

CytoScript : Running Scripts in CytoScape

Kyongryun Lee

Human Computer Interaction

Iowa State University

1. Introduction

CytoScript is a program to execute scripts in languages such as R, Scheme, Python, and Perl in Cytoscape. The goal of CytoScript is to extend Cytoscape through scripts, with the target application being the analysis of network or experimental data. The results of the analysis may then be represented in the Cytoscape network view. The scope of this document is limited to running R in Cytoscape.

R is a scripting language for statistical computing and graphics. It provides various statistical methods such as clustering, classical statistical tests, linear or nonlinear modeling and various graphical techniques. Moreover, the Bioconductor project provides many R packages (essentially plugins for R) designed for data analysis in bioinformatics. Thus, researchers often use R to analyze biological experimental data and networks.

Cytoscape is a bioinformatics software to visualize networks and to visually intergrate these networks with gene expression profiles and other experimental data. Many researchers are developing useful programs to analyze data and networks in Cytoscape through plugins like CytoScript.

2. Design

Figure 2.1 shows the structure of a system to run R in cytoscape. The step of the structure is as follows. First, we use the GUI provided by CytoScript to load a file written in R. Second, CytoScript has uses javax.script to execute scripts written in any language with a javax.script engine implementation. The R engine is provided by the edu.iastate.metnet.R package that relies may eventually support JRI, SOAP, and Rserve backends, though only JRI is supported currently. JRI is a simple Java to R interface written by Simon Urbanek that allows calling R from Java.

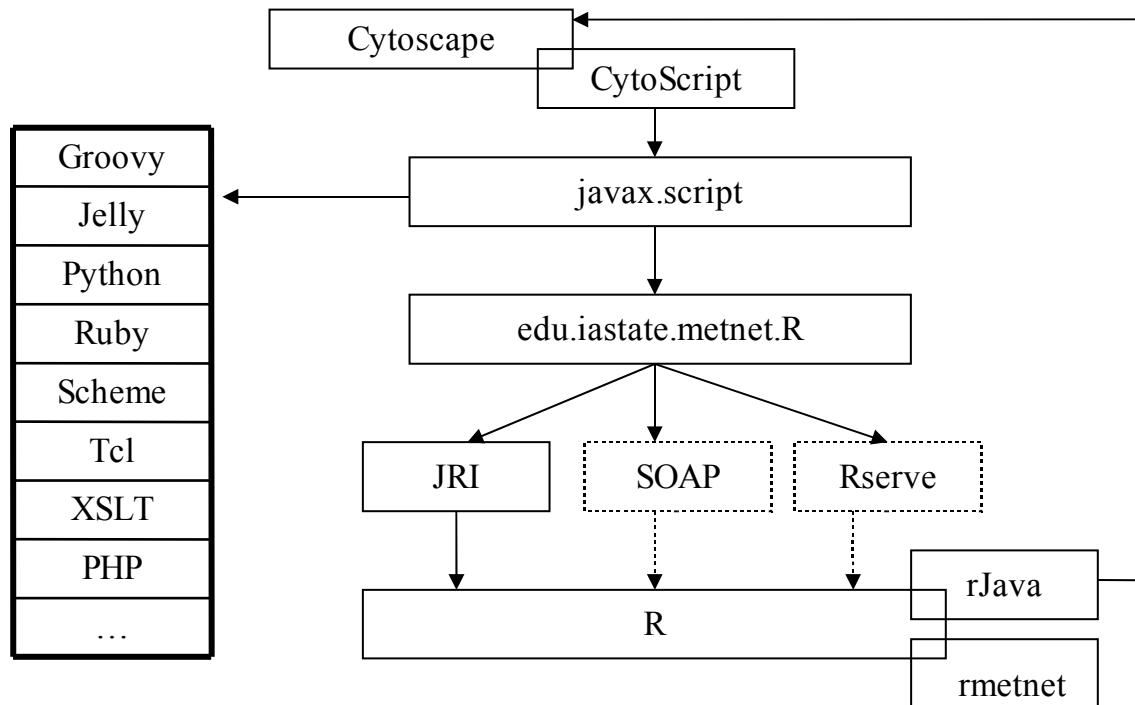


Figure 2.1 System Design

Finally, the R script is executed by JRI.

The R script uses the rJava and rmetnet packages. rJava is the inverse of JRI in that it provides a link from R to Java. It allows the script to show the results of analysis in Cytoscape. The rmetnet package provides convenience functionality for interacting linking R with the MetNet platform. Currently its sole feature is an implementation of the abstract graph class (provided by the Bioconductor graph package) using a CyNetwork object from Cytoscape.

3. Example

As an example, let us find strongly connected components (SCCs) for a given network, using R.

1. Call Cytoscape to get the current network, using rJava.

```
cy_network <- .jcall("cytoscape/Cytoscape", "Lcytoscape/CyNetwork;",
"getCurrentNetwork")
```

2. Create our graph object.

```
network <- new("graphCytoscape", cy_network)
```

3. Find SCCs in the graph.

```
comps <- strongComp(network)
```

4. Animations

The Animator plugin provides animation support to Cytoscape, so scripts run via Cytoscript may use animations to show results. Animations may affect anything controlled with the VizMapper tool, which includes color, shape, font face, arrow type, line type, size of font, node height and width, label and label color. Each frame of the animation consists of a Cytoscape VisualStyle object with an associated duration.

