Check for updates

SOFTWARE TOOL ARTICLE

CyREST: Turbocharging Cytoscape Access for External Tools via a RESTful API [version 1; referees: 2 approved]

Keiichiro Ono[1], Tanja Muetze[2], Georgi Kolishovski[3], Paul Shannon[4], Barry Demchak[1]

[1]Department of Medicine, University of California, San Diego, CA, 92093-0688, USA
[2]Imperial College London, London, SW7 2AZ, UK
[3]Bioconductor, Boston, MA, 02125, USA
[4]Fred Hutchinson Cancer Research Institute, Seattle, WA, 98109, USA

Abstract

As bioinformatic workflows become increasingly complex and involve multiple specialized tools, so does the difficulty of reliably reproducing those workflows. Cytoscape is a critical workflow component for executing network visualization, analysis, and publishing tasks, but it can be operated only manually via a point-and-click user interface. Consequently, Cytoscape-oriented tasks are laborious and often error prone, especially with multistep protocols involving many networks.

In this paper, we present the new cyREST Cytoscape app and accompanying harmonization libraries. Together, they improve workflow reproducibility and researcher productivity by enabling popular languages (e.g., Python and R, JavaScript, and C#) and tools (e.g., IPython/Jupyter Notebook and RStudio) to directly define and query networks, and perform network analysis, layouts and renderings. We describe cyREST's API and overall construction, and present Python- and R-based examples that illustrate how Cytoscape can be integrated into large scale data analysis pipelines.

cyREST is available in the Cytoscape app store (http://apps.cytoscape.org) where it has been downloaded over 1900 times since its release in late 2014.

This article is included in the Cytoscape apps channel.

Open Peer Review

Referee Status: ✓ ✓

	Invited Referees	
	1	**2**
version 1 published 05 Aug 2015	✓ report	✓ report

1 **Sergey Nepomnyachiy**, Polytechnic Institute of New York University USA

2 **Guanming Wu**, Ontario Institute for Cancer Research Canada

Discuss this article

Comments (0)

Corresponding author: Barry Demchak (idekerlab.bdemchak@gmail.com)

Competing interests: No competing interests were disclosed.

Introduction

Because of its robust network analysis and visualization capabilities coupled with its vibrant user and developer community, Cytoscape 3 has become a tool of choice for studying large network-oriented *omics data sets on common workstations and for publishing results. However, even as Cytoscape[1] is well positioned to handle customized *omics workflows, bioinformaticians' need to quickly and efficiently create complex, varied, and repeatable workflows exceeds the capabilities of Cytoscape's existing automation features. At the same time, bioinformaticians have embraced a class of highly flexible tools consisting of fully fledged programming environments (e.g., IPython/Jupyter Notebook[2], RStudio, and MATLAB) coupled with programming languages (e.g., Python and R) and highly capable and flexible bioinformatic libraries.

Inasmuch as these tools address the data collection and analysis portions of typical bioinformatic workflows, Cytoscape complements them by addressing visualization, additional analysis, and network publication. To date, combining these tools with Cytoscape has seen only limited success, largely because of the limitations of Cytoscape's automation interfaces and its point-and-click user interface. Consequently, this integration has been labor intensive, inconvenient, and often unrepeatable, particularly as the complexity of analysis and visualization processing increases.

We created the cyREST Cytoscape app to enable automated access to the Cytoscape network and visualization models directly from within these tools, thereby exposing Cytoscape visualization, analysis, and publishing features in complex, varied, and reproducible bioinformatic workflows as shown in Figure 1.

cyREST transforms Cytoscape into a REST-based microservice[3] easily callable by workflows coded in REST-enabled languages such as Python, R, and Java. It is complemented by language-specific libraries that simplify Cytoscape access and harmonize native data models with Cytoscape's network model as shown in Figure 2. (REST[4] is short for Representational State Transfer.)

cyREST complements Cytoscape's existing Command Line Tool, where cyREST operates on Cytoscape's data and Command

executes Cytoscape commands. Since its original beta release date in late 2014, cyREST has been downloaded over 1900 times.

In this paper, we explain how cyREST relates to existing Cytoscape-oriented automation solutions, and then describe the design and use of cyREST itself. Finally, we give examples of cyREST's use from applications written in Python (using IPython Notebook) and R (using RStudio).

Existing tools

Several Cytoscape apps and plugins tackle tool interoperability and workflow reproducibility challenges (Table 1), most notably Cytoscape 3's Command core module, but also including plugins deprecated along with Cytoscape 2.

