NextBrowse: An integrated and interactive web-based genome browser for analyzing and interpreting genomic data

Phillip J. Whisenhunt

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
In
Computer Science and Applications

David A. Mittelman, Chair
Harold R. Garner
Manuel A. Perez-Quinones

April 23rd, 2012
Blacksburg, Virginia

Keywords and phrases: Genome Browser, Genomic Visualization, 3D Interaction, File Streaming

ABSTRACT

With the advent of high throughput sequencing technologies over the past decade there has been a surge in the amount of genomic data that needs to be analyzed and interpreted. Despite the availability of software frameworks such as the Genome Analysis Toolkit, data interpretation and analysis still requires human intervention and refinement. Genome browsers enable developers and users of sequence analysis tools to visualize, compare, and better interpret genomic data such as gene expression and functional annotations. We developed a next generation cross platform web-based genome browser, NextBrowse, for visualizing General Feature Format and Binary Alignment Map files. NextBrowse uses advanced visualization techniques such as 3D feature selection and transparency based on mapping quality, and improved Graphical User Interface elements such as individual track searching and textual and graphical reference location. NextBrowse is the first genome browser to allow BAM files to be streamed and visualized, the first genome browser to employ security measures, and the first to use only client side rendering. NextBrowse takes advantage of the open-source community, allowing developers and users to extend the project to fit their needs. NextBrowse along with all documentation is available for use at http://www.nextbrowse.vbi.vt.edu.

ACKNOWLEDGEMENTS

First, I would first like to thank my primary advisor, Dr. David Mittelman, for his support, funding, and guidance in this work. Dr. Mittelman's passion to help and mentor his students surpasses that of any professor I have ever met and it is through his passion that this work was made possible. Thank you Dr. Mittelman.

I would like to thank my committee members, Dr. Harold Garner and Dr. Manuel Perez-Quinones, for serving on my committee and for their suggestions throughout this work. I would like to thank Robin Varghese, Zach Pavlosky, Gareth Highnam, John McCormick, and the rest of Dr. Mittelman's lab for their willingness to answer my never-ending biological questions and for their feedback on NextBrowse. I would especially like to thank Ashwin Puthige for his help in writing and configuring the BAM file streaming portion of NextBrowse.

I owe my deepest gratitude to my parents for always pushing me to do my best. Thank you for your love, guidance, and support; you guys rock. I would like to thank my love, Pamela Spivey, for supporting me in all of my decisions over the years. Lastly, I would like to thank the Virginia Bioinformatics Institute, NIH/NINDS, and NVIDIA for funding Dr. Mittelman's lab.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AJAX	Asynchronous JavaScript and XML
BAM	Binary Alignment Map
CSS	Cascading Style Sheets
DNA	Deoxyribonucleotide
DOM	Document Object Model
GFF	Generic Feature Format
GIF	Graphics Interchange Format
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
NoSQL	Non-relational database
ORM	Object Relational Mapper
REST	Representational State Transfer
RIA	Rich Internet Application
RNA	Ribonucleic acid
ROR	Ruby on Rails
SAM	Sequence Alignment Map
SNP	Single Nucleotide Polymorphism
SQL	Structured Query Language
SSL	Secure Sockets Layer
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WWW	World Wide Web

Chapter 1

Introduction, Background, and Motivation

This chapter presents an introduction to the topic of genome browsers, background information on the state-of-art of genome browsers, motivation to the problem, and a road map for this thesis.

1.1 Introduction

With the advent of high throughput next generation sequencing technology, there has been an eruption in the amount of genomic data that must be analyzed and interpreted. The end result of sequencing and many biological experiments is a large quantity of digital genomic data, such as gene expression or sequence data that requires analysis and interpretation. Despite the availability of semi-automated software frameworks such as the Genome Analysis Toolkit, data interpretation and analysis still requires human intervention and refinement [1]. Therefore, data presentation tools, specifically genome browsers, have been developed to assist users in the interpretation and analysis of genomics data through the use of visualization.

Genome browsers were first developed to visualize early draft genome assemblies for human and model organism systems [2]. Since their first inception, genome browsers have grown by leaps and bounds and currently there are over fifteen genome browser platforms available. Genome browsers have been created for specific data types, audiences, and visualization techniques. There are genome browsers for genomic data ranging from exon array analysis [3] too molecular pathways [4]. Each

genome browser typically has a unique feature that sets it apart from the others, such as customizing the way in which data is visualized, but overwhelmingly they share much of the same functionality.

Genome browsers arrange genomic data in the form of tracks, a horizontal viewing pane, where each track ordinarily displays one type of genomic data, such as exons or genes. Figure 2 displays an example of horizontal track visualization from the University of California Santa Cruz Generic Genome Browser (GBrowse) [5]. Horizontal tracks are the most widely accepted form of genomic visualization because it facilitates the vertical inspection and comparison of genomic data. This representation is likely inspired by the way in audio and video data is edited in horizontal tracks. Many genome browsers allow users to shuffle tracks and change the order in which they are vertically arranged. Even though horizontal tracks appear to be the most optimal manner to visualize genomic data, a few genome browsers, such as the CGView Server [6] and the Genome Projector [4], display data in the form of a circular genome, as seen in Figure 1. These circular genome visualizations are useful for showing how genes are connected and interact.

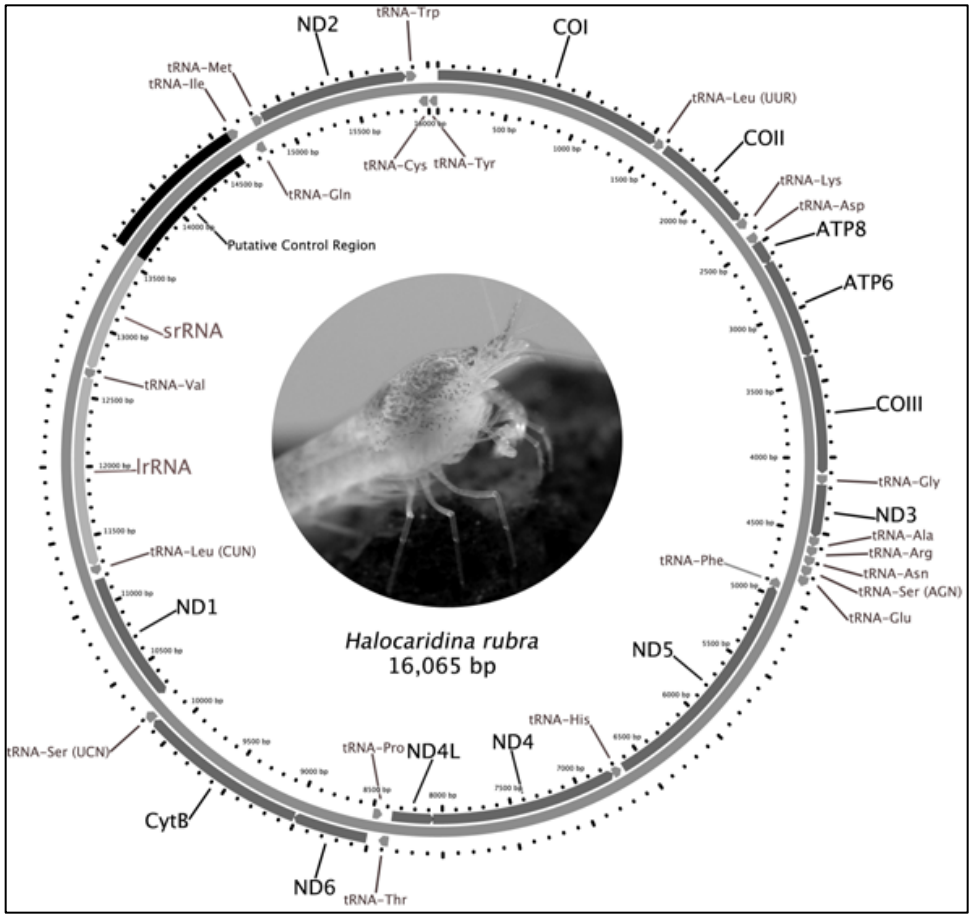


Figure 1 Circular genome visualization from CGView [6].

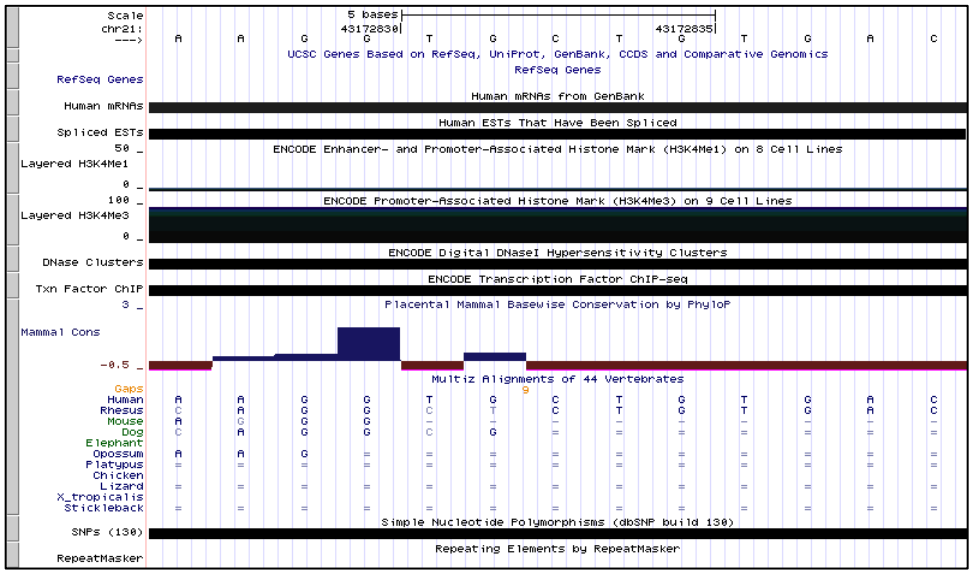


Figure 2 Horizontal track visualization from UCSC GBrowse [5].

Horizontal tracks are usually interactive by allowing users to manipulate the way in which the genomic data is viewed. Zooming is one of the most popular forms of interaction, allowing genomic data be viewed at different resolutions; e.g. from nucleotides to chromosomes. Zooming is typically done using a graphical widget, a form of clicking, or the mouse wheel. Genome browsers allow users to enter a fixed resolution for zooming or have cascaded levels of zoom where users can view their data in incremental levels of resolution. Current state-of-art genome browsers only allow users to view data at one resolution or level of zoom. That is, all tracks in the genome browser are viewed at the same level of resolution and at the same location. By offering only one level of zoom and resolution, current genome browsers offer minimal interaction with the user and this facilitate poor cross comparison of genomic data in multiple locations and levels of zoom.

In addition to users interacting with horizontal tracks through the use of zoom, tracks can be panned horizontally. That is, a user can perform some form of interaction, such as clicking and dragging, to move the track left or right to reveal further genomic data. Many current state-of-art genome browsers offer Graphical User Interface (GUI) elements to aid in horizontal panning. As well as horizontal panning, tracks can be navigated by search where a user enters a search query, often as a chromosome and position within the chromosome, to move the horizontal track to a

specific location of genomic data. This searching functionality allows users to quickly jump to locations of interest.

Often genome browsers will graphically display what portion of an organism a horizontal track is currently viewing with respect to the entire genome through the use of text or a GUI element. For clarification purposes, this feature will herein be referred to as reference location. Figure 3 displays horizontal track searching, zooming, panning, and reference location found in two different genome browsers, JBrowse [7] and GBrowse [5]. An in depth analysis of the GUI elements in Figure 3 is discussed further in section chapter 3.

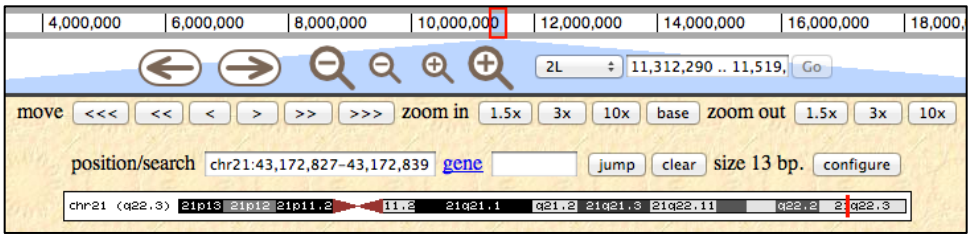


Figure 3 Example of searching, zooming, panning, and user reference location seen jBrowse (top) and UCSC GBrowse (bottom).

An important feature of many genome browsers is their portability across multiple computing platforms; Windows, Mac, *nix, etc. Some of the latest web-based genome browsers have become truly platform independent by visualizing data through a web browser. Current state-of-art genome browsers allow users to upload their own genomic data for visualization. Uploaded genomic data is often in a predetermined format, such as General Feature Format (GFF). GFF is an exchange format for feature description used to describe genes and other features associated with Deoxyribonucleotide (DNA), Ribonucleic acid (RNA), and Protein sequences specified by the Sanger Institute [5]. Besides GFF, many genome browsers accept other annotation formats.

In addition to GFF, a small number of genome browsers also allow users to view Binary Alignment Map (BAM) files. Sequence Alignment/Map (SAM) format files are the end product of genomic sequencing. The first step of genomic sequencing entails taking the organisms DNA and breaking it into numerous small fragments of DNA or reads. These reads are then run through a sequencing machine and a FASTQ file, the raw sequence data of the reads, is produced. The FASTQ file is then piped into an alignment program where the reads are aligned to the reference genome, where the

reference genome is the understood gold standard for the sequenced organism. The mapping of the sequence reads computes additional information for each read, such as insertions and deletions in the sequence. The newly computed aligned reads along with their additional information are output into a Sequence Alignment Map (SAM) file, hence the name. Given that each location of the reference organism can have many reads associated with it, SAM files can grow beyond 200 Gigabytes for Humans. Therefore, SAM files are compressed into index-able BAM files using SamTools, an open source sequence analysis command-line tool. A BAM file is the compressed binary version of the SAM format, a compact and index-able representation of nucleotide sequence alignments compressed using bzip [8].

Since BAM files are indexed, they facilitate fast random access of data through the use of SamTools, typically only requiring one seek to retrieve alignments in a region. This in turn allows users to traverse and search BAM files nearly instantaneously. Even though BAM files are compressed, Human BAM files can still exceed 50 Gigabytes. Given these large file sizes, BAM files are typically shared either through HTTP download or sent by physical copy in the form of hard drives. Among the most established genome browsers, GBrowse and Integrative Genomics Viewer (IGV) are the most widely used to visualize BAM files [9]. BAM files are visualized in a track representation, with each browser offering different visualization techniques.

The genomic features within a track can very often be hovered to reveal further information or clicked to either display or redirect a user to a new webpage for further information. Figure 4 displays an example of clicking a genomic feature inside JBrowse. A summary of the most commonly found features in genome browsers can be found in Table 1.

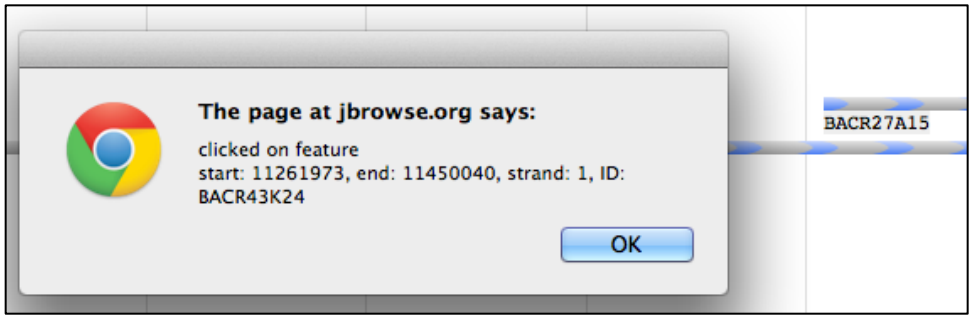


Figure 4 Clicking a feature in jBrowse to reveal further information.

Feature Name	Description
Horizontal Tracks	Genomic data is displayed in the form of horizontal stackable tracks.
Track Searching	Horizontal tracks are searchable by chromosome and sequence location.
Portability	The browser is compatible with numerous platforms.
Various File formats	The browser allows users to upload various genomic file formats.
Zooming	Horizontal tracks can be zoomed in and out to view genomics data at different resolutions.
File Upload	The browser allows users to upload their own data to the genome browser.
Clickable Features	Horizontal track features can be clicked or hovered to reveal further information.
Display Mode	The browser allows users to modify the way in which genomics data is displayed.

Table 1 Genome Browser Features: An overview of the general features found in genome browsers.

1.2 Stand-alone Genome Browsers

The implementation of genome browsers falls into one of two categories, stand-alone or web-based. Stand-alone genome browsers are software packages downloaded and installed onto a user's machine. Stand-alone genome browsers are typically implemented in the Java programming language. The Java programming language allows stand-alone genome browsers to be truly platform independent and are therefore, very easy to install. Stand-alone genome browsers are advantageous because users are not required to upload their genomic data to a server and thus, users do not have to wait for a response from a server. Additionally, since genomic data is stored on the users local machine, the probability of data compromise is somewhat lessened. Lastly, users can view files of any size through stand-alone genome browsers because they are not confined to file size restrictions found in web-based genome browsers, but that is not to say that they do not struggle visualizing files of size 50 Gigabyte+.

There are quit a few disadvantages to stand-alone genome browsers. First, although installation is traditionally very easy, users are still required to install a software package on their machine. Secondly, stand-alone browsers rely on the resources of the user's local machine for computation

and they are therefore restricted to the computational power of the user's machine. Lastly, genomic data is only accessible via the user's machine. If the user wishes to view their genomic data on another machine, they are forced to download and install the genome browser software package again and to move their genomic data, which hinders the users ability in accessing their data and does not facilitate collaboration.

1.3 Web-based Genome Browsers

With the advent of the web, nearly all software applications can now be easily accessed through a web browser. Even applications that were once thought of being stand-alone are now accessible through the World Wide Web (WWW). For example, Microsoft Office was initially thought of as a desktop only application, but now much of the same functionality can be found in Google Docs. Therefore, it is only a natural progression for genome browsers to take to the web. Over the past few years, web-based genome browsers have gained significant traction. A web-based genome browser is a website that offers roughly the same functionality as a stand-alone genome browser, but is accessible via the Internet. Web-based genome browsers often require a specific web browser for use, but more advanced genome browsers are cross compatible for all web browsers.

Web-based genome browsers have many advantages. First, users do not have to download and install any software tools to their machine. Secondly, web-based genome browsers allow users to quickly visualize their data with any machine that has an Internet connection. Additionally, web-based genome browsers facilitate more collaboration between users because data can be shared with the click of a mouse. Lastly, computation is lifted from the user's machine to the server or shared between the client and server.

As with stand-alone genome browsers, web-based genome browsers also have many disadvantages. First, web-based genome browsers often restrict the size of data that can be uploaded. Secondly, web-based genome browsers force users to be dependent upon a third party server. If the server goes offline then so does the user's ability to visualize and analyze their data. Current web-based genome browsers do not offer a way for users to store their information. This in turn forces users to re-upload their genomic data each time they wish to view it. The last disadvantage to web-based genome browsers is installation. If a user wants to install their own web-based genome browser

server, installation is typically very complex for the average user, requiring the installation of numerous third party packages.

Traditionally, as [7] points out, web-based genome browsers were implemented using the Common Gateway Interface (CGI) protocol, where web pages were generated based on requests from the user. This implementation forces the user to reload the webpage for each interaction. Furthermore, CGI leaves the majority of computational work to the server, forcing the server host to scale the computational power of the server to meet the needs of users.

With the rise of the Internet, there has been a new type of web application, Rich Internet Applications (RIA). RIAs as described in [10] are defined as combining the best user interface functionality of desktop software applications with the broad reach and low-cost development of Web application and the best of interactive, multimedia communication. More specifically, a RIA is a web application that typically mimics that of a desktop application through the use of some sort of plug-in such as Adobe Flash or Microsoft Silverlight or through advanced JavaScript. RIAs move away from forcing the web application to refresh the web page and instead employ some form of asynchronous communication between the server and the client, allowing interaction with the website without web page reload. This asynchronous communication is typically done in the form of Asynchronous JavaScript (AJAX). An overview comparison between web-based and stand-alone genome browsers can be found in Figure 5.

Web-based	Stand-alone
No Software Installation	Requires Software Installation
Requires File Uploading	No File Uploading
File Size Restriction	No File Size Restriction
Computation is placed on the Server	Computation is placed on User
Facilitates Collaboration	Limited Collaboration
Cross-platform	Sometimes Cross-platform

Figure 5 Comparison between web-based and stand-alone genome browsers. Green cells represent pros and red cells represent cons.

1.4 Current State-of-Art and Literature Review

As previously stated, over the past decade there has been an explosion of genome browsers in the field of genomics. Genome browsers have been developed for specific data types, visualization techniques, and audience. However, over the past few years only a few browsers have really gained traction. Therefore, only the most popular browsers will be discussed herein. For a comprehensive review of genome browsers see [2]. The author acknowledges that there are many genome browsers besides the ones discussed herein, but it is in the authors opinion that the following are the most widely used genome browsers: the Integrative Genomics Viewer (IGV) [9], University of California Santa Cruz Generic Genome Browser (GBrowse) [5], and JBrowse [7].

IGV, developed by the Broad Institute, currently dominates the landscape of stand-alone browsers and was first introduced in 2009 by [11]. IGV is written in the Java programming language and includes all of the features listed in Table 1, with a few additions. IGV allows users to share data across the Internet, define arbitrary sample annotation data, and the ability to view different data types together. One of the biggest features of IGV is its zooming ability, which mirrors that of Google maps zooming capabilities. IGV allows users to zoom from the entire genome all the way down to individual nucleotides. Additionally, IGV allows users to share data across the web and access data URL, IGV data servers, and DAS for data sources.

IGV is one of the only stand-alone browsers to allow users to visualize BAM files. BAM files can be viewed at multiple resolutions, all the way from the genome to the individual reads of the BAM file. The author attempted to load a sample BAM file but continually ran into issues where the program would not properly redraw the reads. However, the author were able to find an image of a BAM file being visualized along side a GFF file on the WWW; e.g. Figure 6. IGV displays BAM file reads as single gray bar with potential nucleotide variations in color. This visual queue makes it easy for users to distinguish possible mutations. Users can hover their mouse over a read to see information about the read such as mapping quality. Coverage in IGV is displayed at the very top of the track as a bar graph where the height of the bar corresponds to the number of reads in that region.

Figure 6 IGV visualizing a BAM and GFF file.

Currently GBrowse and jBrowse dominate the web-based genome browser arena. It should be noted that GBrowse is more then a web-based genome browser; it is an open source platform for genomic data. GBrowse is developed and maintained by the Genome Bioinformatics Group at UCSC and was first introduced in 2002 [5]. GBrowse uses the Bioperl toolkit, a library to for the life sciences written in the Perl programming language [12]. GBrowse offers the standard genome browser features found in Table 1. GBrowse's server is implemented in the Perl language and tracks are rendered out as static images. Many other browsers such as SynBrowse [13], SynView [14], and GBrowse_syn use the GBrowse backend server-side framework to build their own customized genome browser. GBrowse was originally developed using a CGI based approach where each interaction with the browser triggered the entire webpage to reload; offering little interaction. More recently with GBrowse version 2, tracks are rendered and sent via AJAX, allowing for a more seamless experience for the user without web page reload. One of the biggest features of GBrowse is the amount and type of data that can be displayed within the browser.

JBrowse is an open-source, portable, JavaScript web-based genome browser. JBrowse is novel in that it is the first genome browser to leverage rendering on the client side. The author's of JBrowse ran experimental benchmarks on both client and server side rendering. More specifically they

compared JBrowse's client-side processing and rendering with HTML elements versus GBrowse's server-side processing and rendering using tiled images. Their experiments yielded that client-side processing is much faster then that of server-side with respect to computational time and disk space usage. However, even though JBrowse's primary rendering uses client-side processing, it still renders out images on the server side for display by the client. Therefore, JBrowse splits rendering between the client and server. It should be noted that JBrowse harnesses the GBrowse application framework for retrieving data [7].

1.5 Improvements to Genome Browsers

The state of art in genome browsers can be improved upon drastically. Neilsen et al. state next generation genome browsers will need to include easy cross-platform access, data and display customization, the ability to perform on-the-fly computation within the visualization, and security of user genomic data [2]. Neilsen et al. further convey that visualization tools will need to be capable of integrating diverse data sources, visualize large scale data sets, seamlessly navigate across different resolutions, and improve integration between automated computation and analysis through user refined searching.

As Neilsen et al. bring to light, one of the biggest problems within the web-based genome browser community is security. There is yet to be a stand-alone or web-based genome browser that securely stores a user's genomic data. For web-based genome browsers this forces users to resubmit their data each and every time they wish to browse it, which is inefficient, redundant, and insecure. Additionally, as the cost of next generation sequencing continues to drop, physicians will need a secure manner to share clinical patient data. Therefore, future genome browsers must employ user accounts and security measures.

While the improvements that [2] points out are valid, there are more fundamental improvements to the way in which users interact with their genomic data that should be consider by future genome browsers. Many genome browsers have added features, but have not given much thought about the way in which users will interact with those features. For example, more recent genome browsers such as [2] provide users with Google Map like zooming capabilities. At first glance, this additional feature seems to add to the user experience, but in reality, it adds an extra task for the user by

forcing the user to find the correct resolution at which to view their data. This functionality could potentially be improved by constraining zooming to a few specific levels of zoom.

Additionally, to further improve user interaction, future genome browsers should explore the use of new visualization techniques. Current genome browsers represent genomic data as static, two-dimensional objects on a track. Genome browser visualization could potentially be improved by including three-dimensional visualization, which has yet to be explored by the genome browser community. Furthermore, genome browsers should examine other visualization techniques to help visually convey information to the user, such as transparency, gray scale coloring, etc.

Moreover, as [15] points out, most genome browsers are reserved for the most skilled computational biologists leaving non-specialist users overwhelmed by the shear amount of options and information presented to them. Therefore, with the advent of personalized genomic medicine around the corner, there is a growing need for an intuitive, easy to use genome browser for the masses.

Next generation genome browsers must allow users to share and view BAM files in a secure manner. IGV and GBrowse both allow users to visualize BAM files, but neither offers a hosting solution. Currently, there is no known solution for BAM file hosting. In developing the Gaggle browser in 2010 Bare et. al. stated that the web was an enticing area to pursue but due to file size constraints they decided to develop a stand-alone browser [16]. Thus, next generation web-based genome browsers should provide an efficient manner in which to visualize large genomic data files. Current web-based genome browsers force users to upload entire files for visualization, which can often take days due to size constraints. Therefore, next generation web-based genome browsers should explore streaming file visualization.

1.6 STATEMENT OF THE PROBLEM

Genomic analysis, interpretation, and visualization are an aspect to genomics that has yet to reach its full potential. More specifically, current genome browsers can be improved upon in many ways. Genome browsers do not provide a secure web-based solution for users to store and visualize their genomic data. Genome browsers have not investigated the use of three-dimensional visualization techniques. Genome browsers have poor user interfaces, feature designs, and overall design that

hinder users ability to efficiently find and analyze data. Lastly, web-based genome browsers have not explored streaming BAM file visualization or BAM file hosting.

