Vizitig: a pangenome and pantranscriptome explorer

Bastien Degardins[1], Charles Paperman[1] and Camille Marchet[1]
[1] UMR9189 CRIStAL, Univ Lille, CNRS, Centrale, F-59000 Lille, France
camille.marchet@univ-lille.fr

Abstract

Vizitig is the first platform for real-time exploration and querying of DNA and RNA sequence graphs across many samples, unifying visualization, metadata, and flexible search. It integrates raw and reference-based data, handles complex variation, and provides a human-readable query language with scalable graph loading. Vizitig enables fast, reproducible analysis in both pangenomics and pantranscriptomics.

Main

Advances in high-throughput sequencing allow the construction of large genome and transcriptome collections, driving discoveries in evolution, disease, and functional biology. Biological variants often require comparisons across several individuals, strains, or samples. Tools such as BLAST and IGV [14] have highlighted the power of interactive exploration, but remain tied to linear models. We now need models that integrate diverse sequence collections, such as population-scale pangenomes and multi-sample transcriptomes (pantranscriptomes).

Current pangenomes conversely rely on graphs whose topology can vary across methods and software versions [3]. Existing software offers valuable yet domain-specific functionality: assembly [15] and variation graph viewers, toolkits [7], comparative genomics and phylogenetics frameworks [8]. Pantranscriptomes still lack a factorized, graph-based representation.

A key challenge is to abstract the complexity arising from many accumulated variants, most irrelevant to a given question, while still enabling precise sequence extraction from regions or individuals of interest. Vizitig meets this need through a novel, query-driven graph exploration across DNA/RNA datasets, accessed via a web interface. Nucleotide segments form graph nodes connected by edges, with branches representing variation. Vizitig's novelty rests on three key components. First, an alignment-free index enables rapid sequence search using principles shared with assemblies and large-scale indexes [9]. Second, a database links nodes with metadata, and a logic layer supports expressive, composable queries across three modalities:

a. Sequence of arbitrary span with abundance conditions,

b. Features (e.g., transcript ID, gene name, chromosome),

c. Experimental factors (e.g., strain, sample).

To our knowledge, Vizitig is the first method to allow this set of queries and a combination thereof, e.g., *retrieve a given mutant BRCA1 sequence in two selected samples with abundance* $\geq$*10*. Third, another novelty is on-demand subgraph loading, retrieving only regions of interest while allowing exploration of neighboring context. We provide a comparison with related methods in Supp. Fig. S1–S3, Supp. Table S3. Vizitig offers fast parsing, indexing, and querying, with distributed loads that enables construction of large datasets within hours and query responses in seconds (Supp. Table S2).

Our first case-study is transcriptomic events relevant to cancer, typically rare or weakly expressed, and therefore difficult to detect: fusion transcripts (exons joined from different genes) and circular RNAs (covalently linked 5'-3' loops that linear models capture poorly). For each, we selected known positives from prior studies and constructed Vizitig graphs using the corresponding samples and RefSeq transcripts. Datasets are listed in Supp. Table S2, and all graphs and visualizations and command lines are released for reproducibility.

We re-analyzed RNA-seq data containing two circular RNAs, MAN1A2 and FBXW4, originally described in [11] (Fig 2A). Starting from querying gene-IDs, we could visually detect alternative back-splicing junctions by identifying cyclic node connections and we extracted the exact junction sequences

