

NodeXL for Network analysis

Hands-on at NICAR 2014, Baltimore, Feb 28

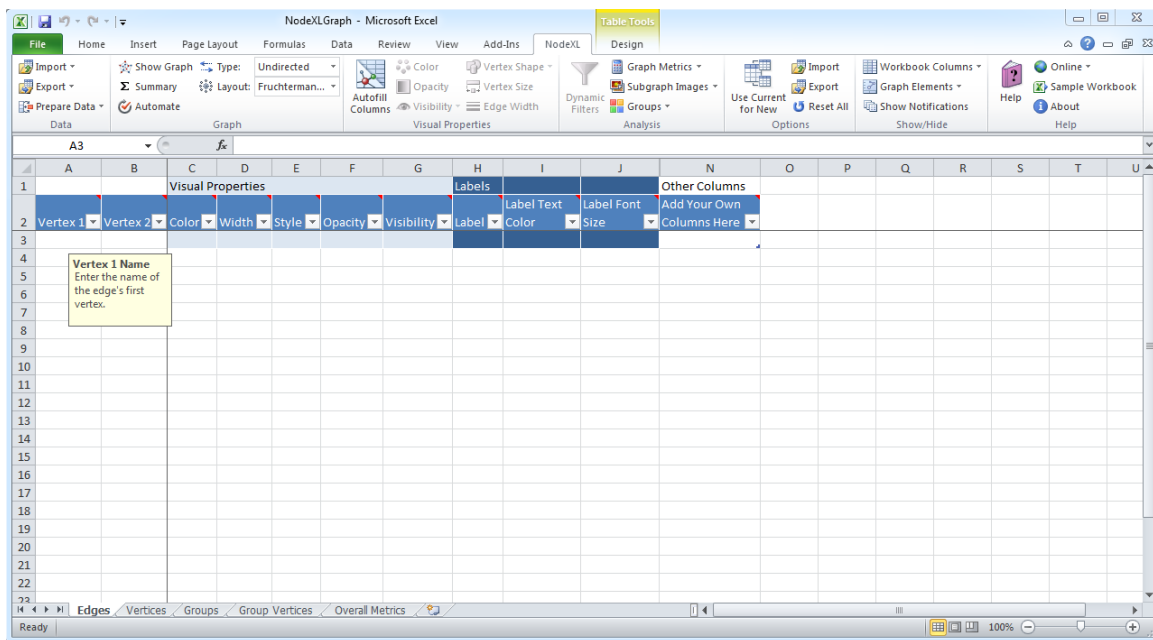
Peter Aldhous

peter@peteraldhous.com

[@paldhous](#)

NodeXL is a template for Microsoft Excel 2007 and later, which allows you to run network analyses in a familiar spreadsheet environment.

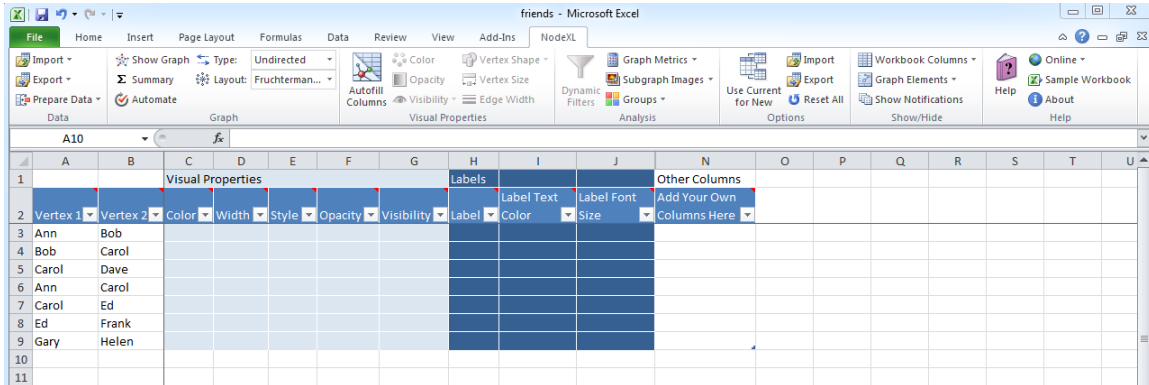
Download the template from [NodeXL site](#), then open:



Notice that NodeXL has its own menu ribbon, and that the first worksheet is called **Edges**.

Network graphs show the connections, or **Edges**, between entities such as people or organizations. These entities are known as **Nodes** or **Vertices**.

