SUPPLEMENTAL METHODS

BIOSURFER FOR SYSTEMATIC TRACKING OF REGULATORY MECHANISMS LEADING TO PROTEIN ISOFORM DIVERSITY

1. Retrieving information

1.1 Integrating the input data

Biosurfer reads input data from several files: (i) transcript FASTA file for nucleotide sequences, (ii) protein FASTA file for amino acid sequences, and (iii) GTF file for exon structures, coding regions (CDS), and transcript coordinates. These input files are required to capture information about transcript isoforms and their corresponding protein products. Functions from the database modules are used to manage the SQLite3 database and load data from GTF and FASTA files.

1.2 Populating SQLite Database

The Biosurfer utilizes SQLite database to organize genomic information extracted from input files (GTF, transcript FASTA, and protein FASTA). Upon loading data using functions like *load_gencode_gtf*, *load_pacbio_gtf* (for GTF files generated from Long-Read Proteogenomics pipeline (1)), *load_transcript_fasta*, and *load_translation_fasta*, the database is populated with tables representing genomic entities. These tables include Chromosome, Gene, Exon, ORF (Open Reading Frame), Transcript, and Protein, each designed to store specific attributes such as gene accession IDs, transcript names, genomic coordinates, sequence information, and feature annotations. Relationships between entities are established through foreign keys, such that data integrity is ensured.

2. Multi-layered comparison of isoforms

Biosurfer compares pairs of transcript-, ORF-, and protein-level sequences for each gene. For each gene, one isoform is selected as reference and the other isoforms are denoted as alternative. The designation of the reference isoform is user-defined. We first align the transcript sequences based on transcript coordinates and generate t-blocks (see Section 2.1). Then, we align the open reading frames (ORFs) at codon-centric level (built on t-blocks) and create c-blocks (see Section 2.2). Finally, we align the protein sequences by grouping c-block information to define p-blocks (see Section 2.3). These three block types define the alternative splicing events between the reference and alternative isoforms at three different levels.

2.1 Transcript-centric alignment (t-blocks)

The transcript-centric layer represents the alignment of a pair of transcript isoform sequences. The basic unit of each sequence in this layer is a nucleotide. The transcripts of the isoform pair are assumed to have high-quality alignments to a reference genome, so one can use the genome coordinates to align the reference and alternative isoforms at the transcript level. The aligned nucleotide locations are compared, so that ranges of contiguous segments of the transcript that are present in one, the other, or both isoforms are found. These shared or unique contiguous segments are called t-blocks. Note that a t-block can include several exons that are separated by introns in genomic coordinates. The t-blocks are defined by an algorithm that models each transcript as a bi-partite graph of matching or partially overlapping introns, as well as custom functions to classify the patterns (see **Supplementary File 1 step 2** for more details).

The t-blocks are categorized into three types. If both, reference and alternative, isoforms have the same t-block, that t-block is defined a Match t-block. If the t-block is exclusive to the reference isoform and absent in the alternative isoform, that t-block is defined a Deletion t-block. If the t-block is exclusive to alternative isoform and absent in reference isoform, that t-block is defined an Insertion t-block. The variations observed at transcript level include: altTSS, exon skipping, exon alternative donor, exon

alternative acceptor, intron retention, and APA. The code classifies alternative splicing events into two classes: basic transcript event class and compound transcript event class. A basic transcript event class includes intron splice events, donor splice events, acceptor splice events, exon splice events, and exon bypass events. A compound transcript event class includes combination(s) of several basic transcript events (*e.g.*, exon skipping in addition to alternative donor, see **Supplementary File 1 step 3** for more details).

2.2 Codon-centric alignment (c-blocks)

The codon-centric layer represents an alignment of open reading frames (ORFs) of the reference and alternative isoforms. The basic unit of each sequence in this layer is a codon. The alignment for this layer is based on the transcript alignment done in the previous step. Specifically, we first convert the coordinate of ORF region on each transcript into a 0-based coordinate. The first adenine of the AUG is considered index 0, and the last index is the last nucleotide of the stop codon (e.g., A of TGA). Therefore, the coordinates in 5' UTR are negative. The relative position of reference and alternative sequences remains the same as it was previously aligned at the transcript level.