Any improvements in the genome browser field will help benefit biologists, scientists, and doctors by providing an improved way to analyze, interpret, and visualize genomic data. With the lowering cost of sequencing technologies, there is a growing need for an improved web-based genome browser in the field of genomics.

1.7 STATEMENT OF OBJECTIVES

The objectives of the work described herein consist of the following: (a) develop a web-based next generation genome browser, NextBrowse, that provides a secure central repository to analyze, interpret, and view data (b) provide a web-based solution for genomic visualization with improved user interfaces through the use of three-dimensional interaction and improved GUI design (c) present novel ways to refine the visualization techniques and GUI elements found in current genome browsers that improve upon genome browsing (d) provide an improved web-based secure solution for BAM file hosting and BAM file streaming visualization.

1.8 OVERVIEW OF THESIS

This thesis is organized as follows: Chapter 2 provides an overview of the technologies, software architectures, and implementation of NextBrowse. Chapter 3 presents the features and improvements of GUI elements in NextBrowse. Chapter 4 covers the improved visualization techniques found in NextBrowse. Chapter 5 describes NextBrowse through a walkthrough of the software. Chapter 6 gives an evaluation of NextBrowse that includes testing, self-evaluation, and documentation. Lastly, Chapter 7 presents conclusions, contributions, and future work.

Chapter 2

Architecture and Technologies

This chapter provides an in depth examination of the many technologies and software architectures used by NextBrowse.

2.1 Technologies and Architecture Overview

As mentioned in Chapter 1, there are two categories of genome browsers, web-based and stand-alone. NextBrowse is a web-based genome browser or RIA that harnesses many programming languages, architectures, and technologies for its implementation. An overview of the programming languages and technologies used in the implementation of NextBrowse is depicted in Figure 7. As in traditional software architectures, web application architectures make use of a front-end and a back-end software application layer. The front-end of web applications is referred to as all parts of the web application that are directly visible to the user and the back-end as the database and all application code used to generate the front-end.

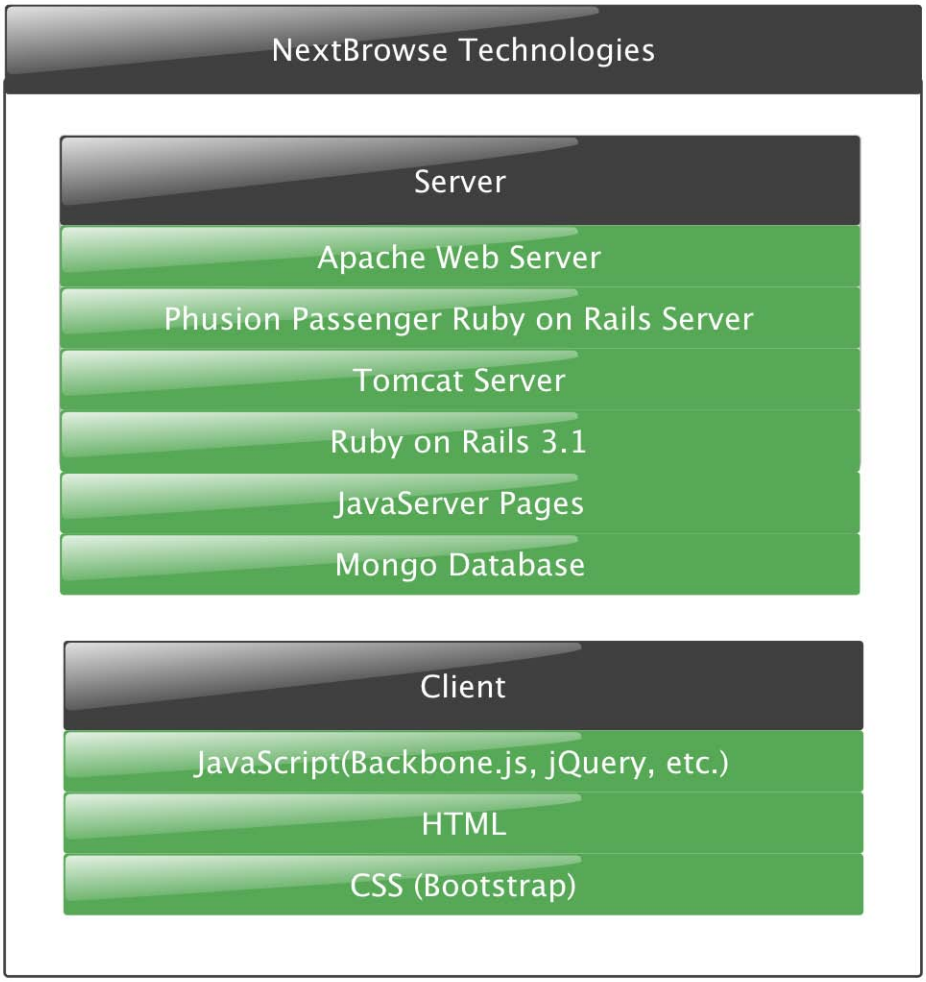


Figure 7 An overview of the different client and server technologies used by NextBrowse.

2.2 Server-side Architecture and Stack

NextBrowse's server-side software layer is primarily implemented in the Ruby on Rails (ROR) version 3.1 web application framework and partially in JavaServlet application to handle BAM file streaming. NextBrowse employs use of the Mongo NoSQL database for storage. NextBrowse can be deployed in any ROR and JavaServlet compatible server, such as Apache. The author chose to deploy NextBrowse using the Apache Web Server found on Apple Mac OS Lion along with the Phusion Passenger ROR web server and Tomcat web server. The following subsections highlight the previously mentioned server side technologies in closer detail.

2.2.1 Ruby on Rails

A web application framework is a software framework built to alleviate software developers with the overhead of common web application development by abstracting out common practices and low level details. There are a number of web application frameworks available today, such as Django, Struts, and ROR.

ROR is an open source web application framework, first created by David Heinemeier Hansson, written in the Ruby programming language. ROR is currently at version 3.1 and follows a Model-View-Controller software architecture, where application programming is broken down into the model, view, and controller software layers. As with traditional Model-View-Controller software architectures, each software layer has its corresponding purpose. The model layer defines the data to be stored in the database, provides an abstraction of the database, and handles all behavior of the data. The view layer renders user interfaces and data from the model to the user. The controller layer acts as the glue between the model and view layer. The controller receives input from the user, fetches data from the model, and renders out the appropriate view to the user.

ROR can store data in a variety of databases including both relational and non-relational. However, unlike traditional web application development where developers issue direct Structured Query Language (SQL) commands on the database, ROR uses an Object Relational Mapper (ORM). An ORM abstracts out the database tables along with relationships between tables into Objects. Therefore developers manipulate objects corresponding to tables in the database instead of issuing SQL commands.

ROR follows the Representational State Transfer(REST) web architecture, that was first introduced by [17]. RESTful architecture defines web services where Uniform Resource Locators (URL) map to a specific resource on the server dependent on the Hypertext Transfer Protocol (HTTP) method sent to the server (GET, PUT, POST, or DELETE). That is, ROR will map specific URLs to actions within a controller, which will retrieve data from the model if needed, and will then render out a view. Figure 8 displays an overview of the architecture of ROR.

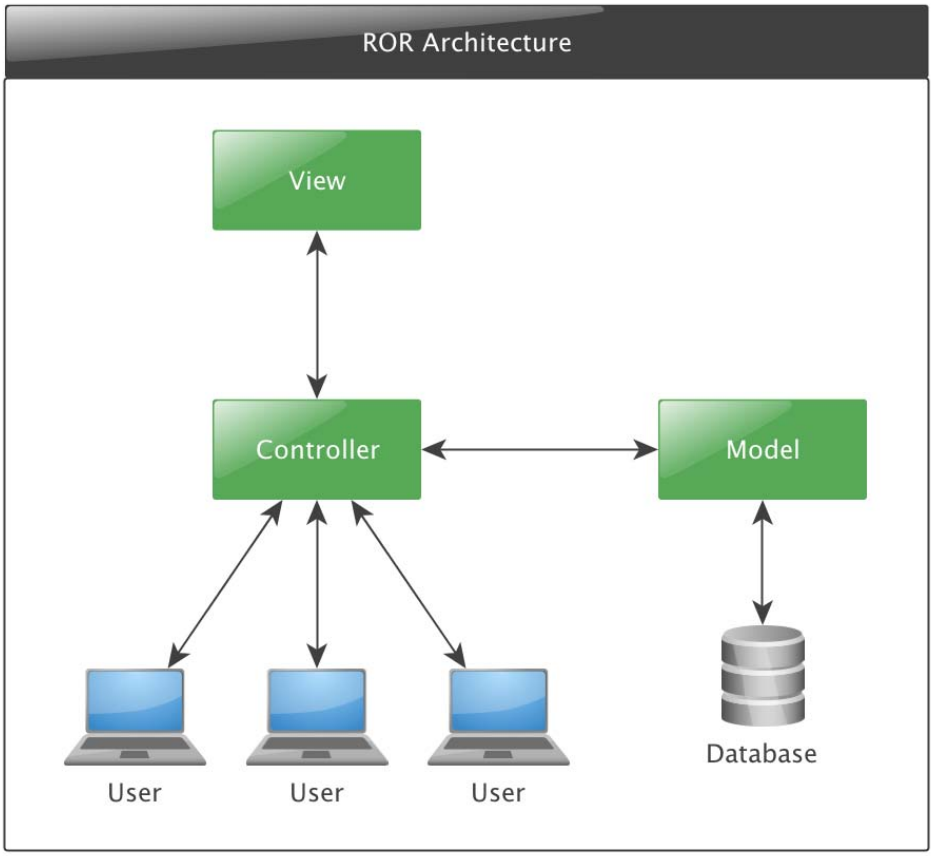


Figure 8 Requests from users are sent to the application controller where they are then redirected to the appropriate view and data from the database is retrieved.

2.2.2 Ruby Gems

The Ruby programming language makes use of Ruby libraries called Ruby Gems, which can be easily downloaded and installed. Given that ROR is written in Ruby, ROR can also make use of Ruby Gems. There are thousands of Ruby Gems available today, for everything from security to file uploads. Therefore, instead of reinventing the wheel, NextBrowse utilizes numerous Ruby Gems, all of which can be viewed in Table 2.

Gem Name	Description
Devise [18]	A flexible authentication solution for Rails based on Warden.

Mongo [19]	A Ruby driver for MongoDB.
Mongo_mapper [20]	A Ruby Object Mapper for Mongo.
Jnunemaker-validatable [21]	Library for adding validations.
mm-devise [22]	MongoMapper for the devise gem.
Will_paginate [23]	Pagination library for Rails 3, Sinatra, Merb, DataMapper, and more.
json [24]	JSON implementation as a Ruby extension in C.
Sprockets [25]	Rack-based asset packaging system.

Table 2 Ruby Gems used by NextBrowse.

2.2.3 Mongo Database and Database Design

Relational (SQL) and non-relational (NoSQL) databases can be used with ROR, as well as a number of different types of databases, such as SQLite, MongoDB, and MySQL. The NoSQL database MongoDB was chosen for NextBrowse's implementation. MongoDB uses a key-value storage schema, where documents are stored in the database on disk much the same way as hash maps [26]. MongoDB was chosen over a relational database because NoSQL databases do not require fixed table schemas. This in turns makes non-relational databases faster than traditional relational databases because the database only has to worry about dumping data as opposed to making sure the data fits correctly into the table. Furthermore, NextBrowse must be able to rapidly retrieve reference genome sequence data, thus, speed was sought after over consistency guarantees.

Although NextBrowse uses a NoSQL database for storage and it does not have a strict Relational Schema, it does still have tables. Figure 9 displays the databases loose schema along with its relationships. Due to the massive size of reference sequence data, reference sequence data is stored in blocks. That is, the reference sequence is broken down into small chunks and stored. This in turn allows for quick and efficient searching within the database.

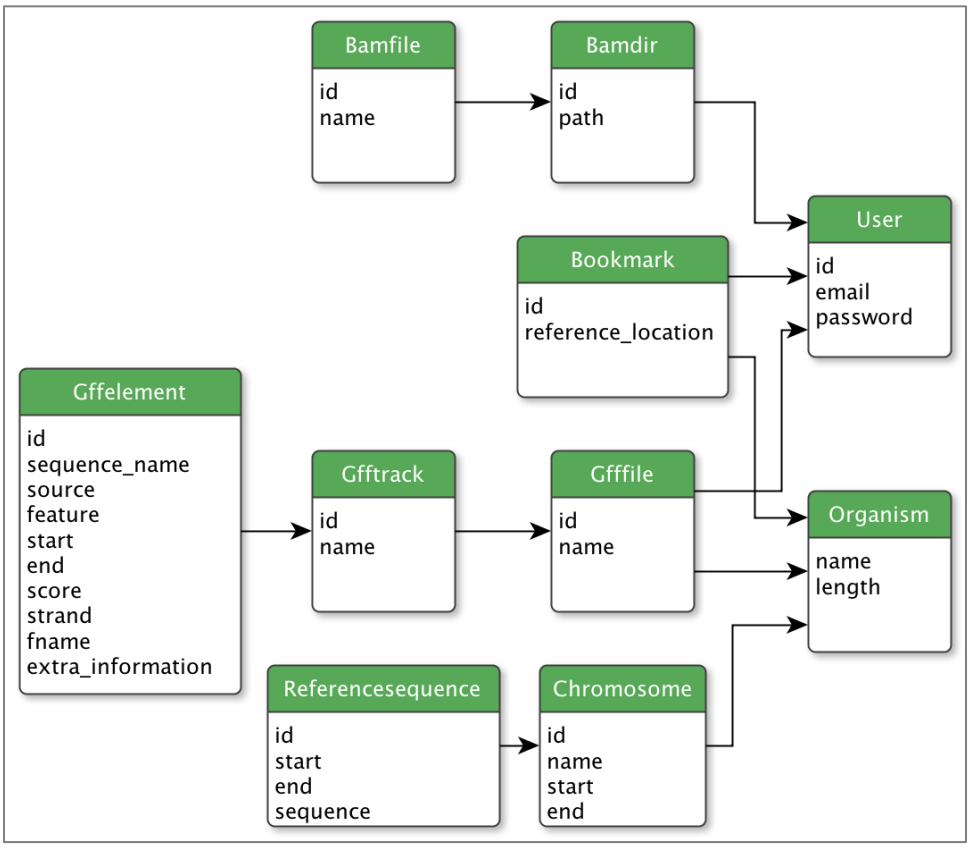


Figure 9 An overview of the loose database schema used by NextBrowse.

2.2.4 JavaServlet Application

While the main application is written in ROR, NextBrowse uses a JavaServlet application to handle all BAM file streaming. The JavaServlet application runs on a Tomcat server on the same machine as the ROR application under Apache through a proxy to port 8080. We sought out using JavaServlets so that BAM file streaming can have its own dedicated server to handling the heavy IO.

BAM file streaming works as such:

1. User uploads an index file that is stored in a temporary location on the server.
2. User selects their sorted BAM file.
3. User adds the BAM track specified by their selected BAM file.
4. JavaScript makes a request to the JavaServlet application for a specific range of BAM reads.

5. JavaServlet application calls SamTools to determine the byte range of data to fetch from the BAM file selected by the user, which is determined by the user uploaded index file. The byte range is returned to the client JavaScript call.
6. The client uses HTML5 to fetch the requested byte range of data and pushes it to the server. If more data is needed, the server will return with another byte range.
7. The JavaServlet application constructs all of the HTML5 pushed chunks of BAM file data and extracts the BAM reads using SamTools. These reads are returned to the JavaScript client.
8. The front end renders out the visualized reads.

It should be noted that users can visualize multiple BAM files at from their client machine and that BAM files must be sorted format for file streaming to properly work. Figure 9 depicts a graphical representation of how this process works.

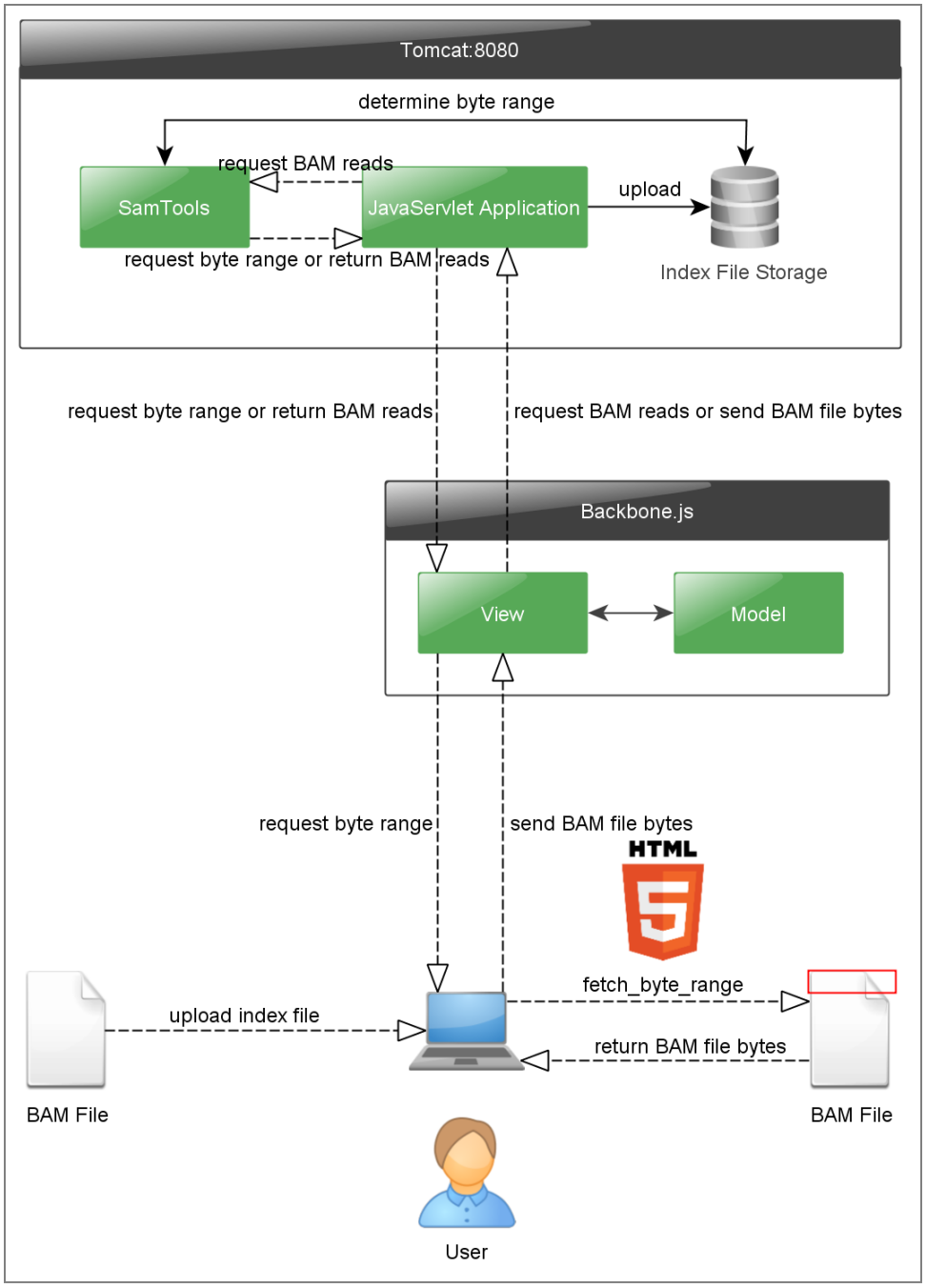


Figure 10 Architectural overview of BAM file streaming with Tomcat, Backbone.js, and HTML5. First users upload their index file. Next, Backbone.js sends a request for BAM reads. The JavaServlet application calls SamTools to determine the byte range of data to request from the uploaded index file. The byte range requested is then propagated back to Backbone.js and using HTML5 the byte range of data is sent back to the JavaServlet application. The JavaServlet application then either sends more requests for BAM data or passes the constructed BAM data to SamTools. SamTools reads the BAM reads from the constructed BAM data and returns the reads to the Backbone.js view where they are stored in the model. Storage of reads in the model triggers an update to re-render the view.

2.2.5 Apache Web Server, Phusion Passenger, and Tomcat

NextBrowse can be distributed on a number of ROR enabled web servers. The web available version, http://www.nextbrowse.vbi.vt.edu, is deployed using Apache Web Server and Phusion Passenger. Phusion Passenger is an easy and robust deployment server to deploy Ruby on Rails applications and Phusion Passenger works with both Apache and Nginx webservers and can be found on the web at http://www.modrails.com. The web available version of NextBrowse is deployed on a Mac Mini with 2.3 GHz Intel Core i5 and 2GB of ram running 64-bit Lion 10.7.2. Instructions on how to install NextBrowse can be found in the instruction manual online at http://nextbrowse.vbi.vt.edu/.

2.2.6 SamTools

To retrieve BAM file reads from BAM files, NextBrowse uses SamTools. As previously stated, SamTools is a software utility package that provides various utilities for manipulating alignments in the SAM format, including sorting, merging, indexing and generating alignments in a per-position format. BAM files are indexed binary compressed versions of SAM files. SamTools uses the freely available bzip data compressor to compress SAM files into BAM. Additionally, SamTools can be used to retrieve BAM file data using random access since BAM files are indexed. Each BAM file comes with an index file that is used by SamTools to rapidly search the BAM file for data. Given a chromosome and region of interest, SamTools will return the given reads from the specified region. SamTools is used by NextBrowse to grab BAM file data from BAM files on the server for visualization and to determine the byte range of data in a BAM file to retrieve [8].

The one caveat with SamTools is that there is no way to limit the number of reads or coverage of reads returned. Therefore, SamTools will sometimes return a very large amount of reads for a given location. Drawing one thousand or more reads will bring NextBrowse graphical rendering to a grinding halt. This is primarily due to two reasons. First, drawing one thousand or more reads using CSS, HTML, and JavaScript requires heavy manipulation of the Document Object Model (DOM) and will thus bring visualization to a stand still. Secondly, displaying one thousand or more elements to a user is trivial since it is highly unlikely that a user can make sense of one thousand or more elements at once. Therefore, instead of potentially visualizing and returning thousand of reads, it is more feasible to take a sampling of reads. NextBrowse takes a uniform sampling of size three hundred from the reads returned by SamTools. Limiting the number of reads returned by SamTools

improves the drawing speed of visualizations drastically and decreases the number of reads that users are left to examine.

2.2.7 JSON

JavaScript Object Notation or JSON is a lightweight, text-based, data-interchange format specified by RFC 4627 [27]. NextBrowse uses JSON to return data to the client whenever an AJAX request is sent to the server.

2.2.8 Rationale for Server Technologies

There are many reasons for use of the previously described software technologies over others. The following list highlights a few of the reasons for the selection of the chosen server-side technologies:

1. Ruby on Rails is designed for rapid prototyping and development.
2. Ruby on Rails handles database relations through an ORM, abstracting out complex SQL queries.
3. MongoDB is one of the fastest industry graded databases in existence.
4. SamTools is used throughout the genomics community as the go to tool for examining BAM files.
5. JSON due to ROR's ability to easily generate JSON and for JavaScripts easy ability to parse JSON data.
6. JavaServlet application so that all file heavy IO can be handled by a dedicated server and application.

2.3 Client-side Architecture and Stack

The NextBrowse client-side software layer uses JavaScript, Hypertext Markup Language (HTML), and Cascading Style Sheets (CSS) 3. HTML and CSS are defined by the World Wide Web Consortium (W3C) as two of the core technologies for building web pages. HTML is a formatting language for describing the structure of web pages. CSS is a formatting language for describing the presentation of web pages and includes everything from font size to page layout. Generally HTML and CSS are supported by all major web browsers, but each browser has it's own caveats that it will and won't support [28].

JavaScript is a web-based scripting language that is executed by the browser either after a web page has loaded or from a user triggered event such as a mouse click or mouse hover [28]. JavaScript can be used to manipulate the structure (HTML) and presentation (CSS) of a webpage. Therefore, JavaScript allows for the creation of extremely dynamic webpages.

2.3.1 jQuery

jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animation, and Ajax Interactions for rapid web development [29]. NextBrowse uses jQuery version 1.6.2. Like Ruby Gems, jQuery has a number of additional libraries, jQuery plugins, which can be used. NextBrowse uses a host of jQuery plugins and other JavaScript libraries, all of which can be viewed in Table 3. The rest of this subsection highlights these plugins and describes the manner in which NextBrowse utilizes them.

jQuery Plugin or JavaScript Library	Description
jQuery Plugin – jQuery UI [30]	"Provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets, built on top of the jQuery JavaScript Library,…"
jQuery Plugin – jQuery Transform	Custom written plugin to provide 3D and 2D manipulations of DOM elements through CSS.
jQuery Plugin – jQuery qTip [31]	"qTip is an advanced tooltip plugin for the ever popular jQuery JavaScript Framework."
Underscore.js [32]	"Underscore is a utility-belt library for JavaScript that provides a lot of the functional programming support…"
Backbone.js [33]	"supplies structure to JavaScript-heavy applications by providing models with key-value binding…"
Apprise.js [34]	"The attractive alternative for jQuery."
Log4JavaScript [35]	"JavaScript logging framework based on the Java logging framework log4j."

Table 3 Listing of all of the JavaScript libraries and frameworks used by NextBrowse.

NextBrowse uses the jQuery UI plugin, which provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets [30]. NextBrowse uses jQuery UI for the majority of the GUI elements in NextBrowse. The following elements use jQuery UI.

- Top track slider: jQuery UI Slider
- Top track reference: jQuery UI Draggable
- Zooming UI Element: jQuery UI Slider
- GFF Buttons: jQuery UI Draggable
- Track Horizontal Slider: jQuery UI Draggable

NextBrowse also makes use of the jQuery Transform plugin and jQuery qTip plugin. The transform plugin will be discussed in the next section. jQuery qTip is used for all tooltips within NextBrowse, from displaying information for BAM file reads to displaying useful tips to users about icon functionality.

NextBrowse uses Underscore.js, Backbone.js, and Log4Javascript. Backbone.js will be discussed further in a later section. Underscore.js is a library that provides many commonly found functions in other programming languages, such as array manipulation [32]. Underscore.js is primarily included in NextBrowse because of Backbone.js's dependence of Underscore.js.

NextBrowse uses log4Javascript for debugging. Log4Javascript is a logging framework for JavaScript that makes it extremely easy for developers to log messages [35]. NextBrowse provides a flag, DEBUG, that can be set to either 1 or 0 inside of setup.js to turn debugging off or on. If DEBUG is set to 1, NextBrowse will log all actions of the application within the Log4Javascript window. The log messages within NextBrowse are structures as follows:

time: type_of_log – file_name : method_name -> message
time = the time of the log message
type_of_log = the type of log message (error, debug, etc.)
file_name = the file name that the log came from
method_name = the method name that the log came from

message = the message logged

An example log message would like:

15:36:59 INFO – setup.js:get_user_bamdirs -> successfully got bamdirectory data for user.

