

UC Berkeley

UC Berkeley Previously Published Works

Title

JBrowse 2: a modular genome browser with views of synteny and structural variation

Permalink

<https://escholarship.org/uc/item/7c10t2hw>

Journal

Genome Biology, 24(1)

ISSN

1474-760X

Authors

Diesh, Colin

Stevens, Garrett J

Xie, Peter

et al.

Publication Date

2023

DOI

10.1186/s13059-023-02914-z

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, available at <https://creativecommons.org/licenses/by/4.0/>


Peer reviewed

SOFTWARE

Open Access



JBrowse 2: a modular genome browser with views of synteny and structural variation

Colin Diesh¹, Garrett J Stevens¹, Peter Xie¹, Teresa De Jesus Martinez¹, Elliot A. Hershberg¹, Angel Leung¹, Emma Guo¹, Shihab Dider¹, Junjun Zhang², Caroline Bridge², Gregory Hogue², Andrew Duncan², Matthew Morgan³, Tia Flores³, Benjamin N. Bimber⁴, Robin Haw², Scott Cain², Robert M. Buels¹, Lincoln D. Stein² and Ian H. Holmes^{1*} 

*Correspondence:
ihh@berkeley.edu

¹ Department of Bioengineering, Stanley Hall, University of California, Berkeley, CA 94720, USA

² Adaptive Oncology, Ontario Institute for Cancer Research, MaRS Centre, 661 University Avenue, Suite 510, Toronto, ON M5G 0A3, Canada

³ Center for Applied Systems and Software, 224 Milne Computer Center, 1800 SW Campus Way, Oregon State University, Corvallis, OR 97331, USA

⁴ Oregon National Primate Research Center, Oregon Health and Science University, Beaverton, OR 97006, USA

Abstract

We present JBrowse 2, a general-purpose genome annotation browser offering enhanced visualization of complex structural variation and evolutionary relationships. It retains core features of JBrowse while adding new views for synteny, dotplots, break-points, gene fusions, and whole-genome overviews. It allows users to share sessions, open multiple genomes, and navigate between views. It can be embedded in a web page, used as a standalone application, or run from Jupyter notebooks or R sessions. These improvements are enabled by a ground-up redesign using modern web technology. We describe application functionality, use cases, performance benchmarks, and implementation notes for web administrators and developers.

Background

Genome browsers are a fundamental visualization and analysis tool for genomics. As the technology underpinning the field has progressed—from the study of individual genes, through whole genomes, up to multiple related genomes—the linear DNA sequence has provided a natural visual frame for presenting biological hypotheses (such as annotated gene and variant locations) alongside the primary evidence for those hypotheses. While the genome browser has proved long-lived as a visualization tool, the progression of sequencing technology has influenced the types of visualization needed. Sequencing is now sufficiently affordable that population genomics and comparative genomics have become commonplace. Long-read sequencing has enabled the investigation of structural variation, resolution of individual genotypes from a mixture, long haplotypes, and improved genome assemblies that were inaccessible using short reads. In addition, a diversity of sequencing kits are generating a wealth of data on epigenetic and transient states of the cell, such as DNA–protein associations, methylation, and RNA transcript levels. All of this information is genome-mappable and therefore viewable in a genome



© The Author(s) 2023. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>. The Creative Commons Public Domain Dedication waiver (<http://creativecommons.org/publicdomain/zero/1.0/>) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

browser, but the new technology demands new visualization tools—and modalities—to represent it appropriately.

History of JBrowse and the GMOD project

The tools and resources maintained by the Generic Model Organism Database project (GMOD) have enabled many genome projects to develop their own genome databases and websites. The GMOD project has developed and maintained two genome browsers: (i) the Perl-based GBrowse genome annotation browser [1], the first portable web-based genome browser to achieve widespread adoption, and (ii) the JavaScript browser JBrowse [2, 3], which introduced client-side rendering, a single-page user interface that avoided page reloads, drag-and-drop annotation tracks, animated panning and zooming transitions, and a static-site deployment model.

The original JBrowse app (henceforth “JBrowse 1”) has been reliable and extremely popular. However, it has become increasingly difficult to extend JBrowse 1 due to deep-rooted design assumptions (such as the assumption that only one genome would ever be displayed) and its dependence on older software libraries. This paper describes JBrowse 2, a complete rewrite of JBrowse 1 with a similar user interface but a modern software architecture. As we report in this manuscript, JBrowse 2 goes well beyond the capabilities of JBrowse 1. JBrowse 2 is particularly well-suited to visualizing genomic structural variants and evolutionary relationships among genes and genomes with syntenic visualizations.

Structural variant and synteny visualization tools

Many genome browsers, including the GMOD browsers listed above, use a reference genome to provide a coordinate system in which to align annotations and evidence, including related genomes. This paradigm is ideal for visualizing individual reference genomes and small localized variants, such as single nucleotide polymorphisms and small indels. However, within this paradigm, it can become complicated to visualize structural variations whose alignment to the reference coordinate system departs strongly from collinearity, such as big duplications, deletions, translocations, insertions, inversions, and other complex rearrangements. A similar point holds regarding visualization of synteny between genomes: inter-genome alignments can be collinear over small scales but structurally disrupted over larger scales.

Several specialized tools have been developed to visualize synteny or structural variation. GBrowse-Syn is an interactive tool that allows comparison of regions of multiple genomes against a reference sequence [4]. It uses a joining database representing links between different species and maps sequence coordinates in the aligned segments or synteny blocks [4]. The Artemis Comparison Tool (ACT) enables comparisons between sequences and annotations at the genome and base pair level [5]. Other dedicated synteny views such as SimpleSynteny and Cinteny are capable of visualizing synteny across multiple genomes. SimpleSynteny is a web-based tool providing a pipeline that enables customization of contig organization instead of pure computational predictions for visualizing synteny [6].

Tools to analyze structural variants (SVs) have also been developed in the past. Ribbon, for example, is a visualization tool developed to support long-read evidence in the

analysis of structural variation [7]. By displaying long-read and whole genome alignments, Ribbon is able to display genomic links that could span several genes going through multiple variants. The general-purpose circular visualization tool Circos [8] also supports views that visualize large-scale variation. Copy number variant (CNV) viewers such as the CNSpector can visualize copy number variation and large-scale structural variation that enable the analysis of CNV to detect abnormalities or sequence variants between multiple samples [9]. General purpose genome browsers such as IGV also remain popular for analyzing SVs [10]. An overview of the various visual paradigms of structural variation can be found in [11].

Results

Advances in JBrowse 2

JBrowse 2 combines the well-established paradigms of general-purpose genome browsers with specialized views of synteny and structural variation. It still uses the fundamental concept of a linear coordinate scheme based on a reference genome, but it also introduces alternative views including circular views, dotplot views, comparative synteny views, and the ability to show discontinuous regions in the Linear Genome View. This provides a number of different views on structurally disrupted genomes.

Compared to other tools for visualizing structural variants and synteny, JBrowse 2 is most similar to general-purpose genome browsers (GBrowse-Syn and Artemis) in that it renders syntenic relationships between generic linear visualizations of genomes, their annotations, and supporting evidence. However, JBrowse 2 also includes views that draw extensively on user interface concepts pioneered by Ribbon and Circos, and indeed includes many of the views described in [11]. These are all tied together by a modern web application framework that enables researchers to navigate between these different views, combining multiple coarse-grained, fine-grained, and non-linear views of the genome. In this way, users can begin by visualizing large-scale variation, zoom in to examine a particular feature in detail, and interactively examine the supporting evidence, for example, tracing the local context of a genomic breakpoint.

JBrowse 2 also includes other new features such as the ability to export tracks as publication-quality SVG files; sorting, filtering, and coloring options for alignments tracks; multi-threaded rendering to accelerate the display of multiple tracks at once; and session management, so that users can easily save, restore, export, and share the state of their browser session.

The JBrowse 2 product range

JBrowse 2 is a family of several apps and modular components produced from the same codebase, specialized for different types of users. These various products are listed in Table 1.

The two most significant products are the web-based and desktop versions of JBrowse 2, known as “JBrowse Web” and “JBrowse Desktop.” The former runs on any modern web browser; the latter is compiled using the Electron framework to run on macOS, Windows, and Linux. JBrowse Desktop generally has more access to the local filesystem; user’s files can be opened as tracks and will persist across sessions. JBrowse Desktop also works without an Internet connection or behind a firewall.