It is not trivial to construct VisualStyle objects, so the Animator plugin takes a language-oriented approach and provides a simple expression-based mechanism for describing the mappings that constitute a VisualStyle. This language of so called “Mapping Rules” are described in the next section.

For instance, let us make it animate to fill the red color of some nodes.

Example

1. We make animation object from the program in R, using rJava.

```
animation <- .jnew("edu/iastate/cytoscape/animation/SimpleAnimation").
```

2. Next we create a frame for each strongly connected component.

2-1. We use mapping rules to specify that the nodes in the component are to be colored red.

```
rules <- paste("node canonicalName =", comp, "-> fill color = 255,0,0", col = ";")
```

2-2. Make a frame object which is the basic mapping rule (VisualStyle) storage unit in the animation framework. Here 0.5 is the duration of the frame.

```
frame <- .jnew("edu/iastate/cytoscape/animation/SimpleFrame", rules, 0.5)
```

2-3. Add the frame to the animation.

```
.jcall(animation,, "addFrame",  
        .jcast(frame, "edu/iastate/cytoscape/animation/Frame"))
```

3. Play the animation.

```
.jcall("Animator",, "playAnimation",  
      .jcast(animation,"edu/iastate/cytoscape/animation/Animation"))
```

Following above steps, we can make animations easily from R using mapping rules.

5. Mapping Rule Specification

The mapping rule syntax is inspired by that of the FCModeler mapping rules. The general syntax is given below:

```
type attribute-test -> appearance
```

The `type` indicates the type of element that is being mapped and may be either node or edge. The `attribute-test` follows the syntax:

```
attribute-name operator attribute-value
```

where the `attribute-name` is the name of an attribute in Cytoscape's `CyAttributes` object, for the given element type, `operator` is one character in the set `{=, <, >}` that tests the named attribute against the `attribute-value`. Finally, the `appearance` relates a visual property-name to a property-value by the syntax:

```
property-name = property-value
```

`property-name` could be "fill color", "border color", "font face", "font size", "height", "width", "shape", "lable", "label color", "tooltip", "line type" for node and "color", "line type", "source arrow", "target arrow", "lable", "tooltip", "font face", "font size" for edge.

As a simple example, if one wished to fill all of the nodes with a value for "expression" greater than 3 with blue:

```
node expression > 3 -> fill color = 0,0,255 //R,B,G for property-value
```

Multiple mapping rules may be separated by the semicolon.

6. Graphical Demonstration

The following figures demonstrate the generation of an animation by an Rscript via CytoScript.

