Published online 6 October 2022

NAR Genomics and Bioinformatics, 2022, Vol. 4, No. 4 **1**
https://doi.org/10.1093/nargab/lqac077

SODA: a TypeScript/JavaScript library for visualizing biological sequence annotation

Jack W. Roddy[1,2,*], George T. Lesica[1] and Travis J. Wheeler[1,2,*]

[1]Department of Computer Science, University of Montana, Missoula, MT 59812, USA and [2]Department of Pharmacy Practice and Science, University of Arizona, Tucson, AZ 85721, USA

Received May 12, 2022; Revised August 27, 2022; Editorial Decision September 26, 2022; Accepted September 27, 2022

ABSTRACT

We present SODA, a lightweight and open-source visualization library for biological sequence annotations that enables straightforward development of flexible, dynamic and interactive web graphics. SODA is implemented in TypeScript and can be used as a library within TypeScript and JavaScript.

INTRODUCTION

Annotation of biological sequences and the visualization of these annotations is central to molecular biology. It is common to represent features (e.g. genes) annotated in the context of a reference sequence (e.g. a genome) using a combination of glyphs (rectangles, lines, arrows, etc.) and plots (line plots, bar plots, heatmaps, etc.). Countless software tools have been developed that produce such visualizations, with modern applications typically developed using web technologies. Most prominent among these are genome browsers such as JBrowse (1), and hosted services such as Ensembl (2) and the UCSC genome browser (3). Recent innovations include the grammar-based JavaScript toolkit for visualizing genomics data, Gosling (4). Libraries also exist for visualizing annotation specific to protein sequences, such as ProtVista (5) and its successor Nightingale (6). In order to constrain the scope of this application note, we refrain from an extensive comparison of alternative browsers and visualization libraries.

These frameworks provide extensive visualization functionality, but may not meet the needs that arise in the development of custom visualization systems. Custom visualizations may require visual features or data types not supported by out-of-the-box solutions, may involve interactive response to user actions that exceed the options provided by these frameworks, and may demand integration of multiple visual facets into a coherent whole. Furthermore, the use of standardized browsers may impose hefty dependencies and inclusion of unwanted GUI components. To avoid these compromises, developers of custom visualization systems often turn to core web browser technologies (HTML, CSS, SVG, Canvas, WebGL), or libraries that lightly abstract over those core technologies (D3 (7), PixiJS (8)).

These developers would be well served by a front-end software library that enables extensive control over the form and function of annotation visualization elements useful for representing genome annotations, while further abstracting over most of the complexity of low-level libraries. To our knowledge, there exists no lightweight web-based library that meets this need. Here, we present SODA (Soda Obediently Draws Annotations), an open-source TypeScript/JavaScript library that aims to facilitate the development of flexible, dynamic, and interactive annotation visualization systems. SODA is designed to enable augmentation with lower-level visualization libraries where needed, and it can also be used to generate simple one-time-use figures. The SODA source code is available at https://github.com/sodaviz/soda, the library is packaged through NPM (@sodaviz/soda), and the website https://sodaviz.org/ provides documentation and examples of visuals implemented with SODA.

MATERIALS AND METHODS

Design

SODA is a lightweight, purely front-end library implemented in TypeScript. TypeScript is compiled to JavaScript, which means SODA can be used from either TypeScript or JavaScript. Under the hood, SODA figures are created with Scalable Vector Graphics (SVG). SODA is an object oriented library and depends only on the popular JavaScript visualization library D3. SODA objects are designed with extension in mind, and they expose functions that allow for a great deal of programmatically-driven change at runtime. Currently, SODA is focused on the presentation of annotations in linear context, with light support for representation in a circular context (e.g. for bacterial genomes).

SODA is a toolkit with which a developer may *create* a visualization system, rather than a visualization system in and of itself. The main consequence of this philosophy is that SODA does not produce visualizations using templating mechanisms commonly found in genome browser tools

*To whom correspondence should be addressed. Email: twheeler@arizona.edu
Correspondence may also be addressed to Jack W. Roddy. Email: jroddy@arizona.edu

and the like. Instead, SODA places fine-grained control in the hands of developers with a modest complexity trade-off: developers must define simple data structures to house their data, and functions that utilize SODA's rendering API to produce a visualization.