Table 1 JBrowse 2 consists of multiple products, aimed at different applications but sharing a common code base

Product name	Where to find it	Brief description
JBrowse Web	https://jbrowse.org/jb2/download/	Static-site compatible app which can display multiple view types in the same session
JBrowse Desktop	https://jbrowse.org/jb2/download/	Cross-platform desktop app with ability to save user sessions to disk, and display multiple view types in the same session
JBrowse CLI	@jbrowse/cli on NPM [12]	A command line tool used for administering JBrowse Web instances
JBrowse Image CLI	@jbrowse/img on NPM	A command line tool for generating static images (SVG, PNG) of JBrowse sessions
JBrowse Embedded Components	@jbrowse/react-linear-genome-view, @jbrowse/react-circular-genome-view on NPM	Libraries that web developers can use to display JBrowse views on their website
JBrowseR [13]	JBrowseR on CRAN [14]	An R package using JBrowse Embedded components that can be used in the RStudio IDE or Shiny apps
JBrowse Jupyter [15]	jbrowse-jupyter on PyPI [16]	A Python package for JBrowse Embedded components that can be used in Jupyter Notebooks

While these apps are mostly identical in look and feel, several key operational details involving sharing and data access are different depending on whether web or desktop is being used. Unless otherwise noted, references to JBrowse 2 in this paper are inclusive of both JBrowse Web and JBrowse Desktop.

Sessions, assemblies, views, and tracks

Some of the concepts used by JBrowse 2 to organize and integrate different visualizations include *sessions*, *assemblies*, *views*, and *tracks*.

Sessions

JBrowse 2 uses the term session to represent the current state of the browser. These sessions encompass the state of all views, including the user's current location in the genome and any data they may have imported. Sessions can be saved, restored, exported, or shared with other users.

Assemblies

An assembly in JBrowse 2 refers to a sequence resource, e.g., a FASTA file, and optionally includes a list of aliases describing chromosome names that are to be treated identically, e.g., chr1 and 1. Assemblies can also contain cytoband information that is used to draw ideogram overviews. Multiple assemblies can be loaded at the same time in JBrowse Web and JBrowse Desktop, so a user can load the genome assemblies of multiple species that they want to compare, or different versions of a genome assembly of a single species.

Views

JBrowse 2 views are panels that can show data visualizations or other generic things like tabular lists. In JBrowse Web and JBrowse Desktop, the user interface is a vertical arrangement of view panels. By use of these views, different datasets can be arranged next to each other to compare different sets of data, or different visualizations of the same data.

A variety of different views are included with JBrowse 2 in order to accomplish this goal, including the traditional Linear Genome View (Fig. 1A, B), Circular View (Fig. 1C), Dotplot View (Fig. 1D), Tabular View (Fig. 1E), Linear Synteny View (Fig. 1F), and other composite views (Fig. 1G, H).

Tracks

Many JBrowse 2 views can display different genome annotation “tracks”: datasets that align in the view and can be selectively hidden, exposed, or reordered by the user. Such annotation tracks are among the earliest established user interface elements in genome browser design, implicitly present in ACeDB [19] and well-established by the time of

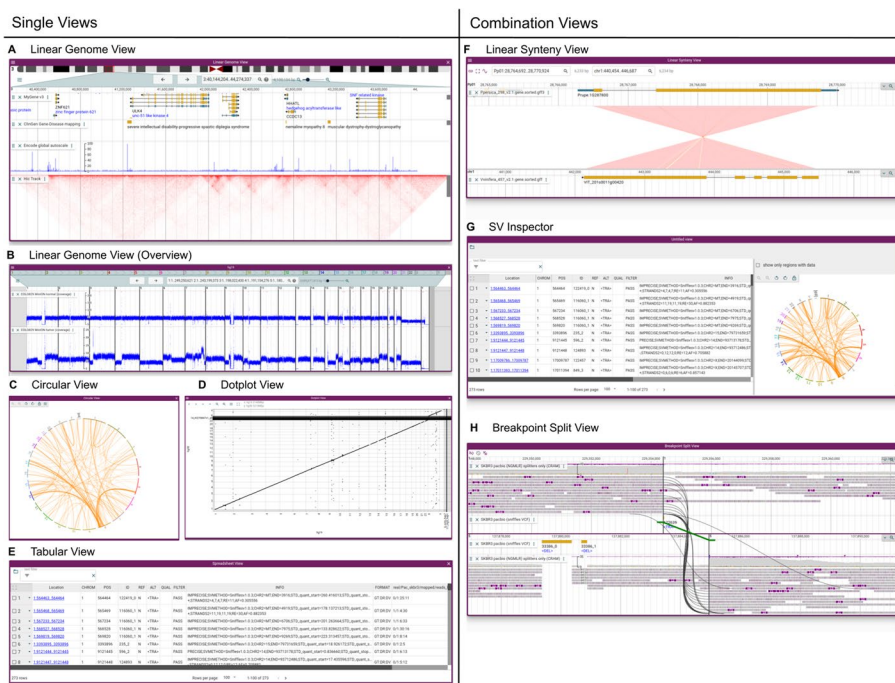


Fig. 1 JBrowse 2 integrates many views into a single application. **A** The Linear Genome View displaying gene annotations, quantitative signals, and a Hi-C track. **B** The Linear Genome View can provide a whole genome overview, here showing tumor vs normal sequencing coverage in the COLO829 cell line [17]. **C** The Circular View gives an overview of long-range relationships within and between chromosomes (here, they are translocations in an SKBR3 cancer genome). **D** The Dotplot View shows relationships between two sequences (in this case the relationship between hg19 and hg38 human genomes). **E** The Tabular View summarizes features in a sortable, filterable list, showing in this example the SKBR3 variant calls from Sniffles. **F** The Linear Synteny View shows relationships between two genomes (in this case, peach and grape) each of which is rendered using a Linear Genome View. **G** The SV Inspector allows inspection of structural variants by combining a Tabular View and a Circular View; here, both the Tabular and Circular views are visualizing a VCF file of translocations in an SKBR3 cancer genome called using Sniffles [18]. **H** The Breakpoint Split View shows events such as gene fusions and translocations (in this case, in the SKBR3 cancer genome) by aligning two Linear Genome Views and tracing the split or paired read mappings across the two views

GBrowse [1] and the UCSC Browser [20]. A list of track types that are available are listed in Table 2. Tracks can be toggled using the track selector widget.

The Linear Genome View

The Linear Genome View is the primary view in JBrowse 2, and the most similar in look and feel to JBrowse 1, GBrowse and the UCSC Genome Browser. This view shows genome annotation tracks and other genome-mapped data in a horizontally scrollable panel. A screenshot of the Linear Genome View, annotated with key user interface elements, is shown in Fig. 2.

At the top of the Linear Genome View is the navigation bar (Fig. 2 (E–J)). Key elements of this area are the currently selected reference sequence, shown either as a ruler or as an ideogram (Fig. 2 (E)); navigational controls for panning (Fig. 2 (G)) and zooming (Fig. 2 (J)); and a location display that doubles as a text search box (Fig. 2 (H)).

Beneath the navigation bar is the area where annotation tracks are shown (Fig. 2 (K–R)). This area has ruled vertical lines to help see where features are aligned. Handles on the track allow them to be vertically resized, reordered by drag-and-drop, or closed; a track menu exposes more display options and track metadata.

The track selector

The track selector (Fig. 2 (S–Y)) can be used to add or remove new tracks to the current view using a check box. The track selector is associated with one particular view at any given time, so if there are two Linear Genome Views open (e.g., one for the grape

Table 2 The list of available track types in JBrowse 2, which are specialized to render different kinds of data from various sources or file formats. Some of the tracks can be used in multiple view types as well

Track type	Appears in	Function	Supported file types
Quantitative Track	Linear Genome View	Displays dense, continuous, quantitative data	BigWig, GC content (from sequence files), GWAS scores (from BED files)
Synteny Track	Dotplot View, Linear Synteny View	Displays alignments between different genome assemblies	PAF [21], delta from MUMmer [22], mashmap.out files [23], chain (UCSC), MCScan. anchors files [24]
Alignments Track	Linear Genome View	Displays a combination of a pileup and a coverage visualization of alignments	BAM, CRAM
Hi-C Track	Linear Genome View	Displays Hi-C contact matrix	.hic files, generated by Juicebox [25]
Variant Track	Linear Genome View, Circular View	Displays feature glyphs corresponding to variants; specialized feature details panel show all genotypes in multi-sample VCF	VCF (plaintext or tabix)
Feature Track	Linear Genome View	Displays feature glyphs corresponding to genome annotations, e.g. genes	GTF (plaintext), GFF3 (tabix or plaintext), BigBed, BED (tabix or plaintext), features from REST APIs, etc
Reference Sequence Track	Linear Genome View	Displays a reference/assembly sequence and a three-frame translation	FASTA (indexed FASTA or bgzipped indexed FASTA), TwoBit (.2bit)

