SOFTWARE TOOL ARTICLE

RCy3: Network biology using Cytoscape from within R [version 1; peer review: 1 approved]

Julia A. Gustavsen1, Shraddha Pai2, Ruth Isserlin2, Barry Demchak3, Alexander R. Pico4

1University of British Columbia, Vancouver, B.C., Canada
2The Donnelly Centre, University of Toronto, Toronto, ON, Canada
3Department of Medicine, University of California at San Diego, La Jolla, CA, USA
4Institute of Data Science and Biotechnology, Gladstone Institutes, San Francisco, CA, USA

Abstract

RCy3 is an R package in Bioconductor that communicates with Cytoscape via its REST API, providing access to the full feature set of Cytoscape from within the R programming environment. RCy3 has been redesigned to streamline its usage and future development as part of a broader Cytoscape Automation effort. Over 100 new functions have been added, including dozens of helper functions specifically for intuitive data overlay operations. Over 40 Cytoscape apps have implemented automation support so far, making hundreds of additional operations accessible via RCy3. Two-way conversion with networks from \textit{igraph} and \textit{graph} ensures interoperability with existing network biology workflows and dozens of other Bioconductor packages. These capabilities are demonstrated in a series of use cases involving public databases, enrichment analysis pipelines, shortest path algorithms and more. With RCy3, bioinformaticians will be able to quickly deliver reproducible network biology workflows as integrations of Cytoscape functions, complex custom analyses and other R packages.

Keywords

Bioconductor, Cytoscape, Automation, Scripting, Network biology

This article is included in the Bioconductor gateway.

This article is included in the RPackage gateway.
Introduction

In the domain of biology, network models serve as useful representations of interactions, whether social, neural or molecular. Since 2003, Cytoscape has provided a free, open-source software platform for network analysis and visualization that has been widely adopted in biological and biomedical research fields. The Cytoscape platform supports community-developed extensions, called apps, that can access third-party databases, offer new layouts, add analytical algorithms, support additional data types, and much more.

In 2011, the CytoscapeRPC app was created to enable R-based workflows to exercise Cytoscape v2 functionality via functions in the corresponding RCytoscape R package over XML-RPC communications protocols. In 2015, the CyREST app was created to enable R-based workflows to exercise Cytoscape v3 functionality. This was achieved by the first version of the RCy3 R package, which re-implemented much of RCytoscape’s organization, data structures and syntax over REST communications protocols.

Here, we describe version 2.0 of the RCy3 package, which is better aligned with Cytoscape’s CyREST API. We have rewritten every function, deprecated 43 functions and added over 100 new functions. This work provides a more intuitive and productive experience for Cytoscape users learning about the RCy3 package, and it positions RCy3 to take advantage of future Cytoscape Automation development and evolution. The goal of this paper is to describe the implementation and operation of the updated RCy3 package and to provide detailed use cases relevant to network biology applications in common and advanced bioinformatics workflows.

Methods

Design and Implementation

RCy3 is a component of Cytoscape Automation. At the core of Cytoscape Automation is CyREST, which implements an interface to the Cytoscape Java application via the REST protocol. A collection of GET, POST, PUT and DELETE operations practically cover the complete feature set of the Cytoscape desktop software. Additional features, including those provided by user-installed Cytoscape apps, are covered by a separate command-line interface called Commands (Figure 1). For version 2.0, we redesigned the RCy3 package to parallel CyREST and Commands APIs to standardize the syntax and organization of its functions and streamline its future development. RCy3 functions are grouped into categories to aid the parallel development with Cytoscape Automation APIs and to facilitate navigation and comprehension of the overall package (Table 1).

The basics

All RCy3 functions are ultimately implemented as calls to a small set of utility functions that execute the CyREST or Commands REST protocol (e.g., cyrestGET and commandsPOST). The internals of these functions handle the composition of operations and parameters to be sent via http functions to CyREST, as well as the processing of results from JSON to appropriate R objects using RJSONIO.