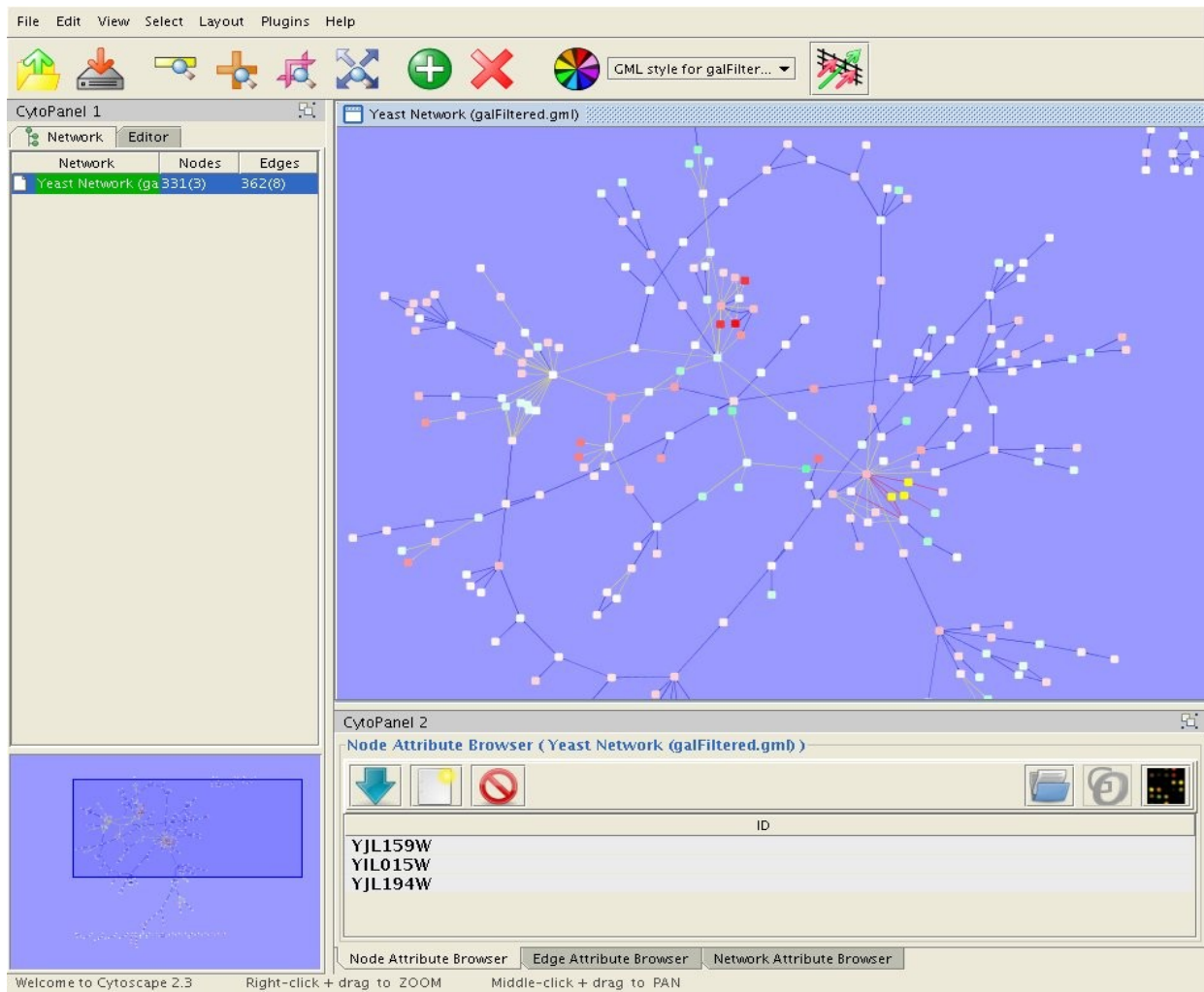


Figure 6.1 Current Network

- Demonstration
- Used script language : R
 - Goal : find strongly connected components (SCCs)
 - Input : Yeast Network
 - Output : colored strongly connected components with red.
(Others are colored by white)

