Cytoscape 2.1

User Manual

Feb. 2005

The Cytoscape Collaboration

The **Cytoscape** project is an ongoing collaboration between:


University of California at San Diego

Institute for Systems Biology



Memorial Sloan-Kettering Cancer Center

Institut Pasteur


Funding for Cytoscape is provided by a federal grant from the U.S. National Institute of General Medical Sciences (NIGMS) of the National Institutes of Health (NIH) under award number GM070743-01. Corporate funding is provided through a contract from Unilever PLC.

Table of Contents

1. Introduction

Cytoscape is an open-source community software project for integrating biomolecular interaction networks with high-throughput expression data and other molecular states into a unified conceptual framework. Although applicable to any system of molecular components and interactions, Cytoscape is most powerful when used in conjunction with large databases of protein-protein, protein-DNA, and genetic interactions that are increasingly available for humans and model organisms. A software "Core" provides basic functionality to layout and query the network; to visually integrate the network with expression profiles, phenotypes, and other molecular states; and to link the network to databases of functional annotations. The Core is extensible through a straightforward plug-in architecture, allowing rapid development of additional computational analyses and features. **The central organizing metaphor of Cytoscape is a network graph, with genes, proteins, and molecules represented as nodes and interactions represented as links, i.e. edges, between nodes.**

Development

Cytoscape is a collaborative project between the Institute for Systems Biology (Dr. Hamid Bolouri), the University of California San Diego (Dr. Trey Ideker), Memorial Sloan-Kettering Cancer Center (Dr. Chris Sander) and the Institut Pasteur (Dr. Benno Schwikowski).

Visit http://www.cytoscape.org for more information.

License

Cytoscape is protected under the GNU LGPL (Lesser General Public License). The License is included as an appendix to this manual, but can also be found online: http://www.gnu.org/copyleft/lesser.txt Cytoscape also includes a number of other open source libraries, which are detailed in Section 10, Acknowledgements below.

2. Launching Cytoscape

Currently, Cytoscape runs under Java on Linux, Windows, and Mac OS X. Although Cytoscape handles arbitrary types and sizes of interaction network, it is most powerful when used in conjunction with large interaction data sets such as are currently available for species such as *Saccharomyces cerevisiae* (budding yeast).

System requirements:

The system requirements for Cytoscape depend on the size of the networks the user wants to load, view and manipulate. We recommend a recent computer (1GHz CPU or higher) with a high-end graphics card and at least 256MB of free physical RAM. Cytoscape expects a minimum screen resolution of 1024x768.

(1) Download and unpack the distribution. Cytoscape is distributed as a compressed archive (tar.gz or zip) containing the following files and directories:

`cytoscape.jar`	Main Cytoscape application (Java archive)
`cytoscape.props`	User-configurable properties and preferences
`vizmap.props`	User-configurable visual mapping preferences
`cytoscape.sh`	Script to run Cytoscape from command line (Linux, Mac OS X)
`cytoscape.bat`	Script to run Cytoscape (Windows)
`LICENSE.txt`	Cytoscape GNU License
`Cytoscape2_1Manual.pdf`	Cytoscape 2.1 Manual (the document you are reading now)
`sampleData/`	
`galFiltered.gml`	Sample molecular interaction network file *
`galFiltered.sif`	Identical network in Simple Interaction Format *
`galExpData.pvals`	Sample gene expression matrix file *
`BINDyeast.sif`	Network of all yeast protein-protein interactions in the BIND database as of July, 2004 **
`BINDhuman.sif`	Network of all human protein-protein interactions in the BIND database as of July, 2004 **
`yeastHighQuality.sif`	Sample molecular interaction network file ***
`annotation/`	Directory containing Gene Ontology database entries (currently for yeast only). Info in this directory is used to associate gene names with synonyms as well as process, function, and cellular location data.
`plugins/`	Directory containing cytoscape PlugIns, in .jar format.

* From Ideker et al, Science 292:929 (2001)
** Obtained from data hosted at http://www.blueprint.org/bind/bind_downloads.html
** From von Mering et al, Nature, 417:399 (2002) and Lee et al, Science 298:799 (2002)

(2) If necessary, install Java. If not already installed on your computer, download and install the Java 2 Runtime Environment, version 1.4.2 or higher. It can be found at:

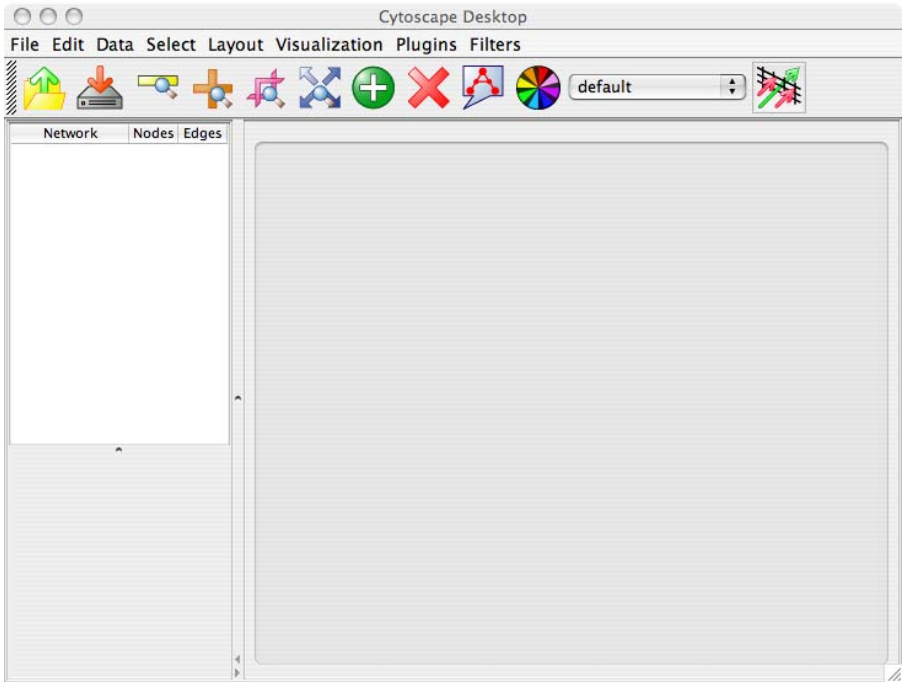
http://java.sun.com/j2se/1.4.2/download.html

(3) Launch the application by running "cytoscape.sh" from the command line (Linux or Mac OS X) or double-clicking "cytoscape.bat" (Windows). Alternatively, you can pass the .jar file to Java directly using the command "java -jar cytoscape.jar". In Windows, it is also possible to directly double-click the .jar file to launch it. However, this does not allow specification of command-line arguments (such as the location of the annotation data directory, see the section *4. Command Line Arguments* for details). On **Mac OS X**, users who downloaded the Mac OS X version of Cytoscape, can double-click on the Cytoscape icon to start Cytoscape. Either double-

clicking or dragging onto the Cytoscape application any .sif or .gml file will load that file into Cytoscape.

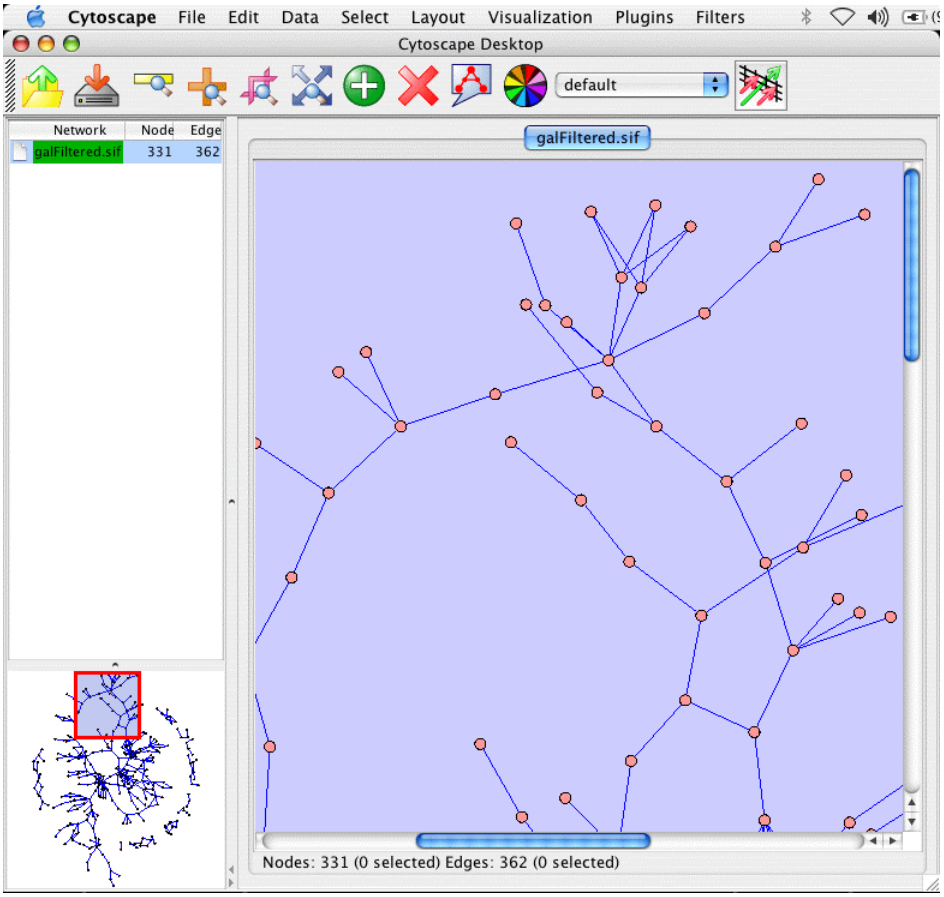
> **! Important Note:**
> For the application to work properly, ALL FILES MUST BE LEFT IN THE DIRECTORY IN WHICH THEY ARE UNPACKED. The core Cytoscape application assumes this directory structure when looking for certain files, such as cytoscape.props, vizmap.props, and the annotation/ database.

Cytoscape Window
When you succeed in launching Cytoscape, a window will appear that looks like this:



3. Quick Tour of Cytoscape

When a network is loaded, Cytoscape will look something like the image on the next page:

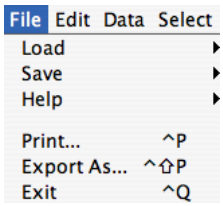


The main window has five components:

1. The menu bar at the top (See below for more information about each menu item).
2. The toolbar, which contains icons for commonly used functions. These functions are also available via the menus. Hover the mouse pointer over an icon and wait momentarily for a description to appear as a tooltip.
3. The network management window (top-left white box).
4. The overview window (bottom-left overview of the network).
5. The main network view window, which displays the network.

The Menus

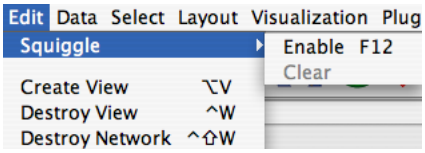
File



The File menu contains most basic file functionality: File / Load for loading a variety of file types; File / Save for saving. File/Help displays a credits screen. File / Print allows printing. File / Export As... allows you to export to a file in a number of graphics formats (such as postscript). File / Exit closes all windows of Cytoscape and exits the program.

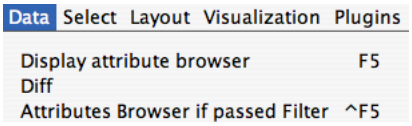
Edit

The Edit menu contains a Squiggle feature that enables you to mark up your network. This can be particularly useful during live presentations. There are also options for creating and destroying views (graphical representations of a network) and networks (the network data – not yet visualized).



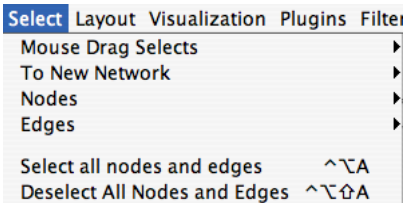
Data

The Data menu allows you to display the attribute browser, which lets you view attributes assigned to both nodes and edges. (See the section *7. Node and Edge Attributes*) Other options shown relate to plugins which are packaged with Cytoscape 2.1.



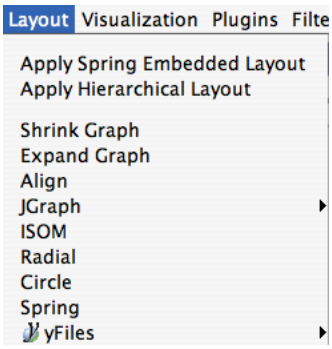
Select

The Select menu contains methods and operations for selecting nodes and edges, and using the current selection to create a new network and an associated view.