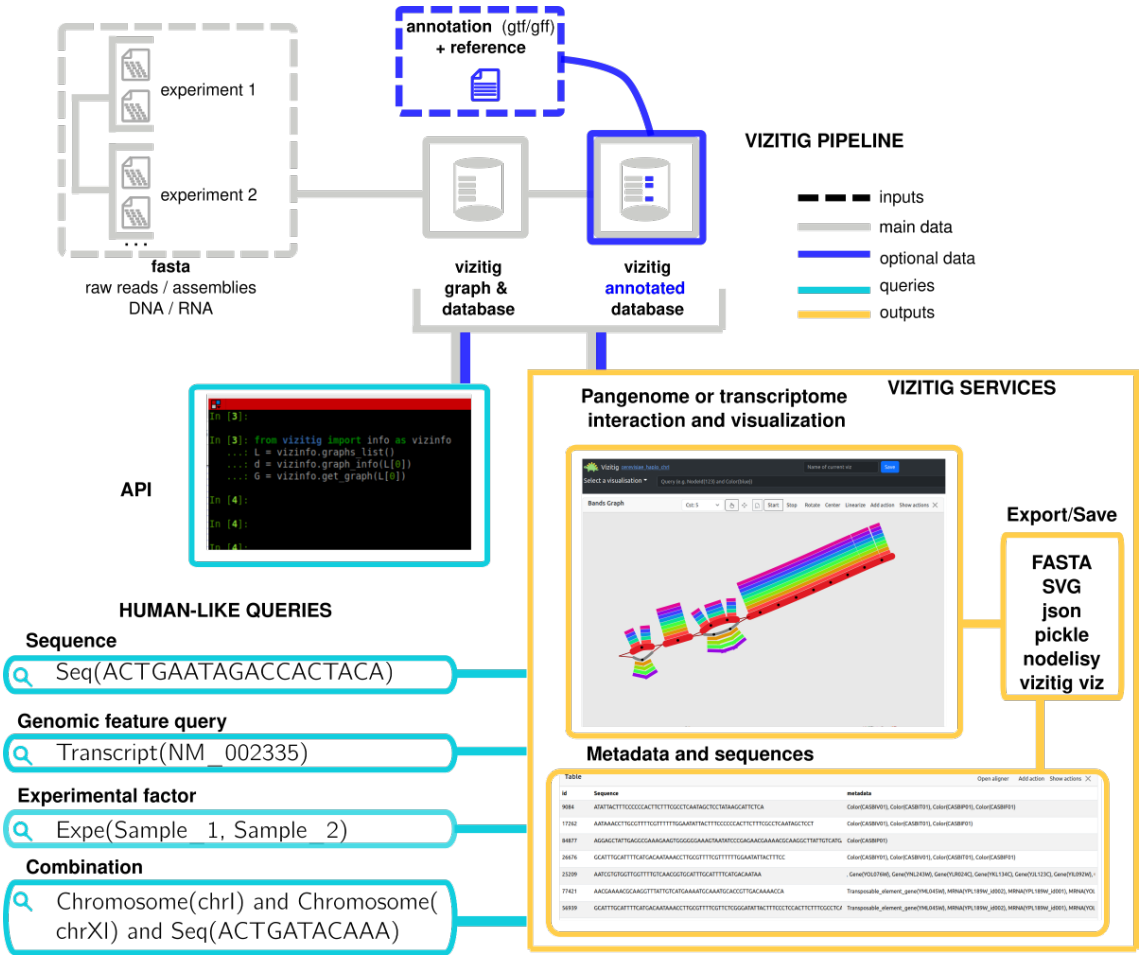

Figure 1: Vizitig integrates diverse data types (reads and references, DNA and RNA) in a graph. Users can filter and query graphs by metadata and sequence traits. An interactive web interface streams only relevant subgraphs, enabling scalability. Results, metadata, and views can be exported for sharing or downstream analysis.