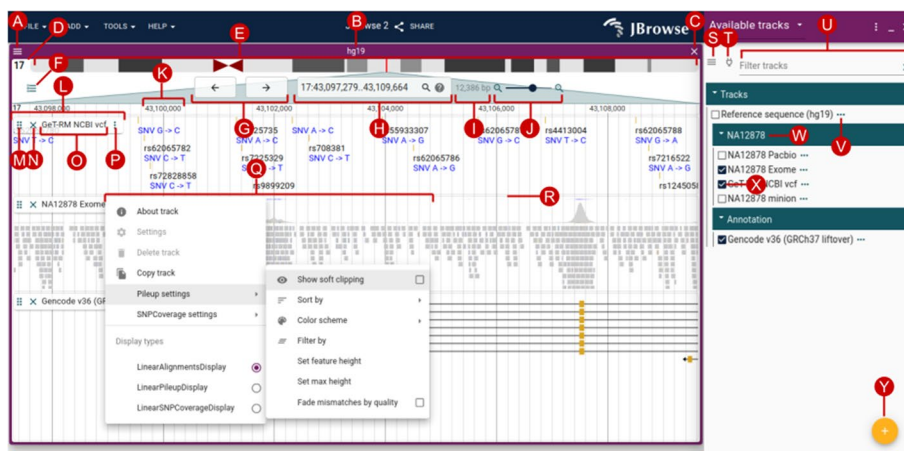


Fig. 2 The Linear Genome View is the core view of JBrowse, allowing flexible and interactive examination of a genome sequence and its annotations. The user interface elements annotated on this diagram include (A) view menu, (B) view name, (C) close view button, (D) reference sequence name, (E) reference sequence overview with optional ideogram, (F) open track selector button, (G) pan buttons, (H) location and search box, (I) view size, (J) zoom buttons and slider, (K) major ruler coordinates, (L) track label, (M) track drag handle, (N) track close button, (O) track name, (P) track menu button, (Q) track menu, (R) track resize handle, (S) track selector menu button, (T) connection menu button, (U) track selector filter, (V) track configuration menu button, (W) collapsible category label, (X) track select box, and (Y) add track/connection button

genome and one for peach genome), the tracks they display can be configured independently. Furthermore, track selectors are not solely associated with Linear Genome Views; they can also be associated with some of the other views described in later sections, such as the Dotplot View and the Linear Synteny View. For each associated view, the track selector will display tracks relevant to that particular type of view; for example, the track selector for the Dotplot View will only display tracks that are relevant to dotplots.

Beyond the linear genome view

Complementing the Linear Genome View, several alternate views show different kinds of annotated data, including inter-sequence relationships and large-scale variation. JBrowse 2 provides some mechanisms that link these different views together to facilitate navigation between them; first, through generic inter-view navigation menus (automatically constructed to link alternate views compatible with the same kind of data), and second, through specifically tailored user interface features. For example, right-clicking in an alignments track opens a menu that can launch a Dotplot or Linear Synteny View, as in Fig. 7; clicking and dragging a region in a Dotplot View will launch a Linear Synteny View; clicking on breakpoints in the SV inspector will launch the Breakpoint Split View, and so on.

Displaying and comparing multiple assemblies

JBrowse 2 features several specialized synteny views, including the Dotplot View and the Linear Synteny View. These views can display data from Synteny Tracks, which themselves can load data from formats including MUMmer [22], minimap2 [21], MashMap [23], UCSC chain files [26], and MCSan [24].

The Dotplot View (Fig. 3) can be used at different zoom scales to display whole-genome overviews of synteny, close-ups of individual syntenic regions, and even individual long reads aligned to the reference sequence (see the “[Visualizing long reads](#)” section). Users can click and drag on the Dotplot View to open a Linear Synteny View of the region.

The Linear Synteny View (Fig. 4A) shows two linear genome view panels stacked vertically. This feature allows users to view Synteny Tracks, representing regions of similarity between two different assemblies. The top and bottom panels are each fully featured Linear Genome Views, to which annotation tracks can be independently added. In addition, by exploiting the feature of the Linear Genome View whereby discontinuous regions can be shown, the user can view distal gene duplications within the Linear Synteny View (Fig. 4B).

Displaying structural variation

Structural variants can be classified into simple types (e.g., duplications, inversions) and more complex types arising from combinations of the simpler ones. The visualization of such SVs is challenging because the derived genome (e.g., the genome incorporating the structural variant) may be significantly different from the reference

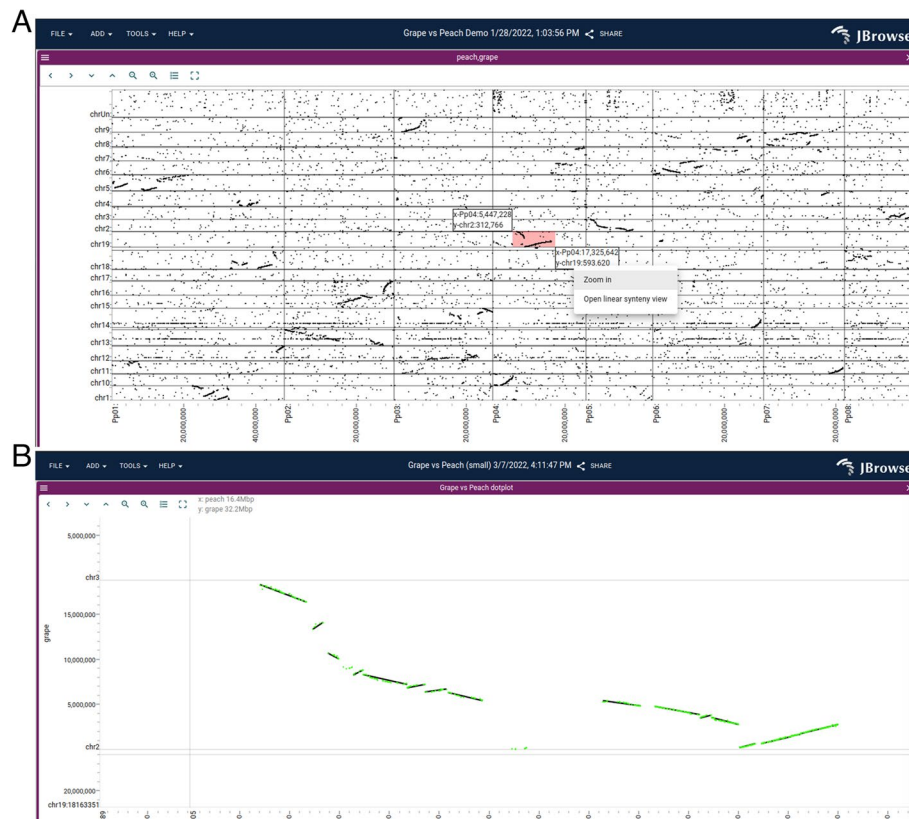


Fig. 3 **A** A dotplot showing a whole-genome alignment of the grape vs peach genome, computed by minimap2 and loaded in PAF format, reveals a large-scale syntenic structure. The user can click and drag on this view, highlighting an area shown by the small pink rectangle, to open up a detailed view (**B**) showing multiple syntenic tracks, with individual gene pairs (green) and larger syntenic blocks (black) from MCScan



Fig. 4 **A** The Linear Synteny View comparing the grape and peach genomes using data from MCSScan reveals a complex rearrangement. **B** A close-up view of a gene duplication visualized with the Linear Synteny View, with discontinuous regions (chr3 and chr4) displayed side by side on the bottom panel

genome. As a result, it is often appropriate to use different visualization modalities depending on the type of SV.

Overviews of structural variation The availability of multiple view types can help users visualize SVs through different lenses. For example, whole-genome overviews are often helpful to visualize large-scale patterns of structural variation. The Linear Genome View can be used to get a quick visualization of copy number variation by visualizing read depth from BigWig files representing genome sequencing coverage, employing its facility to display multiple chromosomes side-by-side to get a whole-genome overview (Fig. 1B).

Users can also apply one of the specialized JBrowse 2 views designed for whole-genome or multi-genome overviews. These include the Circular, Tabular, and SV inspector views.

The Circular View displays annotations in a circular format as popularized by Circos [8]. Because of its compact arrangement, this circular view is beneficial for exploring long-range structural variations encoded as breakends or translocations in VCF files [27], BEDPE files, or STAR-fusion [28] results.

The Tabular View is different from other views described in that it is a textual list of features rather than a graphical visualization. The table columns show key fields from the variant file, such as the type of SV, the location, and the ID. Controls in each column allow the tabular view to be filtered or sorted to drill down into the variant list.