High-level description of developing with SODA

Development using SODA is largely focused on the configuration and management of Chart objects, which are wrapper objects that control SVG-based viewports in web pages. Creating a SODA visualization typically involves implementing simple objects that describe the annotation data being visualized, designing a data rendering payload that contains collections of those objects, and using the SODA API to configure the visual representation of the payload and the ways in which that visualization will respond to user inputs (e.g. clicking, panning, zooming).

SODA's treatment of data

SODA is designed around a minimal abstraction of annotation data, rather than specific annotation data formats (e.g. BED, GFF3, GTF). For gene-like annotations that describe an interval, this is a simple object composed of an identifier string, a start coordinate, and an end coordinate. For annotations that describe position-specific information throughout an interval, SODA uses the same object specification with an additional array/vector of position-specific values. All of the core SODA rendering features are designed to produce a visual representation of any JavaScript objects that extend this simple pattern. In practice, annotations are commonly augmented with auxiliary data (e.g. alignment strand or score) which, during visualization, can be used to modulate some aspect of the visual representation of the annotation (e.g. color or opacity). With this in mind, SODA rendering features are also designed such that it is easy for a developer to use additional object data fields to control glyph styling parameters.

The result of this design philosophy is that, in principle, SODA supports exactly one data format: JavaScript objects. In practice, this means that SODA may use data in any format from any source, as along as the developer has the means to create JavaScript objects from the data. However, this process is simple under standard database configurations with an accompanying REST API that exposes annotation records via HTTP GET requests. In modern JavaScript, storing the result of a GET request as a string in browser memory may be performed in as little as a single line of code. Commonly, the result string is formatted in JSON (JavaScript Object Notation), which can be serialized directly into JavaScript objects. Typically, these serialized objects fall into three categories: (i) they are immediately suitable for use with SODA; (ii) they can be lightly augmented to become suitable objects or (iii) they contain string records that can be parsed into suitable objects. To aid in (iii), SODA provides parsers for the BED and GFF3 formats. In some cases (e.g. no REST API, using local data files), the burden of implementing the loading and parsing data may fall more heavily on the developer. For more details on the process of loading data, refer to the library documentation (https://sodaviz.readthedocs.io).

SODA's rendering API

To render glyphs with SODA, Annotation objects are passed into rendering functions along with a configuration that specifies the target Chart object and styling parameters. For each style parameter, either a static value or a callback function may be provided. The callback functions are evaluated for each individual glyph, and the represented Annotation object and target Chart are passed as arguments. Styling callback functions provide a simple mechanism to control glyph style using annotation data and may be re-evaluated at runtime to achieve visualization dynamics (e.g. zooming and transforming).

RESULTS

Development of SODA was motivated by a need in our group to create genome-oriented visualizations for which existing genome visualization tools either were unable to provide desired functionality or were deemed unnecessarily heavyweight. Here, we present several of these visualizations, with the goal of demonstrating SODA's flexibility in providing options for rendering and interactivity. Working demonstrations for each example can be found at https://sodaviz.org, and the underlying source code can be found at https://github.com/sodaviz/. These examples present applications of SODA for visualizing genome annotations, but it can be used to visualize protein annotations as well.

Dfam visualization

Dfam (9) is an open access database of Transposable Elements (TEs). One feature on the Dfam website allows users to view a representation of the set of annotated TE instances in a relatively short region (up to 100 000 nucleotides) of a chromosome. In collaboration with the maintainers of Dfam, we replaced the original implementation with a more feature-rich variant using SODA (Figure 1).