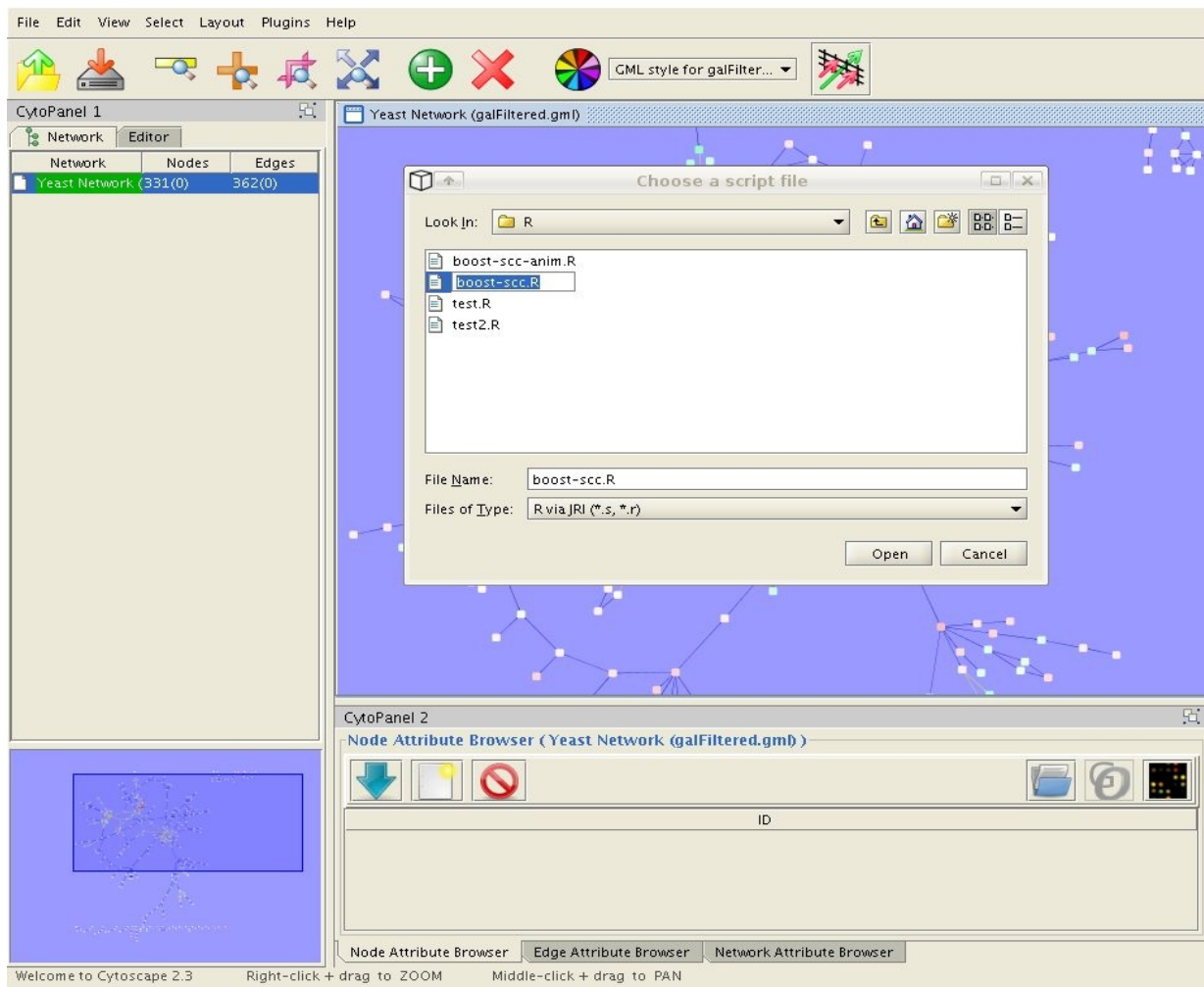


Figure 6.2 Loading a file

We begin with the network shown in Figure 6.1. In Plugins, we select CytoScript, and load the file written in R (see figure 6.2). Figure 6.3 shows strongly connected components with red colors. There were two sets of SCCs found in the Yeast Network. To see each individually, we can create an animation. To play animation, we write an R script that generates the animation from the analysis results. It loads the frames into a new animation and displays the animation GUI. In the play animation dialog, we can set the speed by adjusting the delay bar and the choose the desired cycle mode using radio buttons. Please see Figure 6.4.

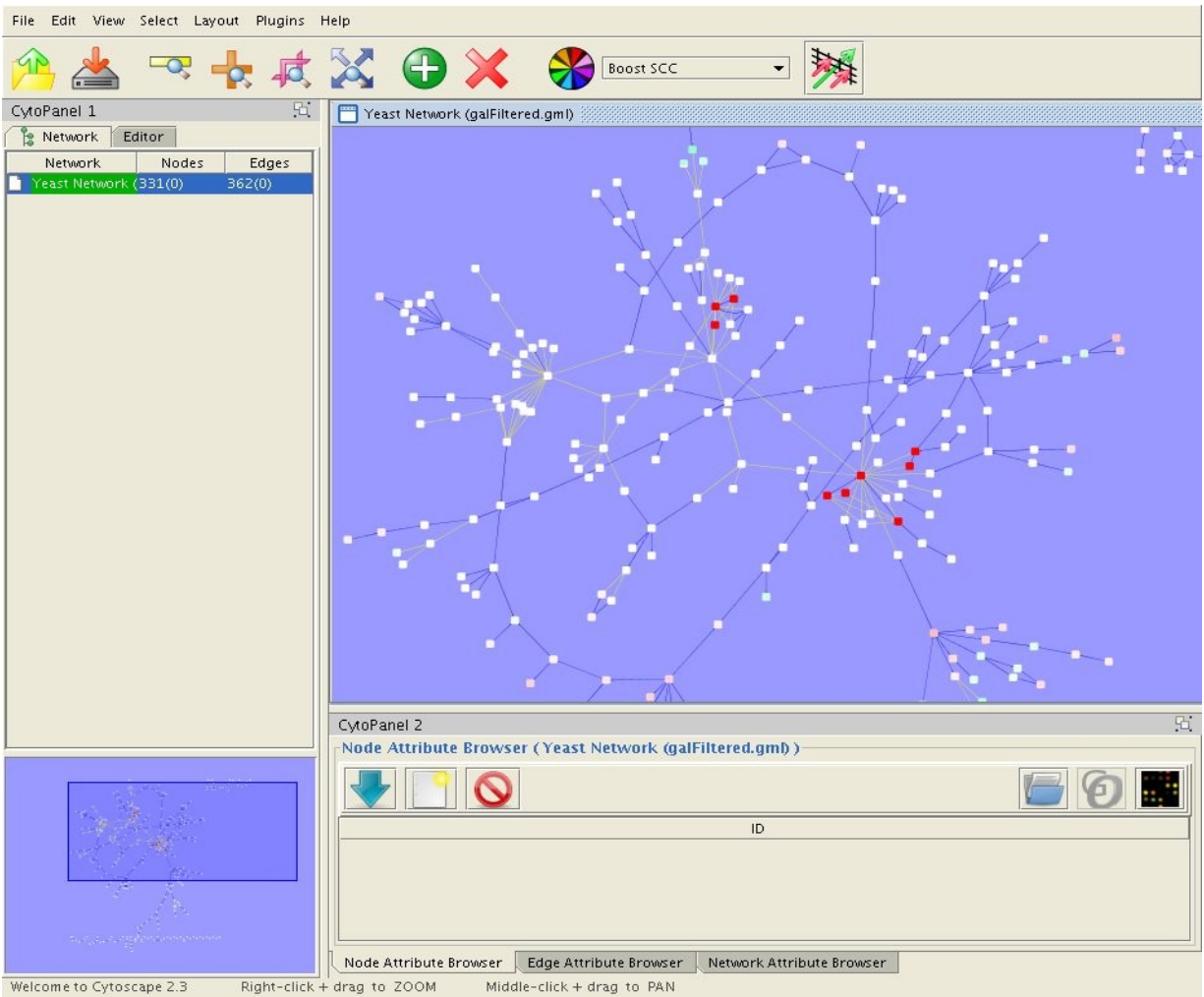
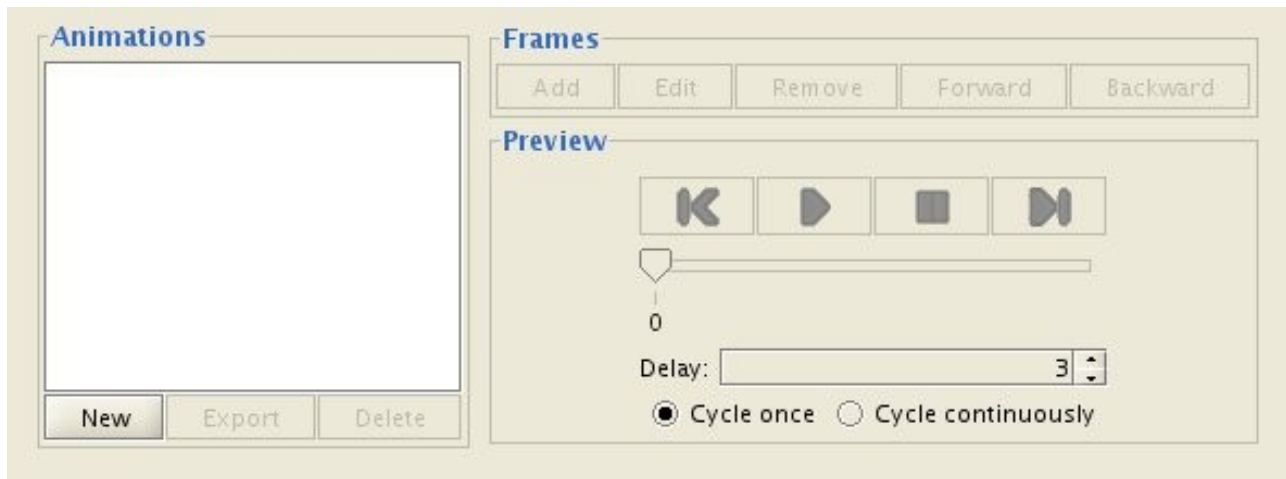