Cytoscape 3's Command Line Tool[5] facilitates task automation via its own domain specific language, which provides access to high-level Cytoscape functions using a separate REST server within Cytoscape. While Command can execute individual commands (e.g., for loading and applying layouts) and sequences of commands (as scripts), it has no provision for accessing the network, style, and visualization information available through cyREST. Command is a complement to cyREST, where the combination greatly improves interoperability between Cytoscape and workflow-capable external tools, which contribute looping and control flow. A workflow can intermix Command and cyREST calls without conflict – they address different capabilities within Cytoscape.

A notable alternative to cyREST is the Cyrface app[6], which allows R and Bioconductor[7] functions to be executed *from* Cytoscape 3, with the results returned *to* Cytoscape 3 – the opposite of a cyREST call. While this approach enables Cytoscape to act as the workflow orchestrator, it requires that a target application act as a server, which often requires idiosyncratic and complex support for each target application. So far, this approach has been taken only for interfacing to R.

Numerous approaches to interoperability were implemented as ScriptingEngine[8]-based plugins for Cytoscape 2, now deprecated. Such plugins were created for executing scripts written in languages
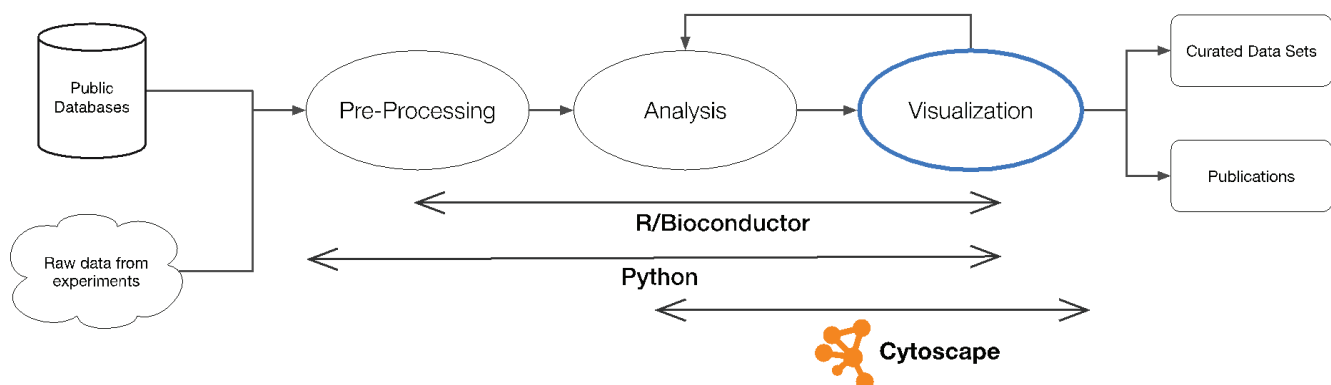
Figure 1. Cytoscape working with other components to create bioinformatic workflows.
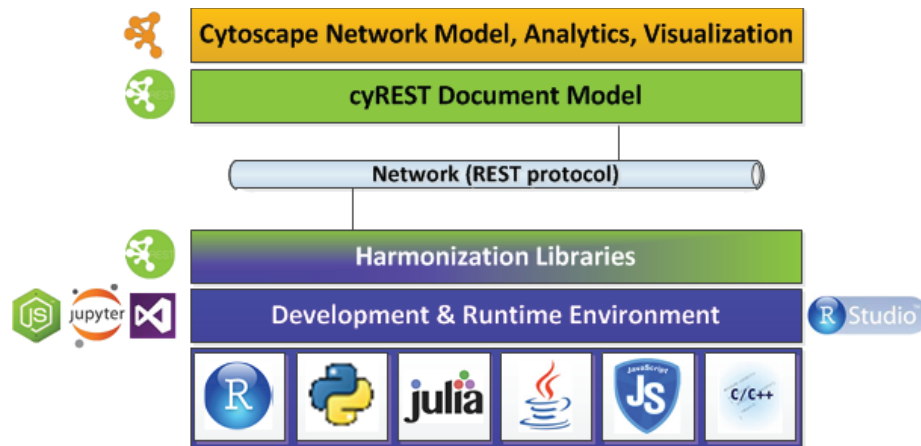
Figure 2. Relationship of Cytoscape to Bioinformatics-oriented Tools and Languages, where cyREST is in green. Harmonization libraries are green and blue, representing an interface between workflow code and cyREST.

Table 1. Existing Cytoscape apps to control Cytoscape from scripts.

