

COMP3340 Data Mining Assignment 1

Eden Fagerstrom

C3328365

Q1)

a)

These Matrices were written to excel using write.csv() in R if the full files are needed please notify me.

Hamming Distance matrix between all pairs of elections:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Years	1864	1868	1872	1880	1888	1900	1904	1908	1916	1924	1928	1936	1940	1944	1948	1956	1964	1972	1860	1876	1884	1892	1896
1864	0	5	6	5	2	5	6	5	1	5	6	5	6	5	4	5	3	0	6	8	6	2	4
1868	5	0	3	4	7	6	3	4	6	6	3	4	3	2	5	4	6	5	5	7	3	7	5
1872	6	3	0	5	4	3	0	5	5	2	3	2	1	4	1	3	6	6	6	4	8	8	8
1880	5	4	5	0	5	6	5	4	4	6	3	6	5	4	5	6	4	5	3	5	3	7	5
1888	2	7	4	5	0	3	4	5	1	5	4	5	6	5	4	3	1	2	6	6	6	4	6
1900	5	6	3	6	3	0	3	6	4	4	3	4	5	4	5	2	2	5	7	7	7	5	9
1904	6	3	0	5	4	3	0	5	5	5	2	3	2	1	4	1	3	6	6	4	8	8	8
1908	5	4	5	4	5	6	5	0	4	6	3	4	3	4	3	6	6	5	7	3	7	5	7
1916	1	6	5	4	1	4	5	4	0	4	5	4	5	4	3	4	2	1	7	7	7	3	5
1924	5	6	5	6	5	4	5	6	4	0	5	4	5	4	3	4	4	5	7	9	5	9	9
1928	6	3	2	3	4	3	2	3	5	5	0	5	4	3	4	3	3	6	4	4	8	8	8
1936	5	4	3	6	5	4	3	4	4	4	5	0	1	2	5	2	4	5	9	7	7	5	5
1940	6	3	2	5	6	5	2	3	5	5	4	1	0	1	4	3	5	6	8	6	6	8	6
1944	5	2	1	4	5	4	1	4	4	4	3	2	1	0	3	2	4	5	7	7	5	7	7
1948	4	5	4	5	4	5	4	3	3	3	4	5	4	3	0	5	5	4	6	6	8	4	8
1956	5	4	1	6	3	2	1	6	4	4	3	2	3	2	5	0	2	5	7	7	5	7	7
1964	3	6	3	4	1	2	3	6	2	4	3	4	5	4	5	2	0	3	5	7	5	5	7
1972	0	5	6	5	2	5	6	5	1	5	6	5	6	5	4	5	3	0	6	8	6	2	4
1860	6	5	6	3	6	7	6	7	7	7	4	9	8	7	6	7	5	6	0	6	2	6	6
1876	8	7	6	5	6	7	6	3	7	7	4	7	6	7	6	7	7	8	6	0	6	10	6
1884	6	3	4	3	6	7	4	7	7	9	4	7	6	5	8	5	5	6	2	6	0	8	4
1892	2	7	8	7	4	5	8	7	3	5	8	7	8	7	4	7	5	2	6	10	8	0	6
1896	4	5	8	5	6	9	8	5	5	9	8	5	6	7	8	7	7	4	6	6	4	6	0
1912	7	6	3	4	5	4	3	6	6	4	3	6	5	4	3	4	4	7	3	5	5	7	9
1920	4	5	8	3	6	9	8	3	5	9	6	7	6	7	6	9	7	4	4	4	4	6	2
1932	5	6	5	8	5	4	5	4	6	8	5	6	5	6	5	6	6	5	7	5	7	5	7
1952	7	8	7	4	5	6	7	6	6	8	5	10	9	8	7	8	6	7	5	3	5	7	7
1960	7	6	5	6	5	4	5	2	6	8	3	6	5	6	5	6	6	7	7	3	7	7	7
1968	6	3	6	3	8	7	6	5	7	5	4	7	6	5	4	7	7	6	2	6	4	6	6
1976	6	7	4	5	4	5	4	5	5	5	4	7	6	5	4	5	5	6	6	2	6	8	8
1980	8	11	8	7	6	5	8	9	7	5	8	7	8	9	8	7	5	8	6	6	8	6	8

Jaccard Distance matrix between all pairs of elections:

Years	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
		1864	1868	1872	1880	1888	1900	1904	1908	1916	1924	1928	1936	1940	1944	1948	1956	1964	1972	1860	1876	1884	1892	1896
1864	0	0.714286	0.857143	0.833333	0.666667	0.833333	0.857143	0.833333	0.333333	0.714286	0.1714286	0.1714286	0.75	0.714286	0.666667	0.833333	0.75	0	0.857143	1	0.857143	0.4	0.666667	0.625
1868	0.714286	0	0.428571	0.571429	1	0.75	0.428571	0.571429	0.857143	0.666667	0.5	0.5	0.375	0.285714	0.625	0.571429	0.857143	0.714286	0.625	0.777778	0.428571	0.777778	0.625	0.888889
1872	0.857143	0.428571	0	0.714286	0.8	0.5	0	0.714286	0.833333	0.625	0.4	0.428571	0.285714	0.166667	0.571429	0.2	0.6	0.857143	0.75	0.75	0.571429	0.888889	0.888889	0.888889
1880	0.833333	0.571429	0.714286	0	1	0.857143	0.714286	0.666667	0.8	0.75	0.6	0.75	0.625	0.571429	0.714286	0.857143	0.8	0.833333	0.5	0.714286	0.5	0.875	0.714286	0.888889
1888	0.666667	1	0.8	1	0	0.75	0.8	1	0.5	0.833333	1	0.833333	0.857143	0.833333	0.8	0.75	0.5	0.666667	1	1	1	0.8	1	0.888889
1900	0.833333	0.75	0.5	0.857143	0.75	0	0.5	0.857143	0.8	0.571429	0.6	0.571429	0.625	0.571429	0.714286	0.4	0.5	0.833333	0.875	0.875	0.875	0.714286	1	0.888889
1904	0.857143	0.428571	0	0.714286	0.8	0.5	0	0.714286	0.833333	0.625	0.4	0.428571	0.285714	0.166667	0.571429	0.2	0.6	0.857143	0.75	0.75	0.571429	0.888889	0.888889	0.888889
1908	0.833333	0.571429	0.714286	0.666667	1	0.857143	0.714286	0	0.8	0.75	0.6	0.75	0.625	0.571429	0.428571	0.571429	0.5	0.857143	1	0.833333	0.875	0.875	0.714286	0.888889
1916	0.333333	0.857143	0.833333	0.8	0.5	0.8	0.833333	0.8	0	0.666667	0	0.666667	0.714286	0.666667	0.6	0.8	0.666667	0.333333	1	1	1	0.6	0.833333	0.888889
1924	0.714286	0.666667	0.625	0.75	0.833333	0.571429	0.625	0.75	0.666667	0	0.714286	0.5	0.555556	0.5	0.428571	0.571429	0.666667	0.714286	0.777778	0.777778	0.9	0.625	0.9	0.888889
1928	1	0.5	0.4	0.6	1	0.6	0.4	0.6	1	0.714286	0	0.714286	0.571429	0.5	0.666667	0.6	0.75	1	0.666667	0.666667	0.666667	1	1	0.888889
1936	0.714286	0.5	0.428571	0.75	0.833333	0.571429	0.428571	0.571429	0.666667	0.5	0.714286	0	0.142857	0.285714	0.625	0.333333	0.666667	0.714286	0.9	0.777778	0.777778	0.777778	0.625	0.888889
1940	0.75	0.375	0.285714	0.625	0.857143	0.428571	0.714286	0.555556	0.571429	0.142857	0	0.142857	0.5	0.428571	0.714286	0.75	0.8	0.666667	0.666667	0.666667	0.666667	0.666667	0.666667	0.666667
1944	0.714286	0.285714	0.166667	0.571429	0.833333	0.571429	0.166667	0.571429	0.666667	0.5	0.5	0.285714	0.142857	0	0.428571	0.333333	0.666667	0.714286	0.777778	0.777778	0.625	0.777778	0.777778	0.777778
1948	0.666667	0.625	0.571429	0.714286	0.8	0.714286	0.571429	0.5	0.6	0.428571	0.666667	0.625	0.5	0.428571	0	0.714286	0.833333	0.666667	0.75	0.75	0.888889	0.571429	0.888889	
1956	0.833333	0.571429	0.2	0.857143	0.75	0.4	0.2	0.857143	0.8	0.571429	0.6	0.333333	0.428571	0.333333	0.714286	0	0.5	0.833333	0.875	0.875	0.714286	0.875	0.875	0.875
1964	0.75	0.857143	0.6	0.8	0.5	0.5	0.6	1	0.666667	0.666667	0.75	0.666667	0.714286	0.666667	0.833333	0.5	0	0.75	0.833333	1	0.833333	0.833333	1	0.888889
1972	0	0.714286	0.857143	0.833333	0.666667	0.833333	0.857143	0.833333	0.333333	0.714286	1	0.714286	0.75	0.714286	0.666667	0.833333	0.75	0	0.857143	1	0.857143	0.4	0.666667	0.666667
1860	0.857143	0.625	0.75	0.5	1	0.875	0.75	0.875	1	0.777778	0.666667	0.9	0.8	0.777778	0.75	0.875	0.833333	0.857143	0	0.75	0.333333	0.75	0.75	0.75
1876	1	0.777778	0.75	0.714286	1	0.875	0.75	0.5	1	0.777778	0.666667	0.777778	0.666667	0.777778	0.75	0.875	1	1	0.75	0	0.75	1	0.75	0.75
1884	0.857143	0.428571	0.571429	0.5	1	0.875	0.571429	0.875	1	0.9	0.666667	0.777778	0.666667	0.625	0.888889	0.714286	0.833333	0.857143	0.333333	0.75	0	0.888889	0.571429	0.888889
1892	0.4	0.777778	0.888889	0.875	0.8	0.714286	0.888889	0.875	0.6	0.625	1	0.777778	0.8	0.777778	0.571429	0.875	0.833333	0.4	0.75	1	0.888889	0	0.75	0.75
1896	0.666667	0.625	0.888889	0.714286	1	1	0.888889	0.714286	0.833333	0.9	1	0.625	0.666667	0.777778	0.888889	0.875	1	0.666667	0.75	0.75	0.571429	0.75	0.571429	0.75
1912	0.875	0.666667	0.428571	0.571429	0.833333	0.571429	0.428571	0.75	0.857143	0.5	0.5	0.666667	0.555556	0.5	0.428571	0.571429	0.666667	0.875	0.428571	0.625	0.625	0.777778	0.9	0.777778
1920	0.666667	0.625	0.888889	0.5	1	1	0.888889	0.5	0.833333	0.9	0.857143	0.777778	0.666667	0.777778	0.75	1	1	0.666667	0.571429	0.571429	0.571429	0.75	0.333333	0.888889
1932	0.714286	0.666667	0.625	0.888889	0.833333	0.571429	0.625	0.571429	0.857143	0.833333	0.714286	0.666667	0.555556	0.666667	0.625	0.75	0.857143	0.714286	0.777778	0.625	0.777778	0.625	0.777778	0.777778
1940	1	0.888889	0.888889	0.666667	1	0.857143	0.875	0.857143	1	0.888889	0.833333	1	0.888889	0.888889	0.875	1	1	0.714286	0.875	0.714286	0.875	0.875	0.875	0.875
1950	1	0.75	0.714286	0.857143	1	0.666667	0.714286	0.4	1	0.888889	0.6	0.75	0.625	0.75	0.714286	0.857143	1	1	0.875	0.5	0.875	0.875	0.875	0.875
1960	0.75	0.875	0.857143	0.666667	0.888889	0.777778	0.666667	0.666667	0.625	0.888889	0.875	0.888889	0.875	0.888889	0.875	0.888889	0.875	0.888889	0.875	0.888889	0.875	0.888889	0.888889	0.888889
1976	0.857143	0.714286	0.571429	0.714286	0.8	0.714286	0.571429	0.714286	0.833333	0.625	0.666667	0.777778	0.666667	0.625	0.571429	0.714286	0.833333	0.857143	0.75	0.333333	0.75	0.888889	0.888889	0.888889
1980	0.888889	0.916667	0.8	0.777778	0.875	0.8	0.777778	0.875	0.43	0.9	0.875	0.555556	0.888889	0.7	0.727273	0.818182	0.8	0.777778	0.714286	0.888889	0.666667	0.666667	0.666667	0.666667

removed. The empty matrix “emptyMat” was assigned to 2 new variables which will be used to store the values of the final matrices; “hammyMat”, “jaccMat”. Two variables i and j were assigned 1 before a nested while loop was constructed where the outer loop is executed while i is less than the number of rows of emptyMat +1 (since we’re starting at 1) and i is incremented by 1 after the inner loop completes its first cycle, the inner loop is executed while j is less than the number of columns of emptyMat +1 and within the innerloop we call the functions “hammyDist” and “jaccDist” which take took arguments of the rows of the US Presidency dataset and will calculate the hamming distances and jaccard distances between the first row and all the rows of US Presidency dataset and then do that process again for the next row once the outer loop has incremented whilst ultimately assigning these values to the new matrices “hammyMat” and “jaccMat” filling them up column by column since in the inner loop the row index for the matrices “hammyMat” and “jaccMat” stay the same until the condition for the inner while breaks whilst the column index is incrementing in the inner loop. Then a list consisting of the two matrices was created and the function “hamJaccRowMatrix” returns that list whenever called.

Hamming and Jaccard distance calculation’s function Descriptions:

“hammyDist” has two parameters (binA and binB) which in our case will be binary data strings (either columns or rows from the US presidency dataset) of the same length. The hamming distance is calculated by taking the sum of the inequality operator between the 2 binary data strings, the inequality operator will check to see if the value is equal for each binary attribute in the same position in which the two binary attributes differ, we get a value of 1 and sum these values up for each differing binary attribute in a particular position and then the summed value is returned.

