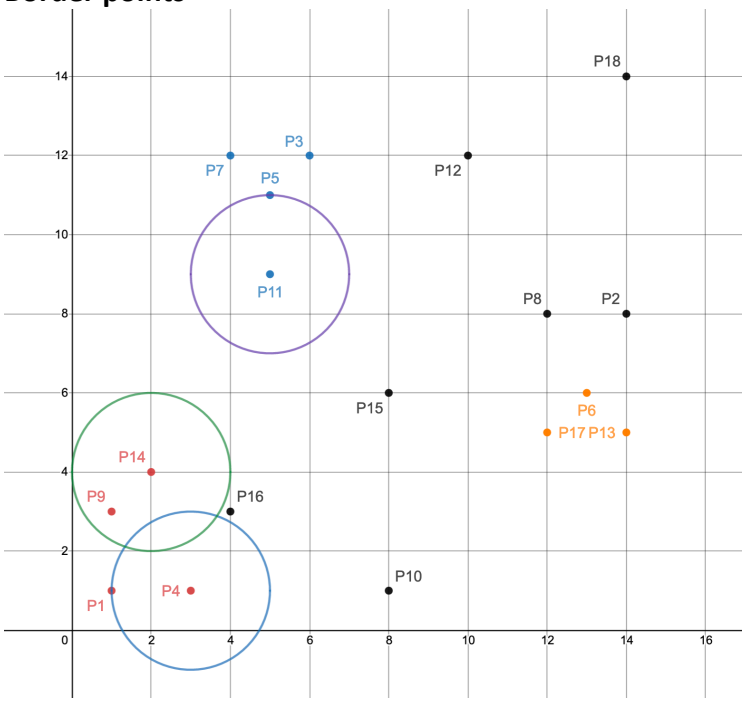
Core points



The circles illustrate *EPS = 2.*

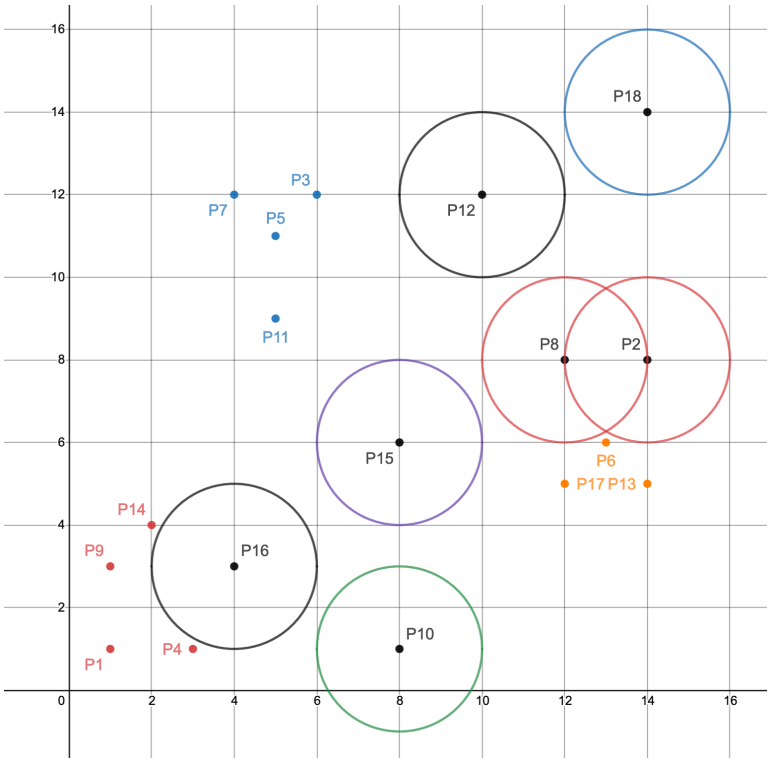
The highlighted points are the *core points*, as there are more than or equal to *minPts = 3* points inside the circle.

Border points



The points that have less than *minPts = 3* points in its *range*, yet is a part of another core nodes neighborhood.

Noise



The points that are neither *core* or *border* points.

