


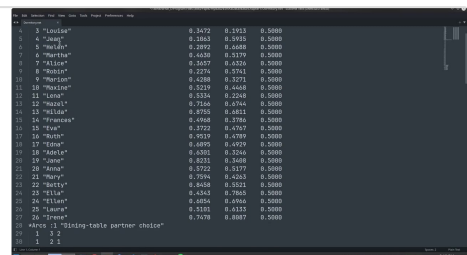
# Pajek

<https://github.com/chaitanyabisht/Pajek-Presentation>

## Introduction to Pajek

Pajek Presentation for CS552 Network Science at Indian Institute of Technology Bhilai.

 <https://www.youtube.com/watch?v=I-WvMAxJCqA>



1	"Ljubljana"	0.34172	0.19113	0.50000
2	"Ljubljana"	0.28613	0.19113	0.50000
3	"Ljubljana"	0.28917	0.48688	0.50000
4	"Maribor"	0.44326	0.51779	0.50000
5	"Ljubljana"	0.28357	0.43251	0.50000
6	"Ravne"	0.22274	0.57141	0.50000
7	"Ravne"	0.40308	0.32171	0.50000
8	"Ravne"	0.52119	0.32171	0.50000
9	"Ljubljana"	0.51116	0.22448	0.50000
10	"Ljubljana"	0.71164	0.44448	0.50000
11	"Ljubljana"	0.47950	0.68111	0.50000
12	"Ljubljana"	0.49603	0.47901	0.50000
13	"Ljubljana"	0.37222	0.47901	0.50000
14	"Ljubljana"	0.48905	0.49291	0.50000
15	"Ljubljana"	0.48905	0.49291	0.50000
16	"Ljubljana"	0.48905	0.49291	0.50000
17	"Ljubljana"	0.48905	0.49291	0.50000
18	"Ljubljana"	0.48905	0.49291	0.50000
19	"Ljubljana"	0.48905	0.49291	0.50000
20	"Ljubljana"	0.48905	0.49291	0.50000
21	"Ljubljana"	0.48905	0.49291	0.50000
22	"Ljubljana"	0.48905	0.49291	0.50000
23	"Ljubljana"	0.48905	0.49291	0.50000
24	"Ljubljana"	0.48905	0.49291	0.50000
25	"Ljubljana"	0.48905	0.49291	0.50000
26	"Ljubljana"	0.48905	0.49291	0.50000
27	"Ljubljana"	0.48905	0.49291	0.50000
28	"Ljubljana"	0.48905	0.49291	0.50000
29	"Ljubljana"	0.48905	0.49291	0.50000
30	"Ljubljana"	0.48905	0.49291	0.50000
31	"Ljubljana"	0.48905	0.49291	0.50000
32	"Ljubljana"	0.48905	0.49291	0.50000
33	"Ljubljana"	0.48905	0.49291	0.50000
34	"Ljubljana"	0.48905	0.49291	0.50000
35	"Ljubljana"	0.48905	0.49291	0.50000
36	"Ljubljana"	0.48905	0.49291	0.50000
37	"Ljubljana"	0.48905	0.49291	0.50000
38	"Ljubljana"	0.48905	0.49291	0.50000
39	"Ljubljana"	0.48905	0.49291	0.50000
40	"Ljubljana"	0.48905	0.49291	0.50000
41	"Ljubljana"	0.48905	0.49291	0.50000
42	"Ljubljana"	0.48905	0.49291	0.50000
43	"Ljubljana"	0.48905	0.49291	0.50000
44	"Ljubljana"	0.48905	0.49291	0.50000
45	"Ljubljana"	0.48905	0.49291	0.50000
46	"Ljubljana"	0.48905	0.49291	0.50000
47	"Ljubljana"	0.48905	0.49291	0.50000
48	"Ljubljana"	0.48905	0.49291	0.50000
49	"Ljubljana"	0.48905	0.49291	0.50000
50	"Ljubljana"	0.48905	0.49291	0.50000
51	"Ljubljana"	0.48905	0.49291	0.50000
52	"Ljubljana"	0.48905	0.49291	0.50000
53	"Ljubljana"	0.48905	0.49291	0.50000
54	"Ljubljana"	0.48905	0.49291	0.50000
55	"Ljubljana"	0.48905	0.49291	0.50000
56	"Ljubljana"	0.48905	0.49291	0.50000
57	"Ljubljana"	0.48905	0.49291	0.50000
58	"Ljubljana"	0.48905	0.49291	0.50000
59	"Ljubljana"	0.48905	0.49291	0.50000
60	"Ljubljana"	0.48905	0.49291	0.50000
61	"Ljubljana"	0.48905	0.49291	0.50000
62	"Ljubljana"	0.48905	0.49291	0.50000
63	"Ljubljana"	0.48905	0.49291	0.50000
64	"Ljubljana"	0.48905	0.49291	0.50000
65	"Ljubljana"	0.48905	0.49291	0.50000
66	"Ljubljana"	0.48905	0.49291	0.50000
67	"Ljubljana"	0.48905	0.49291	0.50000
68	"Ljubljana"	0.48905	0.49291	0.50000
69	"Ljubljana"	0.48905	0.49291	0.50000
70	"Ljubljana"	0.48905	0.49291	0.50000
71	"Ljubljana"	0.48905	0.49291	0.50000
72	"Ljubljana"	0.48905	0.49291	0.50000
73	"Ljubljana"	0.48905	0.49291	0.50000
74	"Ljubljana"	0.48905	0.49291	0.50000
75	"Ljubljana"	0.48905	0.49291	0.50000
76	"Ljubljana"	0.48905	0.49291	0.50000
77	"Ljubljana"	0.48905	0.49291	0.50000
78	"Ljubljana"	0.48905	0.49291	0.50000
79	"Ljubljana"	0.48905	0.49291	0.50000
80	"Ljubljana"	0.48905	0.49291	0.50000
81	"Ljubljana"	0.48905	0.49291	0.50000
82	"Ljubljana"	0.48905	0.49291	0.50000
83	"Ljubljana"	0.48905	0.49291	0.50000
84	"Ljubljana"	0.48905	0.49291	0.50000
85	"Ljubljana"	0.48905	0.49291	0.50000
86	"Ljubljana"	0.48905	0.49291	0.50000
87	"Ljubljana"	0.48905	0.49291	0.50000
88	"Ljubljana"	0.48905	0.49291	0.50000
89	"Ljubljana"	0.48905	0.49291	0.50000
90	"Ljubljana"	0.48905	0.49291	0.50000
91	"Ljubljana"	0.48905	0.49291	0.50000
92	"Ljubljana"	0.48905	0.49291	0.50000
93	"Ljubljana"	0.48905	0.49291	0.50000
94	"Ljubljana"	0.48905	0.49291	0.50000
95	"Ljubljana"	0.48905	0.49291	0.50000
96	"Ljubljana"	0.48905	0.49291	0.50000
97	"Ljubljana"	0.48905	0.49291	0.50000
98	"Ljubljana"	0.48905	0.49291	0.50000
99	"Ljubljana"	0.48905	0.49291	0.50000
100	"Ljubljana"	0.48905	0.49291	0.50000