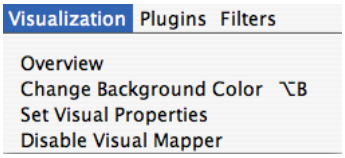
Layout

The Layout menu has an array of features for organizing the network visually according to one of several algorithms, aligning and rotating groups of nodes, and adjusting the size of the network. Most of these features are available from plugins that are packaged with Cytoscape 2.1



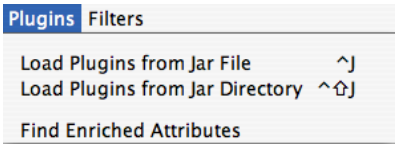
Visualization

The Visualization menu provides options for changing the mapping from biological data to a visual representation: colors of nodes, thickness of edges, etc. These features are explored in-depth in the *9. Visual* Styles section. This menu also provides an Overview (Bird's Eye view) of your entire network, which is helpful for navigating very large networks.



PlugIns

The PlugIns menu has choices for loading plugins individually, or a group of plugins located in a common directory. This menu may also contain choices added by plugins that have been loaded, such as "Find Enriched Attributes".



Filters

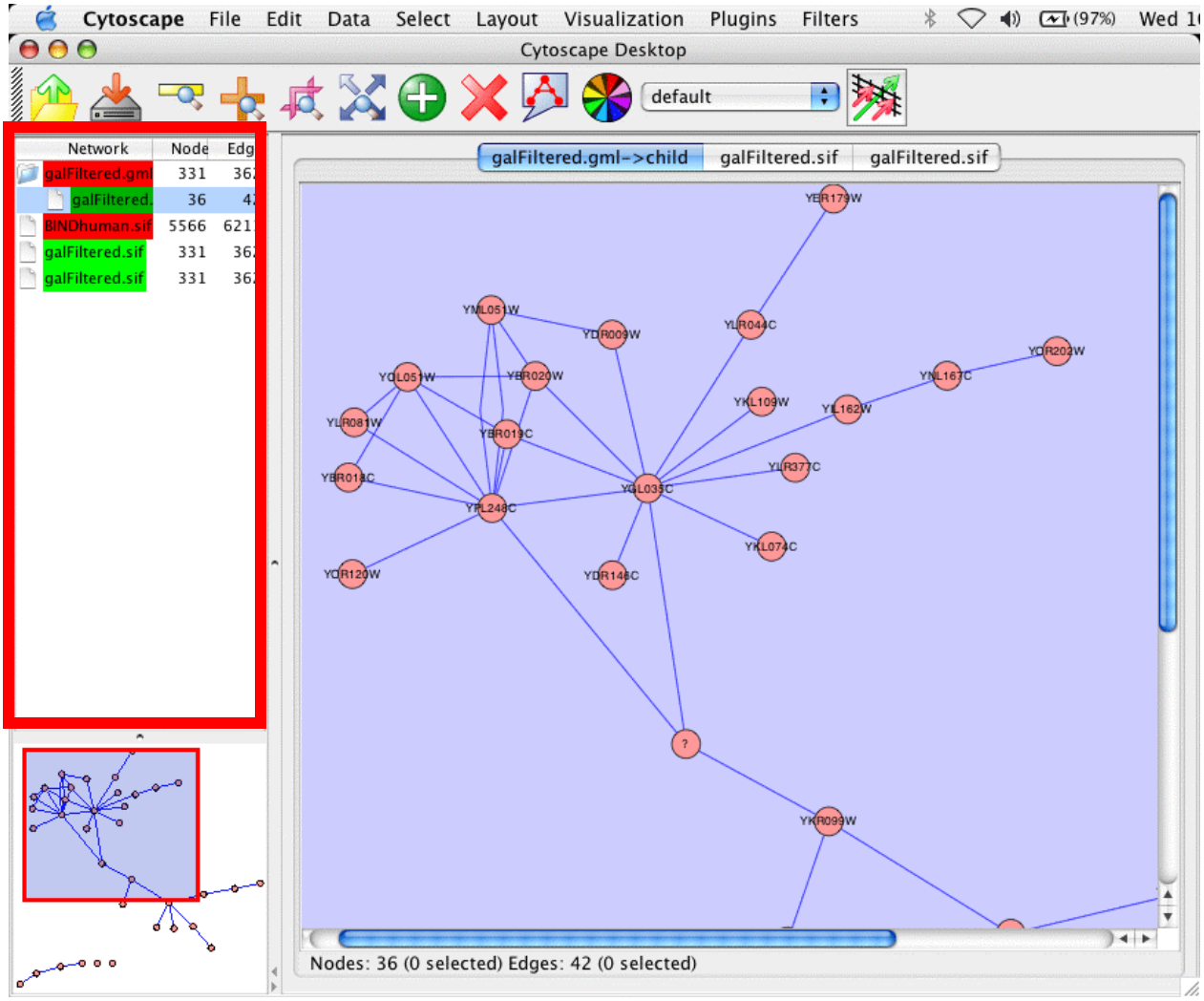
This menu has been added by the plugin in the file "filter.jar". The function of this plugin is not described further in this manual. Additional menus may appear, depending on the set of PlugIns you have chosen to load.

Note: A list of Cytoscape PlugIns with descriptions is available online at: http://cytoscape.org/plugins2.php

The Network Management Window

Cytoscape 2.1 allows multiple networks to be loaded at a time, either with or without a view. A network stores all the nodes and edges that are loaded by the user and a view displays them. You can have many views of the same network. Networks (and their optionally associated views) can be organized hierarchically.

An example where a number of networks have been loaded and arranged hierarchically is shown below:



The network manager (marked by the red square) shows the networks that are loaded. Clicking on a network here will make that view active in the main window, if the view exists (green highlighted networks only). Each network has a name and size (number of nodes and edges), which are shown in the network manager. If a network is loaded from a file, the network name is the name of the file.

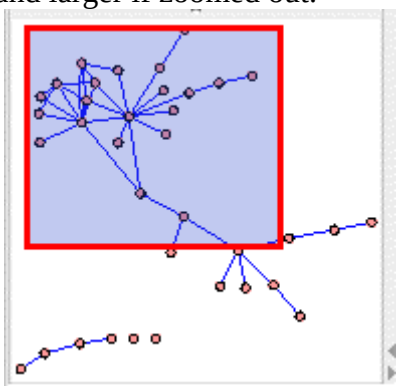
Since some networks are very large (thousands of nodes and edges) and can take a long time to display. For this reason, a network in Cytoscape may not contain a 'view'. Networks that have a view are highlighted in green and networks that don't have a view are highlighted in red. You can create or destroy a view for a network by right-clicking the network name in the network manager or by choosing the appropriate option in the edit menu. You can also destroy previously loaded networks this way. In the picture above, five networks are loaded, three green ones with views and two red ones without views.

Certain operations in Cytoscape will create new networks. If a new network is created from an old network, for example by selecting a set of nodes in one network and copying these nodes to a new network (via the Select->To New Network option), it will be shown as a child of the network that it was derived from. In this way, the relationships between networks that are loaded in Cytoscape can be seen at a glance.

The available network views are also shown as tabs on the top of the view window. You can click on the tab to go to the named network and the network manager will update accordingly. **Advanced users**: Cytoscape also has two viewing modes that alter the way in which these windows are displayed. This mode can only be selected on startup of the program by adding the –t option on the command line (see section *4. Command Line Arguments*).

The Network Overview Window

The network overview window shows an overview (or 'bird's eye view') of the network. It can be used to navigate around a large network view. This feature can be turned on an off via the Visualization menu. The red-outlined blue rectangle in the overview window shown below can be dragged with the mouse to navigate to a part of the network. The size of the navigation rectangle depends on the size of the active view and the zoom level of the view. The rectangle is smaller if the view is zoomed in and larger if zoomed out.



4. Command Line Arguments

Cytoscape recognizes a number of optional command line arguments, including run-time specification of network files and expression data:

-g | -graph <GML network filename> (xxx.gml)	Loads a network file in GML format (see *5. Building and Storing Interaction Networks*)
-i | -interaction <SIF interactions filename> (yyy.sif)	Loads a network file in SIF format (see *5. Building and Storing Interaction Networks*)
-b | -BDS <bioData directory>	(e.g. annotation/manifest) Specifies which directory to use for the BioDataServer annotations
-e | -expression <expression filename> (zzz.pvals)	Loads an expression data file (see *6. Loading Gene Expression Data*)
-n | -nodeAttributes <nodeAttributes filename> (one or more)	Loads node attributes files (see *7. Node and Edge Attributes*)
-j | -edgeAttributes <edgeAttributes filename> (one or more)	Loads edge attributes files (see *7. Node and Edge Attributes*)
-s | -species	Set the default species name
-c | -noCanonicalization	Turn off default node name canonicalization
-h | -help | -help | --help	Help: display these command line arguments
-p | -plugin | --JLD | --JLW | --JLL	Specify a list of plugins (jar files), directories containing plugins, URLs (http://) to jar files, or URLs to manifest files listing jar files
-props <properties file>	specify and load a properties file
-headless | -noView	Run in headless mode; do not create and display the GUI
-noDialog | -suppressView	Do not popup informational dialog when express file is loaded

-vt | --VT <view threshold>	Specify the threshold # of nodes at which views will not automatically be created
-project <project file>	Specify the location of the project file
-script | --script <script text…> -end	Specify script text
-rp | -resourcePlugin <resource plugins…>	Specify the list of resource plugins

Java System properties

-Djava.awt.headless=[true | false]	Similar to command line argument –headless | -noView; run in headless mode, do not create and display the GUI

Most data sets may also be loaded after Cytoscape is running. See the sections on *5. Building and Storing Interaction Networks, 6. Loading Gene Expression Data,* and *7. Node and Edge Attributes* for details.

Additional command line arguments that are not recognized by the Cytoscape core are passed to the PlugIn modules. Please refer to the documentation for each specific PlugIn for more details.

5. Building and Storing Interaction Networks

Cytoscape reads an interaction network in two ways: from a simple interaction file (SIF or .sif format) or from a standard format known as Graph Markup Language (GML or .gml format). SIF specifies nodes and interactions only, while GML stores additional information about network layout and allows network data exchange with a variety of other network display programs. Typically, SIF is used to import interactions when building a network for the first time, since it is easy to create in a text editor or spreadsheet. Once the interactions have been loaded and layout has been performed, the network may be saved to and subsequently reloaded from GML format in future Cytoscape sessions. Both SIF and GML are ASCII text files, and you can edit and view them in a regular text editor. Additionally, GML is supported by some other network visualization tools.

SIF FORMAT:

The simple interactions format is convenient for building a graph from a list of interactions. It also makes it easy to combine different interaction sets into a larger network, or add new interactions to an existing data set. The main disadvantage is that this format does not include

any layout information, forcing Cytoscape to re-compute a new layout of the network each time it is loaded.

Lines in the SIF file specify a source node, a relationship type (or edge type), and one or more target nodes:

```
nodeA <relationship type> nodeB
nodeC <relationship type> nodeA
nodeD <relationship type> nodeE nodeF nodeB
nodeG
...
nodeY <relationship type> nodeZ
```

A more specific example is:

```
node1 typeA node2
node2 typeB node3 node4 node5
node0
```

The first line identifies two nodes, called `node1` and `node2`, and a single relationship between `node1` and `node2` of type `typeA`. The second line specifies three new nodes, `node3`, `node4`, and `node5`; here "node2" refers to the same node as in the first line. The second line also specifies three relationships, all of type `typeB` and with `node2` as the source, with `node3`, `node4`, and `node5` as the targets, respectively. This second form is simply shorthand for specifying multiple relationships of the same type with the same source node. The third line indicates how to specify a node that has no relationships with other nodes. This form is not needed for nodes that do have relationships, since the specification of the relationship implicitly identifies the nodes as well.
Duplicate entries are allowed and indicate multiple edges between the same nodes. For example, the following specifies three edges between the same pair of nodes, two of type `pp` and one of type `pd`:

```
node1 pp node2
node1 pp node2
node1 pd node2
```

Edges connecting a node to itself (self-edges) are also allowed:

```
node1 pp node1
```

Every node and edge in Cytoscape has an identifying name, most commonly used with the node and edge data attribute structures. Node names must be unique as identically names nodes will be treated as identical nodes. The name of each node will be the name in this file by default (unless another string is mapped to display on the node using the visual mapper – see *9. Visual Styles*). The name of each edge will be formed from the name of the source and target nodes plus the interaction type: for example, `sourceName edgeType targetName`.

The tag `<interaction type>` should be one of:
`pp` ..... protein – protein interaction
`pd` ..... protein -> DNA
(e.g. transcription factor binding upstream of a regulating gene.)

