Visualizing Structural Artifacts in Long-read Transcriptomic Sequencing

Baraa Orabi (borabi@cs.ubc.ca) and Nico Ritschel (ritschel@cs.ubc.ca)

The University of British Columbia

1 Introduction

Technological advances in biology research have made it easier than ever before for researchers to obtain large amounts of measurement data. While this development certainly had a positive overall impact on the quality and efficiency of research, it has also created a new challenge, which is to effectively handle and interpret the amounts of collected data. A field where this effect is particularly noticeable is that of genetics, where the ability to read longer DNA sequences at a time, combined with the reduced cost of doing so, has led to an explosion of the amount of available data.

While automated tool-chains provide fundamental support for biologists on a lower level of data processing, the task of interpreting and contextualizing measurements is still highly reliant on human expertise. To work effectively, it is essential for these experts to be able to quickly switch between a coarse overview of the available data-set and a fine-grained analysis of specific data ranges of interest. Modern visualization tools are available for many types of analyses, yet of them are sufficiently suited for the analysis of long-read sequencing data to identify structural variations. This work aims to create a visualization that matches the specific needs of researchers working on this type of analysis task.

Due to his experience in the field of bioinformatics research, Baraa Orabi can support this project both in the visualization design and as a domain expert. He worked with the data for a few months over the last year. This, as well as consultations with members of the Dr Faraz Hach lab at the Vancouver Prostate Centre, prompted the task abstractions in this proposal.

2 Background

Living organisms use DNA to encode genetic information. DNA molecules are chemical molecules and are composed of repeating submolecules known as *nucleotides (nt)*. There are four main types of nucleotides that are often represented with the letters **A**, **C**, **G**, and **T**. Thus, any DNA molecule can be abstractly thought of as a sequence written using a very limited alphabet containing only four letters. The collection of all DNA molecules in a cell is known as the genome. Genome sequences can become very long; for example, each human cell contains three billion nucleotides worth of DNA molecules.

Most of the functions in living organisms are performed by *proteins*. Proteins are molecules that have specific chemical structures essential for their function. The structure of every protein is determined by a *gene*. Genes are specific substrings of the genome. The human genome contains about 20,000 genes which combined constitute about 2% of the whole genome.

To translate a gene to a protein, a copy of the gene called *transcript* is created. However, some parts of the transcript can be *spliced out*, meaning that they are removed before the transcript is translated into a protein. Different transcripts of the same gene can be spliced in different ways. Those alternatively spliced transcripts of the same gene are known as *isoforms*. Alternative splicing allows one gene to encode for multiple proteins with similar chemical structures but potentially completely different functions. The *transcriptome* is the collection of all isoforms sequences in a given cell. The human transcriptome contains about 80,000 isoforms, averaging four isoforms per gene.

Sequencing is a technology that takes real DNA molecules and outputs digital files containing the nucleotide by nucleotide sequence of these different molecules. Ideally, a single DNA molecule is *sequenced* (read) end-to-end with no sequencing errors. However, reads generated by sequencing technologies are far from perfect. *Next-Generation Sequencing (NGS)*, the most commonly used class of sequencing today, generates short reads with lengths ranging from 75nt to 250nt and has a sequencing error rate of 2 to 4 errors per 1000nt [5]. The reasonably low sequencing error rate of NGS reads makes them a powerful tool to detect single nucleotide mutations in a sample. On the other hand, the short length of NGS reads makes them difficult to utilize to identify larger structural mutations such as large deletions, insertions, and inversions.

Long-read Sequencing (LRS) is a newer class of DNA sequencing technologies. LRS read lengths can reach thousands to tens of thousands of nucleotides, orders of magnitudes more than NGS reads. However, LRS suffers from very high sequencing error rate of 10-20% [6]. The high sequencing error makes using LRS to detect mutation of single nucleotides challenging, but its long read sizes make it a suitable tool for identifying structural variations.

Bioinformatics pipelines can automatically generate predictions of genetic relations based on reads, even in the presence of high sequencing error rates. For example, a tool for isoform discovery can use read alignments to predict the structure of thousands of gene isoforms that are present in the sample. Each of these isoform predictions is based on a small subset of reads that provide empirical support for the resulting isoform prediction. While automated predictions are an essential tool for researchers in the field, the results still require manual interpretation. Besides validating them, humans still need to draw conclusions on a higher level and identify the key predictions from the thousands of potential results.