## Introduction

Pajek was developed by Vladimir Batagelj and Andrej Mrvar, who are both professors at the University of Ljubljana in Slovenia. The development of Pajek started in the mid-1990s, with the first public release in 1996.





The name "Pajek" is derived from the Slovenian word for spider, reflecting its ability to analyze and visualize complex networks.

The Slovenian pronunciation is "Pa-yek"

# Installation

Pajek can be downloaded from the following website:

<http://mrvar.fdv.uni-lj.si/pajek/>

	Ver.	32 bit	64 bit
Sep 1, 2023	5.18	Web Start   <u>Install Shield</u> <u>Install-Zip</u> <u>Portable</u>	Web Start    <u>Install Shield</u> <u>Install-Zip</u> <u>Portable</u>
2018	<u>Pajek Book Edition 3</u>		
<a href="mailto:pajek@list.fmf.uni-lj.si">pajek@list.fmf.uni-lj.si</a>		<u>Datasets</u>	

Based on your system, download the installer file. We recommend to download the **Install Shield** in the **64 bit** section.

## For Windows

Windows users can simply open the installer, and follow the installation wizard.

## For Linux

As Pajek is exclusively designed for Windows operating systems, Linux users may emulate it through Wine, which is the recommended approach by the tool's maintainers. For Linux users, particularly those utilizing Debian-based

distributions such as Ubuntu, we offer a script facilitating the installation of Wine.

```
# Enable 32-bit programs
sudo dpkg --add-architecture i386

# Add the repository
sudo mkdir -pm755 /etc/apt/keyrings
sudo wget -O /etc/apt/keyrings/winehq-archive.key https://dl.winehq.org/winehq/releases/archive/winehq-archive.key

# Update package repository
sudo apt update

# Install Wine
sudo apt install --install-recommends winehq
```

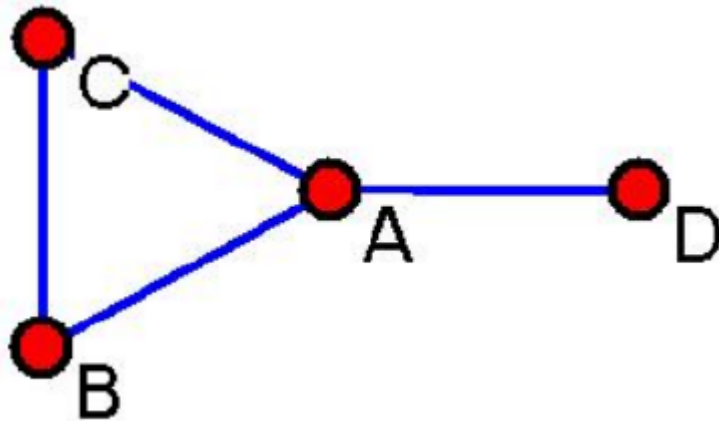
## Basics of .net file

Pajek files have the extension ".net" and contain network data such as vertices, edges, and their attributes.