A developer would recognize that at 15:36:59 the application logged an info message stating that the get_user_bamdirs() method inside of setup.js successfully retrieved bamdirectory data for the user. This built in debugging capability make it extremely easy to debug the application. Additionally, error messages, such as an inability to retrieve an AJAX request are all logged to the log4Javascript window and appear in bright red. The log4Javscript window has many features built in, such as the ability to filter by type of message logged or searching of the output window. An example output of NextBrowse's log4Javascript window can be viewed in Figure 11.

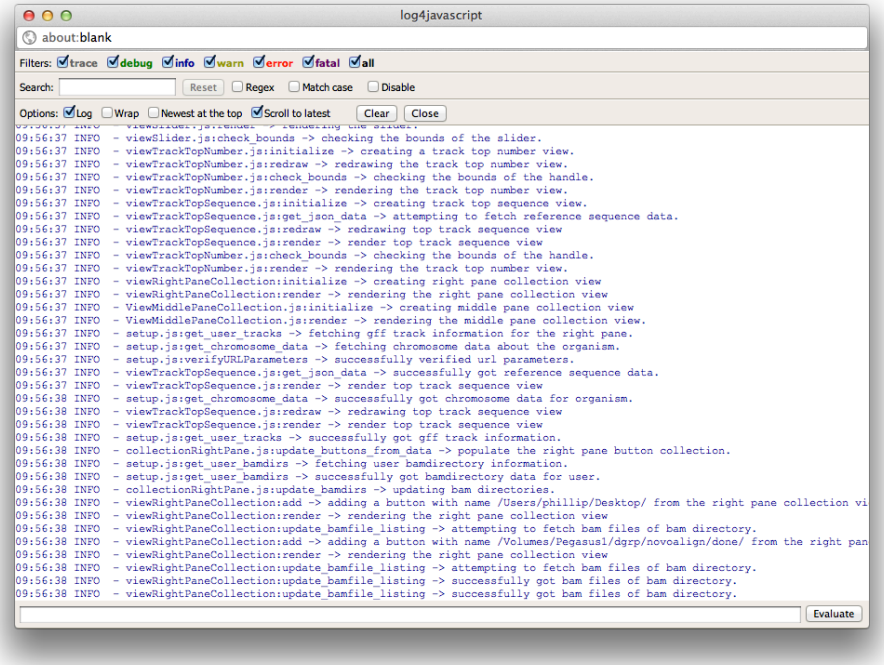


Figure 11 Log4JavaScript example output window as NextBrowse runs.

2.3.3 Achieving 3D on the Web

Achieving 3D animation on the web was first introduced through the use of the animated Graphics Interchange Format(GIF) images. As time progressed, new technologies were introduced into the web, such as WebGL and Adobe Flash. WebGL is a software library that extends JavaScript to enable 3D graphics. Adobe Flash is a multimedia platform to add interactivity to websites and is developed by Adobe Inc. Recently, advanced CSS, CSS 3, has been created to allow 3D transformations of DOM elements. Use of Adobe Flash and WebGL were explored, but found to be over kill for the amount of animation desired. Additionally, Adobe Flash and WebGL do not allow manipulation of DOM elements. Therefore, due to the subtle nature of the amount of 3D animation desired, the author chose to use CSS 3 for NextBrowse 3D animations.

In CSS 3, 3D animations of a DOM element can be performed through the use of rotations, translations, and scaling. CSS 3 is still in its early stages of development and therefore, most web browsers only support a handful of CSS 3 properties. As mentioned, every web browser has their own caveats with CSS and therefore each browser requires specific CSS for proper presentation. This is problem is exemplified in Figure 12, which demonstrates how to apply a simple 45 degree rotation transformation in various browsers.

```
div
{    transform: rotate(45deg);
     transform-origin:20% 40%;
     -ms-transform: rotate(45deg);
     -ms-transform-origin:20% 40%;
     -webkit-transform: rotate(45deg);
     -webkit-transform-origin:20% 40%;
     -moz-transform: rotate(45deg);
     -moz-transform-origin:20% 40%;
     -o-transform: rotate(45deg);
     -o-transform-origin:20% 40%;
}
```

Figure 12 Demonstration on how to transform an element in CSS3 in multiple browsers.

As Figure 12 exemplifies, developing for cross compatibility of web browsers can be extremely tedious. Therefore, due to this constraint, NextBrowse is developed exclusively for the Google

Chrome web browser. As more web browsers begin to adopt CSS 3, it is in the hopes of the authors to extend NextBrowse to other web browsers.

One of the problems within CSS 3 is that continuous animation cannot be achieved. This is due to the fact that CSS only defines static properties about the presentation of the DOM. To further illustrate this idea, examine the code in Figure 13.

```
a { transform: rotate(90deg); }
```

Figure 13 Example of transforming all a elements 90 degrees.

Figure 13 states that all <a> structural elements of the page should follow that of a 90 degree rotation. What if we want to rotate all <a> elements another 90 degrees? If the same property is set, none of the <a> elements will change because the elements are already set to a 90 degree rotation, hence the nature of CSS. However, if another 90 degrees of rotation is added to the element then all <a> structural elements will be rotated another 90 degrees. What if we would like to continually rotate the element? We could extract the CSS style in JavaScript, parse the style, and then update the style, but this would be quite a lot of overhead for hundreds of elements. Therefore, the jQuery plugin, jQuery Transform, was created.

jQuery Transform extends the jQuery object to allow continuous CSS 3 animations of any DOM element. jQuery Transform operates by storing all 3D information of a page element in the DOM element itself through jQuery's .data() key-value storage. The following line of code is an example use of jQuery Transform that rotates all <table> elements by 180 degrees along the x-axis and sets the z coordinate to 60.

```
$("table").transform("setZ", 60).transform("rotateX", 180);
```

Figure 14 Example of transforming all table elements by translating along z-axis by 60 degrees and rotating along the x-axis by 180 degrees.

The beauty of the jQuery Tranform plugin is that DOM elements remember their 3D locations, which in turn allows continuous 3D animations to be performed. To further exemplify this, Figure 15 displays transforming an element through 30 degrees of rotation about the x-axis each time a user clicks a page element with id button.

```
$("#button").click(function(){
     $("a").rotateX(30);
});
```

Figure 15 Example of transforming all a elements by a 30 degree rotation each time an element is clicked that has the id of button.

As previously stated, NextBrowse is developed to operate using the Google Chrome web browser. Therefore, jQuery Transform is also developed for the Google Chrome web browser. It is in the author hopes to extend jQuery Transform to manipulate DOM elements of other web browsers in the future.

2.3.4 Backbone.js

Backbone.js is a JavaScript framework that supplies structure to JavaScript-heavy applications by providing models with key-value binding and custom events, collections with a rich API of enumerable function, views with declarative event handling, and connects it all to existing application over a RESTful JSON interface. More specifically, Backbone.js provides a core front-end model view controller architecture for JavaScript heavy applications [33]. Backbone.js relies heavily and is built upon Underscore.js, the JavaScript utility library discussed previously [32].

The Model-View-Controller architecture of Backbone.js follows closely that of other Model-View-Controller architectures with a few slight differences. The model software layer houses the data to be rendered as well as a host of other user utilities, such as data validation, data conversion, and access control. Models can be populated manually or from a RESTful JSON URL. Collections are sets of models that can be bound to models to listen to changes to the models they contain. Collections are populated manually or from a RESTful JSON URL.

Views render the data stored in collections and models. Views can be bound to a collection or model to listen for changes. A single view ties into a single DOM element. Unlike a traditional Model-View-Controller architecture, views in Backbone.js communicate directly with the model layer, thus, the views exhibit much of the controller functionality found in traditional Model-View-Controller architectures. Controllers in Backbone.js are known as routers. Controllers provide methods for routing client-side pages and connecting them to their respective actions and events. Since NextBrowse operates as a single webpage, it does not use Backbone.js Controllers or Routers.

Nearly every page element in NextBrowse has it's own view and respective model that it listens to for changes. NextBrowse also uses collections to keep track of models. Table 4 displays an overview of NextBrowses Backbone.js views, models, and collections.

Name	Type	Description
collectionMiddlePane	C	Store the collection of track models, both BAM and GFF tracks.
collectionRightPane	C	Store the models of the right pane, primarily modelRightPaneButton.
modelMiddlePaneTrack	M	Stores all information to be rendered out by a track.
modelRightPaneButton	M	Stores information about a right pane button.
modelSlider	M	Stores all information used by the slider.
modelTrackTopNumber	M	Stores all information used by the top reference location number track.
modelTrackTopSequence	M	Stores all information used by the top reference location sequence track.
viewMiddlePaneBamTrack	V	Renders out a BAM track from the information stored in modelMiddlePaneTrack. Updates based on changes made to the model.

viewMiddlePaneTrack	V	Renders out a GFF track from the information stored in modelMiddlePaneTrack. Updates based on changes made to the model.
viewMiddlePaneCollection	V	Renders out all of the models round in collectionMiddlePane.
viewRightPaneButton	V	Renders out a button from modelRightPaneButton
viewRightPaneCollection	V	Renders out the button models found in collectionRightPane.
viewSlider	V	Renders out a slider based on data in the modelSlider.
viewTrackTopNumber	V	Renders out a reference location based on modelTrackTopNumber.
viewTrackTopSequence	V	Renders out a reference location sequence based modelTrackTopSequence.

Table 4 Listing of all Backbone.js models, views, and collections used by NextBrowse.

2.3.5 CSS

NextBrowse makes use of Twitter Bootstrap CSS for primary styling. Twitter Bootstrap is an open-source CSS style that automatically overrides the styling of HTML elements [36].

2.3.6 File Uploads

NextBrowse allows two different types of files to be uploaded to the server. The first type of file is GFF. GFF files can be uploaded as long as they conform to the specification put in place by the Sanger Institute [37]. The raw files that are sent to the server are not stored on the server. Instead, they are parsed and stored in their relevant data structures in the database. GFF files are parsed into GFF tracks, where each track corresponds to a different type of feature found within the GFF file.

The second type of file uploaded to the server is a FASTA file. A FASTA file is a text-based format for representing nucleotide sequences. FASTA files are used to hold the reference sequence of model organisms. NextBrowse only allows users with administrative access to upload FASTA files. Again, once the file is uploaded it is not stored in the database, but instead its parsed contents are

stored in the database. Since only users with administrative privileges are allowed to upload reference sequences for model organisms, it is the administrator's responsibility to upload the reference sequence FASTA files for all reference organisms. As one can imagine, FASTA files can be extremely large in size, often in the Gigabyte range. Therefore, if a user wishes to deploy their own NextBrowse application, it is highly recommended that they have physical access to the server so that they will not have to transfer reference files, but can instead load them locally.

2.3.7 Rendering Tracks

As previously mentioned, tracks are rendered entirely in CSS, HTML, and JavaScript. When a new track is added to the middle-viewing pane of NextBrowse, a new Backbone.js view and Backbone.js model are created that correspond to a track. Next, an AJAX request is sent to the server to retrieve the data to be displayed in the track and this data is stored in the corresponding Backbone.js model. NextBrowse retrieves enough data to create a buffer for each track. That is, more data is returned then the user can view at once. This lowers the amount of requests that must be sent to the server for data and helps facilitate seamless navigation.

Once the track model receives the AJAX data, the view tied to the model dynamically renders out the data in the form of HTML page elements. More precisely, each piece of data is rendered as a div element. Each div's properties, such as width and height, are dynamically determined. Once the track features have been created, any CSS styles or events, such as mouse hovering, are added. Given that track elements are rendered using HTML and CSS, there will sometimes be misalignment in track elements. More specifically, since each elements nucleotide position is dynamically calculated and is cast to an integer, there are some instances where there will be round off error.

Although GFF and BAM file tracks use the same model, they differ in rendering of their 'reference' and 'gene' level of resolution. NextBrowse allows users to view tracks at four different levels of zoom: genome, chromosome, gene, and reference. The furthest zoomed out level being the genome, and the reference view being close enough to view the individual nucleotides. These levels of zoom are further examined in the following chapter. GFF tracks render each element in the gene view as a rounded div. The reference view of GFF tracks is simply a styled HTML table element holding the reference sequence.

The BAM track gene level view renders out page elements in the form of div elements, except that they are not rounded. The reference view of BAM tracks is again a table, except the cells of which are dynamically populated with the BAM reads. Experimentation was done of drawing multiple tables for each individual BAM file read, but since there is inherently round off error with the positioning of each read, the idea was abandoned. By using one table to draw the reads, it guarantees that BAM file reads are stacked and positioned correctly. However, as with GFF elements, there will inherently be some form of round off error when positioning the table as well.

2.4 Client and Server Communication

The client-side and server-side software layers are closely tied together. The client and server communicate via AJAX requests and the server returns data in the form of JavaScript Object Notation (JSON) or plain text. Figure 16 presents an overview of how the client and server layers communicate. Any other interaction with the browser follows the same general sequence of events between the client and server:

1. User interacts with a Backbone.js view or DOM element.
2. The Backbone.js view registers user interaction and sends a request to the server for data if needed.
3. The URL that is sent is mapped to the correct ROR controller or JavaServlet application, dependent upon the type of data being visualized.
4. The controller processes the request and fetches data from the model layer or the JavaServlet application fetches data from the client side BAM file.
5. The data from the database or BAM file is returned to the Backbone.js view in the form of JSON or text.
6. The returned data is stored in the corresponding Backbone.js model.
7. Updating the model's data triggers the Backbone.js view to redraw or update any data tied to it.

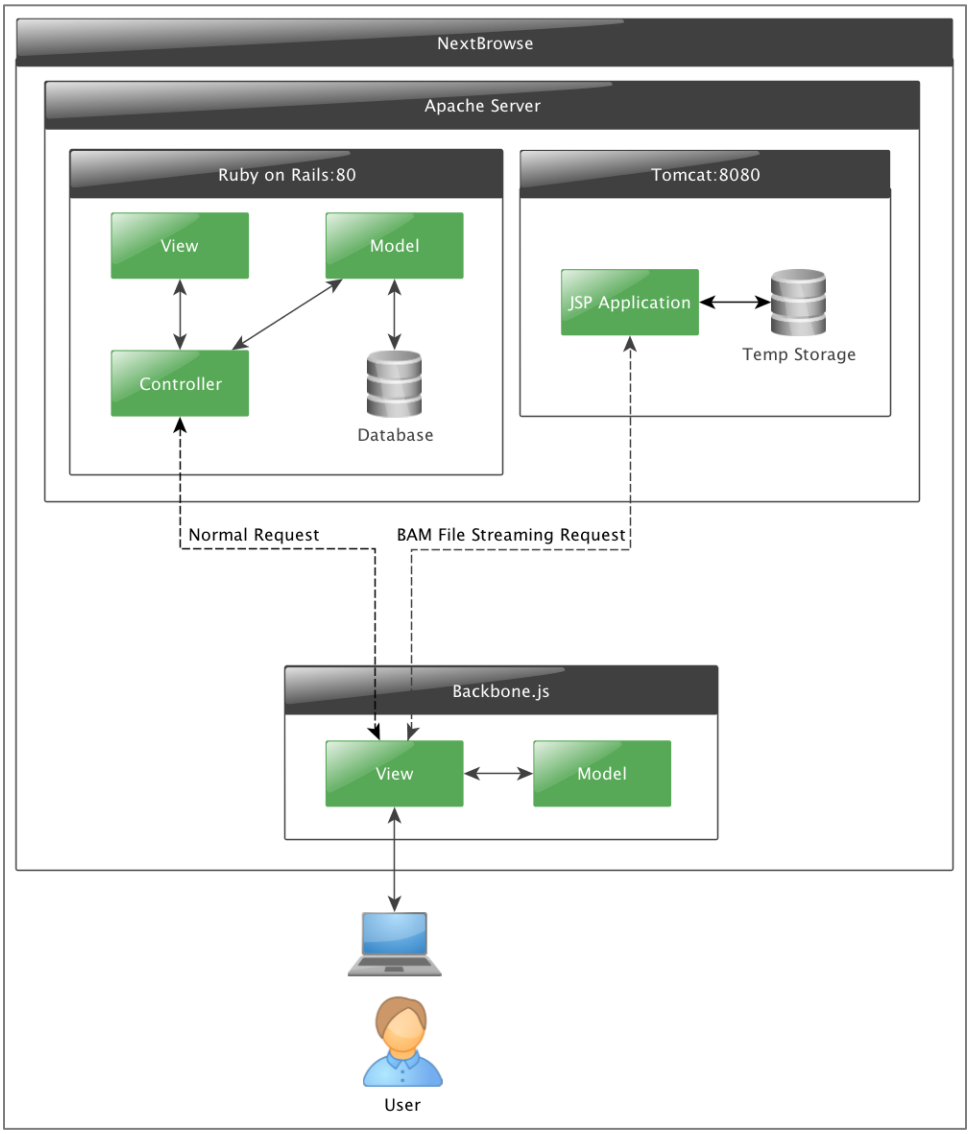


Figure 16 An overview of how the client and server architectures communicate. A user interacts with a Backbone.js view. If the view needs more data, an AJAX request is sent. Either the ROR or JavaServlet server returns data in JSON or text to the Backbone.js view. The view stores the data in the model and the change in the model triggers the view to render the new data.

Chapter 3

Features and Improvements in UI

The following chapter first presents the features of NextBrowse, followed by an examination of the improvements made to GUI elements of current genome browsers. Given that JBrowse is the web-based client side genome browser that most closely resembles NextBrowse in functionality, the following subsections offer a comparison of features and GUI elements between JBrowse and NextBrowse.

3.1 General Features

The following subsections outline the major features found in NextBrowse.

3.1.1 Security and User Accounts

NextBrowse is the first genome browser to offer user accounts and security through the use of an industry standard security system, Devise Ruby Gem. Users are cast into one of two roles, normal or administrator. Users that sign up for NextBrowse are automatically cast into the normal user role. Normal users have the ability to upload, delete, and view GFF files. GFF files can be viewed in raw text form or through visualizations using NextBrowse. Normal users can navigate BAM directories and download or visualize any BAM files that they have been granted access too. Lastly, normal users can visualize their own BAM files.

Administrators are granted the same privileges as normal users, with a few additions. First, administrators can view, delete, and add any GFF files regardless of who uploaded the file. Secondly, they can add and remove BAM directories to user accounts. Thus, it is the sole responsibility of administrators to control BAM directory and BAM file access. Lastly, administrators can create or delete any user account. Given the high responsibility of administrative accounts, it is highly recommended that there is at least one administrator. Details of how to create, add, and delete administrative accounts can be found in installation instructions of NextBrowse at http://www.nextbrowse.vbi.vt.edu. An overview of the privileges granted to normal and administrative users can be found in Table 5.

Abilities	Normal Users	Administrators
Upload GFF Files		
Delete GFF Files		
Download BAM Files		
View BAM Directories		
Visualize BAM Files		
Visualize BAM Directories		
Add and Remove Users		
Add and Remove BAM Directory Access		

Table 5 Comparison of user account capabilities between normal and administrative accounts. A green cell indicates a granted privilege while a red cell indicates a denied privilege.

3.1.2 Sanity Checks and Accessibility

NextBrowse is accessible from any devise with an Internet connection. Therefore, NextBrowse can be accessed using a mobile device, laptop, or desktop computer. Currently NextBrowse does not explicitly change its display settings based on the device accessing it, but it is in the hopes of the authors that this functionality will be added in future work. Additionally, users can attempt to visualize data with any web browser, but it is only guaranteed that Google Chrome web browser will display data correctly, reasons for this are explained in detail in Chapter 2.

In addition to accessibility, NextBrowse employs sanity checks of data on all fronts. Many genome browsers, such as JBrowse, allow users to submit incorrect data to the browser. For example, JBrowse allows users to enter negative regions when searching for a position. This facilitates possible errors and confusion for the user. To avoid situations such as these, NextBrowse applies sanity checks to all forum fields and data whenever possible. For example, if a user enters a negative location in a start or stop location to search a track, NextBrowse will notify the user that the values are incorrect and out of bounds.

3.1.3 File Uploads, Central Repository of Files, and BAM File Access Control

The security measures and user accounts employed by NextBrowse allow users to have a central location for all of their genomic data or GFF files. Users can upload unlimited GFF files for viewing, interpretation, and analysis as long as the GFF files conform to the specification detailed by the Sanger Institute [37]. Providing a central repository of files means that users no longer need to re-upload their genomic data each time they wish to view it.

NextBrowse allows GFF files to be visualized or viewed in plain text format. Each uploaded GFF file is split up into multiple tracks with each track corresponding to a different feature type, e.g. exon. These tracks are then further split into track elements. NextBrowse provides built-in BAM file hosting. Administrators grant users access to directories on the server where BAM files are located. Users have the ability to download and visualize the BAM files within each of their granted directories. This gives scientists an easy and robust way in which to serve BAM files. Clinicians will find this feature extremely useful. After sending in a patient's genome for sequencing, clinicians can simply have the sequencing facility grant the clinician access to the generated BAM file. This removes the time it would take to download or transport the BAM file to the clinician, which can often take a few days. Therefore, there is a decreased time to discovery because clinicians will have immediate access to patient sequencing data.

3.1.4 BAM File Streaming

NextBrowse allows users to visualize any number of streaming BAM files. Users simply have to upload the index file of a BAM file, select their BAM file and then click the Add BAM Track button to visualize the BAM file. The GUI components to add a streaming BAM track can be found in

Figure 25. It should again be noted that the BAM files must be sorted for visualization to properly work.

3.1.5 Lite Browser

In creating NextBrowse the author's realized that many users will not want to create accounts and be forced to sign in. Therefore, NextBrowse features a lite version that does not require users to login. Instead, users simply choose the reference organism they wish to visualize data against and then launch the browser; e.g. Figure 48. However, users are only able to visualize streaming BAM files while using the lite browser.

3.2 Improvements in GUI Elements

NextBrowse has made improvements in many GUI elements currently found in genome browsers as well as created many new GUI elements. The following subsections highlight these improvements in closer detail.

3.2.1 Individual Tracks and Track Types

As previously mentioned, NextBrowse allows users to visualize BAM and GFF files. Therefore, NextBrowse supports two different track types, GFF and BAM. Traditionally tracks within genome browsers have been treated as a single entity and therefore offer a single viewing experience where data can only be viewed one region at a time. More specifically, tracks are not treated as individual viewing areas and each track can only be viewed at one location and one level of depth. This hinders the user's ability to examine multiple areas of interest at once and each track at a different resolution.

NextBrowse treats each track as an individual viewing area. This allows each track to view a different region of interest. Individual tracks can be shuffled vertically within NextBrowse allowing users to manipulate the order in which data is presented. Each track can also be viewed at any resolution, independent of other tracks. This allows users to view multiple locations at once and each location at any level of depth or resolution. Even though each track is treated individually, NextBrowse allows tracks to be searched or snapped to the same location. This feature is extremely useful when comparing multiple GFF or BAM files at once. For example, if a clinician has ten different patient BAM files they can quickly snap all of the BAM files to a single location to examine variation

between the patients. Therefore, NextBrowse still preserves the single viewing experience as found in previous genome browsers.

3.2.2 Location in the Reference

As previously stated, each genome browser provides either a graphical or textual representation, or both, of where the user is located in relation to the genome. Reference location is achieved in JBrowse with the graphical element displayed in Figure 17.

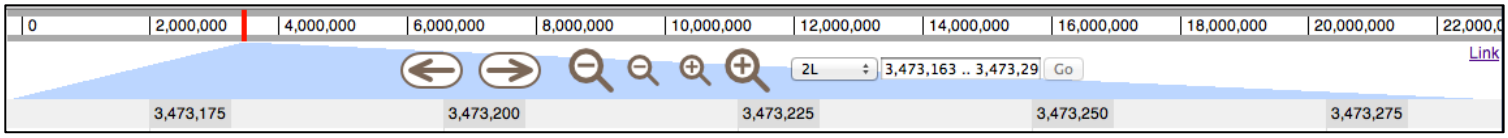


Figure 17 The jBrowse GUI elements for location reference.

The reference location within JBrowse is comprised of two bars. The top bar represents the current chromosome that the user is located on. The bottom bar represents the current nucleotide range of data being viewed within the chromosome. The numbers overlaid on top of the reference location top bar correspond to the nucleotide length of the current chromosome. The top bar reference location has a red bar overlaid on top of it that corresponds to where the user is in relation to the current chromosome the user is viewing. The red bar can be dragged to change the users position within the current chromosome. When dragging the red bar, the user changes the location of every track in the genome browser. The red bar also expands and contracts based on the amount of data being displayed, where the nucleotide viewing range is positively correlated to the size of the bar. The reference location top and bottom bar has a blue graphical element running from the top bar to bottom bar and is used to assist the user in visually identifying where they are located in the current chromosome.

JBrowse's reference location approach is very useful, but it does have a few caveats. First, users are allowed to select negative regions within the left most chromosome of the organism. This can lead to confusion for the user because they may accidently move the slider to a negative region. Secondly, there is no visual or textual queue as to where the user is located within the entire organism, but only within the current chromosome. Additionally, dragging the red overlaid bar of the reference location updates every track within the genome browser. This forces every track within the genome browser

to update, which in turn means that users can only view one area at a time. Lastly, when viewing a very small region the red handle of the reference location sometimes grows to only ~3 pixels wide making it difficult to click and drag using the mouse.

In addition to graphically displaying a reference location, jBrowse also shows the user where they are located in the current chromosome through the textual representation in the search bar. This feature is useful, but too often the range of data being viewing is displayed beyond that of the search text field. This lends itself to users not being fully aware of the amount of data they are viewing.

NextBrowse's reference location is quit similar to jBrowse and can be viewed in Figure 18. NextBrowse's reference location is comprised of a top bar and bottom reference sequence bar. The top bar has overlaid the start, middle, and last nucleotide position of the entire genome. The top bar also has overlaid green oval chromosomal graphics that represent the chromosomes of the organism. In doing so, users are aware of where they are in the entire genome as well as their current chromosome. The reference location bottom reference sequence displays the reference sequence of the current region of interest of the currently selected track. This allows users to quickly examine the reference sequence of the organism at any location.

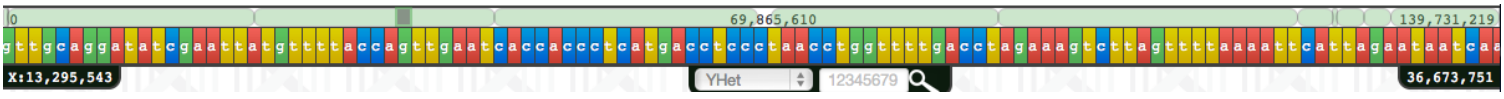


Figure 18 The NextBrowse GUI elements for location reference.

The reference location top bar has a green bar that can be dragged to change where the user is in the entire genome. The dragging of this bar does not change any tracks in the genome browser. Instead, it updates the reference sequence displayed below the top bar location. The reference sequence displayed below can also be dragged horizontally to reveal further reference sequence information. Dragging either the green bar of the reference location top bar or the reference location reference sequence automatically updates the other corresponding reference location element. That is, if a user drags the reference location top green handle, then the reference sequence below will automatically update with the reference sequence data of the new position in real time. The overlaid graphical chromosomes of the reference location top bar can also be hovered to reveal the chromosome

name, chromosome start position, and chromosome end position. This allows users to quickly find chromosomes of interest.