Any text string will work, but the above are the conventions that have been followed thus far.

Additional interaction types are also possible, but not widely used, e.g.:

`pr` ..... protein -> reaction
`rc` ..... reaction -> compound
`cr` ..... compound -> reaction
`gl` ..... genetic lethal relationship
`pm` ..... protein-metabolite interaction
`mp` ..... metabolite-protein interaction

Even whole words or concatenated words may be used to define other types of relationships e.g.
`geneFusion, cogInference, pullsDown, activates, degrades, inactivates, inhibits, phosphorylates, upRegulates`

Delimiters. Whitespace (space or tab) is used to delimit the names in the simple interactions file format. However, in some cases spaces are desired in a node name or edge type. The standard is that, if the file contains any tab characters, then tabs are used to delimit the fields and spaces are considered part of the name. If the file contains no tabs, then any spaces are delimiters that separate names (and names cannot contain spaces).

If your network unexpectedly contains no edges and node names that look like edge names, it probably means your file contains a stray tab that's fooling the parser. On the other hand, if your network has nodes whose names are half of a full name, then you probably meant to use tabs to separate node names with spaces.

Networks in simple interactions format are often stored in files with a ".sif" extension, and Cytoscape recognizes this extension when browsing a directory for files of this type.

GML FORMAT:

In contrast to SIF, GML is a rich graph format language supported by many other network visualization packages. The GML file format specification is available at:

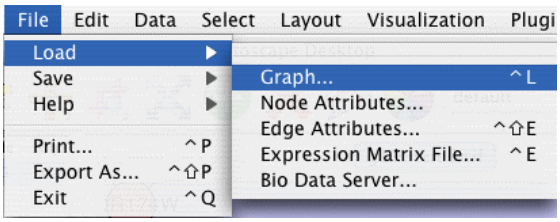
http://www.infosun.fmi.uni-passau.de/Graphlet/GML/

It is generally not necessary to modify the content of a GML file directly. Once a network is built in SIF format and then laid out, the layout is preserved by saving to and loading from GML. Colors and other visual attribute defined in the GML file are not currently honored by Cytoscape, only the node labels and layout information.

COMMANDS:
Load and save network files using the File menu of Cytoscape. Network files may also be loaded directly from the command line using the –i (SIF format) or -g (GML format) options.

FOR EXAMPLE:

To load a sample molecular interaction network in SIF format, use the menu File / Load / Graph. In the resulting file dialog box, select the file "sampleData/galFiltered.sif". After a few seconds, a small network of 331 nodes should appear in the main window. To load the same interaction network as a GML, use the menu: File / Load / Graph again. In the resulting file dialog box, select the file "sampleData/galFiltered.gml". Node and edge attribute files as well as expression data and extra annotation can be loaded as well.



NODE NAMING ISSUES IN CYTOSCAPE:

Typically, genes are represented by nodes, and interactions (or other biological relationships) are represented by edges between nodes. For compactness, a gene also represents its corresponding protein. Nodes may also be used to represent compounds and reactions (or anything else) instead of genes.

If a network of genes or proteins is to be integrated with Gene Ontology (GO) annotation or gene expression data, the gene names must exactly match the systematic ORF names specified in the other data files. We strongly encourage naming genes and proteins by their systematic ORF name or standard accession number; common names may be displayed on the screen for ease of interpretation, so long as these are available to the program in the annotation directory or in a node attribute file. Cytoscape ships with all yeast ORF-to-common name mappings in a synonym table within the annotation/ directory. Other organisms will be supported in the future.

Why do we recommend using standard gene names? All of the external data formats recognized by Cytoscape provide data associated with particular names of particular objects. For example, a network of protein-protein interactions would list the names of the proteins, and the attribute and expression data would likewise be indexed by the name of the object.

The problem is in connecting data from different data sources that don't necessarily use the same name for the same object. For example, genes are commonly referred to by different names, including a formal "location on the chromosome" identifier and one or more common names that are used by ordinary researchers when talking about that gene. Additionally, database identifiers from every database where the gene is stored may be used to refer to a gene (e.g. protein accession numbers from Swiss-Prot). If one data source uses the formal name while a different data source used a common name or identifier, then Cytoscape must figure out that these two different names really refer to the same biological entity.

Cytoscape has two strategies for dealing with this naming issue, one simple and one more complex. The simple strategy is to simply assume that every data source uses the same set of names for every object. If this is the case, then Cytoscape can easily connect all of the different data sources.

To handle data sources with different sets of names, as is usually the case when manually integrating gene information from different sources, Cytoscape needs a data server that provides

synonym information (See section *Appendix: Annotation Server Format*). A synonym table gives a canonical name for each object in a given organism and one or more recognized synonyms for that object. Note that the synonym table itself defines what set of names are the "canonical" names. For example, in budding yeast the ORF names are commonly used as the canonical names.

If a synonym server is available, then by default Cytoscape will convert every name that appears in a data file to the associated canonical name. Unrecognized names will be preserved. This conversion of names to a common set allows Cytoscape to connect the genes present in different data sources, even if they have different names – as long as those names are recognized by the synonym server.

For this to work, Cytoscape must also be provided with the species to which the objects belong, since the data server requires the species in order to uniquely identify the object referred to by a particular name. This is usually done in Cytoscape by specifying the species name on the command line with the –s option or by adding a line to the cytoscape.props file of the form:

```
defaultSpeciesName=Saccharomyces cerevisiae
```

The automatic canonicalization of names can be turned off with the -c command line argument (i.e. java -jar cytoscape.jar -c) or by not loading any annotation. This canonicalization of names currently does not apply to expression data. Expression data should use the same names as the other data sources or use the canonical names as defined by the synonym table.

6. Loading Gene Expression Data

Interaction networks are certainly useful as stand-alone models. However, they are most powerful for answering scientific questions when integrated with further information about the biology associated with the network, such as gene or protein expression levels. Once loaded, expression ratios/levels may be visually superimposed on the network, used in a filter to select a subset of nodes, or used to identify active modules and subsystems (via plugin analysis tools). An expression data set can be loaded at any time, but are only relevant once a network has been loaded.

FORMAT:
Gene expression ratios are specified over one or more experiments using a text file. The file consists of a header and a number of space- or tab-delimited fields, one line per gene, with the following format:

```
GeneName [CommonName] ratio1 ratio2 ... ratioN [pval1 pval2 ... pvalN]
```

Brackets [] indicate fields that are optional. The first two fields are the systematic gene name followed by an optional common name. Expression ratios are provided for each experiment, optionally followed by a p-value per experiment or other measure of the significance of each

ratio, i.e. whether the ratio represents a true change in expression (according to some statistical model.) Significance values are generated by a variety of software packages for analyzing expression data generated by DNA microarrays, for instance a program VERA from the Institute of Systems Biology (http://www.systemsbiology.org/VERAandSAM). A list of other microarray analysis packages is available at: http://www.nslij-genetics.org/microarray/soft.html

Example:

```
GENE DESCRIPT gal1RG.sig gal2RG.sig gal3RG.sig gal1RG.sig gal2RG.sig gal3RG.sig
YHR051W   COX6 -0.034 -0.052 0.152 1.177 0.102 0.857
YHR124W   NDT80 -0.090 -0.000 0.041 0.130 0.341 0.061
YKL181W   PRS1 -0.167 -0.063 -0.230 -0.233 0.143 0.089
```

The first line is a header line giving the names of the experimental conditions. Note that each condition is duplicated; the first set of columns gives expression ratios and the second set gives significance values. The significance columns can be omitted if your data doesn't include significance measures. Every remaining row specifies the values for a gene, starting with the formal name of the gene, then a common name, then the ratios, then the significance values.

Some variations on this basic format are recognized: see the formal file format specification below for more information. Expression data files commonly have the file extensions ".mrna" or ".pvals", and these file extensions are recognized by Cytoscape when browsing for data files.

COMMANDS:

Load an expression data file using the File menu of Cytoscape, or by specifying the filename using the -e option at the command line. Mac OS X users, who have downloaded the Mac OS X version of Cytoscape, can also drag SIF and GML file to the Cytoscape application to load those files. The –x command line option indicates that the expression data should not be loaded into node attributes. This is an advanced option, and is typically only used when the number of expression conditions is sufficiently large that it becomes unwieldy in the normal user interface.

FOR EXAMPLE:

Load a sample gene expression data set using the menu: File / Load / Expression Matrix File. In the resulting file dialog box (shown at right), select the file “sampleData/galExpData.pvals”. As described in the following sections, Cytoscape is now ready to integrate these data with the underlying molecular interaction network. **Advanced Note:** the checkbox in the lower left corner of the file dialog asks whether to “Copy Expression Data to Graph Attributes” – un-checking this box has the same effect is as the command line option -x, and it is left checked by default. If checked, this means that expression data values will be stored internally in Cytoscape in two places, once in an internal expression data object and once in node attributes. The advantage of also storing this information on node attributes is that the expression information can be easily visualized.

Detailed file format (Advanced users)

In all expression data files, any whitespace (spaces and/or tabs) is considered a delimiter between adjacent fields. Every line of text is either the header line or contains all the measurements for a particular gene. No name conversion is applied to expression data files (see the section on name resolution in section

5. Building and Storing Interaction Networks). The names given in the first column of the expression data file should match exactly the names used elsewhere (i.e. in SIF or GML files).

The first line is a header line with one of the following three formats:

```
<text> <text> cond1 cond2 ... cond1 cond2 ... [NumSigConds]
<text> <text> cond1 cond2 ...
<tab><tab>RATIOS<tab><tab>...LAMBDAS
```

The first format specifies that both expression ratios and significance values are included in the file. The first two text tokens are ignored; these columns will contain names for each gene. The next token set specifies the names of the experimental conditions; these columns will contain ratio values. This list of condition names must then be duplicated exactly, each spelled the same way and in the same order. Optionally, a final column with the title `NumSigConds` may be present. If present, this column will contain integer values indicating the number of conditions in which each gene had a statistically significant change according to some threshold.

The second format is similar to the first except that the duplicate column names are omitted, and there is no `NumSigConds` fields. This format specifies data with ratios but no significance values.

The third format specifies an MTX header, which is a commonly used format. Two tab characters precede the `RATIOS` token. This token is followed by a number of tabs equal to the number of conditions, followed by the `LAMBDAS` token. This format specifies both ratios and significance values.

Each line after the first is a data line with the following format:

```
FormalGeneName CommonGeneName ratio1 ratio2 ... [lambda1 lambda2 ...] [numSigConds]
```

The first two tokens are gene names. The names in the first column are the keys used for node name lookup; these names should be the same as the names used elsewhere in Cytoscape (i.e. in the SIF or GML files). Traditionally in the gene expression microarray community, who defined these file formats, the first token is expected to be the formal name of the gene (in systems where there is a formal naming scheme for genes), while the second is expected to be a synonym for the gene commonly used by biologists, although Cytoscape does not make use of the common name column. The next columns contain floating point values for the ratios, followed by columns with the significance values if specified by the header line. The final column, if specified by the header line, should contain an integer giving the number of significant conditions for that gene.
Missing values are not allowed and will confuse the parser. For example, using two consecutive tabs to indicate a missing value will not work; the parser will regard both tabs as a single delimiter and be unable to parse the line correctly.

Optionally, the last line of the file may be a special footer line with the following format:

```
NumSigGenes int1 int2 ...
```

This line specified the number of genes that were significantly differentially expressed in each condition. The first text token must be spelled exactly as shown; the rest of the line should contain one integer value for each experimental condition.

7. Node and Edge Attributes

Cytoscape allows the user to add arbitrary node and edge information to Cytoscape as node and edge **attributes**. Attributes could be, for example, annotation data on a gene or confidence values in a protein-protein interaction. These attributes can then be visualized in a custom user-defined way by setting up a mapping from data attributes to visual attributes (colors, shapes, etc.) (see section *9. Visual Styles*).

Node and edge attribute files are very simply formatted: A node attribute file begins with the name of the attribute on the first line, and on each following line, has the name of the node, followed by an equals sign, followed by the value of that attribute. Numbers and text strings are the most common attribute types. All values for a given attribute must have the same type. For example:

```
FunctionalCategory
YAL001C = metabolism
YAR002W = apoptosis
YBL007C = ribosome
```

An edge attribute file has much the same structure, except that the name of the edge is the source node name, followed by the interaction type in parentheses, followed by the target node name. Directionality counts, so switching the source and target will refer to a different (or perhaps non-existent) edge. The following is an example edge attributes file:

```
InteractionStrength
YAL001C (pp) YBR043W = 0.82
YMR022W (pd) YDL112C = 0.441
YDL112C (pd) YMR022W = 0.9013
```

Cytoscape treats edge attributes as directional, so note that the second and third edge attribute values refer to two different edges (source and target are reversed, though the nodes involved are the same).