Figure 6.3 SCCs in the network

7. Animation GUI

The Animator plugin also provides animation GUI to Cytoscape, so we may run animation of a given network in our favorite. We are able to select the properties of visualization through VizMapper on the GUI. A following example is shown in detail.

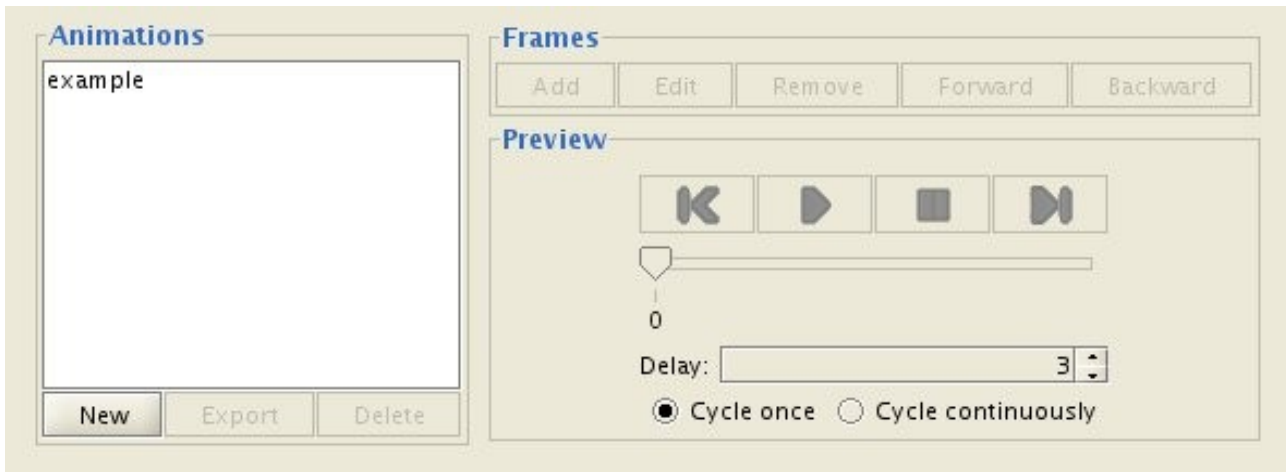
1. Import a network. For this example, we use a sample network, “GalFiltered.sif”
2. Choose “Create Animation” on menu of Plugins. Animation GUI would be shown up as follows.



