Supplementary Material
NanoSplicer: Accurate identification of splice junctions using Oxford Nanopore sequencing

Yupei You [1], Michael B. Clark [2,*] and Heejung Shim [1,*]

[1]School of Mathematics and Statistics and Melbourne Integrative Genomics, The University of Melbourne, Parkville, Australia

[2]Centre for Stem Cell Systems, Department of Anatomy and Physiology, The University of Melbourne, Parkville, Australia.

1 Supplementary Methods

1.1 Junction within reads (JWRs)

Junction within reads (JWRs) are defined as the subsequences in mapped long reads which split and map to different exons, supporting potential splice junctions. Each JWR requires 1. the location of mapped splice junction (a pair of mapped 5' and 3' splice sites) and 2. subsequences from both exons. NanoSplicer takes as input mapped long reads in BAM/SAM format, from which the location of mapped splice junctions can be quickly extracted. Then, we identify subsequences from both exons as follows. To ensure a JWR contains information about all its candidate splice junctions, we choose the subsequences to be included in the JWR after obtaining all candidate junctions. Section 2.2 in the main text describes how NanoSplicer selects candidate splice junctions and supplementary section 1.2.1 provides a detailed procedure to identify nearby canonical candidate splice junctions (one option for candidate splice junctions).

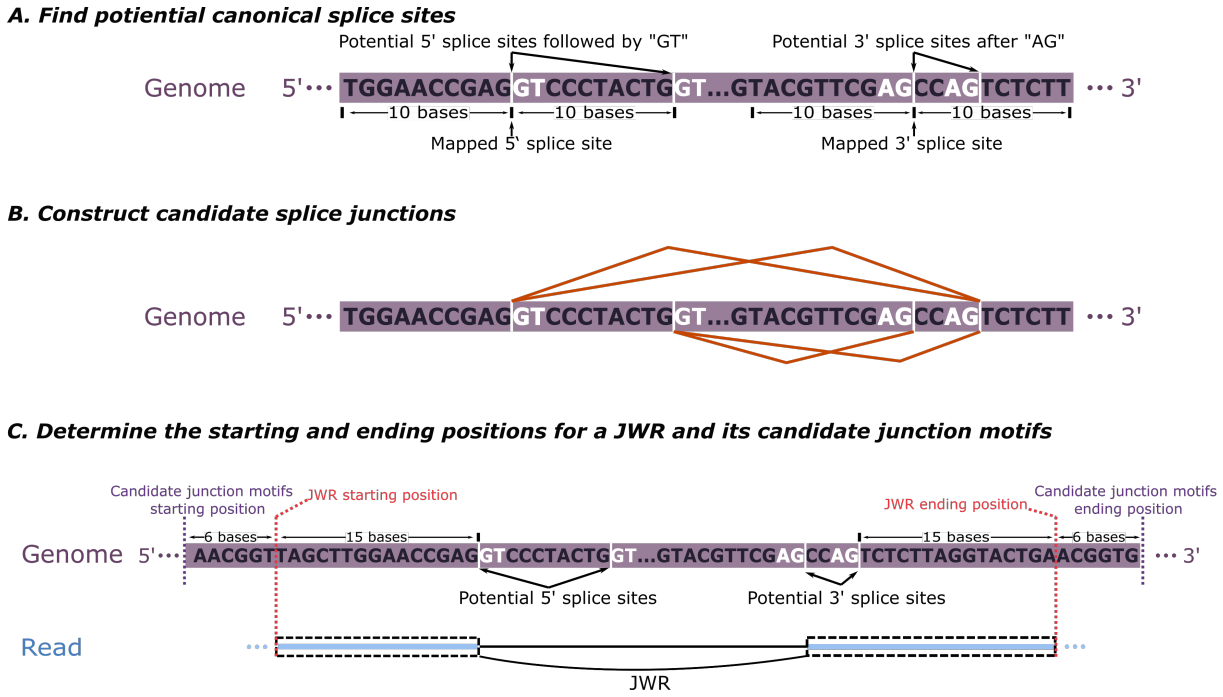

Fig. S1: **Candidate splice junctions, candidate junction motifs, and junction within reads (JWRs)** A: All potential 5' and 3' canonical splice sites within 10nt of the mapped splice sites. B: Construct canonical candidate splice junctions by considering all 5'-3' combinations of the splice sites identified in A. C: Panel shows the start and end positions of a JWR and its candidate junction motifs for a list of candidate splice junctions

Once we obtain all candidate splice junctions, we identify the 5' and 3' splice sites farthest away from each other among all 5' and 3' splice sites in the candidate junctions (e.g., leftmost 5' splice site and rightmost 3' splice site in supplementary Fig. S1C). Then, we use the 15th nucleotide preceding the 5' splice site and the 15th nucleotide after the 3' splice site as the first and last nucleotides of the subsequences for the JWR, respectively.