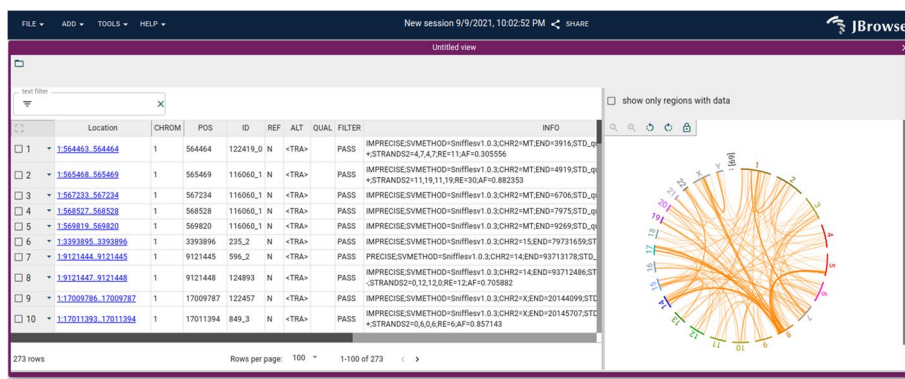


Fig. 5 The SV Inspector showing structural variants in the SKBR3 breast cancer long read dataset. The SV Inspector places the Circular and Tabular views side-by-side. On the left, the Tabular view can be filtered using simple text expression filters (text box at top left), or column filters (controls in each column header). The results of filtering are reflected in the circular view at right

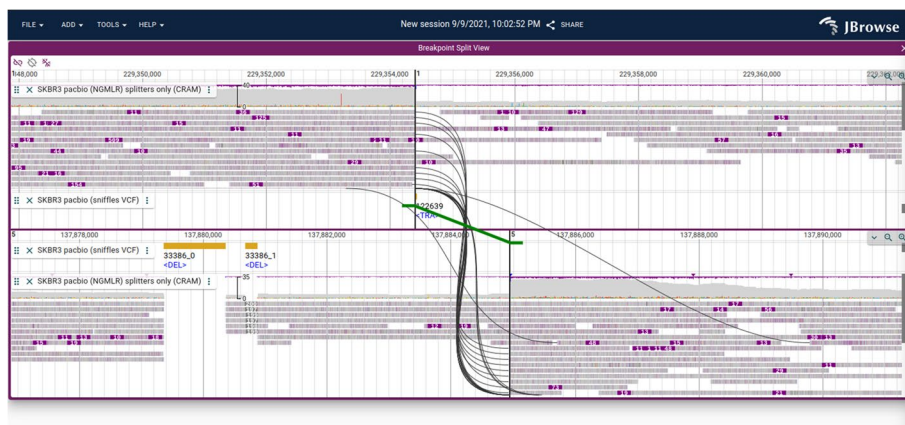


Fig. 6 A Breakpoint Split View showing a chromosomal translocation connecting chr1 and chr5 in the SKBR3 breast cancer cell line. The split long read alignments are connected using curved black lines, and the variant call itself is shown with a green line with directional “feet” showing which sides of the breakpoint are joined

The SV Inspector (Fig. 5) combines the Circular View and the Tabular View to allow users to prioritize their structural variants. This composite view was developed to address a common workflow in cancer bioinformatics research: examining a list of putative variants to visually evaluate them in the context of relevant information such as canonical gene models, RNA-seq results, chromatin interactions, or other genome annotation data. The SV Inspector includes a Tabular View of a set of candidate structural variants with controls to mark features for later inspection and a Circular View visualizing where these variants lie in the genome. Each variant is presented as a row in the tabular view and as a chord in the circular view. Clicking on a chord in the Circular View or a row in the Tabular View launches a Breakpoint Split View (Fig. 6) showing the read evidence for a selected structural variant.

Fine detail of structural variation To visualize a single breakpoint, such as a gene fusion and the evidence for that breakpoint, we introduce the Breakpoint Split View.

The Breakpoint Split View consists of two Linear Genome Views stacked vertically. The power of this view lies in its ability to visualize genomic evidence for structural variants between discontinuous regions of the genome (Fig. 6). The read evidence for the structural variant is shown using curved black lines for long split alignments or paired-end reads. The structural variant call itself is shown using green lines, based on information from breakends [27] or translocation type features from VCF files.

Visualizing long reads

Long-read sequencing technology, such as the platforms developed by Pacific Biosciences and Oxford Nanopore Technologies, has proven useful in the resolution of haplotypes, structural variants, and complex repetitive regions. JBrowse 2 includes several features to highlight the information contained in long reads, including advanced alignments track features to sort, filter, and color reads and a “read vs reference” feature that enables users to view alignments of long reads in a Dotplot View or Linear Synteny View (Fig. 7).

In addition to elucidating long-range structure, long-read sequencing platforms can provide a direct readout of chemical modifications such as methylation on DNA and RNA sequences. The modifications can be called by tools such as nanopolish [29] and primrose [30], which stores this information in the MM tag in BAM/CRAM files [31]. JBrowse 2 can then use the MM tag to render the positions of these modifications on individual reads.

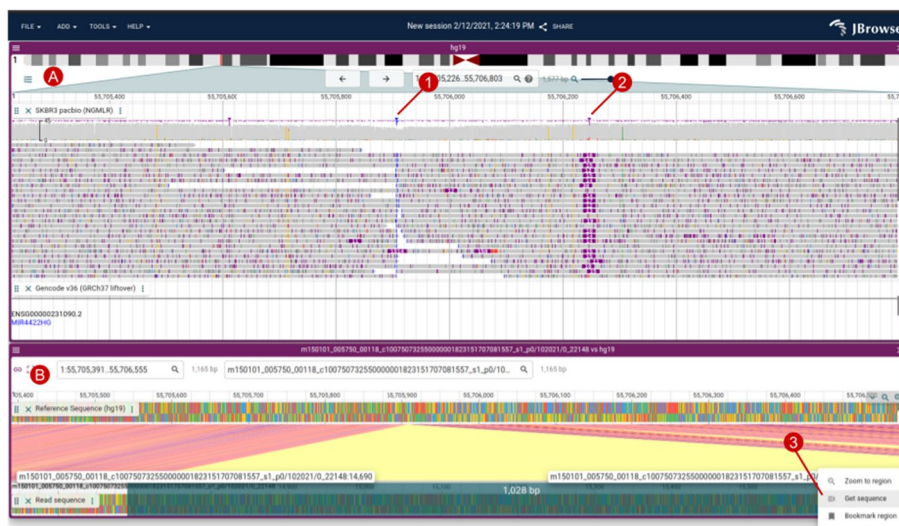


Fig. 7 **A** An alignments track showing SKBR3 PacBio alignments, with a large (> 1000 bp) insertion highlighted by soft-clipped reads in blue (1) and a smaller insertion in purple (2). **B** The “Read vs ref” view created by right-clicking a read in the Alignments Track creates a Linear Synteny View comparing the read vs the reference genome, which allows you to see the inserted non-reference bases easily. Users can also select regions on the read or reference sequence and select “Get sequence” (3)

Ways to access data

Connections

JBrowse 2 allows users to create “[Connections](#)” to sets of tracks or assemblies that are understood to be managed outside of the JBrowse instance. JBrowse 2 includes two types of Connections by default: JBrowse 1 Connections, which allow viewing tracks from a JBrowse 1 installation on the web, and UCSC Track Hub Connections, which allow viewing tracks in a UCSC Track Hub [\[32\]](#). To help users navigate the latter type of Connection, JBrowse 2 includes an interface for browsing the UCSC Track Hub Registry [\[33\]](#).

Direct access to local files

JBrowse Web and JBrowse Desktop allow users to open tracks directly from a user’s local filesystem. This functionality keeps the data private on the user’s computer. On JBrowse Web, due to the limitations of web browsers, local files must be re-opened when the page refreshes or a session is re-opened. JBrowse Web provides a message to alert users to this necessity. This limitation does not exist on JBrowse Desktop.

Sharing sessions

In JBrowse Web, users can share sessions with other users by generating a share link, which produces a shortened URL containing the contents of the user session. Visiting the link will restore all the same views at the same locations, with the same tracks displayed. This also includes track data that a user has added to the session: if a user opens a track in their session that references a remote file, then their share link will include this track.

Authentication

JBrowse 2 natively supports Google Drive, Dropbox, and HTTP Basic authentication. Plugin developers can extend this to connect the genome browser to any application that needs authentication to access data, as described in the section titled “[Extending JBrowse 2 with Plugins](#).”

Performance and scalability

JBrowse 2 is structured to take heavy computations off the main thread using remote procedure calls (RPC). In JBrowse Web and JBrowse Desktop, we use web workers to handle RPCs, which perform data parsing, rendering, and other computationally time-consuming tasks in a separate thread. This approach allows the app to remain responsive to the user even when displaying large datasets or multiple tracks. Note that JBrowse Embedded does not use web workers currently, so it is single threaded, but may gain web worker support in the future.