Each attribute is stored in a separate file. Node and edge attribute files use the same format. Node attribute file names often use the suffix ".noa", while edge attribute file names use the suffix ".eda". Cytoscape recognizes these suffixes when browsing for attribute files.

Node and edge attributes may be loaded at the command line using the –n and –j options or via the File / Load menu.

When expression data is loaded using an expression matrix file (See *6. Loading Gene Expression Data*), it is automatically copied into the Node Attributes data structure unless explicitly specified not to.

Detailed file format (Advanced users)

Every attribute file has one header line that gives the name of the attribute, and optionally some additional meta-information about that attribute. The format is as follows:

```
attributeName class=formal.class.of.value category=attributeCategory
```

The first field is always the attribute name: it cannot contain spaces. The file may optionally include either of the class and category fields.

The category, if present, is saved and can be used by Cytoscape tools and plugins to group or filter the set of available attributes.

If present, the class field defines the formal (package qualified) name of the class of the attribute values. For example, `java.lang.String` for Strings, `java.lang.Double` for floating point values, `java.lang.Integer` for integer values, etc. If the value is actually a list of values, the class should be the type of the objects in the list. The value class must implement the Serializable interface (see the object serialization section of the Java tutorial), allowing the data to be saved in an efficient binary form (this binary attribute format is not directly supported by Cytoscape). If the class is not a basic String or Number class, it should have a String representation and a constructor that takes a String argument, allowing Cytoscape to construct an instance from the String representation in the file.

If no class is specified in the header line, Cytoscape will attempt to guess the type from the first value. If the first value contains numbers in a floating point format, Cytoscape will assume `java.lang.Double`; if the first value contains only numbers with no decimal point, Cytoscape will assume `java.lang.Integer`; otherwise Cytoscape will assume `java.lang.String`. Note that the first value can lead Cytoscape astray: for example,

```
floatingPointAttribute
firstName = 1
secondName = 2.5
```

In this case, the first value will make Cytoscape think the values should be integers, when in fact they should be floating point numbers. It's safest to explicitly specify the value type to prevent confusion.

Every line past the first line identifies the name of an object and the String representation of the attribute value. The delimiter is always an equals sign; whitespace (spaces and/or tabs) before and after the equals sign is ignored. This means that your names and values can contain whitespace, but object names cannot contain an equals sign and no guarantees are made concerning leading or trailing whitespace. Usually the object names should be the same as the names in your graph file, unless name conversion is being used (see the section on name resolution in section

5. Building and Storing Interaction Networks). Edge names are all of the form

```
sourceName (edgeType) targetName
```

Note that this format is different from the specification of interactions in the SIF file format. To be explicit: in a SIF file, an interaction looks like

```
sourceName edgeType targetName
```

To set an attribute for the edge defined by this interaction, the matching line in a attributes file should look like

```
sourceName (edgeType) targetName = value
```

(Yes, this is confusing; we're planning on fixing this in the next file format update for Cytoscape).

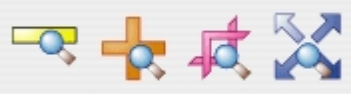
To specify lists of values, use the following syntax:

```
listAttributeName class=java.lang.String
firstObjectName = (firstValue::secondValue::thirdValue)
secondObjectName = (onlyOneValue)
```

This defines an attribute which is a list of Strings. The first object has three strings, and thus three elements in its list, while the second object has a list with only one member. In the case of a list every attribute value should be specified as a list, and every member of the list should be of the same class. Again, the class will be inferred if it is not specified in the header line. Lists are not supported by the visual mapper, so can't be mapped to visual attributes.

8. Navigation and Layout

BASIC FEATURES:
Use the zooming buttons located on the toolbar to zoom in / out of the interaction network shown in the current network display. Zoom icons are detailed below:



From Left to Right:

- Zoom Out
- Zoom In
- Zoom Selected Region
- Zoom out to Display all of Current Network

You can also zoom in/out by holding down the right mouse button and moving the mouse to the right (zoom in) or left (zoom out).

Use the left mouse button to select a node (hold down the Shift key to select more than one node). Use the right mouse button to launch a context sensitive menu with additional information about the node that was clicked on.

NETWORK LAYOUT:
To lay out your network using a Spring Embedded Layout, select Layout → Apply Spring Embedded Layout from the main menu. Sample screenshot is provided below:

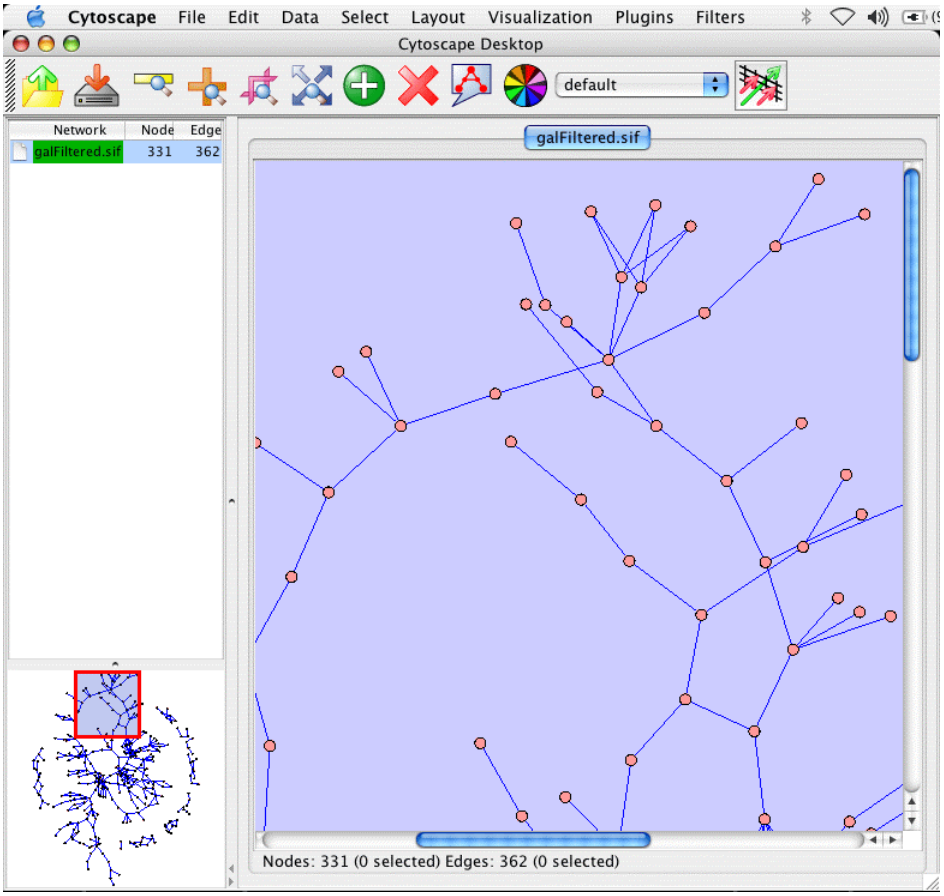


Figure: Applying the Spring Embedded Layout to a sample network.

9. Visual Styles

With the Cytoscape Visual Style feature, you can easily customize the visual appearance of your network. For example, you can specify a default color and shape for all nodes, use specific line types to indicate different types of interactions, or visualize gene expression data using a color gradient. All these features are available by selecting Visualization → Set Visual Properties from the main menu or clicking on the color wheel in the main button bar menu.

9.1 Introduction to Visual Styles

The Cytoscape distribution you have downloaded includes three predefined visual styles to get you started. To demonstrate these styles, try out the following example:

- Load a sample network: From the main menu, select File → Load → Graph, and select sampleData/galFiltered.sif.

- Load a sample set of expression data: From the main menu, select File → Load → Expression Matrix File, and select sampleData/galExpData.pvals.

By default, the Visual Style labeled "default" will be automatically applied to your network. This default style has a blue background, circular pink nodes, and blue edges (see sample screenshot below).

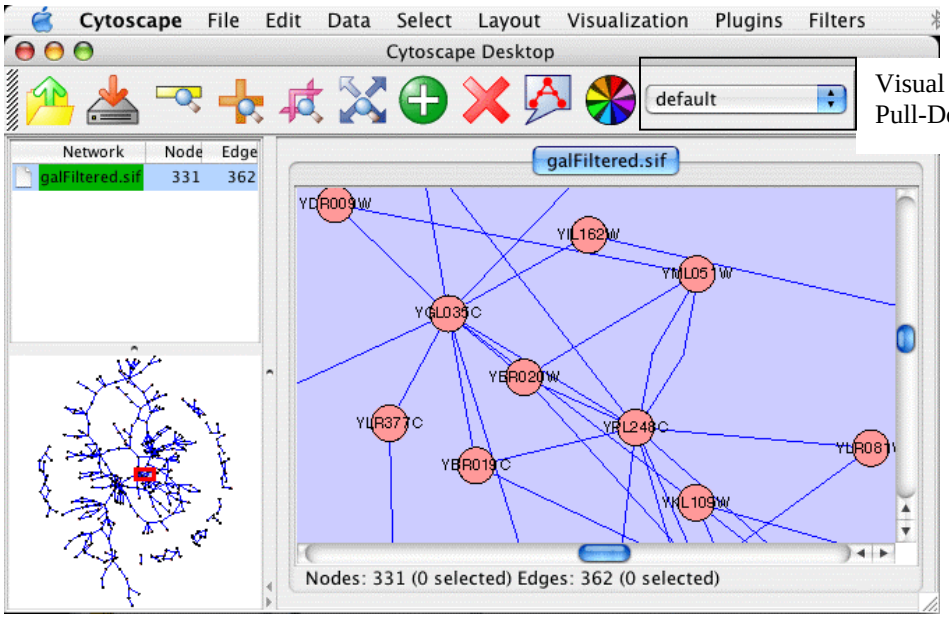


Figure: Using the default Visual Style.

The vizmap.props File: All Cytoscape Visual Style settings are automatically stored in a file called vizmap.props. Upon startup, Cytoscape will first try to locate the vizmap.props file in the "user home" directory. For example, on Windows XP, this corresponds to the user "Documents and Settings" directory, e.g. c:\Documents and Settings\cerami. On Linux or Mac OS X, this corresponds to the user home directory, e.g. /Users/cerami or ~. If no vizmap.props file is found in the user's home directory, Cytoscape will next search the current local directory. Note: vizmap.props is a text file that can be edited, but it is not recommended. If you do edit this file, make sure it is saved in text format and not that of any other editor.

[!] If you are upgrading from Cytoscape 1.1: If you are upgrading from Cytoscape 1.1, you may have an existing vizmap.props file in your home directory. If this is the case, you will not have the sample1 and sample2 visual styles described below. To get around this issue, backup your current vizmap.props file to safe place, and copy the new Cytoscape 2.1 vizmap.props file to your home directory.

You can flip through different visual styles by making a selection from the Visual Style pull down menu. For example, if you select "Sample1", a new visual style will be applied to your

network, and you will see a green background and round blue nodes. Additionally, protein-DNA interactions (specified with the label: pd) are drawn with dashed edges, whereas protein-protein interactions (specified with the label: pp) are drawn with drawn with a light yellow color which is difficult to discern on the green background (see sample screenshot below). The background can be changed through the Visualization menu.