### Example 1

```
*Vertices 4
1 "A"
2 "B"
3 "C"
4 "D"
*Edges
```

```
1 2 1
1 3 1
2 3 1
1 4 1
```



The `*Vertices` section specifies the number of vertices in the network (in this case, 4) and each vertex is represented by a line containing its ID and optional label.

The `*Edges` section specifies the edges between vertices along with their optional weights.

## Example 2

This method simply defines the arcs, which are directed links in the Pajek

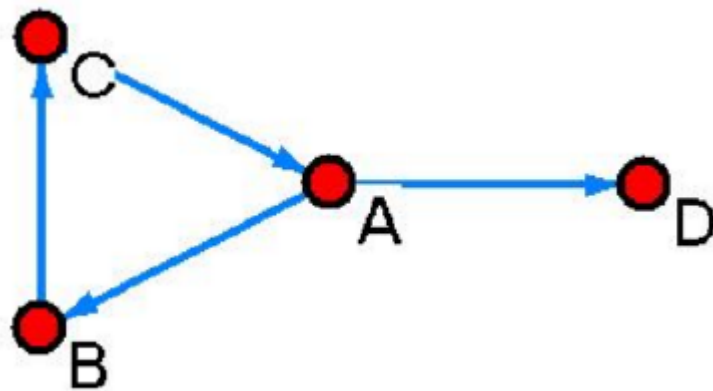
```
*Vertices 4
1 "A"
2 "B"
```

```

3 "C"
4 "D"

*Arcs
1 2 1 # It means a link from 1 to 2 and it has a edge weight
2 3 1
3 1 1
1 4 1

```



## Example 3

This method simply defines the arcs in the arcslist representation.

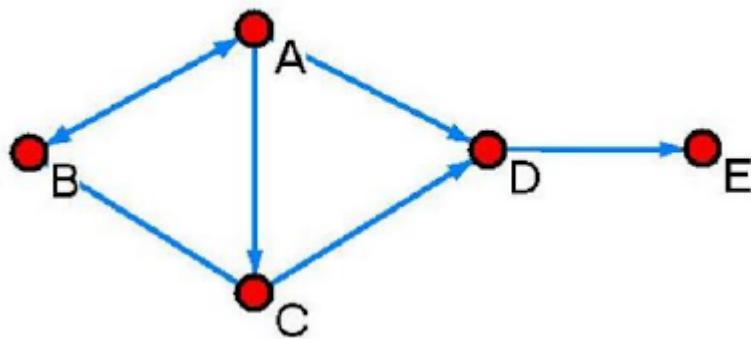
```

*Vertices 4
1 "A"
2 "B"
3 "C"
4 "D"

*Arcslist
1 2 3 4 # This means that there is a link from 1 to 2, 3, and

```

```
2 1
3 4 2
4 5
```



## Example 4

This method uses the adjacency matrix representation to define directed links.

```
*Vertices5
```

```
1 "A"
```

```
2 "B"
```

```
3 "C"
```

```
4 "D"
```

```
5 "E"
```

```
*Matrix
```

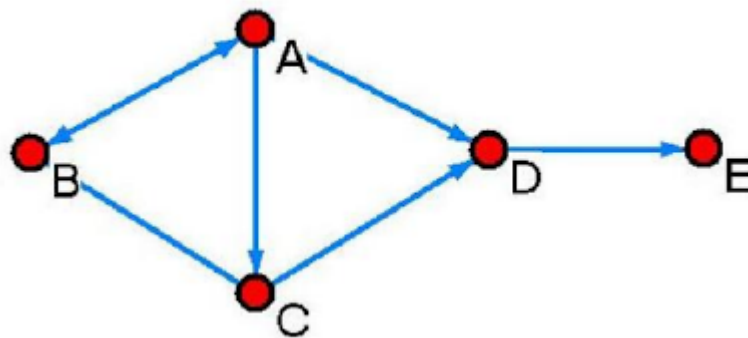
```
0 1 1 1 0
```

```
1 0 0 0 0
```

```
0 1 0 1 0
```

```
0 0 0 0 1
```

```
0 0 0 0 0
```



## Example 5

This is an example taken from Chapter 1 of Exploratory Social Network Analysis with Pajek. Here we are also using labels for the relations, such as "Dining-table partner choice"

File: dormitory.net

```

*Vertices 26
1 "Ada" 0.1646 0.2144 0.5000 # These define the x,y,z coordi
2 "Cora" 0.0481 0.3869 0.5000
3 "Louise" 0.3472 0.6969 0.5000

...

*Arcs :1 "Dining-table partner choice"
1 3 2
1 2 1
2 1 1

...

