Orange3-Network Documentation

Biolab

Mar 11, 2022

Contents

CHAPTER 1

Widgets

1.1 Network File

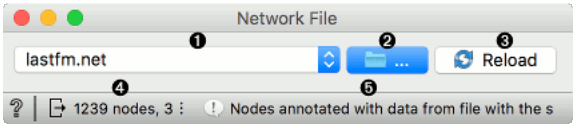
Read network graph file in Pajek format.

Outputs

- Network: An instance of Network Graph.
- Items: Properties of a network file.

Network File widget reads network files and sends the input data to its output channel. History of the most recently opened files in maintained in the widget. The widget also includes a directory with sample data sets that come pre-installed with the add-on.

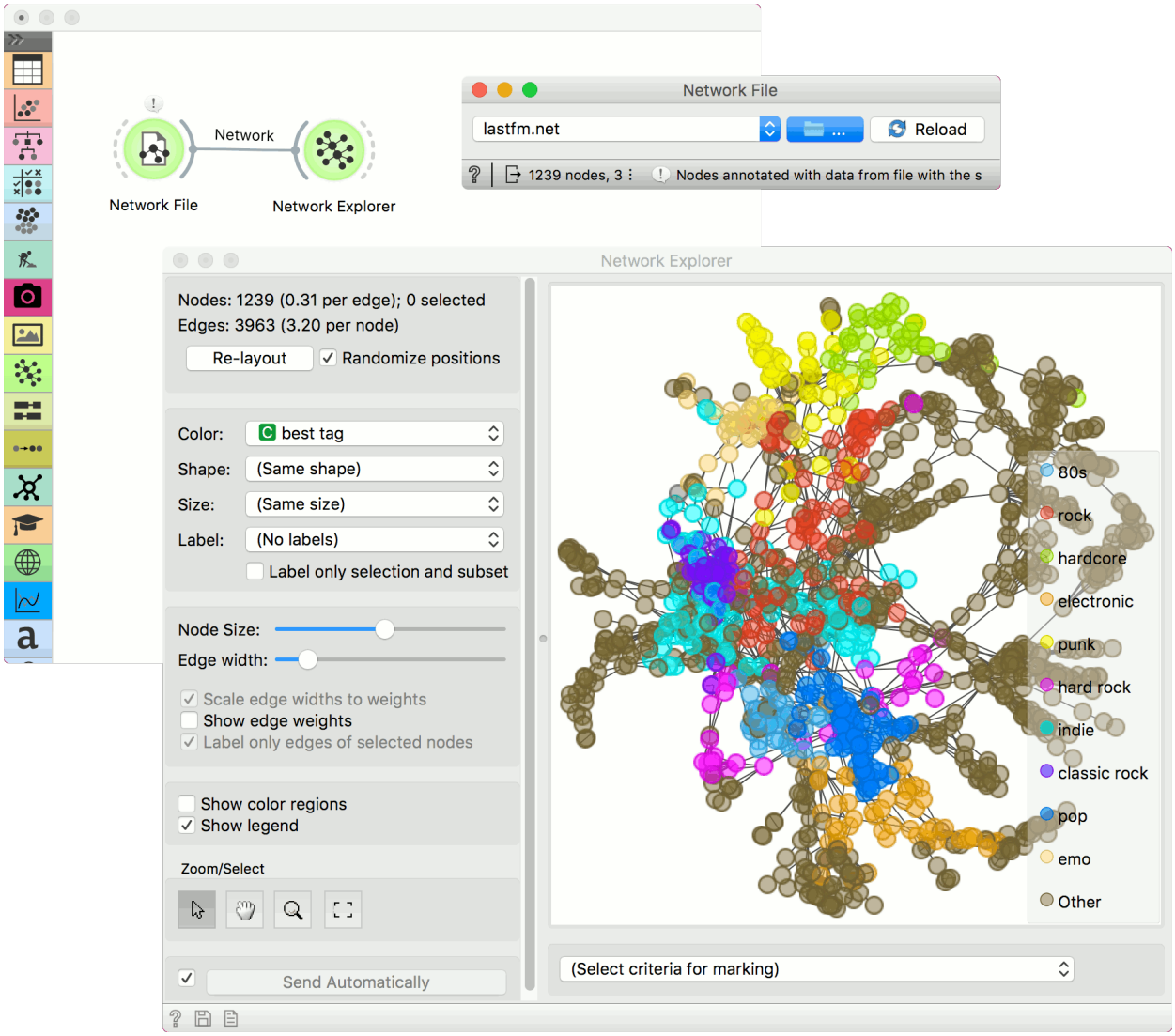
The widget reads data in .net and .pajek formats. A complimentary .tab, .tsv or .csv data set can be provided for node information. Orange by default matches a file with the same name as .net file.



1. Load network file. The widget construct a data table from the data whose filename matches the graph filename (i.e. *lastfm.net* and *lastfm.tab*) or, if no match is found, from the graph. A dropdown menu provides access to documentation data sets with *Browse documentation networks...*.
2. The folder icon provides access to local data files.
3. Reload the data file from 1.
4. Status bar reports on the number of nodes and edges and the type of the graph.
5. Information, warnings and errors. Hover over a message to read it all.

1.1.1 Example

We loaded the *lastfm.net* from documentation data set (dropdown → Browse documentation networks). The nicest way to observe network data is with the Network Explorer widget. **Network File** widget automatically matched the corresponding data file (*lastfm.net* with *lastfm.tab*), so node attributes are available in the widget.



1.2 Network Explorer

Visually explore the network and its properties.

