Setting Up the JBrowse Genome Browser

Mitchell E. Skinner[1] and Ian H. Holmes[1]

[1]University of California at Berkeley, Berkeley, California

ABSTRACT

JBrowse is a Web-based tool for visualizing genomic data. Unlike most other Web-base genome browsers, JBrowse exploits the capabilities of the user's Web browser to make scrolling and zooming fast and smooth. It supports the browsers used by almost all Internet users, and is relatively simple to install. JBrowse can utilize multiple types of data in a variety of common genomic data formats, including genomic feature data in bioperl databases, GFF files, BED files, and quantitative data in wiggle files. This unit describes how to obtain the JBrowse software, set it up on a Linux or Mac OS X computer running as a Web server, and incorporate genome annotation data from multiple sources into JBrowse. After completing the protocols described in this unit, the reader will have a Web site that other users can visit to browse the genomic data. *Curr. Protoc. Bioinform.* 32:9.13.1-9.13.13. © 2010 by John Wiley & Sons, Inc.

Keywords: genome browser • visualization • JBrowse • gff • bed • wiggle • GenBank

INTRODUCTION

JBrowse (Skinner et al., 2009) is a compact and easily-installed software package for the Web-based browsing of genomes and genome annotations.

Like other genome browsers, such as the UCSC browser (*UNIT 1.4*; Kent et al., 2002) or GBrowse (*UNIT 9.9*; Stein et al., 2002), JBrowse enables the user to visualize discrete features and gene structures alongside quantitative information. Unlike other Web-based genome browsers, with JBrowse, the Web browser (e.g., Firefox, Internet Explorer, Safari, etc.) does most of the work; this allows the user to interact with the data without having to wait for the server to respond. This approach of moving work from the Web server to the Web browser makes JBrowse easier to set up and maintain than other browsers, while being faster and easier to utilize due to the use of advanced client-side user interface enhancements. JBrowse is part of the Generic Model Organism Database (GMOD) project (*http://www.gmod.org*), a suite of tools for generating and maintaining genomic databases. Like the other tools in the GMOD project (of which it is a part), JBrowse is a piece of software that is self-installed and used with one's own data. This is in contrast to services such as the UCSC genome browser (*UNIT 1.4*; Kent et al., 2002) or the Ensembl genome browser (*UNIT 1.15*; Hubbard et al., 2007), which focus mainly on hosting the most-in-demand genomes with a more limited scope for adding one's own data.

Each of the protocols in this unit describes a different way of setting up a JBrowse instance. An "instance" is a copy of the JBrowse software combined with genomic data for an organism, placed on a server so that an end user can visit the site in their Web browser and browse a genome. The Basic Protocol describes setting up a JBrowse instance for genome annotations stored in the common FASTA, GFF3, and/or BED formats. Alternate Protocol 1 describes an alternate way to set up JBrowse, for genome annotations that are stored in a relational database. Alternate Protocol 2 describes the process of setting up a JBrowse instance from a GenBank record. The Support Protocol describes the initial installation process for JBrowse and its dependencies.

Current Protocols in Bioinformatics 9.13.1-9.13.13, December 2010
Published online December 2010 in Wiley Online Library (wileyonlinelibrary.com).
DOI: 10.1002/0471250953.bi0913s32

SETTING UP A JBrowse INSTANCE USING FLATFILES AS A DATA SOURCE

This protocol details the installation of JBrowse and the set-up process, assuming that the user has successfully completed the installation steps described in the Support Protocol.

In JBrowse, software running in the Web browser downloads specially formatted data files from the server and displays a visual representation of them. The process of setting up a JBrowse instance involves running the programs that come with JBrowse to generate those specially formatted data files from a data source.

After JBrowse is downloaded and installed (see Support Protocol), the JBrowse directory contains a file named "index.html". This file is the main HTML front-end to JBrowse, and will (eventually) be displayed in the end user's Web browser (e.g., Firefox). However, before it is ready to do this, the user will need to add genomic data.

JBrowse supports a variety of data sources; it uses the same underlying BioPerl machinery as the GBrowse genome browser, and therefore supports all of the same data sources. In addition to genomic databases (described in Alternate Protocol 1), JBrowse supports several flat-file formats, including FASTA (*APPENDIX 1B*) for reference sequence data and GFF3 and BED for genomic feature data.