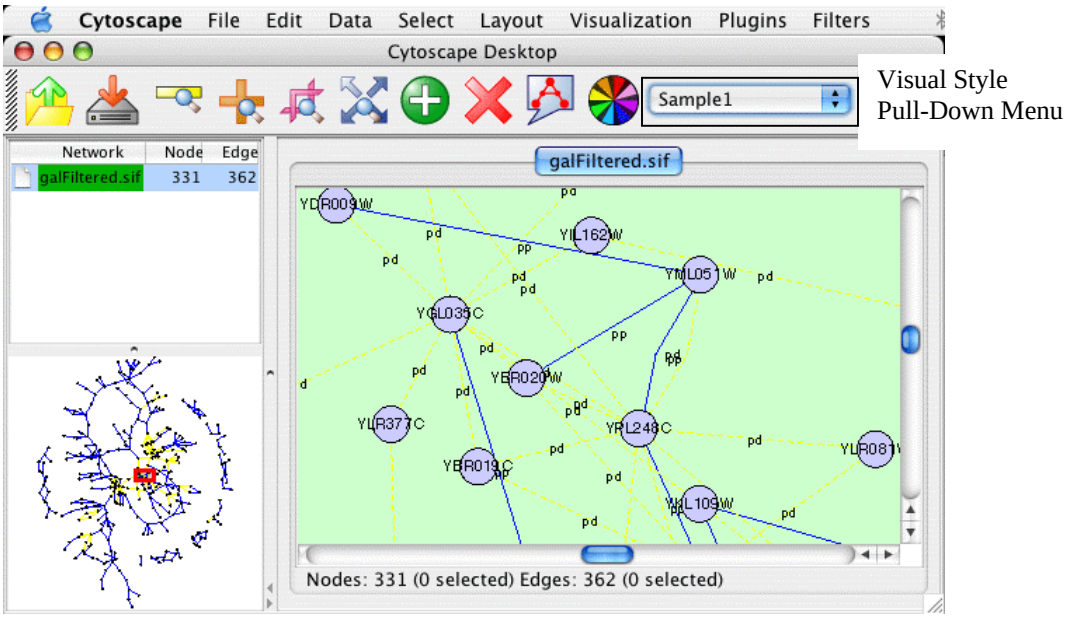


Figure: Using the Sample1 Visual Style. Protein-Protein interactions (solid lines) are now distinguishable from Protein-DNA interactions (dashed lines).

Finally, if you select "Sample2", gene expression values for each node will be colored along a color gradient between red and green (where red represents a low expression ratio, and green represents a high expression ratio - with thresholds set for the gal1RGexp experiment bundled with Cytoscape in the sampleData/galExpData.pvals file). See sample screenshot below:

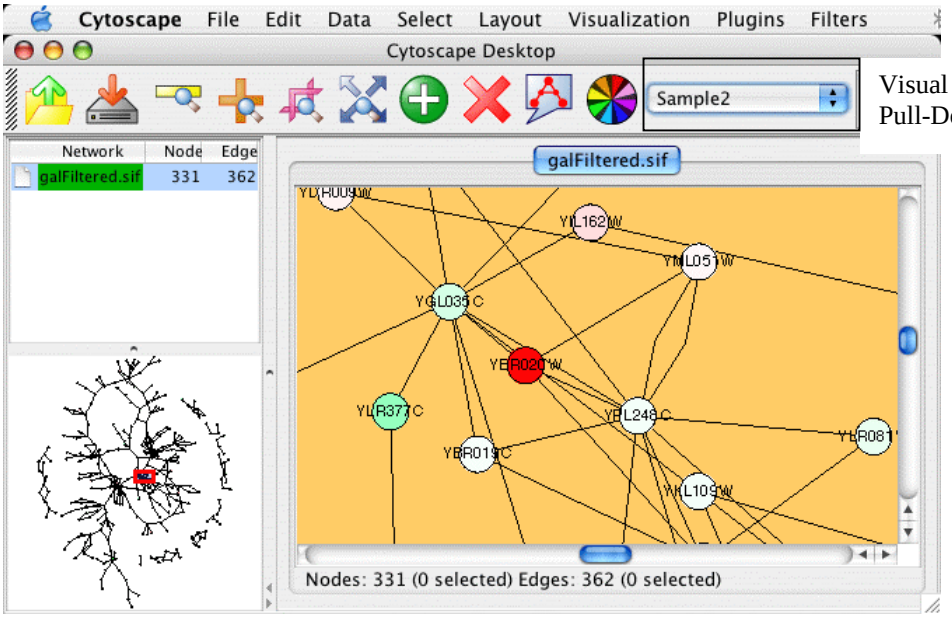


Figure: Using the Sample2 Visual Style. Gene expression values are now displayed along a red/green color gradient.

9.2 Visual Attributes, Graph Attributes and Visual Mappers

The Cytoscape Visual Mapper has three core components: *visual attributes*, *graph attributes* and *visual mappers*:

- A *visual attribute* is any visual setting that can be applied to your network. For example, you can change all nodes to squares by setting the node shape visual attribute.

- A *graph attribute* is any attribute associated with a node or an edge. For example, each edge in a network may be associated with a label, such as "pd" (protein-DNA interactions), or "pp" (protein-protein interactions).

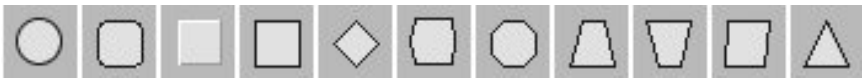
- A *visual mapper* maps graph attributes to visual attributes. For example, a visual mapper can map all protein-DNA interactions to the color blue, and all protein-protein interactions to the color red.

Cytoscape includes a large number of visual attributes. These are summarized in the tables below.

Visual Attributes Associated with Nodes:

- Node Color
- Node Border Color
- Node Border Type. The following options are available:

- Node Shape. The following options are available:

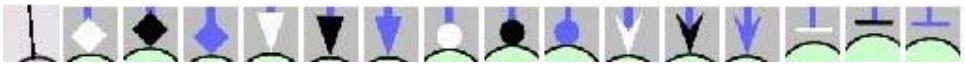


- Node Size: width and height of each node.
- Node Label: the text label for each node.
- Node Font: node font and size.

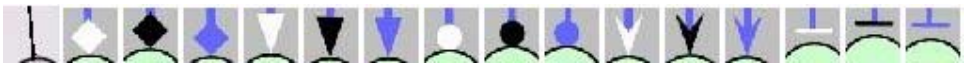
Visual Attributes Associated with Edges:

- Edge Color
- Edge Line Type. The following options are available:

- Edge Source Arrow. The following options are available:



- Edge Target Arrow. The following options are available:



- Edge Label: the text label for each edge.
- Edge Font: edge font and size.

Global Visual Properties:
• Background Color

For each visual attribute, you can specify a default value or define a visual mapping. Cytoscape currently supports three different types of visual mappers:

- **Passthrough Mapper:** graph attributes are passed directly through to visual attributes. A passthrough mapper only works for node / edge labels. For example, a passthrough mapper can draw the common gene name on all nodes.

- **Discrete Mapper:** discrete graph attributes are mapped to discrete visual attributes. For example, a discrete mapper can map all protein-protein interactions to the color blue.

- **Continuous Mapper**: continuous graph attributes are mapped to visual attributes. Depending on the visual attribute, there are two types of continuous mappers:

    - **continuous to continuous mapper**: for example, you can map a continuous value (0..1) to a color gradient (red..green) or node/font size (10..100).

    - **continuous to discrete mapper**: for example, all values below 0 are mapped to square nodes, and all values above 0 are mapped to circular nodes. However, there is no way to smoothly morph between circular nodes and square nodes.

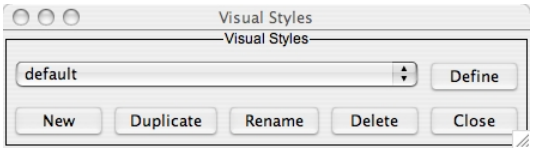
The matrix below shows visual mapper support for each visual property.

	Passthrough Mapper	Discrete Mapper	Continuous Mapper
Node Properties			
Node Color	⊘	λ	λ
Node Border Color	⊘	λ	λ
Node Border Type	⊘	λ	ω
Node Shape	⊘	λ	ω
Node Size	⊘	λ	λ
Node Label	λ	λ	ω
Node Font Family	⊘	λ	ω
Node Font Size	⊘	λ	λ
Edge Properties			
Edge Color	⊘	λ	λ
Edge Line Type	⊘	λ	ω
Edge Source Arrow	⊘	λ	ω
Edge Target Arrow	⊘	λ	ω
Edge Label	λ	λ	ω
Edge Font Family	⊘	λ	ω
Edge Font Size	⊘	λ	λ

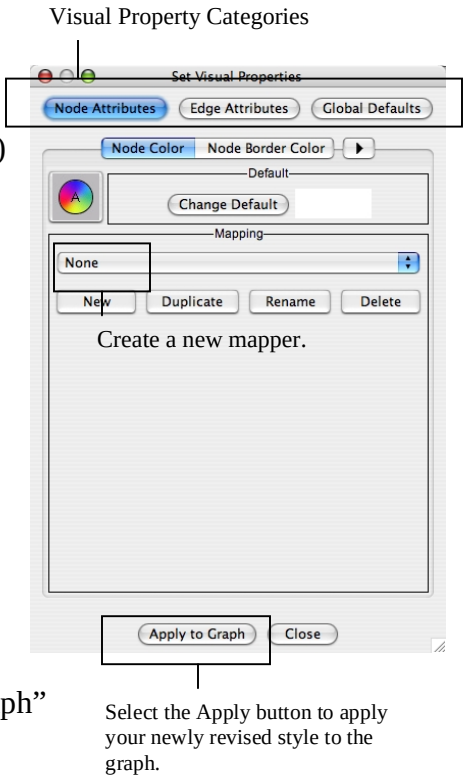
Legend	
⊘	Mapping is not supported for specified visual property.
λ	Mapping is fully supported for specified visual property.
ω	Mapping is partially supported for specified visual property. Support for "continuous to continuous" mapping is not supported.

9.3 Tutorial: Creating a New Visual Style

To create a new visual style, select Visualization → Set Visual Properties from the main menu, or select the color wheel icon in the main button bar. You will now see a new Visual Styles dialog box (shown at right.)



Click the New button, and enter a name for your new visual style when prompted. Then click the Define button. You will now see the main Visual Styles Properties dialog box (shown at right.)



From this dialog box, you can flip between Node Attributes, Edge Attributes, and Global Defaults. You can also specify default values for any visual property, or define a new custom mapping.

For example, to set the default node shape to triangles, select Node Attributes → Node Shape. Then, click the "Change Default" button, and select the Triangle icon from the selection list.

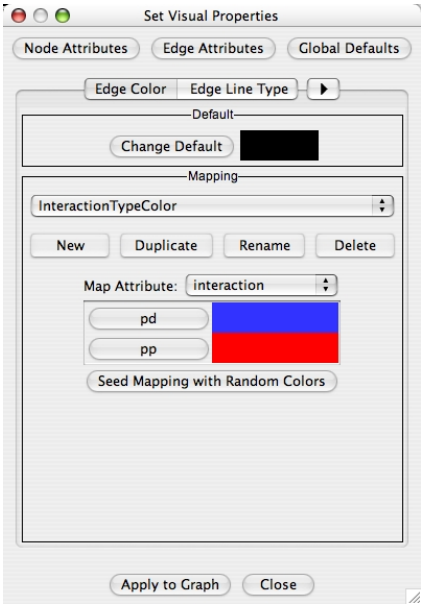
Applying Changes to the Network

To apply your visual style to your network, hit the "Apply to Graph" button, available in the bottom of the dialog panel.

Saving a Visual Style

When you exit Cytoscape, new visual styles or newly modified visual styles will automatically be saved in the vizmap.props file. You can therefore create a new visual style and apply it to all future networks.

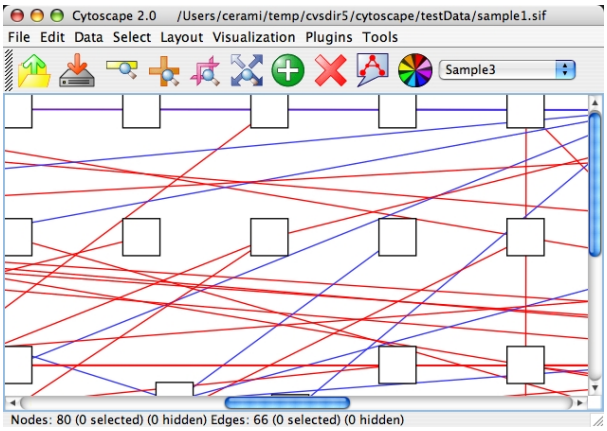
9.4 Tutorial: Creating a New Visual Style with a Discrete Mapper



The following tutorial demonstrates how to create a new visual style with a discrete mapper. The goal is to draw Protein-DNA interactions with blue edges, and Protein-Protein interactions with red edges.