In addition to a graphical representation of the data, NextBrowse displays under the reference location reference sequence the current chromosome and position within the chromosome on both the left and right hand side. This range is determined by the size of the users window and is automatically determined at run time. Therefore, NextBrowse displays both graphically and textually where users are in the genome in relation to chromosome and position within the chromosome location, as well as graphically within the entire genome. The reference location can also be searched by chromosome and chromosomal start position. Since NextBrowse uses constrained zooming, it is unnecessary to grow the handle of the of the reference location top bar. Instead, NextBrowse uses a fixed sized handle. Using a fixed sized handle makes it easy for users to use because it does not grow too small to manipulate.

As already stated, changing the top reference location of NextBrowse does not change the location of tracks displayed in NextBrowse. Instead, the top reference location automatically changes locations to match that of the currently selected track. That is, whatever track the user is currently manipulating or viewing, the top track will updates its position to match that of the currently selected track. The top reference location also updates in real time as users move throughout the currently selected track. This allows users to rapidly view different areas of the reference sequence for different tracks and NextBrowse is the first genome browser to achieve this functionality. This feature is particularly useful when viewing BAM tracks because users can load multiple BAM tracks and view the tracks at different locations for comparison.

3.2.3 Searching

As previously mentioned, NextBrowse makes use of individual tracks. Therefore, track can be individually searched by chromosome and chromosomal region. This allows users to view multiple locations of data at once, a feature not previously found in genome browsers. In jBrowse, users search all tracks using the drop down chromosome list and text field as displayed in Figure 17. Having users enter the chromosome, stop location, and start location of viewing in one textfield increases the likelihood that users will make a mistake when searching. Instead of using a single text field, NextBrowse splits up the search text field into two distinct text fields; the start and stop

location. This reduces the likelihood of users entering incorrect locations to search. In addition to searching the currently selected track, tracks can all be searched all at once. The multiple magnifying glass icon symbolizes this functionality. This feature allows users to view multiple tracks at the same location. Figure 19 displays the search GUI element found in NextBrowse. Since NextBrowse uses constrained depth searching, when a user searches for a region using the start and stop location a red bar is drawn to indicate the start and stop location.

Figure 19 The search user interface elements of NextBrowse highlighted within the green box.

3.2.4 Location within each Track

Since NextBrowse allows each track to be searched individually, it is imperative to show the user where they are located within each track. This is achieved by displaying the location of where the user is in the bottom left and bottom right of each track. The right hand position corresponds to the far right side of viewing data. The location is represented by displaying the chromosome and the position within the chromosome of where the user is. These textural queues can be viewed in Figure 20

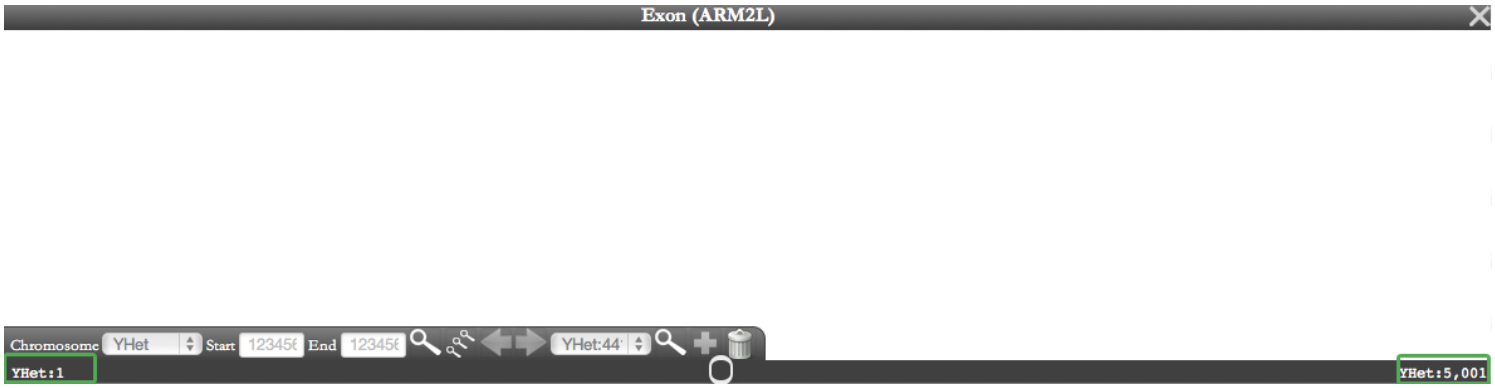


Figure 20 The location of where the user is located is highlighted within the green box at the bottom of the track.

3.2.5 Navigation

Navigation within NextBrowse consists of zooming and horizontal panning with a track. That is, a track can be zoomed in to reveal further details or zoomed out to reveal less detail.

3.2.5.1 Horizontal Panning

Traditionally in genome browsers horizontal panning is done through either a GUI element or through clicking and dragging a track. Horizontal panning is characterized as moving a track either left or right to reveal further information. JBrowse allows users to click and drag a track or use the left and right arrow GUI buttons reveal further information. NextBrowse allows each track to be horizontally panned using a GUI slider in the middle of each track. The handle of the slider can be dragged left or right, with each direction revealing information in either direction of the track. Behind the handle of the slider are horizontal arrows to further indicate to the user the use of the slider. The horizontal GUI slider can be viewed in Figure 21.

Figure 21 The horizontal GUI slider found on each track.

3.2.5.2 Zooming with Constrained Zooming

In traditional genome browsers, users are forced to find the correct level of depth at which to view their genomic data. This adds an extra task that users must perform to find their data. Even though this does add the ability for users to view their data at multiple and often any resolution, it still adds an extra task to the user. NextBrowse explores using constrained zooming.

Constrained zooming only allows users to view data at certain intervals or resolutions. NextBrowse allows genomic data to be viewed at the genome, chromosomal, gene, or reference level. The genome level displays one element that represents the entire genome. This element contains additional information about the genome. The chromosomal level displays every chromosome of the organism. The genome level displays data in a 5,000 nucleotide range. The reference level displays data at the nucleotide range. At the reference level BAM tracks display the individual nucleotides of reads and GFF tracks display the actual reference sequence of the organism. The range of nucleotide data displayed at the reference level is dynamically determined at run time based on width of the users screen.

JBrowse allows users to zoom in using four different buttons, which are displayed in Figure 17 as the magnifying glasses. The larger icons zoom in further than the smaller icons. The levels of zoom

can be used in any order. Users also have the option to enter their own level of zoom by entering a range of data to view or search in the search box. This allows users to view data at any depth. However, this feature forces the user to find the correct level of zoom at which to view their data. Often many resolutions reveal no data at all, giving the user the impression that there is no data available for viewing when in fact that is, but at a higher resolution.

Instead of buttons, NextBrowse uses a single slider GUI element for all tracks. The slider dynamically moves to the currently selected track. The slider has four levels of zoom with each level illustrated by an icon. Each icon within the slider symbolizes the level of depth or zoom a track is being viewed at. An overview of the icons can be viewed in Figure 22. The very bottom icon, the genome icon, represents the genome level of zoom where users view the entire organism. The icon above the genome icon, the chromosomes icon, represents the chromosome level of zoom where users view all of the chromosomes of the organism. The icon above the chromosomes icon, the gene icon, symbolizes the gene level zoom where features can be viewed in a GFF file and individual reads in a BAM file. The ATGC icon is used to represent the reference level and represents when users can view the actual nucleotide sequence of the data they are viewing.

Constrained depth zooming does not offer users the ability to view data at any depth. However, it provides users with less options of zooming. This in turn alleviates the time users spend searching for the correct resolution at which to view their data and improves the speed at which users can find a suitable level of zoom for their data.

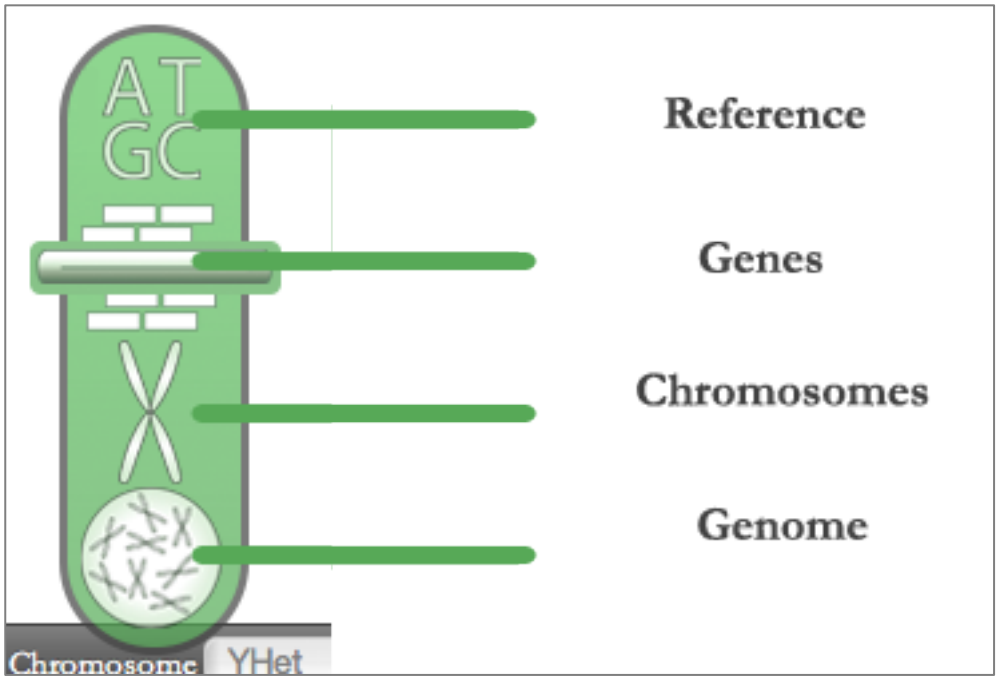


Figure 22 Different slider icons present in NextBrowse. The top slider displays the icons of GFF tracks and the bottom slider displays the icons of the BAM track.

The slider's opacity changes throughout the application. When there are no tracks loaded into the browser, the sliders opacity is extremely low and the slider is disabled. When a track is loaded into NextBrowse the sliders opacity is raised slightly. The opacity of the slider is toggled between when the user hovers over the slider. That is, when the user hovers over their mouse over the slider the opacity is raised slightly and when the user removes their mouse the opacity is lowered. Having opacity be lowest when there are no tracks helps the user distinguish when the slider is enabled and disabled. Additionally, raising the sliders opacity when hovered gives a visual queue to the user that they are currently hovering over the slider.

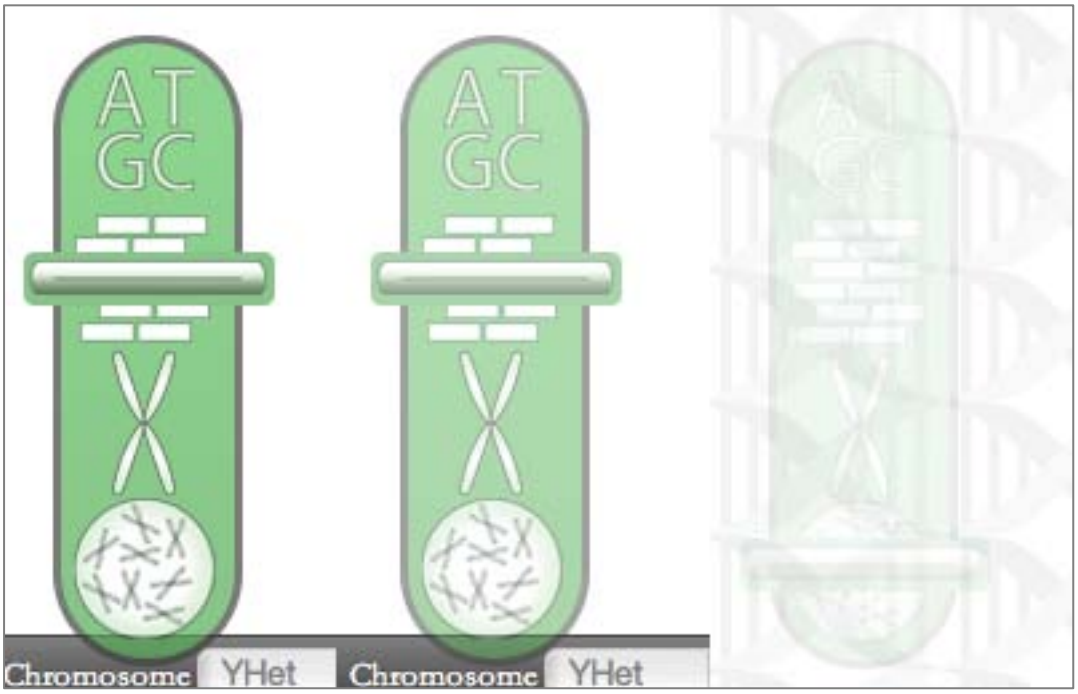


Figure 23 NextBrowse GUI slider (left) the slider's opacity with a mouse hover, (center) the slider's opacity with the slider enabled but without mouse hover, (right) the slider's opacity with the slider disabled.

3.2.6 Track Selection

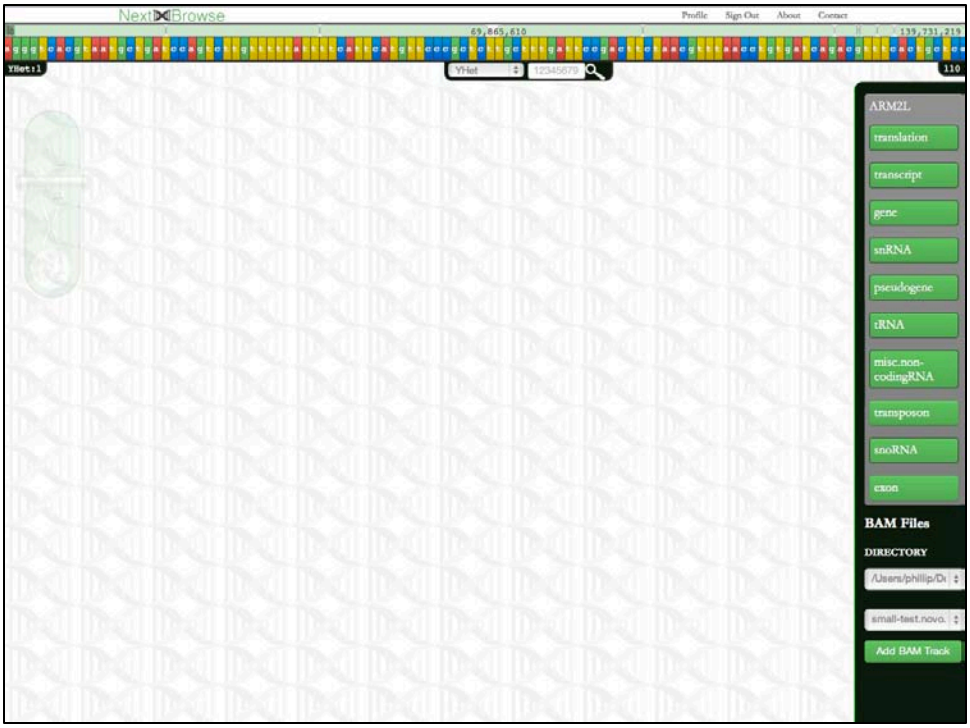
NextBrowse allows users to add tracks from the right hand panel of the browser. When the user hovers their mouse over the right panel the panel expands to reveal all possible tracks for visualization. When the user removes their mouse from the panel the panel moves back to its original location; e.g. Figure 24.

NextBrowse originally adopted the way in which tracks are added in JBrowse, by dragging and dropping tracks. This functionality was experimented with GFF files. GFF files were contained in a gray box with their respective tracks as green buttons that could be dragged within the gray box. Each GFF track button could be dragged onto the middle pane. Upon mouse release of dragging the green button one of two actions would happen. If the user releases the button over the middle panel then a new track would be created based on the track data contained in the button. If the button is not released over the middle track then the button was snapped back to it's original location on the right panel. This functionality of buttons can be viewed in Figure 24.

At first glance this functionality seems extremely interactive and useful, but once the user has a large number of tracks the track listing goes off of the screen. If a user has one hundred available tracks it

would be extremely time consuming to select a track. Additionally, this functionality only allows users to add one instance of a track at a time. Therefore, instead of having users drag and drop buttons that correspond to tracks, users select a track to add from a drop down menu. More precisely, they select the GFF file they wish to view and then a GFF track from the selected GFF file. This allows users to have many tracks without worry of the tracks disappearing off the screen. Furthermore, this allows users to add multiple instances of the same track.

BAM tracks are added in much the same way GFF tracks are. Users select a BAM directory that they wish to view BAM files for. Next users select the BAM file that they wish to visualize. The browser automatically loads the first instance of BAM files for the first BAM directory. Users can click the Add BAM Track button to add the selected BAM file. If a user changes the BAM directory, then the BAM file list will automatically update to the reveal the respective BAM files within that directory. Client side BAM file tracks are adding by the user first selecting the index file that corresponds to their BAM file. Once the user uploads the BAM file index they then selects the BAM file and click the Add BAM track button. Figure 25 displays the final adding track functionality found in NextBrowse.



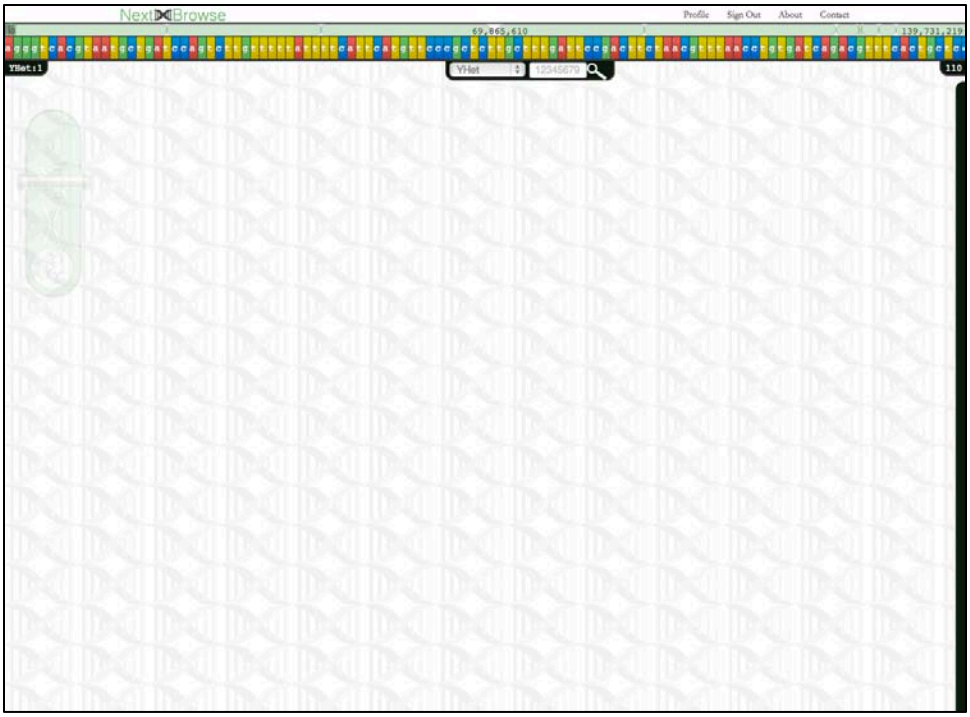


Figure 24 NextBrowse displaying the right hand pane hidden and displayed. The top image displays the right hand pane visible and the bottom pane displays it hidden.

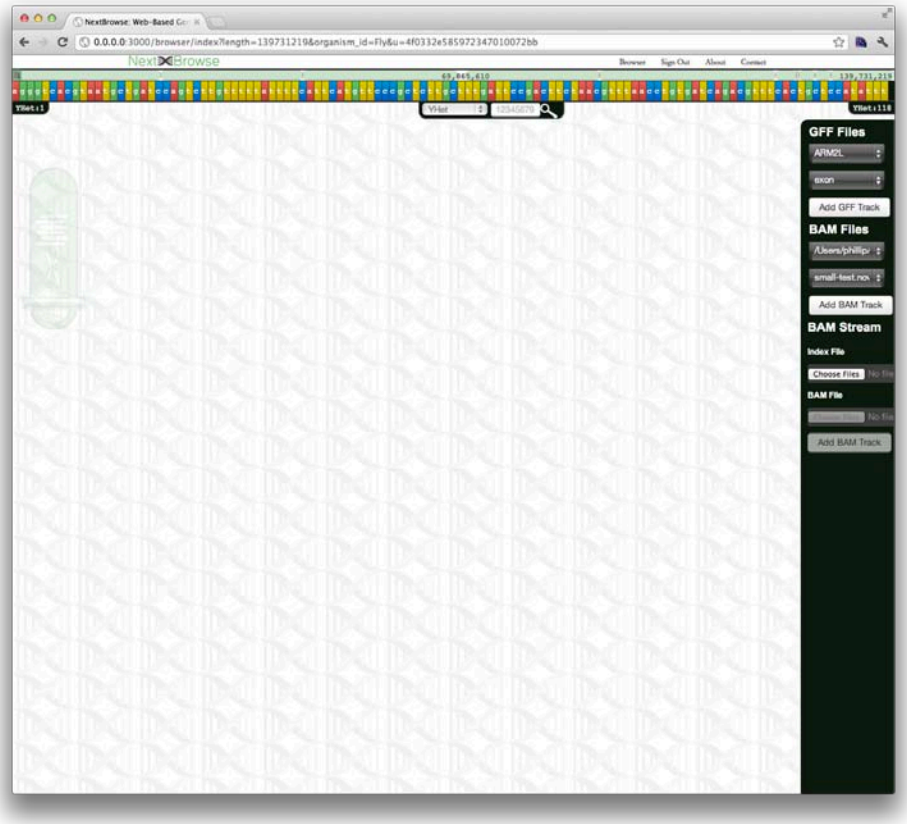


Figure 25 NextBrowse right hand panel containing all of the possible tracks to add.

3.2.7 Closing a Track

Clicking the close button in the top right corner of each track closes tracks in NextBrowse. When the close button is clicked the user is displayed with a validation message confirming that they do in fact wish to close the track. The validation is achieved through using the jQuery Apprise plugin [34]. The plugin dims every part of the application except for the close confirmation box. This forces the user to focus on the confirmation box and make a decision. An example screen shot of the confirmation close box can be viewed in Figure 26.

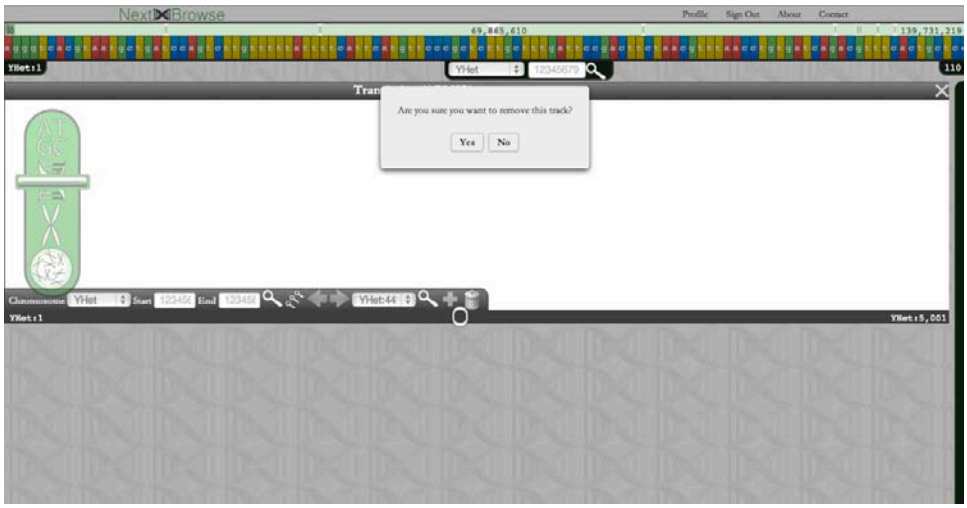


Figure 26 The confirmation box being displayed to ask the user if they do in fact wish to close the selected track.

3.2.8 Bookmarking

NextBrowse allows users to bookmark specific organism locations by chromosome and chromosome position. Figure 27 displays the bookmarking GUI elements, contained within the green box, found on tracks in NextBrowse. Table 6 displays a listing of each individual bookmarking GUI element with the icon for the element and a description of elements use.

Name of Element	Icon	Use
Left Arrow		Used to move the previous bookmark location.
Right Arrow		Used to move to the next bookmark location.
Select Menu	n/a	Used to display the current listing of bookmarks.
Magnifying Glass		Used to search to the currently selected bookmark.
Add Bookmark		Used to add the current location of the track to the user's bookmark list.
Delete Bookmark		Used to delete the currently selected bookmark.

Table 6 Description of all bookmarking icons along with their respective icon.

Figure 27 The bookmark user interface elements of NextBrowse highlighted within the green box.

3.2.9 Curate Data

Genome browsers often display or link to additional information about an element in a horizontal track. Very often the user is unaware of were they are being redirected or the information that is being displayed is redundant. As Figure 4 displays, JBrowse simply displays a pop-up window indicating to the user that they have clicked on a feature. Other elements in NextBrowse send the user to a third party website for additional information such as the fly database when viewing tracks that pertain to *Drosophila Melanogaster*. NextBrowse attempts to curate data in the track itself. As discussed in the next chapter, when a GFF element is clicked the element rotates to reveal further information about it. More specifically, each GFF element in NextBrowse displays the information that NextBrowse contains about the element as well as the top four links to information from NCBIs PubMed and Gene database. While this is still a redirection, it is still an improvement over current models because users are aware of were they will be taken as opposed to being blindly redirected. This functionality is viewable in Figure 28. The author would have liked to pull information from the top PubMed and Gene articles and displayed it in the track itself, but due to NCBIs strict access policies this was not allowed.

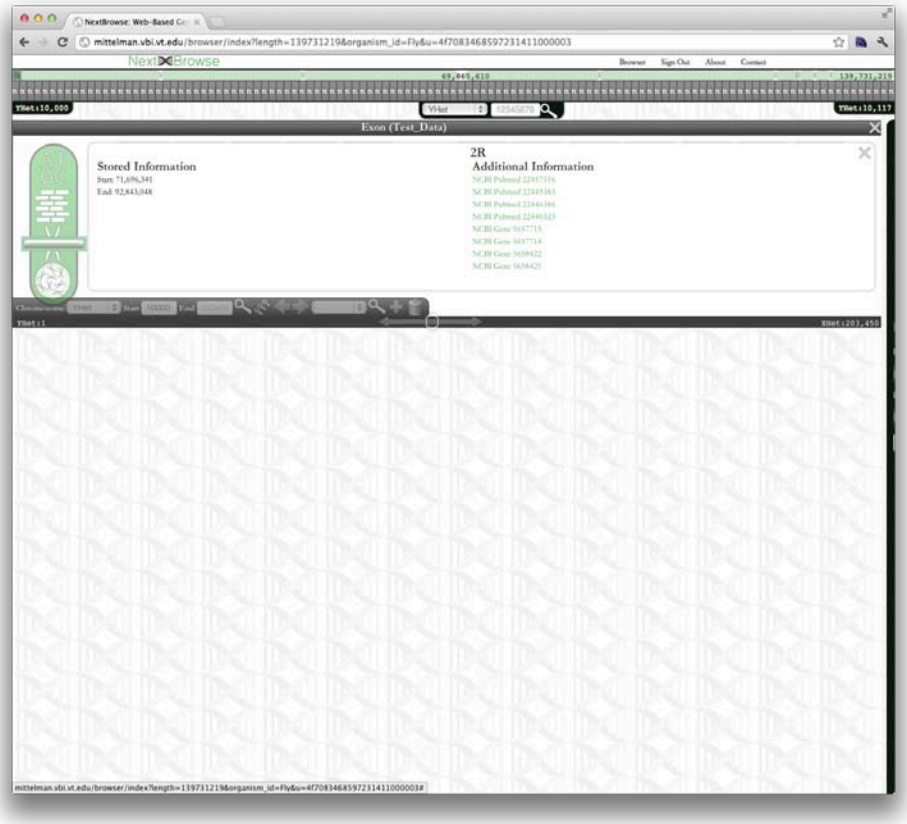


Figure 28 Chromosome 2R clicked to reveal further information from the PubMed and Gene databases.

