Variation graph visualization: an exploration of visualizing graph genomes in layers using Three.js

William Gao

***Abstract*—A pangenome represents genetic variations in a population as a *variation graph*, which greatly reduces the reference bias that comes from using a linear reference genome. However, a linear reference genome is more intuitive to understand and has been the traditional way that bioinformaticians use. As an effort to make it easier to visualize and interpret variation graphs, I present `pgv`, an interactive visualization tool built on top of previous work that aims to display structural variations in a variation graph interactively. Instead of fitting the nodes, edges, and paths of a variation graph in a 2-dimensional space, `pgv` draws the sequence graph itself on the x-y plane, and paths are rendered as separate layers that can be interactively highlighted.**

***Index Terms*—bioinformatics, reference genome, pangenome, variation graphs, visualization**

I. Introduction

THE improvements to sequencing technologies have led to the rapid growth in sequencing data. Many meaningful analyses can be performed on these data to help with the understanding and diagnosis of human diseases. However, the traditional way of comparing and analyzing genomes often involves using a linear reference genome, which is known for its reference bias [1].

A pangenome reduces bias by representing genetic diversity as a graph [2]. Instead of building one linear reference genome from a selected few individuals in a population that only represents one version of each locus, a graph structure can represent any number of sequences and be updated to include new variations. There are various file formats for storing graph genomes. In particular, the vg toolkit defines the *variation graph* data structure that consists of nodes, edges, and paths [2]. A node stores sequence data and an edge describes a bi-directed linkage between two nodes [2]. Together, the nodes and edges of the graph encode every possible variation of the genome. A path, or a "walk" on the graph, represents one haplotype or sequence from the population.

Over the years, pangenomes are emerging as a replacement for linear reference genomes for sequence analysis [1]. However, visualization of such nonplanar pangenome graphs is inherently more difficult in a limited dimension [3]. As an effort to make it easier to visualize and interpret pangenome graphs, I propose a new way of visualizing variation graphs that aims to work alongside existing visualization tools to display structural variations in a variation graph in an intuitive and interactive layout.

II. Methods

The following subsections describe the methods that `pgv` uses to display a visualization of a variation graph.

A. Data pre-processing

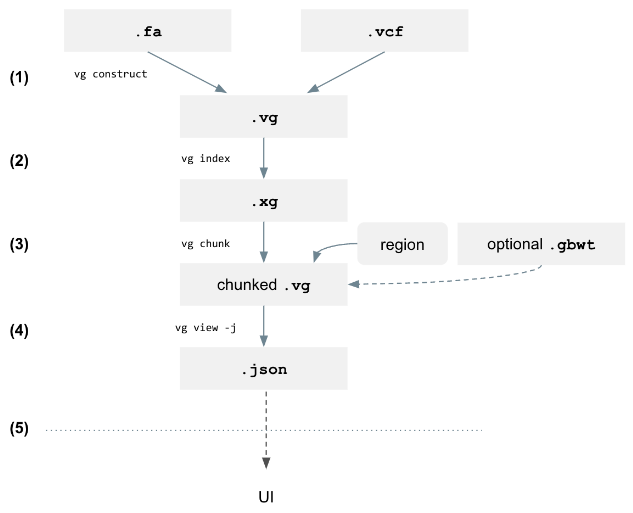


Fig. 1. A high-level overview of the data flow of `pgv`. (1) The `vg construct` command takes in a list of FASTA and VCF files and outputs a vg graph that encodes the reference sequence along with the variations. (2) `vg index` computes an index file of a graph that can then be used to efficiently access arbitrary regions. (3) Optionally, for large graphs, `vg chunk` can be used to select a specific region of interest, along with additional haplotypes that can be included in the output graph. (4) `vg view -j` converts the graph into JSON, a text-based format that is widely supported across platforms. Finally, (5) the graph in JSON format can be passed to the UI for rendering.

Figure 1 shows the high-level data flow of `pgv` to pre-process a graph for rendering. `pgv` primarily works with the file formats used by the vg toolkit [2] including FASTA (*.fa*), VCF (Variant Call Format, *.vcf*), and GBWT (Graph Burrows-Wheeler Transform, *.gbwt*) files.

B. Layout

Graph layout is a complex subject. There are many algorithms for generating layouts of a traditional graph, but there is no optimal and universal way to generate a layout that always looks good for any graph. Nevertheless, for traditional graphs, tools such as graphviz [4] are good starting points to tinker with different ways to display a graph.

For variation graphs, the layout engine also needs to account for the additional set of paths. Many visualization tools have been developed with different layouts and at different scales [3]. Visualization tools that are designed specifically for variation graph visualization such as sequenceTubeMap [5] often focus on the pangenome relationship at the nucleotide level. Other visualization methods such as Bandage focus on the overall topology of the graph at a more zoomed out level [3]. In general, pangenome visualization tools are either designed to display the overall structure of the graph that omits the nucleotide-level detail of the variations, or display at the nucleotide level to reflect the pangenome aspect of the graph but hard to scale to the chromosome or genome level [3].