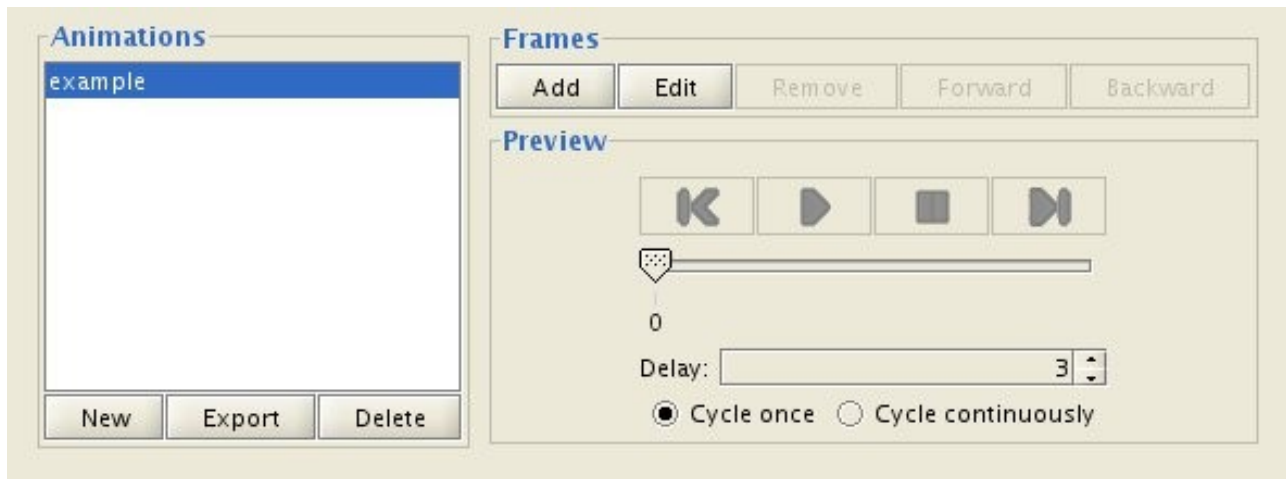
3. Click “New” button. This button is to create the name of animation.



4. Click “OK” button. Then the animation name would be registered on the GUI.

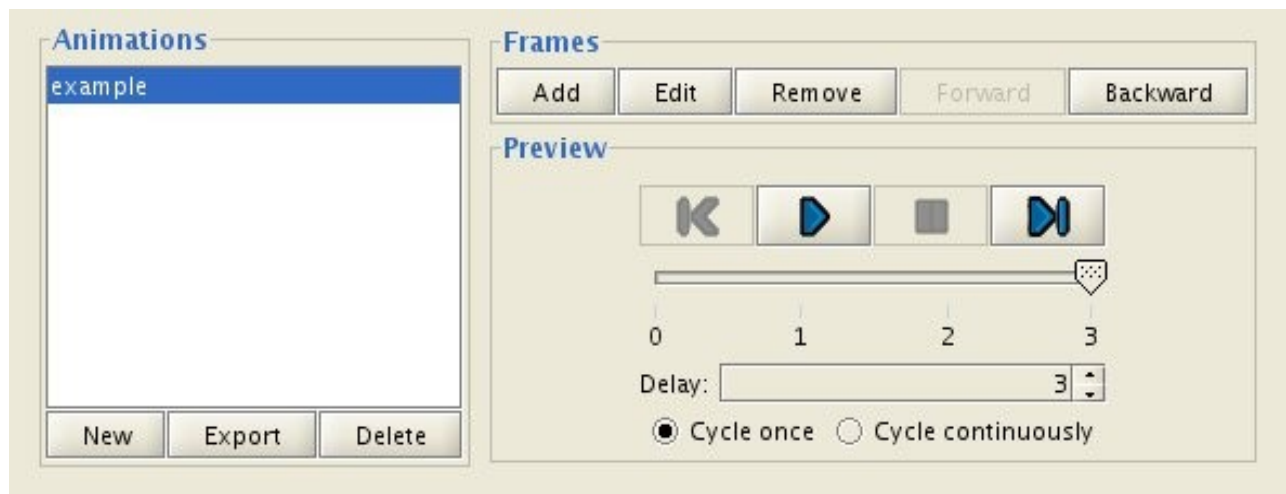


5. We need to make the frame of animations to run. When we click animation name, example, the buttons, ”Add” and “Edit”, are active. “Add” button is to make the frame and “Edit” button is to set up the visual properties on the frame.



6. Click “Add” button. Then one frame would be created for the animation. On this frame, we set up the visual property. Click “Edit” button then VizMapper would show up. We edit visual properties through this VizMapper such node color, node shape, etc. For example, we set the node color with the value of attributes, “1”, to red. If we want to make more frames, click “Add” button and click “Edit” button and set up the visual properties. In this example, we created three frames and set the node color with the value of attributes, “1”, to red, with the value of attributes, “2” to yellow and with the value of attributes, “3” to blue.

7. Then, we can see final GUI to run as follows.



8. For more explanation, on the Preview, we can see the frame bar. Through this frame bar, we can see visual style of animation on each frame. On the Frame, “Backward” button moves the frame

on the backward and “Forward” button moves the frame on the forward.

9. Let us run our animation. To do this, click “Play” button.

10. We can see the animations we created. Each frame is shown as follows.

The screenshot displays the Cytoscape 2.4.0 software interface. At the top, there is a menu bar with options: File, Edit, View, Select, Layout, Plugins, and Help. Below the menu bar is a toolbar containing various icons for file operations and navigation. A search bar is located to the right of the toolbar, with a dropdown menu currently set to 'node'. The main workspace is divided into two panels: CytoPanel 1 and CytoPanel 2.