The bulk of the annotations underlying the Dfam visualization are the result of comparing a genome to a database of known TE families. The resulting TE annotations are supplemented with annotations for instances of simple tandem repeats (repetitive sequence such as 'atgatgatgatg'). In the linear genome portion of the visualization (top of Figure 1), each annotation record is represented by a rectangle glyph that is colored according to the TE family assigned to the position. There are three SODA components in the visualization, stacked vertically in the following order:

- TEs annotated on the forward strand of the chromosome
- Simple tandem repeat annotations (black glyph on gray genome background)
- TEs annotated on the reverse strand of the chromosome

Immediately following the SODA components are two additional visual components, not built with SODA: (i) a legend describing the colors used in the visualization, and (ii) a tabular description of each annotation rendered above (the table is a fixture of the Dfam website, found at https://www.dfam.org/search/annotations; it is not found in the example on sodaviz.org). Within the SODA components, hovering over glyphs triggers a highlight effect along with an in-

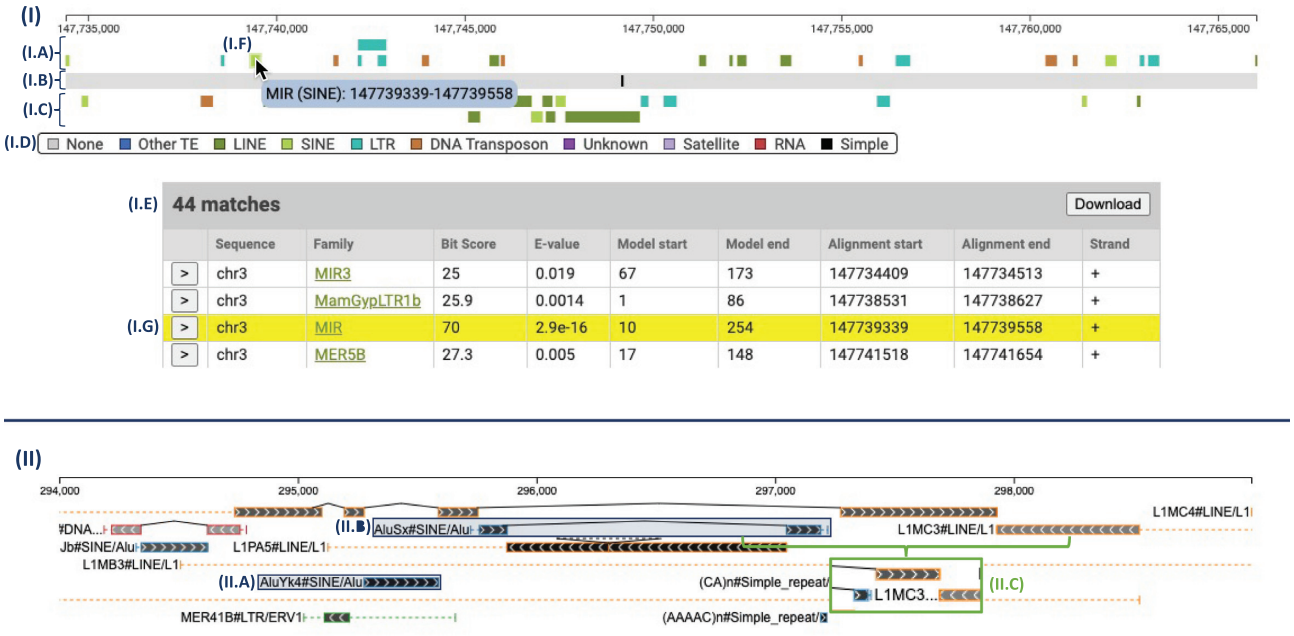

Figure 1. (**I**) depicts the Dfam-SODA visualization. At (**I.A**), (**I.B**) and (**I.C**) are the forward strand TE annotations, simple repeat annotations, and reverse strand TE annotations, respectively. At (**I.D**) is the repeat classification color map, and at (**I.E**) is the table describing the annotations visualized—neither (**I.D**) nor (**I.E**) are created with SODA. At (**I.F**), the green rectangle has been hovered and clicked, causing the descriptive tooltip to appear and the annotation's corresponding row at (**I.G**) to be highlighted. (**II**) depicts the RepeatMasker-SODA visualization. At (**II.A**) is a simple glyph representing a full-length TE annotation. At (**II.B**) is a compound glyph representing a joined group of TE fragment annotations. At (**II.C**), an image is shown of the L1MC3 text label after the visualization is zoomed out—when the label has less screen space, it renders a less detailed string.