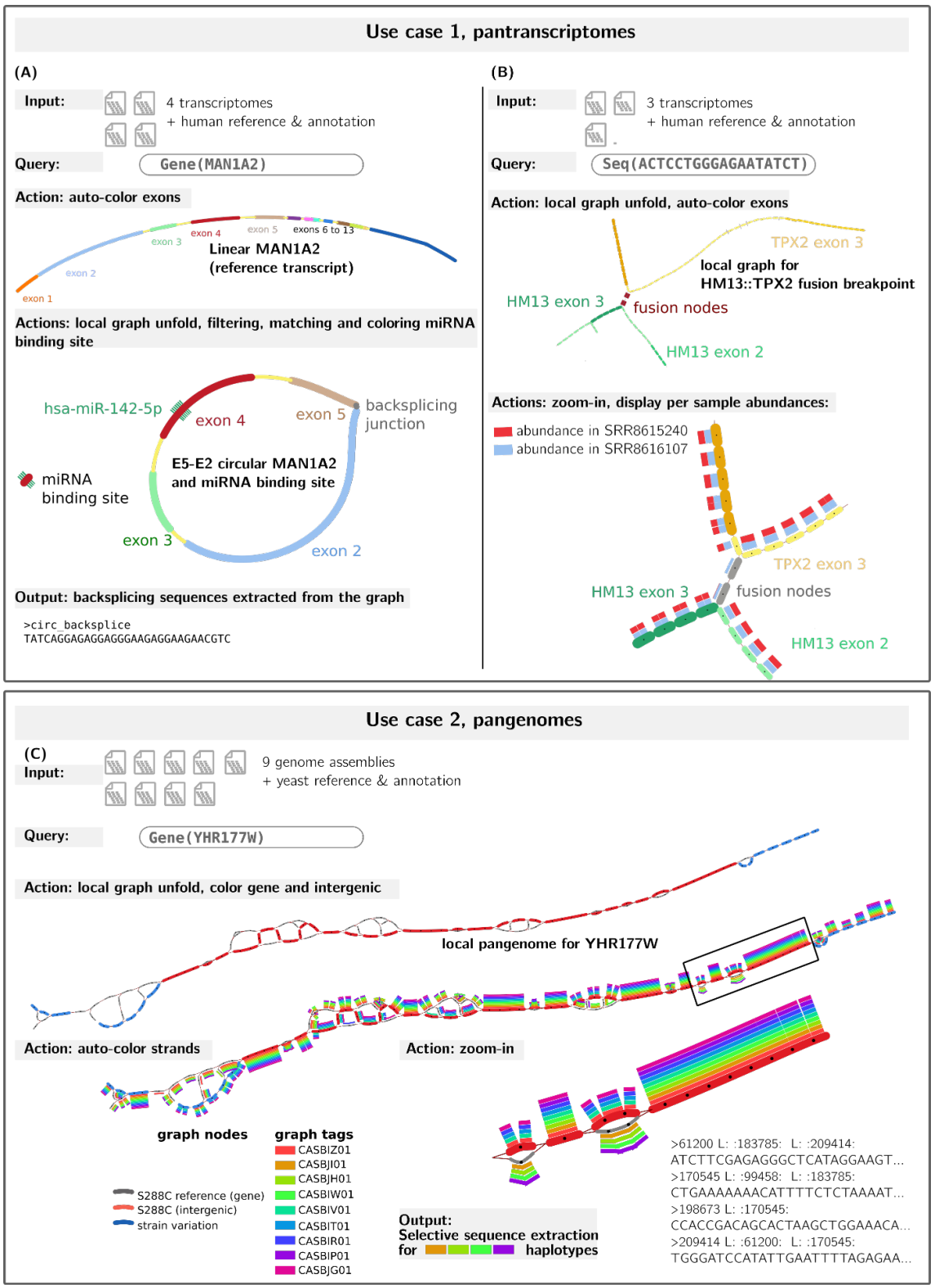

Figure 2: (A) Circular RNA detection: visualization of linear and circular MAN1A2 isoforms, including the E4-E2 back-splice junction and a predicted miRNA site on exon 4. (B) Fusion exploration: a sequence query recovered the HM13::TPX2 breakpoint, highlighting sample specificity. (C) Yeast pangenome: close-up of *S. cerevisiae* YHR177W. The top panel shows the reference (red), variants across nine strains (grey), and intergenic regions (blue). The bottom panel colors nodes by strain, revealing shared reference-like regions and specific variants. Variant sequences can be extracted directly for downstream analysis. All visualizations and colorings shown in (A)-(C) are exported directly from Vizitig as SVG.

absent from the previous publication. We also overlaid miRNA sites predictions [5]. Figure 2A shows the E4-E2 back-splice junction with a predicted miRNA site in exon 4. We then examined the HM13::TPX2 fusion from CCLE/DepMap [2], where querying the breakpoint sequence retrieved the junction node in CCLE RNA-seq and confirmed expression specificity to the fusion-positive sample (Fig. 2B). Long-read evidence further supported other breakpoints (Supp. Fig. S7b).