C. Rendering

Once the layout is computed, rendering is the matter of drawing each component of the graph efficiently. In this step, it is important to think about different trade-offs. For example, a web-based tool would be more accessible than a software that requires download, but browser resources are often limited compared to a native application. Although, with technologies such as WebGL [6] for rendering interactive 2D and 3D graphics on most modern browsers with GPU acceleration, a web-based platform seems suitable for a tool that is designed to be interactive and accessible. Moreover, Three.js [7] is a library that makes it easy to work with WebGL using JavaScript with API calls.

Ideally, the visualization tool should also be responsive across different devices and screen sizes. This can be done on the web using CSS (Cascading Style Sheets). Other design considerations include the color choices as well. For example, it is important to use high-contrast colors when differentiating different elements, and color palettes that are colorblind friendly.

III. Results

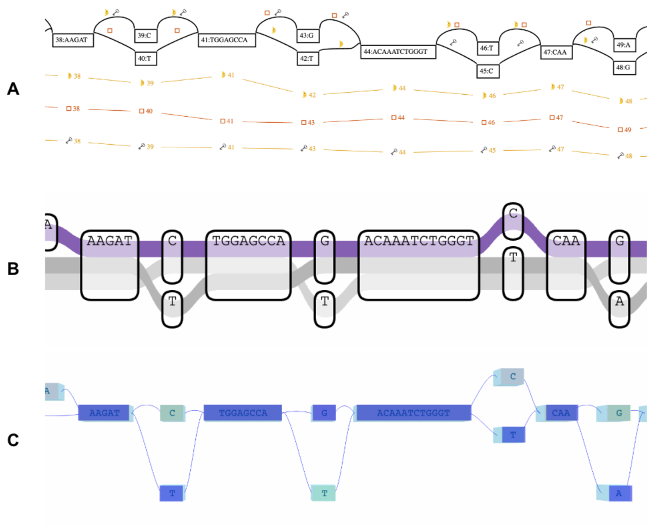


Fig. 2. A side-by-side comparison of a variation graph visualized using (A) vg view (graphviz), (B) sequenceTubeMap and (C) `pgv`. (A) vg view displays the paths at the bottom in addition to annotated edges to represent paths. It is rendered as a static SVG. (B) is a screenshot of sequenceTubeMap that displays the graph as a tube-like structure inspired by transit system maps [3]. Paths are embedded within the nodes and edges. (C) is a screenshot of `pgv` that uses the same layout as sequenceTubeMap, but renders paths as layers in the z-axis and can be interactively selected.

Here, I present `pgv`, a web-based interactive visualization tool that implements the methods described above. At this stage, `pgv` is in alpha version and under development, but a prototype implementation is available to demonstrate the concept. The prototype is available at: https://w-gao.github.io/pgv/ and runs on any modern web browsers on mobile and desktop. Figure 2 shows the comparison of a variation graph visualized using vg view (graphviz), sequenceTubeMap, and `pgv`.

A. Architecture and implementation

At a high level, there are two main parts to `pgv` – a command-line interface (CLI) and a web-based component as the UI. The CLI programmatically implements the data pre-processing pipeline described in the Methods section. It is a Python wrapper of vg commands that takes in various sequence files and produces a JSON file that can then be passed to the UI for rendering.

The UI is implemented in TypeScript and is modularized into three components: repository, layout, and renderer. Each component provides an interface that can be extended to different implementations. For the prototype, it uses a local repository, the sequenceTubeMap layout, and a Three.js renderer. For the layout, sequenceTubeMap [5] is pulled in as an external dependency with minimal modification to expose some private functions.

The nodes and edges of the variation graph are drawn on the x-y plane, whereas the paths are rendered as separate layers that can be interactively selected using the navigation buttons or keyboard shortcuts. By default, each layer has an opacity value set to 0.1. In effect, the more paths that walk over a particular node, the darker the node is. This suggests a rough estimate of the allele frequency at each node. It is also possible to hover a node to get the exact number along with additional information such as the length (number of bases) of the node. Each path can also be individually selected by using the up and down arrow buttons. This is useful to display the sequence of a particular path of interest, where each node that is passed by the path will be highlighted.