Inputs

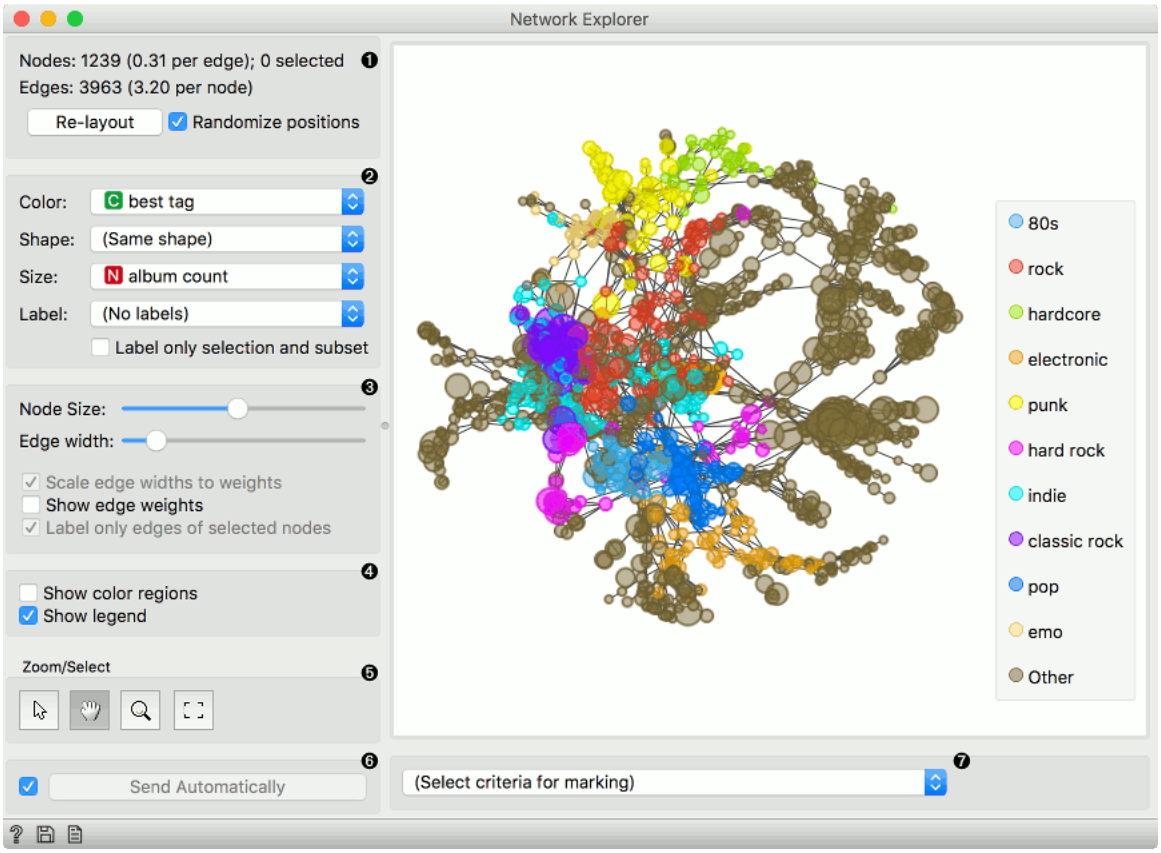
- Network: An instance of Network Graph.
- Node Subset: A subset of vertices.
- Node Data: Information on vertices.
- Node Distances: Data on distances between nodes.

Outputs

- Selected sub-network: A network of selected nodes.
- Distance Matrix: Distance matrix.
- Selected Items: Information on selected vertices.
- Highlighted Items: Information on highlighted vertices.
- Remaining Items: Information on remaining items (not selected or highlighted).

Network Explorer is the primary widget for visualizing network graphs. It displays a graph with Fruchterman-Reingold layout optimization and enables setting the color, size and label of nodes. One can also highlight nodes of specific properties and output them.

The visualization in **Network Explorer** works just like the one for **Scatter Plot**. To select a subset of nodes, draw a rectangle around the subset. Shift will add a new group. Ctrl-Shift (Cmd-Shift) will add to the existing group. Alt (Option) will remove from the group. Pressing outside of the network will remove the selection.

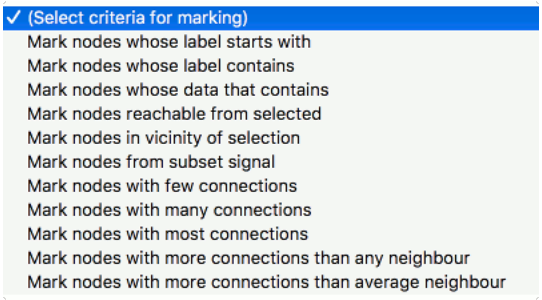


1. Information on the network. Reports on the number (and proportion) of nodes and edges. Press 'Re-layout' to re-compute nodes with Fruchterman-Reingold optimization. Select 'Randomize positions' starts from random position of nodes.
2. Set the color, shape, size and label of the nodes by attribute. Color will display the 10 most frequent values and color the rest as 'Other'. Shape is assigned to the 5 most frequent values and the rest is marked as 'Other'. *Label only selection and subset* is handy for keeping the projection organized.
3. Set the (relative) node size and edge width. By default, edge widths correspond to their weights. To see the weight value, select *Show edge weights*. By default, only the edges of selected nodes are labeled to keep the projection organized.

4. *Show color regions* colors the projection according to the majority node value. Deselect *Show legend* to hide the legend.
5. Select, zoom, pan and zoom to fit are the options for exploring the graph. The manual selection of data instances works as an angular/square selection tool. Double click to move the projection. Scroll in or out for zoom.

####Marking

Pressing *Select* will select and output the highlighted nodes. *Add to Group* adds to the existing selection, while *Add New Group* creates a new group.



The widget enables selection of nodes by the specified criterium:

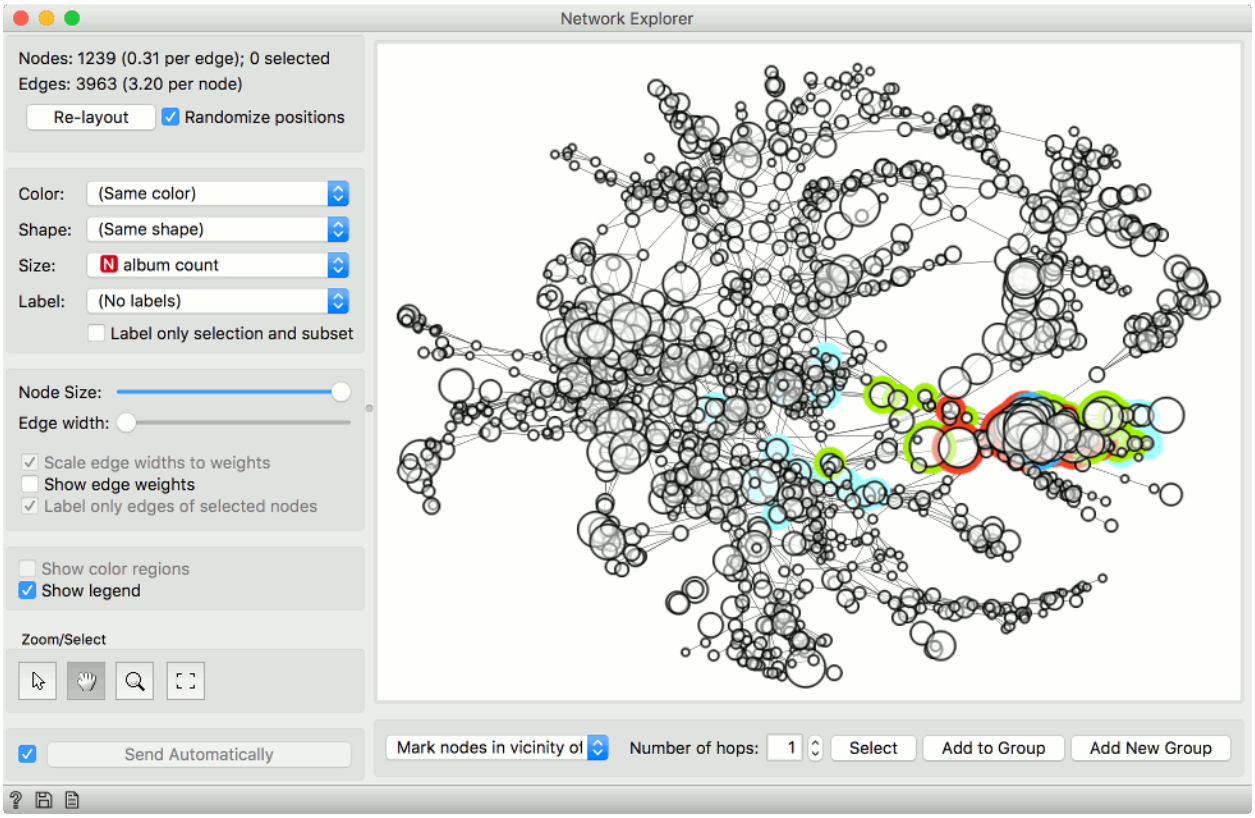
- *Mark nodes whose label starts with*. Set the condition to highlight the nodes whose label starts with the specified text. Label must be set for the highlighting to work. Press *Select* to select the highlighted nodes.
- *Mark nodes whose label contains*. Set the condition to highlight the nodes whose label contains the specified text. Label must be set for the highlighting to work. Press *Select* to select the highlighted nodes.
- *Mark nodes whose data contains*. Set the condition to highlight the nodes whose attributes contain the specified text. Press *Select* to select the highlighted nodes.
- *Mark nodes reachable from selected*. Highlight the nodes that can be reached from the selected nodes. At least one node has to be selected for the highlighting to work.
- *Mark nodes in vicinity of selection*. Highlight the nodes that are a selected number of hops away (first degree neighbors, second degree neighbors, etc.).
- *Mark nodes from subset signal*. Highlight the nodes that are neighbors of the nodes from the *Node Subset* input.
- *Mark nodes with few connections*. Highlight the nodes that have equal or less connections than the set number.
- *Mark nodes with many connections*. Highlight the nodes that have equal or more connections than the set number.
- *Mark nodes with most connections*. Highlight the nodes that have the most connections. The number of marked specifies how many top connected nodes to highlight (list is ranked).
- *Mark nodes with more connections than any neighbor*. Highlight the most connected nodes.
- *Mark nodes with more connections than average neighbor*. Highlight nodes whose degree is above average.

####Selecting a subset

Just like **Scatter Plot** widget, the **Network Explorer** supports group selection. To create the first group, either select nodes from the plot or highlight them by setting the criterium and pressing *Select*.

In the example below, we selected a single node (blue). Then we used *Mark nodes in vicinity of selection* to highlight neighbors of the selected node. We used *Add New Group* to create a new (red) group. Pressing *Add New Group* again will create yet another (green) group.

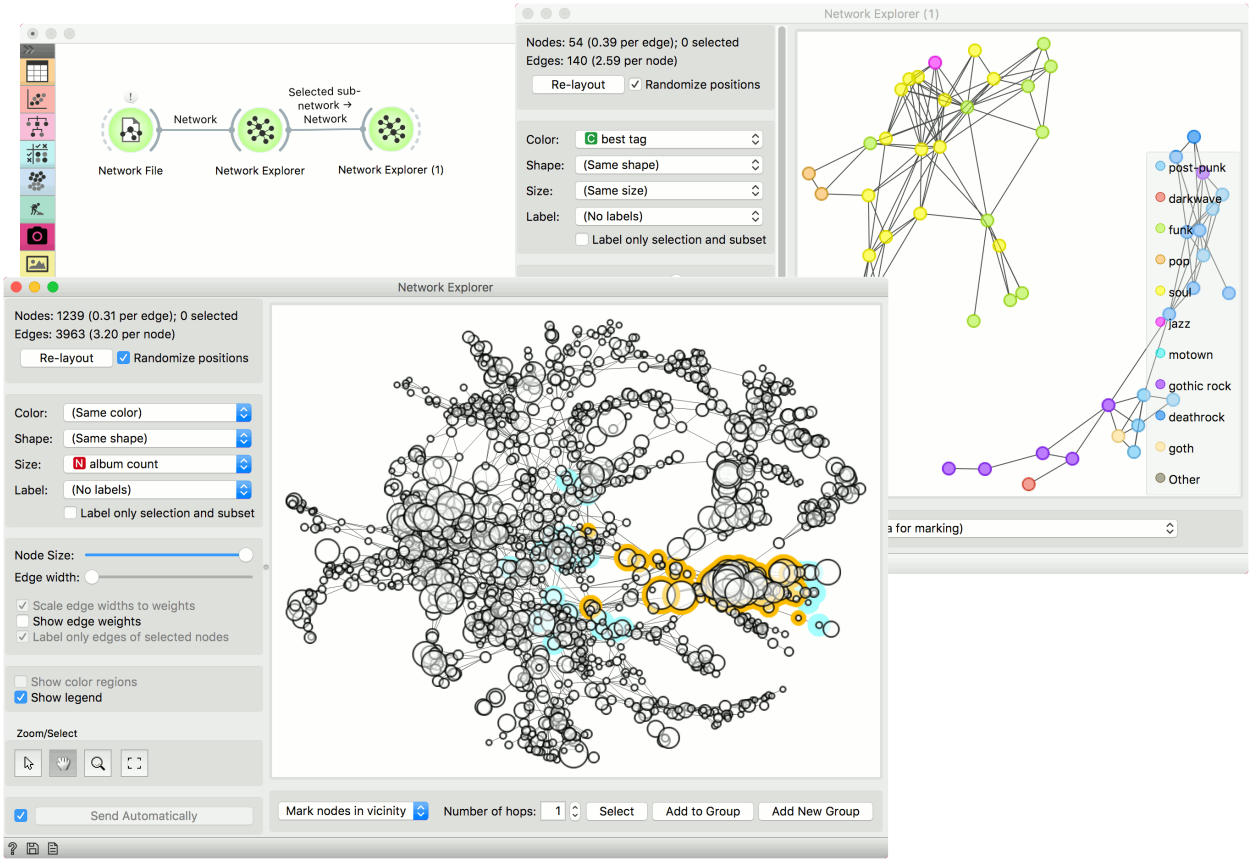
Add to Group would add the highlighted (light blue) nodes to the last (green) group. This make this widget a nice tool for visualizing hops and network propagation.