“jaccDist” has two parameters (bin1 and bin2) which in our case will be binary data strings (either columns or rows from the US presidency dataset) of the same length. Firstly, we add each pair of binary attributes together from the binary data string and whenever they are equal to 2 (we check if they are equal to 2 using relational operator ==) it means each pair of attributes are both equal to 1 and we take the sum all those resultant true values and assign to variable “inters”. Now we do the same process except we set the relational operator == to 1 to check the number of times in which only one of the binary attributes is 1 and we sum all those resultant true values and then add the previous variable “inters” and assign to variable “join”. Finally, we divide the “join” variable by the “inters” variable and subtract the result of that division from 1 to find the dissimilarity.

b)

These Matrices were written to excel using write.csv() in R if the full files are needed please notify me.

Hamming distance matrix between columns of US Presidency dataset:

A	B	C	D	E	F	G	H	I	J	K	L	M
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Q1	0	9	18	12	22	16	14	18	17	18	16	19
Q2	9	0	17	21	13	15	11	17	22	17	13	18
Q3	18	17	0	10	16	14	14	14	11	8	16	9
Q4	12	21	10	0	24	12	16	18	11	8	16	13
Q5	22	13	16	24	0	20	14	16	21	18	16	17
Q6	16	15	14	12	20	0	20	14	13	10	12	9
Q7	14	11	14	16	14	20	0	18	19	18	10	19
Q8	18	17	14	18	16	14	18	0	11	18	14	19
Q9	17	22	11	11	21	13	19	11	0	15	13	12
Q10	18	17	8	8	18	10	18	18	15	0	14	7
Q11	16	13	16	16	16	12	10	14	13	14	0	15
Q12	19	18	9	13	17	9	19	19	12	7	15	0

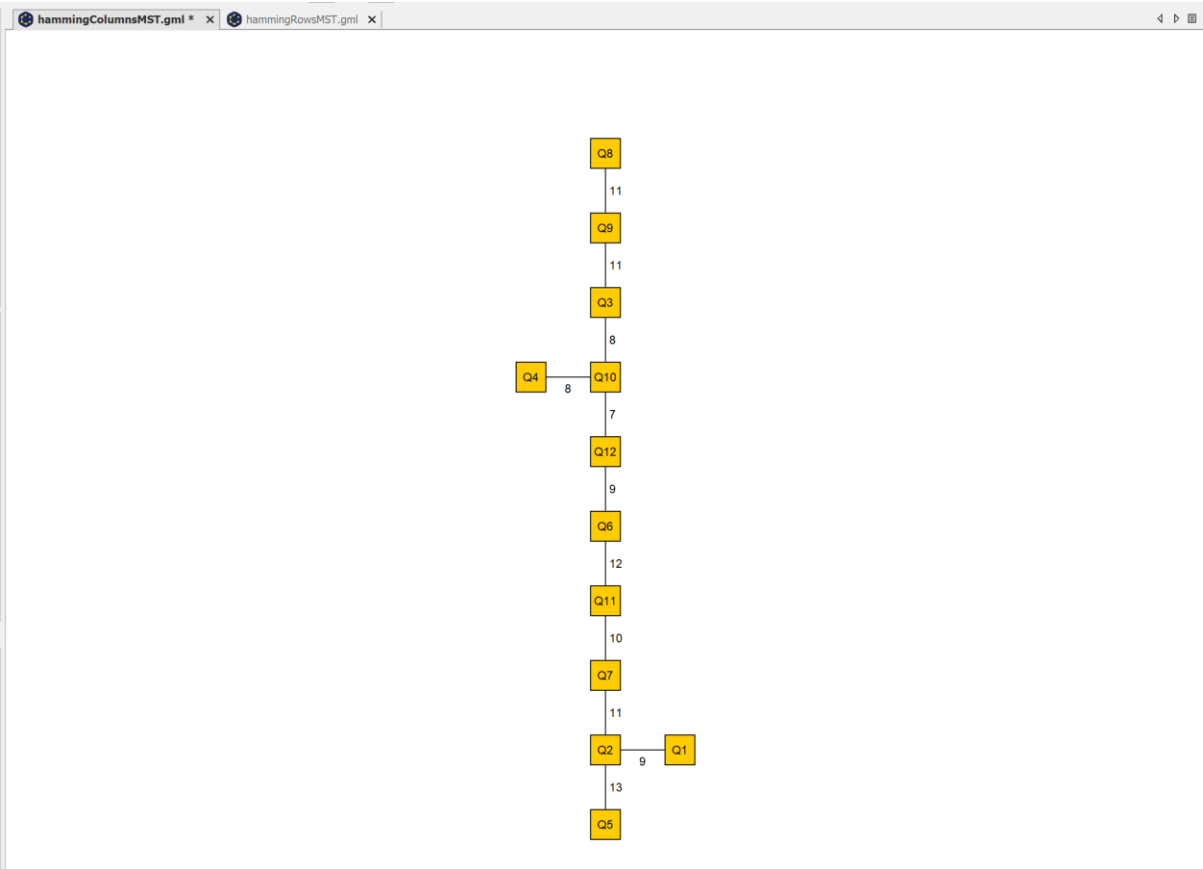
Jaccard distance matrix between columns of US Presidency dataset:

A	B	C	D	E	F	G	H	I	J	K	L	M
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Q1	0	0.391304	0.818182	0.571429	0.733333	0.727273	0.56	0.692308	0.73913	0.857143	0.727273	0.863636
Q2	0.391304	0	0.809524	0.84	0.52	0.714286	0.478261	0.68	0.88	0.85	0.65	0.857143
Q3	0.818182	0.809524	0	0.714286	0.761905	0.933333	0.736842	0.777778	0.785714	0.8	1	0.818182
Q4	0.571429	0.84	0.714286	0	0.888889	0.75	0.727273	0.818182	0.6875	0.666667	0.888889	0.866667
Q5	0.733333	0.52	0.761905	0.888889	0	0.833333	0.56	0.64	0.84	0.857143	0.727273	0.809524
Q6	0.727273	0.714286	0.933333	0.75	0.833333	0	0.869565	0.736842	0.8125	0.833333	0.8	0.75
Q7	0.56	0.478261	0.736842	0.727273	0.56	0.869565	0	0.72	0.826087	0.9	0.555556	0.904762
Q8	0.692308	0.68	0.777778	0.818182	0.64	0.736842	0.72	0	0.611111	0.947368	0.736842	0.95
Q9	0.73913	0.88	0.785714	0.6875	0.84	0.8125	0.826087	0.611111	0	1	0.8125	0.857143
Q10	0.857143	0.85	0.8	0.666667	0.857143	0.833333	0.9	0.947368	1	0	1	0.777778
Q11	0.727273	0.65	1	0.888889	0.727273	0.8	0.555556	0.736842	0.8125	1	0	1
Q12	0.863636	0.857143	0.818182	0.866667	0.809524	0.75	0.904762	0.95	0.857143	0.777778	1	0