Application Name	Description	Supported Version
Command[5]	A domain specific language to automate simple tasks in Cytoscape	3.x and 2.x
Cyrface[6]	An interface to call R packages from Cytoscape	3.x
ScriptingEngine[8]	Exposes Cytoscape API to scripting engines for JVM (JavaScript, Jython, JRuby, etc.)	2.x
CytoscapeRPC[15]	Plugin to call low-level Cytoscape API through XML-RPC	2.x
RCytoscape[17]	Bioconductor package to call Cytoscape from R using CytoscapeRPC	2.x

(e.g., JRuby[9], Jython[10], Groovy[11], Clojure[12], and JavaScript[13]). While these scripts had full access to Cytoscape's comprehensive public API, their tight coupling to the Cytoscape runtime made them difficult to write, debug, and maintain. Because they were built on top of the Java virtual machine (JVM) and shared Cytoscape's JVM, they had little access to increasingly capable and standardized third party native libraries (e.g., SciPy[14] for Python). By contrast, the cyREST approach allows control of Cytoscape by best-of-breed tools and languages running independently as separate processes and leveraging best-of-breed native libraries. Conversely, while plugin implementations could interact with the user via dialog boxes directly within Cytoscape 2, scripts executing in separate processes run within their own windows, disconnected from Cytoscape 3.

Similar to the cyREST approach, the CytoscapeRPC plugin[15] enabled independent scripts (e.g., Python) to create, query, and modify networks and visual styles in Cytoscape 2, but using an XML-RPC[16] protocol instead of REST. Given the rapid adoption of REST conventions and supporting infrastructure, use of XML-RPC is becoming less common. Notably, RCytoscape[17] is a Bioconductor package that leveraged CytoscapeRPC to enabled R applications to control Cytoscape 2. For Cytoscape 3, RCytoscape has been replaced by the RCy3 package in Bioconductor release 3.2, which leverages cyREST instead and is described below.

What is cyREST?

cyREST is a Cytoscape app that exposes Cytoscape network-related data structures and publishing functionality as a microservice callable via a REST protocol by external tools and languages. To date, it offers over 113 API calls, as documented at http://ideker-lab.github.io/cyREST, where each API call accepts JSON-encoded values[18] and returns JSON-encoded results.

Given that most modern tools and languages can call JSON-oriented REST services either directly or through well-vetted libraries, cyREST enables near-universal access to Cytoscape. However, such tools and languages often define data structures well tuned for use with their own specialized libraries that manipulate network-oriented data. To ease and accelerate the programming process,

cyREST provides harmonization libraries designed to make calling cyREST natural and native within a tool or language. Harmonization libraries are described below.

Note that while cyREST enables Cytoscape to act as a service, it is intended to serve only one client application at a time, where the client and Cytoscape run on the same workstation. Cytoscape itself remains capable of working on a single Cytoscape session at a time and maintains a visible window accessible to a user – Cytoscape does not operate in so-called headless mode. As a result, a client application is free to implement a workflow that intentionally sets up a network within Cytoscape so that a user can work further with it.

This section describes both the cyREST design and implementation and the implementation of harmonization libraries. It then presents example workflows created by combining standard data analysis tools with Cytoscape/cyREST.

Design
cyREST is a Cytoscape app that exposes the Cytoscape network data model to external tools and languages. It presents an API based on principles of REST, as do other popular biology-related data services, including those provided by EBI[19]. As a result, cyREST leverages REST facilities in existing tools and languages already built and vetted for use with other REST-based services. The definition and packaging of individual API functions takes advantage of lessons learned in building similar interfaces for Cytoscape 2.

cyREST APIs represent all Cytoscape data objects and functions as resources according to principles of Resource-oriented Design (ROD)[20]. Data objects include networks, tables, and Visual Styles. Functions include applying layout algorithms on networks, updating Visual Styles, and performing statistical analysis. Under REST and ROD, each resource is encoded as a URL where hierarchy is represented as segments within the URL. For example, the URL http://localhost:1234/v1/tables/count can be decomposed into a

REST server (http://localhost), port number (1234), an API version (v1), a resource (tables), and an attribute of the resource (count). So, this URL represents the count of global tables maintained by Cytoscape. Table 2 shows a sampling of resources available under the http://localhost:1234/v1 URL, with a more comprehensive list in the cyREST document at http://idekerlab.github.io/cyREST).

cyREST follows ROD recommendations for sensible mappings between CRUD operations (create, read, update, and delete) and HTTP operations (POST, GET, PUT, DELETE) on data objects. Unless otherwise specified in the cyREST documentation, all HTTP operations accept or return values encoded as JSON. For example, GET http://localhost:1234/v1/networks returns a list of networkIds in an array (e.g., [1,2,3]). GET http://localhost:1234/v1/networks/networkId returns all nodes, edges, tables, and other data relating to network networkId in the Cytoscape.js[21] JSON format.