1.2.1 Example

In this example we will use the *lastfm* data set that can be loaded in the **Network File** widget under *Browse documentation networks*. The nodes of the network are musicians, which are characterized by the genre they play, number of albums produced and so on. The edges are the number of listeners on LastFm.

The entire data set is visualized in **Network Explorer**. In the widget, we removed the coloring and set the size of the nodes to correspond to the *album count*. Then we selected some nodes from the network. We can observe the selection in **Network Explorer (1)**.



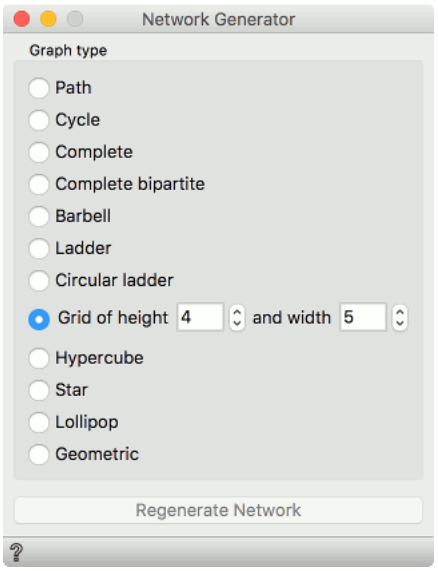
1.3 Network Generator

Construct example graphs.

Outputs

- Generated Network: An instance of Network Graph.

Network Generator constructs exemplary networks. It is mostly intended for teaching/learning about networks.



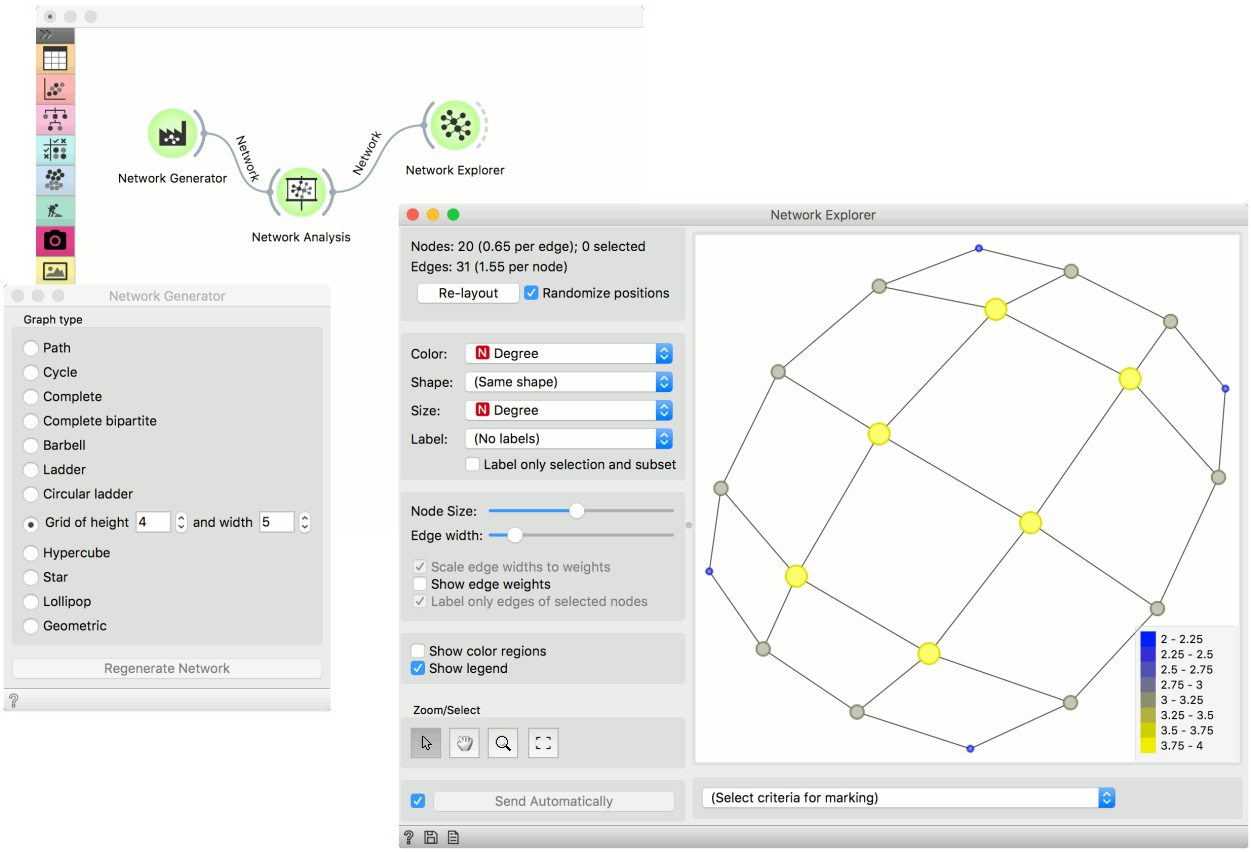
Graph options:

- Path: a graph that can be drawn so that all of its vertices and edges lie on a single straight line.
- Cycle: a graph that consists of a single cycle, i.e. some number of vertices (at least 3) are connected in a closed chain.
- Complete: simple undirected graph in which every pair of distinct vertices is connected by a unique edge.
- Complete bipartite: a graph whose vertices can be divided into two disjoint and independent sets.
- Barbell: two complete graphs connected by a path.
- Ladder: planar undirected graph with 2n vertices and 3n-2 edges.
- Circular ladder: Cartesian product of two path graphs.
- Grid: a graph whose drawing, embedded in some Euclidean space, forms a regular tiling.
- Hypercube: a graph formed from the vertices and edges of an n-dimensional hypercube.
- Star: Return the Star graph with n+1 nodes: one center node, connected to n outer nodes.
- Lollipop: a complete graph (clique) and a path graph, connected with a bridge.
- Geometric: an undirected graph constructed by randomly placing N nodes in some metric space.

Press *Regenerate Network* to output a new graph instance.

1.3.1 Example

Network Generator is a nice tool to explore typical graph structures.



Here, we generated a *Grid* graph of height 4 and width 5 and sent it to Network Analysis. We computed node degrees and sent the data to Network Explorer. Finally, we observed the generated graph in the visualization and set the size and color of the nodes to *Degree*. This is a nice tool to observe and explain the properties of networks.

1.4 Network Analysis

Statistical analysis of network data.