Chapter 4

Genomic Data Visualization Techniques

The following chapter provides an examination of the different visualization techniques used by NextBrowse to visualize genomic data. Given that NextBrowse allows two different types of tracks, BAM and GFF, to be visualized, the following subsections are broken down to reflect these track types.

4.1 GFF Element Shape Visualization

Given that NextBrowse dynamically creates tracks from GFF files, each GFF track uses the same visualization technique. In future work it is in the hopes of the author to update this functionality to where different track types reflect different visualization techniques. NextBrowse treats every GFF element feature as a separate element on the screen. Each element displays its start and stop position along with its name. When elements are hovered they come forwards, resembling the cover flow effect found in Apple's iTunes Album artwork traversing, e.g. Figure 29 and Figure 30. This subtle use of 3D movement assists users in element selection and improves interaction.

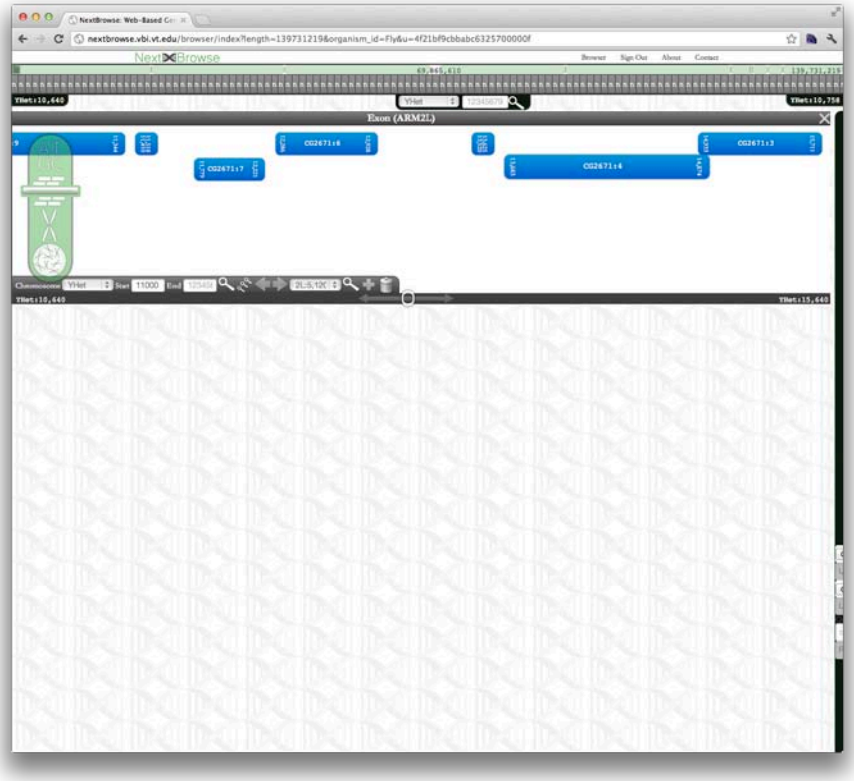


Figure 29 NextBrowse GFF element being hovered. Notice the top second from the right element.

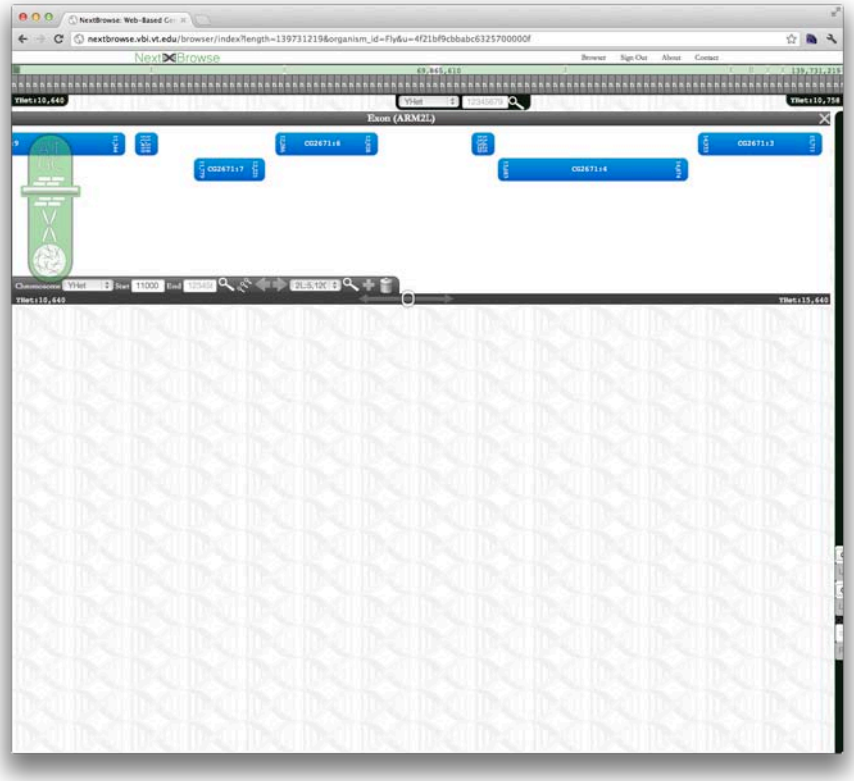


Figure 30 NextBrowse GFF elements not being hovered.

User can click a GFF element to reveal further information about the element. When a GFF element is clicked the element rotates 180 degrees and expands to display further information; e.g. Figure 31. The reference level of viewing of GFF tracks simply displays the reference sequence data of the organism, e.g. Figure 32. This is put in place so that incase a user has multiple BAM files loaded they could have a reference sequence of data that they could quickly and easily move around to help compare the BAM files to the reference.

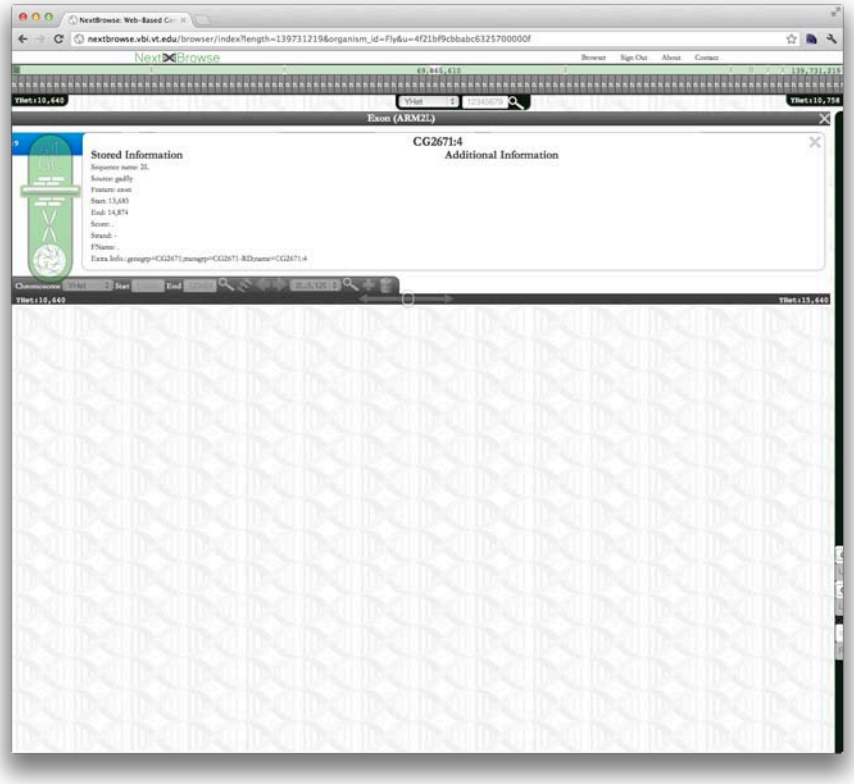


Figure 31 NextBrowse displaying a GFF element rotated to reveal further information.

Figure 32 NextBrowse GFF reference level visualization.

4.2 BAM File Visualization

Currently only a small number of genome browsers support BAM file visualization. The only web-based genome browser to support BAM file visualization is the UCSC Genome Browser. Therefore, the following section will compare BAM File visualization techniques between UCSC and NextBrowse. NextBrowse makes use of as much read data as possible for visualization. As previously mentioned, BAM file reads are represented in two different levels, the 'gene' and 'reference' level. At the 'gene' level reads are viewed in a 5,000 nucleotide range, with each read being represented as a colored rectangle. A rectangular shape was chosen because of each BAM reads distinct start and stop position. Each track has enough vertical room to display nine horizontal reads. Although some areas may in fact have more than nine reads, it was chosen to only display nine read coverage for many reasons. First, it is extremely difficult for a user to decipher 100+ reads a time. To illustrate this idea, Figure 33 and Figure 34 shows UCSC Genome Browser which displays all possible reads at a given location, overloading the user with visual information and data. Secondly, due to the computational complexity of rendering each track, nine reads was chosen as the maximum coverage displayed.

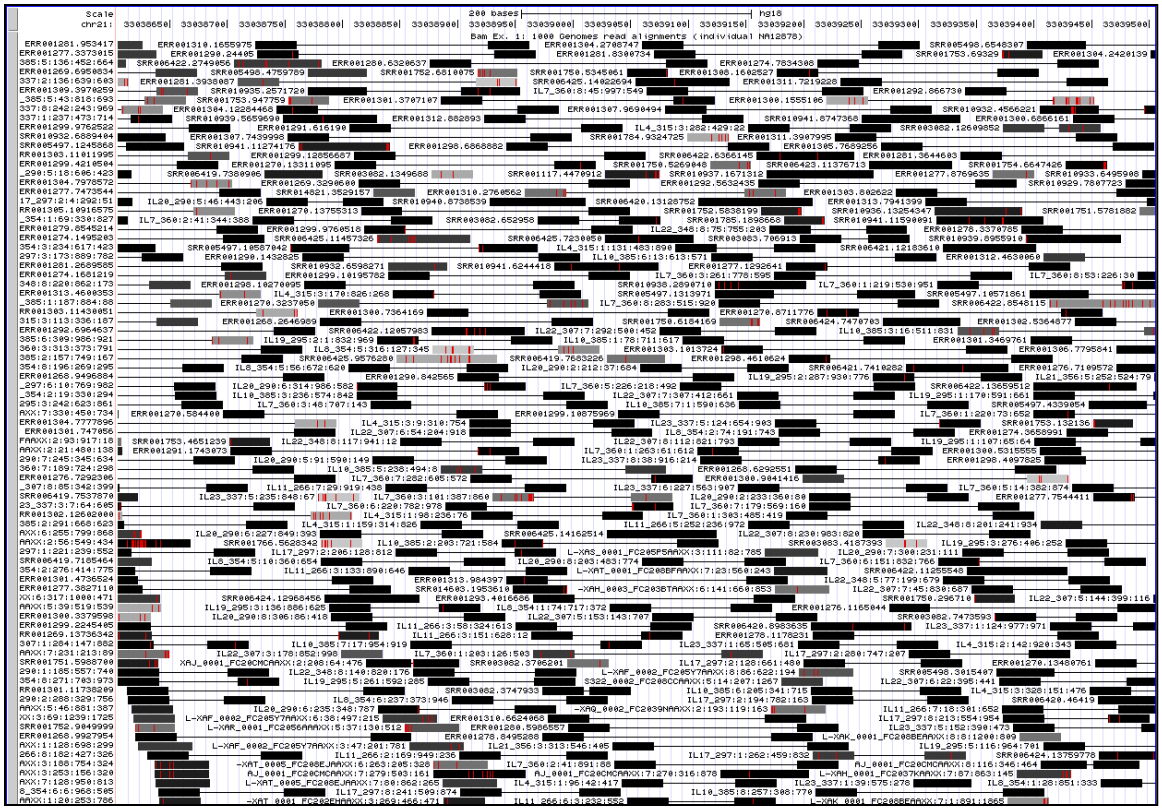


Figure 33 UCSC Genome Browser visualizing BAM file reads.

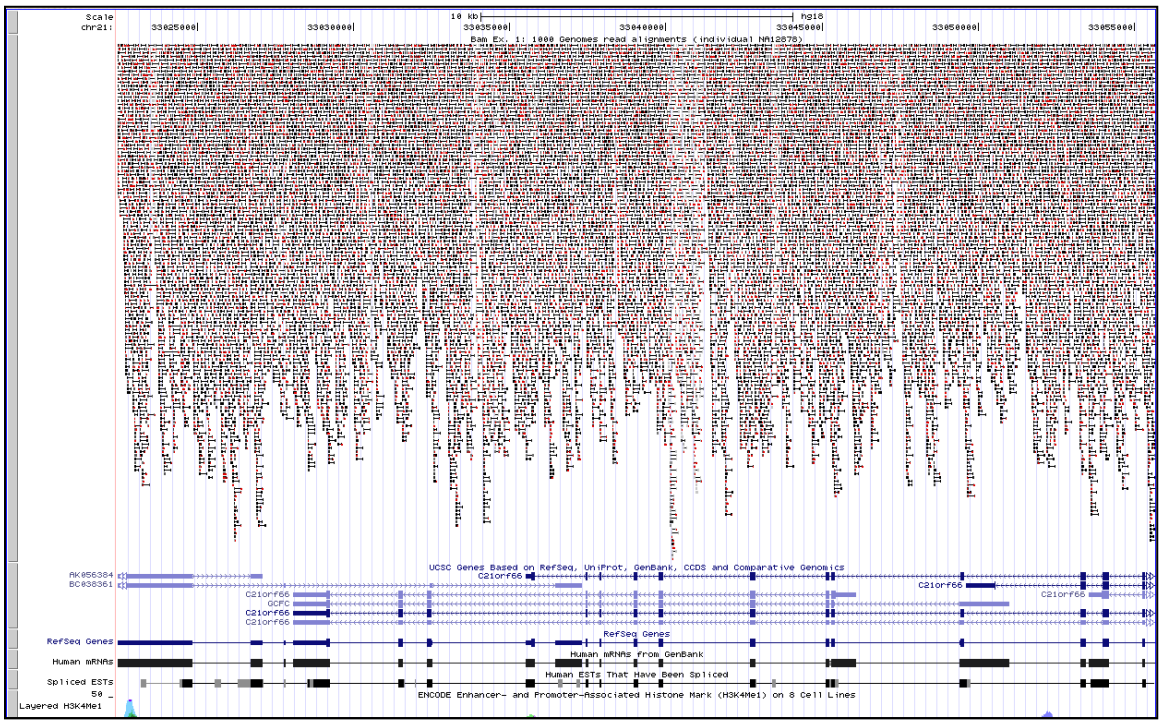


Figure 34 UCSC Genome Browser visualizing BAM file reads zoomed all the way out.

As [38] states, users should not trust aligned reads with many errors. Therefore, it is imperative to visually convey error information to the user, which can be done using mapping quality and phred quality. Mapping quality of a BAM read corresponds to the probability that a read is mapped correctly. Mapping quality of each read in UCSC Genome Browser is visualized through the use of gray scale or saturation, where a read with good quality is completely black and a read with poor quality is white. Instead of gray scale or saturation, NextBrowse maps mapping quality to the opacity of each read, where poorly mapped reads correlate to a low opacity and well mapped reads to a high opacity. Mapping quality is binned to 0.1-1.0 so that even extremely poor quality reads will still be visible to the user. This functionality of opacity is demonstrated in Figure 36. It is imperative to communicate to the user the mapping quality of a read because users do not want to take into high consideration poorly mapped reads.

Figure 35 NextBrowse BAM file visualization highlighting a read being hovered.

Further information about a read can be viewed in UCSC Genome Browser by clicking on an individual read. Once a read is clicked the user is redirected to another webpage for information about the read. An example UCSC Genome Browser BAM read information page can be viewed in Figure 37. If the user wishes to return to the browser, they must go back and the webpage must re-render the BAM reads. This type of redirection slows the user's ability to view data and adds time to the user's viewing experience. Therefore, instead of redirecting a user to another webpage, NextBrowse allows reads at the 'gene' level to be hovered to reveal all of the information of the read, such as the CIGAR string; e.g. Figure 35. Table 7 displays a listing of all information along

with a description of each information item that is displayed to the user when a BAM read is hovered. This allows users to quickly examine reads without leaving the webpage.

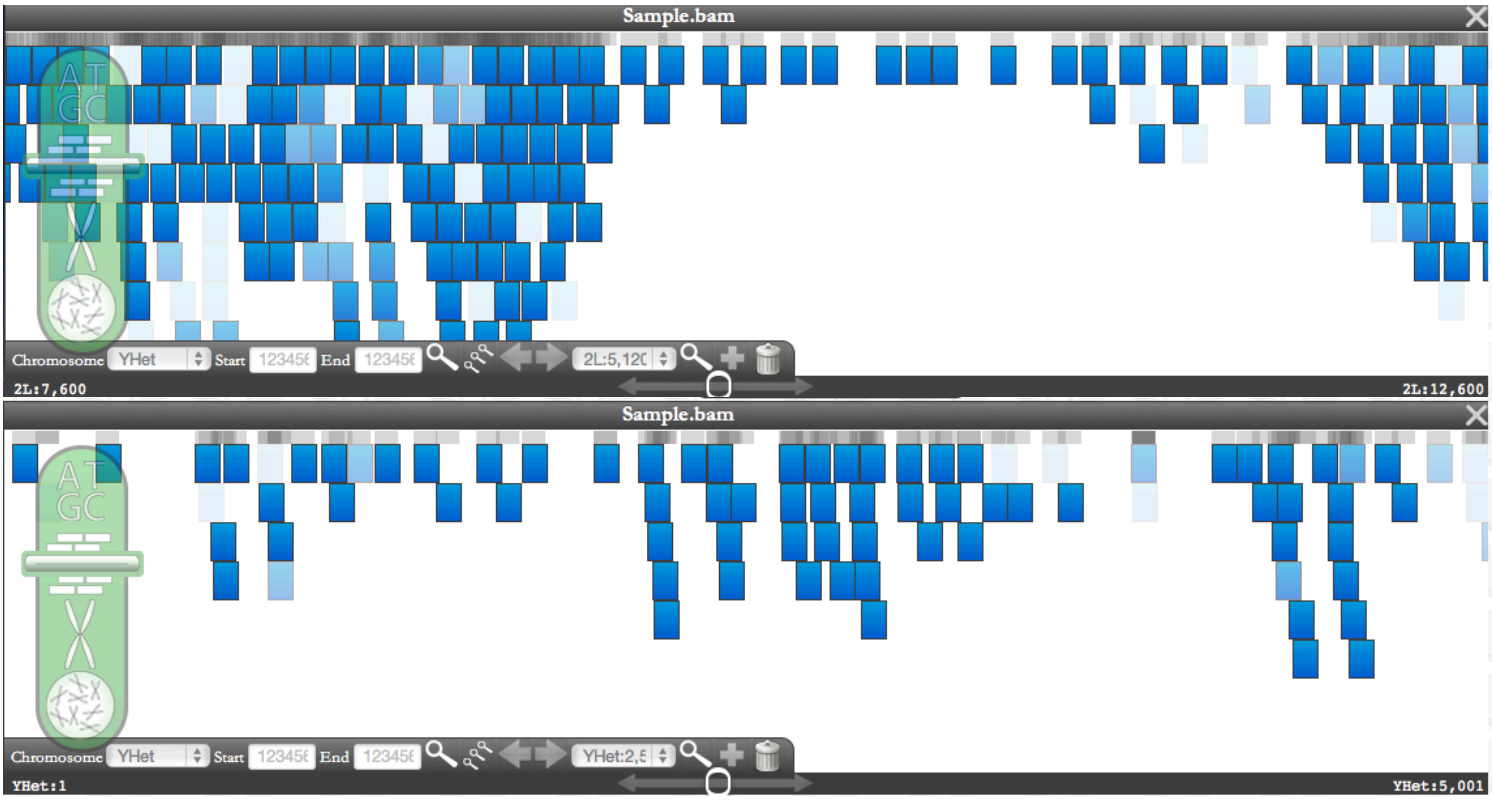
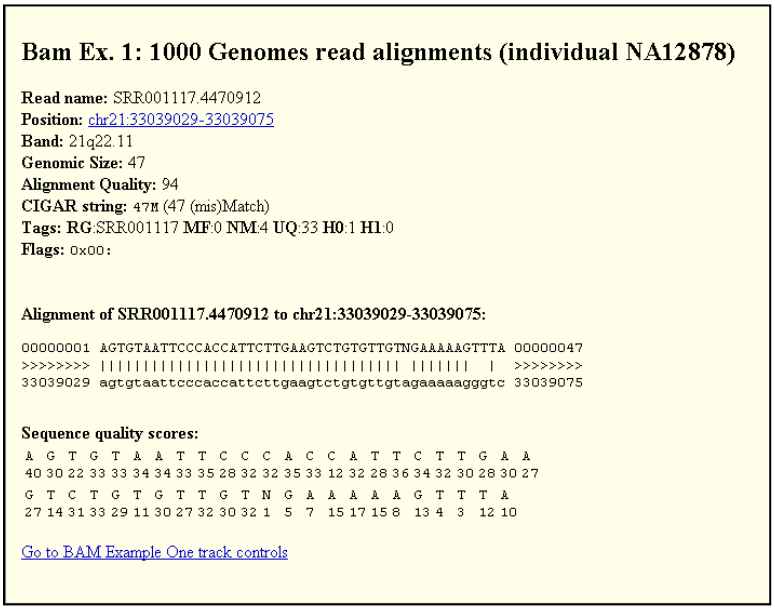


Figure 36 NextBrowse visualization of BAM reads at the 'gene' level. Opacity of reads corresponding to mapping quality.

Attribute	Description
id	An individual id of the read.
start	The start position of the read within its chromosome.
end	The end position of the read within its chromosome.
CIGAR	String representing how a read maps to the reference.
flag	Bitwise flag.
mapping quality	Numerical value corresponding to the probability that a read is mapped correctly.
rname	Reference sequence name of the alignment.

sequence	The sequence data of the read.

Table 7 Listing of BAM File read attributes.



Bam Ex. 1: 1000 Genomes read alignments (individual NA12878)

Read name: SRR001117.4470912
Position: chr21:33039029-33039075
Band: 21q22.11
Genomic Size: 47
Alignment Quality: 94
CIGAR string: 47M (47 (mis)Match)
Tags: RG:SRR001117 MF:0 NM:4 UQ:33 H0:1 H1:0
Flags: 0x00:

Alignment of SRR001117.4470912 to chr21:33039029-33039075:

```
00000001 AGTGTAATTCCCACCATTCTTGAAGTCTGTGTTGTNGAAAAAGTTTA 00000047
>>>>>>>> |||||||||||||||||||||||||||||||||||| |||||||  |  >>>>>>>>
33039029 agtgtaattcccaccattcttgaagtctgtgttgtagaaaaagggtc 33039075
```

Sequence quality scores:

```
A G T G T A A T T C C C A C C A T T C T T G A A
40 30 22 33 33 34 34 33 35 28 32 32 35 33 12 32 28 36 34 32 30 28 30 27
G T C T G T G T T G T N G A A A A A G T T T A
27 14 31 33 29 11 30 27 32 30 32 1  5  7  15 17 15 8  13 4  3  12 10
```

Go to BAM Example One track controls

Figure 37 UCSC Genome Browser BAM file read information page.

The coverage of each BAM track is displayed at the top of the track in the 'gene' level of viewing as opaque gray stacked bars. Therefore, areas of high coverage correlate to high opacity and areas of low coverage to low opacity. The coverage area of a BAM track in the 'gene' level can be viewed in Figure 38. Coverage is represented in UCSC as a histogram plot, where the y-axis of the bar correlates to the coverage of that area. Using opacity instead of a histogram plot allows users to quickly examine what areas of the BAM file have high and low coverage by examining the x-axis instead of the y and x-axis.

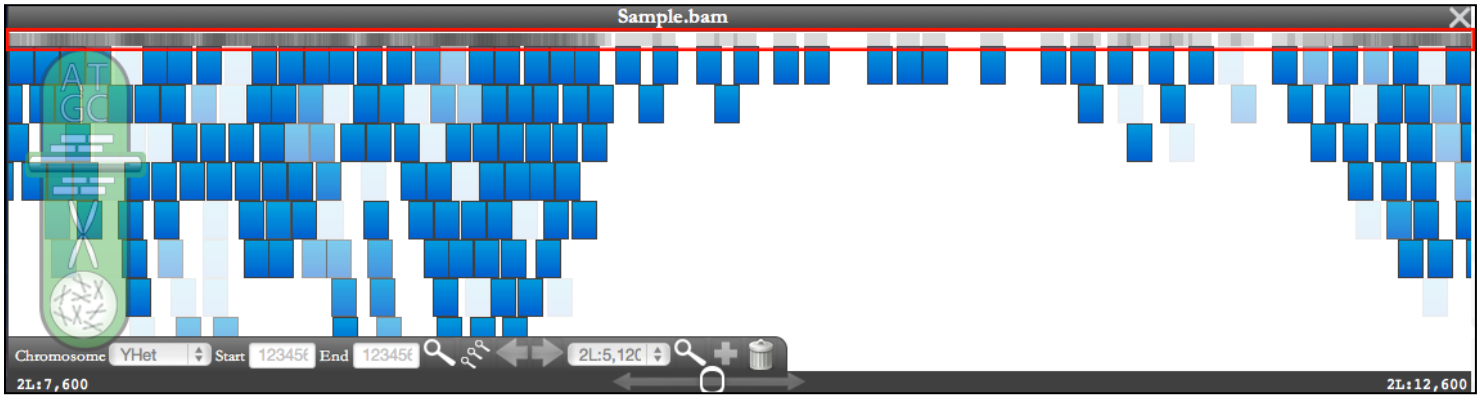


Figure 38 BAM file track at the 'gene' level with the coverage area enclosed in a red rectangle.

When users view a track in the 'reference' level they view each read at the individual nucleotide level. UCSC Genome Browser 'reference' level view can be viewed in Figure 42. UCSC only displays the nucleotides that are mismatched to the reference. This makes it easy to decipher what nucleotides bases are possible mutations. However, without having a vertical queue for the eye to follow, it makes it difficult to decipher where each nucleotide base matches to the top reference sequence.