```

```
*Edges :2 "Cooperation"
```

```
1 2 1
```

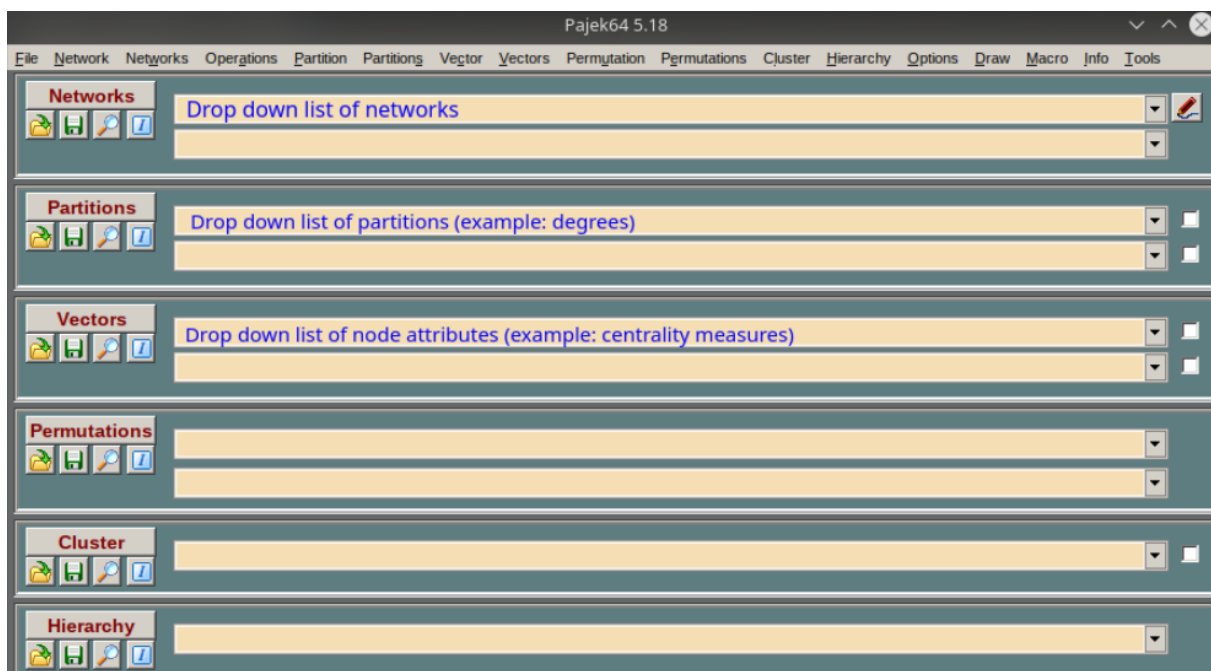
```
2 4 1
```

```
1 4 1
```

```
. . .
```

## Interface

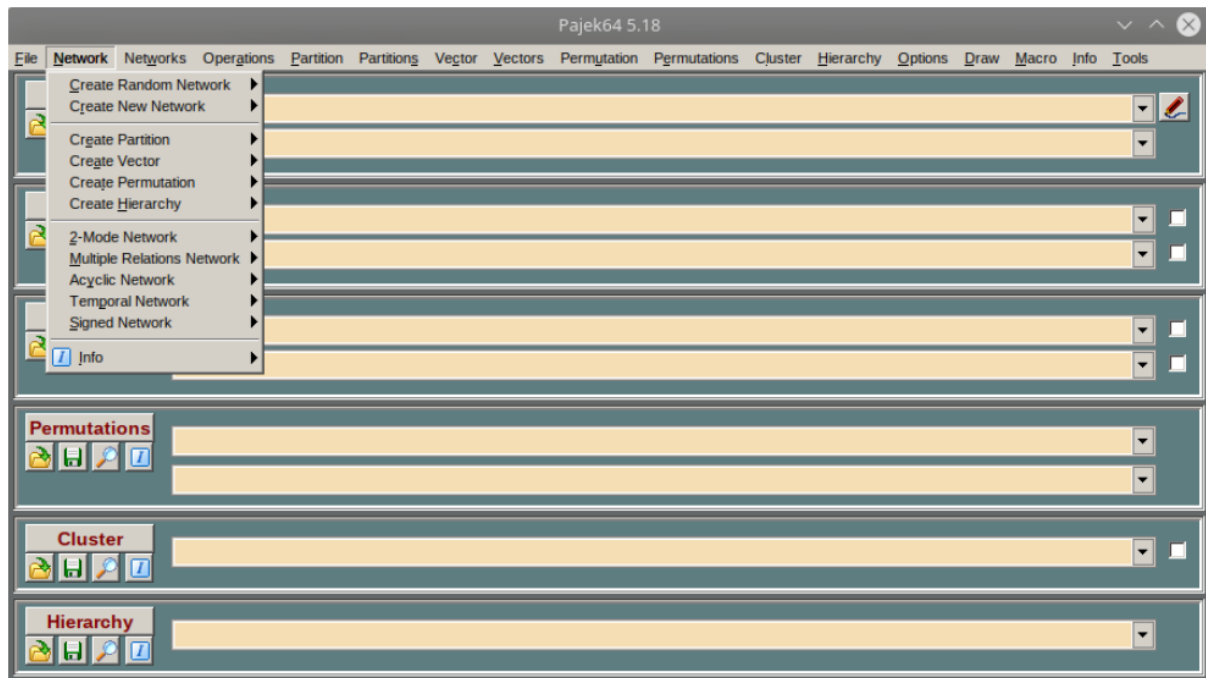
Pajek interface is very unique, its imperative to understand what each section means.