For functions, ROD provides less guidance for CRUD/HTTP mappings or URL encoding. cyREST addresses this by grouping actions under http://localhost:1234/v1/apply (using GET operations) as illustrated in Table 3.

Implementation
cyREST is implemented as a Cytoscape app written in the Java programming language. It uses the Jersey JAX-RS[22] library to implement the RESTful API, and provides access to data object and function operations as calls to public Cytoscape APIs. Under REST, each client request is phrased as an HTTP command (e.g., GET http://localhost:1234/v1/networks HTTP/1.1) and the reply is returned as a JSON structure.

cyREST uses an embedded Grizzly HTTP server to receive and process client requests, where each HTTP request's URL is mapped to a method in a resource manager class created by cyREST and registered with Grizzly. Each resource method declares the URL it services. When Grizzly receives a REST request, it calls the resource function registered for the URL, which calculates and returns a REST reply.

Table 2. Examples of Cytoscape data resources exposed by cyREST.

URL Segment	Data Resource
networks	list of all networks (as networkId)
networks/networkId/edges	edges within network networkId
networks/networkId/nodes	nodes within network networkId
networks/networkId/tables	tables within network networkId
networks/networkId/views	views of network networkId
networks/networkId/groups	node groupings within network networkId
session	session-wide attributes (e.g., name)
styles	visual styles that can apply to networks
styles/styleName/mappings	values of visual properties for style styleName

Table 3. Examples of Cytoscape function resources exposed by cyREST.

URL Segment	Function Resource
edgebundling/networkId	apply edge bundling to edges in network networkId
fit/networkId	fit network networkId to its window
layouts/algorithm/networkId	apply a layout algorithm to network networkId
styles/styleName/networkId	apply visual style styleName to network networkId

For example, the NetworkResource defines a function that returns the number of Cytoscape networks in the current session, shown in the code fragment below. Note that the fragment defines its associated HTTP command, URL, and JSON output via Java annotations.

```java
@Path("/v1/networks")
public class NetworkResource extends
    AbstractResource {
  @GET
  @Path("/count")
  @Produces(MediaType.APPLICATION_JSON)
  public String getNetworkCount() {
    // Call Cytoscape to get count of network
       - return value as COUNT
    return
        getNumberObjectString(JsonTags.COUNT,
        networkManager.getNetworkSet().size());
}
```

As with most Cytoscape apps, cyREST is initialized in its cyActivator function, which creates resource classes that reference all Cytoscape APIs to be used in servicing client requests. These include factories and managers for networks, network views, visual mapping, layout algorithms, groups, tables, sessions, and others.

The default HTTP port for cyREST is 1234, which can be changed by creating or modify the Cytoscape rest.port property (via Cytoscape's Edit │ Preferences │ Properties dialog). Note that security-conscious workstations should firewall the cyREST port to prevent unintended outside access.

To test for the availability of a cyREST server, use an Internet browser to view the URL http://localhost:1234/v1/, which returns JSON-formatted version information similar to:

```json
{
        "apiVersion":"v1",
        "numberOfCores":4,
        "memoryStatus": {
                "usedMemory":517,
                "freeMemory":1445,
                "totalMemory":1963,
                "maxMemory":6917
        }
}
```

Each cyREST function is exercised and validated before release by a suite of JUnit-based tests.

Harmonization libraries
While most programming languages make calling REST APIs and composing or parsing JSON simple, the data returned by cyREST may not be organized efficiently for ease of use in a particular language or with that language's libraries. To maximize programmer productivity, we provide harmonization libraries (see Figure 2) to perform efficient cyREST calls on one hand, and present an interface easily used by programmers on the other hand. To date, we provide harmonization libraries for Python and R, and we expect to produce others.

py2cytoscape harmonization library for Python
The Python programming language has become popular among scientists and data analysts because of its rich collection of open source data analysis packages and a large developer community. It is an excellent tool for data cleansing, manipulation, analysis, and visualization; its igraph[23], NetworkX[24], and graph-tool[25] packages are useful components in network data analysis workflows. In a workflow, it functions well as a glue that connects multiple heterogeneous computing resources, public databases, and private data files to build data analysis pipelines on workstations and computing clusters.