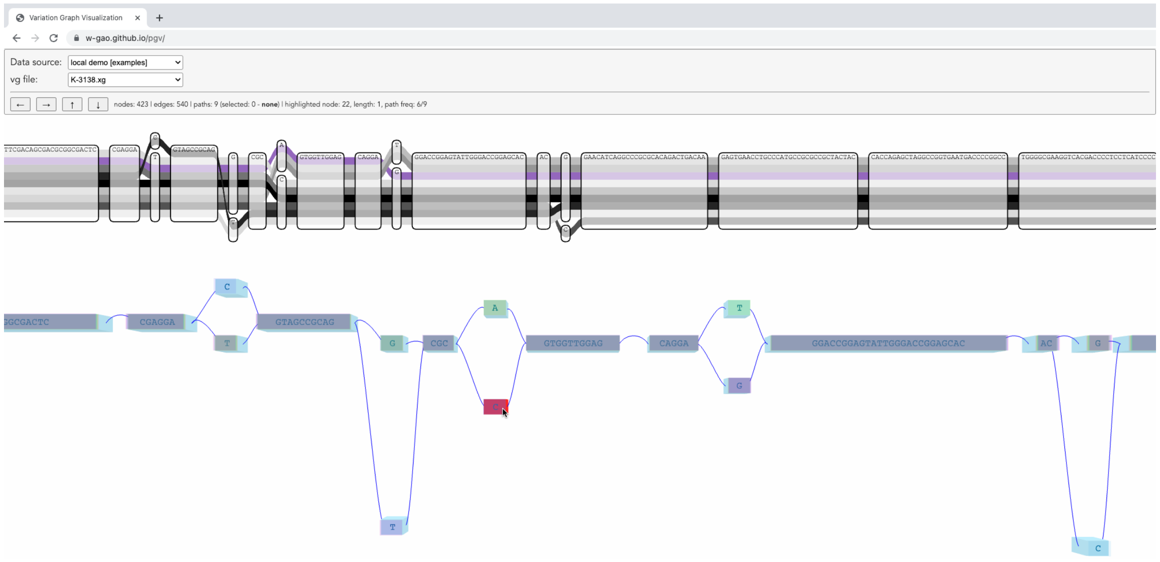
B. Example visualizations

Figure 3 is a screenshot of `pgv` rendering a small region of the HLA-K pseudogene in GRCh38 with 9 alternative sequences [8].

IV. Discussion

For a linear reference genome, visualization tools such as the UCSC Genome Browser [9] display alignments as tracks in two dimensions. For pangenomes, because the nodes and edges in the variation graph are already in two dimensions, it is often complicated to display the paths in the same space. `pgv` aims to extend the traditional way of visualizing pangenome graphs into the third dimension using modern web-based technologies. As a tool, it intends to provide a platform for researchers to visualize and explore variation graphs interactively.

The current state of `pgv` is a proof of concept to demonstrate this idea. Future directions include adding support for different zoom levels and resolution, display edits within



Data source: local demo [examples]
vg file: K-3138.xg

nodes: 423 | edges: 540 | paths: 9 (selected: 0 - **none**) | highlighted node: 22, length: 1, path freq: 6/9

Fig. 3. A screenshot of `pgv` rendering a small region of GRCh38 and 9 of its alternative sequences in the HLA-K pseudogene (gene ID: 3138) [8] built with vg msga (Variation Graph multiple sequence / graph aligner) [3]. The UI consists of three sections – the header for graph selection and navigation, the sequenceTubeMap render of the graph, and the `pgv` render of the graph.

the paths for partial matches, and dynamic render of graphs based on the window view. With pangenomes emerging as the replacement for linear reference genomes, there is a pressing need for more graph genome tools that make it easier for researchers to work with graph genomes. *pgv* aims to be one of the tools in a researcher's toolbox for variation graph visualization.

Availability and Implementation

No new data were collected for this paper. The source code is available at: https://github.com/w-gao/pgv under the MIT open-source license. A working demonstration is hosted at: https://w-gao.github.io/pgv/.

References

[1] J. Sirén et al., "Pangenomics enables genotyping of known structural variants in 5202 diverse genomes," Science, vol. 374, no. 6574, p. abg8871, doi: 10.1126/science.abg8871.
[2] E. Garrison et al., "Variation graph toolkit improves read mapping by representing genetic variation in the reference," Nature Biotechnology, vol. 36, no. 9, pp. 875–879, Oct. 2018, doi: 10.1038/nbt.4227.
[3] J. M. Eizenga et al., "Pangenome Graphs," Annual Review of Genomics and Human Genetics, vol. 21, no. 1, pp. 139–162, Aug. 2020, doi: 10.1146/annurev-genom-120219-080406.
[4] Graphviz. [Online]. Available: https://graphviz.org/. [Accessed: 16-Mar-2023].
[5] W. Beyer et al., "Sequence tube maps: making graph genomes intuitive to commuters," Bioinformatics, vol. 35, no. 24, pp. 5318–5320, Dec. 2019, doi: 10.1093/bioinformatics/btz597.
[6] "WebGL," The Khronos Group, 19-Jul-2011. [Online]. Available: https://www.khronos.org/webgl/. [Accessed: 16-Mar-2023].
[7] "Three.js, JavaScript 3D Library." [Online]. Available: https://threejs.org/. [Accessed: 16-Mar-2023].
[8] "HLA-K major histocompatibility complex, class I, K (pseudogene) - NCBI," National Center for Biotechnology Information. [Online]. Available: https://www.ncbi.nlm.nih.gov/gene/3138. [Accessed: 18-Mar-2023].
[9] W. J. Kent et al., "The human genome browser at UCSC," Genome Res, vol. 12, no. 6, pp. 996–1006, Jun. 2002, doi: 10.1101/gr.229102.