Broadly speaking, the GFF3 and BED formats are used to store similar data—mainly, the boundaries of genomic features such as genes (e.g., a line in a GFF3 or BED file might describe the gene TP53 and the state that it is found on chromosome 17 from bases 7,571,720 to 7,590,863). Beyond such basic feature information, though, the two formats differ; GFF3 files can contain richer information about each feature, and can describe multi-level feature hierarchies (e.g., a gene feature being the parent of a transcript feature, which itself is the parent of several exon, intron, and CDS or UTR features). BED, on the other hand, is simpler and more limited in the kinds of data it can store and the depth of the feature hierarchies. Both formats are commonly used to store and communicate genomic feature information. JBrowse can use either one, or both.

Each feature in the GFF3 format consists of a line of text, containing several columns separated by tab characters; here are a few lines from the example GFF3 file included with JBrowse:

```
ctgA  example  remark  13280  16394  .  +  .  Name=f08
ctgA  example  remark  15329  15533  .  +  .  Name=f10
```

A brief description of the GFF3 columns is shown in Table 9.13.1; for full details, refer to the specification at *http://www.sequenceontology.org/gff3.shtml.*

The BED format is an alternative to GFF3. The BED format also represents each feature by a line containing tab-separated columns. Here are a few example lines:

```
ctgA  1659  1984  f07   .   +
ctgA  3014  6130  f06   .   +
ctgA  4715  5968  f05   .   -
ctgA  13280 16394 f08   .   +
ctgA  15329 15533 f10   .   +
```

The columns of the BED format are briefly described in Table 9.13.2. For full details, refer to the format FAQ from UC Santa Cruz at *http://genome.ucsc.edu/FAQ/FAQformat.html.*

Table 9.13.1 Description of Columns in GFF3 Format

Column	Description
Reference sequence name	Name of the sequence that the feature is defined on (commonly a chromosome or contig)
Source	The source of the feature. For features that were generated by software, this is commonly the name of the software; for human-curated features, this is commonly the name of the group performing the curation (e.g., a model organism database)
Type	The type of the feature from the Sequence Ontology (*http://www.sequenceontology.org/*)
Start	The start position of the feature
End	The end position of the feature
Score	The score of the feature (optional)
Strand	The strand on which the feature occurs (optional)
Phase	The phase of the feature (e.g., for CDS features, indicates that the first full codon in the feature starts this many bases after the start of the feature) (optional)
Attributes	A set of name = value pairs specifying information not contained within the other GFF columns (often contains name and ID information) (optional)

Table 9.13.2 Description of Columns in BED Format

Reference sequence name	Name of the chromosome or scaffold
Start	Start position of the feature
End	End position of the feature
Name	Name of the feature (optional)
Score	Score of the feature (optional)
Strand	Strand on which the feature is defined (optional)
Thick start	Position where the feature should be drawn thickly (e.g., for transcript models, the start of the coding sequence) (optional)
Thick end	Position where the feature should no longer be drawn thickly (e.g., for transcript models, the end of the coding sequence) (optional)
Item Rgb	Color to use to draw the feature (optional)
Block count	Number of blocks (e.g., for transcript models, the number of exons) (optional)
Block sizes	Sizes of blocks (e.g., for transcript models, the size of each exon) (optional)
Block starts	Start points of blocks (e.g., for transcript models, the starting point of each exon) (optional)

This protocol utilizes the term "user" to describe the person installing JBrowse and importing data into a JBrowse instance; the phrase "end user" will describe the person who visits that JBrowse instance in their Web browser. In this protocol, the user will set up an instance of JBrowse using the included example data. The tutorial example refers to the "volvox" genome, but the sequence and all annotation data is simulated and does not correspond to volvox or any other living organism.

Necessary Resources

Hardware

Linux or Mac OS X machine
RAM (~512 megabytes; more for large data sets)
Internet connection

Software

Installation of all necessary software as described in the Support Protocol

1. Import reference sequence data into the JBrowse instance. To do this, first open a terminal window, go into the JBrowse directory, and then run the prepare-refseqs.pl program from the command line (the "$" symbol represents the Unix shell prompt).

```
$ cd <JBrowse directory>
$ bin/prepare-refseqs.pl --fasta
  docs/tutorial/data_files/volvox.fa
```

   where, `<JBrowse directory>` is the directory created in the Support Protocol; commonly /var/www/html/jbrowse (on Red Hat/CentOS/Fedora machines), /var/www/jbrowse (on Ubuntu), or /Library/WebServer/Documents/jbrowse (on OS X).