We created the py2cytoscape library to enable Python-based workflows to easily incorporate Cytoscape functionality by wrapping Python calls to cyREST and performing automatic translations between these packages' data structures and cyREST's JSON. For example, the following code creates a new Cytoscape network by using py2cytoscape calls, and replaces 16 lines that would be necessary when calling cyREST directly – see https://github.com/idekerlab/py2cytoscape/blob/develop/README.md for the direct cyREST calls.

```python
from py2cytoscape.data.cyrest_client import
    CyRestClient

cy = CyRestClient()
network = cy.network.create(name='My Network',
    collection='My network collection')
print(network.get_id())
```

py2cytoscape is open source and is available from the PyPI repository (https://pypi.python.org/pypi/py2cytoscape).

Note that py2cytoscape provides a widget that renders a network in cytoscape.js JSON format and then visualizes the network interactively within a Jupyter/IPython Notebook[26] document, an example of which is at http://nbviewer.ipython.org/github/idekerlab/py2cytoscape/blob/develop/examples/New_wrapper_api_sample.ipynb.

RCy3 harmonization library for R
R is a particularly important platform for biologists because of the complimentary Bioconductor library. We are collaborating with the Bioconductor group to produce the RCy3 harmonization library for R[27], which leverages cyREST to realize native R network visualization, analysis, and publishing functions. Its igraph, graph[28], and RBGL[29] packages are useful components for network data analysis workflows.

Sample workflows
A typical workflow performs data acquisition and integration, analysis, network visualization, and publishing. Often, these steps are performed one at a time by humans executing one discreet tool after another, possibly resulting in high labor costs, low throughput, high error rates, and an inability to reproduce the workflow reliably. In contrast, Figure 1 shows a workflow orchestrated by external tools such as Python and R, which interact with Cytoscape to perform parts of the workflow. As supplementary material, we provide downloadable sample workflows that incorporate and demonstrate cyREST functionality using py2cytoscape and RCy3 harmonization libraries.

Note that Cytoscape/cyREST is designed to run on the same workstation as the workflow that calls it – Cytoscape maintains its own application window, and workflows may find advantage in soliciting users directly within Cytoscape.

Python examples
Our Python-based sample workflows are simple reflections of real world data analysis and visualization pipelines (see Figure 1) and use standard Python packages as much as possible. They are located in https://github.com/idekerlab/cy-rest-python and are viewable using the nbviewer web application (http://nbviewer.ipython.org/) in Jupyter Notebook format.

Some Python packages are more capable or faster than equivalent Cytoscape functions, so the examples use them instead of calling Cytoscape. For example, Pandas[30] prepares and analyzes data by using NumPy[31] and SciPy library for processor-intensive tasks such as community detection.

The examples use the py2cytoscape harmonization libraries to demonstrate efficient cooperation between Python workflows by using NetworkX, igraph, and Cytoscape to integrate and visualize data generated in external tools. They show:

- Data import from multiple data sources (remote/local)
- Reformat and integration
- Statistical network analysis
- Visualization

For instance, the "Import KEGG pathways from web service" example demonstrates a typical biological data integration and visualization process involving KEGG databases[32]:

- Send a disease name query to the KEGG API
- Filter the result and reformat it
- Import disease pathway data directly from the KEGG database
- Visualize pathway data in Cytoscape
- Embed the result as an interactive network diagram in the Jupyter Notebook

This workflow is simple to do with Cytoscape – the alternative would be a custom program or manual, file based operations that are hard to reproduce. With this workflow script, collaborators or reviewers can easily execute the same process on their environment, which is essential for reproducible scientific research.

R examples
For network analysis and visualization, igraph is an important and much used package by R users, and our sample R workflows (https://github.com/idekerlab/cy-rest-R) use it to complement the graph analysis features in Cytoscape.

In our Workflow 1 example, we scripted typical network visualization techniques using igraph's graph analysis functions and Cytoscape's data visualization features. First, we used igraph to detect community structure using a fast greedy modularity optimization algorithm[33], and we calculated basic statistics of the network, including PageRank[34] and betweenness centrality[35]. Our R code calls Cytoscape to create the resulting network, set properties for both layout and visual mapping, and generate an interactive network visualization. Output of this workflow helps users to visually understand the basic structure of the network (see Figure 3, which shows community structures color coded and used as weights for the Kamada-Kawai layout algorithm[36]).

In many cases, users apply an automatic layout algorithm early in a workflow to visually check the overall structure of a network. However, such layouts are often based on a simple force simulation and tend to produce uninformative "hairballs." Our example illuminated network sub structures by using a community detection algorithm and igraph's statistical analysis algorithms and visual styling. The alternative would be manual operation of both R and Cytoscape, which is laborious and error prone even for proficient users, and which is not reusable for subsequent networks.