The 'reference' level view in NextBrowse is quite different from that of UCSC, as displayed in Figure 40. As with UCSC Genome Browser, NextBrowse makes use of mapping quality of each read at the 'reference' level. However, the opacity of the mapping quality is binned to 0.4-1.0 so that the individual nucleotides can still be visible to the user. Additionally, the 'reference' level view in NextBrowse makes use of the Phred Quality scoring of each read. Phred Quality score was first introduced in [39] and is the probability that a base call is correctly mapped. In other words, how certain are we that a specific nucleotide base is correct at its given position. Each BAM read has a Phred quality score associated with each nucleotide base in the sequence of the read. The Phred quality score of each nucleotide is mapped to the coloring of each nucleotide, ranging from gray to white, where gray equates to a poor Phred quality and white to a good Phred Quality. Figure 39 displays an example of this coloring where the beginning of the top right read has poor Phred quality. This gives a visual queue to the user that they should probably not give much consideration to these nucleotide bases.

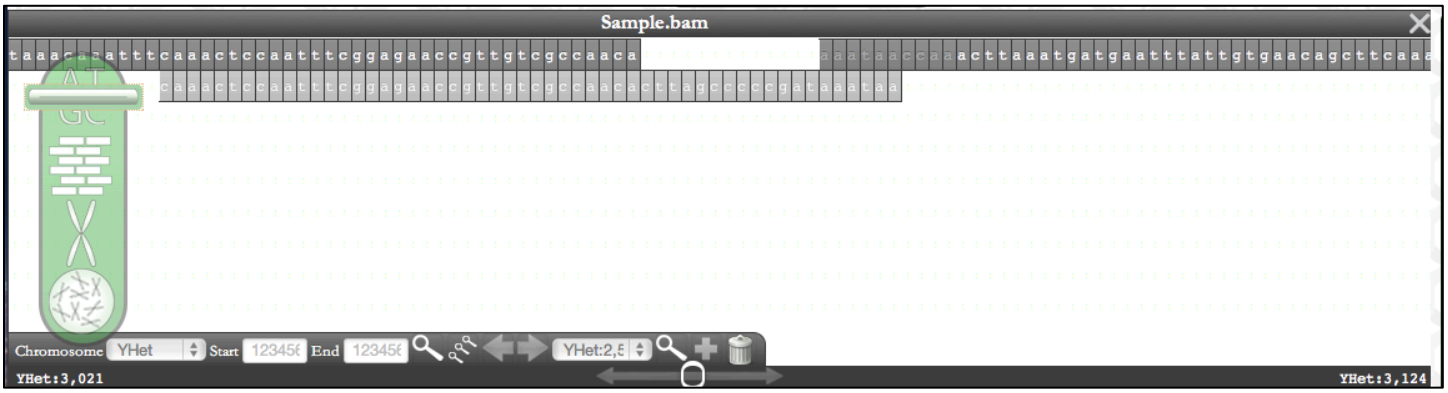


Figure 39 Example of mapping quality and Phred quality found at the 'reference' level. The bottom read has a lower mapping quality then the two top reads. The beginning of the read at the top right has poor phred quality.

When examining BAM reads, users are only interested in viewing reads that do not agree to the reference. This is because reads that do not match may be due to mutation or variation. In the UCSC Genome Browser, these mutations or disagreements are the only bases drawn on the read. However, this does not give the user a context as to where the nucleotide mismatch is in relation to the nucleotide bases. This equates to the user spending more time figuring out where the mismatch read occurs in the reference sequence. Instead of only drawing the mutations or mismatches, NextBrowse draws all of the reads in gray and the mutations and mismatches in color. This makes it extremely fast and easy for users to quickly distinguish possible mutations and single nucleotide polymorphisms (SNP). NextBrowse also colors the mismatched reads with the color of the read that is used in the reference sequence at the top reference location. Additionally, once a mismatch between the reference and a nucleotide is detected, NextBrowse colors all other nucleotides in the vertical position of the mismatched nucleotide. This gives the user a visual queue for their eye to follow to the top reference and it makes it easier to decipher if the mismatch is a variation. Furthermore, since the top reference sequence snaps to location of the currently selected track, users can quickly determine what nucleotide base should be expected at a region. This functionality is demonstrated in Figure 40.

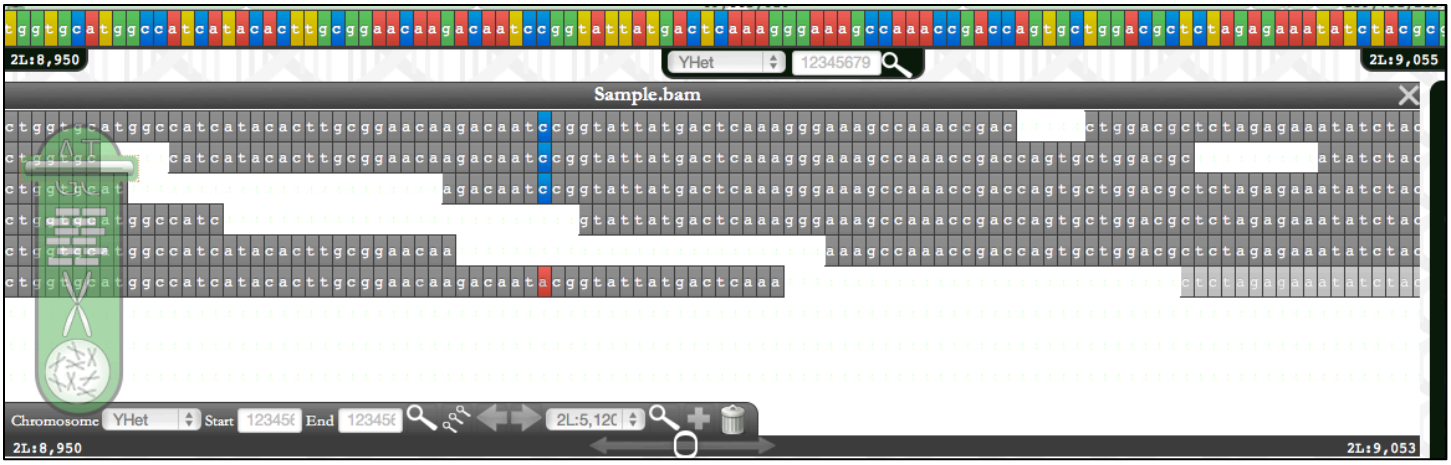


Figure 40 Example of having only nucleotide base disagreements colored. Through quickly examining the top reference sequence it is easy to tell that the 'c' reads are correct.

NextBrowse makes use of many visualization techniques. Mapping quality, Phred quality, and coverage are all taken into account in visualization. Additionally, only nucleotide bases that have mismatches between reads are colored and this coloring matches that of the reference sequence in

the reference location. All of the visualization techniques present in NextBrowse make it extremely easy for users to visualize their data fast and efficiently. An overview of all of the visualization techniques used in NextBrowse for BAM files can be found in Figure 41.

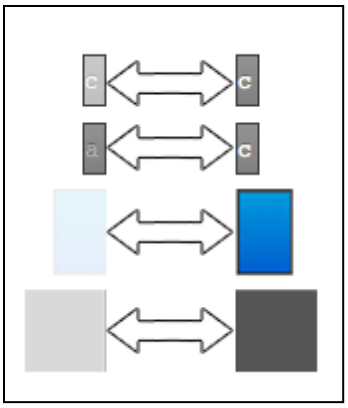


Figure 41 Listing of some of the visualization techniques used by NextBrowse to visualize BAM Files. Top shows Mapping quality of individual nucleotides, second from top displays Phred quality of individual nucletodies, third from bottom displays mapping quality of reads at the 'gene' leve, and the bottom displays the difference in coverage quality coloring.

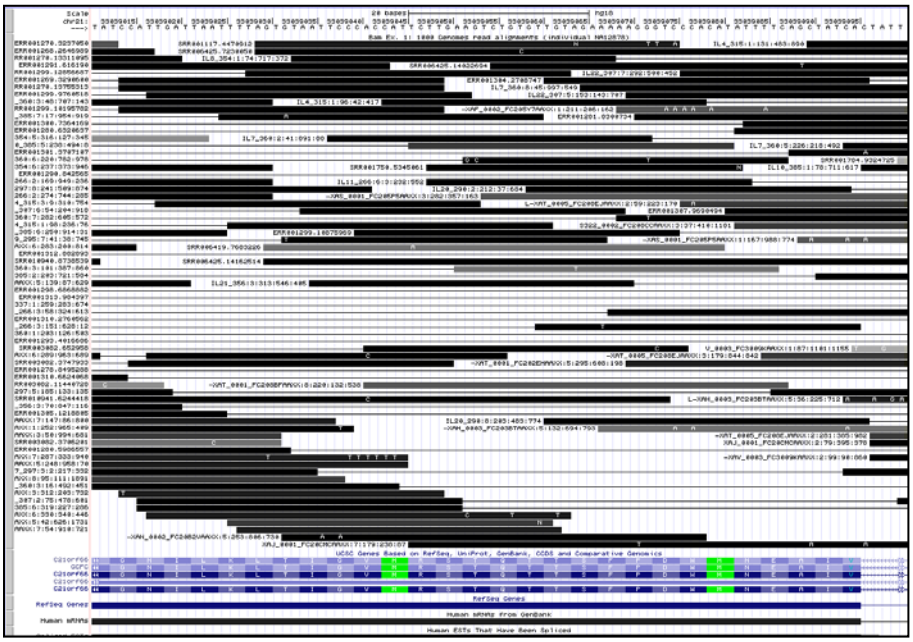


Figure 42 UCSC Genome Browser 'reference' level of browsing.

4.3 Chromosome and Genome Visualization

BAM and GFF tracks both visualize the 'chromosome' and 'genome' level of resolution in the same manner. Chromosomes are displayed in much the same way as GFF elements. Each chromosome

element displays its start and stop location along with its name, e.g. Figure 43. Each chromosome can be hovered and clicked to reveal further information. The genome is displayed as a single element in NextBrowse in the same manner as chromosomes, e.g. Figure 44. The genome can also be hovered and clicked to reveal further information.

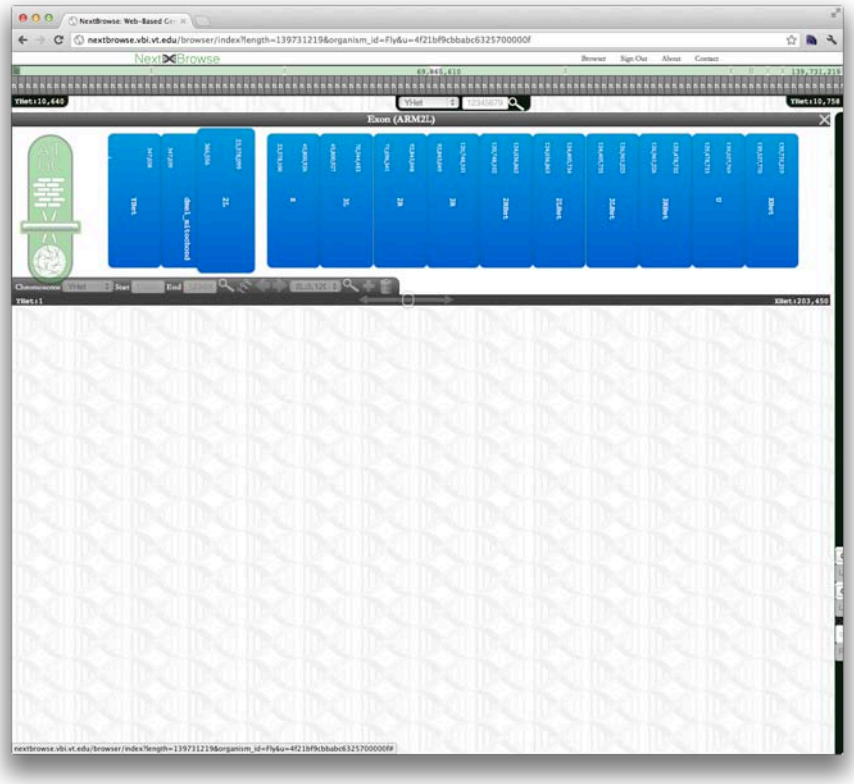


Figure 43 NextBrowse displaying chromosomes.

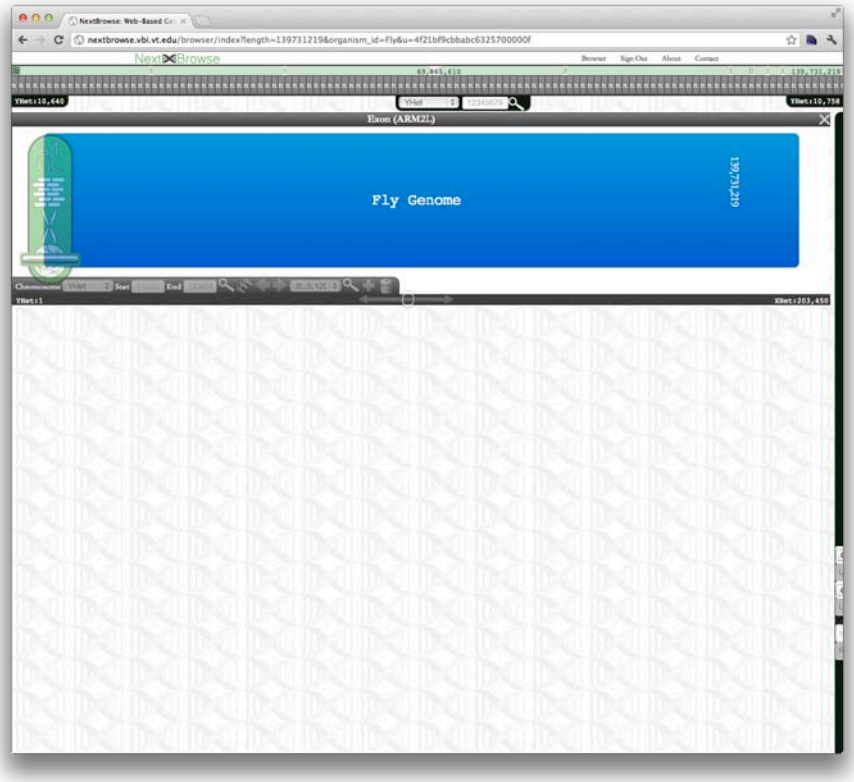


Figure 44 NextBrowse displaying the entire genome.

Chapter 5

NextBrowse Walkthrough

This chapter provides a walkthrough of the NextBrowse software discussed in this thesis.

5.1 Home Screens

To launch NextBrowse users must visit the web address http://www.nextbrowse.vbi.vt.edu using the Google Chrome web browser. Once the webpage loads, the user will be presented with the home page of the application; e.g. Figure 45. From the home screen, the user can choose to visit the sign in, about, contact, or lite browser web page. The about page, Figure 46, lists information about the software and the contact page, Figure 47, provides a way for users to directly contact the developers with feature suggestions, inquiries, or bug reports. The lite browser page provides a way for users to visualize data without the need of logging in, which can be viewed in Figure 48.

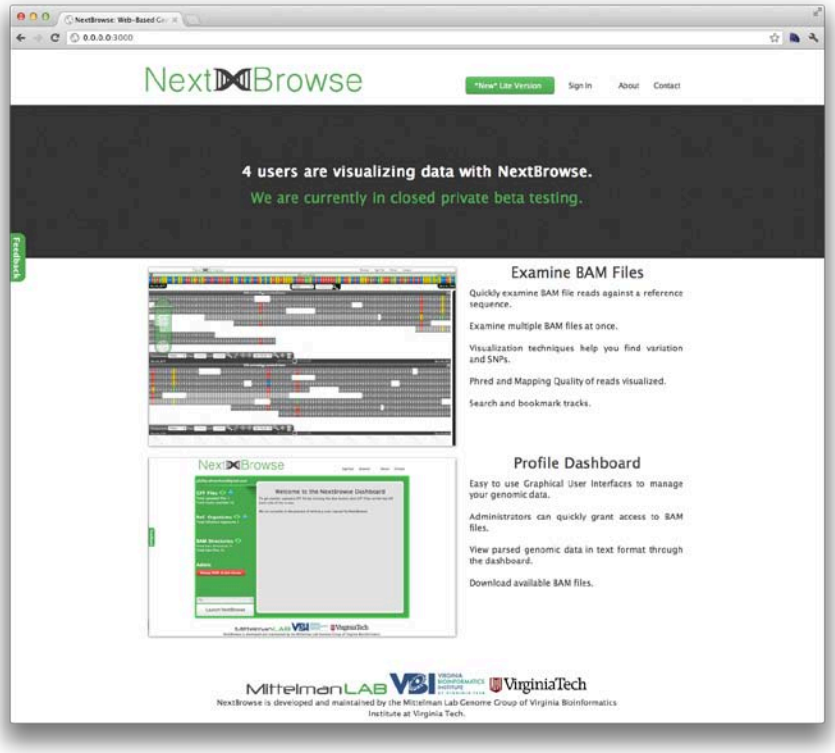


Figure 45 Home screen of NextBrowse. Users can visit the sign in, about, contact, or lite browser page.

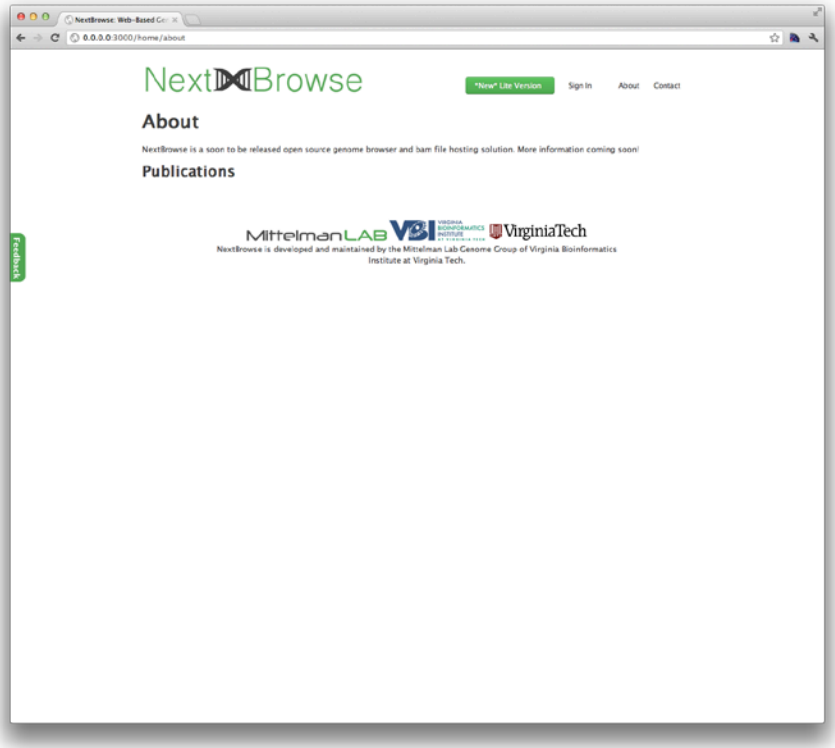


Figure 46 NextBrowse about us page.

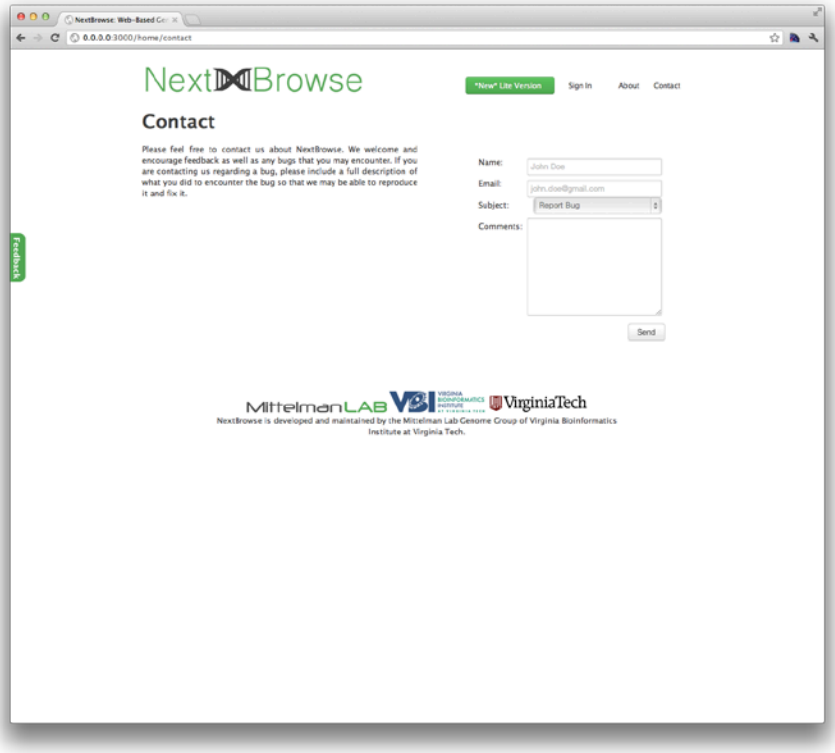


Figure 47 NextBrowse contact us page.

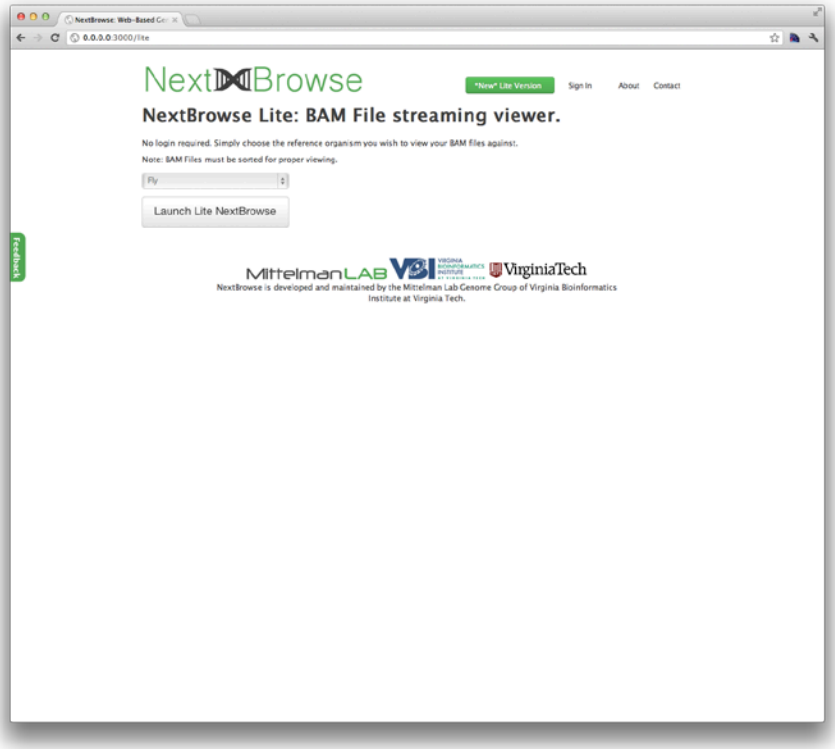


Figure 48 Lite browser page. Users can launch NextBrowse without having to login.

5.2 Authentication Screens

As previously stated, NextBrowse offers user and administrative accounts. Both administrators and users sign up and sign in using the same web pages. If the user clicks a link to the sign in page they will be prompted with a sign in page; e.g. Figure 49. If the user enters incorrect login information they will be notified as seen in Figure 52. If the user does not have an account already, they have the option to sign up for an account and will be presented with the screen similar to Figure 50. If a user forgets their password they have the option to retrieve it from via the forgot password page; e.g. Figure 51. Once a user logs in successfully they have the option to edit their account information. If they choose to edit their account information they will be directed to the edit user page; e.g. Figure 53.

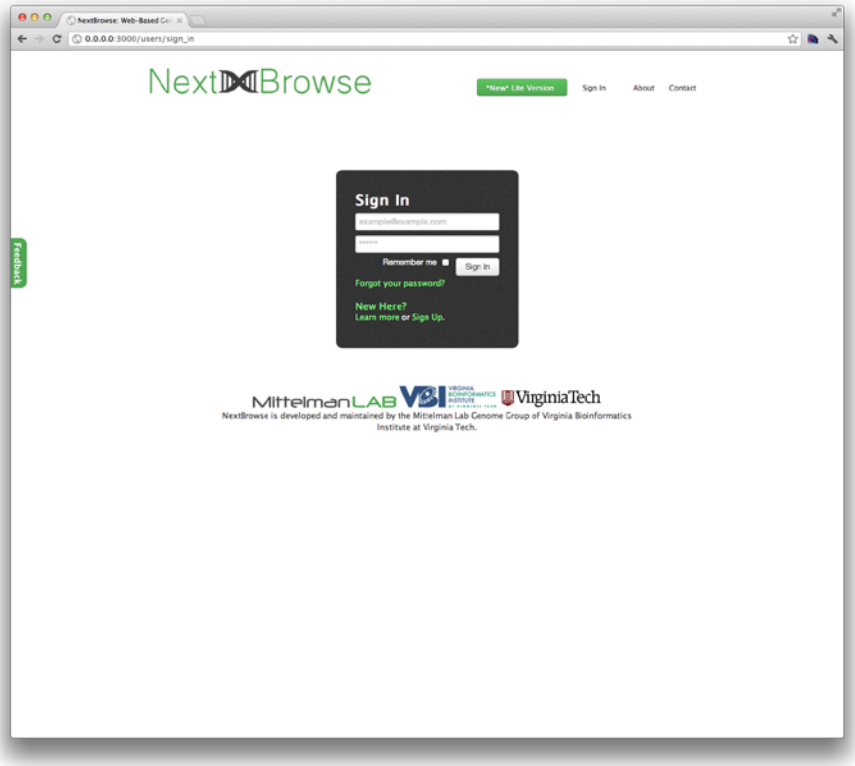


Figure 49 NextBrowse sign in screen.

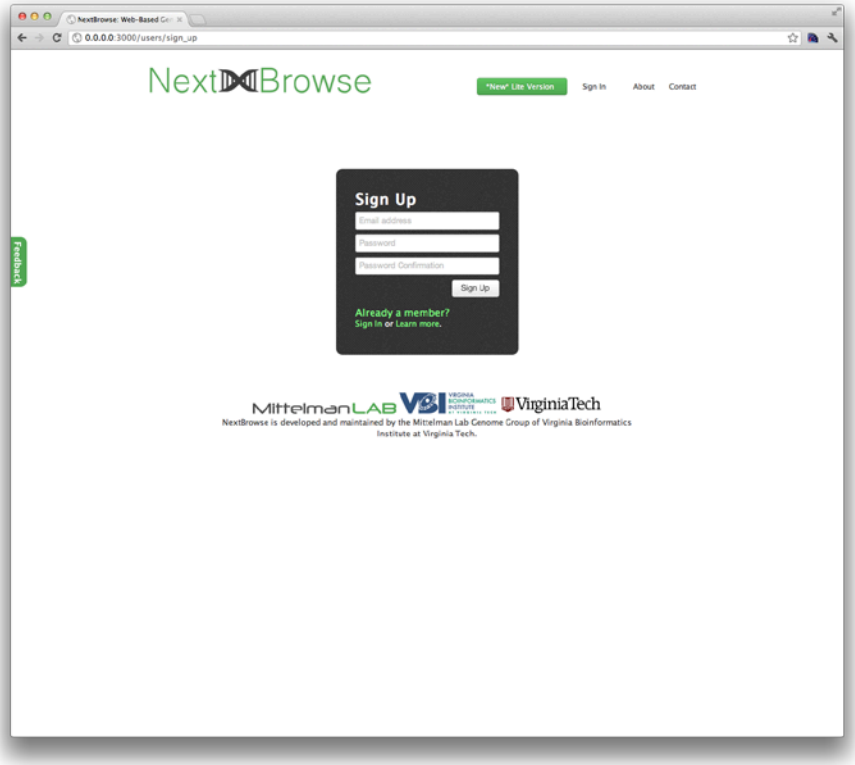


Figure 50 NextBrowse sign up screen.