   If an end user visits the JBrowse instance now, they should see the JBrowse UI (user interface) elements: buttons for zooming and scrolling, a drop-down box for selecting a reference sequence, and a box showing the current genomic location being viewed. If the end user zooms in all the way, they should see DNA sequence data (Fig. 9.13.1).

   To see this, load or reload the "index.html" page in the browser (http://localhost/jbrowse).

2. Add genomic feature tracks. Run `bin/flatfile-to-json.pl` to add the gene track:

```
$ bin/flatfile-to-json.pl --tracklabel gene --key
  "Gene" --gff docs/tutorial/data_files/volvox.gff3
  --cssclass feature2 --getLabel --type gene
  --autocomplete all
```

   This step uses the command-line arguments shown in Table 9.13.3. Note that this example command wraps across several lines, but should be entered as a single line. Use the backslash (\) character to continue the command across several lines.

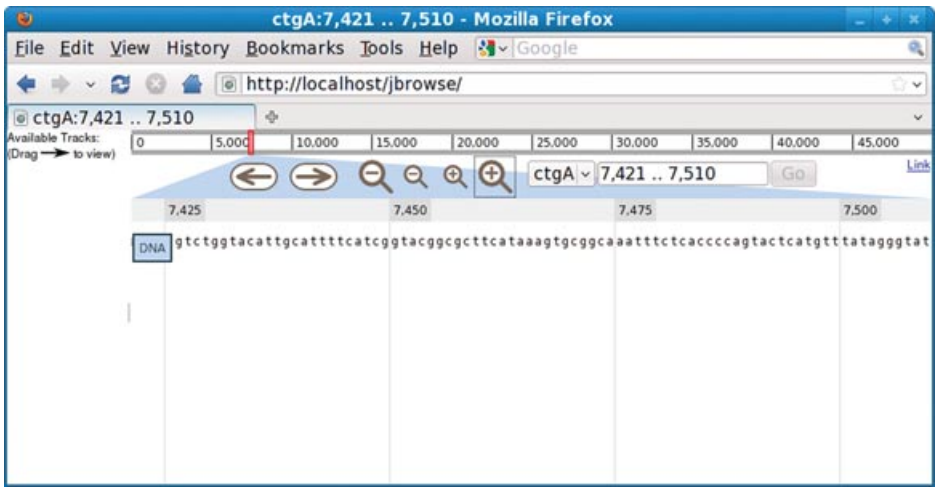

Figure 9.13.1 JBrowse instance with just DNA sequence track, zoomed in to the individual base level.

Table 9.13.3 Description of Command-Line Arguments Used with the `flatfile-to-json.pl` Script

Argument	Required	Purpose
`--tracklabel`	Yes	Unique identifier for the track
`--key`	Yes	Human-readable label for the track
`--gff` or `--gff2` or `--bed`	Yes	Path to input flat file
`--cssclass`	No	Describes how the features should look
`--getLabel`	No	Include feature labels in the output
`--type`	No	Only process features of the given type
`--autocomplete`	No	Make it possible to search for features in this track

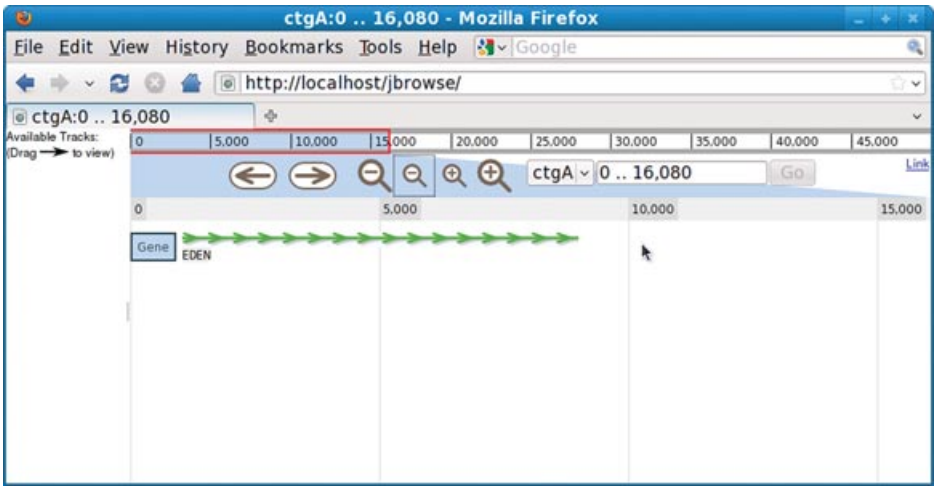

Figure 9.13.2 Screenshot of JBrowse with newly created "gene" track.