To profile the end-to-end performance of loading and rendering tracks, we used Puppeteer [\[34\]](#) to run JBrowse 2 (both with parallel rendering enabled and disabled), JBrowse 1, and igv.js. Each tested browser was given the task of rendering BAM and CRAM files containing long and short reads at varying coverage. Full details of the

benchmark can be found under “Performance and scalability benchmark details” in the “Methods” section.

When rendering a single track, JBrowse 2 has comparable performance to igv.js [35], as shown in Fig. 8. However, when rendering multiple tracks, the parallel rendering strategy in JBrowse 2 can improve performance (Fig. 9). The parallel strategy also yields a more responsive user interface, since the main thread (whose frame rate determines the apparent speed with which the browser responds to user input) does not become tied up by rendering and data parsing (Fig. 10). Some of the slower performance of JBrowse 2 observed in the performance profiling is due to its relatively slower startup time, which includes starting up the main thread and web worker threads, an area which may be amenable to further optimization.

Administration and configuration

Extending JBrowse 2 with plugins

JBrowse 2 has a plugin system which provides developers with the ability to customize and extend JBrowse to suit the specialized needs of the organization or researcher. Table 3 describes the elements that can be extended via plugins, such as data adapters, track types, and view types.

JBrowse Web and JBrowse Desktop feature an in-app Plugin Store where users can install plugins. Examples of third-party plugins that can be installed include a multiple sequence alignment viewer, an ideogram viewer with Reactome [36] pathway visualization (Fig. 11), and plugins that provide data adapters for fetching data from the mygene.info [37] and the CIVIC API [38].

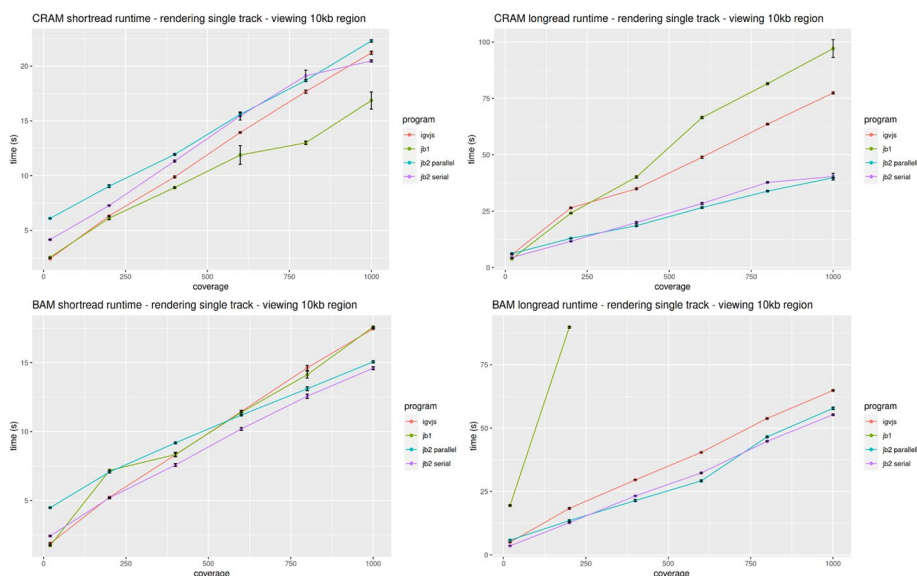


Fig. 8 JBrowse 2’s performance is comparable to igv.js and significantly exceeds JBrowse 1’s performance on large long read datasets, as reflected in these benchmarks rendering aligned reads of varying coverage and file formats in a 10-kb region. The incomplete data for JBrowse 1 on the BAM long-read benchmark reflects the fact that JBrowse 1 times out this benchmark (i.e., its rendering time exceeds 5 min). Full details of the benchmark can be found under “Performance and Scalability benchmark details” in the “Methods” section

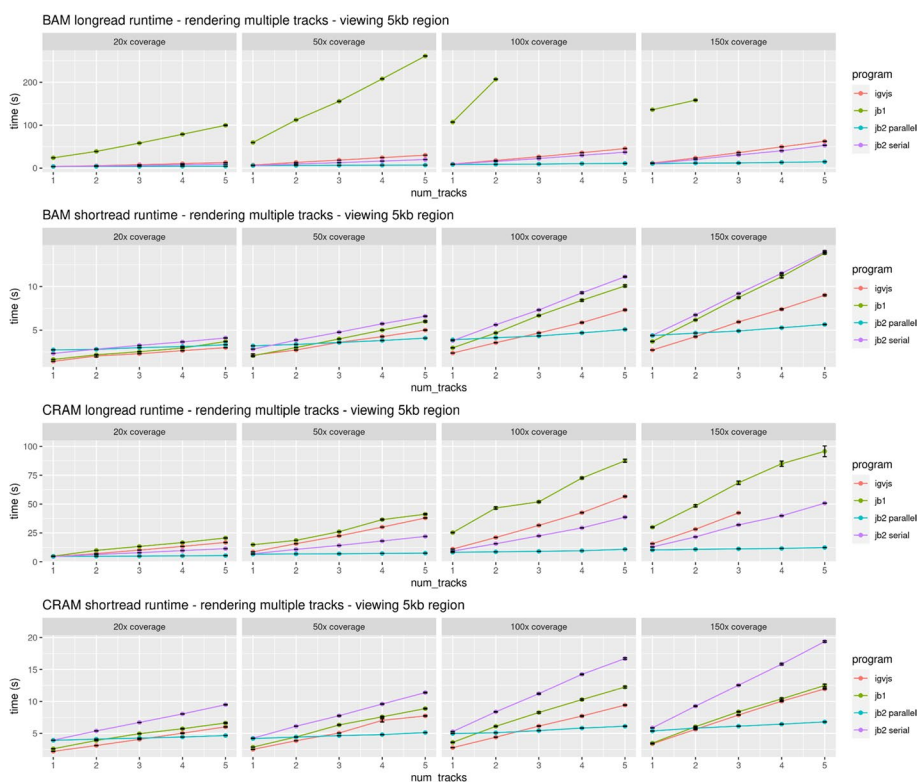


Fig. 9 When displaying multiple tracks at once, JBrowse 2’s parallel rendering strategy shows significant gains compared to single-threaded (serial) strategies, as reflected in these benchmarks rendering aligned reads of varying coverage and file formats in a 5-kb region. The incomplete data for JBrowse 1 and igv.js on some of the benchmarks reflects the fact that these apps time out our benchmark under some circumstances (i.e., rendering time exceeds 5 min). Full details of the benchmark can be found under “Performance and Scalability benchmark details” in the “Methods” section

Static site compatibility

JBrowse 2 is a “static site” compatible application: since all data parsing and rendering occurs on the client, its ongoing deployment does not require any server-side code beyond a basic web server. Static sites are low-cost because they can be hosted on inexpensive or free hosting services like Amazon S3 and Github pages. In addition, many security issues are mitigated, and the sites may require less maintenance.

JBrowse CLI

The JBrowse CLI is an administrative tool that can load assemblies, tracks, and indices for gene name searching. The JBrowse CLI tool can be installed from NPM and runs on both Unix-like and Windows systems, somewhat more portably than JBrowse 1, which required Perl scripts whose installation on Windows-like systems was more involved. The JBrowse CLI also includes an admin-server command which allows changes made in the web GUI to be persisted to the config file on disk.