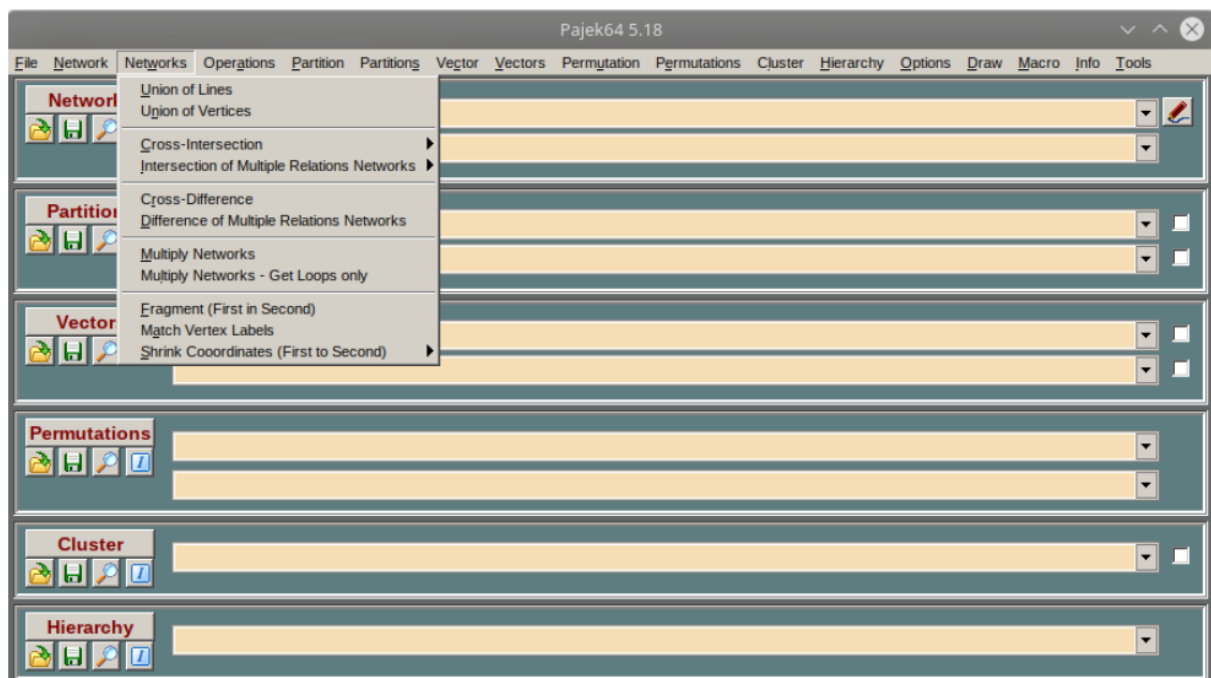
Each of these menu items are categorized according to its functionality.