If an end user visits the JBrowse instance in his or her Web browser, then he or she will see a new track as shown in Figure 9.13.2.

In JBrowse, genomic feature data is organized into `tracks`*, which usually contain features of the same or similar types. For example, there might be a "transcript" track, an "EST" track, and an "SNP" track. JBrowse's* `bin/flatfile-to-json.pl` *program takes as input a file with genomic feature data and also takes some command-line arguments that specify how to process the file, and uses those inputs to generate JBrowse's data files for one track. If the user wants to create multiple tracks, they can run the program multiple times and specify different settings each time.*

3. Run `bin/flatfile-to-json.pl` again, with different arguments, to create a new track from a BED file (Fig. 9.13.3):

```
$ bin/flatfile-to-json.pl --bed
  docs/tutorial/data_files/volvox-remark.bed
  --tracklabel remark --key "Remark" --cssclass
  feature3 --getLabel --autocomplete label
```

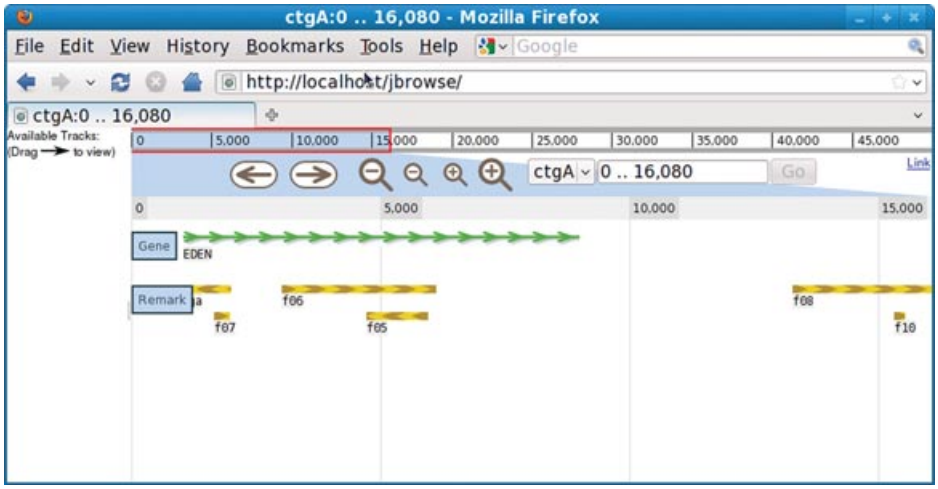

Figure 9.13.3 Screenshot of JBrowse instance with newly created “remark” track.

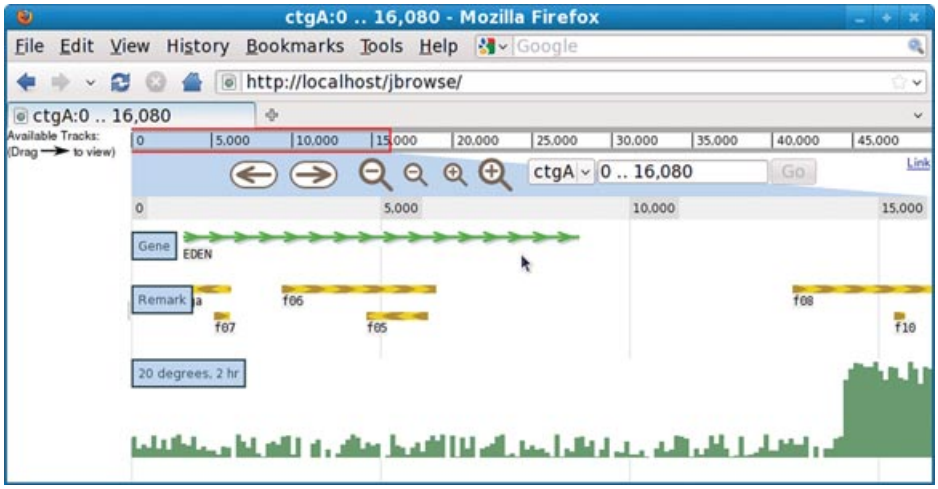

Figure 9.13.4 Screenshot of JBrowse instance with newly created quantitative track.