Inputs

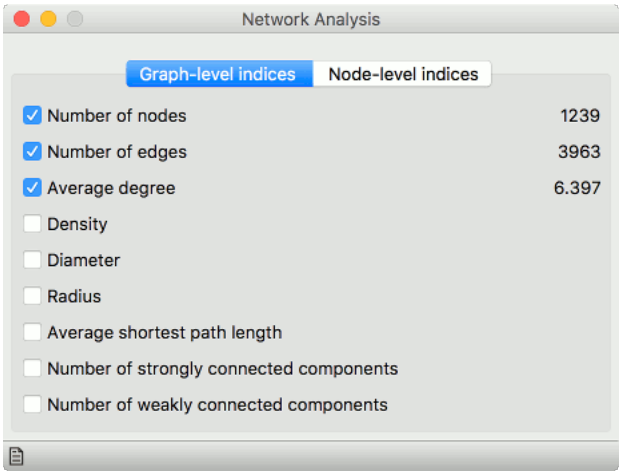
- Network: An instance of Network Graph.
- Items: Properties of a network file.

Outputs

- Network: An instance of Network Graph with appended information.
- Items: New properties of a network file.

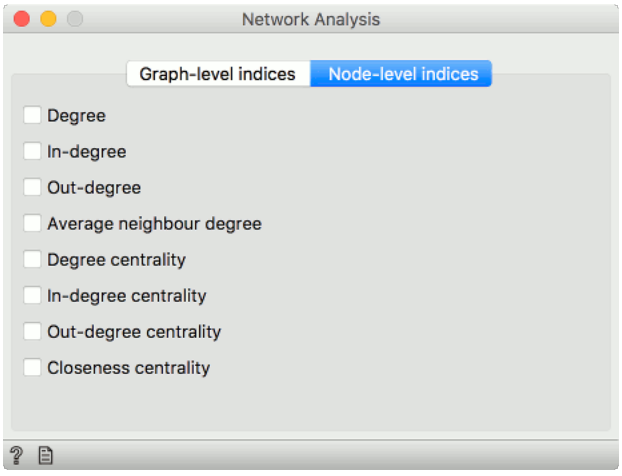
Network Analysis widget computes node-level and graph-level summary statistics for the network. It outputs a network with the new computed statistics and an extended item data table (node-level indices only).

####Graph level



- Number of nodes: number of vertices in a network.
- Number of edges: number of connections in a network.
- Average degree: average number of connections per node.
- Density: ratio between actual number of edges and maximum number of edges (fully connected graph).
- Diameter: maximum eccentricity of the graph.
- Radius: minimum eccentricity of the graph.
- Average shortest path length: expected distance between two nodes in the graph.
- Number of strongly connected components: parts of network where every vertex is reachable from every other vertex (for directed graphs only).
- Number of weakly connected components: parts of network where replacing all of its directed edges with undirected edges produces a connected (undirected) graph (for directed graphs only).

####Node level



- Degree: number of edges per node.
- In-degree: number of incoming edges in a directed graph.
- Out-degree: number of outgoing edges in a directed graph.
- Average neighbor degree: average degree of neighboring nodes.

- Degree centrality: ratio of other nodes connected to the node.
- In-degree centrality: ratio of incoming edges to a node in a directed graph.
- Out-degree centrality: ratio of outgoing edges from a node in directed graph.
- Closeness centrality: distance to all other nodes.

1.4.1 Example

This simple example shows how **Network Analysis** can enrich the workflow. We have used *lastfm.net* as our input network from Network File and sent it to **Network Analysis**. We've decided to compute *degree*, *degree centrality* and *closeness centrality* at node level.

We can visualize the network in Network Explorer. In the widget we color by *best tag*, as is the default for this data set. But now we can also set the size of the nodes to correspond to the computed *Degree centrality*. This is a great way to visualize the properties of the network.

1.5 Network Clustering

Detect clusters in a network.

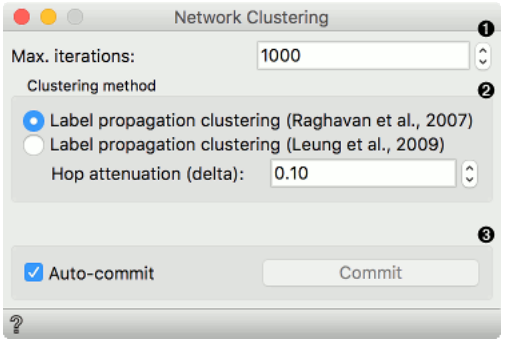
Inputs

- Network: An instance of Network Graph.

Outputs

- Network: An instance of Network Graph with clustering information appended.

Network Clustering widget finds clusters in a network. Clustering works with two algorithms, one from Raghavan et al. (2007), which uses label propagation to find appropriate clusters, and one from Leung et al. (2009), which builds upon the work from Raghavan and adds hop attenuation as a parameters for cluster formation.