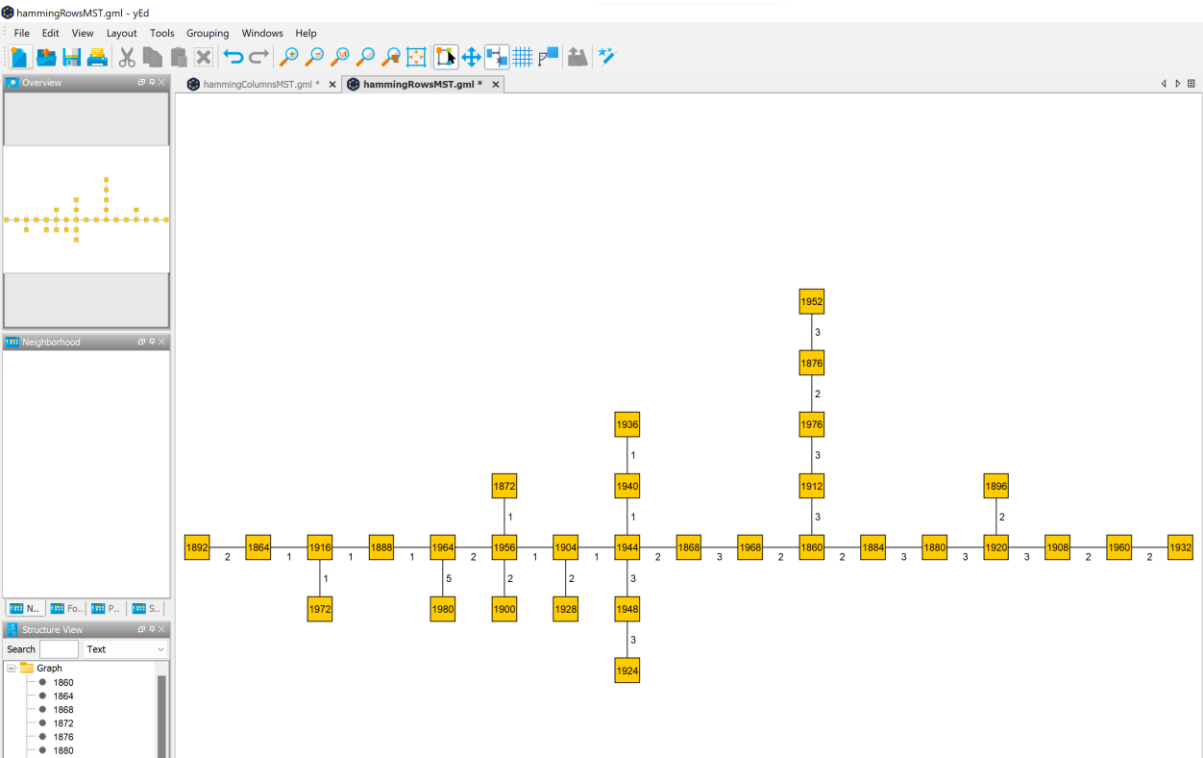
These matrices were produced using the same process as above although when the “hammyDist” and “jaccDist” functions were called in the while loop the arguments were columns (i.e., attributes) of the US presidency dataset instead of rows. The function that returns these matrices is “hamJaccColMatrix”.

c)

Minimum spanning tree constructed from Hamming distance matrix between columns of US Presidency dataset:



Minimum spanning tree from Hamming Distance matrix between all pairs of elections of US Presidency dataset:



Function `mstGraph` returns these graphs. Both Minimum spanning trees were constructed by using the hamming distance matrices for the columns and the rows and using them as an argument in the R function `graph_from_adjacency_matrix()` from the `igraph` package in R;

Value

An `igraph` graph object.

Arguments

<code>adjmatrix</code>	A square adjacency matrix. From <code>igraph</code> version 0.5.1 this can be a sparse matrix created with the <code>'Matrix'</code> package.
<code>mode</code>	Character scalar, specifies how <code>igraph</code> should interpret the supplied matrix. See also the <code>'weighted'</code> argument, the interpretation depends on that too. Possible values are: <code>'directed'</code> , <code>'undirected'</code> , <code>'upper'</code> , <code>'lower'</code> , <code>'max'</code> , <code>'min'</code> , <code>'plus'</code> . See details below.
<code>weighted</code>	This argument specifies whether to create a weighted graph from an adjacency matrix. If it is <code>'NULL'</code> then an unweighted graph is created and the elements of the adjacency matrix gives the number of edges between the vertices. If it is a character constant then for every non-zero matrix entry an edge is created and the value of the entry is added as an edge attribute named by the <code>'weighted'</code> argument. If it is <code>'TRUE'</code> then a weighted graph is created and the name of the edge attribute will be <code>'weight'</code> . See also details below.
<code>diag</code>	Logical scalar, whether to include the diagonal of the matrix in the calculation. If this is <code>'FALSE'</code> then the diagonal is zeroed out first.
<code>add.colnames</code>	Character scalar, whether to add the column names as vertex attributes. If it is <code>'NULL'</code> (the default) then, if present, column names are added as vertex attribute <code>'name'</code> . If <code>'NA'</code> then they will not be added. If a character constant, then it gives the name of the vertex attribute to add.
<code>add.rownames</code>	Character scalar, whether to add the row names as vertex attributes. Possible values the same as the previous argument. By default row names are not added. If <code>'add.rownames'</code> and <code>'add.colnames'</code> specify the same vertex attribute, then the former is ignored.

In this case the arguments occupied were `adjmatrix` which was hamming distance matrices from questions 1a and 1b, `mode` which was `undirected` and `weighted` which was set equal to `true`. This led to producing two weighted graphs with the edge attribute `'weight'` which was used to name the label the edges of the weights on the graph.

The `mst` function from the `igraph` package in R was also used with the only argument being the new graph objects created;

Description

A subgraph of a connected graph is a *minimum spanning tree* if it is tree, and the sum of its edge weights are the minimal among all tree subgraphs of the graph. A minimum spanning forest of a graph is the graph consisting of the minimum spanning trees of its components.

Usage

```
mst(graph, weights = NULL, algorithm = NULL, ...)
```

Arguments

`graph` The graph object to analyze.

`weights` Numeric algorithm giving the weights of the edges in the graph. The order is determined by the edge ids. This is ignored if the `unweighted` algorithm is chosen. Edge weights are interpreted as distances.

`algorithm` The algorithm to use for calculation. `unweighted` can be used for unweighted graphs, and `prim` runs Prim's algorithm for weighted graphs. If this is `NULL` then `igraph` tries to select the algorithm automatically: if the graph has an edge attribute called `weight` or the `weights` argument is not `NULL` then Prim's algorithm is chosen, otherwise the `unweighted` algorithm is performed.

`...` Additional arguments, unused.

This produced Minimum spanning trees for the hamming distance matrix between columns and between rows of the US Presidency data set using Prim's algorithm since the graph has an edge attribute called `weight`.

The `mst` of the graph object was then used as an argument in the function `write_graph()` from the `igraph` package;

Description

`write_graph` is a general function for exporting graphs to foreign file formats, however not many formats are implemented right now.

Usage

```
write_graph(  
  graph,  
  file,  
  format = c("edgelist", "pajek", "ncol", "lgl", "graphml", "dimacs", "gml", "dot",  
    "leda"),  
  ...  
)
```

Arguments

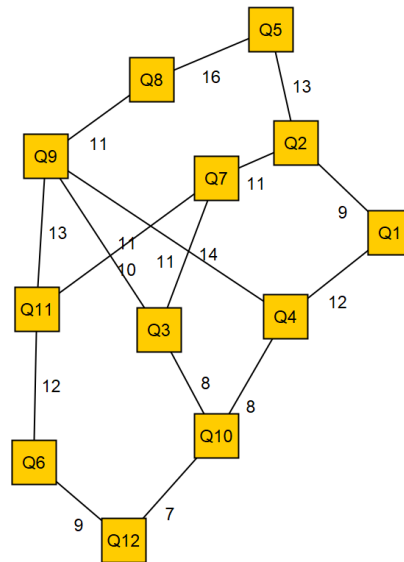
`graph` The graph to export.