CytoPanel 1: This panel contains a 'Network' list and an 'Editor' tab. The network list has the following data:

Network	Nodes	Edges
galFiltered.sif	331(0)	362(0)

The main visualization area in CytoPanel 1 shows a complex network graph with numerous nodes and edges. Some nodes are highlighted in red, while others are white. The graph is dense and interconnected.

CytoPanel 2: This panel is currently empty, showing only a header 'ID' and a large empty space for displaying node or edge attributes.

At the bottom of the interface, there are three tabs for attribute browsing: 'Node Attribute Browser', 'Edge Attribute Browser', and 'Network Attribute Browser'. The status bar at the very bottom provides instructions: 'Welcome to Cytoscape 2.4.0', 'Right-click + drag to ZOOM', and 'Middle-click + drag to PAN'.

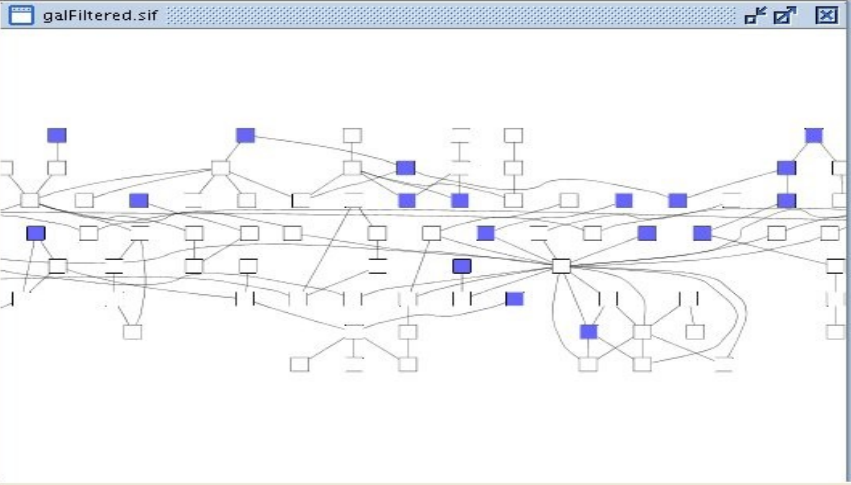
File Edit View Select Layout Plugins Help

node Search:

CytoPanel 1

Network Editor

Network	Nodes	Edges
galFiltered.sif	331(0)	362(0)



galFiltered.sif

CytoPanel 2

ID

Node Attribute Browser Edge Attribute Browser Network Attribute Browser

Welcome to Cytoscape 2.4.0 Right-click + drag to ZOOM Middle-click + drag to PAN

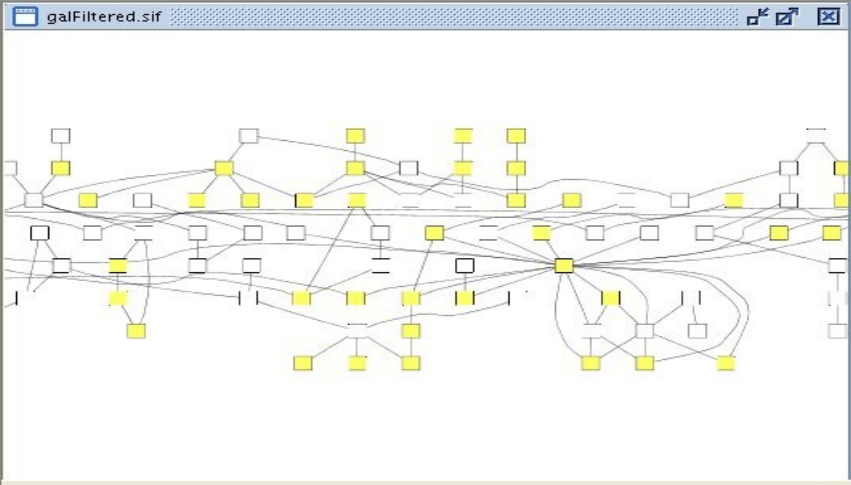
File Edit View Select Layout Plugins Help

node Search:

CytoPanel 1

Network Editor

Network	Nodes	Edges
galFiltered.sif	331(0)	362(0)



galFiltered.sif

CytoPanel 2

ID

Node Attribute Browser Edge Attribute Browser Network Attribute Browser

Welcome to Cytoscape 2.4.0 Right-click + drag to ZOOM Middle-click + drag to PAN

8. Installation

Dependencies :

The *javax.script*-compliant R engine provided by *edu.iastate.metnet.R* (part of CytoScript) requires R (of course) and the JRI library. JRI is included with the rJava R package, which is also required for running the example R snippets in this document.

