BIOINFORMATICS APPLICATIONS NOTE

Vol. 27 no. 17 2011, pages 2451–2452
doi:10.1093/bioinformatics/btr388

Systems biology

Advance Access publication June 27, 2011

CytoscapeRPC: a plugin to create, modify and query Cytoscape networks from scripting languages

Jan J. Bot[1,2,*] and Marcel J. T. Reinders[1,2]

[1]Delft Bioinformatics Lab, Delft University of Technology, Delft and [2]Netherlands Bioinformatics Centre, Nijmegen, The Netherlands

Associate Editor: Olga Troyanskaya

ABSTRACT

Summary: CytoscapeRPC is a plugin for Cytoscape which allows users to create, query and modify Cytoscape networks from any programming language which supports XML-RPC. This enables them to access Cytoscape functionality and visualize their data interactively without leaving the programming environment with which they are familiar.

Availability: Install through the Cytoscape plugin manager or visit the web page: http://wiki.nbic.nl/index.php/CytoscapeRPC for the user tutorial and download.

Contact: j.j.bot@tudelft.nl

Received on February 3, 2011; revised on May 8, 2011; accepted on June 22, 2011

1 INTRODUCTION

Cytoscape (Shannon *et al.*, 2003) is a network visualization and analysis tool which is mainly used in the life sciences. It enables users to display the data as graphs and helps them to perform basic actions such as gathering network-topological statistics. It comes with its own plugin management system to facilitate the easy implementation of new functionality.

Researchers in the life sciences often use scripting languages to process their data and use Cytoscape to visualize it. Their workflow when using Cytoscape comes down to running their algorithm, converting the output data such that Cytoscape can use it, loading the data into Cytoscape and then setting the appropriate layout and visualization rules to control how the data is displayed. The last three steps of this process are repetitive and time consuming.

We present CytoscapeRPC, a Cytoscape plugin created to solve the aforementioned problems by enabling users to create a connection to Cytoscape using XML-RPC, allowing them to create, modify and query Cytoscape networks from within their own programming environment.

2 METHODS AND IMPLEMENTATION

CytoscapeRPC builds on top of the Cytoscape libraries, making these components easily accessible. Nodes, edges and networks e.g. can be accessed through their human-readable identifiers. Attributes for these components can be created, queried and modified. They can also be linked to visual mappers. For example, the user can perform a mapping from molecule type (receptor, ligand) to the shape of a node (triangle, square) or, by using the *P*-value of an interaction, change the colour of the edges between molecules.

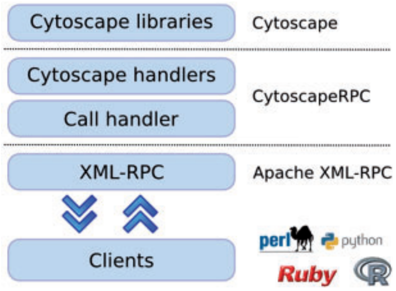


Fig. 1. CytoscapeRPC design. The plugin is split up into a function (handlers), a translation (call handler) and a communication layer. These layers ensure that the API can be kept stable.

Layouts can be applied or the positions of nodes can be set manually by specifying the *x*- and *y*-coordinates.

The plugin uses XML-RPC (Winer, 1999), a communication protocol which uses XML to transfer messages between client and server applications. It allows programs written in different programming languages and running on different machines to communicate by passing messages which perform function calls. XML-RPC was chosen based on the relative simplicity of the protocol which ensures that server and client libraries are available for virtually every programming language (e.g. Perl, Python, Ruby etc.), thus giving access to Cytoscape through those languages.

The user can specify whether the plugin should be started on Cytoscape start-up and on which port the XML-RPC server should listen for incoming calls. This can be used to run Cytoscape on headless (i.e. non-GUI) computers.

The XML-RPC service normally listens to any incoming call. To limit access to the plugin, the XML-RPC security mode has been implemented. The user can request the plugin to only allow incoming calls from processes running on the same computer as Cytoscape itself.