Future development
In this paper, we demonstrated Python- and R-based workflow examples. In the near future, we expect to demonstrate cyREST usage in MATLAB and JavaScript (via Node.js[38]). While the cyREST and Command apps implement different automation features, we expect to unify the two APIs through a common implementation library in the Cytoscape core in the next Cytoscape release. Existing Cytoscape implementations manage a single Cytoscape
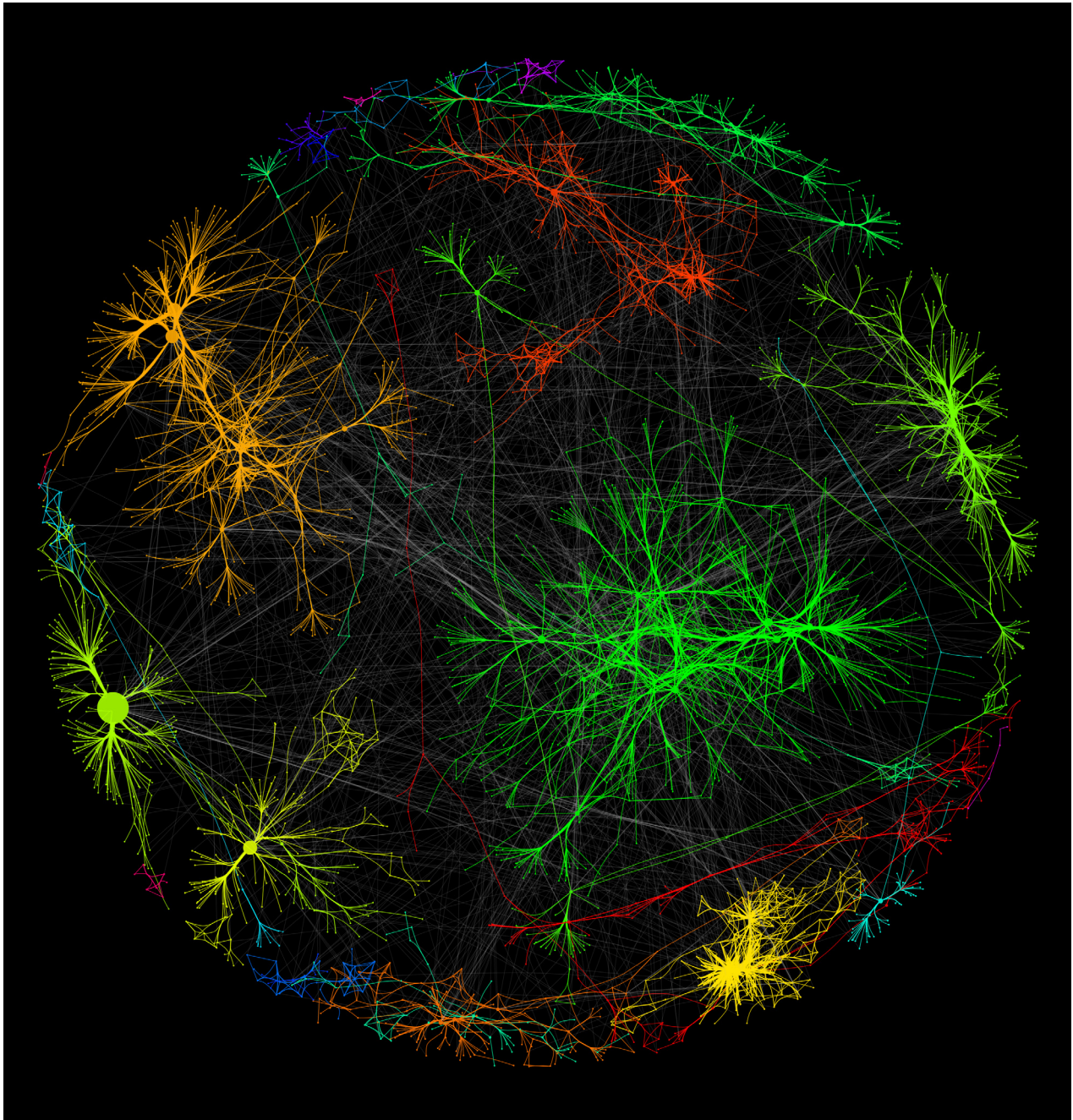
Figure 3. Yeast network[37] visualization with sample R script and Cytoscape.

session on behalf of a single user, can produce screen visualizations, and can potentially solicit user input even while under the control of cyREST. Future versions of Cytoscape will run headlessly and service multiple sessions simultaneously.