Text indexing

The JBrowse CLI includes a text indexing command that creates trix formatted indices [40]; these indices are easier to manage than the index files used by JBrowse 1 (which

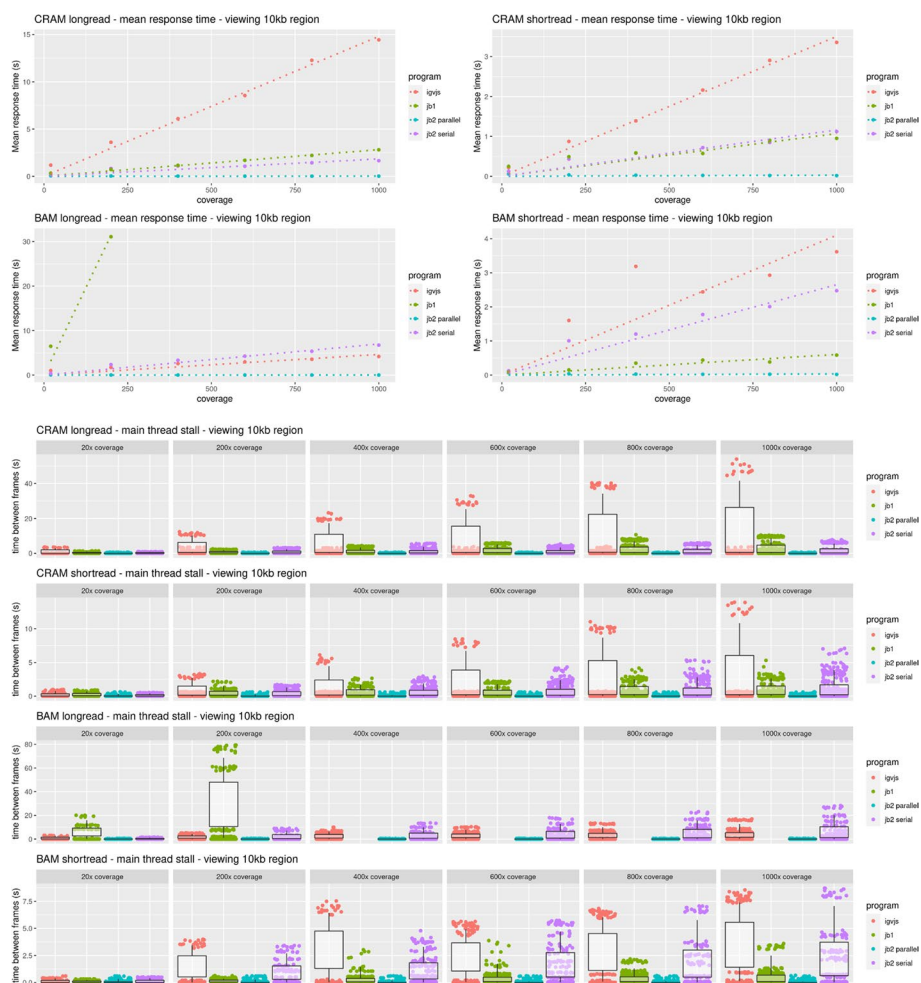


Fig. 10 JBrowse 2's parallel rendering strategy yields significant improvements in user interface responsiveness, as reflected in these benchmarks rendering aligned reads of varying coverage and file types in a 10-kb region. We define the *response time* as the delay, during the rendering phase of the benchmark, from a randomly sampled time point until the time the next frame is rendered. This directly reflects the perceived delay between when a user initiates an action and when the app responds. The response time is a random variable; its expectation gives a sense of average lag, while its variation gives a sense of how unpredictable the user interface delays can be. Panel **A** plots the expectation of the response time as a function of sequencing coverage. At high coverage, JBrowse 2's parallel strategy maintains a low response time, in contrast to single-threaded strategies whose response time can grow large, with the perception that the browser is “hanging” or “frozen.” The relationship between response time and coverage is approximately linear for all browsers, as shown by the dotted linear regression fit. The incomplete data for JBrowse 1 on the BAM long read benchmark reflects the fact that the simulation times out (rendering time > 5 min). Panel **B** shows the same data plotted as a scatterplot of time between frames. The plotted points show the raw time between frame values and are overlaid with boxplots that show the variation in response times (25th and 75th percentiles shown in the boxes, 5th and 95th percentiles shown in the tails). Full details of the benchmark can be found under “[Performance and Scalability benchmark details](#)” in the “[Methods](#)” section

consisted of an on-disk hash table built from many small files). The text indexing in JBrowse 2 allows for either per-track indexes or aggregate indexes containing data from multiple tracks. The tool can index gene IDs, full-text descriptions, or other arbitrary data fields from GFF and VCF files. Text searching can also be extended using plugins to adapt to custom search systems.

Table 3 A listing of JBrowse 2 elements that can be extended by third-party plugins

Plugin-extensible element	Description	Example
Data adapters	Classes through which reading and parsing unique data formats is done, including retrieving data from RESTful APIs	BamAdapter: processes bam files, remote or local, along with their coordinating.bai or.csi index files
Text search adapters	Classes through which searches for features by name are processed	TrixTextSearchAdapter: used for UCSC trix indexes, generated by the jbrowse text-index command
Track types	A high-level concept in the configuration system that associates a name, trackId, and metadata with a data adapter. There is not a lot of logic attached to track types; instead, the display types and renderers are used to draw and add logic to tracks	AlignmentsTrack: displays data typically associated with BAM and CRAM type data adapters
Display types	The code to “display” a track type in a particular view. This layer is important because it allows track types to be displayed in multiple view types, and often contains logic such as menus, click actions, React components, and more	LinearSyntenyDisplay, DotplotDisplay: these display types help display a SyntenyTrack in different view types
Renderer types	Fetches data from a data adapter and draws the features, typically rendering to either SVG or HTML5 canvas. Renderers run on the remote side of an RPC call and are instructed to render a particular genomic region or regions	PileupRenderer: draws the reads from a.bam or.cram file
Widgets	User interfaces that provide utility or information to the user. In JBrowse Web and JBrowse Desktop, these appear as a side drawer. In embedded, these appear in a dialog box	Track selector: Provides a list of tracks for user to toggle in the user interface
RPC calls	Custom code that a plugin runs on the remote side of an RPC call (e.g., in a web worker) to avoid doing heavy work in the main thread	WiggleGetMultiRegionStats: estimates quantitative statistics for the given genomic regions
View types	Containers for visualizations that permit wholly unique visualizations to be displayed in the same context as the standard linear genome view	CircularView: provides a circular whole-genome overview of structural variants
Extension points	A named list of callbacks that the code can use to create arbitrary modifications to various parts of the code	The ‘extendPluggableElement’ extension point allows plugins to add to state tree models e.g., add menu items
Connection types	Connections provide a way to connect to a provider of tracks or assemblies	UCSCTrackHubConnection: adds tracks from a UCSC track hub to the tracklist
Internet account type	Adds a custom authentication method to a remote data provider	GoogleDriveOAuthInternetAccount: Adds ability to open share links from Google Drive

Discussion

In this paper, we have introduced JBrowse 2. This document is not intended to recap the complete JBrowse 2 user guide; instead, it covers foundational concepts (sessions, assemblies, views, tracks, connections, and plugins), the various views (linear genome view, other basic views, composite views, and overviews), and the data access modalities, with a focus on use cases involving comparative genomics and structural variation. We have also briefly introduced some technical details (including

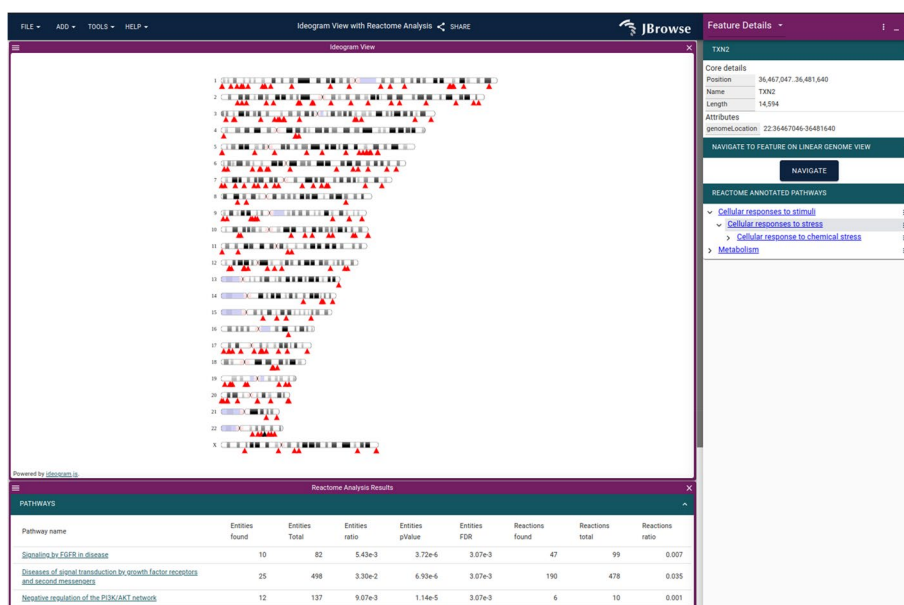


Fig. 11 The ideogram view showing the Reactome pathway analysis on a gene list using the JBrowse 2 Ideogram plugin, which uses ideogram.js [39]

administration, performance engineering, and plugin development) and deployment on non-web platforms, including R, Jupyter/Python, and desktop.

In large part, the potential of JBrowse 2 arises from its portability, extensibility, and customizability. It can run as a desktop application for personal use, as a web app to support research groups and communities, as a visualization component within Jupyter Notebooks or R for use by data scientists, or as a series of embedded components for developers of biomedical data portals and other integrative projects. In addition, JBrowse 2 is customizable at multiple levels, ranging from an intuitive GUI for view and track configuration, to scriptable customization via its CLI, to bespoke visualizations and user interfaces created using a developer's plugin API.

