A meta-viewer for biomolecular data

Olaf Delgado-Friedrichs Tobias Dezulian Daniel H. Huson

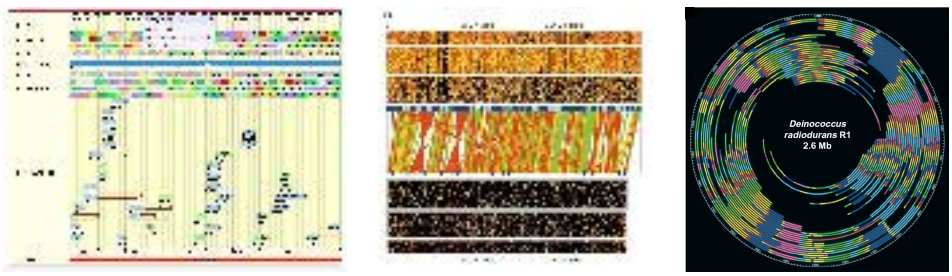
Center for Bioinformatics (ZBIT), University of Tübingen, Sand 14, D-72076 Tübingen

Abstract: The development of powerful visualization tools is a major challenge in bioinformatics. Although many good special purpose viewers exist, there is a need for configurable meta-viewers that provide enough flexibility to support many different types of data and visualizations. Here we present CGViz, a new software tool that fulfills many of the requirements placed on such a configurable meta-viewer.

1 Introduction

A major challenge in bioinformatics is the development of powerful tools that help biologists to navigate, explore and analyze huge amounts of genomic and other forms of biomolecular data.

Visualization plays a central role here [T$^+$01]. A number of well designed visualization programs exist for displaying particular types of data. For example, many biological databases have graphical front-ends such as the Ensembl Genome Browser [H$^+$02], other programs [V$^+$01] are designed to visualize comparisons of homologous data and a third kind of application is aimed at the presentation of genomic data annotations, such as e.g. GenVision [L$^+$99], see Figure 1. These and similar applications have become indispensable tools in molecular biology.



(a) (b) (c)

Figure 1: Visualizations of (a) a genomic database [H$^+$02], (b) comparative data [V$^+$01] and (c) genomic data annotations [L$^+$99].

Each such application is a special-purpose program that was designed to address a specifi c

visualization problem. Such dedicated programs usually do a good job at solving their specific problem. However, there are two major drawbacks to most (if not all) current visualization tools for biomolecular data: first, it is often difficult to persuade the software to include additional or new types of data and second, it is usually impossible to extend the visualization to new types of views of the given data. Additionally, in practice the development of many good special-purpose visualization tools becomes "frozen" as the maintenance of a tool with a very narrow range of applications is not an attractive task.

Hence, we believe that there is a need for a bioinformatics *meta-viewer* whose main characteristic is flexibility, allowing for the interactive configuration of many different types of visualizations of biomolecular data. In one use-case, we envision the system as being employed as a meta-tool for crafting dedicated viewers for recurring visualization tasks. In a second use-case, a researcher interactively creates and modifies an appropriate visualization as he or she brings together different types of data as it becomes available.

We have developed such a meta-viewer called *CGViz* that provides fully-configurable visualization of biomolecular data. Configurability of this program is based on the concept of a "visualization DAG" that determines which data is drawn where, and how. Extensibility of the program's visualization capabilities is ensured by a plug-in design. The render engine employed by the program uses a binary space partition [dB$^+$00] to enhance performance on large datasets. For availability, please see:
`www-ab.informatik.uni-tuebingen.de/software/cgviz`.

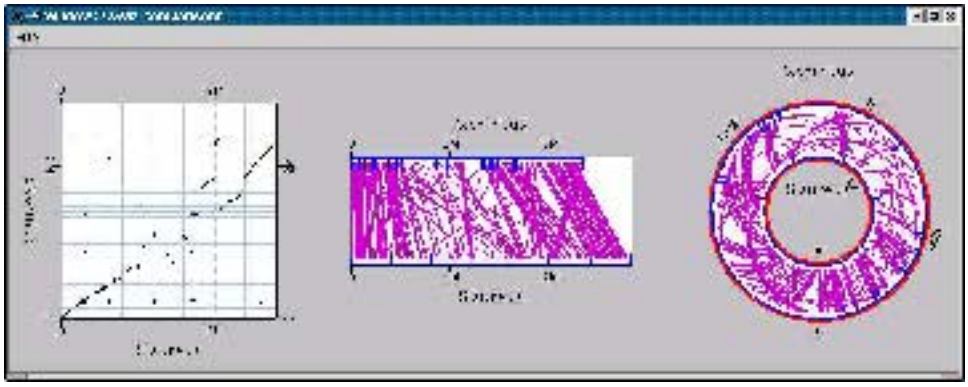
2 Elements of a visualization

In this section we will determine some of the requirements that a meta-viewer for biomolecular data should fulfill by studying a simple example. In Figure 2 we show three different views of a comparison of two closely related bacterial genomes. The input data to each of these views consists of three sets of objects: The first genome, *S.aureus*, is a finished genome. For computational reasons, it is given as seven *contigs* (pieces of contiguous sequence), each $400Kb$ long. The second (unfinished) genome, *S.cornsus*, is given as 19 contigs of different lengths. Thirdly, we are given a set of matches between both genomes, computed using BLAST [A$^+$90].

In Figure 2, we display these three sets of data in three different ways, namely as a dot-plot, linear match-plot and circular match-plot, respectively. In the first case, we use horizontal and vertical "rulers" to indicate the boundaries of the contigs, whereas in the latter two cases the contigs are represented by "bars". In the first case, matches are represented by "dots", whereas in the latter two cases they are represented by lines or curves, respectively, connecting homologous positions in the two genomes.

In this example we can distinguish between the following components of the visualization:

- *data collections*, such as the set of all contigs of one genome, or the set of matches,
- *glyphs*, that is the graphical objects used to represent a given collection of data, such as "dots", "bars" or "lines" etc.,



(a) (b) (c)

Figure 2: A comparison of two bacterial sequences that are both given by a number of pieces, displayed (a) as a dot-plot, (b) as a set of matches between linear sequences and (c) as a set of matches between circular sequences

- *panes*, these are regions in the program's windows where data is displayed, and
- *windows*, the actual windows that contain the given visualization panes.

A meta-viewer should allow arbitrary combinations of these different components.

3 The CGViz meta-viewer

We have developed a meta-viewer for biomolecular data called *CGViz*, which stands for *comparative genomics visualization tool*, as a main area of application of the tool will be in comparative genomics. The program is built around a *visualization DAG*, that is an acyclic directed graph whose nodes represent the different components mentioned in the previous section, which we will refer to as *data-*, *glyph-*, *pane-* and *window nodes*, respectively.

In the above example depicted in Figure 2, we require three *data nodes*, one for each set of contigs for either genome and a third to represent the set of BLAST matches. The information stored in a data node consists of individual records, each containing a list of coordinates, an optional label and an optional list of supplemental information specified by name-value pairs. The latter may be used by transformers to filter for relevant data or by drawer plug-ins to modify the way a record is displayed. A record may represent an individual BLAST match, a contig, an annotation, a sampled CG content value, and so on. The simplicity of the structure and the accompanying file format make it easy to convert from data produced by standard bioinformatics tools.

A *glyph node* describes how a given set of data is to be displayed on the screen, i.e. as dots or curves etc. A *pane node* describes the properties of a drawing area within a given

(a) (b) (c)

Figure 3: (a) This graph describes the visualization shown in the previous figure, together with configurators for the properties of (b) the glyphs that are used to represent the given data and (c) the panes in which they are displayed.

window. In Figure 2 we see three types of panes, namely (a) a rectangular pane in which the two orthogonal axes represent the two different genomes, (b) a "horizontal" pane in which the bottom and top edges represent the two genomes and (c) a circular pane which, in this case, is the most appropriate type of visualization of the two genomes, as they are circular. Finally, a *window* node represents the window in which panes are to appear.

Logical connections between the different nodes are represented by edges in the graph. The graph depicted in Figure 3(a) represents the configuration of the visualization shown in Figure 2. The nodes are partitioned into vertical layers. From left to right, these are the data-, glyph-, pane- and window nodes. Details of the visualization are determined by setting properties of the different types of nodes as indicated in Figure 3(b,c). Additionally, *transformation nodes* can be inserted between data nodes to process data.

The CGViz visualization graph bears some resemblance to both the data-flow networks used in classical visualization systems such as AVS [U+89] and the scene graphs known from computer graphics system such as Open Inventor [Wer94]. The combination of Data, Glyph, Pane and Window nodes establishes a kind of specialized scene graph, while the addition of transformer nodes provides a data-flow oriented component.

4 Additional features of the CGViz program

As indicated briefly in the previous section, CGViz provides the functionality of a meta-viewer through the visualization graph. Based on this, a visualization can initially be designed by interactively constructing the graph and then saving it to a file. Later runs of the program then first read the visualization graph to configure the visualization and then read the given user data to display it.

For maximum flexibility, a number of aspects of the program are designed as plug-ins, including the "transformation operations" between data nodes, the "drawers" that draw the glyphs to represent the data and the "panes" in which the glyphs appear.

Biological data sets can be very large. To address this, CGViz employs a binary space partition [dB+00] to reduce the over-painting of pixels when zoomed-out from the data

and to avoid drawing in regions outside of the visible area when zoomed-in. Each partition is along an axis-parallel plane. Individual data records are stored in the leaves of the tree. Every interior node stores a bounding box and a summary of the data in its attached subtree. A subtree whose bounding box does not intersect the currently displayed region or is below a certain size need not be traversed. This simple scheme has already proved extremely helpful, but further improvements are under way.

It goes without saying that the program supports the usual navigational features. The program is written in Java and its file format is based on XML.

CGViz is still under active development. Plans for the near future include:

- Designing standalone viewers for common data such as GenBank files, BLAST output, and so on.
- Extending provisions for user interaction.
- Adding support for multiple genome comparison and other complex tasks.

5 Example

We now show an example of a view constructing using *CGViz* in Figure 4. Here, we compare the genomes of two related species, *D.melanogaster* and *D.pseudoobscura*. Each genome is given as a collection of contigs layed out along the x and y-axes of each dot-plot, respectively. The dots represent sequence matches computed by BLAST. The aim is to show the effect of an algorithm that attempts to find the best mutual order and orientation of the contigs in both assemblies. In the top-left dot-plot the contigs are unordered. The bottom-left, top-right and bottom-right dot-plot shows the effect of optimally ordering the contigs in *D.melanogaster*, in *D.pseudoobscura*, and in both genomes, respectively.

Acknowledgments

We would like to thank Aaron Halpern (TCAG), Michael Kaufmann, Christian Klug, Nils Krugmann, Daniel Richter, Michael Schröder and Georg Zeller for contributing ideas and code to CGViz.

References

[A+90] S. F. Altschul et al. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[dB+00] M. de Berg et al. *Computational Geometry — Algorithms and Applications*. Springer, 2nd edition, 2000.

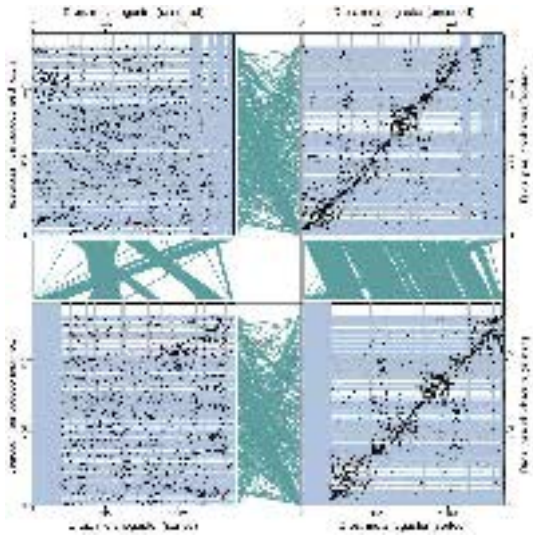


Figure 4: The four dot-plots show comparisons of two assemblies of *D.melanogaster* and *D.pseudoobscura*, differing only by the ordering of the contigs in either or both assemblies. The thin pane between any two dot-plots indicates how the corresponding contigs were permuted.

[H+02] T. Hubbard et al. The Ensembl genome database project. *Nucleic Acids Res*, 30(1):38–41, 2002.

[L+99] Jieyi Lin et al. Whole-Genome Shotgun Optical Mapping of Deinococcus radiodurans. *Science*, 285:1558–1562, 1999.

[T+01] R.J. Turner et al. Visualization Challenges for a New Cyberpharmaceutical Computing Paradigm. *2001 Symposium on Parallel and Large-Data Visualization and Graphics*, pages 7–18, 2001. Invited contribution.

[U+89] C. Upson et al. The Application Visualization System: A Computational Environment for Scientific Visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, 1989.

[V+01] J. C. Venter et al. The sequence of the human genome. *Science*, 291:1145–1434, 2001.

[Wer94] J. Wernecke. *The Inventor Mentor*. Addison Wesley, 1994.