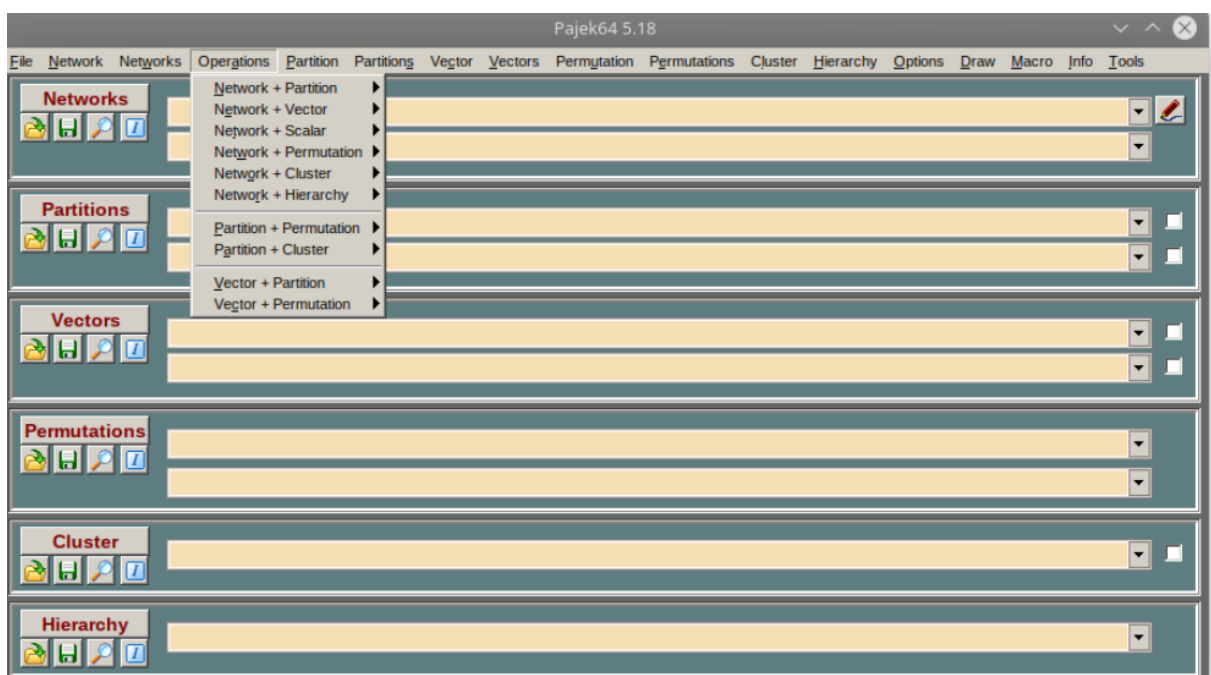
If you want to work on a single Network, you select the Network Menu.



If you want to work on two networks for example, you select the Networks menu.



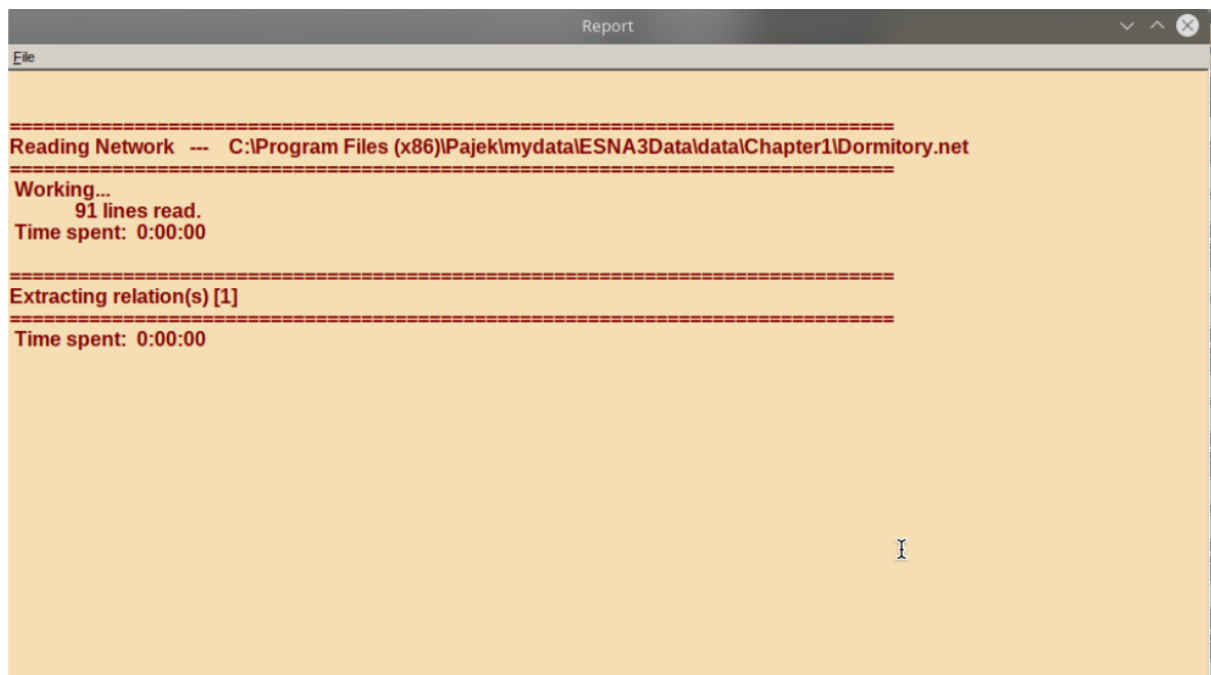
If you want to work with a Network and a Partition for example, you select the Operations menu.