Conclusions

JBrowse 2 is a multi-purpose platform for showing genome visualizations, including but not limited to the linear genome browser view. The multiple view types available in JBrowse 2 can be incorporated into existing analysis and annotation workflows. The synteny and structural variant visualization features are designed to help users understand genome assemblies, population-level variation, or inter-species variation. Developers can extend JBrowse 2 to perform in-app analyses, create novel visualization types, and create portals for their data repositories or model organism databases. JBrowse 2 can be used as a web application, a desktop application, a CLI tool, and/or a Jupyter/R Shiny widget, making genomes and their annotations available to a wider audience.

Methods

Software methodology

The JBrowse development team follows agile practices to plan, design, and implement new software. Requests for new features and bug reports are documented using Github

issues and reviewed by the development team during backlog grooming sessions. Teammates hold pair programming sessions to discuss and review implementations. Pull requests with new features are peer-reviewed by developers to test and approve changes. A Github project board is used to track in progress issues and organize incoming ones according to priority. Finally, a test suite is run for all changes to the codebase using Github Actions. All source code for JBrowse 2 is distributed under the Apache License version 2.0 [41]. Many utility libraries and data parsers that are used by the JBrowse project are also published on NPM, which can be used independently of JBrowse itself.

Software design sprints

We held two design sprints focused on specific use cases of JBrowse 2, collaborating with members of the existing JBrowse 1 user community and other selected groups. The first took place at the Ontario Institute for Cancer Research in Toronto in 2019, focusing on prioritization of structural variants. The second took place remotely over Zoom in 2020 and focused on comparisons between genomes. Both of these sprints approximately followed the Google Ventures design sprint model. For a week, working in small teams, we conducted expert interviews, mapped out user journeys, sketched out competing solutions, built prototypes, and then presented these prototypes to users for testing.

Implementation details

JBrowse 2 relies on several technological innovations that were not available when JBrowse 1 was created. JBrowse 2 uses TypeScript [42], which adds compile-time type checking to JavaScript. JBrowse 2 also uses React [43] for rendering the user interface and mobx-state-tree [44], which provides a centralized way of storing, accessing, and restoring the application state. In addition, JBrowse Web and JBrowse Desktop use Web Workers [45] to enable parallel processing and rendering of data tracks. JBrowse Desktop is built using Electron [46], a system for building cross-platform desktop applications.

Performance and scalability benchmark details

For performance profiling, we generated reads from an arbitrary region chr22:25,000,000–25,250,000 on hg19. We simulated 50-kb long reads to a coverage of $\sim 1000\times$ using pbsim2 v2.0.1 (“pbsim ref.fa –depth 1000 –hmm_model data/R103.model –length-mean 50,000 –prefix 1000x”) and also simulated 150-bp paired-end short reads to a coverage of $\sim 1000\times$ using wgsim v1.15.1 (“wgsim -1 150 -2 150 -N 1,000,000 hg19mod.fa 1000x.1.fq 1000x.2.fq”). We aligned the reads to the genome using minimap2 (2.24-r1122) and subsampled the resulting different coverages using samtools (1.15.1). We instrumented igv.js to output a console log when it completed rendering. We then compared the timing of igv.js (v2.12.1) with JBrowse 2 Web, referred to as “jb2 parallel” (v2.4.0); JBrowse 2 Embedded, referred to as “jb2 serial” (v2.4.0); and JBrowse 1, referred to as “jb1” (v1.16.11). The benchmarking script uses puppeteer v3.16.0 [34] and measures the time taken to complete the rendering of a track. Each step is run $N=10$ times, and the mean time with standard error bars ($\pm \sigma/\sqrt{N}$) is plotted. The frame rate is calculated using the requestAnimationFrame API. A reproducible benchmark script is available at <https://github.com/GMOD/jb2profile>. Note that JBrowse 2 Embedded can also use parallel rendering, but is tested in serial for the purposes of demonstration.

Supplementary Information

The online version contains supplementary material available at <https://doi.org/10.1186/s13059-023-02914-z>.

Additional file 1. Additional performance profiling, derivation of formula for performance profiling time between frames metric, and list of data parser libraries on NPM.

Additional file 2. Review history.

Acknowledgements

The expert interviews in the design sprints directly informed the software presented here. We especially wish to thank Jonathan Torchio, Jared Simpson, Fabien Lamaze, and Heather Gibling of OICR, Viswateja Nelakuditi and Hiromichi Suzuki of SickKids (The Hospital for Sick Children), Michael Schatz and Michael Alonge of Johns Hopkins University, and Xingang Wang of Cold Spring Harbor National Laboratory for their participation and invaluable feedback in design sprints. We also want to thank all contributors of code, feedback, bug, and feature requests on the project's GitHub pages.

Peer review information

Kevin Pang was the primary editor of this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Review history

The review history is available as Additional file 2.

Authors' contributions

CD, GS, PX, TDJM, EH, JZ, CB, GH, AD, and RB formed the core software development team (led by RB) and worked on all areas of the project. AL contributed design elements and organized design sprints. EG worked on JBrowse Jupyter. MM, TF, and BB worked on text indexing. RH and SC worked on outreach. LDS and IHH led the project including the preparation of manuscripts and grant proposals and overall direction. The authors read and approved the final manuscript.

Funding

The JBrowse project was supported by NIH grant R01HG004483. Apollo is supported by NIH grant R01GM080203. Features targeted toward the visualization of structural variants in cancer were also supported by NIH grant U24CA220441. Text searching functionality was also supported by NIH grant R24OD021324.

Availability of data and materials

The datasets analyzed during the current study are listed below, with links from which they can be downloaded and citations to the primary scientific publications associated with the data.

- The grape genome (Vvinifera_145_Genoscope.12X.fa) and annotations (Vvinifera_457_v2.1.gene.gff3) are available from Phytozome https://phytozome-next.jgi.doe.gov/info/Vvinifera_v2_1 [47].
- The peach genome (Ppersica_298_v2.0.fa) and annotations (Ppersica_298_v2.1.gene.gff3) are available from https://phytozome-next.jgi.doe.gov/info/Ppersica_v2_1 [48].
- The SKBR3 breast cancer cell line PacBio sequencing and Illumina sequencing are available from <http://schatz-lab.org/publications/SKBR3/> [49].
- The COLO829 melanoma cancer cell line Nanopore sequencing is available from ENA PRJEB27698 [17].
- Whole genome alignment of grape vs peach generated with minimap2 2.24-r1122 ("minimap2 -c Vvinifera_457_Genoscope.12X.fa.gz Ppersica_298_v2.0.fa") [21]
- Gene based alignments for grape vs peach generated with MCScan "python -m jcv.compara.catalog ortholog grape peach --no_strip_names following" using guide from <https://github.com/tanghaibao/jcvi/wiki/MCscan-%28Python-version%29> [24].

The source code of JBrowse 2 is available at <https://github.com/GMOD/jbrowse-components> [50] and is available under the Apache 2.0 license. An archival copy of the source code is available on Zenodo doi:10.5281/zenodo.7710472 [51]. Installation guides and documentation are available at <https://jbrowse.org/jb2/docs>.

A list of live demos associated with the figures in this paper is available at <https://jbrowse.org/demos/paper2022/>.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Received: 24 August 2022 Accepted: 20 March 2023