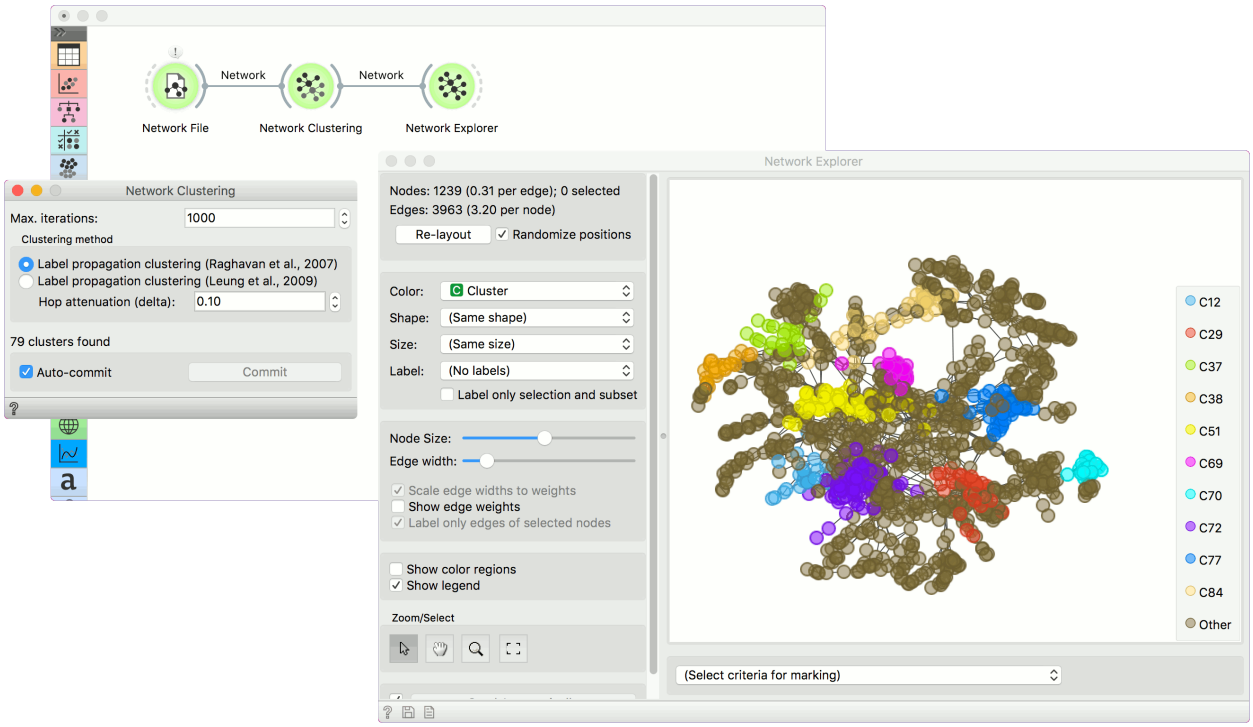


1. Max. iterations: maximum number of iteration allowed for the algorithm to run (can converge before reaching the maximum).
2. Clustering method:
   - Label propagation clustering (Raghavan et al., 2007)
   - Label propagation clustering (Leung et al., 2009) with hop attenuation.
3. Above appears the information on the number of clusters found. If *Auto-commit* is ticked, results will be automatically sent to the output. Alternatively, press *Commit*.

1.5.1 Example

Network Clustering can help you uncover cliques and highly connected groups in a network. First, we will use Network File to load *lastfm.net* data set. Then we will pass the network through **Network Clustering**. The widget found 79 clusters in a network. To visualize the results, use Network Explorer and set *Color* attribute to *Cluster*. This will color network nodes with the corresponding cluster color - this is a great way to visualize highly connected groups in dense networks.

Keep in mind that **Network Explorer** will color the 10 largest clusters and color the rest as 'Other'.



1.6 Network Of Groups

Group instances by feature and connect related groups.

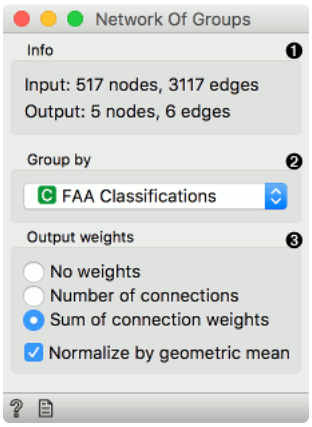
Inputs

- Network: An instance of network graph.
- Data: Properties of a network graph.

Outputs:

- Network: A grouped network graph.
- Data: Properties of the group network graph.

Network of Groups is the network version of the group-by operation. Nodes with the same values of the attribute, selected in the dropdown, will be represented as a single node.



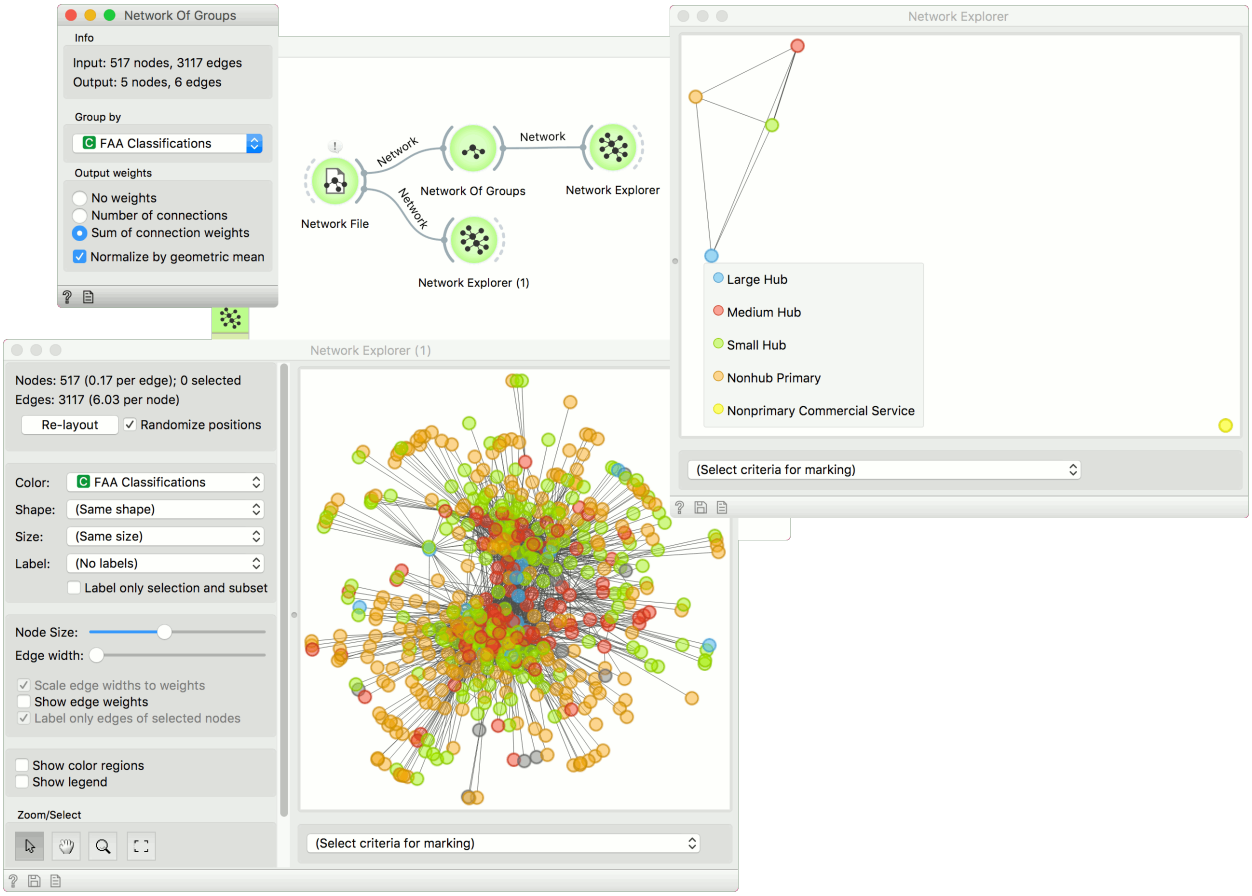
1. Information on the input and output network.
2. Select the attribute to group by.
3. Compute weights:
    - *No weights*: all weights are set to 1.
    - *Number of connections*: weight edges by the number of connections between the groups.
    - *Sum of connection weights*: weight edges by the sum of weights of connections between the groups. *Normalize by geometric mean* divides weights by the geometric mean of the number of connections between the two groups.

1.6.1 Example

In this example we are using *airtraffic* data set, that we loaded in the Network File widget. We see the entire data set in **Network Explorer (1)**.