Figure 1. The Cytoscape Automation stack. The Java desktop version of Cytoscape (bottom) supports Commands (red) and CyREST (green) interfaces for scripting and automation either directly (blue) or via R and Python harmonization libraries (purple).
<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apps</td>
<td>Inspecting and managing apps for Cytoscape</td>
<td>installApp, disableApp, getInstalledApps</td>
</tr>
<tr>
<td>Collections</td>
<td>Getting information about network collections</td>
<td>getCollectionList, getCollectionNetworks</td>
</tr>
<tr>
<td>Commands</td>
<td>Constructing any arbitrary CyREST API or Commands API method via standard GET, PUT, POST and DELETE protocols</td>
<td>cyrestGET, commandsPOST, cyrestAPI</td>
</tr>
<tr>
<td>CyNDEEx</td>
<td>Communicating with NDEEx from within Cytoscape</td>
<td>importNetworkFromNDEEx, exportNetworkToNDEEx</td>
</tr>
<tr>
<td>Cytoscape System</td>
<td>Checking Cytoscape System information, including versions and memory usage</td>
<td>cytoscapePing, cytoscapeVersionInfo</td>
</tr>
<tr>
<td>Filters</td>
<td>Selecting nodes and edges based on filter criteria</td>
<td>createDegreeFilter, createColumnFilter</td>
</tr>
<tr>
<td>Groups</td>
<td>Working with groups in Cytoscape</td>
<td>createGroup, collapseGroup</td>
</tr>
<tr>
<td>Layouts</td>
<td>Performing layouts in addition to getting and setting layout properties</td>
<td>layoutNetwork, getLayoutNames</td>
</tr>
<tr>
<td>Networks</td>
<td>Creating and managing networks and retrieving information on networks, nodes and edges</td>
<td>createNetworkFrom*, createFromNetwork, getNetworkSuid, exportNetwork, getAllNodes, getEdgeCount, getFirstNeighbors</td>
</tr>
<tr>
<td>Network Selection</td>
<td>Manipulating selection of nodes and edges in networks</td>
<td>selectNodes, invertNodeSelection, selectFirstNeighbors</td>
</tr>
<tr>
<td>Network Views</td>
<td>Performing view operations in addition to getting and setting view properties</td>
<td>getCurrentView, fitContent, exportImage, toggleGraphicsDetails</td>
</tr>
<tr>
<td>Session</td>
<td>Managing Cytoscape sessions, including save, open and close</td>
<td>openSession, saveSession, closeSession</td>
</tr>
<tr>
<td>Style Bypasses</td>
<td>Setting and clearing bypass values for visual properties</td>
<td>setColorBypass, setEdgeLineStyleBypass, hideNodes</td>
</tr>
<tr>
<td>Style Defaults</td>
<td>Getting and setting default values for visual properties</td>
<td>setNodeShapeDefault, setEdgeLineWidthDefault</td>
</tr>
<tr>
<td>Style Dependencies</td>
<td>Getting and setting style dependencies</td>
<td>lockNodeDimensions</td>
</tr>
<tr>
<td>Style Mappings</td>
<td>Defining mappings between table column values and visual properties</td>
<td>mapVisualProperty, updateStyleMapping, setNodeSizeMapping, setEdgeColorMapping</td>
</tr>
<tr>
<td>Style Values</td>
<td>Retrieving current values for visual properties</td>
<td>getWidth, getColor, getNetworkZoom</td>
</tr>
<tr>
<td>Styles</td>
<td>Managing styles and retrieving general lists of properties relevant to multiple style modes</td>
<td>createVisualStyle, exportVisualStyles, getArrowShapes</td>
</tr>
<tr>
<td>Tables</td>
<td>Managing table columns and table column functions, like map and rename, as well as loading and extracting table data in Cytoscape</td>
<td>getTableColumns, renameTableColumn, loadTableData, mapTableColumn</td>
</tr>
<tr>
<td>Tools</td>
<td>Performing actions found in the Tools Menu in Cytoscape</td>
<td>cybrowserDialog, diffusionBasic</td>
</tr>
<tr>
<td>User Interface</td>
<td>Controlling the panels in the Cytoscape user interface</td>
<td>hidePanel, floatPanel</td>
</tr>
</tbody>
</table>
In most RCy3 functions there is an optional argument for `base.url`. This is the URL used by RCy3 to connect to the Cytoscape desktop application via CyREST, and it defaults to "http://localhost:1234/v1". The default CyREST port is 1234, and it can be changed in Cytoscape through Edit/Preferences/Properties or by command-line (see CyREST setup guide). If you change the CyREST port, you should reflect the change in the `base.url` argument per function call or change each function’s default value using the `default` package.