Published online: 17 April 2023

References

1. Stein LD, Mungall C, Shu S, Caudy M, Mangone M, Day A, et al. The generic genome browser: a building block for a model organism system database. *Genome Res.* 2002;12(10):1599–610.
2. Skinner ME, Uzilov AV, Stein LD, Mungall CJ, Holmes IH. JBrowse: a next-generation genome browser. *Genome Res.* 2009;19(9):1630–8.
3. Buels R, Yao E, Diesh CM, Hayes RD, Munoz-Torres M, Helt G, et al. JBrowse: a dynamic web platform for genome visualization and analysis. *Genome Biol.* 2016;17:66.
4. McKay SJ, Vergara IA, Stajich JE. Using the Generic Synteny Browser (GBrowse_syn). *Curr Protoc Bioinformatics.* 2010;Chapter 9:Unit 9.12.
5. Carver TJ, Rutherford KM, Berriman M, Rajandream MA, Barrell BG, Parkhill J. ACT: the Artemis comparison tool. *Bioinformatics.* 2005;21(16):3422–3.
6. Veltri D, Wight MM, Crouch JA. SimpleSynteny: a web-based tool for visualization of microsynteny across multiple species. *Nucleic Acids Res.* 2016;44(W1):W41–5.
7. Nattestad M, Aboukhalil R, Chin CS, Schatz MC. Ribbon: intuitive visualization for complex genomic variation. *Bioinformatics.* 2020;37(3):413–5.
8. Krzywinski M, Schein J, Birol I, Connors J, Gascoyne R, Horsman D, et al. Circos: an information aesthetic for comparative genomics. *Genome Res.* 2009;19(9):1639–45.
9. Markham JF, Yerneni S, Ryland GL, Leong HS, Fellowes A, Thompson ER, et al. CNspector: a web-based tool for visualisation and clinical diagnosis of copy number variation from next generation sequencing. *Sci Rep.* 2019;9(1):1–9.
10. Robinson JT, Thorvaldsdóttir H, Wenger AM, Zehir A, Mesirov JP. Variant Review with the Integrative Genomics Viewer. *Cancer Res.* 2017;77(21):e31–4.
11. Yokoyama TT, Kasahara M. Visualization tools for human structural variations identified by whole-genome sequencing. *J Hum Genet.* 2020;65(1). [cited 2022 Mar 15]. Available from: <https://pubmed.ncbi.nlm.nih.gov/31666648/>
12. Npm. [cited 2022 Apr 27]. Available from: <https://npmjs.com/>
13. Hershberg EA, Stevens G, Diesh C, Xie P, De Jesus MT, Buels R, et al. JBrowseR: an R interface to the JBrowse 2 genome browser. *Bioinformatics.* 2021;37(21):3914–5.
14. Hornik K. The comprehensive R archive network. *Wiley Interdiscip Rev Comput Stat.* 2012;4(4):394–8.
15. De Jesus Martinez T, Hershberg EA, Guo E, Stevens GJ, Diesh C, Xie P, et al. JBrowse Jupyter: A Python interface to JBrowse 2. *Bioinformatics.* 2023;39(1). [cited 2022 May 18]. Available from: <https://academic.oup.com/bioinformatics/article/39/1/btad032/6989625>
16. PyPI - the Python Package Index. PyPI. [cited 2022 Apr 27]. Available from: <https://pypi.org/>
17. Espejo Valle-Inclan J, Besselink NJM, de Bruijn E, Cameron DL, Ebler J, Kutzera J, et al. A multi-platform reference for somatic structural variation detection. *Cell Genomics.* 2022;2(6):100139.
18. Sedlazeck FJ, Rescheneder P, Smolka M, Fang H, Nattestad M, von Haeseler A, et al. Accurate detection of complex structural variations using single-molecule sequencing. *Nat Methods.* 2018;15(6):461–8.
19. Durbin R, Thierry-Mieg J. The ACEDB Genome Database. In: Suhai S, editor. *Computational Methods in Genome Research.* Boston: Springer, US; 1994. p. 45–55.
20. Kent WJ, Sugnet CW, Furey TS, Roskin KM, Pringle TH, Zahler AM, et al. The human genome browser at UCSC. *Genome Res.* 2002;12(6):996–1006.
21. Li H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics.* 2018;34(18):3094–100.
22. Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, et al. Versatile and open software for comparing large genomes. *Genome Biol.* 2004;5(2):1–9.
23. Jain C, Dilthey A, Koren S, Aluru S, Phillippy AM. A fast approximate algorithm for mapping long reads to large reference databases. *J Comput Biol.* 2018;25(7):766–79.
24. Tang H, Bowers JE, Wang X, Ming R, Alam M, Paterson AH. Synteny and collinearity in plant genomes. *Science.* 2008;320:486–8. <https://doi.org/10.1126/science.1153917>.
25. Durand NC, Robinson JT, Shamim MS, Machol I, Mesirov JP, Lander ES, et al. Juicebox Provides a Visualization System for Hi-C Contact Maps with Unlimited Zoom. *Cell Syst.* 2016;3(1):99.
26. Chain Format. [cited 2022 Apr 28]. Available from: <https://genome.ucsc.edu/goldenPath/help/chain.html>
27. The Variant Call Format Specification - VCFv4.3 and BCFv2.2. 2022 [cited 2022 May 12]. Available from: <https://samtools.github.io/hts-specs/VCFv4.3.pdf>
28. Haas BJ, Dobin A, Stransky N, Li B, Yang X, Tickle T, et al. STAR-Fusion: Fast and Accurate Fusion Transcript Detection from RNA-Seq. *bioRxiv.* 2017. p. 120295. [cited 2022 Mar 8]. Available from: <https://www.biorxiv.org/content/10.1101/120295v1.abstract>
29. Loman NJ, Quick J, Simpson JT. A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nat Methods.* 2015;12(8):733–5.
30. primrose: Predict 5mC in PacBio HiFi reads. Github; [cited 2022 Apr 28]. Available from: <https://github.com/PacificBioSciences/primrose>
31. HTS format specifications. [cited 2022 Apr 28]. Available from: <https://samtools.github.io/hts-specs/>
32. Raney BJ, Dreszer TR, Barber GP, Clawson H, Fujita PA, Wang T, et al. Track data hubs enable visualization of user-defined genome-wide annotations on the UCSC Genome Browser. *Bioinformatics.* 2013;30(7):1003–5.
33. The Ensembl Core Team. The Track Hub Registry. [cited 2022 Apr 27]. Available from: <https://www.trackhubregistry.org/>
34. puppeteer: Headless Chrome Node.js API. Github; [cited 2022 Apr 27]. Available from: <https://github.com/puppeteer/puppeteer>
35. Robinson JT, Thorvaldsdóttir H, Turner D, Mesirov JP. igv.js: an embeddable JavaScript implementation of the Integrative Genomics Viewer (IGV). *Bioinformatics.* 2022;39(1). Available from: <https://academic.oup.com/bioinformatics/article/39/1/btac830/6958554>
36. Gillespie M, Jassal B, Stephan R, Milacic M, Rothfels K, Senff-Ribeiro A, et al. The reactome pathway knowledgebase 2022. *Nucleic Acids Res.* 2021;50(D1):D687–92.

37. Xin J, Mark A, Afrasiabi C, Tsueng G, Juchler M, Gopal N, et al. High-performance web services for querying gene and variant annotation. *Genome Biol.* 2016;17(1):91.
38. Griffith M, Spies NC, Krysiak K, McMichael JF, Coffman AC, Danos AM, et al. CIVIC is a community knowledgebase for expert crowdsourcing the clinical interpretation of variants in cancer. *Nat Genet.* 2017;49(2):170–4.
39. Weitz E. ideogram: Chromosome visualization for the web. Github; [cited 2022 Apr 27]. Available from: <https://github.com/eweitz/ideogram>
40. Trix Indices. [cited 2022 Apr 15]. Available from: <https://genome.ucsc.edu/goldenPath/help/trix.html>
41. Apache License, Version 2.0. [cited 2022 Apr 15]. Available from: <https://opensource.org/licenses/Apache-2.0>
42. JavaScript With Syntax For Types. [cited 2022 Apr 15]. Available from: <https://www.typescriptlang.org/>
43. React. [cited 2022 Apr 15]. Available from: <https://reactjs.org/>
44. Welcome to MobX-State-Tree!. [cited 2022 Apr 15]. Available from: <https://mobx-state-tree.js.org/>
45. Using Web Workers. [cited 2022 Apr 27]. Available from: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers
46. Electron. [cited 2022 Apr 27]. Available from: <https://www.electronjs.org/>
47. The French-Italian Public Consortium for Grapevine Genome Characterization. The grapevine genome sequence suggests ancestral hexaploidization in major angiosperm phyla. *Nature.* 2007;449(7161):463–7.
48. Verde I, Abbott AG, Scalabrin S, Jung S, Shu S, Marroni F, et al. The high-quality draft genome of peach (*Prunus persica*) identifies unique patterns of genetic diversity, domestication and genome evolution. *Nat Genet.* 2013;45(5):487–94.
49. Nattestad M, Goodwin S, Ng K, Baslan T, Sedlazeck FJ, Rescheneder P, et al. Complex rearrangements and onco-gene amplifications revealed by long-read DNA and RNA sequencing of a breast cancer cell line. *Genome Res.* 2018;28(8):1126–35.
50. Diesh C, Stevens G, Xie P, De Jesus Martinez T, Hershberg E, Leung A, et al. GMOD/jbrowse-components. Github; [cited 2023 Mar 8]. Available from: <https://github.com/GMOD/jbrowse-components>
51. Diesh C, Stevens G, Xie P, De Jesus Martinez T, Hershberg E, Leung A, et al. JBrowse v2.4.0. Zenodo; 2023. Available from: <https://zenodo.org/record/7710472>

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Ready to submit your research? Choose BMC and benefit from:

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

At BMC, research is always in progress.

Learn more biomedcentral.com/submissions