Then we use **Network of Groups** to group the network by the *FAA Classifications* attribute. All nodes with the same value of this attribute will be represented as a single node in the output. There is an edge between the two nodes, if they share connections in the original network.

The grouped network is shown in Network Explorer.



1.7 Network From Distances

Constructs a network from distances between instances.

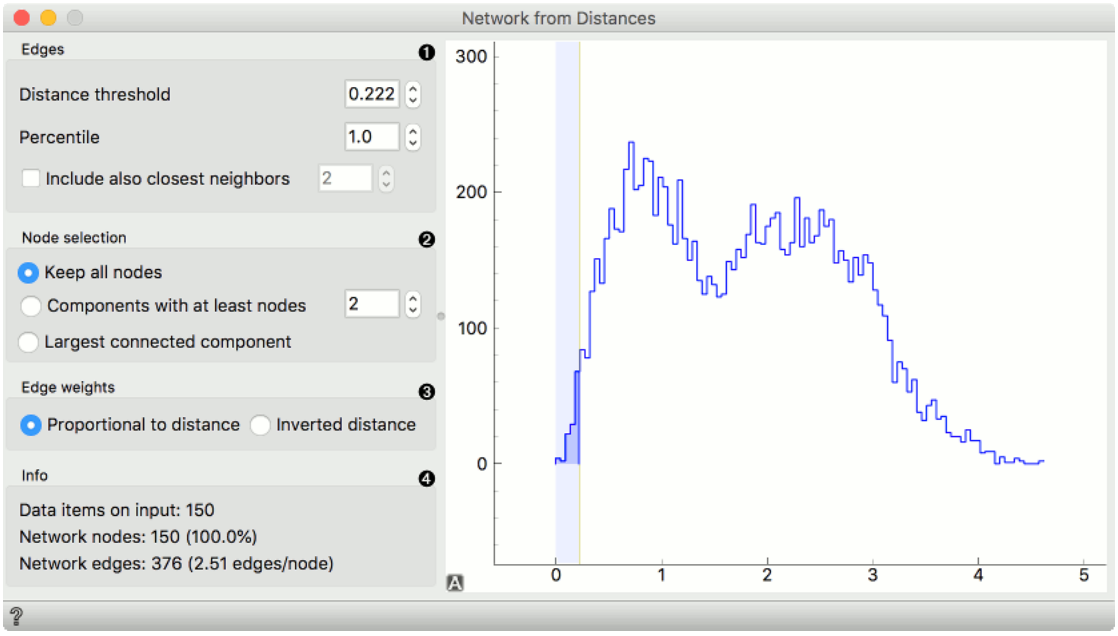
Inputs

- Distances: A distance matrix.

Outputs

- Network: An instance of Network Graph.
- Data: Attribute-valued data set.
- Distances: A distance matrix.

Network from Distances constructs a network graph from a given distance matrix. Graph is constructed by connecting nodes from the matrix where the distance between nodes is below the given threshold. In other words, all instances with a distance lower than the selected threshold, will be connected.

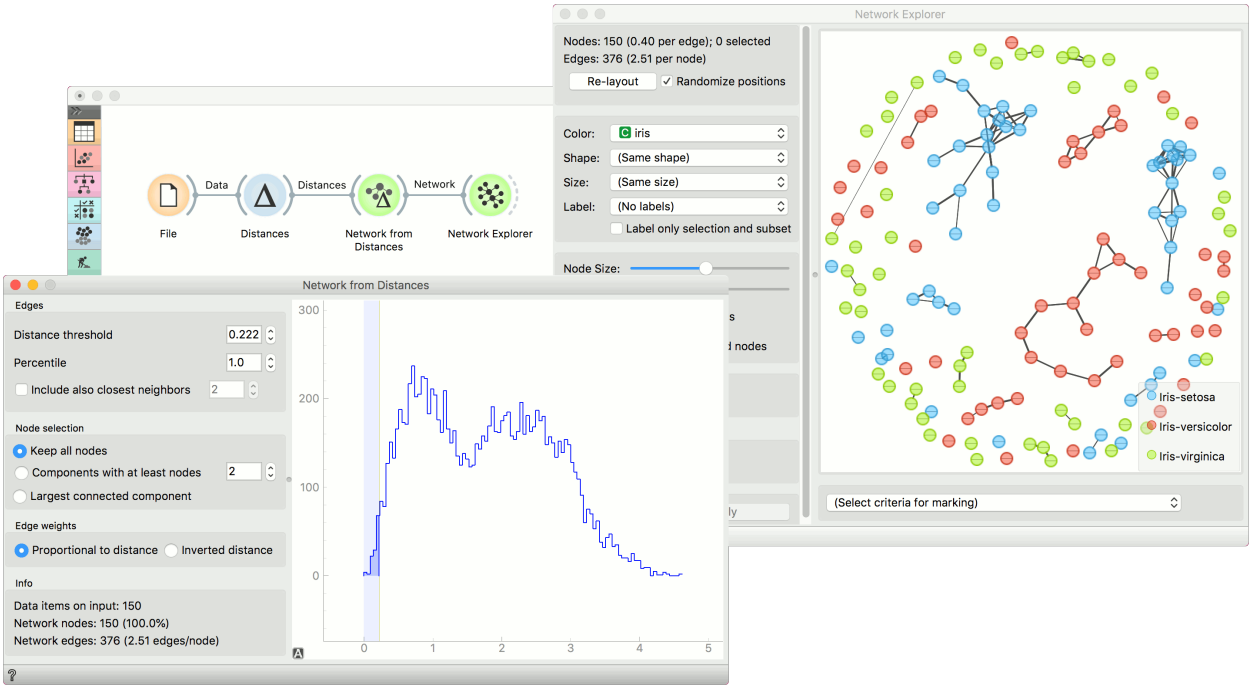


1. Edges:
   - Distance threshold: a closeness threshold for the formation of edges.
   - Percentile: the percentile of data instances to be connected.
   - *Include also closest neighbors*: include a number of closest neighbors to the selected instances.
2. Node selection:
   - Keep all nodes: entire network is on the output.
   - Components with at least X nodes: filters out nodes with less than the set number of nodes.
   - Largest connected component: keep only the largest cluster.
3. Edge weights:
   - Proportional to distance: weights are set to reflect the distance (closeness).
   - Inverted distance: weights are set to reflect the inverted distance (difference).

4. Information on the constructed network:
    - Data items on input: number of instances on the input.
    - Network nodes: number of nodes in the network (and the percentage of the original data).
    - Network edges: number of constructed edges/connections (and the average number of connections per node).

1.7.1 Example

Network from Distances creates networks from distance matrices. It can transform data sets from a data table via distance matrix into a network graph. This widget is great for visualizing instance similarity as a graph of connected instances.



