NetObs: a software for studying multivariate networks

What is NetObs

NetObs is a software for the visualization of networks that involve many variables.

NetObs only works with **numeric** variables, accepting integers and decimals. It does not matter whether these variables are ordinal, cardinal, categorical, etc. NetObs will *interpret* these numbers as *measures*. So, these variables should say whether there is more or less of *something*. The meaning of each supplied variable is up to the user.

The rationale behind NetObs

As we had complex network data, we needed methodologies and tools to study it. One idea was the use of visualization to simplify the analysis. But how can we use visualization to do this? Traditional network visualization techniques give emphasis to topological properties of networks, displaying them clearly (avoiding edge intersection, for example), showing communities, isolating clusters, etc. As this problem is well studied, and it does not take into account the nature of data, our focus had to be different.

How can we use visualization to study complex networks? Two ideas came to mind:

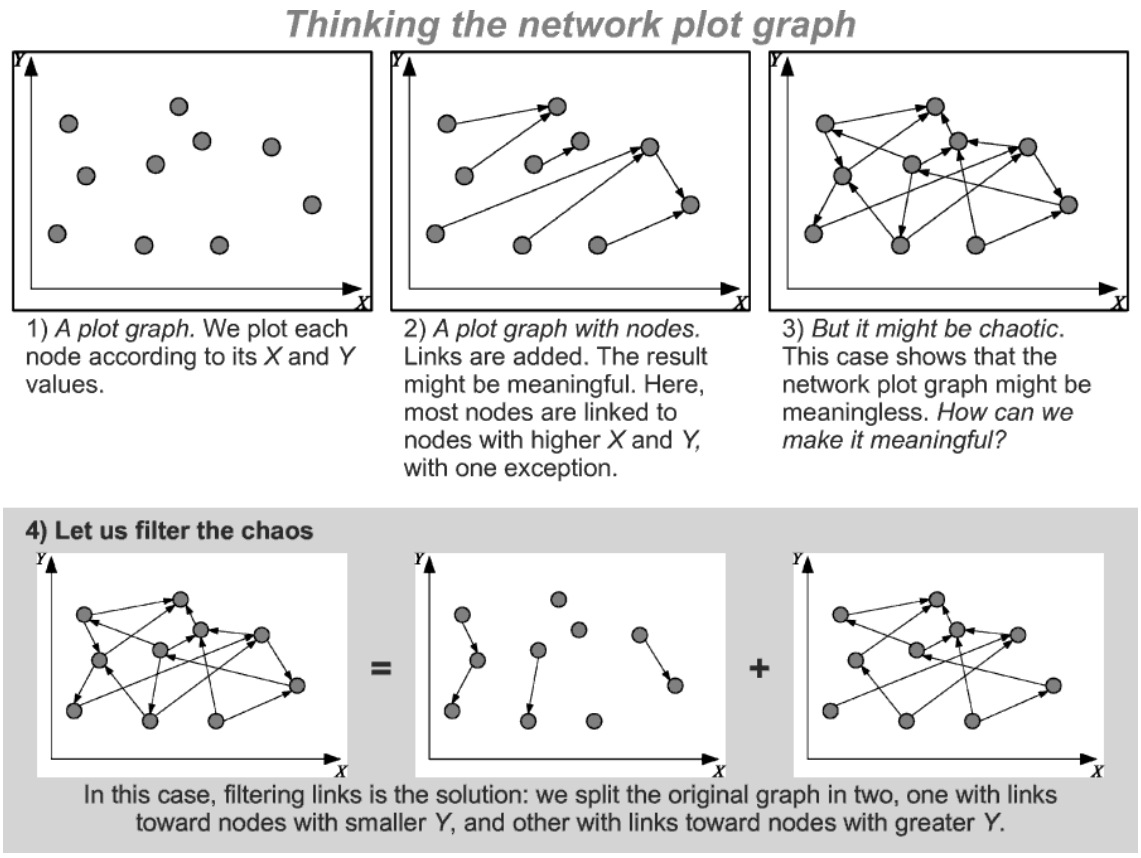
- **Plot graphs**. Plot graphs are useful for discovering the relationship between two variables. But in this case, links represent the relationship between nodes.
- **Bihistograms**. Histograms show univariate probability density or distribution functions. Bihistograms are the same but for bivariate functions. In this case, links are the relationship between two nodes.