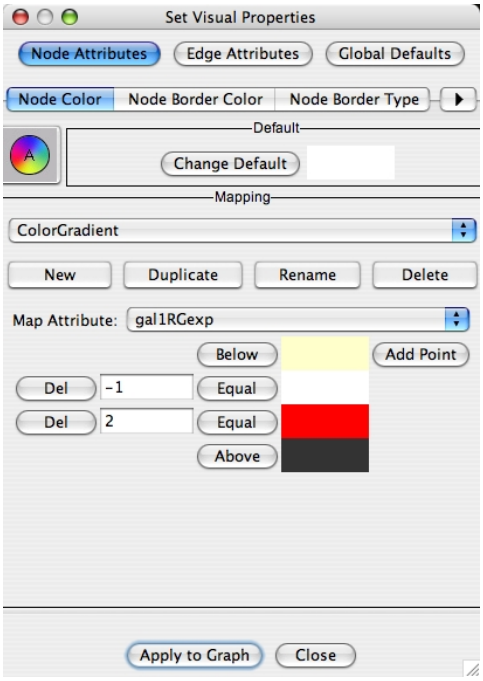
- Load a sample network: From the main menu, select File → Load → Graph, and select sampleData/galFiltered.sif.
- Select Visualization → Set Visual Properties.
- Select "New" to create a new Visual Style. Name your new style: "Sample3".
- Click the "Define" button to define the newly created Visual Style.
- In the "Set Visual Properties" Dialog box, select Edge Attributes → Edge Color.
- Click the New button in the mapping panel.
- You will be prompted to select a mapping type: passthrough mapper, discrete mapper or continuous mapper (for an overview of the differences between these mappers, please refer back to section 8.2.) Select "discrete mapper", and enter a descriptive name. For example, enter: "InteractionTypeColor".
- From the "Map Attribute" pull-down menu, select "interaction." You should now see two buttons, one for pd (Protein-DNA interactions), and one for pp (Protein-Protein interactions).
- Click the "pd" button and select a blue color.
- Click the "pp" button and select a red color.
- Click the "Apply to Graph" button.

You network should now show "pd" interactions in blue, and "pp" interactions in red. Sample screenshot is provided at right



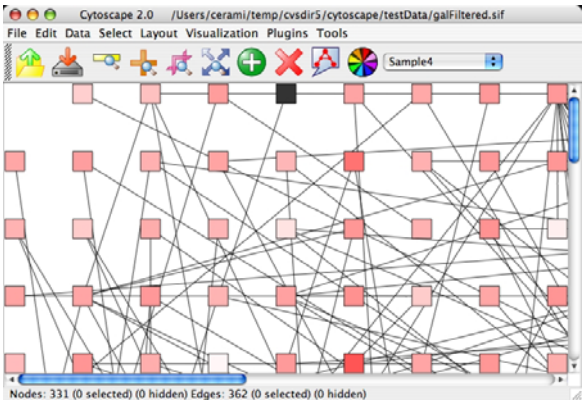
9.5 Tutorial: Visualizing Expression Data on a Network

The following tutorial demonstrates how to create a new continuous mapper. The goal is to superpose gene expression data onto a network, and to display gene expression values along a color gradient.



- Load a sample network: From the main menu, select File → Load → Graph, and select sampleData/galFiltered.sif.
- Load a sample set of expression data: From the main menu, select File → Load → Expression Matrix File, and select sampleData/galExpData.pvals.
- Select Visualization → Set Visual Properties.
- Select "New" to create a new Visual Style. Name your new style: "Sample4".
- Click the "Define" button to define the newly created Visual Style.
- In the "Set Visual Properties" Dialog box, select Node Attributes → Node Color.
- Click the New button in the mapping panel.
- You will be prompted to select a mapping type: passthrough mapper, discrete mapper or continuous mapper (for an overview of the differences between these mappers, please refer back to section 8.2.) Select "continuous mapper", and enter a descriptive name. For example, enter: "ColorGradient".
- From the "Map Attribute" pull-down menu, select "gal1RGexp."
- Click the "Add Point" button twice to add two data points.
- Set the first point to "-1", Below = Yellow, Equal = White.
- Set the second point to "2", Equal = Red, Above = Black.
- Click the "Apply to Graph" button.

This visual mapper will set all nodes with a gal1RGexp value less than –1 to Yellow, and all nodes with a gal1RGExp value greater than 2 to Black. Additionally, all values between –1 and 2 will be painted with a white/red color gradient. Sample screenshot is shown at right.

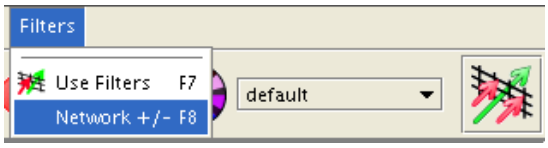


10. Filters

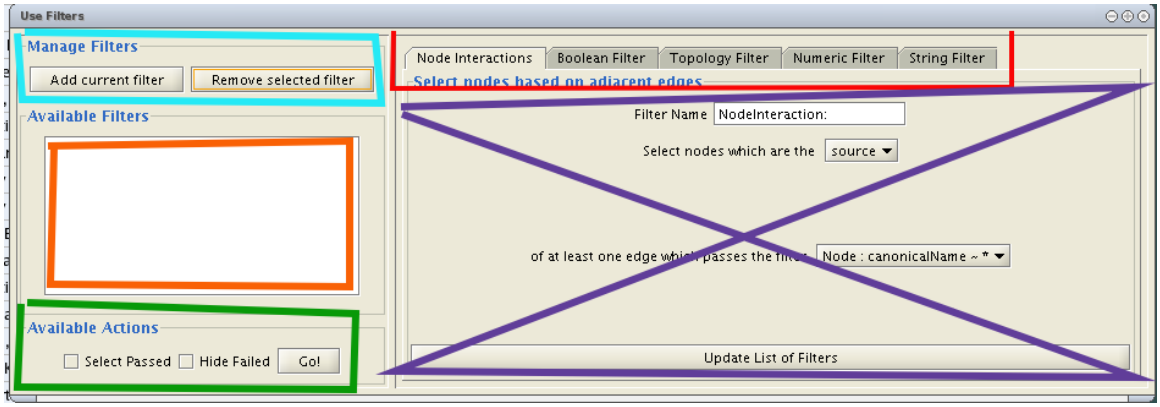
The Cytoscape Filter plugin, which is packaged with the official Cytoscape 2.1 release and is active by default, allows for a wide variety of fine-tuned filtering on node and edge attributes loaded onto Cytoscape networks. For example, you can easily select all the nodes whose name contains a specific pattern that you define. Example filters are shipped with the plugin to get started. Base filters only operate on String and Scalar data i.e. any names, descriptions or numerical node or edge attributes can be filtered. A Boolean filter is also available that can be used to combine any number of existing filters in logical combinations, so as you add filters, the plugin becomes more powerful.

Using the Plugin

If the Filter plugin is loaded, then you should see a menu called “Filters” and the filter icon:



Activate the filters using either the “Use Filters” menu entry or the large filters icon (red and green arrows). The filters dialog looks like the following (without the colors...)



As for the colors:

The Red Box: Each available filter has its own tab. The default filter is **Node Interactions** (which allows you to filter nodes, based on the edges that they are connected to), **Boolean** (which allows you to combine filters together using AND, OR and XOR operators), **Topology** (which allows you to filter nodes based on the number of edges to other nodes), **Numerical** (which allows for >, =, and < filtering operations on numerical attributes) and **String** (which allows for filtering using * and ? as wildcards).

The Purple Box: An existing or newly created filter can be edited in this area. Each filter type has its own user interface for editing.

The Orange Box: All available filters are shown in this list. Initially, this list will contain sample filters, but as you create more, they will be added here.

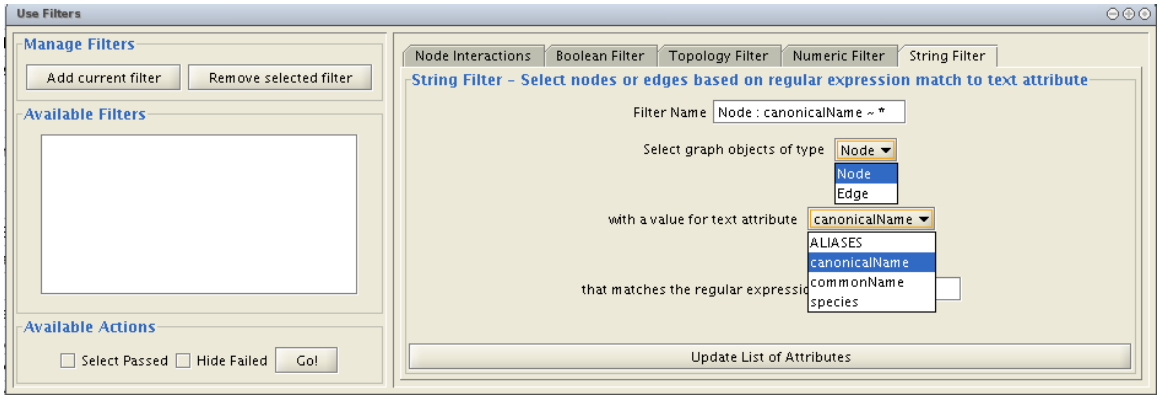
The Cyan Box: Pressing "Add Current Filter" adds the filter being edited to the "Available Filters" box, and "Remove Selected Filter" deletes the currently selected filter.

The Green Box: This area contains default actions for a given filter. These specify how Cytoscape should display the network components (nodes and edges) that pass a filter. You can choose to have Cytoscape select the nodes and/or edges that pass a given filter or alternatively have Cytoscape hide the nodes and/or edges that do not pass a given filter. A useful operation is to have a filter select a set of nodes and then send these nodes to a new network (through the **Select** menu).

Creating Filters

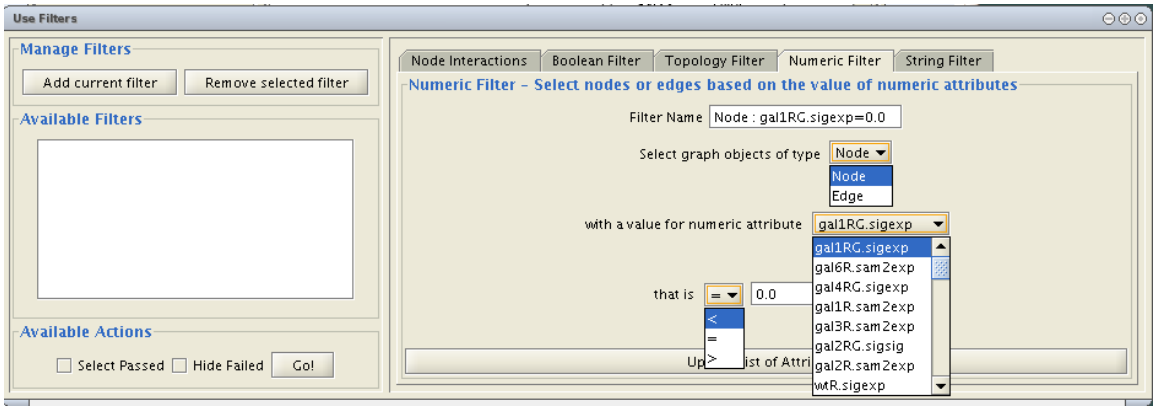
The important thing to realize when creating a filter is that the filter does not **do** anything by itself. Once created, the filter must be run.

String Filter:

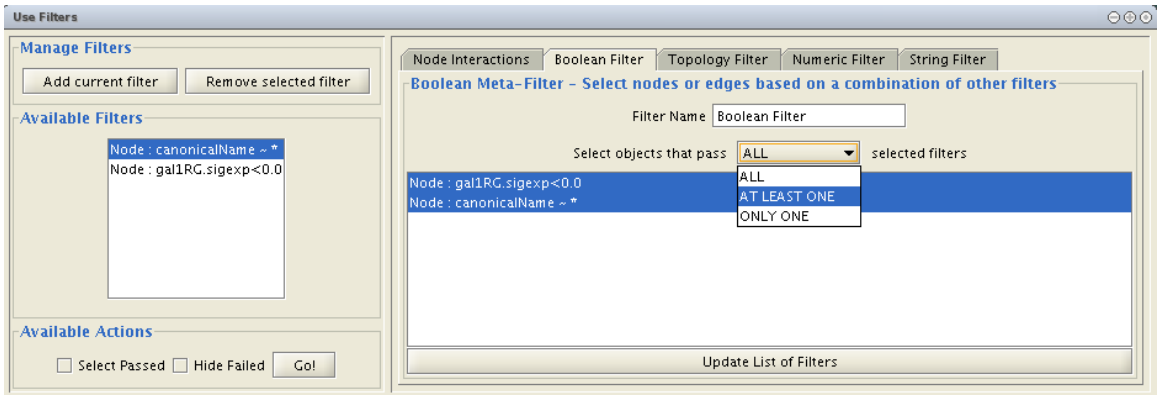


The String Filter allows you to choose to Filter Node or Edges, and gives you a list of available attributes to search for each (those attributes that are loaded on the network). If you have the filters dialog box active and you load new graph attributes, you can click the update button to refresh the attribute list. Search terms are entered in the text box at the bottom. For example to match any Node whose canonicalName starts with "YDL" you would select "Node", "canonicalName" and type "YDL*". The * is important as it matches anything for any number of characters after YDL. If you want to be more specific and only select Nodes whose canonicalName starts with YDL00 followed by any other two characters, you would type "YDL00??". The "?" denotes any single character, while the "*" represents zero or more characters. Full regular expression searching is supported, although is not covered here. Once the filter is defined, it will be assigned a default descriptive name, although this name can be edited. Pressing the "Add Current Filter" button will add your filter to the list of available filters to the left.