Summary

Cytoscape is a highly popular desktop application for network biology analysis, visualization, and publication. The cyREST app extends Cytoscape into the realm of reproducible and high volume bioinformatic workflows by exposing a RESTful API that recasts Cytoscape as a visualization and rendering microservice. Using cyREST, data acquisition and analysis workflows previously limited to low quality (if any) visualizations can now leverage Cytoscape's substantial library of network layouts, visualization features, and rendering options. Similarly, cyREST's seamless integration with research and publication tools such as IPython/Jupyter Notebook

improves individual researcher productivity by avoiding the need to manually operate Cytoscape.

Because it presents a RESTful interface, cyREST benefits can be realized in workflows built in most modern programming languages, and represents a significant contribution to productivity and reproducibility in data driven biology.

Software availability
CyREST software is available from the Cytoscape App Store: http://apps.cytoscape.org/apps/cyrest

Latest source code of cyREST: https://github.com/idekerlab/cyREST

Full REST API v1 document: http://idekerlab.github.io/cyREST/

Py2cytoscape is in beta and is installable from PyPI repository: https://pypi.python.org/pypi/py2cytoscape

Py2cytoscape source code: https://github.com/idekerlab/py2cytoscape

Python sample workflows in Jupyter Notebook format: https://github.com/idekerlab/cy-rest-python

R sample workflows: https://github.com/idekerlab/cy-rest-R

License for cyREST, py2cytoscape, and all example workflows: MIT: http://opensource.org/licenses/MIT

RCy3 source code: https://github.com/tmuetze/Bioconductor_RCy3_the_new_RCytoscape

References

1. Shannon P, Markiel A, Ozier O, *et al.*: **Cytoscape: a software environment for integrated models of biomolecular interaction networks.** *Genome Res.* 2003; **13**(11): 2498–2504.
   PubMed Abstract | **Publisher Full Text** | **Free Full Text**
2. Pérez F, Granger BE: **IPython: a system for Interactive Scientific Computing.** *Comput Sci Eng.* 2007; **9**(3): 21–29.
   Publisher Full Text
3. Lewis J, Fowler M: **Microservices.** 2014.
   Reference Source
4. Fielding RT, Taylor RN: **Principled design of the modern web architecture.** *ACM Trans Internet Technol.* 2002; **2**(2): 115–150.
   Publisher Full Text
5. **Command Tool.** 2013.
   Reference Source
6. Gonçalves E, Saez-Rodriguez J: *Cyrface*: **An interface from *Cytoscape* to *R* that provides a user interface to *R* packages. [v1; ref status: indexed, http://f1000r.es/1tv].** *F1000Res.* 2013; **2**: 192.
   PubMed Abstract | **Publisher Full Text** | **Free Full Text**
7. Gentleman RC, Carey VJ, Bates DM, *et al.*: **Bioconductor: open software development for computational biology and bioinformatics.** *Genome Biol.* 2004; **5**(10): R80.
   PubMed Abstract | **Publisher Full Text** | **Free Full Text**
8. **ScriptingPlugins.** December 2009.
   Reference Source
9. **RubyScriptingEngine.** October 2012.
   Reference Source
10. **PythonScriptingEngine.** October 2012.
    Reference Source
11. **GroovyEngine.** August 2011.
    Reference Source
12. **ClojureEngine.** August 2011.
    Reference Source
13. **JavaScriptEngine.** August 2011.
    Reference Source
14. Jones E, Oliphant T, Peterson P: **{SciPy}: Open source scientific tools for {Python}.** 2001.
15. **CytoscapeRPC.** October 2011.
    Reference Source
16. Cerami E: **Web services essentials: distributed applications with XML-RPC, SOAP, UDDI & WSDL.** "O'Reilly Media, Inc.", 2002.
    Reference Source
17. Shannon PT, Grimes M, Kutlu B, *et al.*: **RCytoscape: tools for exploratory network analysis.** *BMC Bioinformatics.* 2013; **14**(1): 217.
    PubMed Abstract | **Publisher Full Text** | **Free Full Text**
18. **Json.**
    Reference Source
19. **EMBL-EBI Web Services.** April 2015.
    Reference Source
20. Sletten B: **Resource-Oriented Architecture: The Rest of Rest.** December 2009.
    Reference Source
21. **Cytoscape.js.**
    Reference Source

22. **Apache CXF**.
     Reference Source

23. Csardi G, Nepusz T: **The igraph software package for complex network research.** *InterJournal, Complex Systems.* 2005; **1695**(5): 1–9.
     Reference Source

24. AA Hagberg, Schult DA, Swart P: **Exploring Network Structure, Dynamics, and Function using NetworkX**. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008).* 2008; **2008**: 11–16.
     Reference Source