4. Add a quantitative track using `bin/wig-to-json.pl` (Fig. 9.13.4):

```
$ bin/wig-to-json.pl --wig
  docs/tutorial/data_files/volvox_microarray.wig
  --tracklabel wiggle_example --key "20 degrees, 2 hr"
```

JBrowse also supports visualizing quantitative data in the “wiggle” format. This type of data is useful for dense quantitative data like transcription level data (e.g., from microarrays or RNA-seq). If you have data with a numeric value at a series of genomic locations, that data can be put into a wiggle file. The genomic locations can be specified in a few different ways as described at http://genome.ucsc.edu/goldenPath/help/wiggle.html, but the wiggle file used in this protocol is in the simplest format. Here are a few lines from the example wiggle file included with JBrowse; the first line specifies that this file contains data points that are separated by a fixed distance (indicated below by “fixedStep”); the data points are 100 bases wide (indicated by “span=100”), starting at base 1 (indicated by “start=1”), and occurring every 100 bases thereafter (indicated by “step=100”):

```
fixedStep chrom=ctgA start=1 step=100 span=100
281
183
213
191
288
```

The result is shown in Figure 9.13.4.

5. Run the `bin/generate-names.pl` program to enable the end user to search for feature names or identifiers:

```
$ bin/generate-names.pl
```

This step enables end users to search for specific features by name. Earlier in this protocol the —`autocomplete` option passed to flatfile-to-json.pl caused this program to generate lists of feature names (in addition to the locations of the associated features) and place them in separate track files. The generate-names.pl program takes all of the separate per-track lists of names, collects them together, and generates another set of specially-formatted files. Those files allow the web browser to search for a specific feature name or ID. If generate-names.pl generates no error messages, that means the program has completed successfully. If an end user visits the page now, enters a feature name (e.g., "f05") into the text box that shows the current genomic location, and clicks "Go", they will be taken to that feature.

SETTING UP A JBrowse INSTANCE USING A RELATIONAL DATABASE AS THE DATA SOURCE

ALTERNATE PROTOCOL 1

JBrowse can also access data in any of the databases that work with the BioPerl Bio::DB API. Two common database schemas that support that API are the chado schema and the Bio::DB::SeqFeature::Store schema. This protocol describes using JBrowse with Bio::DB::SeqFeature::Store and the MySQL database server.

Necessary Resources

Hardware

Linux or Mac OS X machine
RAM (~512 megabytes; more for large data sets)
Internet connection

Software

Installation of all necessary software as described in the Support Protocol

1. Create the database using the MySQL command-line tool:

```
$ mysql -u root

mysql> create database jbrowse_example;
Query OK, 1 row affected (0.00 sec)

mysql> grant all on jbrowse_example.* to
  ''@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

2. Load the database using BioPerl's `bp_seqfeature_load.pl` script

```
$ bp_seqfeature_load.pl --dsn=dbi:mysql:jbrowse_example
  --adaptor=DBI::mysql --fast --create
  docs/tutorial/data_files/volvox.fa
  docs/tutorial/data_files/volvox.gff3
```

In the Basic Protocol, the user runs the `bin/flatfile-to-json.pl` *program once per track. This protocol, by contrast, uses one program that generates multiple tracks from a data source. The detailed settings for each track (supplied as command-line arguments in the Basic Protocol) are instead specified in a configuration file. This configuration file is a convenient place to store those track-specific settings so that they can be reused. Then, if the data source is updated with new data, the user can re-run steps 5 and 6 with the same configuration file to update the JBrowse instance with the new data.*

The JBrowse distribution includes an example configuration file; however, it needs to be adjusted to work with the database that the user created in steps 1 and 2 of this protocol (see step 3).

3. Edit the example configuration file. Find the "`db_args`" setting and change it from:

```
"db_args": { "-adaptor": "memory",
  "-dir": "docs/tutorial/data_files" },
```

to:

```
"db_args": {"-adaptor": "DBI::mysql",
  "-dsn": "dbi:mysql:jbrowse_example"},
```

This tells JBrowse that the database is a MySQL database, and, of the mysql databases on the machine, to use the one called "jbrowse_example".

Most of the remainder of the file contains the per-track settings. These mostly correspond to the command-line parameters used with the `flatfile-to-json.pl` *program described in the Basic Protocol (Table 9.13.3).*

4. As in the Basic Protocol, use the `prepare-refseqs.pl` program to import reference sequence data into this JBrowse instance:

```
$ bin/prepare-refseqs.pl --conf
  docs/tutorial/conf_files/volvox.json --refs ctgA
```

This performs the same function as in the Basic Protocol, but uses different command-line arguments. The `--conf` *command-line argument specifies the configuration file; the* `--refs` *command line argument gives the names of the reference sequences in the format of a comma-separated list (e.g., if there was a second contig named "ctgB", the* `--refs` *argument would then be ctgA,ctgB).*

5. Run `bin/biodb-to-json.pl` to extract data from the database and add it to this JBrowse instance.

```
$ bin/biodb-to-json.pl --conf
  docs/tutorial/conf_files/volvox.json
```

6. Run `generate-names.pl` to enable name/ID searching:

```
$ bin/generate-names.pl
```

ALTERNATE PROTOCOL 2

SETTING UP A JBrowse INSTANCE USING NCBI GenBank FILES AS A DATA SOURCE

JBrowse does not directly support files in GenBank formats, but a bioperl conversion program can take GenBank files and create GFF3 files that JBrowse can process. This protocol describes the process of taking a GenBank file, converting it into GFF3, and incorporating it into a JBrowse instance.

NOTE: Data can be imported into JBrowse from GenBank .gbk files, by converting those files into GFF3 with scripts from BioPerl.

Necessary Resources

Hardware

Linux or Mac OS X machine
RAM (~512 megabytes; more for large data sets)
Internet connection

Software

Installation of all necessary software described in the Support Protocol

Files

Go to: *ftp://ftp.ncbi.nih.gov/genomes/Saccharomyces_cerevisiae/CHR_I/*
Download the file: NC_001133.gbk

1. Use the BioPerl `bp_genbank2gff3.pl` script to generate a GFF3 file:

```
$ bp_genbank2gff3.pl NC_001133.gbk
```

2. Edit the generated file, named `NC_001133.gbk.gff`, and change this line:

```
# sequence-region NC_001133 1 230208
```

to the following:

```
##sequence-region NC_001133 1 230208
```

This corrects a small bug in the `bp_genbank2gff3.pl` *script.*

3. Go into the JBrowse directory and use `bin/prepare-refseqs.pl` to initialize the JBrowse instance. JBrowse comes with a suitable configuration file (named `yeast_genbank.json`) in the docs/examples/config directory:

```
$ bin/prepare-refseqs.pl --conf
  docs/examples/config/yeast_genbank.json --refs
  NC_001133
```

4. Use `bin/biodb-to-json.pl` to populate the tracks:

```
$ bin/biodb-to-json.pl --conf
  docs/examples/config/yeast_genbank.json
```

5. Make the features searchable with `bin/generate-names.pl`:

```
$ bin/generate-names.pl
```

DOWNLOADING/INSTALLING JBrowse

SUPPORT PROTOCOL

The protocol describes how to obtain and install the JBrowse software and its prerequisites. After performing the steps in this protocol, the user should be able to proceed with the Basic Protocol, or Alternate Protocols 1 or 2.

Necessary Resources

Hardware

Linux or Mac OS X machine
RAM (~512 megabytes; more for large data sets)
Internet connection

1. Install perl and related packages (usually available from the operating system provider; installation procedures depend upon the operating system). This protocol

describes how to install the necessary software on the Red Hat, CentOS, Fedora, and Ubuntu flavors of Linux, and on Apple's Mac OS X operating system.

a. Red Hat/CentOS/fedora
```
$ sudo yum install perl
```
b. Ubuntu
```
$ sudo apt-get install perl
```
c. OS X

Perl is installed on OSX by default.

2. For quantitative tracks, install a C++ compiler program, the 'make' program, the 'libpng' library, and the libpng header files. As with Perl, these are usually provided by the operating system provider:

a. Red Hat/CentOS/fedora
```
$ sudo yum install gcc-c++ make libpng libpng-devel
```
b. Ubuntu
```
$ sudo apt-get install g++ make libpng12-0 libpng12-dev
```
c. OS X

For the C++ compiler and 'make' program, install XCode tools, either using the install CD/DVD that came with the computer, or by downloading them directly from Apple at *http://developer.apple.com/*.

The 'libpng.dylib' library file and 'png.h' header file should already be installed in directories /usr/X11R6/lib and /usr/X11R6/include, respectively.

3. Install the necessary perl modules from CPAN:

```
$ sudo cpan
cpan> install CJFIELDS/BioPerl-1.6.1.tar.gz JSON
  JSON::XS
```

4. For Alternate Protocol 1, install MySQL:

a. Red Hat/CentOS/fedora
```
$ sudo yum install mysql perl-DBD-MySQL
```
b. Ubuntu
```
$ sudo apt-get install mysql-server libdbd-mysql-perl
```
c. OS X

There are multiple ways of installing MySQL on OS X; go to the MySQL downloads Website (*http://dev.mysql.com/downloads/mysql/*) and download a .dmg (disk image) file containing the MySQL server.

5. Download the JBrowse 1.1 release from http://jbrowse.org/releases/jbrowse-1.1.zip.

Alternatively, download the zipfile directly from the command line, using the 'wget' utility, if it is installed on the system:

```
$ wget http://jbrowse.org/releases/jbrowse-1.1.zip
```

6. Uncompress the zip file downloaded in step 5:

```
$ unzip jbrowse-1.1.zip
```

7. Build the quantitative track (wiggle) renderer:

   a. Red Hat/CentOS/Fedora/Ubuntu
   ```
   $ cd jbrowse
   $ ./configure
   $ make
   ```
   b. OS X
   ```
   $ cd jbrowse
   $ ./configure
   $ make GCC_LIB_ARGS=-L/usr/X11R6/lib GCC_INC_ARGS=-
   I/usr/X11R6/include
   ```

8. Move the JBrowse directory to where it will be served by the Web server:

   a. Red Hat/CentOS/fedora
   ```
   $ cd ..
   $ sudo mv jbrowse/ /var/www/html/jbrowse
   ```
   b. Ubuntu
   ```
   $ cd ..
   $ sudo mv jbrowse/ /var/www/jbrowse
   ```
   c. OS X
   ```
   $ cd ..
   $ sudo mv jbrowse/ /Library/WebServer/Documents/
   jbrowse
   ```

9. To check that the JBrowse instance is accessible, go to *http://localhost/jbrowse* in the Web browser. A completely blank page should be seen; this page will be populated when a genome sequence and tracks are added as described in the Basic Protocol.

GUIDELINES FOR UNDERSTANDING RESULTS

The results of running each major step of the Basic Protocol can be seen in the figures referenced. After completing the Support Protocol, and after each major step of the Basic Protocol, go to the URL corresponding to the JBrowse installation, providing immediate feedback on the success or failure of that step:

a. Following installation (see Support Protocol), a Web browser pointing at the appropriate URL (typically *http://localhost/jbrowse*) should display a blank page. Some JavaScript errors may be displayed in the console.
   If, instead of a blank page, an error such as “403 Forbidden” or “404 Not Found” is observed, then problems with the installation may have occurred (see Troubleshooting).
b. After adding the sequence data, reload the Web page to yield a view similar to that shown in Figure 9.13.1.
c. After adding a gene annotation track, and successive annotation tracks, reload the Web page to yield a view similar to that shown in Figures 9.13.2 and 9.13.3.
d. After adding a quantitative (wiggle) track, reload the web page to yield a view similar to that shown in Figure 9.13.4.

Problems and potential resolutions to these problems are described below in the Troubleshooting section.

COMMENTARY

Background Information

A variety of Web-based genome browsers have existed for several years. Those systems generally have database engines running on the Web server alongside a body of genome-browser software. When a user visits a genome browser page, that software queries the database and draws a picture of a genomic region and sends that picture to the Web browser, which just displays it.

Over the last 10 years, many software systems have begun to shift work away from the Web server and onto the user's Web browser. Having part of the application running on the user's machine allows for richer interaction between the user and the application. It also reduces the amount of work that the server must do, which reduces or eliminates the time users have to spend waiting for the server to respond.

JBrowse is an example of this shift; it does part of the work on the server but moves the bulk of the work to the user's Web browser. The initial server-side work that JBrowse does (getting data from some data source) is almost the same as the first part of what GBrowse (*UNIT 9.9*) does. JBrowse uses many of the same software components for data access that were built for the GBrowse genome browser. This sharing and reuse of software is one of the ways that JBrowse builds on the work done for GBrowse, and it means that JBrowse is compatible with the same set of data sources as GBrowse.

The next step differs; instead of using that data to draw a picture on the server, JBrowse uses it to create files on the server that the Web browser downloads and uses to present a graphical representation of a region of a genome. In addition to the advantages listed above, this approach simplifies the server environment, which helps make JBrowse simple to set up. When a user visits a JBrowse instance, the server does not have to run any JBrowse software; it just sends the files that were created in advance by the JBrowse programs. Many Web-hosting environments enable users to put such static files online, but restrict the user's ability to install and run software. JBrowse can be used in those environments; all the user has to do is install and run JBrowse on his or her own computer and then upload the resulting JBrowse instance to the server.

Critical Parameters and Troubleshooting

If problems with JBrowse are encountered, the best thing to do is to consult the email list, first by searching the archive at the following URL: *http://gmod.827538.n3.nabble.com/JBrowse-f815920.html*.

And then, if none of the messages displayed solve the problem, send a message to the list at *gmod-ajax@sourceforge.net*.

Common problems that users encounter include the following:

1. The JBrowse programs fail with an error message—there are multiple reasons why this can happen; the main ones are:

a. Running programs from outside of the JBrowse directory—JBrowse works best if the programs are run from within the JBrowse directory (the directory that contains the index.html file) as shown above.

b. Missing or outdated dependencies—some JBrowse programs use parts of BioPerl in which bugs have recently been fixed. Using the latest available stable version of BioPerl (1.6.1 as of this writing) will help avoid problems.

2. When a user visits a JBrowse instance in their Web browser, they get a 403 (Forbidden) error—this can arise as a result of restrictive permissions in the JBrowse directory; try going into the JBrowse directory and changing the permissions of all files so they are readable by Unix users other than yourself, as follows:

```
$ chmod -R o+r .
```

3. Subfeatures not showing up—subfeatures (such as exons in a transcript) are, by default, not included in the files that JBrowse generates. To include them, specify `--getSubs` (for `flatfile-to-json.pl`) or "subfeatures": true in the config file (for `biodb-to-json.pl`).

Acknowledgements

The authors thank Lincoln Stein for suggesting this article, and Oscar Westesson and Dave Clements for their comments on the manuscript. This work was funded by NIH grant HG004483.

Literature Cited

Hubbard, T.J., Aken, B.L., Beal, K., Ballester, B., Caccamo, M., Chen, Y., Clarke, L., Coates, G., Cunningham, F., Cutts, T., Down, T., Dyer, S.C.,

Fitzgerald, S., Fernandez-Banet, J., Graf, S., Haider, S., Hammond, M., Herrero, J., Holland, R., Howe, K., Howe, K., Johnson, N., Kahari, A., Keefe, D., Kokocinski, F., Kulesha, E., Lawson, D., Longden, I., Melsopp, C., Megy, K., Meidl, P., Ouverdin, B., Parker, A., Prlic, A., Rice, S., Rios, D., Schuster, M., Sealy, I., Severin, J., Slater, G., Smedley, D., Spudich, G., Trevanion, S., Vilella, A., Vogel, J., White, S., Wood, M., Cox, T., Curwen, V., Durbin, R., Fernandez-Suarez, X.M., Flicek, P., Kasprzyk, A., Proctor, G., Searle, S., Smith, J., Ureta-Vidal, A., and Birney, E. 2007. Ensembl 2007. *Nucleic Acids Res.* 35:D610-D617.

Kent, W.J., Sugnet, C.W., Furey, T.S., Roskin, K.M., Pringle, T.H., Zahler, A.M., and Haussler, D. 2002. The human genome browser at UCSC. *Genome Res.* 12:996-1006.

Skinner, M.E., Uzilov, A.V., Stein, L.D., Mungall, C.J., and Holmes, I.H. 2009. JBrowse: A next-generation genome browser. *Genome Res.* 19:1630-1638.

Stein, L.D., Mungall, C., Shu, S., Caudy, M., Mangone, M., Day, A., Nickerson, E., Stajich, J.E., Harris, T.W., Arva, A., and Lewis, S. 2002. The generic genome browser: A building block for a model organism system database. *Genome Res.* 12:1599-1610.

Key Reference

Skinner et al., 2009. See above.

This article describes in detail how JBrowse works.

Internet Resources

http://jbrowse.org/

The main JBrowse Web site: has links to the JBrowse code, documentation, email lists, publications, and other Internet resources relevant to JBrowse.