`file` A connection or a string giving the file name to write the graph to.

`format` Character string giving the file format. Right now `pajek`, `graphml`, `dot`, `gml`, `edgelist`, `lgl`, `ncol` and `dimacs` are implemented. As of `igraph` 0.4 this argument is case insensitive.

`...` Other, format specific arguments, see below.

Relative Neighbourhood graph for Hamming distance matrix between columns of the US Presidency dataset:



The `rng()` from the package `cccd` in R was used;

`rng {cccd}`

R Documentation

Relative Neighborhood Graph.

Description

the relative neighborhood graph defined by a set of points.

Usage

```
rng(x=NULL, dx=NULL, r = 1, method = NULL, usedeldir = TRUE, open = TRUE, k = NA,  
    algorithm = 'cover_tree')
```

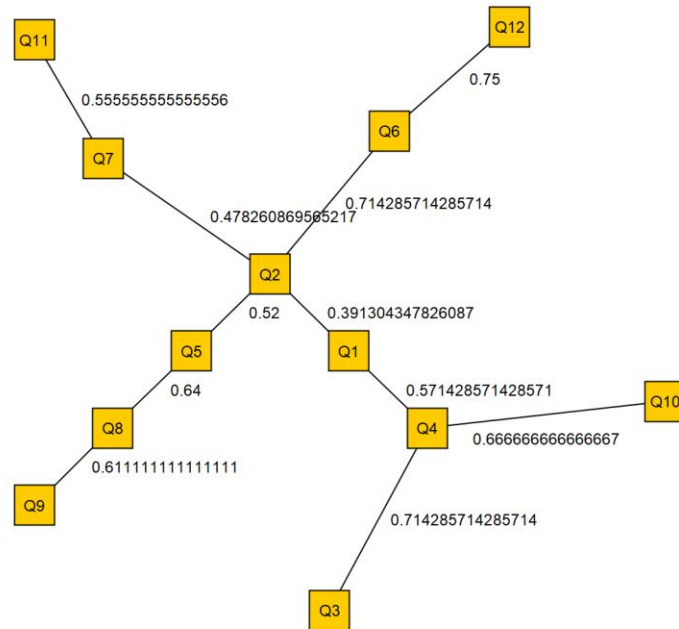
Arguments

- x**
a data matrix. Either `x` or `dx` must be provided.
- dx**
an interpoint distance matrix.
- r**
a multiplier to grow the balls.
- method**
the method used for the distance. See [dist](#)
- usedeldir**
a logical. If true and the data are two dimensional and the `deldir` package is installed, the Delaunay triangularization is first computed, and this is used to compute the relative neighborhood graph.
- open**
logical. If TRUE, open balls are used in the definition.
- k**
If given, `get.knn` is used from FNN to approximate the relative neighborhood graph. Only the `k` nearest neighbors to the points are used to determine whether an edge should be made or not. This will be much faster and use less memory for large data sets, but is an approximation unless `k` is sufficiently large.
- algorithm**
See [get.knn](#).

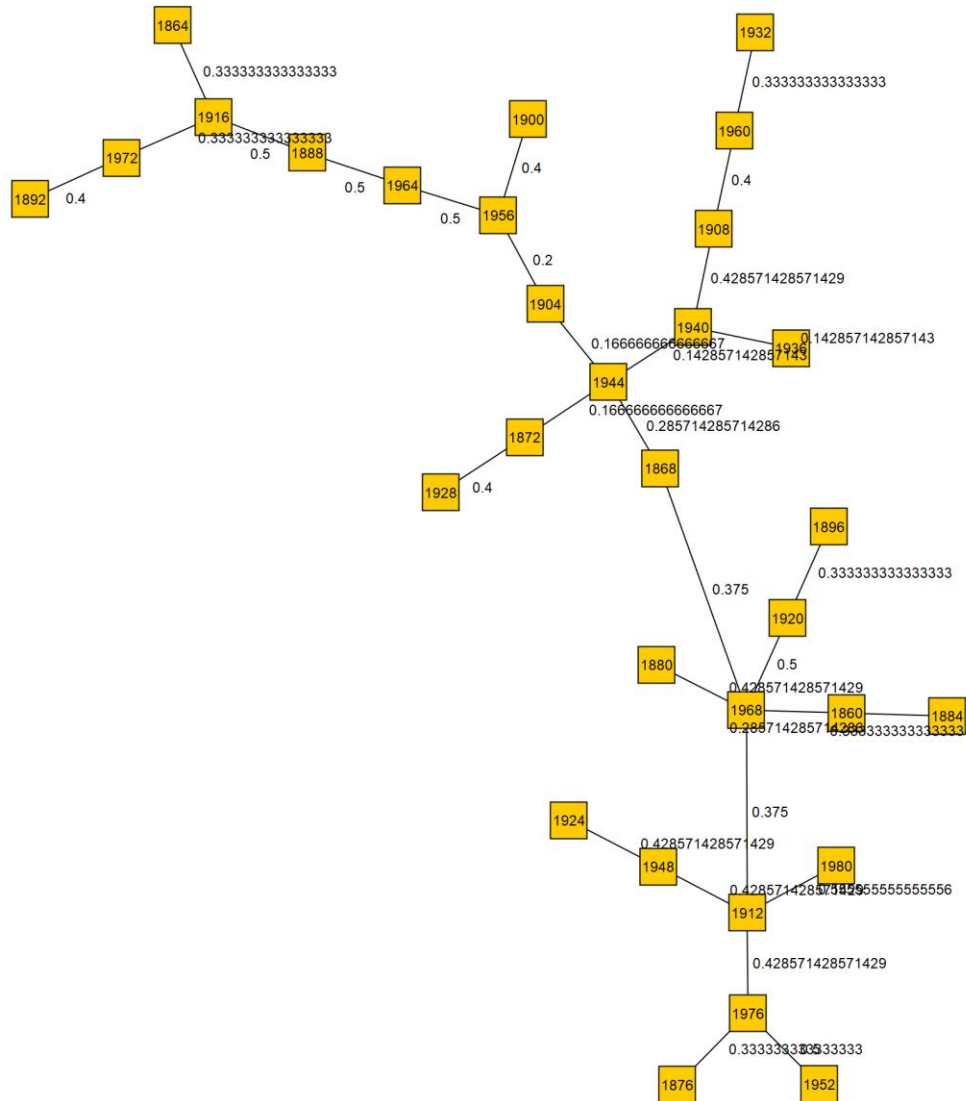
Argument used were the hamming distance matrices from questions 1a and 1b open was set to false `r=1` and `algorithm = cover_tree`. This produces an `igraph` object. Then using the same process as in 1c with `write_graph()` the relative neighbourhood graphs for the hamming distance between the rows of the US presidency dataset and the hamming distance matrix between the columns of the US Presidency dataset produced the graphs in yEd as required.