The plugin exposes the CyCommand API which was created to share functionality between Cytoscape plugins. Through CytoscapeRPC, the user can access this functionality and control third party plugins. Plugin developers can use the CyCommand functionality and CytoscapeRPC to easily share their work with other programmers.

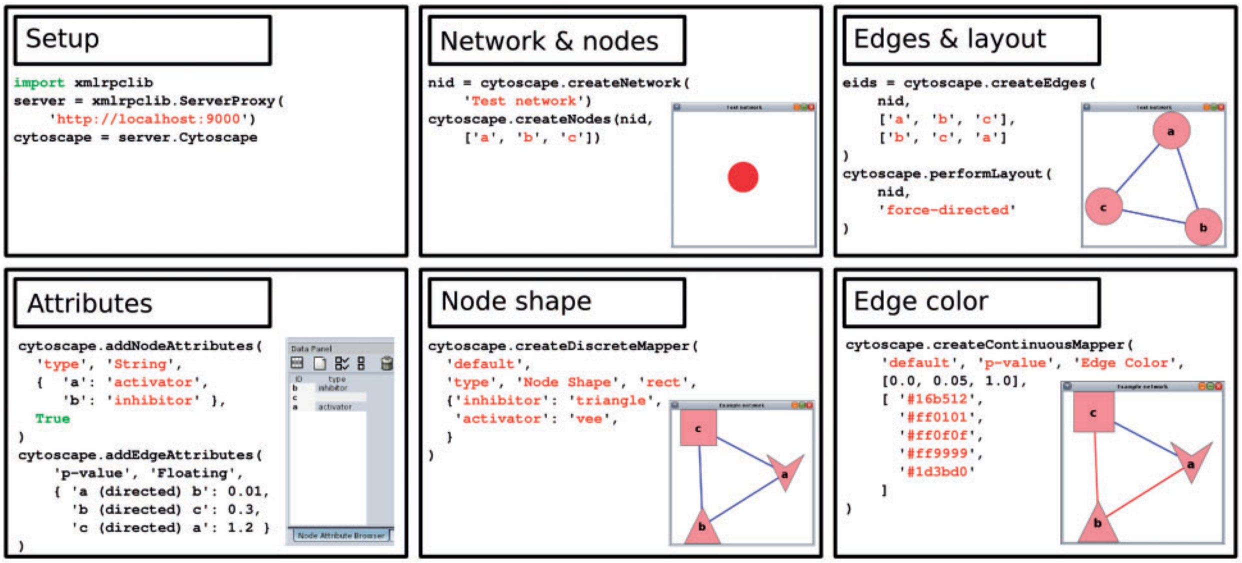
2.1 Plugin design

The plugin consists of and relies on a number of components, as shown in Figure 1. The different components are explained from top to bottom:

(1) Cytoscape and its libraries.

(2) Handler classes: classes which provide abstract functions to Cytoscape. Classes have been implemented for nodes, edges, networks, layouts, vizmappers etc. Their functionality consists of translating basic data types such as numbers and strings that can be

*To whom correspondence should be addressed.



Setup

```
import xmlrpclib
server = xmlrpclib.ServerProxy(
    'http://localhost:9000')
cytoscape = server.Cytoscape
```

Network & nodes

```
nid = cytoscape.createNetwork(
    'Test network')
cytoscape.createNodes(nid,
    ['a', 'b', 'c'])
```

Edges & layout

```
eids = cytoscape.createEdges(
    nid,
    ['a', 'b', 'c'],
    ['b', 'c', 'a']
)
cytoscape.performLayout(
    nid,
    'force-directed'
)
```

Attributes

```
cytoscape.addNodeAttributes(
  'type', 'String',
  {  'a': 'activator',
     'b': 'inhibitor' },
  True
)
cytoscape.addEdgeAttributes(
    'p-value', 'Floating',
    { 'a (directed) b': 0.01,
      'b (directed) c': 0.3,
      'c (directed) a': 1.2 }
)
```