25. **graph-tool: Efficient network analysis**.
     Reference Source

26. **Jupyter**.
     Reference Source

27. **The R Project for Statistical Computing**.
     Reference Source

28. **graph**.
     Reference Source

29. **RBGL**.
     Reference Source

30. McKinney W: **pandas: a Python data analysis library**. 2015.
     Reference Source

31. McKinney W: **Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.** " O'Reilly Media, Inc.", 2012.
     Reference Source

32. Kawashima S, Katayama T, Sato Y, *et al.*: **KEGG API A web service using SOAP/WSDL to access the KEGG system.** *Genome Informatics.* 2003; **14**: 673–674.
     Reference Source

33. Clauset A, Newman MEJ, Moore C: **Finding community structure in very large networks.** *Phys Rev E Stat Nonlin Soft Matter Phys.* 2004; **70**(6 Pt 2): 66111.
     PubMed Abstract | **Publisher Full Text**

34. Page L, Brin S, Motwani R, *et al.*: **The PageRank Citation Ranking: Bringing Order to the Web.** 1999.
     Reference Source

35. Freeman LC: **Centrality in social networks conceptual clarification.** *Social Networks.* 1978–1979; **1**(3): 215–239.
     Publisher Full Text

36. Kamada T, Kawai S: **An algorithm for drawing general undirected graphs.** *Inf Process Lett.* 1989; **31**(1): 7–15.
     Publisher Full Text

37. Lee TI, Rinaldi NJ, Robert F, *et al.*: **Transcriptional regulatory networks in *Saccharomyces cerevisiae*.** *Science.* 2002; **298**(5594): 799–804.
     Pubmed Abstract | **Publisher Full Text**

38. Tilkov S, Vinoski S: **Node.js: Using Javascript to Build High-Performance Network Programs.** *IEEE Internet Computing.* 2010; 6: 80–83.
     Publisher Full Text

Open Peer Review

Current Referee Status:

Guanming Wu

Ontario Institute for Cancer Research, Toronto, ON, Canada

The authors describe a Cytoscape app, CyREST, which exposes core Cytoscape functions as REST APIs for external software components to process network related data sets in automatic and reproducible workflows built using almost any programming languages. Users of workflows can visualize network data in Cytoscape via its powerful visualization features. The accompanied harmonization libraries for Python and R make the use of CyREST much easier and simpler. The manuscript is well organized, and the described app should be highly valuable for users working with big data related to networks for analysis and visualization.

Minor comments:

1. Figure 1: The difference between R/Bioconductor and Python is not clear to me. The double-arrowed line for Python is longer than R/Bioconductor, but I am not sure what extra is covered by Python. Python is a full-fledged programming language and capable for anything. Should some specific packages listed for Python? Further, probably the line for Cytoscape should be extended to cover two boxes for Curated Data Sets and Publications.

2. Figure 2: Will it be better to call "Network (REST protocol)" just as "REST Protocol" to avoid confusing with other uses of "network" in the manuscript?

3. What is the development status of RCy3? It will be nicer to indicate its status somewhere. From the manuscript, it seems that this is still in work-in-progress?

Typos:

1. In abstract: e.g. Python **and R**, JavaScript, and C#

2. There are several typos in this page: https://github.com/idekerlab/cy-rest-R, e.g. **Cytoscaep**, **Lesons**, etc.

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Referee Report 10 August 2015

Sergey Nepomnyachiy

Department of Computer Science & Engineering, Polytechnic Institute of New York University, Brooklyn, NY, USA

The authors present a Cytoscape app that turns it into a REST server. Most core functionality of Cytoscape can now be triggered by external agents using HTTP requests, rather than by point-and-clicking UI. This enables workflow automation and distribution that was hardly possible before. The fact that the protocol is HTTP based, allows working with it programmatically using any modern language. CyREST comes with libraries for Python and R that make the communication with Cytoscape through REST API even simpler, by enveloping the calls in methods and wrapping the returning JSON in objects.

Perhaps in the future work section the authors could shed some light on the plans to allow users extending the set of REST handlers. The virtues of Cytsocape are beyond the core functionality and currently there is no way for a Cyto-app developer to map her own functions to REST (those are hard-wired in the code of CyREST app). A simple registry for binding 3rd party Cyto-app functions to URLs at runtime could bring the community to bring their on plug-in REST.

Possible typos

page 6:
The default HTTP port for cyREST is 1234, which can be changed by creating or *modify* the Cytoscape rest.port property

page 7:
This workflow is simple to do with Cytoscape – the alternative would be a custom program *or manual, file based operations* that are hard to reproduce.

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.