We next turned to pangenomics, and built a pangenome graph by combining the *S. cerevisiae* reference strain S288C, nine telomere-to-telomere assemblies [12], and gff3 annotations. We queried the YHR177W gene-ID, reported to carry structural variations [12], and showed complexity ranging from local synteny to inter-strain differences and conserved segments (Fig. 2, Supp. Fig. S4). Vizitig narrow the view to a selected subset or strain selection, and allows selective extraction of graph segments haplotypes (Supplementary Results).

These applications highlight how Vizitig unifies queries, metadata, and visualization to support discovery and interpretation of complex sequence data. Future work will focus on extending its scalability, enrich the query language, generalize to other graphs, and develop automated tools for variant pattern detection.

Online methods

Software development and practices

Vizitig is an open-source software package designed for flexible deployment and integration in modern genomics workflows. It is distributed under the BSD license. The codebase is modular and actively maintained, with current test coverage exceeding 70%. The platform consists of a Python-based application programming interface (API), which enables advanced analytical operations such as graph-based filtering, locus comparison across samples, and metadata-driven search. This is complemented by a web-based frontend developed in JavaScript[1], designed for interactive visual exploration and intuitive manipulation of graph structures and sample metadata. The backend combines SQLite for portability and ease of deployment and Rust for performance-critical components such as k-mer (fixed-length sequences extracted from consecutive positions in the data) indexation and query execution. This architecture ensures both speed and accessibility, making Vizitig usable across a range of computing environments, from personal workstations to cloud-based infrastructures. The frontend communicates via a lightweight web API with the core command-line engine, allowing researchers to query, visualize, and annotate graphs interactively. In addition, Vizitig supports export of graph views, metadata, and query results standard formats such as GML, JSON, and SVG, to export subgraphs in BCALM2 format ensuring and to share databases themselves.

Graph model

At the core of Vizitig is a graph-based representation of sequence data built on the de Bruijn graph model, a widely used structure in genome assembly and for alignement-free data representation. De Bruijn graphs capture the global structure of sequences, including chromosomes, transcripts, and variants, by representing k-mer overlaps between reads or assembled genomes. These graphs naturally encode biological variation through topological features such as branching paths and "bubbles" [4].

To handle multi-sample datasets and support comparative analysis, we employ colored de Bruijn graphs [9], which associate each node with sample-specific information. This approach allows scalable co-integration of raw sequencing reads from different samples and reference sequences into a single graph, enabling more complete interpretation of partially covered or noisy regions.
For downstream analysis and visualization, the graphs are converted into unitig graphs, a compact, lossless representation that merges linear paths of k-mers into single nodes [4].
Each unitig node stores an abundance value, computed as the average abundance of its constituent k-mers. These values provide a robust estimate of local sequence coverage and can be used directly in transcriptomic applications as a proxy for transcript-level expression [2].

Integrating raw read datasets from diverse sources inevitably introduces noise, which can complicate analysis and interpretation. Vizitig addresses this challenge both upstream and downstream.

[1] Using https://d3js.org/

During graph construction, optional error correction and noise filtering steps reduce sequencing artifacts and low-coverage errors, ensuring that the resulting graphs remain biologically meaningful and robust while accommodating heterogeneous input formats. Downstream, the visualization module includes an automated noise-filtering mechanism inspired by assembly pipelines [1]. A second feature can dynamically hide nodes or edges on demand, enabling users to focus on the relevant and biologically significant portions of the graph.