Exercise 3)

Minimum spanning tree from Jaccard distance matrix between columns of US Presidency dataset:



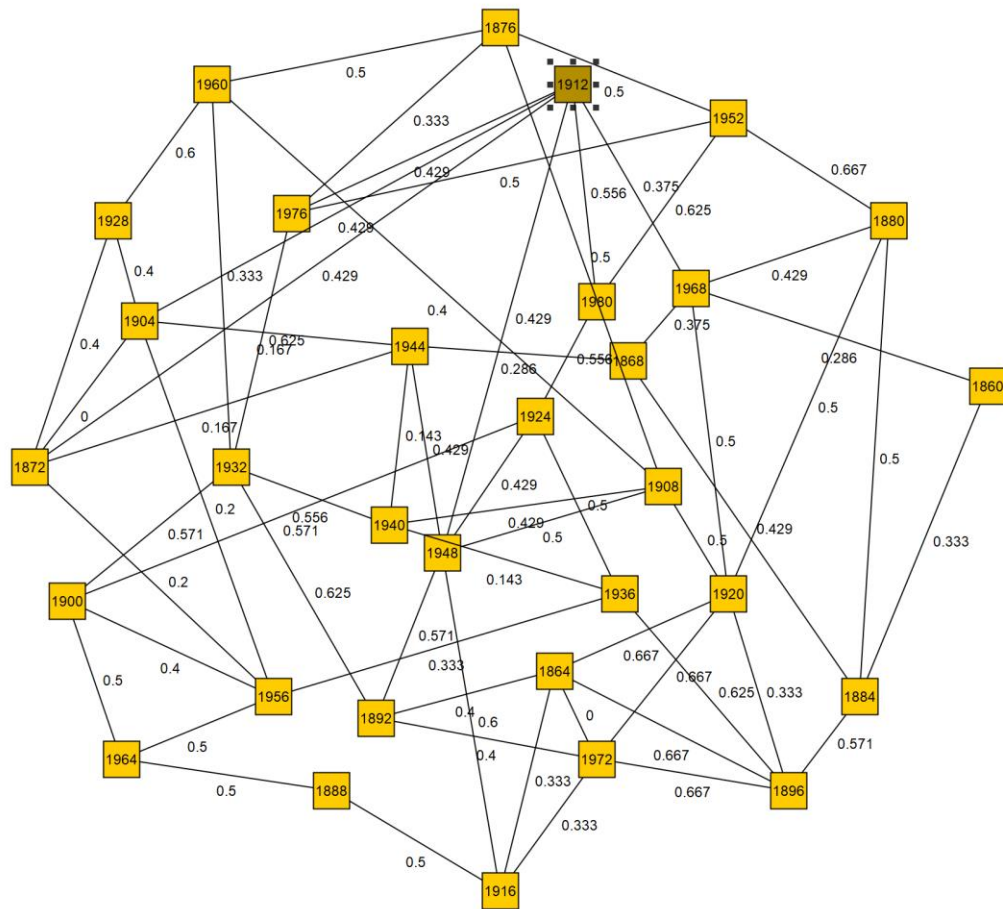
Minimum spanning tree from Jaccard distance matrix between rows of US Presidency dataset:



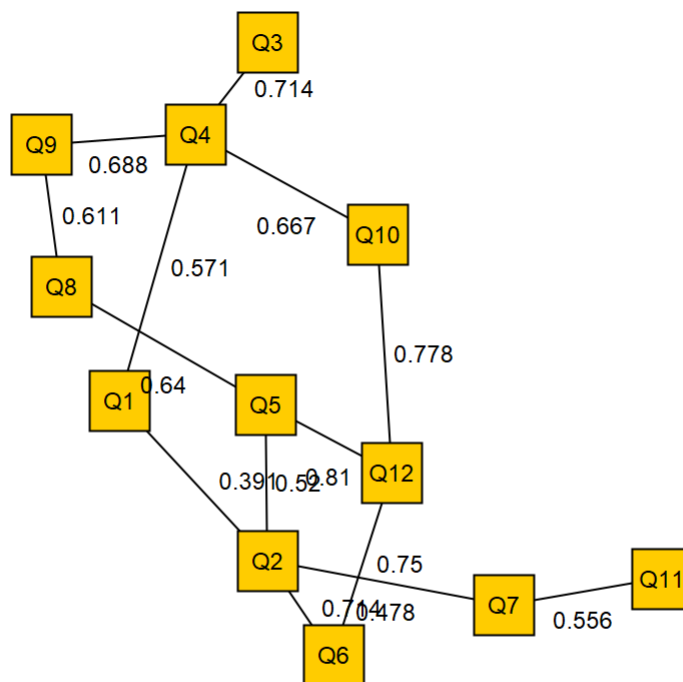
The process used was the same as for q1c except the inputting matrices were the jaccard distance matrices from questions 1a and 1b. The function that returns these graphs is JaccardmstGraph().

Exercise 4:

The Relative neighbourhood graph for the jaccard distance matrix between the rows of the US Presidency dataset:



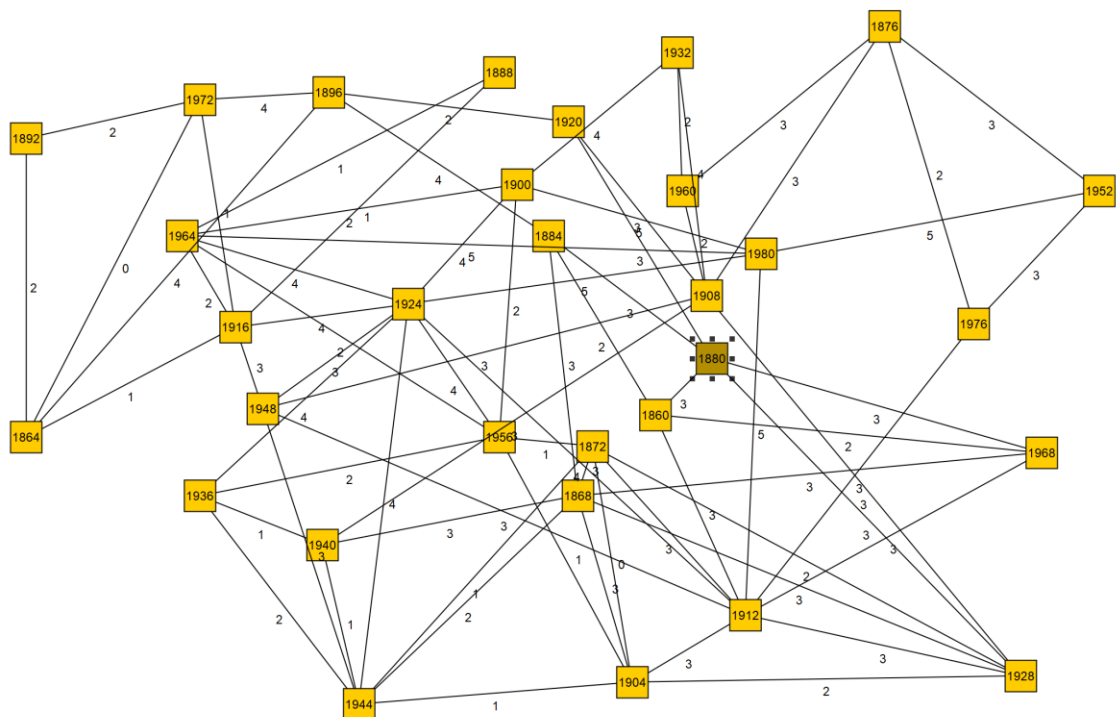
The Relative neighbourhood graph for the Jaccard distance matrix between the columns of the US Presidency dataset:



The same process was used as for exercise 2 except the matrices used are the jaccard distance matrices for the rows and columns of the US Presidency dataset from q1a and q1b. The function that returns these graphs is `jaccardRNG()`.

