Vol. 21 no. 1 2005, pages 135–136
doi:10.1093/bioinformatics/bth458

Network structures and algorithms in Bioconductor

Vincent J. Carey[1,*], Jeff Gentry[2], Elizabeth Whalen[2] and Robert Gentleman[2]

[1]Channing Laboratory, Brigham and Women's Hospital, 75 Francis Street and [2]Division of Biostatistics, Dana Farber Cancer Institute, 44 Binney Street, Boston, MA 02115, USA

Received on May 28, 2004; revised on July 12, 2004; accepted on July 27, 2004
Advance Access publication August 5, 2004

ABSTRACT
Summary: In this paper, we review the central concepts and implementations of tools for working with network structures in Bioconductor. Interfaces to open source resources for visualization (AT&T Graphviz) and network algorithms (Boost) have been developed to support analysis of graphical structures in genomics and computational biology.
Availability: Packages *graph*, *Rgraphviz*, *RBGL* of Bioconductor (www.bioconductor.org). Open source.
Contact: stvjc@channing.harvard.edu

INTRODUCTION

Network structures are ubiquitous in genomics and computational biology. The recent monograph of Collado-Vides and Hofestädt (2002) is devoted to genomic network modeling. A number of software initiatives in bioinformatics has targeted different aspects of network visualization and analysis. For recent examples and references, refer to Han *et al*. (2004) and the Reactome project (www.reactome.org). In this paper, we describe methods that are developed in the Bioconductor project (www.bioconductor.org) to provide network data structures and algorithms for use in R. Key principles of design and implementation are object-oriented programming, leveraging existing open source software and support for standardized representation and serialization. The integration of these components within R provides users with the software infrastructure needed to address bioinformatic problems related to graphs and networks.

DESCRIPTION

graph package. The *graph* package is a collection of structures and methods for working with graphs (sets of nodes and edges) in R. The base virtual class `graph`, has a single slot, `edgemode`, which may take value '`directed`' or '`undirected`'. Classes that extend `graph` include `graphNEL` (graph represented by Node and Edge-List), `distGraph` [graph represented by an R distance (`dist`) structure] and `clusterGraph` (a representation of clustered data).

Generic methods have been defined for the base class, including `acc` (return vector of names of accessible nodes relative to a given node), `complement`, `connComp` (return list of node names defining connected components), `degree`, `dfs` (depth first search), `intersection` and `union`. When appropriate, these methods return `graph` instances; otherwise, the generics define the 'contract' that must be followed to secure consistent behavior for instances of subclasses. As application requirements emerge additional generics for the base class will be introduced.

Rgraphviz package. Graph visualization depends upon *layout*, fixing locations of nodes, edges and labels on a plotting surface. Various classes of layout algorithms have been developed, and the AT&T Graphviz system is an open source implementation suited to rendering large graphical structures encountered in telecommunications and complex software engineering tasks (North *et al*., 1998, http://www.graphviz.org). The Graphviz system defines an abstract data type `agraph` and a large collection of C utilities for populating and visualizing graph structures and associated textual information. The `agraph` structure is reflected in an R object, and layout and annotation information are used by the `plot` method for instances of class `graph`. Default renderings are fairly basic; users with significant enhancement needs (e.g. font changes, edge and node coloring) can save the layout information and use standard R `text` and `line` methods to create a customized visualization of a graphical structure.

RBGL package. The Boost project is an open source implementation of C++ libraries focusing on STL methodologies with an emphasis on portability. The Boost Graph Library (Siek *et al*., 2001) is a C++ library of network data structures and algorithms. The *RBGL* package defines interfaces from R to BGL routines. At present, procedures have been interfaced for minimum spanning tree construction,

*To whom correspondence should be addressed.

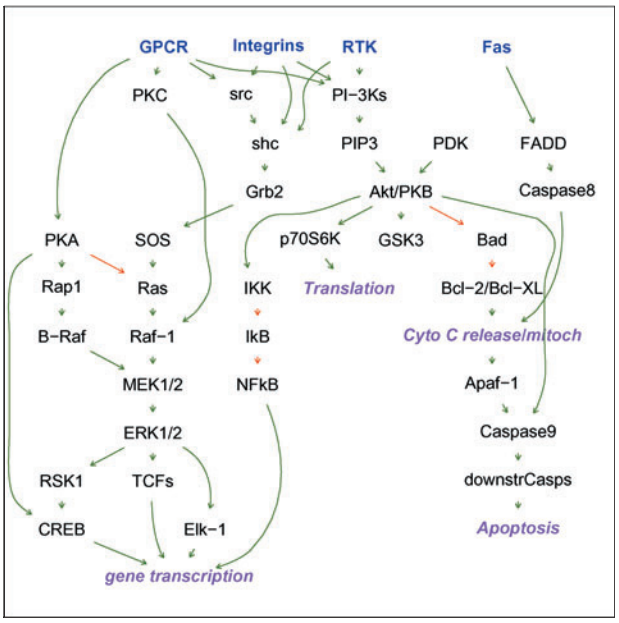

Fig. 1. A rendering of the signal transduction pathway using *Rgraphviz*. Red arcs indicate antagonistic relationships, green arcs indicate supportive relationships, blue text indicates membrane-associated components and purple text indicates biological processes.

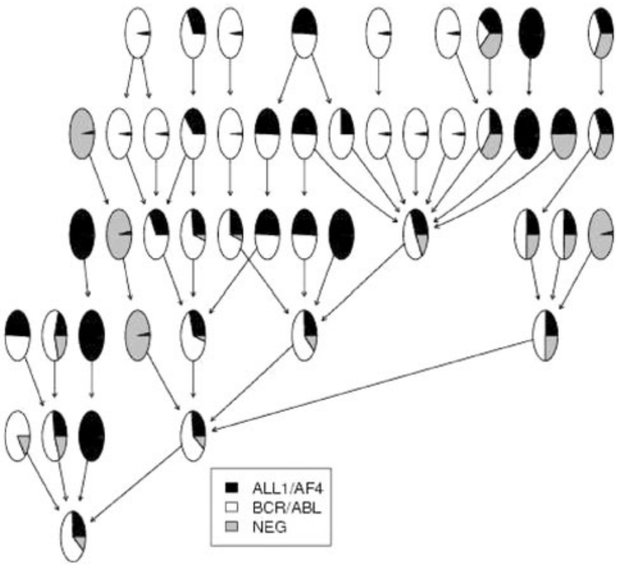

Fig. 2. GO node composition display. Nodes of the graph are elements of the GO cellular component subontology, and arcs are directed from specific to more general terms. Nodes are positioned in the plane using Graphviz 'dot' layout, and are occupied by pie charts indicating the relative frequencies of ALL/AF4, BCR and NEG samples in the set of differentially expressed genes annotated to the GO terms.

shortest path discovery, depth-first search, topological sorting, edge connectivity measurement and connected component decomposition.

EXAMPLES

Figure 1 illustrates the use of Graphviz layout and *Rgraphviz* markup capabilities to deliver an interactively navigable representation of the signal transduction pathway. The graph can be interrogated using R's standard `identify` function, for a variety of possible purposes. For example, a user may wish to subset a collection of microarray probes to retain only those annotated to some elements of this pathway.

Figure 2 illustrates the advantage of combining biological annotation resources, statistical graphics and network layout methods in R. Pie chart and legend computation is routine in R, the graphical structure of the Gene Ontology (Ashburner *et al*., 2000) is obtained from the Bioconductor *GO* package, and layout was obtained from *Rgraphviz*. The figure clearly indicates non-homogeneity in the distribution of phenotypes across cellular component terms, and the graph can be interactively interrogated using R's standard `identify` function to obtain the terms associated with nodes of interest.

DISCUSSION

Our novel integration of graph data types, graph layout and graph algorithm resources with a full-featured data analysis and visualization environment has proven highly productive in genomic and proteomic applications. New Bioconductor packages such as *GOstats*, *GraphAT* and *apComplex* make essential use of this infrastructure. As standard representation methods and curated data resources for genomic networks development, we will enrich the methods and resources for working with these in Bioconductor.

ACKNOWLEDGEMENTS

Support for the development of this software came from the High Tech Industry Multidisciplinary Research Fund at the Dana-Farber Cancer Institute and NIH grant no. 1R33 HG002708: 'A Statistical Computing Framework for Genomic Data'.

REFERENCES

Ashburner,M., Ball,C.A., Blake,J.A., Botstein,D., Butler,H., Cherry,J.M., Davis,A.P., Dolinski,K., Dwight,S.S., Eppig,J.T. *et al*. (2000) Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat. Genet.*, **25**, 25–29.
Collado-Vides,J. and Hofestädt,R. (Eds) (2002) *Gene Regulation and Metabolism: Post-Genomic Computational Approaches*. MIT Press, Cambridge, MA.
Han,K., Ju,B.-H. and Jung,H. (2004) WebInterViewer: visualizing and analyzing molecular interaction networks. *Nucleic Acids Res.*, **32**, W89–W95.
North,S. Gansner,E. and Ellson,J. (1998) http://www.graphviz.org
Siek,J.G., Lee,L.-Q. and Lumsdaine,A. (2001) *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, Reading, MA.