A known parameter sensitivity in de Bruijn graph construction is the choice of k-mer size, which can influence graph resolution and connectivity. In our experiments, we adopted standard k-mer sizes (from 21 to hundreds) that are commonly used in genome and transcriptome studies. To illustrate the robustness of Vizitig, we performed additional experiments (Supp. Fig. S4) demonstrating that the biological signal of interest remains detectable even when different k values are applied. Aside from this parameter, and an optional k-mer frequency filtering to spare noise before graph construction, this graph definitions remains very robust across any data type.

Database design

Vizitig's backend currently utilizes SQLite[2] to manage unitig-defined graph structures, associating metadata directly with nodes. It incorporates sequences by marking nodes containing corresponding k-mers with relevant metadata from DNA/RNA sequences or annotations (GTF/GFF), and supports additional metadata such as color (sample identification) and abundance (occurrence in samples). Efficient retrieval is ensured through indexing, providing sublogarithmic query time for single k-mers and logarithmic performance for other sequence lengths. A Python-based API and command line interface offers comprehensive backend access for data manipulation. We chose not to rely on analytical graph databases such as Neo4j, as no mature, open-source, embedded solutions currently exist that can run on standard machines. Additionally, such databases are typically not optimized for rapid retrieval of localized graph regions, our primary query focus. To ensure optimal performance, we developed ad hoc indexes tailored for fast localized queries and integrated them with NetworkDisk[3], a disk-based Python module compatible with NetworkX, facilitating large-scale graph manipulation with minimal memory usage by loading only relevant subsets during exploration.

Human-based queries and interactions

A human-readable domain-specific query language (DSL), supported by a Python-based query assistant, translates user queries into executable NetworkDisk commands, facilitating graph manipulation. Additional DSL-driven filtering and transformation tools allow targeted data extraction and visualization. Vizitig's domain-specific query language allows one to build queries that can be executed both as a retrieval query as well as a filter query. The former fetches sequences and meta-data from the database, while the latter aims at applying visual transformations on data once it was retrieved. The domain-specific query language is composed of logical operators (AND, OR and NOT), parenthesis, and tokens. Tokens are made of a type keyword, open parenthesis, an optional key and a closing parenthesis. Type keywords correspond to the different types of metadata and are generated based on the annotations files (e.g. `Gene`, `Transcript`, `Exon`). The specific keyword `Expe` allows a second, optional argument `A` that comes with an operator (`<` , `>` , `<=` , `=>` or `=`) and a value, that allows to query abundances.

Trait-based queries typically involve commands such as Gene, Transcript, or Exon to load regions of interest, often combined with abundance thresholds or sample presence (e.g. `Expe(Sample_SRR8967009, Abund > 600)`) will output sequences from the given sample with abundance over 600. Sequence-based queries can be further refined with any available metadata or features, providing a flexible way to zoom into specific biological events. To further assist users, we implemented automated detection of likely fusion and backsplicing junction nodes, both across entire datasets and for user-defined gene sets. The automated detection works by exploring the graph using genes as starting points, and looks for paths to other genes that respect a given set of conditions.

Users can also alter the appearance of the graph to highlight specific samples, genes, or exons using custom colors and node shapes. We also implemented an autocolor tool that allows genes, exons,

[2] https://www.sqlite.org/
[3] https://networkdisk.inria.fr/

or chromosomes to be distinguished automatically, reducing the burden of repetitive manual operations. A built-in *sashimi-like* [10] visualization tool displays abundance levels at individual nodes across multiple samples within the graph (for this tool, we show consistency with the established quantification methods in Supp. Fig. S10, and the alignment module inside Vizitig implements the Smith-Waterman algorithm for sequence alignment.

Performances

We collected the data availability, graph information and performances for each experiment in Supp. Table S1.

All Vizitig builds took less than 20GB of RAM during construction for the different experiments. The query times for the different queries described in the article vary between milliseconds to a maximum of 6 seconds of response time. We built de Bruijn graphs using BCALM2 [4] output format.