Exercise 5)

The k-NN with $k=2$ for the Hamming Distance matrix between all pairs of elections of the US Presidency Dataset:



The `nng()` function from the `cccd` package is used in order to produce a knn graph;

`nng {cccd}`

R Documentation

Nearest Neighbor Graphs

Description

nearest neighbor, k-nearest neighbor, and mutual k-nearest neighbor (di)graphs.

Usage

```
nng(x = NULL, dx = NULL, k = 1, mutual = FALSE, method = NULL,
    use.fnn = FALSE, algorithm = 'cover_tree')
```

Arguments

x
a data matrix. Either `x` or `dx` is required

dx
interpoint distance matrix

k
number of neighbors

mutual
logical. if true the neighbors must be mutual. See details.

method
the method used for the distance. See [dist](#)

use.fnn
logical. If TRUE, `get.knn` from the FNN package is used to obtain the neighbors.

algorithm
see [get.knn](#).

Details

a k-nearest neighbor graph is a digraph where each vertex is associated with an observation and there is a directed edge between the vertex and it's k nearest neighbors. A mutual k-nearest neighbor graph is a graph where there is an edge between `x` and `y` if `x` is one of the k nearest neighbors of `y` AND `y` is one of the k nearest neighbors of `x`.

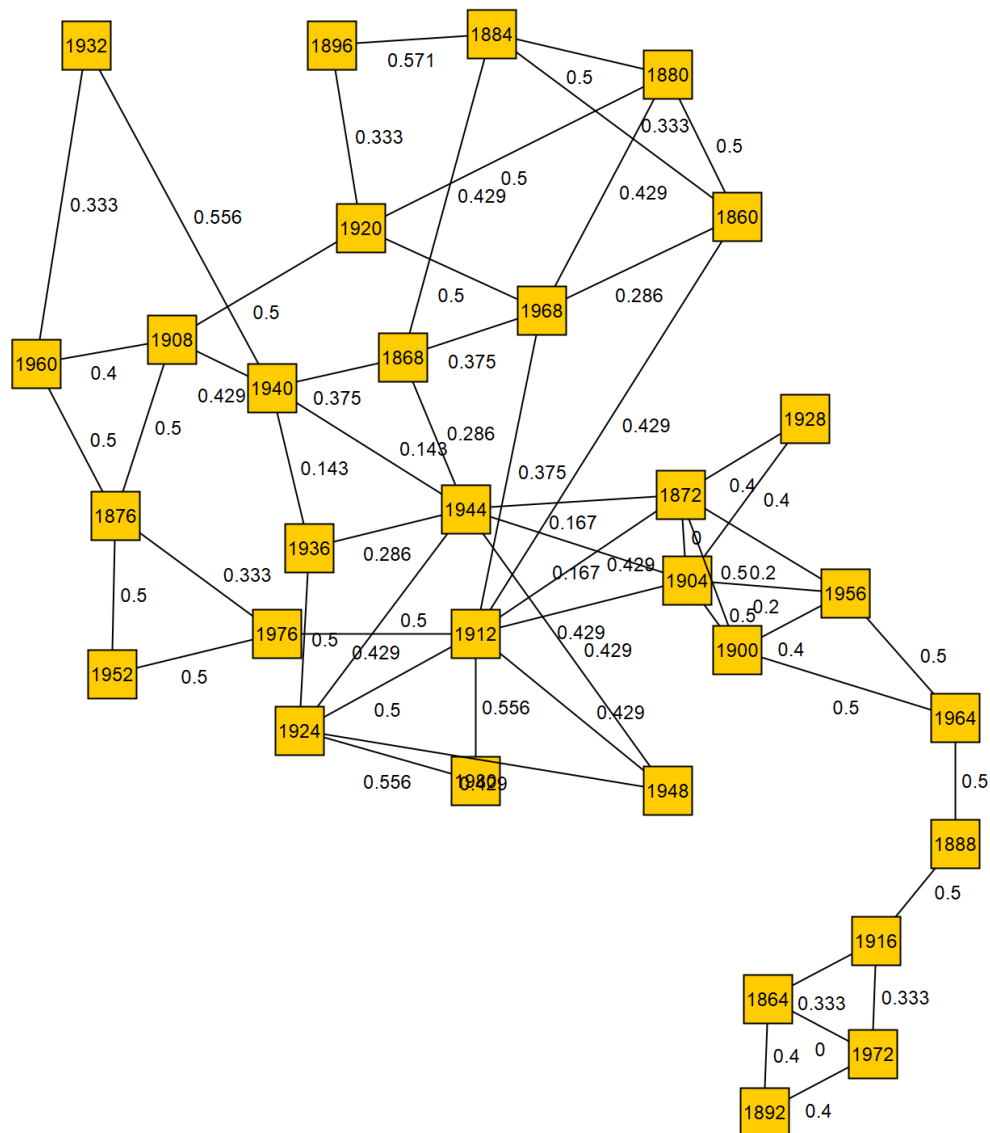
Value

an object of class `igraph` with the extra attributes

the argument `x` is used and is set to the Hamming distance matrix between all pairs of elections of the US Presidency Dataset, `k` is set to 2 as instructed and the algorithm is set = to `cover_tree`. The function that returns the knn graph is `knnHammingRows()`

Exercise 6:

The k-NN with k=2 for the Jaccard Distance matrix between all pairs of elections of the US Presidency Dataset:



The process used for exercise 5 was also used for exercise 6 although the matrix used was the Jaccard Distance matrix between all pairs of elections of the US Presidency Dataset from q1a. The function that returns this graph is `knnJacRows()`.

Exercise 7:

The intersection function from the igraph package was used in both exercise 7 and 8 it returns the edges that are used in both graphs used as arguments within the function:

```
intersection.igraph {igraph}
```

R Documentation

Intersection of graphs

Description

The intersection of two or more graphs are created. The graphs may have identical or overlapping vertex sets.