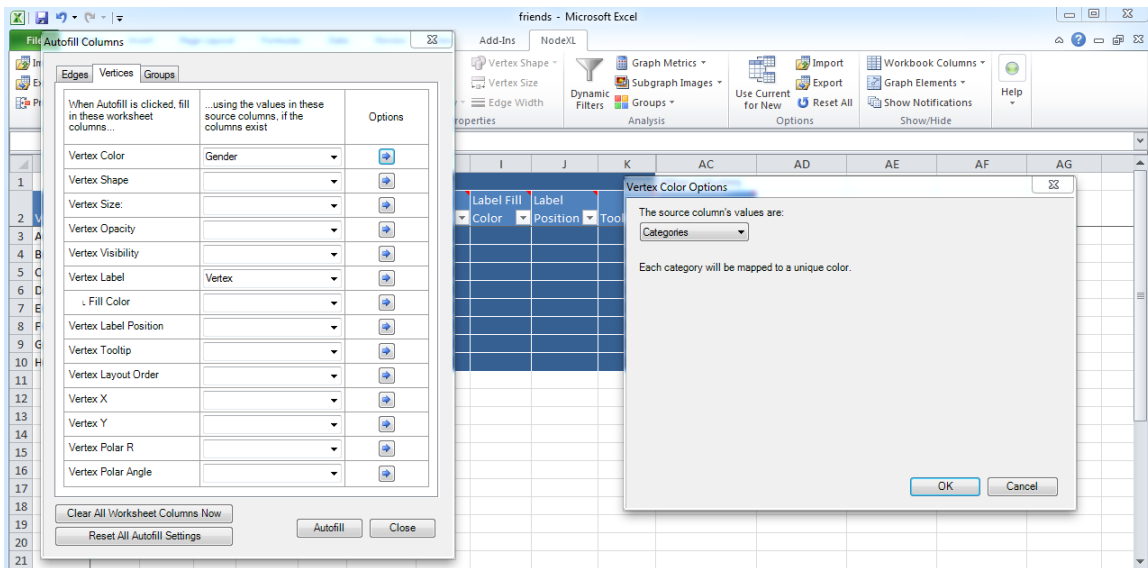
You can enter your own data into NodeXL by typing a list of the edges in the network into this sheet. We'll do that for a series of hypothetical friends. **Save** the template as an Excel Workbook under a new name, and then enter the following data into the **Edges** sheet:



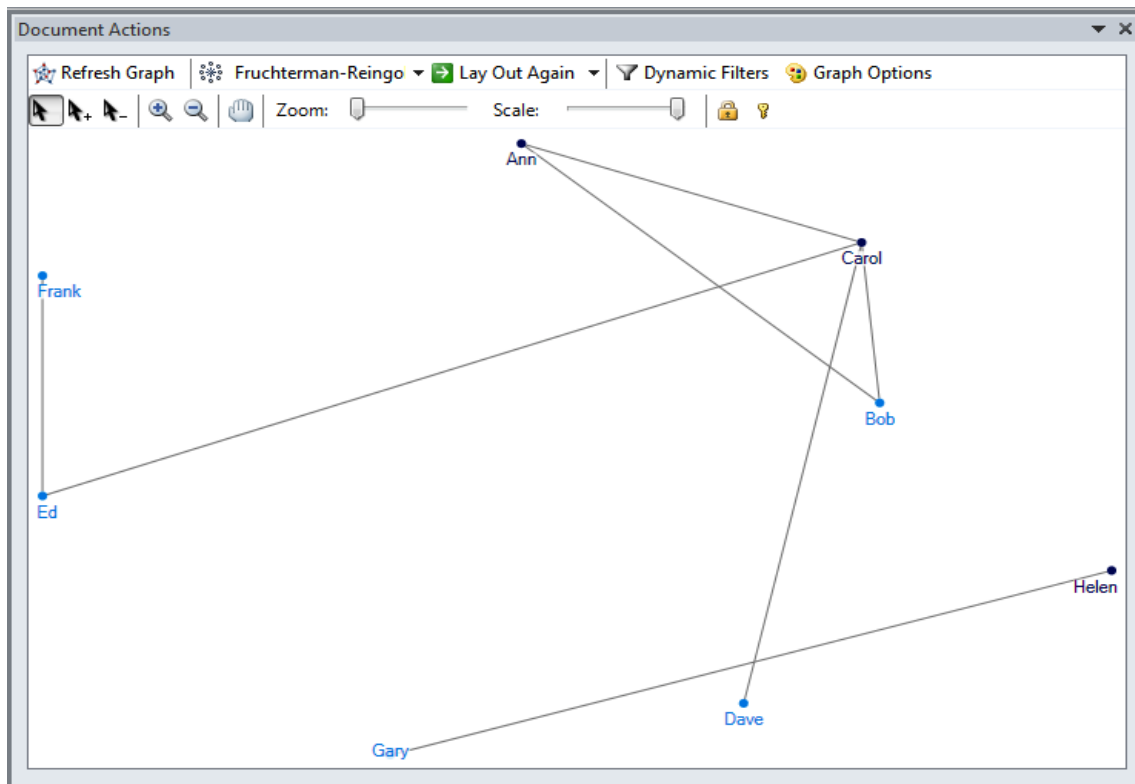
In this case, we're just recording whether the people are friends – a relationship that doesn't have a direction – so we can leave the graph **Type** as **Undirected**.