The adaptation of these ideas, as both are implemented in NetObs, is presented in the following subsections.

Network plot graphs

The idea behind network plot graphs is the plotting of nodes as points in a plot graph. This might help to discover some patterns behind links. In a social network, for example, actors might want to link to wealthier ones. Instead of studying a topology with anonymous actors, a researcher might want to *see* effectively if links start in poorer actors and end in the richer ones. Of course, for this sort of pattern, numerical variables are needed, and plotting nodes in order is essential.

The previous example shows how visualization can help *qualitative* research. Complex networks and complex systems put qualitative research aside. The problem with this is that it is hard to study the full complexity of a system without simplifying things a bit.



Infographic about the idea behind network plot graphs.

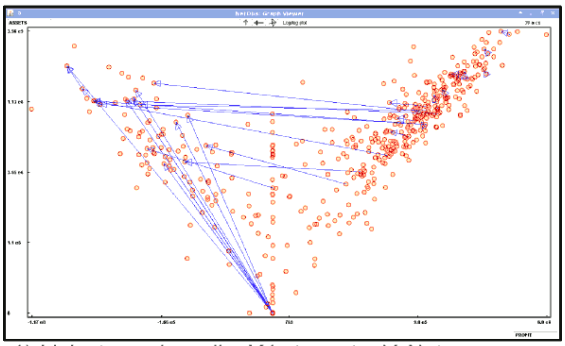
How do we draw a network plot graph? Figure ?? explains this quite well: we first draw the nodes, then draw the links, according to two axises. After this, interpretation should procede.

There is a possibility that network plot graphs are too chaotic to be understood by inspection. This problem is already studied in figure ??; the proposed solution was *filtering* links of a certain kind. In the case of the figure, links pointing upwards or downwards were filtered.

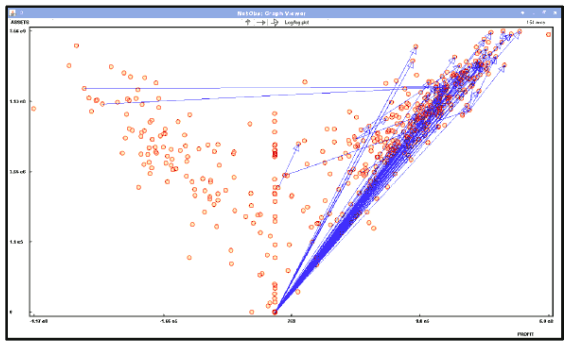
What kind of filters we might use? Many, really. But NetObs incorporates only a few main set of filters: less than, equal to, and greater than, for both axises. Including the “any” filter (or no filter) by axis, there are four filters by axis. This implies sixteen combinations of filters. In other words, sixteen subnetworks. It might sound like a lot of work, but this depends on the situation. Actually, one might tend to look for that *non-chaotic filtered network* that proves useful.

One way to cope with this complexity is looking at the number of links. Filtered graphs might contain very different amounts of links. This is a opportunity: having many links implies a very frequent pattern while few links make it easier to study patterns case by case. This is already shown in figure ??.