Numerical Filter:



The Numerical Filter also allows you to filter Nodes or Edges, and presents you with a list of available attributes. This filter matches greater-than, less-than, or equal-to a number you type in the search box. See the String filter description for more information.



Boolean Filter:

The Boolean filter allows you to define a new filter that is a logical combination of existing filters. Available filters are displayed (although the list can be refreshed by clicking the "Update List of Filters" button). By selecting one or more filters, you can then choose whether Nodes or Edges pass "ALL" (AND), "AT LEAST ONE" (OR), or "ONLY ONE" (XOR) of the selected filters. Once created Boolean filters can then themselves be combined using the Boolean filter to create arbitrarily complex logical combinations of filters. Note that unlike the String and Numerical Filters, Boolean Filters will need to be assigned a name manually.

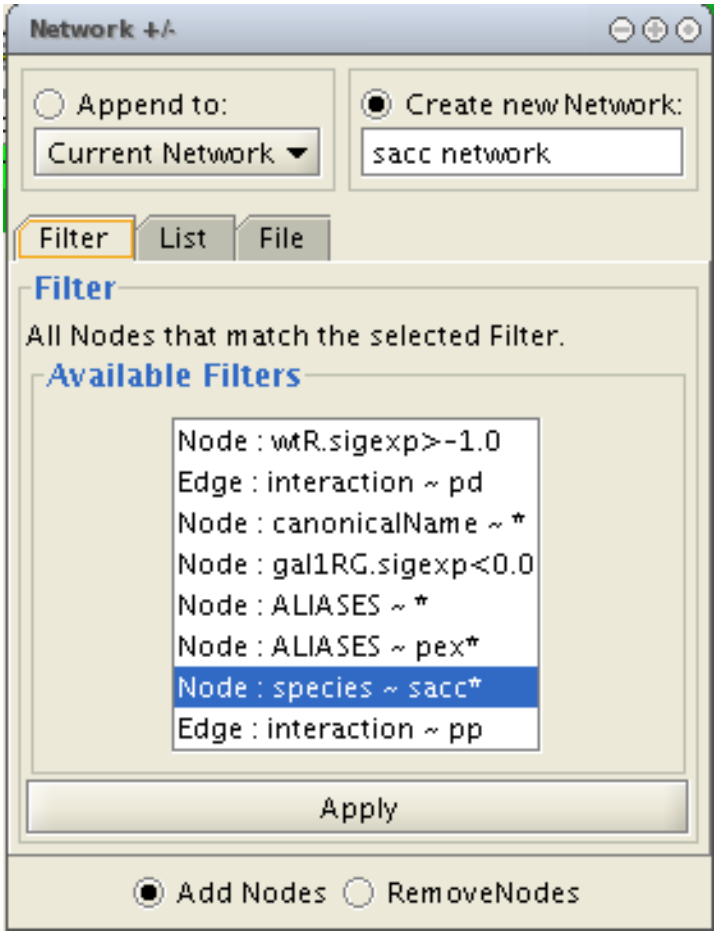
Once created, filters are saved for future sessions, as long as you exit Cytoscape normally via the exit command in the File menu (i.e. not via ctrl-c on Linux).

Running filters

Any available filter can be run by selecting a visualization action for Cytoscape (how your filter results should be displayed) and pressing the 'Go!' button.

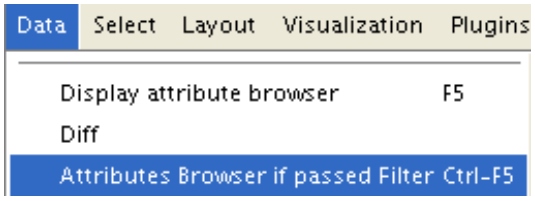
The Network +/- Filter Feature

The "Network +/-" dialog is available from the Filters menu. It allows you to create new networks and add/remove nodes to/from existing networks based on available filters. The dialog box is shown below:

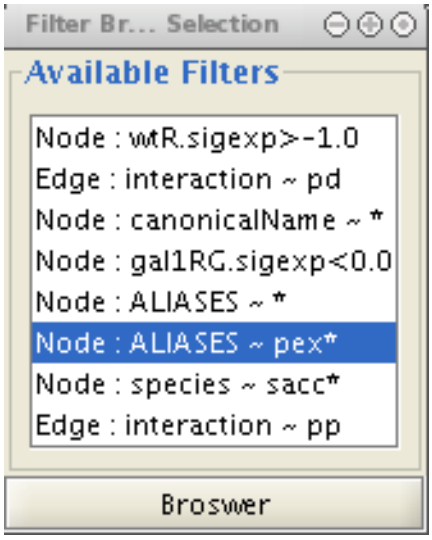


The Attribute Browser Filter Feature

The "Attributes Browser if passed Filter" item from the Data menu allows you to see the normal Cytoscape graph attribute browser for all nodes or edges that pass a given filter.



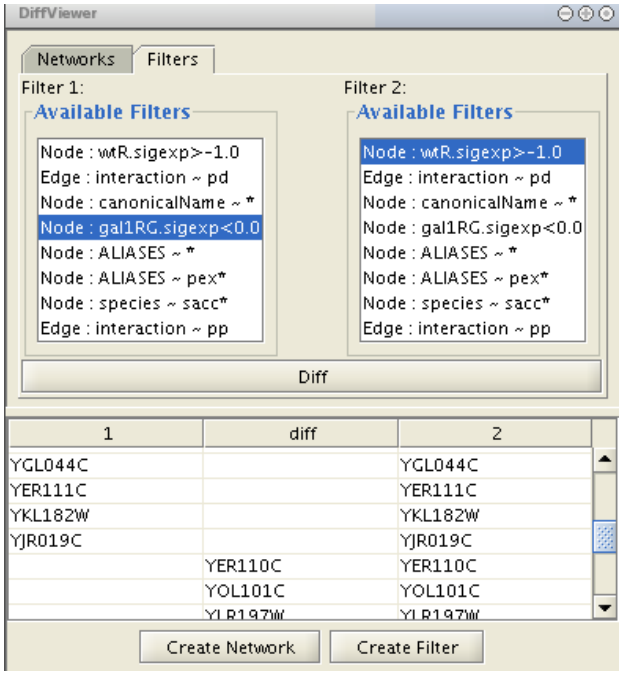
Once clicked, the following dialog box appears to allow you to choose an available filter:



Clicking on the Browser button will open the graph attributes browser.

The Diff Filter Feature

The Diff feature, accessible from the Data menu shown above, allows you to see which nodes or edges pass each of two selected filters and which nodes or edges those filters don't have in common (i.e. it shows the differences between two filters). This is useful for quickly comparing two filters. You can then create a new network that contains the differences ("Create Network" button) or create a new filter that selects the differences ("Create Filter" button).



The same functionality is available here for networks. You can select any two networks that are loaded and create a new network from their differences.

11. Acknowledgements

Cytoscape is built with a number of open source 3rd party Java libraries. The Cytoscape team gratefully acknowledges the following libraries:

- The Colt Distribution: Open Source Libraries for High Performance Scientific and Technical Computing in Java. Information is available at: http://hoschek.home.cern.ch/hoschek/colt/.

- GNU Getopt in Java. Information is available at: http://www.urbanophile.com/arenn/hacking/download.html.

- Graph INterface librarY a.k.a. GINY. Information is available at: http://csbi.sourceforge.net/.

- JDOM. Information is available at: http://www.jdom.org.

- JUnit. Information is available at: http://junit.org.

- JGoodies Looks. Information is available at: http://www.jgoodies.com/freeware/looks/index.html.

- Piccolo. Information is available at: http://www.cs.umd.edu/hcil/jazz/.

- Type-Specific Collections Library, from Sosnoski Software Solutions, Inc. Information is available at: http://www.sosnoski.com/opensrc/tclib/.

- Xerces Java XML parser. Information is available at: http://xml.apache.org/xerces-j/.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

This product includes software developed by the JDOM Project (http://www.jdom.org/).

Appendix: Annotation Server Format

This section for advanced users.

Introduction

Annotation in Cytoscape is stored as a set of ontologies (e.g. the Gene Ontology), a set of ontology controlled vocabulary term annotations for the genes from a given organism and a synonym table for gene names. For example, using the Gene Ontology, the Saccharomyces Cerevisiae **GAL4** gene's biological process is described as "regulation of transcription", to which GO has assigned the number 45449 (a GO ID). You can see below that "regulation of transcription" is a subcategory of "transcription", which is a subcategory of "nucleobase, nucleoside, nucleotide and nucleic acid metabolism", etc.

GO 8150 biological_process
  GO 7582 physiological processes
    GO 8152 metabolism
      GO 6139 nucleobase, nucleoside, nucleotide and nucleic acid metabolism
        GO 6350 transcription
          GO 45449 regulation of transcription

Cytoscape can use this 'hierarchical' ontology to annotate recognized genes, provided the user chooses a level of the hierarchy to use for a given set of annotations. The ontology provided to Cytoscape does not have to be hierarchical, but if it is not, there is no real advantage to storing annotations this way compared to just storing the annotation terms in a node attribute file.

The **annotation server** (originally called the biodata server) is a Cytoscape feature which allows you to load, navigate, and assign annotation terms to nodes in a network.

There are two modes in which an annotation server can be run:

1. As an in-process version of the same code, that runs in the same Java Virtual Machine as Cytoscape. This is the default for the official release of Cytoscape.
2. As a separately running program, an RMI server with which Cytoscape communicates.

The RMI server is especially useful if there are multiple Cytoscape users all running in the same location, using the same annotation: a group of yeast biologists, for example, all within the same institute or if a group studies several organisms (yeast, human, mouse ...). The RMI server is a little bit of trouble to set up, however, and so many people will elect to use the in-process server.

Building your own annotation files

The annotation server requires that the gene annotations, and associated ontology on controlled vocabulary terms, follow a simple format. This simple format was chosen because it is efficient to parse and easy to use.

The flat file formats are explained below:

The Ontology Format

By example (the Gene Ontology - GO):

```
(curator=GO) (type=all)
0003673 = Gene_Ontology
0003674 = molecular_function  [partof: 0003673 ]
0008435 = anticoagulant [isa: 0003674 ]
0016172 = antifreeze [isa: 0003674 ]
0016173 = ice nucleation inhibitor [isa: 0016172 ]
0016209 = antioxidant [isa: 0003674 ]
0045174 = glutathione dehydrogenase (ascorbate) [isa: 0009491 0015038 0016209 0016672 ]
0004362 = glutathione reductase (NADPH) [isa: 0015038 0015933 0016209 0016654 ]
0017019 = myosin phosphatase catalyst  [partof: 0017018 ]
...
```

A second example (KEGG pathway ontology):

```
(curator=KEGG) (type=Metabolic Pathways)
90001 = Metabolism
80001 = Carbohydrate Metabolism [isa: 90001 ]
80003 = Lipid Metabolism [isa: 90001 ]
80002 = Energy Metabolism [isa: 90001 ]
80004 = Nucleotide Metabolism [isa: 90001 ]
80005 = Amino Acid Metabolism [isa: 90001 ]
80006 = Metabolism of Other Amino Acids [isa: 90001 ]
80007 = Metabolism of Complex Carbohydrates [isa: 90001 ]
...
```

The format has three required features:

1. The first line contains two parenthesized assignments, for *curator* and *type*. In the GO example above, the ontology file (which is created from the XML GO provides) nests all three specific ontologies (molecular function, biological process, cellular component) below the 'root' ontology, named 'Gene_Ontology'.

   ```
   (type=all)
   ```

   tells you that all three ontologies are included in that file.

2. Following the mandatory title line, there are one or more category lines, each with the form:

   ```
   number0 = name [isa:|partof: number1 number2 ...]
   ```

   'isa' and 'partof' are terms used in GO; they describe the relation between parent and child terms in the ontology hierarchy.

3. The trailing blank before each left square bracket is *not* required; it is an artifact of the python script that creates these files.