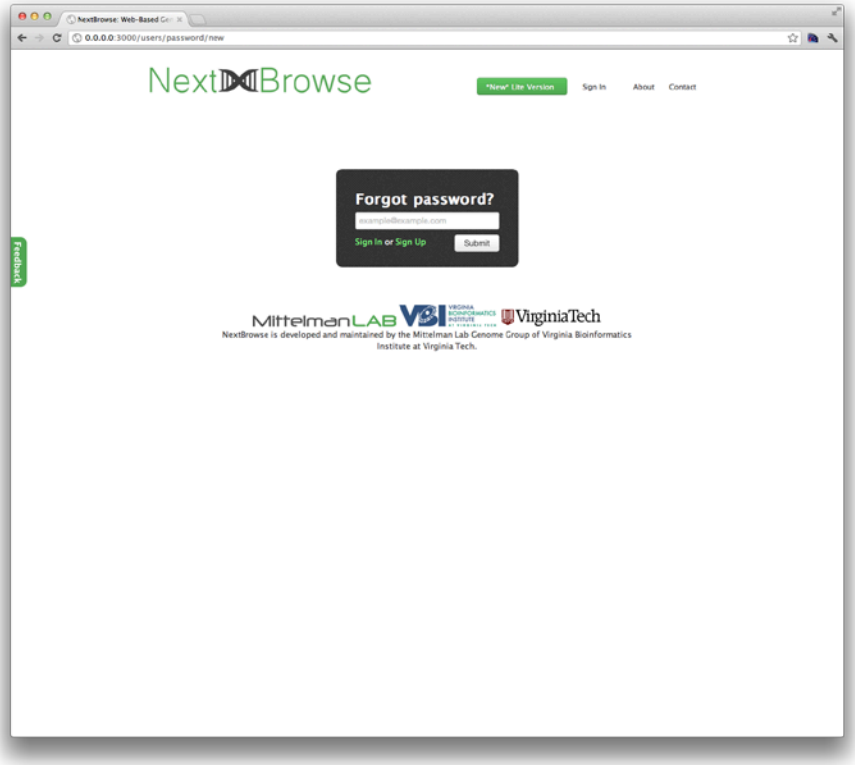


Figure 51 NextBrowse forgot password page.

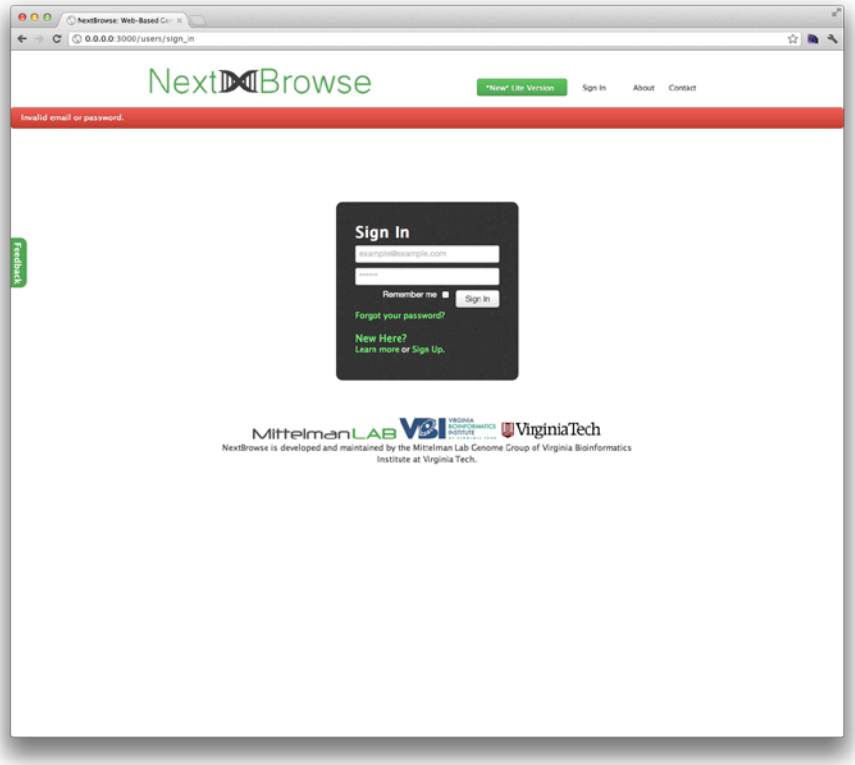


Figure 52 NextBrowse invalid login page.

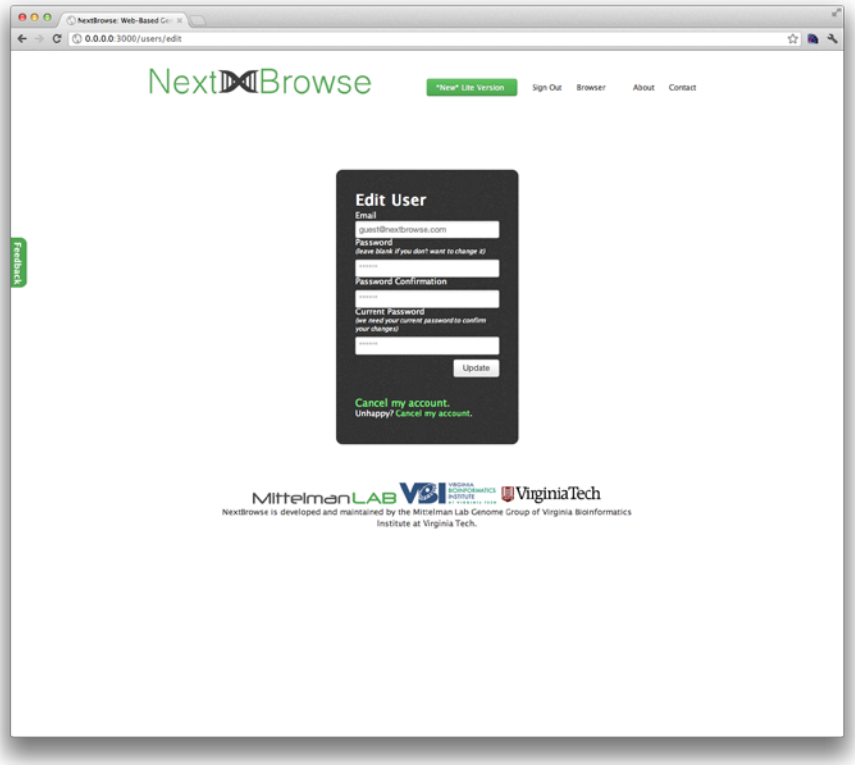


Figure 53 NextBrowse edit user profile page.

5.3 Profile Screens

Once a user successfully logs into NextBrowse, they will be redirected to the profile page, as seen in Figure 54. Once at the profile view, users have a plethora of options. Administrators have the same options as users with a few additions as seen in Figure 55.

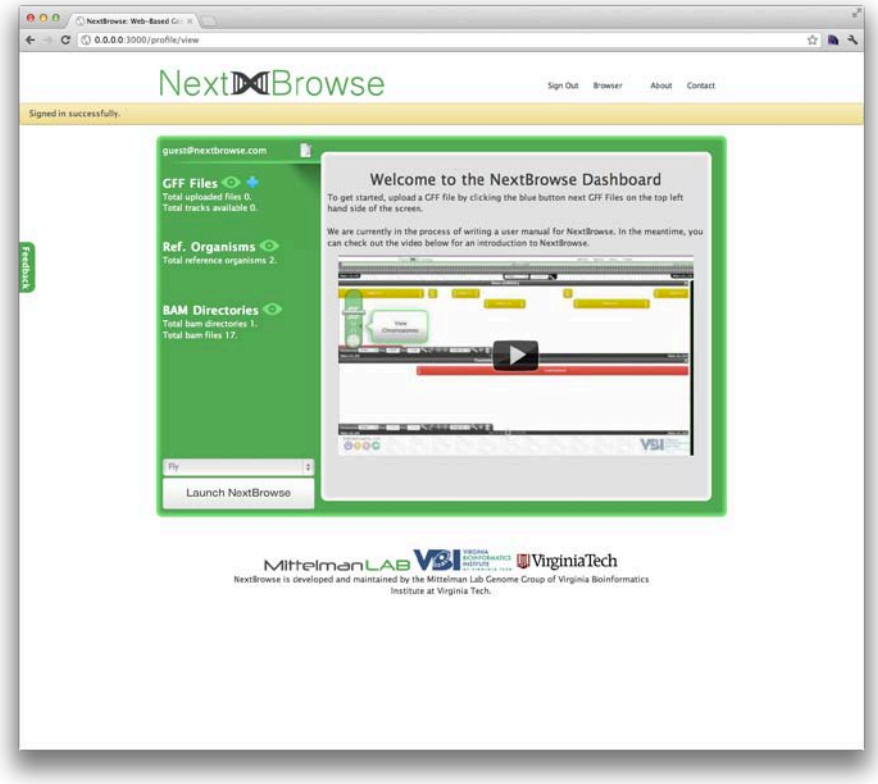


Figure 54 NextBrowse normal user successful sign in profile page.

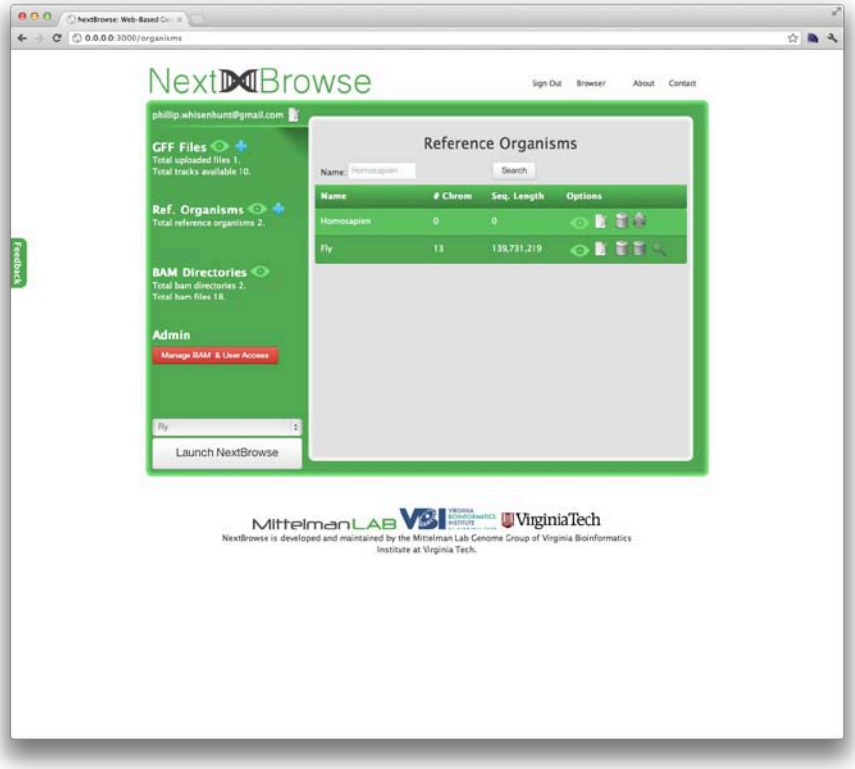


Figure 55 NextBrowse administrative user profile dashboard. Notice the extra options available.

5.4 GFF File Screens

Once logged into NextBrowse users can manage their GFF files using the links in the top left of the profile screen. If a user chooses to upload a GFF file, they will be redirected to the GFF file upload page, as depicted in Figure 56. GFF files are parsed into different tracks based on the features within the file. For example, if a GFF file has five different types of features, uploading it will produce five different types of tracks.

Once the user finishes uploading a GFF file, they will be redirected a page displaying all of the tracks parsed from the GFF file; e.g. Figure 58. Once the file has been uploaded it will be listed under the users available GFF files; e.g Figure 57. If a user clicks on a specific GFF files track they will be directed to a page listing all of the features of that track; e.g. Figure 59.

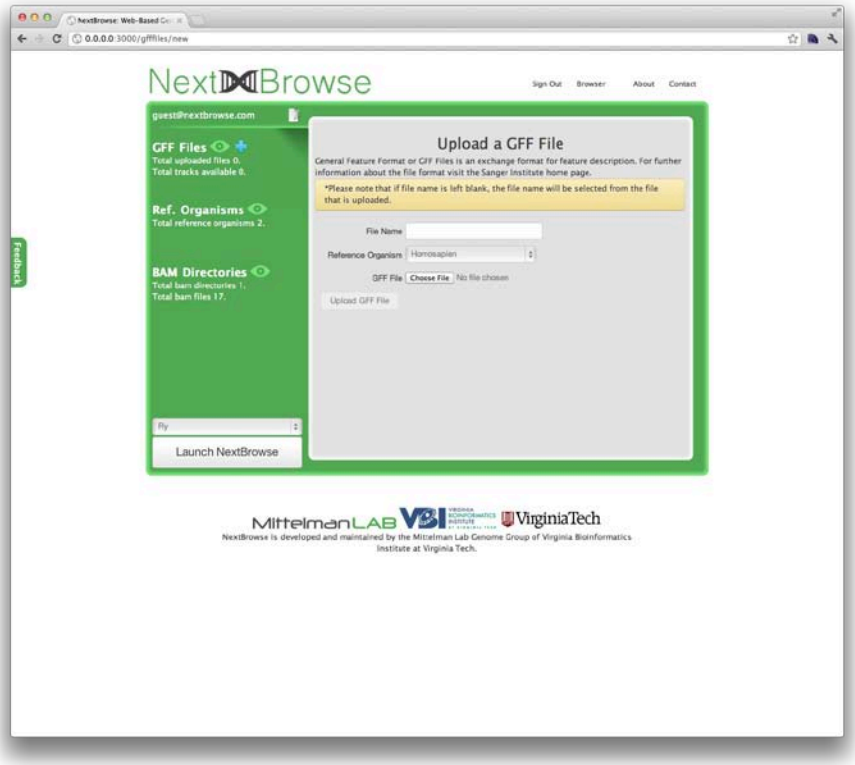


Figure 56 NextBrowse GFF File upload screen.

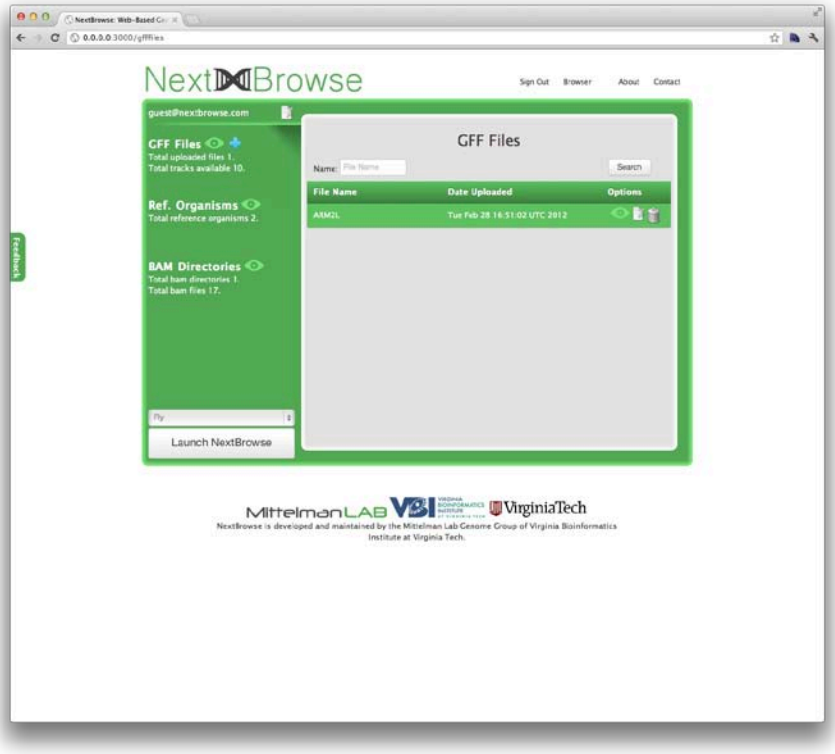


Figure 57 NextBrowse GFF File listing screen.

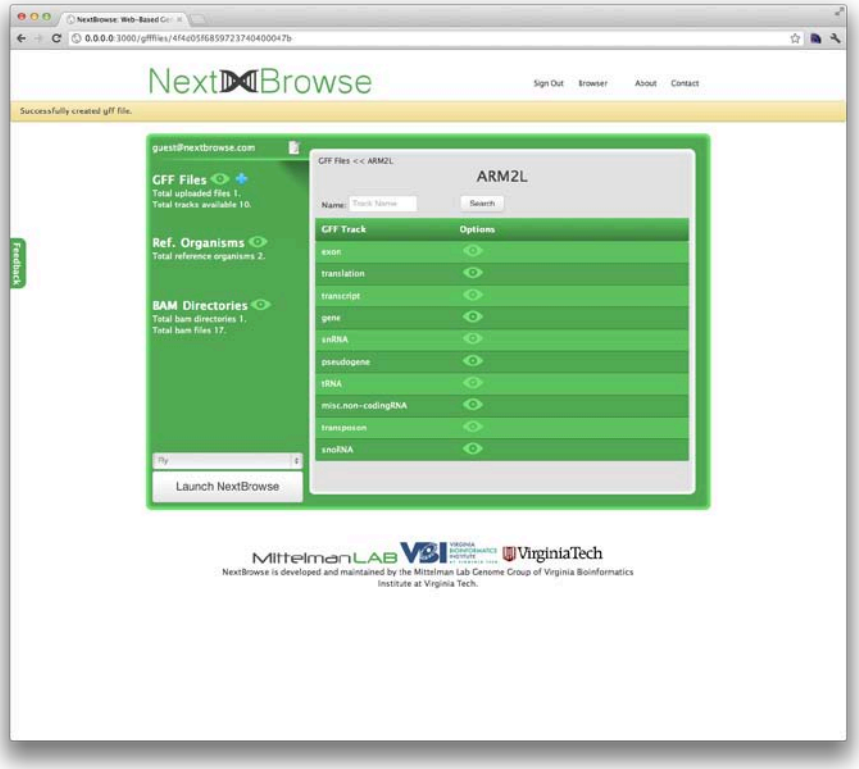


Figure 58 NextBrowse GFF track listing screen for a file named ARM2L.

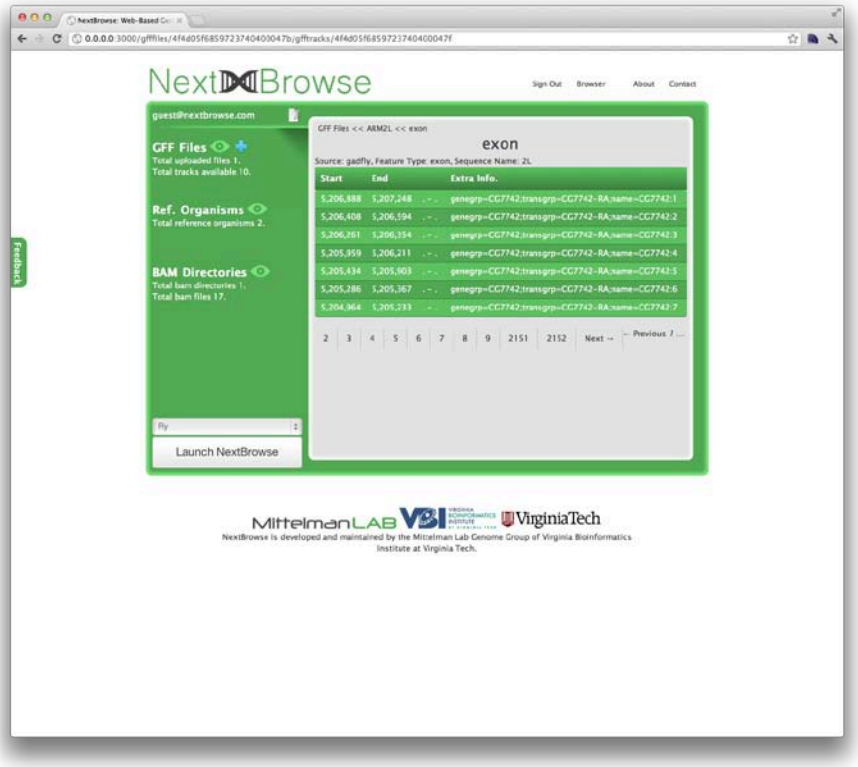


Figure 59 NextBrowse GFF track individual features screen of the track called exon.

5.5 BAM File Screens

Once a user successfully logs into NextBrowse they have the option to view BAM directories that they have access to from clicking the BAM viewing icon; e.g. Figure 60. If a user clicks on a BAM directory they will be directed to a screen listing all of the BAM files in that directory; e.g. Figure 61.

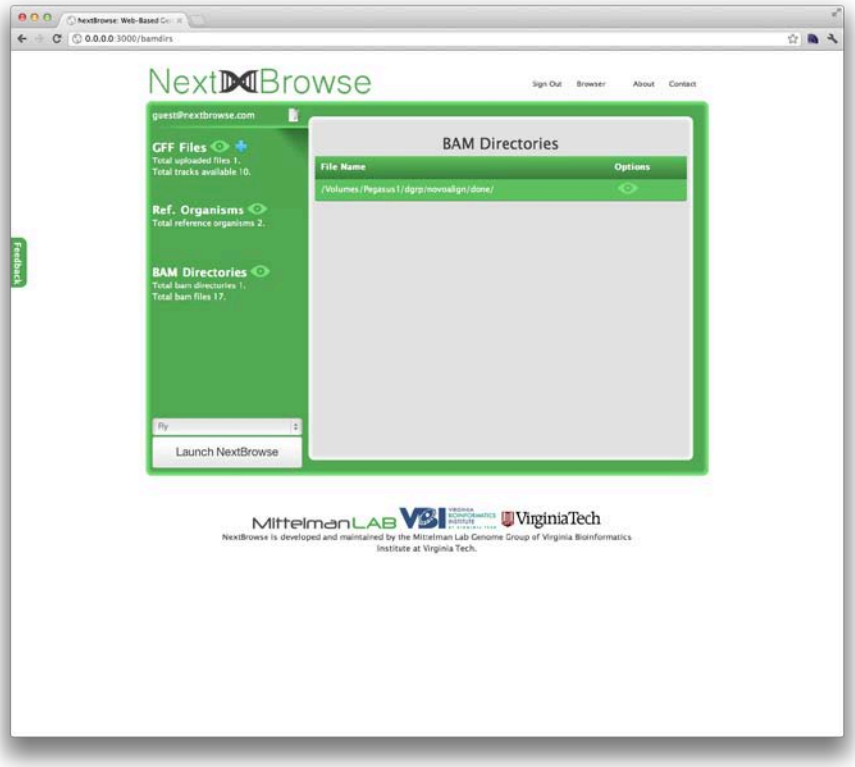


Figure 60 NextBrowse bam directory listing screen.

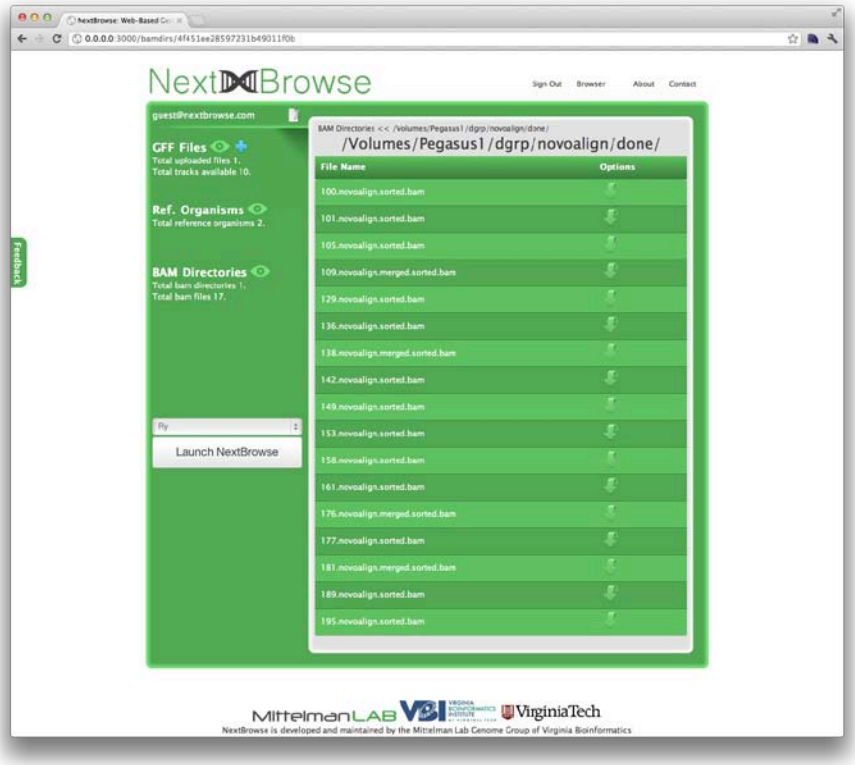


Figure 61 NextBrowse BAM file listing screen.

5.6 Reference Organisms

Once a user logs into NextBrowse they have the option to view the reference organisms. Normal users will be displayed with a page similar to Figure 62 and administrative users will be displayed with a page similar to Figure 63. Users can view the chromosomes of a reference organism by clicking on a reference organisms view icon. Upon viewing a reference organism the user will be directed to a page similar to Figure 64. A user can further click on a chromosome to view the individual reference sequence of that chromosome; e.g. Figure 65. In addition to viewing each reference organisms chromosomes, each user can search a reference organisms sequence; e.g. Figure 66. Administrative users have the option to create a reference organism and to upload a reference sequence for a reference organism; e.g. Figure 67 and Figure 68.