3 Related Work

State-of-the-art visualization tools for sequencing data are designed for NGS short read. For example, *Integrative Genomics Viewer (IGV)* stacks as many short reads in a single view as possible [4]. IGV also restricts the base grid of the view to the reference sequence since NGS reads are not long enough to be a possible alternative grid.

However, this limitation might not be desired for LRS data. Long reads are long enough to be treated as reference themselves even without assembly. For example, if the sample contains a novel structure (not present in reference), LRs would have that in common, and current views don't allow for immediately showing this.

4 Data

The data we use in this project transcriptomic LRS data. In addition to the LRS, we will use the reference gene and transcript sequences from *ENSEMBL* database [2]. Finally, we will also make use of LRS data transformation using *minimap2* genome alignment and mapping tool [3].

Dataset	# of reads	Length (nt)		Throughput (Gnt)
		Mean	Median	
WGSC dRNA	10,302,647	906	677	∼10.0
WGSC cDNA	15,152,101	1,030	771	∼14.0
VPC cDNA	3,361,630	1,587	1229	∼3.3

Table 1. Statistical summary of the three LRS datasets.

4.1 Transcriptomic LRS

We will use three datasets. The first two are publicly accessible datasets from Nanopore Whole Genome Sequencing Consortium (WGSC) [7]. Both WGSC dataset includes LRS of the transcriptome of NA12878, a well studied human cell line. However, one uses direct RNA (dRNA) based sequencing technology and while the other uses complementary DNA (cDNA) sequencing technology. dRNA LRS generates slightly longer, more accurate reads than cDNA LRS.

The third dataset is a private dataset from our collaborators at the Vancouver Prostate Centre (VPC). The VPC LRS dataset is generated from the transcriptome of 22Rv1, a prostate cancer cell line and is generated using a cDNA approach.

Table 1 summarizes the main statistics of the datasets.

4.2 Gene reference sequence and annotations

We have human gene reference sequences and annotations from ENSEMBL. The reference includes over 20,000 protein-coding genes with over 80,0000 isoforms. The reference genome and transcript sequences will be used as a default grid to align the reads to.

4.3 Read alignment

We will use minimap2 for alignment and mapping. Minimap2 is a fast sequence aligner and mapper that utilizes a number of indexing techniques to achieve near-linear alignment (in terms of the read size). When aligning reads to reference gene sequences we will be using splice-aware alignment to ensure robustness against gene splicing. We will also use minimap2 to perform all-vs-all alignment of the reads which will be helpful when switching the base reference grid.

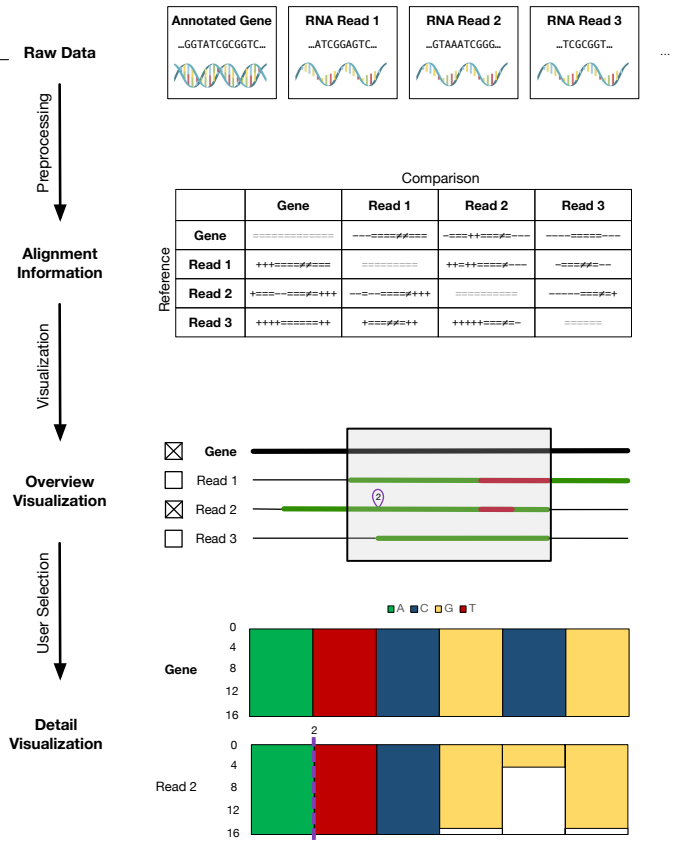

Fig. 1. Proposed visualization pipeline and mock-up. Read lengths and numbers were chosen for illustrative purposes only and would be larger in practice.