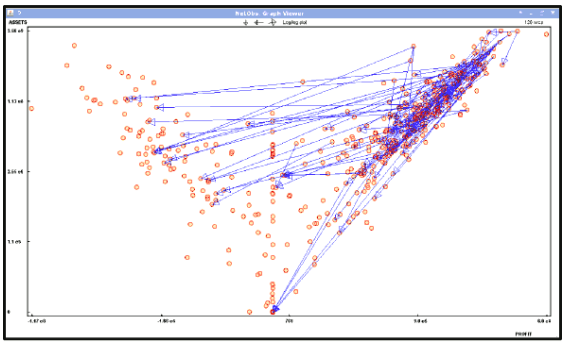
Filtering: plotting according to profits (X) and assets (Y)



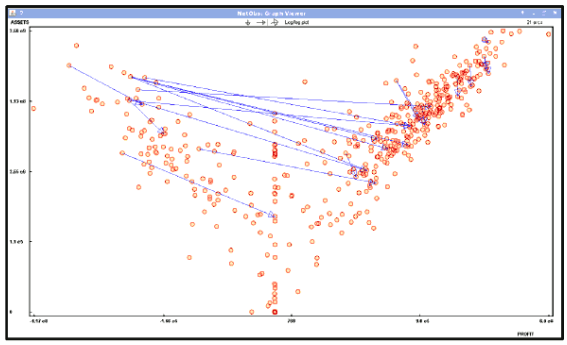
1) *Links toward smaller X but greater Y*. Not many links observed.



2) *Links toward greater X and Y*. Many links. Most of them start in nodes with no assets.



3) *Links toward smaller X and Y*. Many links. Most of them start in the upper right set of nodes.



4) *Links toward greater X but smaller Y*. Not many links.

Using filters with NetObs.

As seen in figure ??, filtering can simplify the analysis of complex graphs. Note the great number of nodes drawn in these pictures. Note how chaotic might be putting all these links in a single visualization. But these links can be easily handled when separated. In fact, these pictures can lead to a single conclusion: financial firms (with $assets = 0$) tend to invest in firms with big assets and big profits, but these big firms are less restrictive when it comes to investing.

Other issue is the way how nodes are plotted. For example, one might want to plot them as in a log/log plot instead of a traditional or metric one. This might be needed when nodes are distributed in inconvenient way. That is why NetObs was implemented with the following arrangement methods:

- *Metric plot*. For the traditional way of plotting.
- *Log/log plot*. When nodes are concentrated around small values. (When metric plotting fails.)
- *Strict-sort plot*. For multimodal distributions. (That is, when points are concentrated around some values.)
- *Sort plot*. Useful when there are many nodes with the same values. Now these nodes will not overlap.

Figure ?? shows why node arrangement is relevant. These nodes are plotted according a log/log fashion. A traditional arrangement would have put almost every node in the same place of the screen. Of course, links are understandable when their extremes are separated.

Link bihistograms

Network plot graphs were an aid for qualitative research. On the other hand, link bihistograms help quantitative research directly.

The idea behind link bihistograms is the presentation of link statistics (frequency). As links relate two nodes, one can build bihistograms of nodes but taking link information into account.

A link bihistogram might answer the following kind of questions:

- What is the likelihood that a node with $X : a \leq X \leq b$ will point a node with $Y : c \leq Y \leq d$?
- What is the expected outdegree of a node with $X : a \leq X \leq b$ linked to a node with $Y : c \leq Y \leq d$?
- How many links are present between nodes with $X : a \leq X \leq b$ pointing nodes with $Y : c \leq Y \leq d$?

Previous questions are very precise. In particular, knowing the answers is equivalent to knowing the distribution of links.