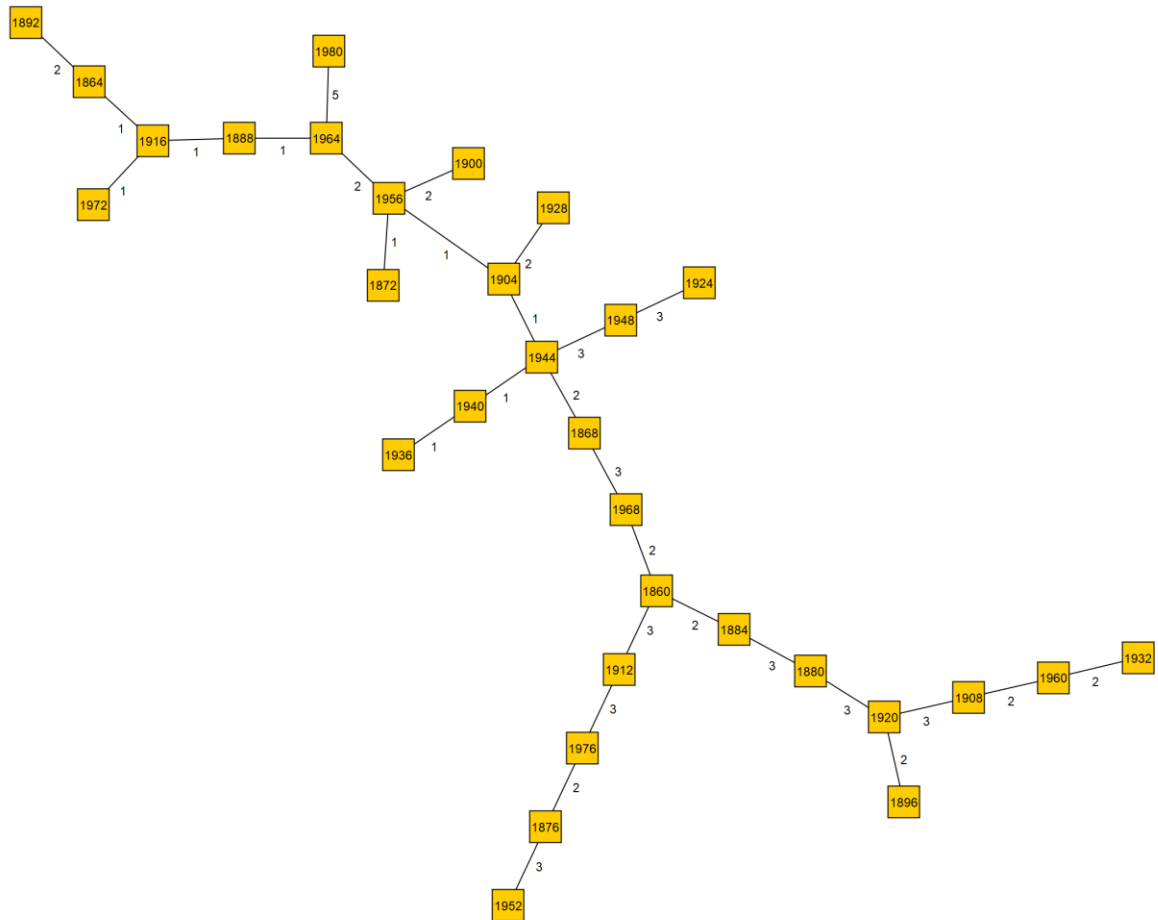
Usage

```
## S3 method for class 'igraph'  
intersection(..., byname = "auto", keep.all.vertices = TRUE)
```

Arguments

...	Graph objects or lists of graph objects.
byname	A logical scalar, or the character scalar auto. Whether to perform the operation based on symbolic vertex names. If it is auto, that means TRUE if all graphs are named and FALSE otherwise. A warning is generated if auto and some (but not all) graphs are named.
keep.all.vertices	Logical scalar, whether to keep vertices that only appear in a subset of the input graphs.

Graph of common edges between the MST for hamming distance between election from question 1c and the knn graph from question 5:

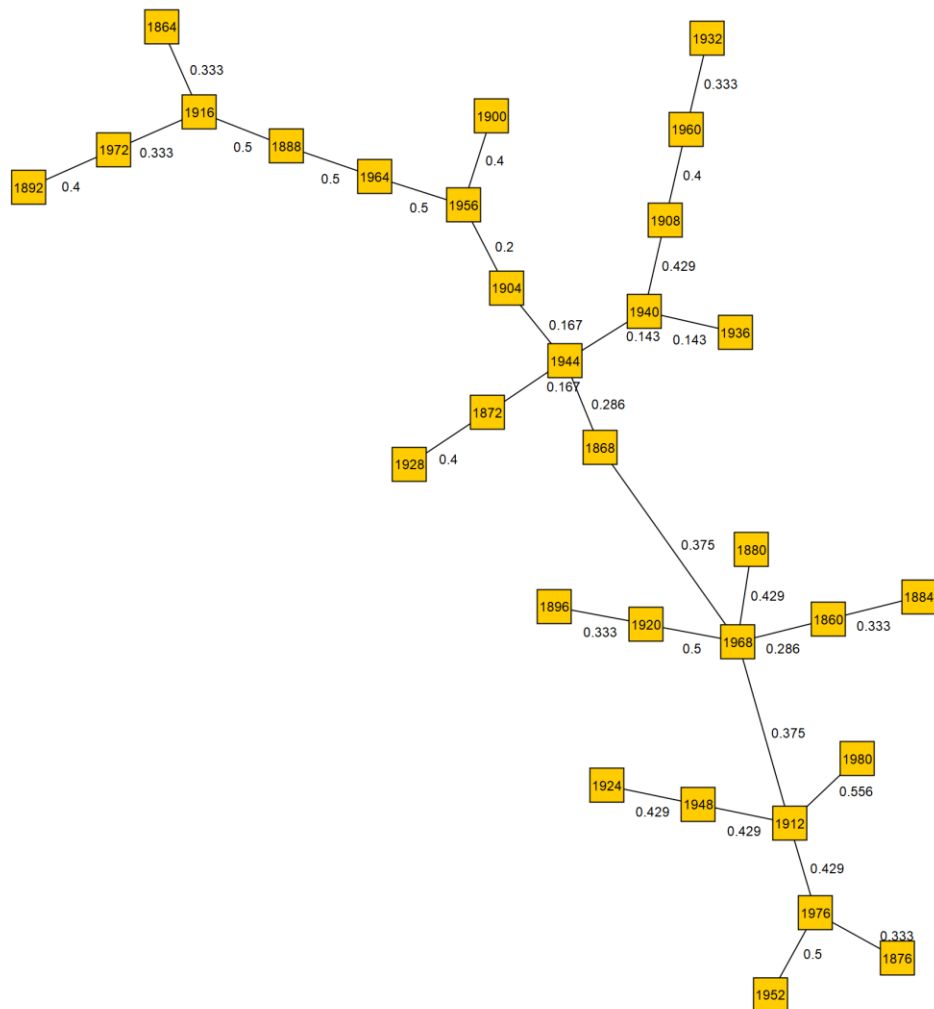


Elections in tabular form grouped by different clusters observed:

Cluster1	Cluster2	Cluster3	Cluster4	Cluster5
1868	1900	1892	1884	1936
1968	1956	1864	1880	1940
1860	1872	1972	1920	1944
1912	1928	1916	1896	1948
1976	1904	1888	1908	1924
1876		1964	1960	
1952		1980	1932	

Exercise 8:

Graph of common edges between the MST for Jaccard distance between elections from exercise 3 and the knn graph from question 6:



Elections in tabular form grouped by the different clusters observed:

Cluster1	Cluster2	Cluster3	Cluster4	
1924	1864	1932	1868	
1948	1892	1960	1896	
1912	1972	1908	1920	
1980	1916	1940	1968	
1976	1888	1936	1880	
1876	1964	1944	1860	
1952	1900	1872	1884	
	1956	1928		
	1904			