5 Task Abstractions

Based on talks with domain experts and our own experience, we have defined the most frequently occurring tasks when these users work with long-read sequences.

5.1 Selecting genes and reads

A typical data-set can contain millions of reads that can be mapped onto thousands of genes. To allow users to gain any meaningful insights from this amount of data, they need to be able to select a single gene to focus on. When LRS technology is used for sequencing, each gene has on average a few hundred reads, from which either an automated tool or the user needs to select a subset that is relevant for the desired analysis task.

5.2 Observation and comparison of indels

The most common type of sequencing error that occurs in LRS are insertions and deletions of one or multiple nucleotides, often called *indels*. While existing bioinformatics tools can identify the positions of indels and handle them automatically when aligning reads, users should still be able to view indels to validate the alignment and judge the overall quality of a read. For this purpose, users should be able to quickly infer indel sizes visually when examining and comparing reads, requiring potential filtering of minuscule indels based on a threshold size.

5.3 Contracting empty regions

Genes often have large regions that are always spliced out when they are transcribed. Some genes can have lengths up to hundreds of thousands of nucleotides of which only a few thousand are found in transcripts. While the presence of spliced out regions is a relevant observation for experts, there is typically no need to display them true to their size in proportion to the overall gene. Contracting them, either automatically or manually, would allow experts to focus on the information that is actually relevant to them.

5.4 Identification of motifs of interest

Certain gene patterns that have a particular meaning to domain experts. An example are *poly-A tail* sequences that mark the end of a gene and that, if they are present in the middle of a read, indicate an sequencing error. Another example is the presence of adapter sequences of the library preparation stage which are expected to be present at the start of the read. Other, similar sequences that should be easily visible to experts.

5.5 Viewing the nucleotide sequence and quality

When inspecting a small number of reads, users may be interested in viewing and comparing the concrete underlying base sequence of the reads and the gene. In addition, users may be particularly interested in the quality of the sequence data. For this purpose, the *Phred* quality score that tells users the likelihood of a single base being assigned correctly can be particularly helpful to dismiss some predictions (e.g. if some reads all say there is an insertion but they all have low quality for the insertion sequence then that might just be a systematic error). Phred score is a log-based score that ranges from 1 to 60. A score of 60, the highest possible score, mean $1/10^6$ chance of being wrong. LRS quality typically does not exceed 16 (2% chance of being wrong.)

5.6 Inspecting the raw signal

On an even more fine-grained level, for example when interpreting the Phred quality score of a single read, users may be interested in inspecting the raw signal that was used to generate the base assignment