## Report Window

In Pajek, the "Report" window is a feature that provides various statistical and descriptive information about the network being analyzed. This information can include network metrics such as the number of vertices, edges, connected components, average degree, clustering coefficients, and more. The Report window allows users to gain insights into the structural properties of the network and facilitates further analysis and interpretation of the data.

Info > Child Windows > Report Window > Show



## Creating a network in Pajek

### Blank Network

We can start with a blank network, having just vertices and no connections.

Network > Create New Network > Empty Network

Enter the number of vertices = 10

File > Network > View/Edit

## Vectors

Vector is one of the data object that Pajek uses to execute different kinds of operations in a network.

In vector data object we save the value associated to each node after the operation completes. Using vector data object we will try to find the following centralities:

1. Between Centrality
2. Closeness Centrality
3. Eigenvector Centrality
4. Degree Centrality

Before that, let's first understand the format in which vector data object is implemented

### Format

The vector data object is saved in `.vec` extension. The format of the file is

```
*Vertices <number of vertices>
<Value associated with node 1>
<Value associated with node 2>
...
<Value associated with node n>
```

The format is very similar to `.net` file structure. We first define the **Vertices** using `*Vertices n` where n is the number of vertices and then in the following n lines we write the values given to the n nodes. In ith line we provide the value for ith node.

## Betweenness Centrality

To find the betweenness centrality, you should follow the following steps.

```
Network > Create Vector > Centrality > Betweenness
```

## Closeness Centrality

To find the closeness centrality, you should follow the following steps

```
Network > Create Vector > Centrality > Closeness
```

There will be three options provided. They are the following with explanation

1. **Input:** centrality of each vertex according to distances of other vertices to selected vertex.
2. **Output:** centrality of each vertex according to distances of selected vertex to all other vertices.
3. **All:** forget direction of lines – consider network as undirected.

## EigenVector Centrality

To find the eigenvector centrality, you should follow the following steps

```
Network > Create Vector > Centrality > Hubs-Authorities
```

Note: Hubs-Authorities is a concept used in directed graph. In undirected graph, it is equivalent to eigenvector centrality.

## Degree Centrality

To find the degree centrality, you should follow the following steps

Network > Create Vector > Centrality > Degree

In this case as well, we are provided with three options

1. Input: Number of lines into the node.
2. Output: Number of lines out of the node.
3. All: Number of neighbors of node.

## Extra Points

- If you try to draw the network using vector values in pajek then the sizes of nodes will change relative to the values provided in the vector.
- To change the size scale, you change it using the following steps

Options > Size > Of Vertices

You can then write the value in the pop up box.

- To be able to see the vector values in the drawn network, follow these steps

Options > Mark Vertices Using > Vector Values

## Partitions

Partition is yet another data object that Pajek uses. A partition classifies the nodes in a network into different clusters. One restriction is that each node can only belong to a unique partition. Using the partition data object, you can perform three types of analyses.

1. You can find the connected components in a graph
2. You can find the distance of all vertices in a graph from a given vertex
3. You can find the degree of each vertex in the graph

## Format

We use the `.clu` extension for the partition data object. The `clu` stands for clusters.

Just like the `.net` file, you start with "`*Vertices` " followed by the number of vertices. Note that the number of vertices in the `.clu` file must match with the number of vertices in the `.net` file for which you are making the cluster. From the next line onwards, you simply have to enter the partition number that each vertex will be classified into. An example is shown below.

```
*Vertices 5
3
2
2
1
3
```

The above example simply states that vertex 0 is assigned to the partition 3, vertex 1 is assigned to the partition 2, vertex 2 is assigned to the partition 2 and so on...

Note that you don't usually have to create this `.clu` file yourself. These files are automatically created using the interface of Pajek

## Connected Components

In order to find the connected components, you can follow the following steps

1. Choose Network from the Interface bar
2. Select Create Partition from the drop-down menu
3. Select Components

You will be presented with 3 options

- a. Weak : Weakly connected components

- b. Strong : Strongly connected components
- c. Strong-period : Will not be discussed here

If you want only those components that are strongly connected (there is a directed edge between every pair of vertices i.e. bidirectional path between any 2 vertices), then you can choose the Strong option.

If you would like to relax this constraint, then you can choose the Weak option.

If the graph you are dealing with is undirected, then both options give the exact same result.

After the partition is generated, you can simply select the partition from the drop-down menu and tickmark the box associated with it and click on the draw button to get your required analysis.