The second most common argument in RCy3 functions is `network`. If left as `NULL` (default), the currently selected network in the Cytoscape application is referenced. Network name or SUID (session unique identifier) can thus be explicitly specified or inferred from the current state of the application. The current network can also be controlled and retrieved by `setCurrentNetwork` and `getCurrentNetwork`. Given a `base.url` and `network` (when needed), the majority of RCy3 functions simply validate parameters and construct arguments in order to call one of the `cyrest*` or `commands*` functions.

The `commandsRun` function is a special RCy3 function that allows users to directly issue commands via Cytoscape’s command-line syntax (e.g., "node list network=current"), including commands implemented by Cytoscape app developers (see Use cases). This single function can perform hundreds of operations made available by both Cytoscape and automation-enabled apps. Over 40 of these RCy3-supported apps are currently registered in the Cytoscape App Store. The `cyrestAPI` and `commandsAPI` open interactive Swagger documentation for the CyREST and Commands programmatic interfaces. Cytoscape Automation can be performed via these Swagger web pages. The same operations and parameters are supported by the `cyrest*` and `commands*` functions in RCy3. Command-line syntax can also be run from the Automation panel in Cytoscape, at manual.cytoscape.org/en/stable/Command_Tool.html.

**Generic and specific**

The primary goal of RCy3 is to provide wrappers for every feature made available by CyREST and Commands. However, we also have a secondary goal of providing useful and intuitive functions for common workflows in R. So, in addition to the generic functions implemented to parallel the CyREST and Commands APIs, we have also implemented sets of specific helper functions.

As an example, consider the common Cytoscape operation of mapping network data values to visual style properties. CyREST has a POST endpoint for `/styles/{style name}/mappings` that takes a JSON data structure defining the mapping. We implemented `updateStyleMapping` which takes a `style.name` and mapping arguments and sends them out via `cyrestPOST`. We also implemented `mapVisualProperty` to help construct the mapping argument. With these generic functions one can perform any of the hundreds of visual style mappings supported by Cytoscape, including new ones added in the future. However, these functions are not simple to use, requiring knowledge of specific property names, like "NODE_FILL_COLOR", and mapping data structures. To simplify usage for common situations, we therefore also implemented specific functions for over a dozen of the most commonly used mappings (e.g., `setNodeColorMapping`). With autocomplete in tools like RStudio, after just typing `setNode...` a script author is presented with a series of intuitively named functions with obvious arguments.

**Networks in R**

Networks are a popular visualization option in R often implemented as graph models by `igraph` and Bioconductor’s `graph` (i.e., graphNEL). RCy3 can create networks in Cytoscape from either `igraph`, `graphNEL` or dataframe objects (`createNetworkFrom*`). Likewise, `igraph` and `graphNEL` objects can be created from networks (`create*FromNetwork`), and dataframes from node and edge tables in Cytoscape (`getTableColumns`).

In the case of `createNetworkFromDataFrames`, two dataframes are accepted as arguments, one for nodes and one for edges. The `nodes` dataframe must include a column named "id", and the `edges` dataframe must include "source" and "target" columns. Additional columns are imported as node and edge attributes into Cytoscape. The function can also work with just one dataframe. If a dataframe of only edges is passed to `createNetworkFromDataFrames`, then a connected network will be created with all of the nodes. If a dataframe of only nodes is passed, then a network with no connections, only nodes, will be created.

RCy3 can also import network file formats supported by Cytoscape natively (e.g., SIF, xGMLML and CX) and via user-installed apps (e.g., GPML and adjacency matrices). With these functions RCy3 can
interoperate with any other Bioconductor packages that deal with networks in a standardized manner, providing advanced network visualization options and advanced network analytics from the Cytoscape ecosystem (see Table 2).