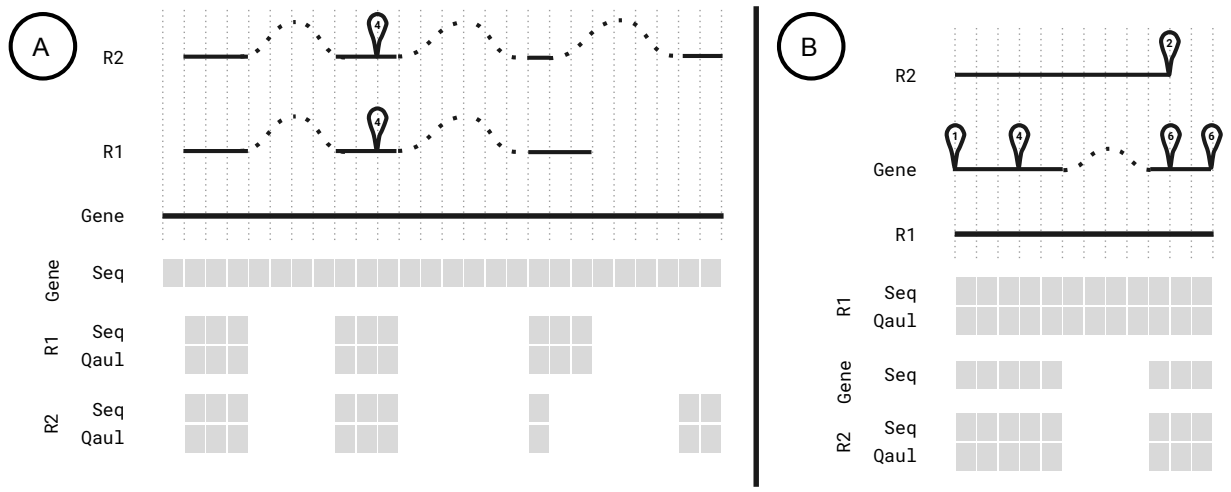

Fig. 2. Mock-up of target switching from the gene (A) to a read R1 (B). Both the overview and detail visualization need to be updated for the switch.

for the nucleotide sequence. This requires the raw signal to be visually aligned positions on the sequence.

5.7 Target switching

One of the main targets of inspecting long reads is the identification of structural mutations. While using the annotated gene as a reference is ideal for identifying many of these mutations, it ignores insertions of nucleotides that are present in reads but not in the annotated gene. If the same insertions are present in multiple reads, they might provide additional insights beyond being noise. Therefore, users might have an interest in switching the target sequence from the gene to one of the reads and compare it to other reads.

6 Proposed Solution

Based on our previously presented task abstractions, we propose a data processing pipeline and a visualization design with the goal of enabling users to execute most of these tasks as effectively as possible. Due to the limited scope of this project, we have excluded the gene selection task, which constitutes its own, mostly independent visualization challenge. Instead, we assume that users have already preselected a single gene of interest and a moderately sized subset of tens, but not hundreds, of reads they consider relevant for them.

6.1 Design

Figure 2 shows the proposed pipeline for processing raw data and creating a visualization based on it. In the first step, raw data containing the reference gene and read sequences is pre-processed using existing bioinformatics tools to generate alignment information. Notably, alignments of sequences are computed relative to a reference gene or read, and this relation is not necessarily symmetric. This means that for a single gene, the number of possible comparisons grows quadratically with the number of considered reads.

In our proposed visualization, the computed alignment information is first visualized as an overview to allow users to examine and compare reads and their relation to the annotated reference gene. Figure 2 shows a mock-up of this overview visualization, using the annotated gene as a reference and highlighting aligned sequences in green and mismatches in red. Inserted nucleotides are marked by a purple loop that is labelled with the number of insertions represented by it. Gaps in the reads are shown as a thinner, black line.

We consider an overview visualization as proposed here ideal for the task of comparing reads and their insertions and deletions. Allowing user interactions can further enable them to contract empty region by selecting the region and using a button or keyboard shortcut, as well as to identify motifs of interest that can be highlighted in a different colour if they occur in a read. In addition, the view can support target switching by double-clicking on a read to make it the new reference sequence. Figure 2 shows a mock-up specifically of the before-after view when the target sequence is switched from the gene to the read R1.

To gain a view of each read's raw signal and quality, users can select a small subset of reads and a range of the gene for a detailed comparison. The resulting detail view is shown in Figure 2 visualizes the different bases as a colour-coded bar chart. The bars, one for each nucleotide in the sequence from left to right, represent the quality score for each base's assignment with their height. For the known reference gene, this quality is perfect for all bases and therefore the height is always maximal for all nucleotide. For reads however, experts should be able to identify mismatches based on sequencing errors by their reduced quality, such as for the 5th bar of Read 2 in this example.

To allow users to compare nucleotide sequences when more than one sequence is selected, we consider it important to always have all detail views horizontally aligned with each other. This however causes issues when one sequence has deletions or insertions compared to the reference. Deletions can be reasonably visualized as gaps in the chart, however insertions require a different form of visualization. Figure 2 shows an insertion represented by a dotted purple line labeled with the number of nucleotides. To give users access to the inserted sequence without switching the target sequence, we are however also considering a mouse-over pop-up showing the inserted sequence. For viewing the raw signal, we consider either an optional overlay over the detail view or, in case of visibility issues, separate, aligned plot below each sequence's bar chart.