The Annotation Format

By example (from the GO biological process annotation file):

```
(species=Saccharomyces cerevisiae) (type=Biological Process) (curator=GO)
YMR056C = 0006854
YBR085W = 0006854
YJR155W = 0006081
...
```

and from KEGG:

```
(species=Mycobacterium tuberculosis) (type=Metabolic Pathways) (curator=KEGG)
RV0761C = 10
RV0761C = 71
RV0761C = 120
RV0761C = 350
RV0761C = 561
RV1862 = 10
...
```

The format has these required features:

1. The first line contains three parenthesized assignments, for *species*, *type* and *curator*. In the example just above, the annotation file (which we create for budding yeast from the flat text file maintained by SGD for the Gene Ontology project, and available both at their web site and at GO's) shows three yeast ORFs annotated for biological process, with respect to GO, described (further above).

2. Following the mandatory title line, there are one or more annoation lines, each with the form:

   canonicalName = ontologyTermID

3. Once loaded, this annotation (along with accompanying ontology) can be assigned to nodes in a Cytoscape network. **For this to work, the species type of the node must agree exactly with the species named on the first line of the annotation file. The canonicalName of your node must exactly match the canonicalName present in the annotation file.** If you don't see the expected results when using this feature of Cytoscape, check this again, as getting either of these wrong is a common mistake.

Load Data In-Process

The easiest way to make annotation available to Cytoscape is by loading annotation into an in-process annotation server. This is the default for the official release of Cytoscape.

The Annotation Manifest

You must first create a text file to specify the files you want Cytoscape to load. Here is an example, from a file which (for convenience) we usually call **"manifest"**

```
ontology=goOntology.txt
annotation=yeastBiologicalProcess.txt
annotation=yeastMolecularFunction.txt
annotation=yeastCellularComponent.txt
```

Use the Cytoscape ***-b*** command line argument to specify the annotation manifest file to read (e.g. -b manifest). Please note that the **-s** switch, which sets the default species for your data is required to exactly match the species named in any annotation file you wish to use.

Getting and Reformatting GO Data

The Gene Ontology (GO) project is a valuable source of annotation for the genes of many organisms. In this section we will explain how to:

1. Obtain the GO ontology file
2. Reformat it into the simpler flat file Cytoscape uses
3. Obtain an annotation file (we illustrate with yeast and human annotation)
4. Reformat the annotation files into the simple Cytoscape format

Obtain the GO ontology file

Go to the GO XML FTP (ftp://ftp.geneontology.org/pub/go/xml/) page. Download the latest **go-YYYYMM-termdb.xml.gz** file.

Reformat GO XML ontology file into a flat file

```
gunzip go-YYYYMM-termdb.xml.gz
python parseGoTermsToFlatFile.py  go-YYYYMM-termdb.xml > goOntology.txt
```

(See below for Python script listing)

Obtain the 'association' file for your organism

GO maintains a list of association files for many organisms; these files associate genes with GO terms. The next step is to get the file for the organism/s you are interested in, and parse it into the form Cytoscape needs. A list of files may be seen at http://www.geneontology.org/GO.current.annotations.shtml. The rightmost column contains links to tab-delimited files of gene associations, by species. Choose the species you are interested in, and click 'Download'.

Let's use '**GO Annotations @ EBI** Human' as an example. After you have downloaded and saved the file, look at the first few lines:

```
SPTR    O00115  DRN2_HUMAN              GO:0003677      PUBMED:9714827  TAS             F
Deoxyribonuclease II precursor  IPI00010348     protein taxon:9606              SPTR
SPTR    O00115  DRN2_HUMAN              GO:0004519      GOA:spkw        IEA             F
Deoxyribonuclease II precursor  IPI00010348     protein taxon:9606        20020425      SPTR
SPTR    O00115  DRN2_HUMAN              GO:0004531      PUBMED:9714827  TAS             F
Deoxyribonuclease II precursor  IPI00010348     protein taxon:9606              SPTR
...
```

Note that line wrapping has occurred here, so each line of the actual file is wrapped to two lines.

The goal is to create from these lines the following lines:

```
(species=Homo sapiens) (type=Molecular Function) (curator=GO)
IPI00010348 = 0003677
IPI00010348 = 0004519
IPI00010348 = 0004531
...
```

or

```
(species=Homo sapiens) (type=Biological Process) (curator=GO)
NP_001366 = 0006259
NP_001366 = 0006915
NP_005289 = 0007186
NP_647593 = 0006899
...
```

The first sample contains molecular function annotation for proteins, and each protein is identified by its IPI number. IPI is the International Protein Index, which maintains cross references to the main databases for human, mouse and rat proteomes.

The second sample contains biological process annotation, and each protein is identified by its NP (RefSeq) number.

These two naming systems, IPI and RefSeq, are two of many that you can use for canonical names when you run Cytoscape. For budding yeast, it is much easier: the yeast community always uses standard ORF names, and so Cytoscape uses these as canonical names. For human proteins and genes, there is no such single simple standard. See section

5. Building and Storing Interaction Networks for more information.

The solution (for those working with human genes or proteins) is, once you have downloaded the annotations file, to:

1. Decide which naming system you want to use.
2. Download ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/HUMAN/xrefs.goa. This cross-reference file, when used strategically, allows you to create Cytoscape-compatible annotation files in which the canonical name is the one most meaningful to you.
3. Examine ***xrefs.goa*** to figure out which column contains the names you wish to use.
4. Make a very slight modification to the python script described below, and then
5. Run that script, supplying both xrefs.goa and that annotation file as arguments.

Here are a few sample lines from ***xrefs.goa***:

```
SP      O00115  IPI00010348           ENSP00000222219;      NP_001366;
        BAA28623;AAC77366;AAC35751;AAC39852;BAB55598;AAB51172;AAH10419;     2960,DNASE2
        1777,DNASE2
SP      O00116  IPI00010349           ENSP00000324567;ENSP00000264167;     NP_003650;
        CAA70591;      327,AGPS       8540,AGPS
SP      O00124  IPI00010353           ENSP00000265616;ENSP00000322580;     NP_005662;
        BAA18958;BAA18959;AAH20694;           7993,D8S2298E
...
```

Note that line wrapping has occurred here – each line in this example starts with the letters SP. See the README file for more information (ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/HUMAN/README)

Finally, run the script to create your three annotation files for human proteins:

- **bioproc.anno** (GO biological process annotation)
- **molfunc.anno** (GO molecular function annotation)
- **cellcomp.anno** (GO cellular component annotation)

using the supplied python script. It may be necessary to modify this script slightly if RefSeq identifiers are not used as canonical names.

```
python parseAssignmentsToFlatFileFromGoaProject.py gene_association.goa_human xrefs.goa
```

(See below for Python script listing)

Python script examples:
These scripts, described above require Python version 2.2 or later.

Script 1 - parseGoTermsToFlatFile.py

```
# parseGoTermToFlatFile.py:  translate a GO XML ontology file into a simpler
#  Cytoscape flat file
#-------------------------------------------------------------------------------
# RCS: $Revision: 1.3 $   $Date: 2003/05/18 00:38:43 $
#-------------------------------------------------------------------------------
import re, pre, sys
#-------------------------------------------------------------------------------
def flatFilePrint (id, name, isaIDs, partofIDs):
  isa = ''
  if (len (isaIDs) > 0):
    isa = '[isa: '
    for isaID in isaIDs:
      isa += isaID
      isa += ' '
    isa += ']'

  partof = ''
  if (len (partofIDs) > 0):
    partof = '[partof: '
    for partofID in partofIDs:
      partof += partofID
      partof += ' '
    partof += ']'

  result = '%s = %s %s %s' % (id, name, isa, partof)
  result = result.strip ()
  if (result == 'isa = isa' or result == 'partof = partof'):
    print >> sys.stderr, 'meaningless term: %s' % result
  else:
    print result
#-------------------------------------------------------------------------------
if (len (sys.argv) != 2):
  print 'usage:  %s <someFile.xml>' % sys.argv [0]
  sys.exit ();

inputFilename = sys.argv [1];

print >> sys.stderr,  'reading %s...' % inputFilename
text = open (inputFilename).read ()
print >> sys.stderr,  'read %d characters' % len (text)
```

```
regex = '<go:term .*?>(.*?)</go:term>';
cregex = pre.compile (regex, re.DOTALL)   # . matches newlines

m = pre.findall (cregex, text)
print >> sys.stderr, 'number of go terms: %d' % len (m)


regex2 = '<go:accession>GO:(.*?)</go:accession>.*?<go:name>(.*?)</go:name>'
cregex2 = re.compile (regex2, re.DOTALL)

regex3 = '<go:isa\s*rdf:resource="http://www.geneontology.org/go#GO:(.*?)"\s*/>'
cregex3 = re.compile (regex3, re.DOTALL)

regex4 = '<go:part-of\s*rdf:resource="http://www.geneontology.org/go#GO:(.*?)"\s*/>'
cregex4 = re.compile (regex4, re.DOTALL)

goodElements = 0
badElements = 0

print '(curator=GO) (type=all)'
for term in m:
  m2 = re.search (cregex2, term)
  if (m2):
    goodElements += 1;
    id = m2.group (1)
    name = m2.group (2)
    isaIDs = []
    m3 = re.findall (cregex3, term);
    for ref in m3:
      isaIDs.append (ref)
    m4 = re.findall (cregex4, term);
    partofIDs = []
    for ref in m4:
      partofIDs.append (ref)
    flatFilePrint (id, name, isaIDs, partofIDs)
  else:
    badElements += 1;
    print >> sys.stderr, 'no match to m2...'
    print >> sys.stderr, "---------------\n%s\n------------------" % term


print >> sys.stderr,  'goodElements %d' % goodElements
print >> sys.stderr, 'badElements %d' % badElements

#-------------------------------------
```

Script 1 - parseAssignmentsToFlatFileFromGoaProject.py

```
#!/tools/bin/python
import sys
#-------------------------------------------------------------------------------
def fixCanonicalName (rawName):
# for instance, trim 'YBR085W|ANC3' to 'YBR085W'
  bar = rawName.find ('|')
  if (bar < 0):
    return rawName
  return rawName [:bar]

#-------------------------------------------------------------------------------
def fixGoID (rawID):

  bar = rawID.find (':') + 1
  return rawID [bar:]

#-------------------------------------------------------------------------------
def readGoaXrefFile (filename):

  lines = open (filename).read().split ('\n')
  result = {}
  for line in lines:
    if (len (line) < 10):
```

```
      continue
    tokens = line.split ('\t')
    ipi = tokens [2]
    np = tokens [5]
    semicolon = np.find (';')
    if (semicolon >= 0):
      np = np [:semicolon]
    if (len (ipi) > 0 and len (np) > 0):
      result [ipi] = np

  return result

#-------------------------------------------------------------------------------
if (len (sys.argv) != 3):
  print 'error!  parse   <gene_associations file from GO> <goa xrefs file> '
  sys.exit ()

associationFilename = sys.argv [1];
xrefsFilename = sys.argv [2]
species = 'Homo sapiens'

ipiToNPHash = readGoaXrefFile (xrefsFilename)
tester = 'IPI00099416'
print 'hash size: %d' % len (ipiToNPHash)
print 'test map: %s -> NP_054861: %s ' % (tester, ipiToNPHash [tester])

bioproc = open ('bioproc.txt', 'w')
molfunc = open ('molfunc.txt', 'w')
cellcomp = open ('cellcomp.txt', 'w')

bioproc.write ('(species=%s) (type=Biological Process) (curator=GO)\n' % species)
molfunc.write ('(species=%s) (type=Molecular Function) (curator=GO)\n' % species);
cellcomp.write ('(species=%s) (type=Cellular Component) (curator=GO)\n' % species);

lines=open(associationFilename).read().split('\n')
sys.stderr.write ('found %d lines\n' % len (lines))

for line in lines:
  if (line.find ('!') == 0 or len (line) < 2):
    continue
  tokens = line.split ('\t')
  goOntology = tokens [8]
  goIDraw = tokens [4]
  goID = goIDraw.split (':')[1]
  ipiName = fixCanonicalName (tokens [10])
  if (len (ipiName) < 1):
    continue


  if (not ipiToNPHash.has_key (ipiName)):
    continue
  refseqName = ipiToNPHash [ipiName]
  printName = refseqName
  #printName = ipiName
  if (ipiName == tester):
    print '%s (%s) has go term %s' % (tester, printName, goID)
  if (goOntology == 'C'):
    cellcomp.write ('%s = %s\n' % (printName, goID))
  elif (goOntology == 'P'):
    bioproc.write ('%s = %s\n' % (printName, goID))
  elif (goOntology == 'F'):
    molfunc.write ('%s = %s\n' % (printName, goID))
#-------------------------------------------------------------------------------
```

Appendix: GNU Lesser General Public License

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License").

Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the

application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

  3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

  Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

  This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

  4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

  If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

  5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

  However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it

contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS