BIOINFORMATICS APPLICATIONS NOTE Vol. 29 no. 3 2013, pages 381–383
doi:10.1093/bioinformatics/bts677

Genome analysis

Advance Access publication November 19, 2012

Scribl: an HTML5 Canvas-based graphics library for visualizing genomic data over the web

Chase A. Miller, Jon Anthony, Michelle M. Meyer and Gabor Marth*

Department of Biology, Boston College, Chestnut Hill, MA 02467, USA

Associate Editor: Martin Bishop

ABSTRACT

Motivation: High-throughput biological research requires simultaneous visualization as well as analysis of genomic data, e.g. read alignments, variant calls and genomic annotations. Traditionally, such integrative analysis required desktop applications operating on locally stored data. Many current terabyte-size datasets generated by large public consortia projects, however, are already only feasibly stored at specialist genome analysis centers. As even small laboratories can afford very large datasets, local storage and analysis are becoming increasingly limiting, and it is likely that most such datasets will soon be stored remotely, e.g. in the cloud. These developments will require web-based tools that enable users to access, analyze and view vast remotely stored data with a level of sophistication and interactivity that approximates desktop applications. As rapidly dropping cost enables researchers to collect data intended to answer questions in very specialized contexts, developers must also provide software libraries that empower users to implement customized data analyses and data views for their particular application. Such specialized, yet lightweight, applications would empower scientists to better answer specific biological questions than possible with general-purpose genome browsers currently available.

Results: Using recent advances in core web technologies (HTML5), we developed Scribl, a flexible genomic visualization library specifically targeting coordinate-based data such as genomic features, DNA sequence and genetic variants. Scribl simplifies the development of sophisticated web-based graphical tools that approach the dynamism and interactivity of desktop applications.

Availability and implementation: Software is freely available online at http://chmille4.github.com/Scribl/ and is implemented in JavaScript with all modern browsers supported.

Contact: gabor.marth@bc.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on August 8, 2012; revised on November 13, 2012; accepted on November 14, 2012

1 INTRODUCTION

Increasingly, multi-faceted and complex high-throughput biological data require sophisticated analysis and visualization of genomic datasets that are typically very large (often in the many terabytes). Traditionally, because of these multi-fold requirements, sophisticated genomic analyses required desktop applications operating on locally stored datasets (Barrett, 2009; Gordon *et al.*, 1998; Huang and Marth, 2008; Purcell *et al.*, 2007). As genomic datasets have grown in size, it is less and less realistic to import large genomic datasets onto local storage, and especially smaller users without large IT infrastructure must rely on datasets stored remotely at specialized data analysis and archival centers, e.g. NCBI (http://www.ncbi.nlm.nih.gov), and large international consortium projects, e.g. the 1000 Genomes Project (http://www.1000Genomes.org). Although some desktop solutions such as the Integrative Genomics Viewer (Robinson *et al.*, 2011) offer access to remote data, new development increasingly centers on web applications to access such data.

Prime examples of web tools accessing remote data are genome browser applications (Down *et al.*, 2011; Kent *et al.*, 2002; Skinner *et al.*, 2009; Stein *et al.*, 2002). Owing to traditional limitations of web technologies, these applications use server-side image generation, requiring the server to render and transmit each new image. Interactivity is thus severely limited by the speed of the network, and the instant feedback seen in desktop applications is never achieved. Although genome browsers provide users with excellent facilities to add their own datasets, they offer very little flexibility to incorporate custom analyses or views. Simply put, the needs of many specialized applications are not met with currently available 'stock' genome browser programs.

The recent release of updated core web technologies, including JavaScript, CSS and HTML, popularly referred to as HTML5, enables development of more sophisticated web applications. These new technologies support client-side image generation, where only data, rather than the image file, are transferred to the client. Indeed, several applications have already taken advantage of client-side graphics (Down *et al.*, 2011; Gel Moreno and Messeguer Peypoch, 2011; Goecks *et al.*, 2011; Sinha and Armstrong, 2012; Tremmel, 2011; Zhou *et al.*, 2011). However, developing such applications from scratch is still a significant undertaking. Software libraries, e.g. BioPerl, BioJava and Biopython (Cock *et al.*, 2009; Holland *et al.*, 2008; Stajich *et al.*, 2002), can simplify this development by reducing duplicative efforts and increasing accessibility to non-expert programmers. More relevantly, the iCanPlot library provides HTML5 image generation for visualizing statistical data.

Our goal here was to develop a library specifically designed for customizable client-side visualization of coordinate-based genomic data in web applications. We are building on top of the HTML5 Canvas element, which has been used for other client-side graphic libraries (Bostock and Heer, 2009; Bostock *et al.*, 2011; Sinha and Armstrong, 2012). Here we report Scribl, an HTML5 Canvas genomics graphic library, to enable bioinformaticians to write dynamic, interactive and powerful client-side

*To whom correspondence should be addressed.

graphical web visualizations, and to merge the power of desktop applications with the inherently collaborative nature and accessibility of the web.

2 RESULTS

2.1 Design considerations and features

The Scribl library was designed for effective visualization of coordinate-based genomic datasets, as detailed in the following sections:

2.1.1 Dynamic and interactive user experience Effective visualization of large and complex data requires dynamic interaction between user and viewer program, but to achieve this, the application must provide immediate response to user control. HTML5 provides functionality to render even very complex graphical objects entirely client-side, i.e. all images are drawn on the local computer. This means that for local data, there is no network latency, and rendering is instantaneous. For remote/centrally stored data, the speed of access is unavoidably limited by the bandwidth of remote data servers. With Scribl, transfer volume is minimized because new views can be rendered with data already in hand, rather than requesting a new image from the server (detailed comparison in Supplementary Text S1). In other words, by using client-side rendering, applications built with Scribl can achieve a level of performance and interactivity that was previously only possible with desktop applications.

2.1.2 Universal accessibility and interactivity Cooperative work from collaborating research groups is becoming the norm in genomic data analysis. Such collaborations typically require access to the same large datasets by multiple groups, often situated in distant locations. Because Scribl uses the Web as its platform, it inherits the universal accessibility of the Internet. Access to remote data is straightforward; collaboration is easier via a common workplace and shareable links to specific data views; user data, results and preferences can be stored in the cloud and running the programs only requires web browsers available to all users.

2.1.3 Genomic orientation A genomic visualization library effectively supports developers only if it provides objects that correspond well to actual genomic data types, and functionality that corresponds well to common data manipulation and visualization modalities. In addition to elementary coordinate-based feature objects, Scribl provides customary composite objects such as gene models. It also provides support for 'sequence' and 'variant' data types, which are of high importance in present-day genomic analyses. Visualization of alignments and variants is possible because the underlying indexed data formats, designed to assist the handling of large genomic data files, permit access to smaller 'slices' of the data representing specific genomic regions. Scribl exploits these formats and offers depth of read coverage, multiple alignment and nucleotide sequence views for read alignments in BAM files, and position as well as population allele frequency views for variant calls in VCF files.

2.1.4 Well-designed API, and intuitive library functions A well-designed library enables developers to build complex and sophisticated applications, but also to rapidly set up simple prototypes. Accordingly, all of Scribl's data access and visualization features have sensible defaults and are exposed to the programmer and easy to incorporate into new web applications. Additional refinements and fine tuning are made possible with a large number of optional customization parameters available for the developer. Importantly, Scribl is implemented entirely in the JavaScript language as an open-source library (http://chmille4.github.com/Scribl/source.html), and therefore readily integrated into web applications, without further software dependencies or the need for additional development tools.

Creating a web application using Scribl visualization only requires making an HTML document linked to the Scribl library. We provide an interactive example of code and the graphic at: http://jsbin.com/afinec/edit#html,live. More complex applications, including viewing data from remote sources as well as click and hover interactions, can be built from this simple code base.

2.1.4 Flexibility and modularity A client application may wish to access multiple different data sources. Also, different applications may wish to provide substantially different views of the same data, as dictated by the specific analysis needs. For these reasons, we have separated the graphics library (Scribl) code form the client code that parses and processes data (e.g. from local files, databases or remote data servers). This choice was made to focus on the effective visualization of the data even in the presence of diverse and ever-changing file formats and data protocols.

2.2 Example applications

Scribl is not itself a web application or genome browser, but a library that provides the functionality to build such web tools (the relationships between Scribl, web applications and data servers are shown in Supplementary Fig. S1). In the Supplementary Material, we present two demo applications to illustrate ways in which Scribl may be used to produce specific web applications for essential bioinformatics problems. Rover (http://chmille4.github.com/Rover; Fig. 1) is a simple but flexible genome browser that can visualize coordinate-based annotation data, read alignments (BAM files) and variant calls (VCF files) via smooth panning and zooming. Futhermore, Rover has semantic zooming that allows information to be visualized intuitively at different resolutions, e.g. at a 'birds-eye view', only analog read coverage is shown; at higher zoom levels, individual aligned reads are displayed as lines and at a resolution >4 pixels per nucleotide, the base sequence of the aligned reads are displayed. For more details on Rover, see Supplementary Text S1. Additionally, we created GenomeChart (http://roz.bc.edu:8090/index.html), a genome context viewer described in Supplementary Text S3.

3 IMPLEMENTATION

Scribl is written in JavaScript and generates graphics with the HTML5 Canvas element (http://dev.w3.org/html5/2dcontext/). Of the other alternatives, SVG is an older technology that treats each glyph as an individual object, adding substantial object management overhead as the number of glyphs increases. Flash is a proprietary browser plug-in. WebGL requires specialized knowledge of GPU programming, crossing purposes with our intention of Scribl as an extendable open-source library to which other developers will contribute. Canvas is fast, supported

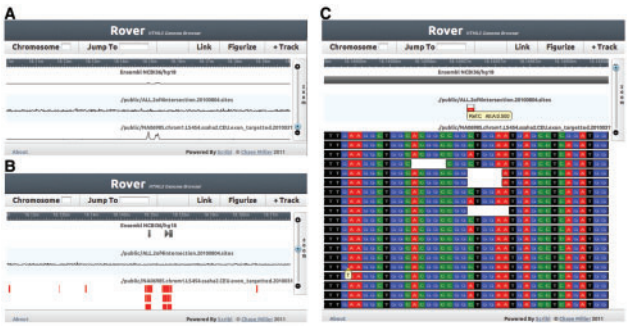

Fig. 1. The Rover genome browser built using the Scribl library. The three screenshots demonstrate the data views corresponding to the various semantic zoom levels. (**A**) High-level 'birds-eye view' of a large region of human chromosome 1, where sequencing read coverage from 1000 Genomes Project exome sequencing data is indicated by wiggle plots. (**B**) Data view at medium zoom level, showing annotation features and individual aligned reads as coordinate-type features. (**C**) Detailed data view at high resolution: the nucleotide sequence of each individual aligned read is shown. Additional information is revealed when hovered over. Hovering over a single-nucleotide polymorphism displays allele frequency and hovering over an aligned read displays insertions relative to the reference

in mainstream browsers and allows application development in JavaScript, the dominant web programming language today. Also, a Scribl program can be encapsulated in a single HTML file, enabling offline uses such as report generation.

The library supports five basic glyphs: rectangle, block-arrow, arrow, sequence and line. Basic glyphs can be used alone or combined to form composite glyphs (e.g. combining rectangles and block-arrows to create a gene model complete with exons). Users can easily add their own custom glyphs that inherit all base functionality. The library implements a hierarchical inheritance system for glyph management: attributes (e.g. color) can be set at the global, type or glyph level. Canvas itself does not natively track objects or states, and therefore provides no method for attaching events such as mouse-clicks and mouse-overs to glyphs. For this reason, we implemented sophisticated event-handling capabilities as part of the Scribl library to facilitate intuitive graphical user interaction (see Supplementary Fig. S2).

Using the basic glyphs, we implemented views for essential genomic data types: (i) coordinate box, (ii) wiggle chart indicating feature density, (iii) nucleotide sequence, (iv) read alignment view and (v) variant allele frequency. The same data may be viewed in multiple alternate views, which can be controlled via 'draw hooks' in the Scribl API. This allows for, e.g. semantic zooming, i.e. where the display mode is chosen according to the zoom level. Each track within a page can be controlled separately. In addition to graphic creation, the API also supports interaction (e.g. by providing function calls to erase tracks, lanes or individual glyphs; to return the exact pixel location of a track, lane or glyph). Such precise control may not be needed for simpler applications but is necessary for more complex tools requiring communication with the graphics layer (see Supplementary Fig. S3). Scribl is well documented (https://github.com/chmille4/Scribl/wiki/v1.1), with examples to guide developers (http://chmille4.github.com/Scribl/index.html#examples).

Scribl performs best in Chrome (19.0+) largely owing to the speed of JavaScript execution, but also works in other browsers that support HTML5 Canvas, including, at present, Firefox (13.0+) and Safari (5.0+). Scribl itself also works in Internet Explorer (9.0+), but the Rover application does not owing to data access constraints.

4 CONCLUSIONS

Scribl is a powerful library harnessing client-side web graphics, supporting the creation of rich and customized genomic graphical web applications. We expect a growing need for custom analyses and visualization of large centralized datasets via the web, and our library provides a foundation to build such web applications. Moving forward, we will expand support for additional genomic data types and file formats, provide new and informative views and implement interactive data-editing capabilities. Already, Scribl enjoys wide support, with multiple developers contributing to the open-source codebase, and powers diverse applications worldwide.

Funding: This research was supported by the National Institutes of Health grant R01HG004719 to G.T.M., and a PhRMA Foundation Research Starter Grant for Informatics to M.M.

Conflict of Interest: none declared.

REFERENCES

Barrett,J.C. (2009) Haploview: visualization and analysis of SNP genotype data. *Cold Spring Harb. Protoc.*, 2009, pdb ip71.

Bostock,M. and Heer,J. (2009) Protovis: a graphical toolkit for visualization. *IEEE Trans. Vis. Comput. Graph.*, **15**, 1121–1128.

Bostock,M. *et al.* (2011) D(3): data-driven documents. *IEEE Trans. Vis. Comput. Graph.*, **17**, 2301–2309.

Cock,P.J. *et al.* (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422–1423.

Down,T.A. *et al.* (2011) Dalliance: interactive genome viewing on the web. *Bioinformatics*, **27**, 889–890.

Gel Moreno,B. and Messeguer Peypoch,X. (2011) GenExp: an interactive web--based genomic DAS client with client-side data rendering. *PLoS One*, **6**, e21270.

Goecks,J. *et al.* (2011) The Galaxy Track Browser: Transforming the genome browser from visualization tool to analysis tool. In *2011 IEEE Symposium on Biological Data Visualization (BioVis)*. pp. 39–46.

Gordon,D. *et al.* (1998) Consed: a graphical tool for sequence finishing. *Genome Res*, **8**, 195–202.

Holland,R.C. *et al.* (2008) BioJava: an open-source framework for bioinformatics. *Bioinformatics*, **24**, 2096–2097.

Huang,W. and Marth,G. (2008) EagleView: a genome assembly viewer for next-generation sequencing technologies. *Genome Res.*, **18**, 1538–1543.

Kent,W.J. *et al.* (2002) The human genome browser at UCSC. *Genome Res.*, **12**, 996–1006.

Purcell,S. *et al.* (2007) PLINK: a tool set for whole-genome association and population-based linkage analyses. *Am. J. Hum. Genet.*, **81**, 559–575.

Robinson,J.T. *et al.* (2011) Integrative genomics viewer. *Nat. Biotechnol.*, **29**, 24–26.

Sinha,A.U. and Armstrong,S.A. (2012) iCanPlot: visual exploration of high-throughput omics data using interactive Canvas plotting. *PLoS One*, **7**, e31690.

Skinner,M.E. *et al.* (2009) JBrowse: a next-generation genome browser. *Genome Res.*, **19**, 1630–1638.

Stajich,J.E. *et al.* (2002) The Bioperl toolkit: Perl modules for the life sciences. *Genome Res.*, **12**, 1611–1618.

Stein,L.D. *et al.* (2002) The generic genome browser: a building block for a model organism system database. *Genome Res.*, **12**, 1599–1610.

Tremmel,G.N. *et al.* (2011) NGS Explorer: An application for Visually Contextualizing and Interrogating Multivariate Omics Data. In *IEEE Symposium on Biological Data Visualization*. Rhode Island, Providence.

Zhou,X. *et al.* (2011) The Human Epigenome Browser at Washington University. *Nat. Methods*, **8**, 989–990.