Workflow Diagram (Sequins analysis based on truth)

Friday, 8 January 2021 12:24 AM

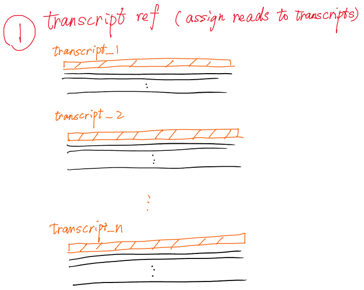
Input:

Basecalled reads (.fastq)
Squiggle (.fast5)
Reference genome (.fa)
Reference transcript (.fa)*
Transcript annotation(.gtf)*

NOTE:
In Sequins(spike-in dataset), the **Reference transcript** or **transcript annotation** are the ground truth of the Spike-ins. In real data analysis, they can be available but not necessarily the ground truth.

1. Get ground truth for splice site in each read

In the spike-in dataset, there are 2191749 reads in total. To get the ground truth of how the original mRNA of each read is spliced, I first mapped the reads to transcript reference. The transcript reference provides the true sequence of each transcript. The resulting BAM file provides mappings from each read to their transcript. Out of the 2191749 reads, 1914214 of them were successfully mapped, which creates 2539244 alignments (each mapped read must have one primary alignement and could have multiple secondary alignment). To ensure the reads are mapped to the correct isoforms with high confidence, I set a relatively high **mapping quality threshold (60),** which results 1551946 alignments passed the threshold. No secondary alignment passed the threshold, which means the remaining 1551946 alignments come from the mapping of the same number of reads.



```
samtools view -F 4 -F 256 sequins_barcode01.sorted.bam | cut -f1 > readid_genome.txt
samtools view -F 4 -F 256 -q 60 transcript_map.sorted.bam | cut -f1 >readid_trans.txt
grep -Fxf readid_genome.txt readid_trans.txt > read_id_inter.txt
samtools view -F 260 sequins_barcode01.sorted.bam | python3 idfilter.py read_id_inter.txt >sequins_barcode01.sorted.inter.sam
samtools view -F 260 -q 60  transcript_map.sorted.bam | python3 idfilter.py read_id_inter.txt > transcript_map.sorted.inter.sam
{ samtools view -H sequins_barcode01.sorted.bam ; cat sequins_barcode01.sorted.inter.sam ; } | samtools view -b >
sequins_barcode01.sorted.inter.bam
{ samtools view -H transcript_map.sorted.bam ; cat transcript_map.sorted.inter.sam ; } | samtools view -b >
transcript_map.sorted.inter.bam
samtools index sequins_barcode01.sorted.inter.bam
samtools index transcript_map.sorted.inter.bam
```

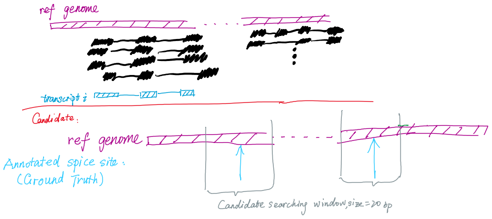
2. Minimap2 analysis

To assess how accurate the splice sites can be identified by minimap2, I mapped the nanopore reads to the reference genome. Although minimap2 can take annotations as actual information to increase the mapping quality, we decide not to do so since there is no real world example of a perfect annotation as sequins, which may inflate the accuracy of splice site identification from minimap2. Mapping all 2191749 reads to the reference genome of sequins results in 1958069 alignment, in which 1957845 are primary alignment. Only primary alignments will be considered in this analysis, since we will need to take only one alignment for each read and the primary alignment will be the best one. In this case, the number of alignment is then matches the number of reads. To assess whether a splice site is correctly mapped, the true position of splice site need to be known. Therefore, we consider only the reads that have confident mapping (mapQ = 60), which results in 1551691 reads left. Among the 1551691 reads, 3578100 junctions within reads have been found after the mapping.

3. NanoSplicer analysis

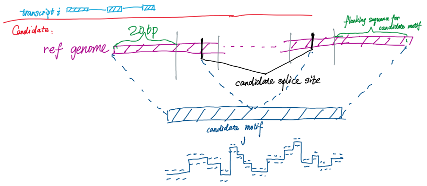
Step 1: Find candidate splice site

To assess whether or not the squiggles have information of which splice site has been used, I identified the exon junctions from the annotation.gtf, which contains the true coordinate of each splice site. The candidates were obtained from candidate searching windows (size = 20) centered at the true pair of donor and acceptor sites. The candidate splice sites are obtained by searching "GT" pattern in the donor site candidate searching window and searching "AG" in the acceptor site candidate searching window.



Step 2: obtaining candidate squiggles

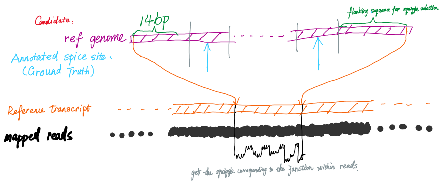
After obtaining the candidate splice site, candidate motifs are obtained from reference genome. To make sure we have enough number of bases included in the candidate motif, flanking sequences of size 20 are included in both sides of the candidate searching window. Candidate squiggles are then obtained from candidate motif using scrappie model (version1.4.0). The following figure shows an example of obtaining a candidate squiggle given one specific candidate splice site.



Step 3. Obtaining junction squiggle

In this version of NanoSplicer, junction squiggles are obtained using the information of reference transcript. The determination of junction within read is based on true splice site (Figure 1).
The start and end positon of a junction within read is first determine from reference genome. They are the boundary of candidate searching window in both sides of splice sites plus flanking sequence of size 14. The flanking size is 20 in step 2, which is slightly larger than the one used here. The purpose of doing this is to ensure the shape of the junction squiggle is captured inside the candidate squiggle. After the determination of the coordinate of the junction within reads, I used tombo (v1.5) to map the squiggles to basecalls and subset the squiggles corresponding to the junction with read.

Note: this version might be not be appropirate to compare with minimap2 result. Because NanoSplicetr actually used the information of the annotation and ref transcript but minimap2 doesn't have those information. But this version is still useful to show that **if we can find the squiggles that corresponding to the true junctions within reads, NanoSplicer can distinguish true splice site with other candidates which are close matches.**


Candidate:
14bp
flanking sequence for squiggle selection
ref genome
Annotated spice site:
(Ground Truth)
Reference transcript
mapped reads
get the squiggle corresponding to the junction within reads

Step 4: Dynamic time warping