(If we were recording whether each friend had invited the other to a party, then this should be changed to **Directed**, and we would need a second edge with the names reversed if Ann, for instance, had also invited Carol to a party.)

Click **Autofill Columns**, select the **Vertices** tab, and tell NodeXL to label each friend with Vertex, or their name, and color them by their gender. For the latter, click on the **Options** arrow, tell NodeXL that the values in the column are **Categories** and click **OK**:



Click **Autofill** then **Close**. The Color column will now have populated with RGB color values, and the Label column will contain the friends' names. The graph should have redrawn, but if necessary, click **Refresh Graph**:



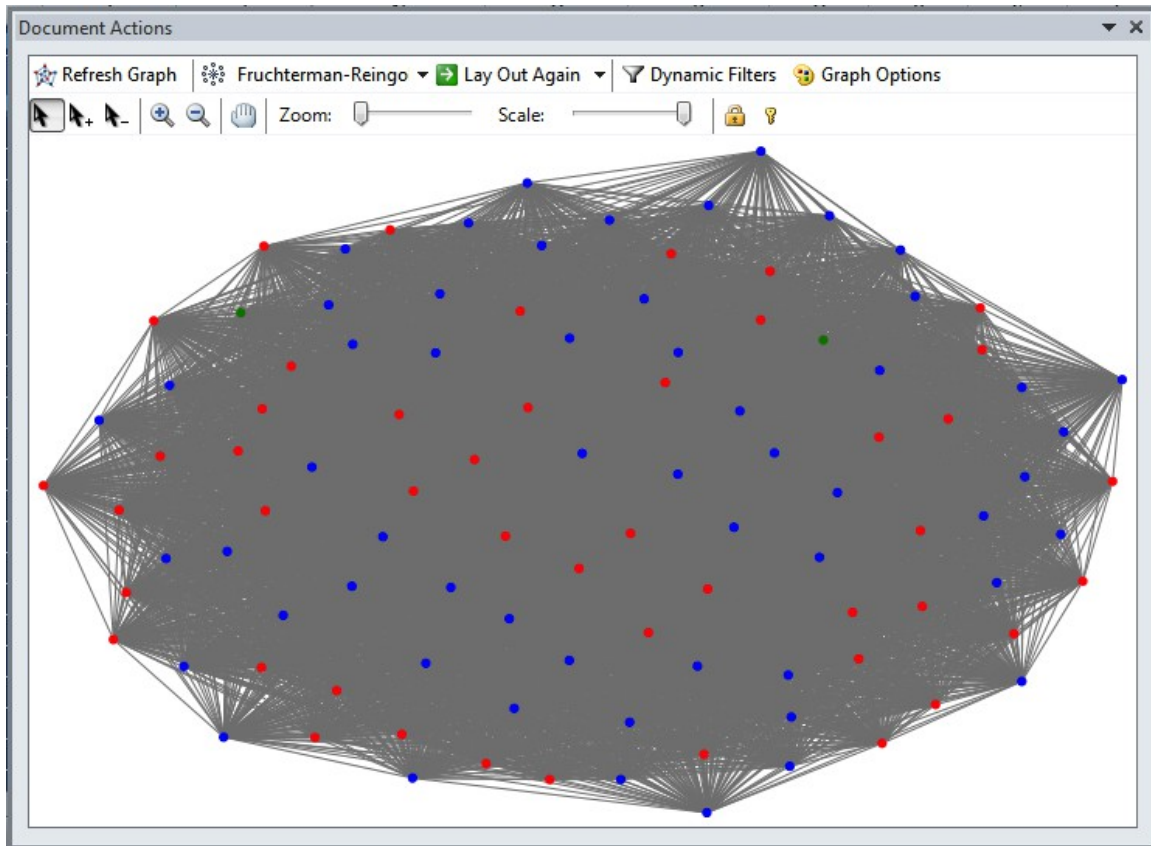
Having learned these basics, we'll now explore a more interesting network, based on voting patterns in the US Senate in 2013. The data can be downloaded [from here](#), and was scraped from [govtrack.us](#) using [this Python script](#). If you wish, you can modify the script to explore Senate voting patterns for other sessions of Congress – just replace the numbers for **congress** and **year** with values from [this list](#).

The data is in a format called [graphml](#). In a blank NodeXL template, select **Import>From Graphml file**, and open the file. (Click **Yes** if asked about turning off text wrapping.) Once the file has imported, save as an Excel workbook under a new name.