Data Panel

ID	type
b	inhibitor
c	
a	activator

Node shape

```
cytoscape.createDiscreteMapper(
  'default',
  'type', 'Node Shape', 'rect',
  {'inhibitor': 'triangle',
   'activator': 'vee',
  }
)
```

Edge color

```
cytoscape.createContinuousMapper(
    'default', 'p-value', 'Edge Color',
    [0.0, 0.05, 1.0],
    [ '#16b512',
      '#ff0101',
      '#ff0f0f',
      '#ff9999',
      '#1d3bd0'
    ]
)
```

Fig. 2. Example Python session for creating a network using CytoscapeRPC. *Setup*: initializing the XMLRPC library of Python and creating a connection to the server. *Network & nodes*: creating a network with the title 'Test network' and adding three nodes, 'a', 'b' and 'c'. *Edges & layout*: adding three edges to the network and performing the 'force directed' layout. *Attributes*: Adding attributes to nodes and edges. Note that two errors were introduced: for one node no 'type' was specified and the edge from node 'c' to node 'a' got a '*P*-value' of 1.2. *Node shape*: setting the node shape based on the 'type' attribute. A default value has been specified to give nodes without that attribute a different shape, in this case a square. *Edge colour*: setting the edge colour based on the '*P*-value' attribute. One has to specify between which points the values should be interpolated. Significant *P*-values get a bright red colour while invalid *P*-values are either green (lower then zero) or blue (larger then one). Colours are represented with their HTML values.

handled by XML-RPC into Cytoscape objects and to provide abstract methods to manipulate the Cytoscape environment with.

(3) CallHandler class: class which provides all the functions that are made available through the external interface. All calls are redirected to the appropriate functions in the handler classes.

(4) XML-RPC: the Apache library providing the XML-RPC server.

(5) External clients: client code which directly calls or provides abstract methods for working with Cytoscape.

2.2 Usage examples

An example scenario with Python source code is provided in Figure 2. Other scenarios which can be found in the documentation include:

- Loading multiple network (sif), node attribute (noa) and edge attribute (eda) files, perform a default layout and set some visual attributes. This provides a starting point for analysis without having to go through the manual steps of data loading.
- Use an external layout algorithm from a package such as iGraph (Csardi and Nepusz, 2006) to determine the position of nodes. Use the plugin to transfer the location information to Cytoscape. This ensures that the computationally expensive network analysis can be performed using optimized software.
- Demonstrate network changes by scripting events for Cytoscape. Time series data of e.g. gene expression can be loaded and used to create visual representations. While the layout of the network can remain static, the colours of the nodes (in this case genes) can change based on their expression values. These different instances of the same network can be saved to image files and can be combined by e.g. the ImageMagick® 'convert' tool to create a movie of these changes.
- Import data from Cytoscape using R, perform an analysis and update the Cytoscape visualization based on its output.

3 CONCLUSION

CytoscapeRPC allows users to develop applications which leverage the functionality of Cytoscape without having to leave their programming environment. This lowers the barrier for people to start developing their own tools for Cytoscape.

ACKNOWLEDGEMENT

We would like to thank Paul Shannon, Marc Hulsman, Jeroen de Ridder and Tamas Nepusz for their early adoption of and feedback on the plugin.

Funding: This work was part of the BioRange programme of the Netherlands Bioinformatics Centre (NBIC), which is supported by a BSIK grant through the Netherlands Genomics Initiative (NGI).

Conflict of Interest: none declared.

REFERENCES

Csardi,G. and Nepusz,T. (2006) The igraph software package for complex network research. *InterJournal* **Complex Systems**, 1695.

Shannon,P. *et al*. (2003) Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.*, **13**, 2498–2504.

Winer,D. (1999) Xml-rpc specification. Technical report.