**Operation**

In order to work with RCy3 you must have Cytoscape v3.7 or later installed and running. Cytoscape can be installed from cytoscape.org. The RCy3 package can be installed from Bioconductor:

```r
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("RCy3")
library(RCy3)
```

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
<th>Network model</th>
<th>RCy3</th>
</tr>
</thead>
<tbody>
<tr>
<td>graph</td>
<td>Essential graph data structures</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>RBGL</td>
<td>Graph algorithms contained in the BOOST library</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>Rgraphviz</td>
<td>Plotting with graphviz library</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>pathview</td>
<td>Data integration and visualization on pathways</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>KEGGgraph</td>
<td>Interface between KEGG pathways and graphs</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>Gostats</td>
<td>Tools for interacting with Gene Ontology</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>TCGAbiolinks</td>
<td>Integrative analysis with GDC data</td>
<td>igrap</td>
<td>✔</td>
</tr>
<tr>
<td>graphite</td>
<td>Graph objects from pathway databases</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>minet</td>
<td>Infers mutual information networks</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>SPIA</td>
<td>Signaling pathway impact analysis</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>RDVIDWebService</td>
<td>Retrieves data from DAVID</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>STRINGdb</td>
<td>Retrieves interaction networks from STRING</td>
<td>igrap</td>
<td>✔</td>
</tr>
<tr>
<td>GENIE3</td>
<td>Gene regulatory network inference</td>
<td>adjacency matrix</td>
<td>✔</td>
</tr>
<tr>
<td>BioNet</td>
<td>Integrated network analysis and module detection</td>
<td>igrap, graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>GDCCRNATools</td>
<td>Integrative analysis of IncRNA, mRNA and miRNA dataframes</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>CEMiTool</td>
<td>Analysis of coexpression gene modules</td>
<td>adjacency matrix</td>
<td>✔</td>
</tr>
<tr>
<td>Linnorm</td>
<td>Linear model and normality based transformation</td>
<td>igrap</td>
<td>✔</td>
</tr>
<tr>
<td>RedeR</td>
<td>Interactive visualization of nested networks</td>
<td>igrap</td>
<td>✔</td>
</tr>
<tr>
<td>qqgraph</td>
<td>Estimate gene and eQTL networks</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>RTN</td>
<td>Reconstruction of transcriptional networks</td>
<td>igrap</td>
<td>✔</td>
</tr>
<tr>
<td>rWikiPathways</td>
<td>Access WikiPathways web services</td>
<td>GPML</td>
<td>✔</td>
</tr>
<tr>
<td>hypergraph</td>
<td>Representing and manipulating hypergraphs</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>apComplex</td>
<td>Bipartite graphs of complexes from AP-MS data</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>cellTree</td>
<td>scRNA-seq data as a hierarchical tree structure</td>
<td>igrap</td>
<td>✔</td>
</tr>
<tr>
<td>FGenet</td>
<td>Functional gene networks from enrichment analyses</td>
<td>igrap</td>
<td>✔</td>
</tr>
<tr>
<td>SpidermiR</td>
<td>Integrative network analysis with miRNA data</td>
<td>igrap</td>
<td>✔</td>
</tr>
<tr>
<td>paxtoolsr</td>
<td>Access to BioPAX models in Pathway Commons</td>
<td>igrap, SIF</td>
<td>✔</td>
</tr>
<tr>
<td>netbibov</td>
<td>Visualization of large biological networks</td>
<td>igrap</td>
<td>✔</td>
</tr>
<tr>
<td>rsbml</td>
<td>Link to libsbml for SBML parsing</td>
<td>graphNEL</td>
<td>✔</td>
</tr>
<tr>
<td>ndexr</td>
<td>Interface to NDEx servers</td>
<td>igrap, CX</td>
<td>✔</td>
</tr>
</tbody>
</table>
Launch Cytoscape and keep it running whenever using RCy3. Confirm that you have everything installed and that RCy3 is communicating with Cytoscape via CyREST:

```r
  cytoscapePing()
  #[1] "You are connected to Cytoscape!"
```

As with any R package, one can access the documentation and browse over a dozen vignettes included in the RCy3 package:

```r
  help(package=RCy3)
  browseVignettes("RCy3")
```