To compare reference and alternative sequences at the codon level, each t-block needs to be further sub-segmented and sub-classified based on the translational status. We created temporary objects called translation status-annotated t-blocks (TSA t-blocks) using the previously created t-blocks. These TSA t-blocks are sub-segmented based on the translational status of each nucleotide (see **Supplementary File 1 step 6** for more details). Any TSA t-blocks that are not translated are discarded. Each remaining TSA t-block is converted into a c-block for further classification.

The classification of c-blocks is the most complicated one in Biosurfer, but it carries information from transcriptional to translational level, therefore it is the most critical in this pipeline. The c-blocks are classified based on the codon pairs between reference and alternative sequences. Therefore, we first need to classify each codon pair based on the alignment. The c-blocks are formed by aligning the ORFs based on overlapping codons in genomic space and systematically comparing codons from the reference and alternative isoforms.

The classification of codon pairings between the reference and alternative sequences is based on the following a four-step protocol that analyzes the mutual position of the codon as well as their base overlaps. **First**, we detect all pairs of codons that are identical and their positions match (Table1, Supplemental Figure S3A). We call them Match codons. Other simple cases include Deletions codon—those in which the reference codon matches to an empty 'placeholder' codon in the alternative isoform, and Insertion codons — those in which the alternative codon matches to an empty 'placeholder' codon in the reference isoform.

Second, we detect the contiguous codons that are frameshifted and have a 2-base overlap. **Third**, we determine the codons that are split and only partially overlapped (1 or 2 bases, see Table1, Supplementary Fig S3B and C). When genomically aligning codons from the reference and alternative isoforms, if a codon is split and only partially overlaps with its counterpart, Biosurfer employs a strategy using placeholder codons. These placeholders are temporary markers within the code, that help to correctly position the codon blocks despite partial overlaps.

The pairs of codons that are annotated during the above three steps are referred to as "paired" codons. **Last**, we collect all other cases, for which the pairs of codons are referred to as "unpaired" (Table1, Supplementary Figure S3B). Overall, the paired and unpaired codons are classified into 9 categories

based on their translation status, frameshift status, and codon topology (Table 1). We will discuss how Biosurfer treats the split codons in more details in Section 2.2.1.

After full classification of every paired codon across the two isoforms, we classify the c-blocks. Specifically, we group the paired codons wherein any contiguous stretch of the same category (*e.g.*, Match codon, Insertion codon, or Edge codon) into codon alignment blocks or c-blocks. The pseudocode for codon-centric alignment is provided below:

1. **Traversing t-blocks:**

   For each t-block:

   - Determine the corresponding codon alignment category

     -- Codon alignment categories are defined within the 'CodonAlignment' class

     -- Within each t-block the corresponding codon alignment category is determined based on the categorization of that t-block (Match, Deletion, or Insertion)

   - Map the c-block to the appropriate t-block

2. **Codon categorizing (Match, Insertion, Deletion, Translated, Untranslated):**

   For each c-block:

   Determine its category based on its properties (if the c-blocks are match or have same genomics position); the code prioritizes complete 3/3 positional matches when categorizing codons:

   - Determine ORF boundaries of the isoforms based on the sequences
   - Match: t-blocks are Match, and both sequences are within ORF boundaries
   - Translated: t-blocks are Match, but reference sequence is out of ORF boundaries
   - Untranslated: t-blocks are Match, but alternative sequence is out of ORF boundaries
   - Insertion: t-blocks are Insertion
   - Deletion: t-blocks are Deletion

3. **Frameshift detection (Frameahead, Framebehind):**

   For each frameshift block detected:

   - Determine if it is ahead or behind the reading frame
   - Overhang calculation

     -- Compare overhang of ref and alt to check if it is Frameahead or Framebehind

```
        -- If the difference results in an overhang of 1 or 2 codons in
        ref compared to alt, then it is Frameahead

        -- If the difference results in an overhang of 1 or 2 codons in
        alt compared to ref, then it is Framebehind
```