An extended version of the HM13::TPX2 fusion experiment using 21 human RNA-seq datasets from SRA provides a thinner view of Vizitig's management of large workload. All decompressed SRR reads weight 1 227 Gigabytes on disk. The references include Hg38 RefSeq Transcripts sequences associated to Hg38 reference metadata.
The experiment was conducted on a laptop (CPU Intel Core Ultra 7 165H processor with 16 cores and 22 logical threads 5.0 GHz, RAM 32 GB of DDR5-5600 RAM (2×16 GB modules) 5600 MT/s, Disk 4.0 TB NVMe SSD with 512-byte logical block addressing, mounted as the secondary data storage partition). The graph was built using BCALM2 (k=31, minimum abundance = 2, final size on disk 67.5 GB, index size on disk 13.8 GB).
Despite data size largely exceeding RAM capacity, Vizitig is capable of managing the workload by chunking data and sharding the graph using its indexes. The coloring phase time is linear in number of samples. Peaks in CPU usage are caused by indexes join operations. Peaks in RAM usage usually follow, depending on the size of the join result. In both cases, the workload is distributed to prevent resource exhaustion that would lead to system interruption (see complete resource monitoring in Supp. Table S2).

Use cases

Pan-transcriptomes and rare variants

Circular RNAs MAN1A2 and FBXW4 were previously described in [11]. In this study, circRNAs were identified through total RNA extraction, poly(A) depletion to enrich for pre-mRNA (including introns and lariats), RNAse R digestion of linear RNA and subsequent validation by sequence mapping. Using Vizitig, we located alternative circular backsplicing junctions for both genes by querying nodes connecting the reported junctions and extracted the corresponding sequences, which had not been documented in the original publications. To demonstrate integration of external data, we incorporated predictions from circInteractome [5], which identifies miRNA binding sites on circRNAs. For MAN1A2, predicted sites were matched to the graph through sequence queries, and conserved sites were located within the Vizitig graph. Additional MAN1A2 and FBXW4 junctions appear in Supp. Fig. S8, S9

We next investigated the HM13::TPX2 human gene fusion transcript, reported in the Cancer Cell Line Encyclopedia (CCLE) and DepMap/COSMIC fusion database [2].

Fusion junction coordinates were obtained from STAR-Fusion [6] alignments provided by DepMap, following guidelines to select events with high reliability. A de Bruijn graph (k=61) was constructed from two control RNA-seq datasets (SRR8615240, SRR8615242) and one dataset known to harbor the fusion event (SRR8616107). Reference transcripts and abundance data were annotated similarly to the previous example. Using the fusion breakpoint sequence as a locus query within Vizitig, capable of querying sequences both shorter and longer than the k-mer size, we identified the exact node corresponding to the fusion junction (Figure 2). This node, not associated with known genes or annotations, remained distinctly uncolored. Applying the `Sashimi` action revealed expression specificity of the fusion to the relevant dataset (Figure 2), confirming its sample-specific presence. Beyond single events, Vizitig allows interactive expansion of the graph around a locus to reveal neighboring structures, exemplified by the detection of an additional BCL1L2::HM13 fusion in the same dataset, also reported in DepMap (Supp. Fig. S7). Building on this, we examined the CYTH1::EIF3H event in the SKBR3 cell line, previously characterized in [13]. Although that study combined long- and short-read sequencing, the fusion sequences themselves were not directly available. Vizitig enabled their retrieval and visualization, showing long-read support for the breakpoint and identifying bridging nodes between the two genes, including one of the reported breakpoints (Supp. Fig. S7 (b)).

Extraction of sequences of interest in pangenomics

We constructed a yeast pangenome by combining the *S. cerevisiae* reference strain S288C (with annotations) and nine additional assemblies from a recent study of 119 high-quality, telomere-to-telomere genomes [12], which characterized structural variation among strains. Annotation data from the S288C GFF3 file were incorporated to enable direct feature queries. We selected the YHR177W gene, reported to carry multiple structural variants in the original study, and demonstrated the levels of complexity Vizitig can expose, from a linear view of the gene to inter-strain variation and conserved regions. Vizitig can automatically display strain support for variants (critical when analyzing many strains) or restrict the view to a subset of interest, focusing on biologically meaningful differences. It also supports selective extraction of graph segments and sequences. As an example, we selected a strain with documented structural variants, visually confirmed in Vizitig, and extracted the sequences corresponding to its haplotype (see Supplementary Results). We obtained similar results for YDR18C (Supp. Fig S5). Finally, we illustrate a complementary use case by integrating long reads into a *C. elegans* pangenome, where they augment two reference strains (Supp. Fig S6).