The **Edges** worksheet gives a list of pairs of Senators, with information on how they voted, including a column called **percent_agree**, which is the number of times the pair voted the same way, divided by the total number of votes in the session:

	Vertex 1	Vertex 2	Color	Width	Style	Opacity	Visibility	Label	Label Text	Label Color	Label Font Size	percent_agree	votes_agree
3	Toomey (R-Cardin (D-MD))	Johnson (D-Cardin (D-MD))										0.274914089	80
4	Johnson (D-Cardin (D-MD))	Begich (D-A-Cardin (D-MD))										0.958762887	279
5	Begich (D-A-Cardin (D-MD))	Wyden (D-C-Cardin (D-MD))										0.886597938	258
6	Wyden (D-C-Cardin (D-MD))	Udall (D-CC-Cardin (D-MD))										0.948453608	276
7	Udall (D-CC-Cardin (D-MD))	Hagan (D-N-Cardin (D-MD))										0.927835052	270
8	Hagan (D-N-Cardin (D-MD))	Merkley (D-Cardin (D-MD))										0.890034364	259
9	Merkley (D-Cardin (D-MD))	Thune (R-SI-Cardin (D-MD))										0.931271478	271
10	Thune (R-SI-Cardin (D-MD))	Lautenberg Cardin (D-MD)										0.312714777	91
11	Lautenberg Cardin (D-MD)	Hatch (R-UT-Cardin (D-MD))										0.109965636	32
12	Hatch (R-UT-Cardin (D-MD))	Warner (D-Cardin (D-MD))										0.384879725	112
13	Warner (D-Cardin (D-MD))	Heller (R-N-Cardin (D-MD))										0.910652921	265
14	Heller (R-N-Cardin (D-MD))	Cochran (R-Cardin (D-MD))										0.288659794	84
15	Cochran (R-Cardin (D-MD))	Tester (D-M-Cardin (D-MD))										0.371134021	108
16	Tester (D-M-Cardin (D-MD))	Menendez Cardin (D-MD)										0.903780069	263
17	Menendez Cardin (D-MD)	Warren (D-I-Cardin (D-MD))										0.969072165	282
18	Warren (D-I-Cardin (D-MD))	Durbin (D-II-Cardin (D-MD))										0.948453608	276
19	Durbin (D-II-Cardin (D-MD))	Sanders (I-I-Cardin (D-MD))										0.972508591	283
20	Sanders (I-I-Cardin (D-MD))	Donnelly (I-C-Cardin (D-MD))										0.93814433	273
21	Donnelly (I-C-Cardin (D-MD))											0.883161512	257

Click **Show Graph** to see the following, in which each of the Senators is connected to all of the others, because every pair voted the same way at least once:



NodeXL's strength is the ease with which you can now filter and customize the network visualization.

The first task with complex networks like this one is often to filter them to reveal their core structure. This can be done in two ways. Clicking **Dynamic Filters** brings up a series of sliders that you can use to adjust the visibility of edges and nodes in the network graph. See how some of the edges disappear as you move the left slider for **percent_agree** toward the right.

But this does not make any changes to the network that is being analyzed. Dynamic filters will not, for instance, cause any change in the results obtained by calculating metrics describing the network.

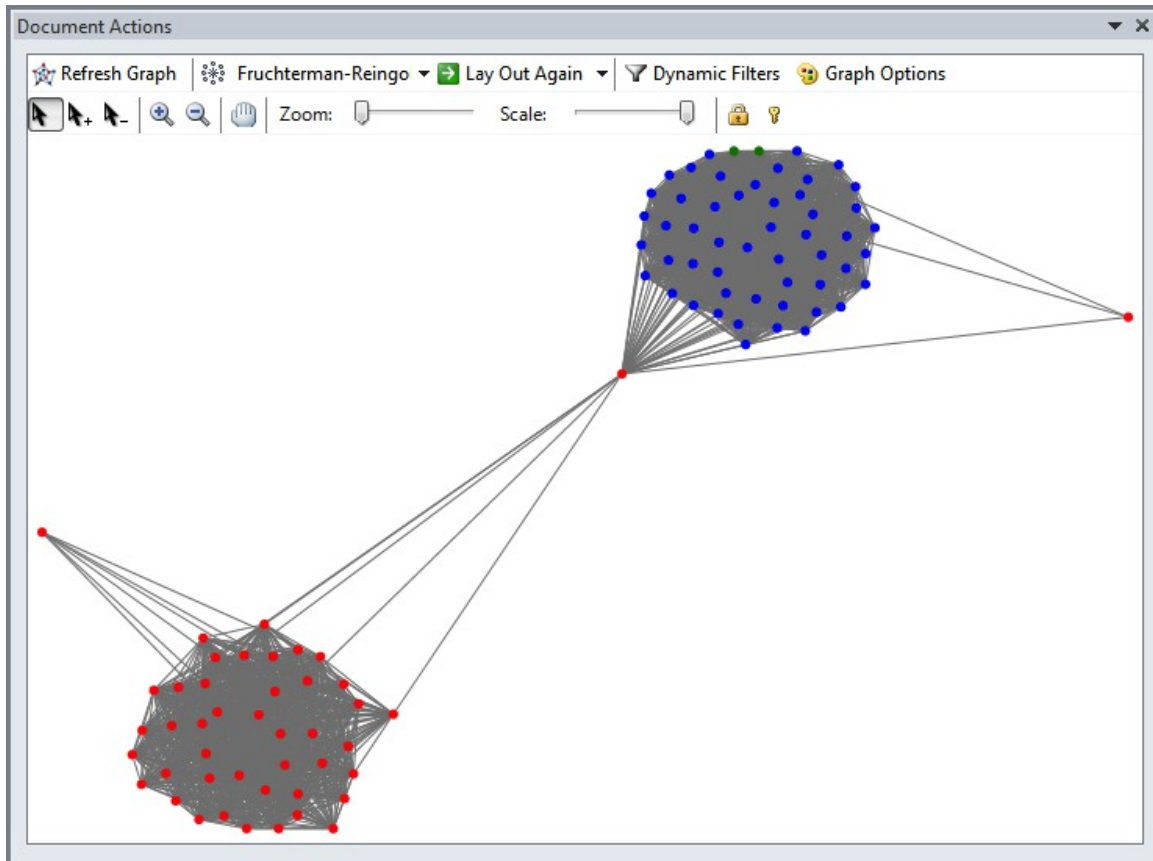
Instead, we are going to filter the network using **Autofill Columns**, allowing us to run subsequent analyses on this filtered view of the data.

Click **Autofill Columns** and select the **Edges** tab. We will filter so Senators are connected only if they voted the same way at least two-thirds of the time. Select **Edge Visibility = percent_agree**, fill in **Options** as follows, and click **OK**:

The screenshot shows the Microsoft Excel interface with the 'Autofill Columns' dialog box open. The 'Edges' tab is selected, and the 'Edge Visibility' is set to 'percent_agree'. The 'Edge Visibility Options' dialog box is also open, showing a filter set to 'Greater than or equal to 0.67'. The background shows a spreadsheet with columns for 'percent_agree' and 'votes_agree'.

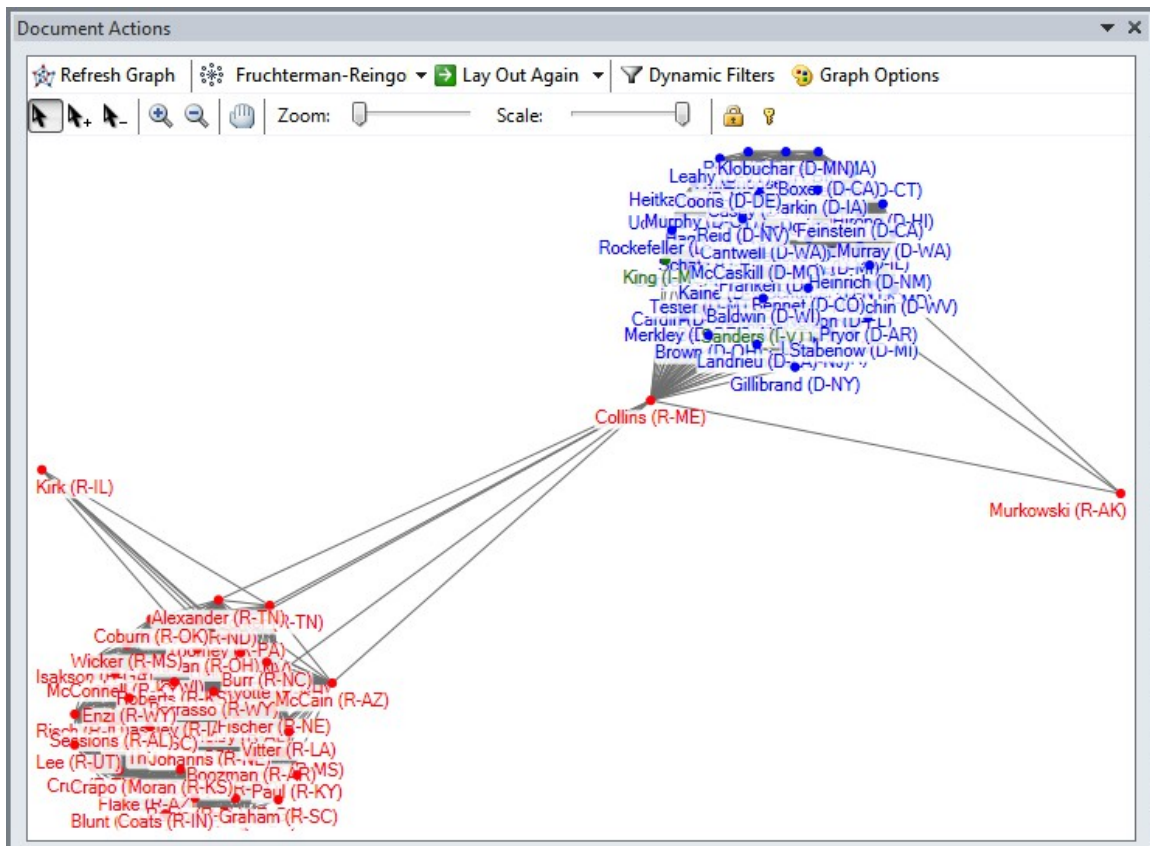
Senator	percent_agree	votes_agree
0.274914089	80	
0.958762887	279	
0.886597938	258	
0.948453608	276	
0.927835052	270	
0.890034364	259	
0.931271478	271	
0.312714777	91	
0.109965636	32	
0.384879725	112	
0.910652921	265	
0.288659794	84	
0.371134021	108	
0.903780069	263	
0.969072165	282	
0.948453608	276	
0.972508591	283	
0.938144433	273	
0.883161512	257	
0.95532646	278	
0.962199313	280	
0.233676976	68	

Then click **Autofill**, and the graph should redraw. Make sure the Layout is set to **Fruchterman-Reingold**, which is the automatic layout algorithm that works best with this data, and click **Lay Out Again** several times until you have two clear clusters:



As you might expect, these clusters corresponds to Democrats and Republicans, but three Republican Senators stand apart from the rest. Who are they? To find out, we'll again use **Autofill Columns**.

On the **Vertices** tab, select **Vertex Label = Vertex** to label each Senator with their name. Click **Autofill** and **Close**, then **Lay Out Again** until you have something like the following:



The most interesting Senators in the network are two Republicans who have stronger connections with Democrats than with members of their own party: Susan Collins of Maine and Lisa Murkowski of Alaska.

Collins, in particular, provides the only connection between the two clusters. We can calculate some network metrics and customize the graph to highlight her role in network.

NodeXL can calculate common metrics used to describe networks, including the following:

Degree is a simple count of the number of connections for each node. For directed networks, it is divided into **In-degree**, for the number of incoming connections, and **Out-degree**, for outgoing connections.

Eigenvector centrality accounts not only for the node's own degree, the also the degrees of the nodes to which it connects.

Betweenness centrality essentially reveals how important each node is in providing a "bridge" between different parts of the network. It highlights the nodes that, if removed,

would cause a network to fall apart.

Closeness centrality is a measure of how close each node is, on average, to all of the other nodes in a network. It highlights the nodes that connect to the others through a lower number of edges – think [Kevin Bacon Game](#).

For our purposes, betweenness centrality is an interesting measure, as it should help highlight the most bipartisan Senators.

Select **Graph Metrics**, check **Vertex betweenness and closeness centralities** and click **Calculate Metrics**:

The screenshot shows the Microsoft Excel interface with the 'Graph Metrics' dialog box open. The dialog box has a list of metrics to calculate, with 'Overall graph metrics', 'Vertex betweenness and closeness centralities', and 'Vertex PageRank' checked. Below the list, a table shows the following overall metrics:

Graph Type	Directed or undirected.
Vertices	The number of vertices in the graph.
Unique Edges	The number of edges that do not have duplicates.
Edges With	The number of edges that have duplicates.

At the bottom of the dialog box, there are buttons for 'Calculate Metrics' and 'Cancel'. The background shows an Excel spreadsheet with columns for 'Graph Metrics', 'Other Columns', 'percent_agree', and 'votes_agree'. The data in the spreadsheet is as follows:

Graph Metrics	Other Columns	percent_agree	votes_agree
		0.274914089	80
		0.958762887	279
		0.886597938	258
		0.948453608	276
		0.927835052	270
		0.890034364	259
		0.931271478	271
		0.312714777	91
		0.109965636	32
		0.384879725	112
		0.910652921	265
		0.288659794	84
		0.371134021	108
		0.903780069	263
		0.969072165	282
		0.948453608	276
		0.972508591	283
		0.93814433	273
		0.883161512	257
		0.95532646	278
		0.962199313	280

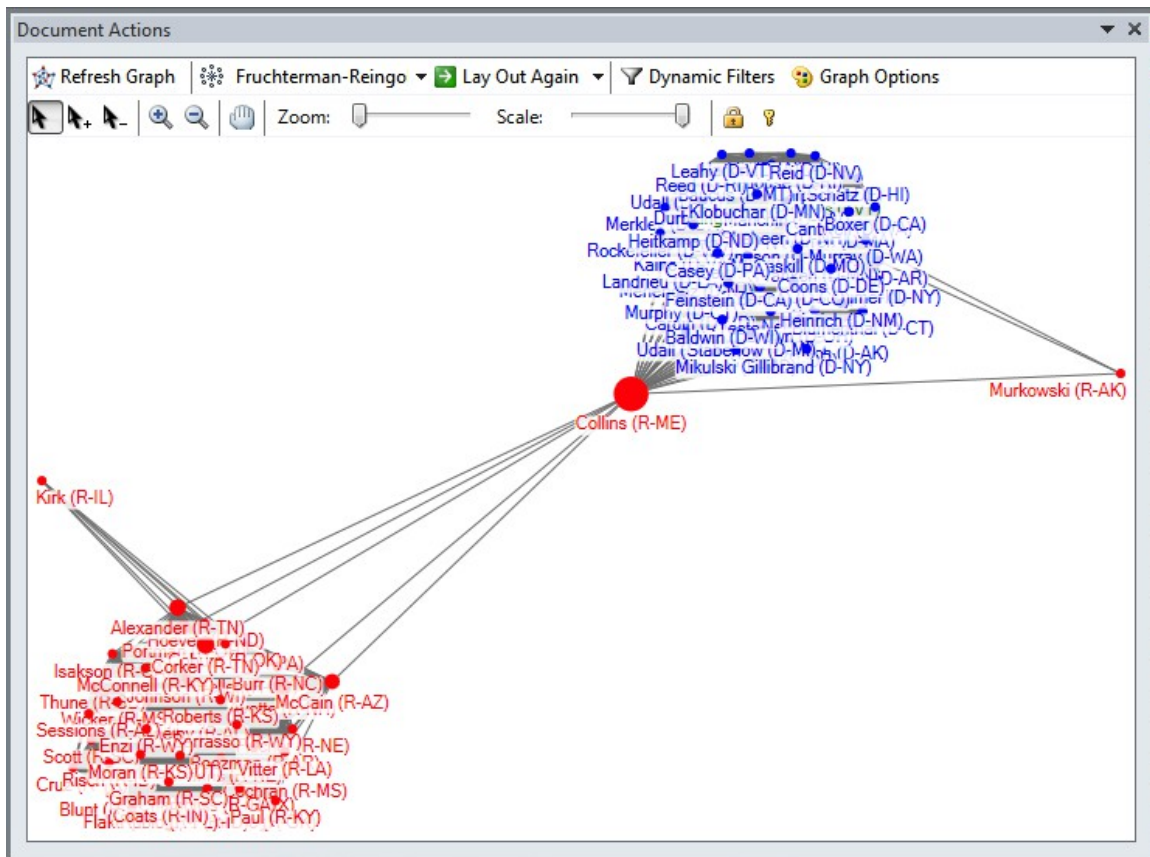
The relevant columns in the Vertices sheet will have been populated with data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
1	Visual Properties							Graph Metrics																
2	Vertex	Color	Shape	Size	Opacity	Image	Visibility	Label	Label Fill	Label Color	Label Position	Label Tooltip	In-Degree	Out-Degree	Betweenness Centrality	Closeness Centrality	Eigenvector Centrality							
3	Toomey (Fred)							Toomey (R-PA)							6.946	0.004								
4	Cardin (D-blue)							Cardin (D-MD)							22.629	0.006								
5	Johnson (blue)							Johnson (D-SD)							22.629	0.006								
6	Begich (D-blue)							Begich (D-AK)							22.629	0.006								
7	Wyden (D-blue)							Wyden (D-OR)							0.000	0.004								
8	Udall (D-blue)							Udall (D-CO)							22.629	0.006								
9	Hagan (D-blue)							Hagan (D-NC)							22.629	0.006								
10	Merkley (blue)							Merkley (D-OR)							0.000	0.004								
11	Thune (R-red)							Thune (R-SD)							0.912	0.004								
12	Lautenberg (blue)							Lautenberg (D-NJ)																
13	Hatch (R-red)							Hatch (R-UT)							0.564	0.004								
14	Warner (D-blue)							Warner (D-VA)							22.629	0.006								
15	Heller (R-red)							Heller (R-NV)							6.946	0.004								
16	Cochran (Fred)							Cochran (R-MS)							0.912	0.004								
17	Tester (D-blue)							Tester (D-MT)							22.629	0.006								
18	Menendez (blue)							Menendez (D-NJ)							0.000	0.004								
19	Warren (D-blue)							Warren (D-MA)							0.000	0.004								
20	Durbin (D-blue)							Durbin (D-IL)							22.629	0.006								
21	Sanders (I-green)							Sanders (I-VT)							0.000	0.004								
22	Donnelly (blue)							Donnelly (D-IN)							22.629	0.006								
23	Nelson (D-blue)							Nelson (D-FL)							22.629	0.006								

Now we can size the Senators according to their betweenness centrality. Click **Autofill Columns**, select the **Vertices** tab, and set **Vertex Size = Betweenness Centrality**. Using **Options**, set the maximum size to **20**, and click **OK**:

The screenshot shows the 'Autofill Columns' dialog box with the 'Vertices' tab selected. The 'Vertex Size' is set to 'Betweenness Centrality'. The 'Vertex Size Options' dialog box is also open, showing the mapping of the source column number to the vertex size. The 'Map this source column number' is set to 'The largest number in the column', and the 'To this vertex size' is set to 20.0.

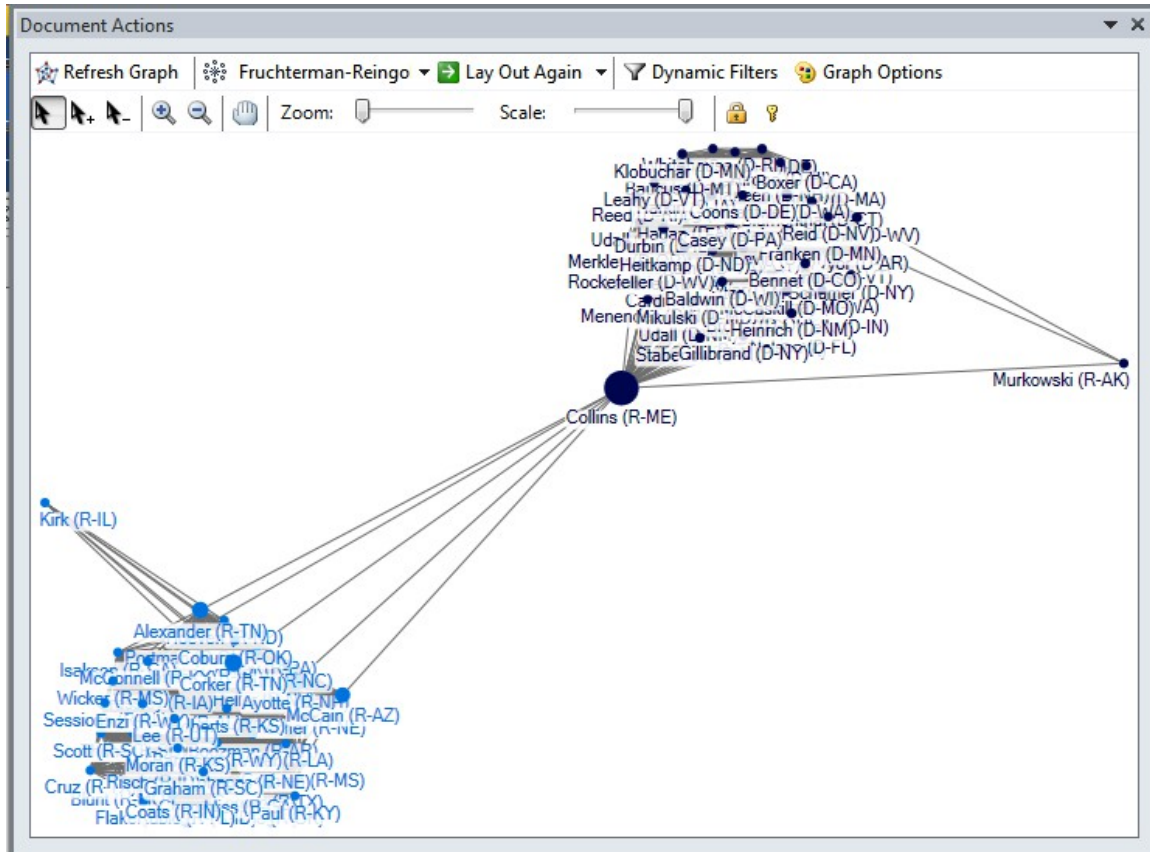
Click **Autofill** and **Close**, then **Lay Out Again** until you have something like the following. (You may need to experiment with maximum size of the Vertices to get a sensible display):



Now is a good time to save the image. **Right click** on the graph, and select **Save Image to File>Image Options** to customize the size. **Right click** again, and then select **Save Image to File>Save Image** to save in formats including PNG and JPEG.

We've colored the Senators by their party affiliation, but NodeXL can also use clustering algorithms to detect clusters of vertices with similar patterns of connections.

Select **Groups>Group by Cluster** and select the **Clusset-Newman-Moore** clustering algorithm. Click **Refresh Graph** and then **Lay Out Again** until you have something like the following:



Collins and Murkowski are Republicans, but on the basis of their patterns of voting in 2013, the clustering algorithm grouped them with the Democrats.

To learn more about how to use NodeXL, refer to the book [Analyzing Social Media Networks With NodeXL](#).