4. Codon boundary adjustment:

- Calculate protein coordinates and infer codon categories for each aligned c-block
- Store boundaries, overhangs, and categories for adjustments
- Iterate to adjust block boundaries based on overhangs and categories
- Merge consecutive c-blocks of the same category

5. Edge case handling:

- Incomplete codons are determined by the overhang, mostly near ORF boundaries or in case of frameshifts
- Use placeholders to accurately align the c-blocks
- This is done in such a manner that it maintains the mapping between the c-block and corresponding t-block

2.2.1. Identification and characterization of split codons in Biosurfer

Junctions divide exons that code for a protein. Three bases that form a codon (the emphasis is on the codon, since it gets translated to an amino acid residue) can be distributed differently by the junction site. Furthermore, at the junction, there can be different combinations of nucleotides from a codon that overlap between the reference and the alternative. The junctions with the offset phase 0 coincide perfectly with codon boundaries, which we refer to as 'contiguous codons'. The junction with phases 1 or 2 lies in between the codons which we refer to as 'split codons'.

2.3 Protein-centric Alignment (p-blocks)

The last data structure, p-block, represents a protein-centric layer defined through t- and c-block guided comparisons of two protein isoforms. The basic unit of each sequence in this layer is an amino acid residue. We first classify p-blocks based on the changes happening between the reference and alternative protein products as Match, Insertion, Deletion, or Substitution. Note that the Substitution p-blocks must arise from a combination of insertion/deletion/frameshift events found at the c-block level. Overall, the changes in p-blocks are agnostic to the upstream mechanism; however, at the same time, the upstream mechanisms can be retrieved and analyzed within Biosurfer, unlike traditional protein aligners, which lack such context. This tracking ability is due to the fact that c-blocks are grouped into p-blocks in a "bottom-up" manner, from the algorithmic perspective, thus maintaining precise links between the protein layer and the underlying codon and transcript layers.

However, at a more granular level, these affected regions are typically the flanked paired codons, which are interposed between the transition of any two p-block types. Such *c-blocks*, which comprise a single paired codon, must be further examined to determine if the identity of the paired AA residues at such boundaries is the same or different. Therefore, for such flanking codons, which we term Edge Codons, we

must determine their AA identity. If the AA identities match, then the c-block is grouped with the adjacent Match p-block. If the AA identities differ, then the c-block is grouped with the adjacent non-Match p-block, which now has a tag as containing a “ragged” codon.

In particular, the Insertion and Deletion p-blocks that contain one or more “ragged” Edge Codons are not polypeptide insertions/deletions, strictly defined, per say, but practically speaking, we still classify them as Insertions/Deletions at the broad p-block category level with a “ragged” codon attribute (see **Table 1**).

3. Information analysis and visualization

Biosurfer generates summary tables for c-blocks and p-blocks derived from open reading frames (ORFs) , representing codons and amino acids from regions that differ between each pair of query isoforms. These tables, which include annotations such as associated splicing mechanisms (see **Figure 1**, bottom panel), are a result of a multi-layered isoform alignment process. The data structures used for isoform comparison at the t-block, c-block, and p-block layers are used in creating these summary tables. The c-block summary table incorporates information on obtained t-blocks, listing respective blocks with annotations across different categories for a pair of reference and alternative isoforms. Additionally, Biosurfer annotates events such as alternative transcription, splicing, and polyadenylation events, elucidating the transcriptional processing events leading to transcript variation. The example summary table outputs for GENCODE v42 and WTC11 can be found in **Supplementary Tables S3, S4, S11, S12**.

Biosurfer's plotting module, leveraging Python's Matplotlib library, enables the simultaneous visualization of transcript structure, ORF boundaries, and altered protein regions across isoforms of a gene (**Figure 1**, bottom panel). The APPRIS principal isoform serves as the reference for comparison via the multi-layered isoform comparison. This module incorporates customizable features for track spacing, exon-intron visualization, and legend annotations, allowing it to represent intron-exon boundaries, start/stop codons, splice events, frameshifts, and codon/protein alignment blocks in a clear and informative manner.

4. References

1) https://github.com/sheynkman-lab/Long-Read-Proteogenomics