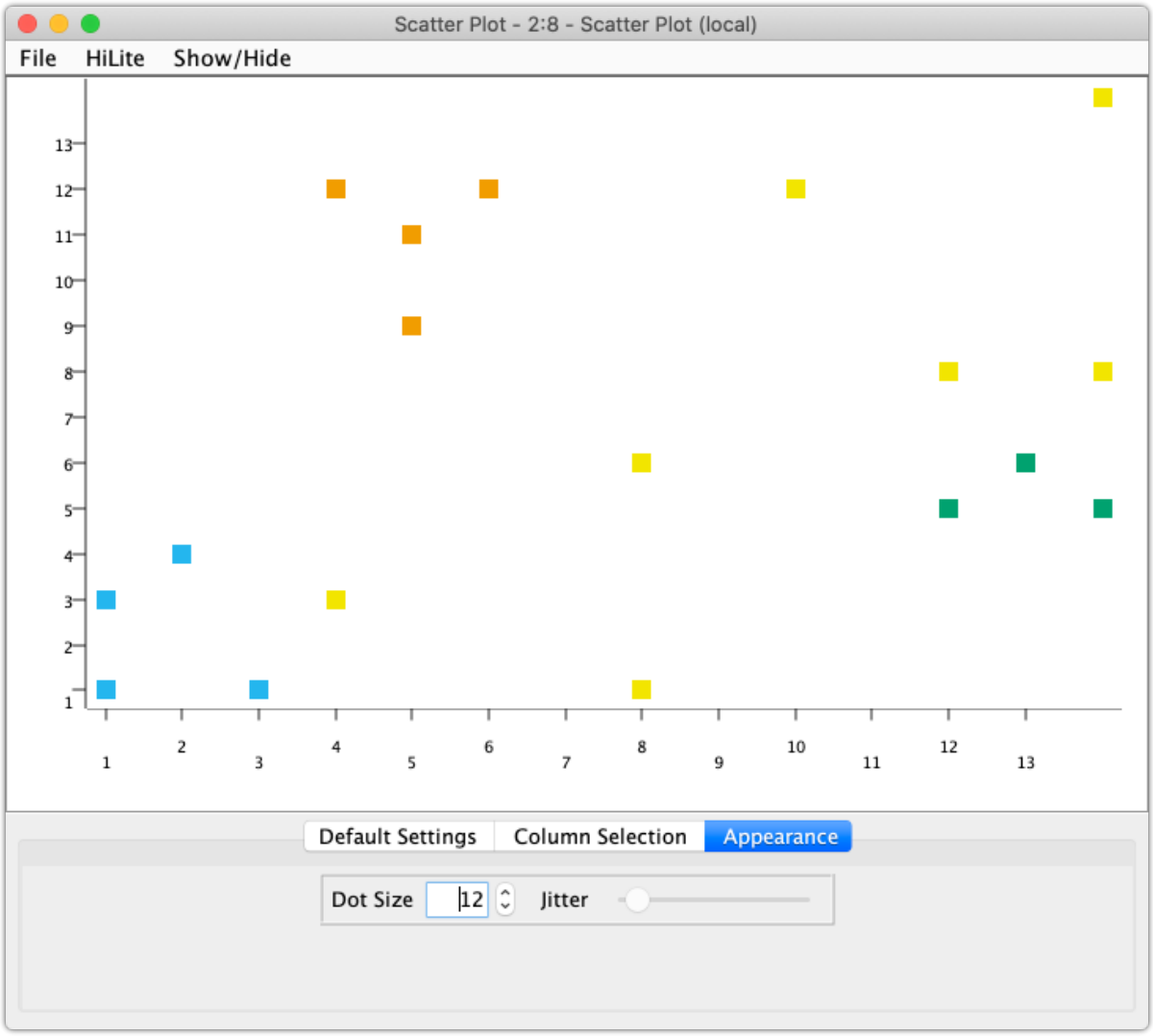
Code

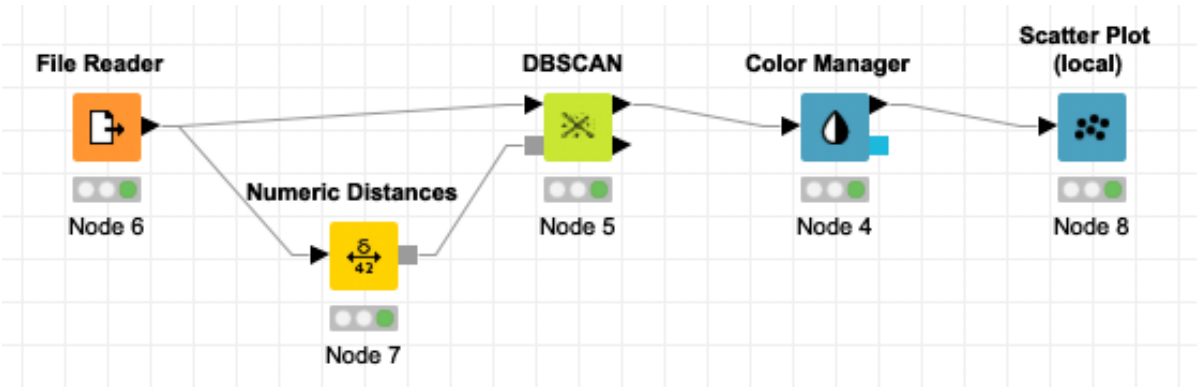
A file, *scan.py*, is provided with this delivery. It verifies the results found above, as well as the ones found with KNIME.

The output of the script looks like this:

```
▶ python3 scan.py
TDT4300 - Magnus Lauritzen Holtet & Andreas Amundsen
All points:
P1 (1,1)        => cluster 1
P2 (14,8)       => NOISE
P3 (6,12)       => cluster 2
P4 (3,1)        => cluster 1
P5 (5,11)       => cluster 2
P6 (13,6)       => cluster 3
P7 (4,12)       => cluster 2
P8 (12,8)       => NOISE
P9 (1,3)        => cluster 1
P10 (8,1)       => NOISE
P11 (5,9)       => cluster 2
P12 (10,12)     => NOISE
P13 (14,5)      => cluster 3
P14 (2,4)       => cluster 1
P15 (8,6)       => NOISE
P16 (4,3)       => NOISE
P17 (12,5)      => cluster 3
P18 (14,14)     => NOISE

Cores:
P1 (1, 1)
P3 (6, 12)
P5 (5, 11)
P6 (13, 6)
P7 (4, 12)
P9 (1, 3)
P13 (14, 5)
P17 (12, 5)

Noise:
P2 (14, 8)
P8 (12, 8)
P10 (8, 1)
P12 (10, 12)
P15 (8, 6)
P16 (4, 3)
P18 (14, 14)

Border points:
P4 (3, 1)
P11 (5, 9)
P14 (2, 4)
```

b)


Scatter Plot - 2:8 - Scatter Plot (local)
File
HiLite
Show/Hide
Default Settings
Column Selection
Appearance
Dot Size
12
Jitter


File Reader
Node 6
Numeric Distances
Node 7
DBSCAN
Node 5
Color Manager
Node 4
Scatter Plot (local)
Node 8

Sources

[0] https://piazza.com/class/k5iddfq95ew1y7?cid=58