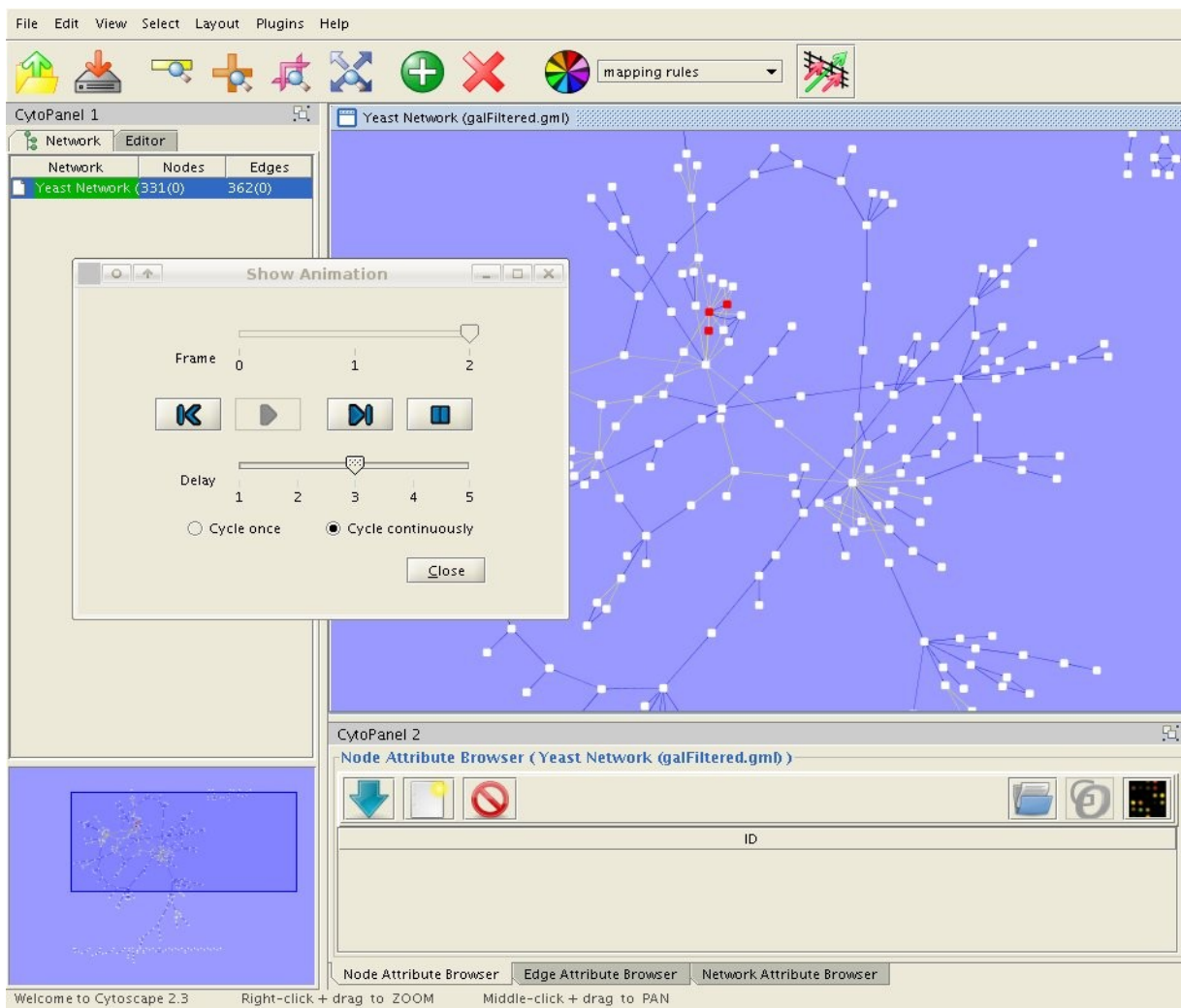


Figure 6.4 Interface of animation

Environment Variables :

Users of all operating systems need to set R_HOME to the root directory of your R installation (ie /usr/lib/R). The JRI library needs to be on your java.library.path. To achieve this, set the following environment variable (depending on your operating system) to the location of JRI.

Linux: LD_LIBRARY_PATH

Windows: PATH

Mac: DYLD_LIBRARY_PATH

Plugin Installation :

Put CytoScript.jar and Animator.jar in the Cytoscape Plugins folder.

Appendix

Useful package or library in R and Java

Javax.script - package to execute scripting languages in a Java application.

=> <http://download.java.net/jdk6/docs/api/javax/script/package-summary.html>.

rJava - simple R to Java interface which provides calling Java from R.
(the inverse of rJava : JRI)

=> <http://rosuda.org/rJava/>

bioconductor - open source to develop softwares of bioinformatics for analysis.

=> <http://www.bioconductor.org/>

bioconductor – GraphsAndNetworks -

graph - package with many functions for representation and analysis of graphs.

=> <http://bioconductor.org/packages/1.8/bioc/html/graph.html>

RBGL - R interface to boost graph library.

=> <http://www.bioconductor.org/packages/bioc/1.6/src/contrib/html/descrips/RBGLDesc.html>

Definition of SCCs

For a directed graph $G = (V, E)$ such as $u, v \in V$ and $(u, v) \in E$, $SCC = (V', E') \in G$ such as $u', v' \in V'$, $(u', v') \in E'$, where is a path P from u' to v' and from v' to u' for $u', v' \in V$ and $(u', v') \in E$