How do we build a link bihistogram? First, we should partition our nodes according to the desired attributes, X and Y, just like a traditional bihistogram. This will define a partition. Then, count the links in each partition, from the subset of X to the subset of Y, and create a statistic out of it. Then, you can visualize any of the following:

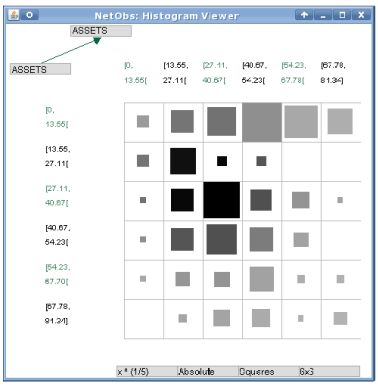
- The absolute amount of links in a subset.
- The relative amount of links, or *density*. This value is the likelihood of linking. It can be used to create the density or distribution function $f(x, y)$ that states the probability a node with attribute $X = x$ will point to a node with $Y = y$. We are very interested in this function as it characterizes the linking of a network.
- The average outdegree.
- The average indegree.

We also should know how solid is our information. For example, we might survey 1000 people: 990 commoners and 10 politicians. Saying 50% of commoners and 50% of politicians are overweight is not the same. You might trust in a survey of 990 people. But you surely should not trust in a survey of 10. This is called *support*: the amount of evidence behind an observation. Of course, we should visualize support along with our statistics.

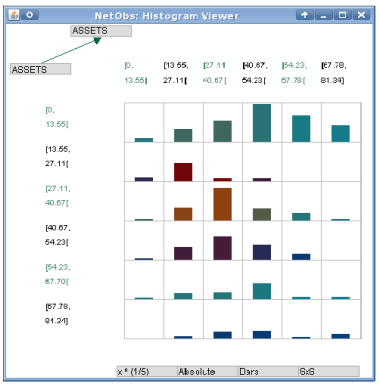
This is a practical symbolic example for all of the above. Let us split nodes according to two criteria: X_i and Y_j. Let us define $n_i = |X_i|$, $m_j = |Y_j|$, and l_{ij} the number of links from X_i to Y_j. Then:

- l_{ij} is the absolute amount of links from X_i to Y_j, by definition;
- $\frac{l_{ij}}{n_i \times m_j}$ is the density of links from X_i to Y_j;
- $\frac{l_{ij}}{n_i}$ is the average outdegree of X_i towards Y_j;
- $\frac{l_{ij}}{m_j}$ is the average indegree of Y_j from X_i; and
- $n_i \times m_j$ is the support of these statistics.

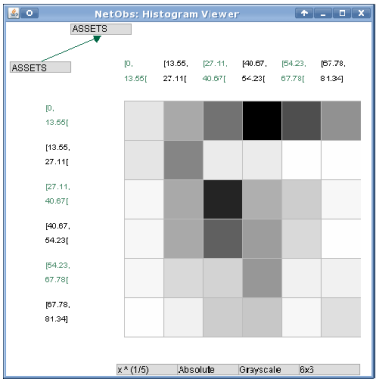
Link bihistogram



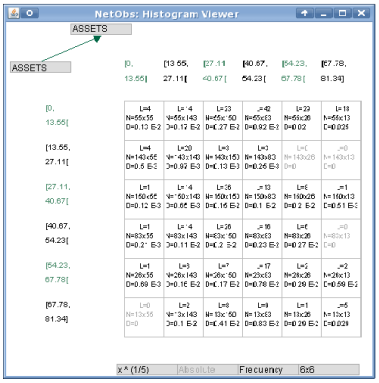
1) *Visualization as squares.* Each size is proportional to the amount visualized. Darker squares have more *support* than lighter ones.



2) *Visualization as bars.* Like traditional histograms, the size of each bar is proportional to the amount visualized. Red bars have more support than the blue ones.



3) *Visualization as grayscales.* Darker blocks represent higher amounts of the studied.



4) *Complete information.* Each block has the number of links and nodes assigned according to the partition. It also displays the *density* of each subgraph.

Many visualization styles in the link bihistogram.

How do we visualize these statistics? Figure ?? shows the four ways NetObs can display these statistics: squares, which size is proportional to the statistics visualized; bars, which height is proportional to the statistics visualized; grayscale, where darker blocks represent higher values of the statistics; and a complete information matrix.