## **Distance of nodes from a given node**

In order to find the distance of all nodes from the given node, you can follow the following steps

1. Choose Network from the Interface bar
2. Select Create Partition from the drop-down menu
3. Select k-Neighbours

You will be presented with 3 options again

- a. Input : Incoming edges only
- b. Output : Outgoing edges only
- c. All : Both types of edges

If you want to consider only those edges that are incoming to the given node, you can choose Input



If you want to consider only those edges that are outgoing from the given node, choose Output

If you want to consider both or if your graph is undirected, then choose All

Following this step, you will be presented with a dialog box asking you to specify which node you want to calculate all the distances from. You will need to remember the numeric label of your node since that is what you will have to enter (labels of the node will not work)

After the partition is generated, you can simply select the partition from the drop-down menu and tickmark the box associated with it and click on the draw button to get your required analysis.

## **Degree of each vertex**

In order to find the degree of each vertex, you can follow the following steps

1. Choose Network from the Interface bar
2. Select Create Partition from the drop-down menu
3. Select Degree

You will be presented with 3 options again

- a. Input : Incoming edges only
- b. Output : Outgoing edges only
- c. All : Both types of edges

Again, input, output and all refer to incoming, outgoing or both types of edges.

After the partition is generated, you can simply select the partition from the drop-down menu and tickmark the box associated with it and click on the draw button to get your required analysis.

Although we have seen how to find the degree using a vector (degree centrality), there are some ways in which using a partition is more beneficial for the task at hand.

One main point is that in vectors, the nodes are differentiated on the basis of their size. Higher the centrality measure, greater the size. Unfortunately, this means that if the centrality measure is 0, then the node is not even visible. However, partitions differentiate nodes by colour. All nodes in a single partition are coloured by a single colour and nodes in different partitions are coloured differently. This allows you to clearly differentiate between nodes of a different degree (or any other measure).

One concern might be that partition does not label the nodes on the basis of any value, which means that we cannot see the degree value as a label. There is a simple work-around for this. In Pajek, you may have noticed that there are two or more drop-down menus for partitions and other data objects.

Using this, if you select the same degree partition that was generated on both the drop-down menus, then you can draw the graph while keeping both the partitions tickmarked. Once inside the draw menu, you can select Options → Mark Vertices Using → Cluster of Second Partitions.

Once you select this, every node will be labelled as per the value of the second partition (in this case, it is the same degree partition). This may not usually make sense because the value of the partition need not be the same as the degree, however, Pajek somehow ensures that the value of the partition is actually the same as the degree of all vertices in that partition.

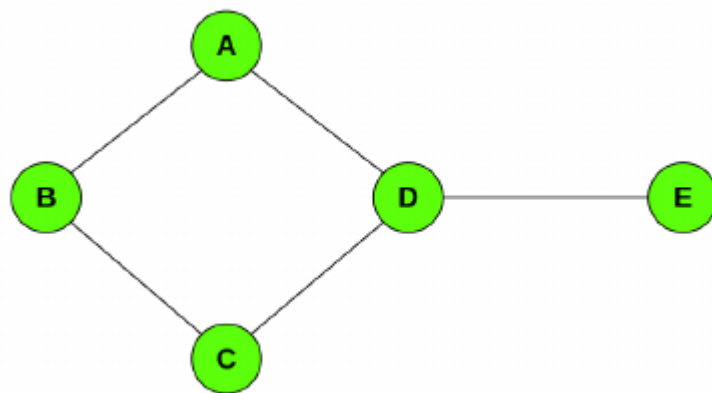
This gives a very nice view of the degree of all the nodes in the graph.

## Performing BFS and DFS in Pajek

In Pajek, a permutation typically refers to a reordering of the nodes or vertices in a network. They are typically used for the following tasks

- BFS and DFS (Both weak and strong are supported!)
- Graph Drawing
- Network Comparison

## Format of **.per** file



```
*Vertices 5
2
1
4
3
5
```

Here, the vertex 2(B) is first in our permutation.

## Performing BFS

Since BFS is merely a breadth wise reordering of the nodes, the functionality comes under the permutation section. To perform BFS on a graph, the following

steps should be performed

- Go to 'Network' menu.
- Under 'Create Permutation', select the breath first option.
- Choose from strong and weak based on your requirements.

## Performing DFS

To perform DFS on a graph, the following steps should be performed

- Go to 'Network' menu.
- Under 'Create Permutation', select the depth first option.
- Choose from strong and weak based on your requirements.

## Limitations of Pajek

Since it was made back in 1996, Pajek has a few limitations

1. Overwhelming UI for beginners.
2. Performance issues for large networks.
3. Limited Export features and analysis features.
4. Non-programmable.
5. Limited community support and less frequent updates.