formational tooltip popup. Additionally, this figure demonstrates the way that SODA objects can interact with the surrounding website based on SODA callback functions: when a glyph is clicked by the user, the table highlights and scrolls to the row corresponding to the clicked glyph.

RepeatMasker visualization

The UCSC genome browser (3) is a popular tool that houses visualization tracks for dozens of kinds of genomic annotation. The tracks are independently configured and many are submitted by external groups; they can differ greatly in visual complexity. The RepeatMasker Viz. track is a highly nuanced and information-dense track that represents the annotation of TEs and other repetitive DNA features as labeled by the tool RepeatMasker (10). Using SODA, we developed a copy-cat implementation of the RepeatMasker Viz. track (Figure 1) to serve as a test-case with complex visual expression requirements.

The RepeatMasker Viz. track presents the same type of annotations as the Dfam visualization described in the previous section, but includes additional visual indicators to represent complex relationships that are not present in the Dfam visualization. As in the Dfam visualization, annotated TEs and simple repeats are represented with rectangle glyphs. The rectangles in the RepeatMasker plot, however, differ in three ways:

- The outline, rather than the entire glyph, is colored according to the TE family color scheme.
- The interior of the rectangles are shaded in grayscale to indicate the inferred biological age (determined by the quality of the sequence alignment that defines the annotation) of the annotated feature. Younger features appear darker, while older features appear lighter.
- The interior of each rectangle is textured with a repeating chevron pattern to indicate the chromosome strand on which the feature is found.

TE annotations in a genome are often fragments of a full-length known TE family. These fragments can arise from partial replication (at time of insertion), partial deletion (some time after full-length insertion), or insertion of a newer element into the middle of a previously inserted TE (resulting in two fragments for the older element). Two indicators are added to the RepeatMasker visualization to represent these events:

- When an annotation is identified as a fragment, dashed horizontal lines are rendered on either side to indicate portions of the consensus TE sequence that are missing from the fragment.
- When discontiguous fragments are inferred to be the result of an older element having been split by a newer insertion, they are joined by two angled lines meeting at a point.

We refer to groups of joined fragments, dashed lines, and angled lines as a compound glyph. A text label is placed immediately to the left of each compound glyph. As the user

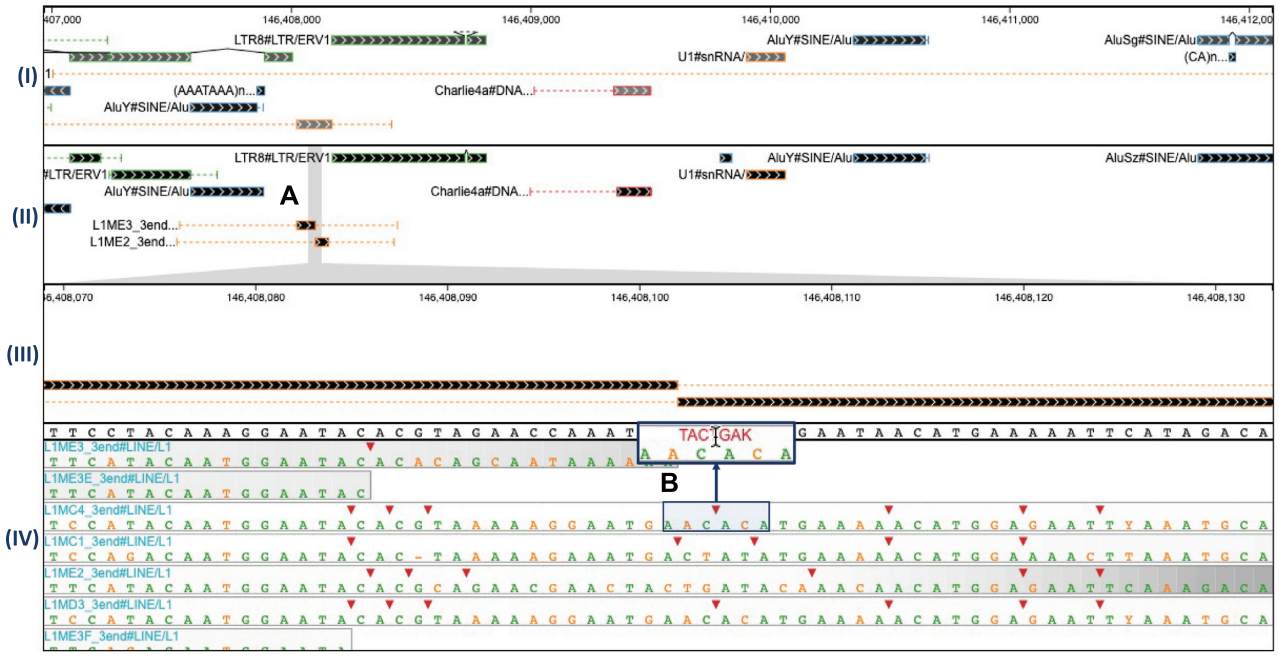

Figure 2. In the PolyA-SODA visualization, pre-existing TE annotations are shown at (**I**) and PolyA-adjudicated annotations are shown at (**II**). At (**A**), a brush selection has been made, and the selected interval is magnified and displayed at (**III**). The alignments and confidence scores corresponding to the selected interval are shown at (**IV**). (**B**) The results of hovering over one of the triangle glyphs that represent alignment insertions relative to the genome—the inserted sequence replaces the triangle while hovered.

zooms in and out, the labels automatically adjust the level of displayed text detail, based on available space.

PolyA visualization

Our group has developed a tool, called PolyA (11), that adjudicates between competing alignment-based annotations by computing position-specific measures of confidence, identifying a trace with maximal confidence, and recursively splicing/stitching inserted elements. For debugging and exploration purposes, we found it helpful to develop a visualization application that provides insight into the differences between annotation results with PolyA and an alternative adjudication method (see Figure 2).

Because PolyA's development was motivated by the goal of improving annotation of TEs, the visualization places PolyA information in the context of RepeatMasker's adjudication results, as produced by its internal tool, ProcessRepeats. Specifically, the PolyA-SODA visualization has four components:

- A component displaying annotations queried from a mirror of the UCSC RepeatMasker database (these represent the results from the adjudication method found in RepeatMasker, called ProcessRepeats).
- A component displaying the PolyA adjudicated annotations for the same region. The user can make a brush selection on this component, resulting in changes to the following two components.
- A component that displays the exact (magnified) contents of the brushed interval in the above component.
- A component that displays a heatmap of the confidence scores for competing alignments in the same region. The sequence alignments from which the confidence scores were calculated are overlaid on top of the heatmap for each alignment.

This tool enables visual inspection of complicated genomic regions, identifying differences between the two annotation adjudication methods, and providing quick visual access to the sequence alignments and corresponding confidence values computed within PolyA.

VIBES visualization

We are currently developing a tool for the identification of phage sequences integrated into bacterial genomes, called VIBES (Viral Integrations in Bacterial gEnomeS). To support data exploration by VIBES users, we have developed a SODA-based application consisting of several interconnected components.

The VIBES pipeline accepts a batch of bacterial genomes and a batch of phage genomes and identifies regions in the bacterial genomes that appear to be the result of phage integration. One output from VIBES is a representation, for each bacterial genome, of the locations of inferred (possibly partial) phage integrations—there can be several per bacterial genome. Another VIBES output is, for each phage genome, a nucleotide-precision representation of the frequency with which each phage nucleotide is part of an observed integration across the batch of genomes. VIBES also identifies all Swiss-Prot and Pfam annotations for each phage genome.

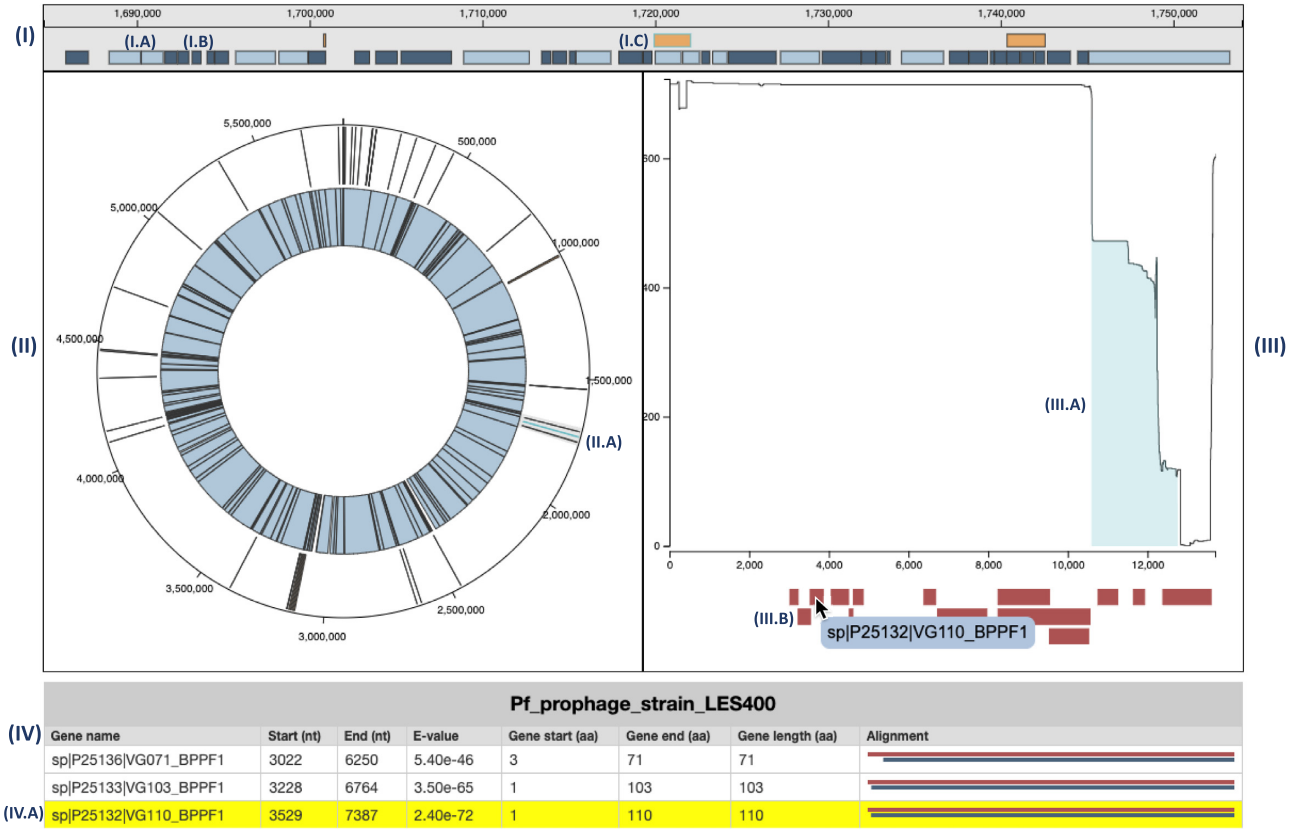

Figure 3. In the VIBES-SODA visualization, the linear bacteria chart is displayed at (**I**). At (**I.A**) is a handful of light blue bacterial gene annotations—each of these rectangles represents an individual gene. At (**I.B**) is a handful of dark blue bacterial gene annotations—each of these rectangles represents a group of genes aggregated to simplify display. At (**I.C**) is an orange phage integration annotation—it is highlighted in cyan to indicate that it is the current *selected* annotation. At (**II**) is the circular bacteria chart. At (**II.A**), we see the same selected phage annotation and the gray shading that indicates the region that is currently rendered in the linear chart. At (**III**) is the occurrences plot, which displays position specific integration counts across all bacterial genomes for the selected phage integration annotation. At (**III.A**) is the interval on the plot that indicates the portion of the phage genome that was identified in the selected phage annotation. At (**III.B**) is the collection of viral gene annotations for the selected phage. At (**IV**) is the table that describes the viral gene annotations shown in (III.B). One of the viral gene annotations has been hovered and clicked, causing the tooltip at (III.B) and the row at (**IV.A**) to be highlighted.

At the top of the VIBES visualization (Figure 3) is a chart that represents a linearized version of the target bacterial genome. The top row of the linear chart displays annotations of phage integrations to the bacterial genome as identified by VIBES, and the bottom row displays gene annotations. At the lowest zoom level, the gene annotations are condensed into groups based on proximity in the chromosome. As the zoom level increases, the gene annotations are continuously re-rendered as more horizontal screen real estate enables finer-grained separation and display.

Below the linear chart on the left is a circular representation of the bacterial genome. Here, phage integrations are shown on the outer ring and a condensed representation of gene annotations are on the inner ring. As the linear chart is zoomed and panned, a brush is drawn on the circular chart that highlights the region of the bacterial genome that is currently shown in the linear view.

On both the linear and circular charts, a single phage integration is highlighted blue, indicating that it is currently 'selected'. The selected annotation determines the contents of the occurrences plot, which is directly to the right of the circular chart. The top of the occurrences plot displays the position-specific integration frequencies of the phage responsible for the current selected annotation. In the occurrences plot, the region of the phage genome that was identified in the current bacterial genome is highlighted in blue (a partial integration will have blue highlight over just the corresponding fragment of the full length phage). The bottom of the occurrences plot indicates positions of genes and protein domains along the entirety of the phage genome. The selected annotation can be switched by clicking on phage annotations on either the linear or circular chart, or by pressing the arrow buttons that appear when hovering over the circular chart.

At the bottom of the visualization is a simple table that describes the gene annotations across the selected phage genome. Clicking on an annotation in the phage occurrences plot will cause the corresponding entry to be highlighted in the table.

DISCUSSION

Here, we have introduced SODA, an open-source TypeScript/JavaScript library that enables the development of web-based biological sequence annotation visualizations. SODA is primarily intended for creating visualization systems, but it also has utility in the generation of one-off figures. In particular, we imagine SODA as a replacement not for existing domain-specific tools, but for lower-level, general purpose web visualization technologies. SODA's feature set is broadly capable, but, in our experience, it is often useful to augment a SODA-based visualization by integration with components built with other web technologies. We have designed SODA to be easy to use, broadly expressive, and extensible; we would be grateful for community engagement, in the form of identifying feature gaps and contributing to SODA's source code to extend expression options.

Future work

Though SODA is rich in features, it will benefit from continued development. In the future, we will expand the collection of available glyph shapes, improve glyph customization options, explore additional interactivity features, and implement a robust system for the export of static images. We will also improve support for rendering annotations in a circular context, raising it to feature parity with linear rendering features. We plan to implement optimized rendering systems for visually linked annotations (e.g. synteny maps (12)) and matrix-like data (e.g. Hi-C (13), and genomic repeat structures (14)). We plan to develop a system that facilitates the addition of SODA visuals to offline software (e.g. command line tools). Finally, we intend to develop a complementary toolkit on top of SODA that operates at a higher level of abstraction with the goal of being more accessible to non-developers.

DATA AVAILABILITY

The SODA source code is available at https://github.com/sodaviz/soda, the library is packaged through NPM (@sodaviz/soda), and the website https://sodaviz.org/ provides documentation and examples of visuals implemented with SODA.

ACKNOWLEDGEMENTS

We are grateful to Jeb Rosen and Robert Hubley for the extensive and valuable feedback they provided during development of the library, and Daniel Olson and Thomas Colligan for helpful comments on drafts of the manuscript.

We thank Audrey Shingleton and Conner Copeland for their aid in development of example applications of SODA (PolyA and VIBES, respectively), and also thank anonymous reviewers for their helpful suggestions.

FUNDING

NIH [R01-GM132600, U24-HG010136]; University of Montana.
Conflict of interest statement. None declared.

REFERENCES

1. Buels,R., Yao,E., Diesh,C.M., Hayes,R.D., Munoz-Torres,M., Helt,G., Goodstein,D.M., Elsik,C.G., Lewis,S.E., Stein,L. *et al.* (2016) JBrowse: a dynamic web platform for genome visualization and analysis. *Genome Biol.*, **17**, 66.
2. Howe,K.L., Achuthan,P., Allen,J., Allen,J., Alvarez-Jarreta,J., Amode,M.R., Armean,I.M., Azov,A.G., Bennett,R., Bhai,J. *et al.* (2021) Ensembl 2021. *Nucleic Acids Res.*, **49**, D884–D891.
3. Navarro Gonzalez,J., Zweig,A.S., Speir,M.L., Schmelter,D., Rosenbloom,K.R., Raney,B.J., Powell,C.C., Nassar,L.R., Maulding,N.D., Lee,C.M. *et al.* (2021) The UCSC genome browser database: 2021 update. *Nucleic Acids Res.*, **49**, D1046–D1057.
4. L'Yi,S., Wang,Q., Lekschas,F. and Gehlenborg,N. (2021) Gosling: A grammar-based toolkit for scalable and interactive genomics data visualization. *IEEE Trans. Vis. Comp. Grap.*, **28**, 140–150.
5. Watkins,X., Garcia,L.J., Pundir,S., Martin,M.J. and Consortium,U. (2017) ProtVista: visualization of protein sequence annotations. *Bioinformatics*, **33**, 2040–2041.
6. Nightingale (2021) Nightingale — a monorepo containing visualisation web components to use with biological data. https://github.com/ebi-webcomponents/nightingale, (1 October 2022, date last accessed).
7. Bostock,M., Ogievetsky,V. and Heer,J. (2011) D3 data-driven documents. *IEEE Trans. Vis. Comp. Grap.*, **17**, 2301–2309.
8. PixiJS (2021) PixiJS — The HTML5 Creation Engine. https://pixijs.com/, (1October 2022, date last accessed).
9. Storer,J., Hubley,R., Rosen,J., Wheeler,T.J. and Smit,A.F. (2021) The Dfam community resource of transposable element families, sequence models, and genome annotations. *Mobile DNA*, **12**, 2.
10. Smit,A.F. (2022) Repeat-Masker Open-3.0. http://www.repeatmasker.org, (20August 2022, date last accessed).
11. Carey,K.M., Hubley,R., Lesica,G.T., Olson,D., Roddy,J.W., Rosen,J., Shingleton,A., Smit,A.F. and Wheeler,T.J. (2021) PolyA: a tool for adjudicating competing annotations of biological sequences. bioRxiv doi: https://doi.org/10.1101/2021.02.13.430877, 14 February 2021, preprint: not peer reviewed.
12. Derrien,T., André,C., Galibert,F. and Hitte,C. (2007) AutoGRAPH: an interactive web server for automating and visualizing comparative genome maps. *Bioinformatics*, **23**, 498–499.
13. Durand,N.C., Robinson,J.T., Shamim,M.S., Machol,I., Mesirov,J.P., Lander,E.S. and Aiden,E.L. (2016) Juicebox provides a visualization system for Hi-C contact maps with unlimited zoom. *Cell Syst.*, **3**, 99–101.
14. Vollger,M.R., Kerpedjiev,P., Phillippy,A.M. and Eichler,E.E. (2022) StainedGlass: Interactive visualization of massive tandem repeat structures with identity heatmaps. *Bioinformatics*, **38**, 2049–2051.