Additionally, it is possible that links are concentrated around certain values. This makes support does not distribute evenly among the partition. In order to minimize this problem, NetObs was equipped with the following transformations for the selected variables:

- $f(x)=x$, identity, the default scaling;
- $f(x)=\sigma(x)\log(1+|x|)$, a logarithmic transformation with sign (σ);
- $f(x)=\exp(x)$, an exponential transformation;
- $f(x)=x^{1/5}$, a power transformation, alternative to the logarithm (if it fails);
- $f(x)=x^{1/3}$, a power transformation, less extreme than $x^{1/5}$ and the logarithm;
- $f(x)=x^{3/5}$, a slight transformation; and
- $f(x)=x^{3}$, a power transformation, suitable when the exponential fails.

Also, the number of partitions is not fixed. Any of following is allowed: 6x6, 7x7, 8x8, ... , 16x16.

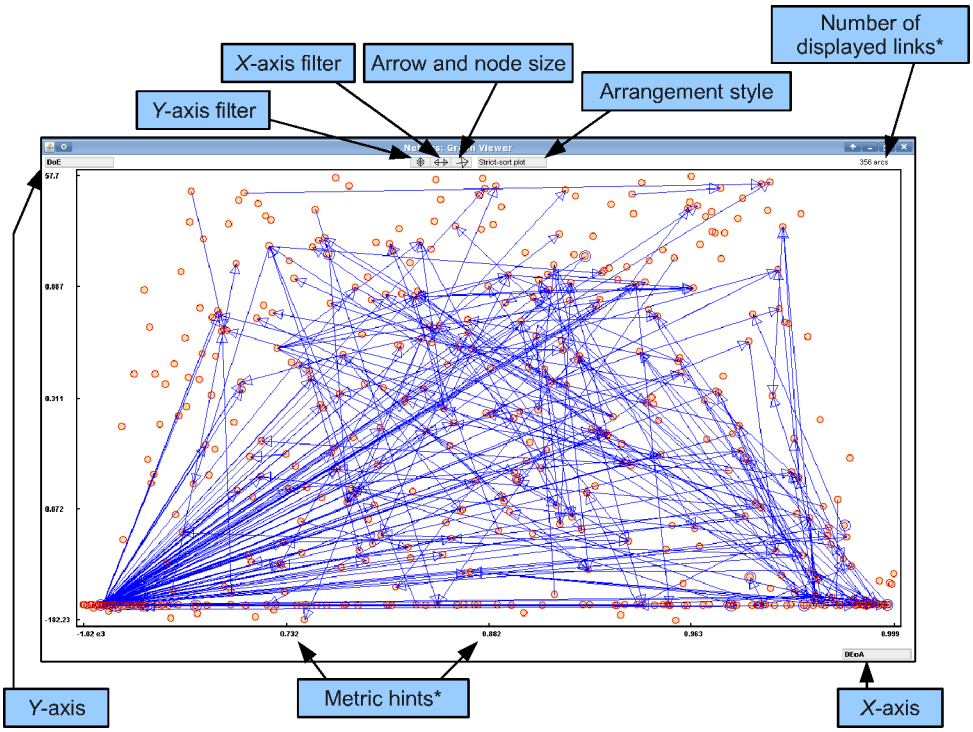
HCI issues

Functional interface

NetObs was designed for quick usage. As mentioned before, there are many ways to visualize the same data. And we did not take into account the number of supplied variables. To put some numbers:

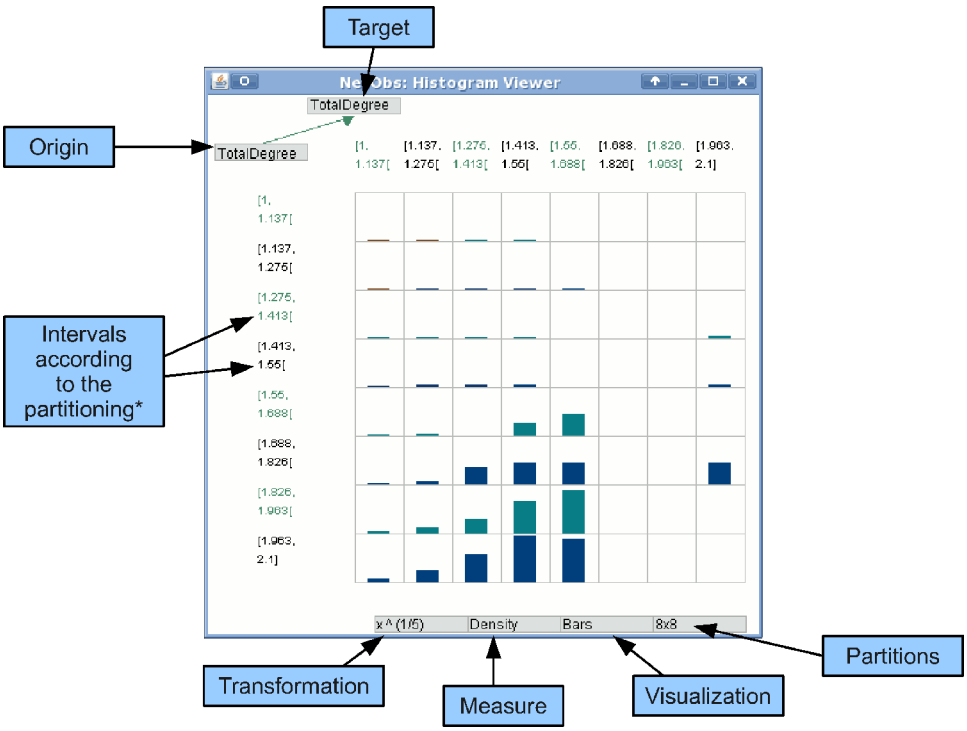
- *Network plot graph*. If we have 3 variables, the number of combinations of variables is 3 (discounting x vs y and y vs x); multiply that value by the combinations of filters (16) and the different arrangements of nodes (3). In total, there are 144 different ways of visualizing this data!
- *Link bihistogram*. If we have 3 variables, the number of combinations is 9 (the order is relevant); multiply that by the number of visualization styles (4), the number of partitions (11), and the number of transformations (7). In total, there are 2772 different ways of visualizing this data!!!

In order to simplify network analysis, NetObs has to answer user queries quickly. Options have to be selected in a dynamic fashion. Sadly, there is no time to select attributes from combo boxes nor filling forms. This is why NetObs has a *game-like* interface: there are not lists or anything. Instead, there are special buttons that react to clicks; right click selects the next alternative whereas left click select the previous one. This enables the user to explore hundreds of alternatives in minutes. (This requires getting used to the interface and quickly understanding whether a network is chaotic or understandable.)



User interface of the Network Plot Graph tool. (*Not clickeable.)

The network plot graph interface is shown in figure ??. Gray boxes with text are special buttons. Right clicking on them will change their setting; left clicking will change in the reverse way. Axises, where variables are selected, are placed in the upper left and lower right corners of the window. Filters and style buttons are centered in the upper part. Additionally, the number of displayed links is shown in the upper right corner.



User interface of the Link Bihistogram tool. (*Not clickeable.)

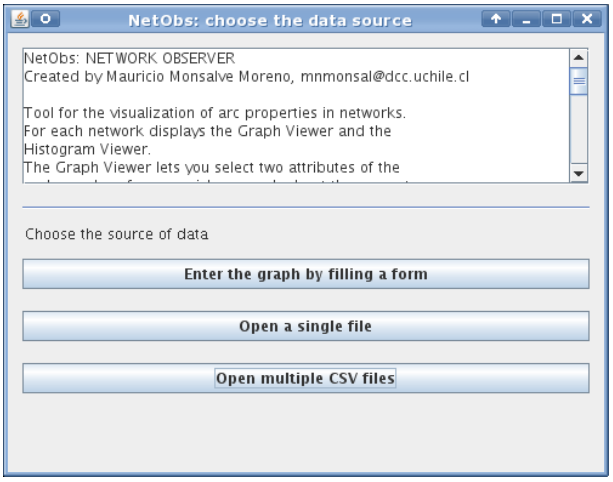
The link bihistogram interface is shown in figure ??.