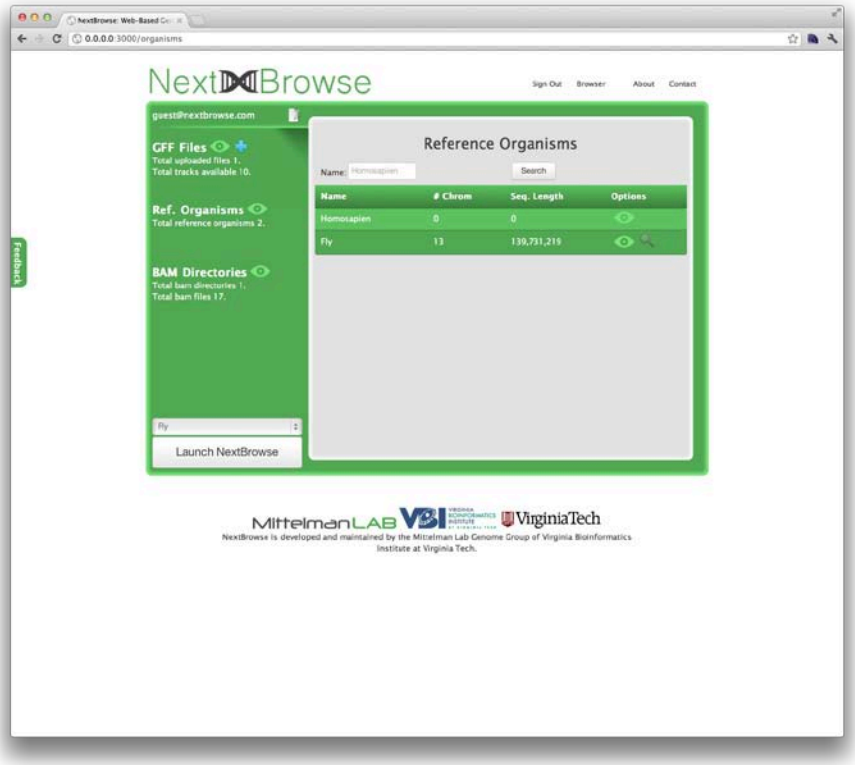


Figure 62 NextBrowse reference organism list screen for normal users.

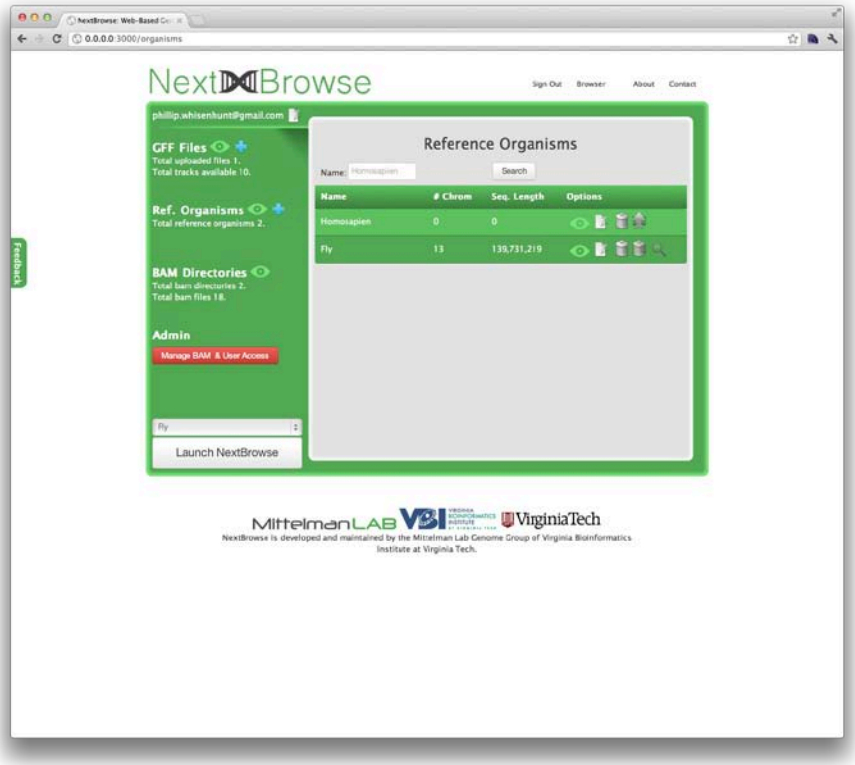


Figure 63 NextBrowse reference organism listing for administrative users. Notice the extra options available.

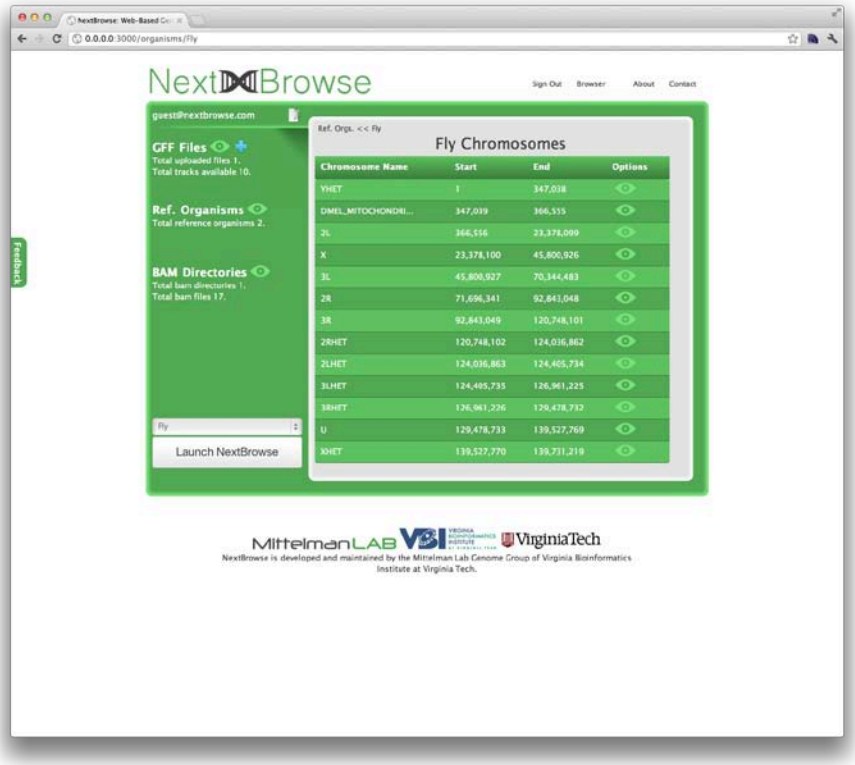


Figure 64 NextBrowse chromosome listing for the reference organism Fly.

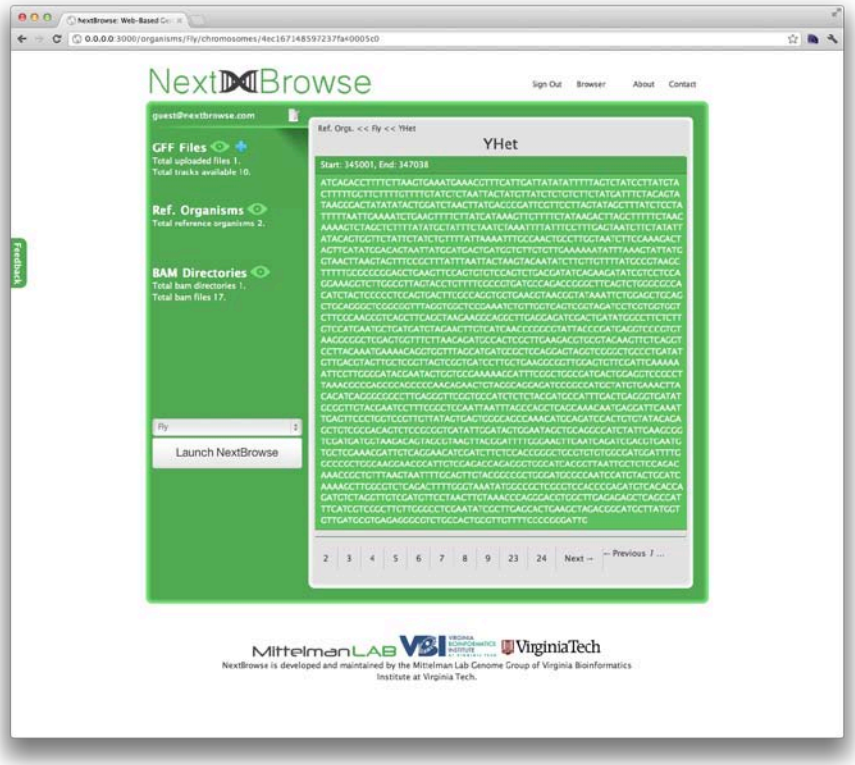


Figure 65 NextBrowse listing reference sequence for a specific chromosome.

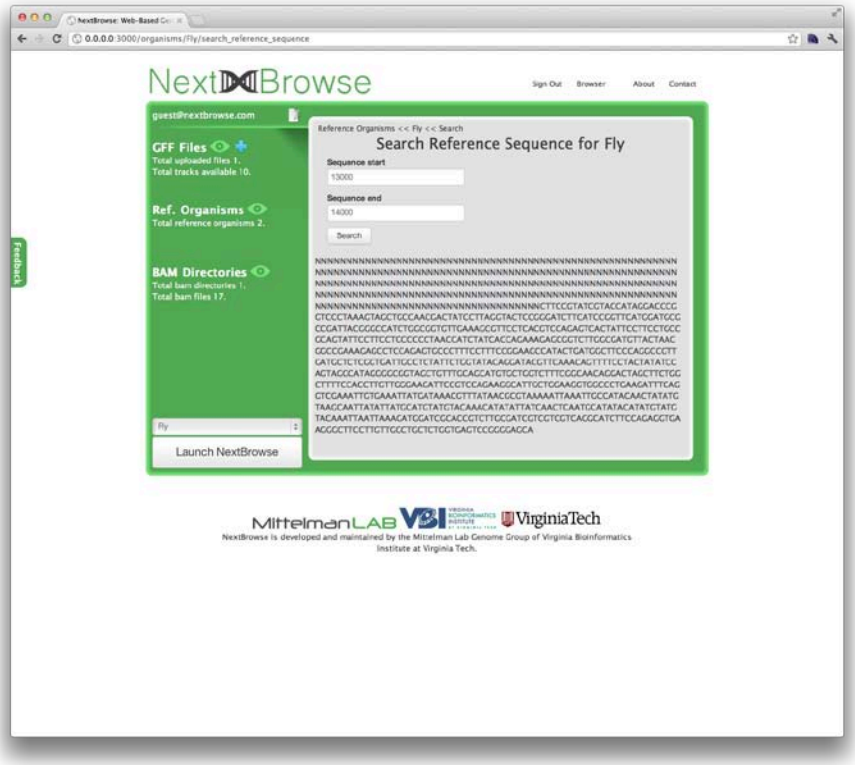


Figure 66 NextBrowse searching a reference organism sequence screen.

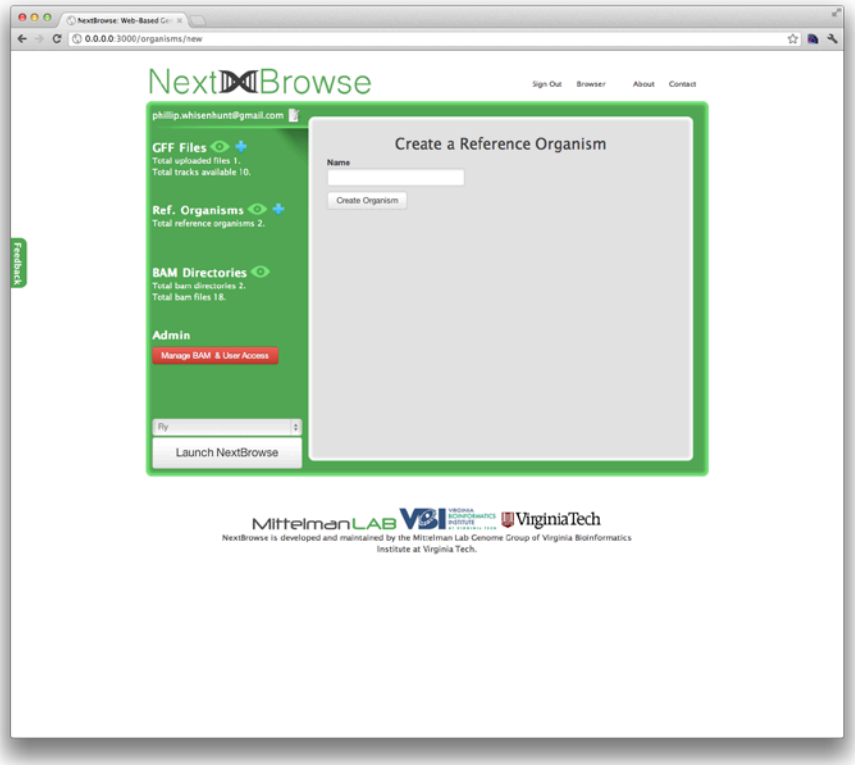


Figure 67 NextBrowse create reference organism screen for administrators.

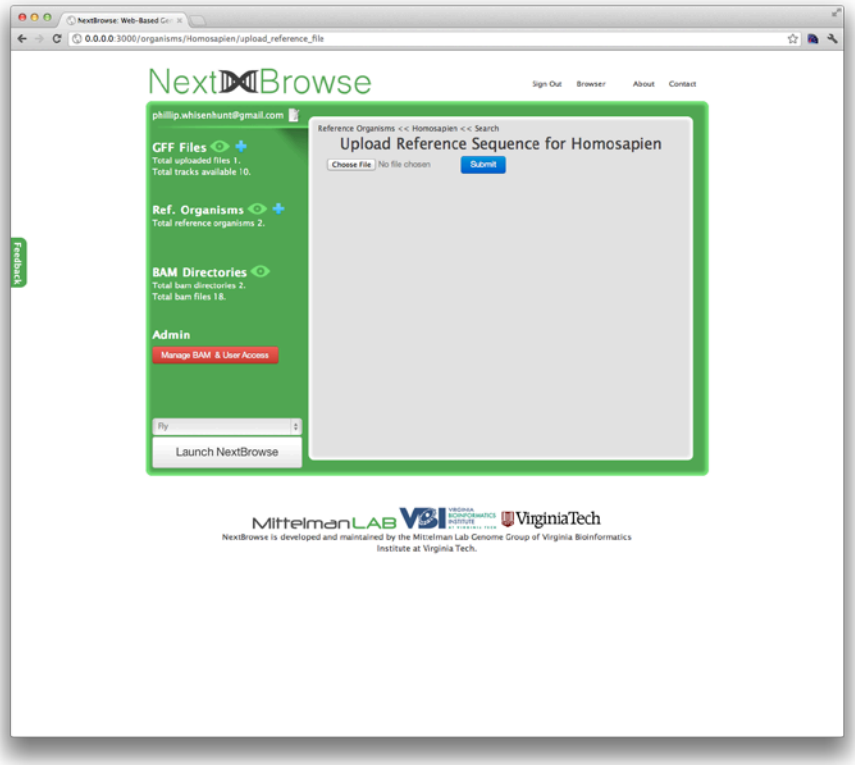


Figure 68 NextBrowse upload reference sequence for a reference organism screen.

5.7 BAM and User Management

Once an administrative user logs in they can access the administrative dashboard by clicking on the Manage BAM and User Accounts red button. Upon clicking a user will be directed to a page similar to Figure 69. Once at the administrative dashboard administrators can delete and add users and add and remove BAM directory access for users.

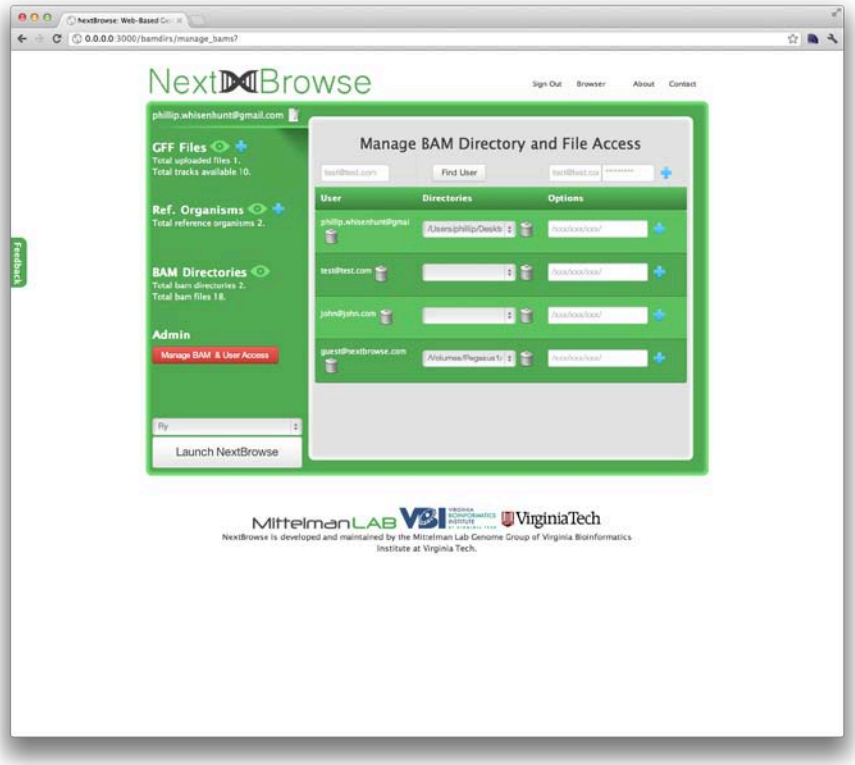


Figure 69 NextBrowse manage BAM and user accounts screen.

Chapter 6

Evaluation of NextBrowse

The following chapter covers a self-evaluation, testing performed, and all documentation of NextBrowse.

6.1 Testing and Validity

Testing of software refers to making sure that a piece of software works as expected. Ruby on Rails is a test driven web application framework. However, given the small scope of Ruby on Rails code developed, it was chosen to not write tests for the software. Instead, testing was performed by the developer as the software was created. JavaScript code can also be tested, but since a large majority of the JavaScript code of NextBrowse handles user events or gestures, testing of JavaScript code was carried out by the developer through manual testing.

Since NextBrowse does not make use of any testing frameworks or test cases, it was imperative to have NextBrowse tested in phases. Alpha testing, the first phase of testing, was performed internally by Dr. David Mittelman's research lab. Bugs and other caveats were reported to the developer and fixed. After Alpha testing, Beta testing was performed. A temporary account was created on NextBrowse and example files were added to the account for Beta testing. SeqAnswers.com is a community forum of biologists and computational biologists. Therefore, account details were posted on the SeqAnswers.com forum for feedback. The thread received over 700 views and the author received feedback on bugs, performance issues, etc. In addition to Alpha and Beta testing, NextBrowse has been extensively tested by its developer.

6.3 Documentation

All source code in NextBrowse is fully documented and commented. This helps facilitate the open source aspect of NextBrowse by allowing any developer to quickly navigate and extend the NextBrowse code base. All JavaScript code follows the jsdoc comment specification and therefore all JavaScript documentation is generated using the jsdoc toolkit [40]. All Ruby code follows the rdoc comment specification, and therefore all Ruby documentation is generated using rdoc [41]. All documentation of NextBrowse is available online at http://www.nextbrowse.vbi.vt.edu.

An analysis of the NextBrowse application code is provided in Table 8 by using cloc [42]. As you can tell for every three lines of code there is roughly one line of comment. This helps demonstrate just how exhaustively the code base is commented.

Language	file	blank	comment	Code
JavaScript	24	285	1,261	4,359
CSS	22	201	442	1,972
Ruby	25	179	624	1,213
SUM	71	665	2,327	7,544

Table 8 Analysis of the program code of NextBrowse.

Chapter 7

Future Work, Contributions, and Conclusions

The following chapter highlights future work that should be explored and summarizes the work described thus far.

7.1 Future Work

Future work of NextBrowse should focus on the following. First, NextBrowse should be expanded to be compatible with other web browsers such as Internet Explorer and Firefox. This functionality would require further development of NextBrowse's CSS and jQuery Transform. In addition to cross compatibility of web browsers, it would be beneficial to provide a mobile version of NextBrowse for mobile devices. This functionality will require the addition of JavaScript source code to register mobile device touch events. To further beef up the security of NextBrowse, Secure Sockets Layer (SSL) security should be considered for NextBrowse to lessen the likelihood that a user account could be compromised.

NextBrowse should extend support for other file types besides GFF and BAM files. Automatic detection of track types should be considered so that different visualization techniques can be used dependent on the track type. Human Computer Interaction usability studies should be performed to analyze the GUI components of NextBrowse to determine the optimal manner in which to configure the GUI.

One of the largest areas that future work should explore is analysis within the browser such as SNP detection and local realignment visualization. All genome browsers should consider adding this functionality. File streaming of other large genomic data types should be strongly considered. Lastly, future work should look to add social networking functionality such as allowing users to compare their genomic data against their relatives or other approved users.

7.2 Contributions

The major contribution of this thesis is the creation of an advanced web-based genome browser, NextBrowse. NextBrowse introduces security and user accounts, improved visualization techniques, improved GUI elements, and BAM file streaming visualization to the field of genomic data visualization. NextBrowse is believed to improve the interpretation of genomic data, specifically BAM files, through the use of improved visualizations and file streaming.

7.3 Conclusions

This thesis has provided a picture of the current state of genome browser visualization and the shortcomings of current genome browsers. To overcome these shortcomings, a new software tool, NextBrowse, was presented for visualizing and interpreting genomic data. We present new visualization techniques of genomic data and improved GUI elements for browsing genomic data. NextBrowse brings many new features to the field of genomic visualization such as security and BAM file streaming visualization. We believe that NextBrowse provides a new and unique way for genomic data to be visualized and interpreted. Furthermore, NextBrowse provides a way for sequencing facilities to rapidly share and host sequence data.

CHAPTER 8 References

1. McKenna, A., et al., *The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data.* Genome research, 2010. **20**(9): p. 1297-303.
2. Nielsen, C.B., et al., *Visualizing genomes: techniques and challenges.* Nat Methods, 2010. **7**(3 Suppl): p. S5-S15.
3. Yates, T., M.J. Okoniewski, and C.J. Miller, *X:Map: annotation and visualization of genome structure for Affymetrix exon array analysis.* Nucleic acids research, 2008. **36**(Database issue): p. D780-6.
4. Arakawa, K., et al., *Genome Projector: zoomable genome map with multiple views.* BMC bioinformatics, 2009. **10**: p. 31.
5. Stein, L.D., et al., *The generic genome browser: a building block for a model organism system database.* Genome Res, 2002. **12**(10): p. 1599-610.
6. Grant, J.R. and P. Stothard, *The CGView Server: a comparative genomics tool for circular genomes.* Nucleic acids research, 2008. **36**(Web Server issue): p. W181-4.
7. Skinner, M.E., et al., *JBrowse: a next-generation genome browser.* Genome Res, 2009. **19**(9): p. 1630-8.
8. Li, H., et al., *The Sequence Alignment/Map format and SAMtools.* Bioinformatics, 2009. **25**(16): p. 2078-9.
9. Robinson, J.T., et al., *Integrative genomics viewer.* Nat Biotechnol, 2011. **29**(1): p. 24-6.
10. Duhl, J., *Rich Internet Applications*, 2003, Macromedia and Intel: Framingham.
11. Nicol, J.W., et al., *The Integrated Genome Browser: free software for distribution and exploration of genome-scale datasets.* Bioinformatics, 2009. **25**(20): p. 2730-1.
12. Stajich, J.E., et al., *The Bioperl toolkit: Perl modules for the life sciences.* Genome research, 2002. **12**(10): p. 1611-8.
13. Pan, X., L. Stein, and V. Brendel, *SynBrowse: a synteny browser for comparative sequence analysis.* Bioinformatics, 2005. **21**(17): p. 3461-8.
14. Wang, H., et al., *SynView: a GBrowse-compatible approach to visualizing comparative genome data.* Bioinformatics, 2006. **22**(18): p. 2308-9.
15. Barnes, M.R., *Exploring the landscape of the genome.* Methods in molecular biology, 2010. **628**: p. 21-38.
16. Bare, J.C., et al., *Integration and visualization of systems biology data in context of the genome.* BMC bioinformatics, 2010. **11**: p. 382.

17. Fielding, R.T., *Architectural Styles and the Design of Network-based Software Architectures*, in *Information and Computer Science*2000, University of California, Irvine.
18. Valim, J. *Devise is a flexible authentication solution for Rails based on Warden*. 2012 [cited 2012; Available from: https://github.com/plataformatec/devise.
19. Jim Menard, M.D., Kyle Banker, Tyler Brock. *A Ruby driver for MongoDB*. 2012 [cited 2012; Available from: http://rubygems.org/gems/mongo.
20. Keepers, B. *A Ruby Object Mapper for Mongo*. 2012; Available from: https://github.com/jnunemaker/mongomapper.
21. Giménez, H. *Validatable is a library for adding validations*. 2012; Available from: https://github.com/jnunemaker/validatable.
22. Mandrup, K. *Mongo Mapper integration for Devise framework*. 2011; Available from: https://github.com/kristianmandrup/mm-devise.
23. Marohnić, M., *Pagination library for Rails 3, Sinatra, Merb, DataMapper, and more*. 2012.
24. Frank, F. *This is a JSON implementation as a Ruby extension in C*. 2012; Available from: http://rubygems.org/gems/json.
25. Peek, J. *Rack-based asset packaging system*. 2012; Available from: https://github.com/sstephenson/sprockets.
26. *mongoDB*. 2012; Available from: http://www.mongodb.org/.
27. Crockford, D., *The application/json Media Type for JavaScript Object Notation (JSON)*, 2006: IETF.
28. W3C. *HTML & CSS*. Web Design and Applications 2012; Available from: http://www.w3.org/standards/webdesign/htmlcss.
29. Project, T.j. *jQuery*. 2012; Available from: http://jquery.com/.
30. Project, T.j. *jQuery User Interface*. 2012; Available from: http://jqueryui.com/.
31. Thompson, C. *qTip - The jQuery tooltip plugin*. 2012; Available from: http://craigsworks.com/projects/qtip/.
32. Ashkenas, J. *Underscore.js*. 2012; Available from: http://documentcloud.github.com/underscore.
33. Ashkenas, J. *Backbone.js*. 2012; Available from: http://documentcloud.github.com/backbone/.
34. Raftery, D. *Apprise The attractive alert alternative for jQuery*. 2012; Available from: http://thrivingkings.com/apprise/.

35. Down, T. *log4javascript is a JavaScript logging framework based on the Java logging framework log4j*. 2012; Available from: http://log4javascript.org/.
36. Twitter. *Twitter Bootstrap*. 2012; Available from: http://twitter.github.com/bootstrap/.
37. Institute, W.T.S. *GFF: an Exchange Format for Feature Description*. 2011; Available from: http://www.sanger.ac.uk/resources/software/gff/.
38. Cline, M.S. and W.J. Kent, *Understanding genome browsing*. Nature biotechnology, 2009. **27**(2): p. 153-5.
39. Ewing, B., et al., *Base-calling of automated sequencer traces using phred. I. Accuracy assessment*. Genome research, 1998. **8**(3): p. 175-85.
40. *jsdoc-toolkit - A documentation generator for JavaScript. - Google Project Hosting*. 2012; Available from: http://code.google.com/p/jsdoc-toolkit/.
41. *RDoc - Documentation from Ruby Source Files*. 2012; Available from: http://rdoc.sourceforge.net/.
42. Danial, A. *CLOC -- Count Lines of Code*. 2011; Available from: http://cloc.sourceforge.net.