**Use cases**

The following sections demonstrate a variety of common and advanced network biology use cases as runnable R code snippets. The code for these use cases is also available as an online Rmd notebook and Rmd file in the Cytoscape Automation repository (see Data availability). The first set focuses on fundamental Cytoscape operations that are common to most use cases:

- Loading networks (from R objects, Cytoscape files and public databases)
- Visualizing network data
- Filtering by node degree or data
- Saving and exporting networks

Additionally, there are examples that demonstrate analytical workflows, relying not only on Cytoscape, but also on Cytoscape apps and other R packages:

- Building maps of enrichment analysis results using EnrichmentMap and AutoAnnotate
- Visualizing integrated network analysis using BioNet
- Performing advanced graph analytics using RBGL

**Loading Networks.** Networks come in all shapes and sizes, in multiple formats from multiple sources. The following code snippets demonstrate just a few of the myriad ways to load networks into Cytoscape using RCy3.

From R objects...

```r
  # From graph objects (graphNEL)
  g <- makeSimpleGraph()
  createNetworkFromGraph(g)
  ## And round-trip back from Cytoscape to graph
  g2 <- createGraphFromNetwork()

  # From igraph objects
  library(igraph)
  ig <- igraph::make_graph("Zachary")
  createNetworkFromIgraph(ig)
  ## And round-trip back from Cytoscape to igraph
  ig2 <- createIgraphFromNetwork()
  ## Note that the Cytoscape model infers directionality

  # From dataframes
  nodes <- data.frame(id=c("node 0","node 1","node 2","node 3"),
                      group=c("A","A","B","B"), #categorical strings
                      score=as.integer(c(20,10,15,5)), #integers
                      stringsAsFactors=FALSE)
  edges <- data.frame(source=c("node 0","node 0","node 0","node 2"),
                      target=c("node 1","node 2","node 3","node 3"),
                      interaction=c("inhibits","interacts","activates","interacts"), #optional
                      weight=as.numeric(c(5.1,3.0,5.2,9.9)), #numerics
                      stringsAsFactors=FALSE)
  createNetworkFromDataFrames(nodes, edges)
```
From Cytoscape-supported file formats...

```r
# From Cytoscape session files
openSession() # default file = galFiltered.cys

# From local network files
importNetworkFromSession() # default file = galFiltered.sif
## Supported file formats: SIF, GML, xGMML, graphML, CX, plus

# From NDEx (https://ndexbio.org), the network database
importNetworkFromNDEx("5be85817-1e5f-11e8-b939-0ac135e8bacf")
## Account information or accessKey are required arguments only
## when accessing private content
```

From public databases via Cytoscape apps...

```r
# From STRING (https://string-db.org), starting with a list of genes/proteins
installApp("stringApp") # http://apps.cytoscape.org/apps/stringapp
gene.list <- c("T53","AKT1","CDKN1A")
gene.str <- paste(gene.list, collapse = ",")
string.cmd <- paste("string protein query cutoff=0.99 limit=40 query",
                  gene.str, sep = ",")
commandsRun(string.cmd)

# From WikiPathways (https://wikipathways.org), starting with a keyword
library(rWikiPathways) # install from Bioconductor
installApp("WikiPathways") # http://apps.cytoscape.org/apps/wikipathways
keyword <- "glioblastoma"
gbm.pathways <- findPathwaysByText(keyword)
gbm.wpid <- gbm.pathways[[1]]$id # let’s just take the first one
wikipathways.cmd <- paste("wikipathways import-as-pathway id",
                          gbm.wpid, sep = ",")
commandsRun(wikipathways.cmd)
```

Visualizing data on networks. Cytoscape excels at generating publication-quality network visualization with data overlays. This vignette demonstrates just one of the hundreds of visual style mapping options using RCy3.

```r
# Load sample network
closeSession(FALSE) # clears all session data without saving
importNetworkFromSession() # default file = galFiltered.sif

# Load sample data
csv <- system.file("extdata","galExpData.csv", package="RCy3")
data <- read.csv(csv, stringsAsFactors = FALSE)
loadTableData(data, data.key.column = "name")

# Prepare data-mapping points
gal80Rexp.min <- min(data$gal80Rexp, na.rm = T)
gal80Rexp.max <- max(data$gal80Rexp, na.rm = T)
## For a balanced color gradient, pick the largest absolute value
gal80Rexp.max.abs <- max(abs(gal80Rexp.min), abs(gal80Rexp.max))

# Set node color gradient from blue to white to red
setNodeColorMapping('gal80Rexp', c(-gal80Rexp.max.abs, 0, gal80Rexp.max.abs),
                    c('#5577FF','#FFFFFF', '#FF7755'), default.color = '#999999')
```
Filtering networks by degree and by data. Network topology and associated node or edge data can be used to make selections in Cytoscape that enable filtering and subnetworking. The filters are added to the Select tab in the Control Panel of Cytoscape’s GUI and saved in session files.

```r
# Load demo Cytoscape session file
goToSession() # default file = galFiltered.cys
net.suid <- getNetworkSuid() # get SUID for future reference

# Filter for neighbors of high degree nodes
createDegreeFilter(filter.name = "degree filter",
                   criterion = c(0,9),
                   predicate = "IS_NOT_BETWEEN")
selectFirstNeighbors() # expand selection to first neighbors
createSubnetwork(subnetwork.name = "first neighbors of high degree nodes")

# Filter for high edge betweenness
createColumnFilter(filter.name = "edge betweenness",
                   type = "edges",
                   column = "EdgeBetweenness",
                   4000,
                   "GREATER_THAN",
                   network = net.suid)
createSubnetwork(subnetwork.name = "high edge betweenness")
```

**Saving and exporting networks.** There are local and cloud-hosted options for saving and sharing network models and images. The Cytoscape session file (CYS) includes all networks, collections, tables and styles. It retains every aspect of your session, including the size of the application window. Network and image exports include only the currently active network. Export to NDEx requires account information you can obtain from ndexbio.org. Files are saved to the current working directory by default, unless a full path is provided.

```r
# Saving sessions
goToSession("MySession") #.cys
## Leave filename blank to update previously saved session file

# Exporting images and networks
goToSession()
exportNetwork() #.sif
## Optionally specify filename, default is network name
## Optionally specify type: SIF(default), CX, cyjs, graphML, NNF, SIF, xGML
exportImage() #.png
## Optionally specify filename, default is network name
## Optionally specify type: PNG (default), JPEG, PDF, PostScript, SVG

# Exporting to NDEx, a.k.a. “Dropbox” for networks
exportNetworkToNDEx(username, password, TRUE)
## Account information (username and password) is required to upload
## Use updateNetworkInNDEx if the network has previously been uploaded
```

Building maps of enrichment analysis results. This workflow illustrates how to plot an annotated map of enrichment results using the EnrichmentMap Pipeline Collection of apps in Cytoscape. An enrichment map is a network visualization of related genesets in which nodes are gene sets (or pathways) and edge weight indicates the overlap in member genes. Following the construction of the enrichment map, AutoAnnotate clusters redundant gene sets and uses WordCloud to label the resulting cluster (Figure 2). The code uses the Commands interface to invoke EnrichmentMap and AutoAnnotate apps. After installing apps, run commandsAPI() to open the live Swagger documentation to browse and execute command-line syntax.
installApp("EnrichmentMap Pipeline Collection") # installs 4 apps

# Optionally browse commands for enrichmentmap
commandsAPI()

# Download sample gmt file of human pathways
gmt.file <- "rcy3_enrichmentmap.gmt"
download.file(file.path("http://download.baderlab.org/EM_Genesets",
    "September_01_2019/Human/symbol/Pathways",
    "Human_WikiPathways_Sep_01_2019_symbol.gmt"),
    gmt.file)

# Run EnrichmentMap build command
em_command <- paste('enrichmentmap build analysisType="generic"
    "gmtFile="
    paste(getwd(), gmt.file, sep="/"),
    "pvalue="
    1,
    "qvalue="
    1,
    "similaritycutoff="
    0.25,
    "coefficients="
    "JACCARD"
)
print(em_command)
commandsGET(em_command)

# Run the AutoAnnotate command
aa_command <- paste("autoannotate annotate-clusterBoosted"
    "clusterAlgorithm=MCL",
    "labelColumn=EnrichmentMap::GS_DESCR",
    "maxWords=3")

print(aa_command)
commandsGET(aa_command)

# Annotate a subnetwork
createSubnetwork(c(1:4),"__mclCluster")
commandsGET(aa_command)

---

**Figure 2.** Annotated enrichment map of pathways. A few of the largest clusters of pathways from WikiPathways are displayed as a network with WordCloud-based labels annotating groups (yellow areas). The size of the labels correspond to the size of the groups.
Visualizing integrated network analysis using BioNet. The BioNet package implements analytical methods to perform integrated network analysis, for example, of gene expression data and clinical survival data in the context of protein-protein interaction networks. Partnered with RCy3, the analytical results from BioNet can be visualized in Cytoscape with vastly more options for customization. Starting with the “Quick Start” tutorial from BioNet, we pass the results directly to Cytoscape for visualization:

```r
library(BioNet) # install from Bioconductor
library(DLBCL) # install from Bioconductor
data(dataLym)
data(interactome)
```

**# The following steps are from BioNet's Quick Start tutorial:**

```r
pvals <- cbind(t = dataLym$t.pval, s = dataLym$s.pval)
rownames(pvals) <- dataLym$label
pval <- aggrPvals(pvals, order = 2, plot = FALSE)
subnet <- subNetwork(dataLym$label, interactome)
subnet <- rmSelfLoops(subnet)
fb <- fitBumModel(pval, plot = FALSE)
scores <- scoreNodes(subnet, fb, fdr = 0.001)
module <- runFastHeinz(subnet, scores)
logFC <- dataLym$diff
names(logFC) <- dataLym$label
plotModule(module, scores = scores, diff.expr = logFC)
```

# Using RCy3 we can generate a custom visualization of the same output

```r
# Create network from graphNEL object and load data calculated above
createNetworkFromGraph(module, "module", "BioNet")
loadTableData(as.data.frame(scores))
loadTableData(as.data.frame(logFC))
```

**## Set styles**
```
setNodeLabelMapping("geneSymbol")
setNodeFontSizeDefault(18)
logFC.max.abs <- max(abs(min(logFC)), abs(max(logFC)))
setNodeColorMapping("logFC", c(-logFC.max.abs, 0, logFC.max.abs),
c("#5577FF","FFFFF8","FF7755"), default.color = "#999999")
createColumnFilter("Positive scores", "scores", c(0,max(scores)),"BETWEEN")
setNodeShapeBypass(getSelectedNodes(), "ELLIPSE")
```

Performing advanced graph analytics using RBGL. As an interface to the BOOST library, the RBGL Bioconductor package offers an impressive array of analytical functions for graphs. Here we will start with a network in Cytoscape, load it into R as a graph object, perform shortest path calculation using RBGL and then visualize the results back in Cytoscape (Figure 3).

```r
library(RBGL) # install from Bioconductor
# Convert a sample Cytoscape network to a graph object
g <- createGraphFromNetwork()
```

**# Identify start and finish nodes (styling is optional)**
```
start <- "YNL216W"
finish <- "YER040W"
setNodeBorderWidthBypass(c(start,finish), 20)
setNodeBorderColorBypass(start,"#00CC33")
setNodeBorderColorBypass(finish,"#CC00CC")
```

# Use RBGL to perform shortest path calculation
```
shortest <- sp.between(g, start, finish)
shortest$'YNL216W:YER040W'$length
```

# Visualize results in Cytoscape
```
selectNodes(shortest$path, "name")
setNodeBorderWidthBypass(shortest.path, 20)
createSubnetwork()
```
Discussion
Every operation exposed by Cytoscape’s REST API has now been implemented as a function in RCy3 2.0. Furthermore, RCy3 provides dozens of higher-level helper functions in support of common usage, such as `setNodeColorMapping`, to make script writing more intuitive and efficient. The issue trackers for CyREST and RCy3 are linked for functions pending implementation, such as `mergeNetworks`, as well as for bug fixes. Thus, RCy3 is expected to keep pace with future development of Cytoscape Automation. More broadly, RCy3 is an integral part of the Cytoscape ecosystem, which includes the network repository NDEx, Cytoscape apps and services, and web components like cytoscape.js. The ecosystem has also defined interfaces and standard formats to interoperate with interaction databases and annotation services. Adopting RCy3 for network analysis will establish a connection to the Cytoscape ecosystem and enable Cytoscape Automation workflows. As the sharing and publishing of analysis scripts and workflows become more routine (if not mandated), software tools designed to work in an open and accessible ecosystem are becoming essential.