Data input

When opened, NetObs asks for data. It accepts three different data sources:

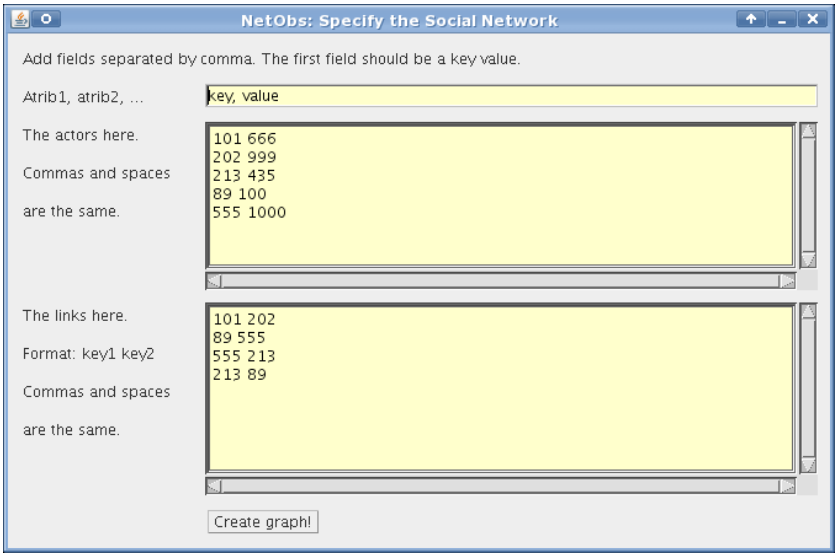
- Manual introduction.
- Using a single file.
- Using multiple CSV files: one for nodes, the other for links.

The last alternative is recommended. But the first one might come in handy when studying small networks.



First screen of NetObs.

The first option lets the user enter the data by filling a form. Figure ?? explain this quite well. But be careful: only numerical data is allowed.



Manual input.

(The second option)

The third option, using multiple CSV files, is recommended. CSV sheets are easy to handle. Software like Gnumeric, OpenOffice Calc, and Microsoft Excel, handle CSV files perfectly. Also, SQL queries can print CSV-like results when string concatenation is used.

The first column of the CSV sheet with node information should be the identification of the nodes. The next columns are not restricted to identification. For example:

ID,ASSETS,EQUITY,LT_DEBT,PROFIT

1,214804011,189957360,21626497,16791204

5,124446,23844,0,22781

6,3477922,113497,2252787,294679

7,9699659,3165350,5683128,512392

36,1555620,1441346,81692,-21940

The two columns of the CSV sheet with links are both identifications of the linked nodes. For example:

1,1119

1,3127

6,5

36,36

66,2203

Technical comments

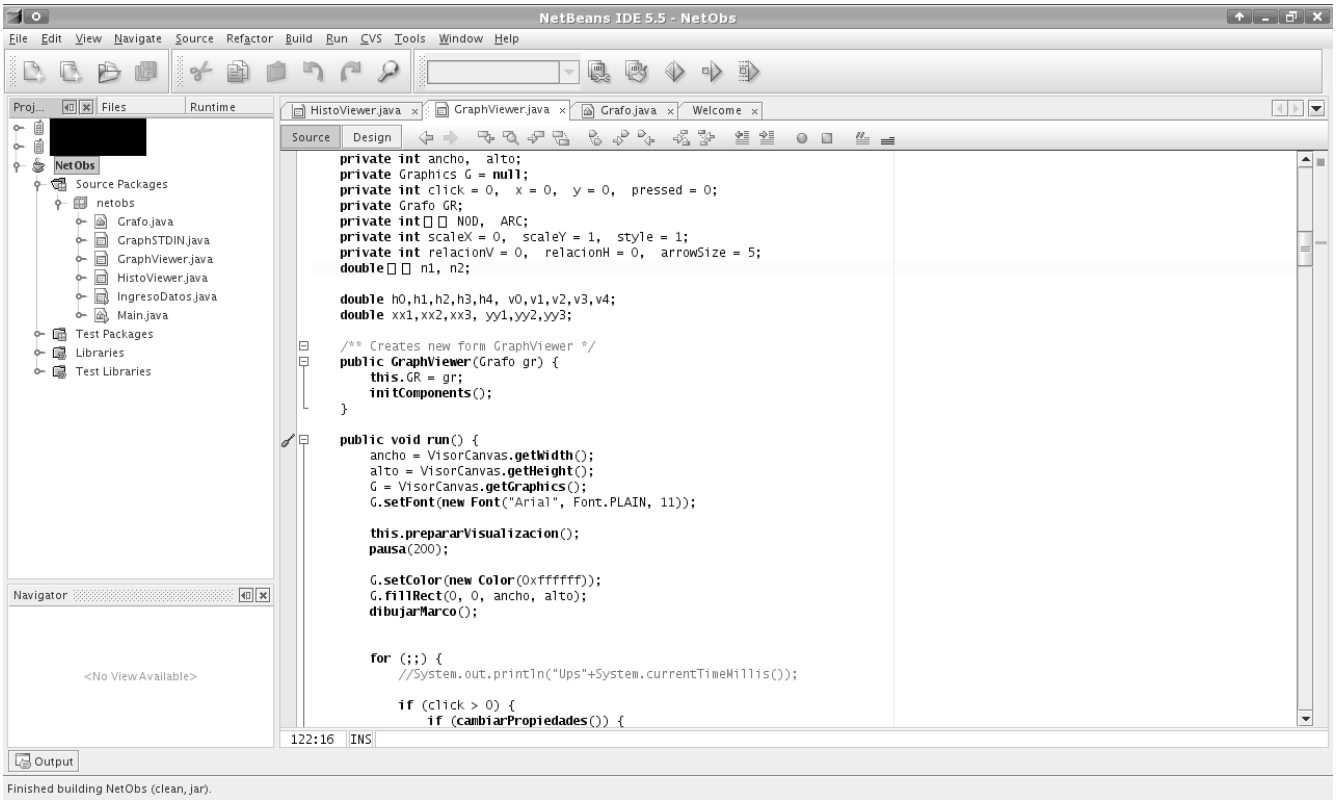
NetObs was developed in Java 1.5.0, but it should work since version 1.4.2. It has been tested on various machines, working in Windows and GNU Linux due to the compatibility of Java. It only

requires a JRE, *Java Runtime Environment*, in order to work. Nevertheless, some GNU Linux distributions include a GNU version of Java. If that version is recent enough, NetObs will work without problems.

Memory usage depends on the amount of data. Nodes and links are stored in separate arrays. Let us say we have n nodes with m variables. Then, the nodes array will consist on $n \times m$ *double*s. And for links, if we have k links, the link array will consist on $2k$ *double*s. Naturally, NetObs will use extra memory as it has to copy certain parts of these arrays. (Note that a *double* type is equivalent to 64 bits, according to IEEE 754.)

Development

NetObs was developed using NetBeans (http://www.netbeans.org), an open source IDE. It was developed in a fix-improve fashion, iteratively. This implies that this software has to be redesigned in order to make it extensible.



NetBeans IDE. NetObs appears as an active project.