Availability and implementation

Vizitig is available as an open-source software, and documentation can be found at: `https://gitlab.inria.fr/vizisoft/vizitig`. All the Vizitig graphs and instructions to reproduce experiments are available at: `https://zenodo.org/records/17136231`.

Funding information

This work is funded by ANR Find-RNA ANR-23-CE45-0003.

Acknowledgements

The authors would like to thank Margaux Mouton, Loup Lobet, Cheyma Karbiche and Loïc Marmey who helped testing Vizitig v0.$*$ to 1.0, and Bruno Guillon for his help regarding NetworkDisk. They would like to thank proofreaders including Thérèse Commes, Rayan Chikhi and Lucien Piat who helped greatly with the manuscript.

References

[1] Anton Bankevich et al. “SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing”. In: *Journal of computational biology* 19.5 (2012), pp. 455–477.

[2] Chloé Bessière et al. “Transipedia. org: k-mer-based exploration of large RNA sequencing datasets and application to cancer data”. In: *Genome Biology* 25.1 (2024), p. 266.

[3] Rayan Chikhi, Yoann Dufresne, and Paul Medvedev. “Constructing and personalizing population pangenome graphs”. In: *Nature methods* 21.11 (2024), pp. 1980–1981.

[4] Rayan Chikhi, Antoine Limasset, and Paul Medvedev. “Compacting de Bruijn graphs from sequencing data quickly and in low memory”. In: *Bioinformatics* 32.12 (2016), pp. i201–i208.

[5] Dawood B Dudekula et al. “CircInteractome: a web tool for exploring circular RNAs and their interacting proteins and microRNAs”. In: *RNA biology* 13.1 (2016), pp. 34–42.

[6] Riaz Gillani et al. “Gene fusions create partner and collateral dependencies essential to cancer cell survival”. In: *Cancer research* 81.15 (2021), pp. 3971–3984.

[7] Andrea Guarracino et al. “ODGI: understanding pangenome graphs”. In: *Bioinformatics* 38.13 (2022), pp. 3319–3326.

[8] Eef M Jonkheer et al. “PanTools v3: functional annotation, classification and phylogenomics”. In: *Bioinformatics* 38.18 (2022), pp. 4403–4405.

[9] Mikhail Karasikov et al. “Efficient and accurate search in petabase-scale sequence repositories”. In: *Nature* (2025), pp. 1–9.

[10] Yarden Katz et al. “Quantitative visualization of alternative exon expression from RNA-seq data”. In: *Bioinformatics* 31.14 (Jan. 2015), pp. 2400–2402. ISSN: 1367-4803.

[11] Tim R Mercer et al. “Genome-wide discovery of human splicing branchpoints”. In: *Genome research* 25.2 (2015), pp. 290–303.

[12] Samuel O’donnell et al. “Telomere-to-telomere assemblies of 142 strains characterize the genome structural landscape in Saccharomyces cerevisiae”. In: *Nature Genetics* 55.8 (2023), pp. 1390–1399.

[13] Qian Qin et al. “CTAT-LR-fusion: accurate fusion transcript identification from long and short read isoform sequencing at bulk or single cell resolution”. In: *BioRxiv* (2024).

[14] Helga Thorvaldsdóttir, James T Robinson, and Jill P Mesirov. “Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration”. In: *Briefings in bioinformatics* 14.2 (2013), pp. 178–192.

[15] Ryan R Wick et al. “Bandage: interactive visualization of de novo genome assemblies”. In: *Bioinformatics* 31.20 (2015), pp. 3350–3352.