Data availability
Underlying data
All data underlying the results are available as part of the article and no additional source data are required.

Extended data
- RCy3 Rmd notebooks: https://cytoscape.org/cytoscape-automation/for-scripters/R/notebooks/
- RCy3 workshop presentations: http://tutorials.cytoscape.org/#automation
- Video demonstrations of RCy3: https://www.youtube.com/playlist
- Cytoscape Automation training: http://automation.cytoscape.org/
- Cytoscape Automation code repository: https://github.com/cytoscape/cytoscape-automation

Software availability

Source code available from: https://github.com/cytoscape/RCy3

Figure 3. Visualizing shortest path results. In Cytoscape, the start and finish nodes were colored green and magenta, respectively (left panel). The shortest path calculated by RBGL was then selected and nodes along the path were highlighted with thick borders (middle panel). Finally, a subnetwork was created from the selected path (right panel). Note that while common gene names are displayed as node labels, official yeast identifiers are the actual node names that are referenced in the script.
Archived source code at time of publication: https://doi.org/10.5281/zenodo.3473421

Issue tracker: https://github.com/cytoscape/RCy3/issues

License: MIT License.

Author contributions
AP, SP and RI redesigned and implemented version 2 of RCy3. AP, JAG, SP and BD drafted the manuscript. AP, JAG, SP and RI contributed use cases.

Acknowledgments
We would like to acknowledge contributions by other developers on the original implementation of RCy3 by Paul Shannon, Tanja Muetze and Georgi Kolishkovski. We also greatly appreciate the input from Mark Grimes (testing), Martin Morgan (Bioconductor), and the excellent work on CyREST and Cytoscape Commands by David Otasek and John “Scooter” Morris.

References
Open Peer Review

Current Peer Review Status: ✓

Version 1

Reviewer Report 31 October 2019

https://doi.org/10.5256/f1000research.22981.r55403

© 2019 Denvir J. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

James Denvir
Department of Biomedical Sciences, Joan C. Edwards School of Medicine, Marshall University, Huntington, WV, USA

This article presents an R package, RCy3, designed to control and automate the network visualization software application Cytoscape. The article describes both a low-level ("generic") and a high-level ("specific") API and presents use-cases with some code samples.

Overall this appears to be an API with great potential utility for the bioinformatics community, and the article presents the package clearly and with several useful and informative examples.

I have a couple of minor comments and suggestions, which I hope will serve to improve the article.

The abstract mentions the potential for RCy3 to deliver reproducible workflows. In my opinion, reproducibility is among the most important benefits of code/script-based workflows over workflows performed using "point and click" GUI-based applications. However, the article does not elaborate on this outside of the abstract. A couple of sentences in the discussion describing the potential for RCy3 to enhance reproducibility would be worthwhile.

The article mentions that the Cytoscape App store currently lists over 40 "RCy3-supported apps". A quick note on what is needed for a third party app to be "RCy3-supported" might be pertinent here. This may be already sufficiently covered in existing publications (e.g. reference 6), in which case a simple note to that effect would suffice.

The code block for the section "Building maps of enrichment analysis results" is not quite as clear as the other code blocks in the article. A comment indicating that commandsAPI() opens a web browser page (externally; at least in my environment) and comments after the print() statements showing the expected output, as in the other blocks of code, would be helpful. The uses of file.path and paste(... sep="/") are a little confusing here (to be honest, I was surprised to find these actually worked on Windows systems); brief comments added to the code might aid readability here.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

**Competing Interests:** No competing interests were disclosed.

**Reviewer Expertise:** Bioinformatics.

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

---

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com