6.2 Implementation

To implement our proposed visualization, we intend to build a two-stage pipeline: To pre-process raw sequence data from reads and the annotated gene, we plan to use the existing sequence alignment tool *minimap 2* [3]. Using this data, we then use the D3.js framework [1] to visualize both the alignment information as well as the raw sequence data using a bar chart as well as custom drawing.

We identified two implementation challenges that we cannot evaluate without having a functional prototype: The first is the time required by the alignment software to produce the derived data necessary for our visualization. To allow real-time switching between target sequences, which requires a re-computation of all alignments, we might consider pre-computing and caching the alignment data for all possible alignment combinations. The second challenge is the amount of data that needs to be handled by D3.js. While we assume that we never have more overall data points to visualize at once than tens of thousands, we might have to use further optimizations like data simplification to manage the performance of our visualization.

Task	Description	Hours/Person	Completion
Definition	Discussion of project idea, biological background and brainstorming about possible design directions	6	Oct. 27th
Proposal	Writing of the project proposal document and creating a first mock-up of the proposed visualization	8	Nov. 4th
Stage 1 Prototype	Completion of a first functional prototype of the visualization for a fixed data set and with limited support for navigation and target switching	15	Nov. 17th
Stage 1 Case Study	Gathering feedback on the first prototype by demonstrating it to 1-2 users of the intended target audience	4	Nov. 20th
Stage 2 Prototype	Completion of a second prototype that supports user interaction to navigate and switch targets	15	Nov. 29th
Stage 2 Case Study	Gathering more feedback on the second prototype by asking 1-2 users of the intended target audience to use the prototype for a given task	4	Dec. 4th
Final Polish	Incorporating feedback from the second case study and potential visualization of more motifs of interest	5	Dec. 7th
Final Presentation	Preparation of the final presentation and the code for submission	5	Dec. 10th
Final Report	Write-up of the final report	10	Dec. 13th

Table 2. Proposed project timeline

7 Milestones and Schedule

Table 2 shows our intended milestones for the project. To allow us to evaluate our visualization prototype with domain experts in an early stage, we plan to split development into two stages: In the first stage, we implement only the static core of our visualization that allows users to view a selected gene and a given number of reads. While we intend to implement the detail visualization in this stage as well, navigation and synchronization between the two views may be limited. By showing our intermediate result to a small number of domain experts, we hope to get early feedback on how our visualization meets their requirements and expectations. In the second stage, we then plan to implement the remaining user interactions and allow users to switch targets. Through a second case study with experts, we expect to receive more feedback on our fully implemented visualization that we can then use for both minor polishing as well as laying out potential future work. Depending on the time constraints of the project, we may also implement additional features like the visualization of selected motifs of interest in the examined data.

References

[1] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

[2] S. E. Hunt, W. McLaren, L. Gil, A. Thormann, H. Schuilenburg, D. Sheppard, A. Parton, I. M. Armean, S. J. Trevanion, P. Flicek, et al. Ensembl variation resources. *Database*, 2018, 2018.

[3] H. Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.

[4] J. T. Robinson, H. Thorvaldsdóttir, W. Winckler, M. Guttman, E. S. Lander, G. Getz, and J. P. Mesirov. Integrative genomics viewer. *Nature biotechnology*, 29(1):24, 2011.

[5] M. Schirmer, R. D'Amore, U. Z. Ijaz, N. Hall, and C. Quince. Illumina error profiles: resolving fine-scale variation in metagenomic sequencing data. *BMC bioinformatics*, 17(1):125, 2016.

[6] R. R. Wick, L. M. Judd, and K. E. Holt. Performance of neural network basecalling tools for oxford nanopore sequencing. *Genome Biology*, 20(1):129, 2019.

[7] R. E. Workman, A. Tang, P. S. Tang, M. Jain, J. R. Tyson, P. C. Zuzarte, T. Gilpatrick, R. Razaghi, J. Quick, N. Sadowski, et al. Nanopore native rna sequencing of a human poly (a) transcriptome. *BioRxiv*, page 459529, 2018.