We took *iris.tab* to visualize instance similarity in a graph. We sent the output of **File** widget to **Distances**, where we computed Euclidean distances between rows (instances). Then we sent the output of **Distances** to **Network from Distances**, where we set the distance threshold (how similar the instances have to be to draw an edge between them) to 0.222. We kept all nodes and set edge weights to *proportional to distance*.

Then we observed the constructed network in a Network Explorer. We colored the nodes by *iris* attribute.

1.8 Single Mode

Convert multimodal graphs to single modal.

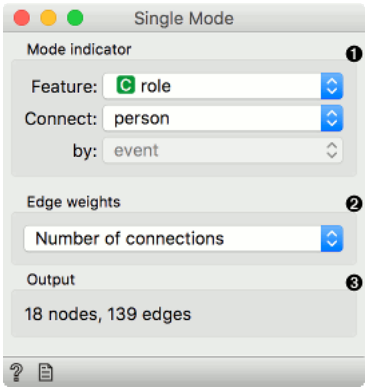
Inputs

- Network: An instance of a bipartite network graph.

Outputs

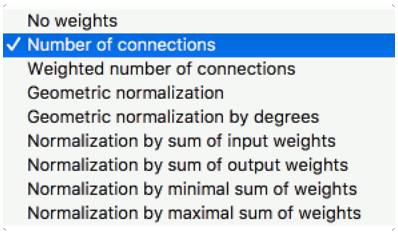
- Network: An instance of single network graph.

Single Mode works with bipartite (or multipartite) networks, where different parts are distinguished by values of the chosen discrete variable. A typical example would be a network that connects persons with events that they attended. The widget creates a new network, which contains the nodes from the chosen group of original network's nodes (e.g. persons). Two nodes in the resulting network are connected if they share a common neighbor from the second chosen group (e.g. events).



1. Mode indicator:
    - *Feature*: discrete feature labeling network subsets.
    - *Connect*: value used as nodes.
    - *by*: value used as edges.
2. Edge weights: compute weights for the output network.
3. Information on the output network.

####Edge weights



- *No weights*: all weights are set to 1.
- *Number of connections*: weights correspond to the number of common connections (e.g. events the two people participated in).
- *Weighted number of connections*: weights correspond to the sum of the product of original edge weights that connect each person with the event.

For details and for description of other options, see Vlado Batagelj's Introduction to Network Science using Pajek, section 7, page 38.

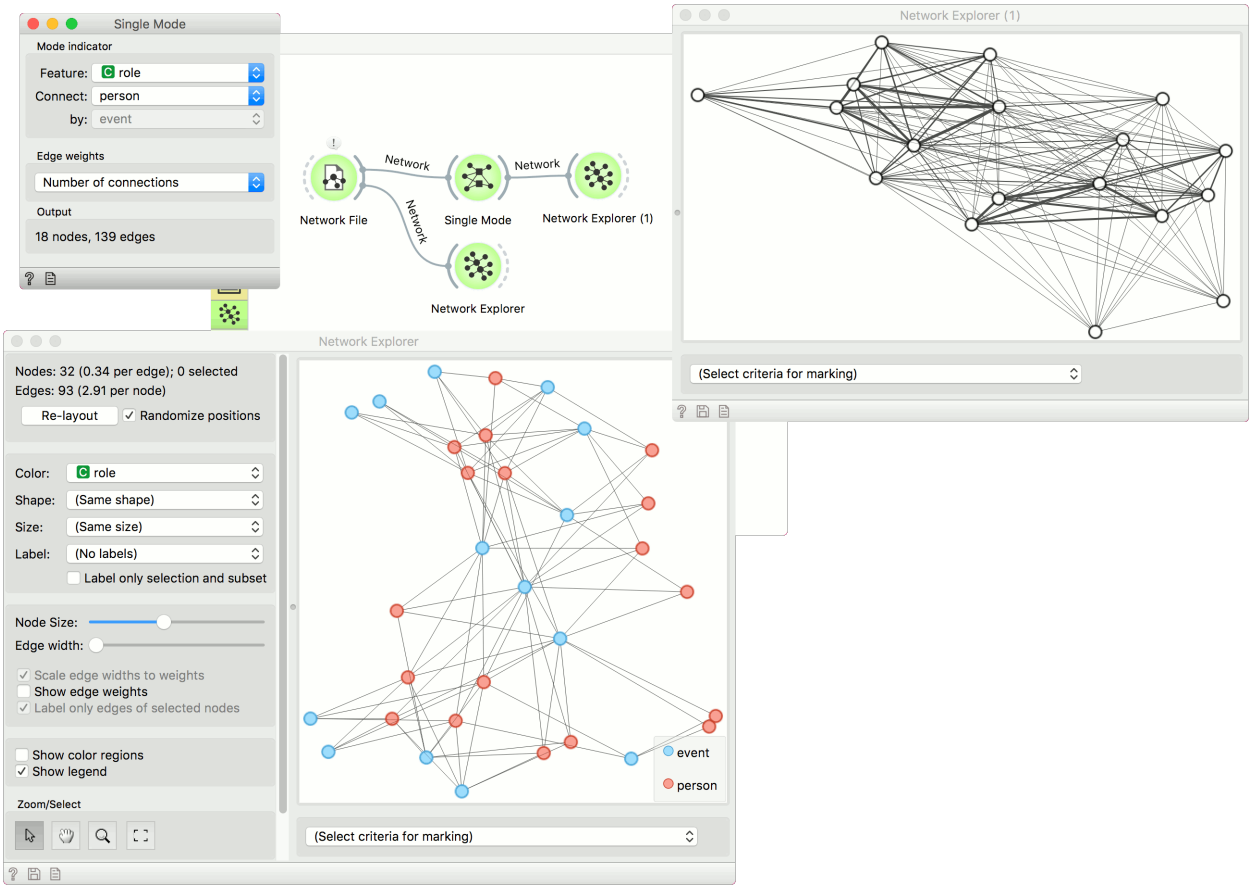
1.8.1 Example

For this example we have used the famous *davis* data set that describes ladies from the south of the United States and the events they have participated in. The network thus consists of nodes that are either persons or events. Node role is described in the attribute *role*. We load the network with Network File.

We see the original file in Network Explorer. The blue nodes are events and the red ones are persons. Events are attended by persons. A node connects a person with the event, if the person attended the event. Now we can observe which people attended the same event or, conversely, which events were attended by the same people.

In **Single Mode** we set the feature describing the role of the nodes. It so happens that our attribute is named *role*. We connect persons (nodes) by the events they attended (edges). Edge weight will be the number of connections in common. In translation, edge weights will be the number of events both people attended.

Network Explorer (1) shows the final network.



CHAPTER 2

Indices and tables

- genindex
- modindex
- search