1.2 Obtaining candidate squiggles

Section 2.2 in the main text describes each step of obtaining candidate squiggles: constructing a list of candidate splice junctions, and then for each candidate, identifying a candidate junction motif and predicting its expected candidate squiggle. Here, we will provides details on the iden-

tification of nearby canonical splice junctions (one option for candidate splice junctions) (supplementary section 1.2.1), the selection of the start and end positions of each candidate junction motif (supplementary section 1.2.2), and the prediction of an expected candidate squiggle using the "expected current level model" in Tombo (supplementary section 1.2.3).

1.2.1 Canonical ("GT-AG") splice junctions as candidate splice junctions

For each JWR, to identify nearby canonical splice junctions (introns that start with "GT" and end with "AG"), we first search for all potential 5' splice sites followed by "GT" within 10nt of the mapped 5' splice site and all potential 3' splice sites after "AG" within 10 nt from the mapped 3' splice site (supplementary Fig. S1A). Then, we consider all possible combinations of these potential 5' and 3' splice sites, creating canonical splice junctions (supplementary Fig. S1B).

If the transcript strand information is recorded in the input BAM/SAM file (i.e. 'ts' tag is present in the BAM/SAM file), we search for canonical candidate splice junctions on the transcript strand of the reference genome. Otherwise, the "GT-AG" pattern from both strands will be included as candidates.

1.2.2 Start and end positions of candidate junction motifs

Each candidate junction motif for a JWR extends 5' and 3' from the candidate splice junction to a common location which is the start and end positions of the JWR (see supplementary Fig. S1C and supplementary section 1.1). This ensures each candidate has the same nucleotide sequence (and squiggle signal) at the beginning and end, ensuring differences between candidate squiggles are solely due to the various splice junctions utilised.

In practice, the start and end of the junction squiggle may not perfectly match that of JWR (or start and end positions of candidate junction motifs). Thus, we include 6 nucleotides at each side of the candidate junction motif as a buffer (see supplementary Fig. S1C), and allow the

junction squiggle to be aligned to a part of the candidate junction motifs (see supplementary section 1.4.2).

1.2.3 Prediction of an expected candidate squiggle from a candidate junction motif

Nanopore sequencing works by recording changes in electrical current when a DNA or RNA molecule traverses through a pore. During the transmission of the DNA/RNA molecule, multiple nucleotides occupy the pore simultaneously. Thus, the current level is influenced by a k-mer (consecutive nucleotides with length of k). We predict a candidate squiggle for each candidate junction motif using an "expected current level model" in Tombo v1.5. The expected current level model provides the mean and standard deviation of the current level for each 6-mer; see https://nanoporetech.github.io/tombo/model_training.html for the procedure in Tombo to compute these means and standard deviations.

For simplicity, in the main text, we use the third nucleotide in a 6-mer to represent the whole 6-mer by following Tombo's procedure to match nucleotides to 6-mers. For example, to predict an expected squiggle for a motif *GCAG*, we first include 2 and 3 nucleotides present before and after the motif, respectively, yielding an extended motif, such as, *AGGCAGCTG*. Then, we convert this extended motif into four 6-mers (*AGGCAG, GGCAGC, GCAGCT* and *CAGCTG*), resulting in four mean-standard deviations, each of which is the mean-standard deviation of current levels for each 6-mer.

Note that the means and standard deviations provided by the expected current level model in Tombo are computed using normalised squiggles; see https://nanoporetech.github.io/tombo/resquiggle.html for details on the normalisation procedure in Tombo. NanoSplicer obtains junction squiggles using the "resquiggle" tool in Tombo which performs the same normalisation procedure. This enables the junction squiggles to be comparable to candidate squiggles.

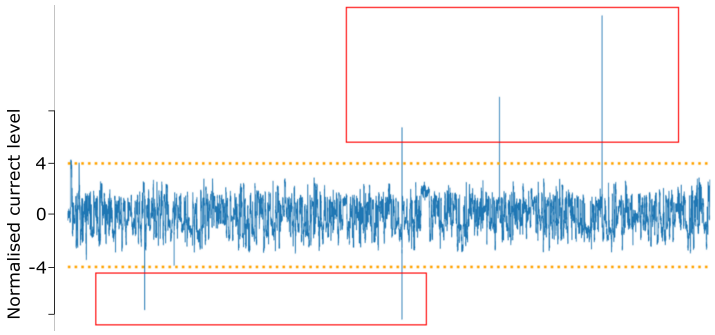

Fig. S2: **Visualisation of a squiggle corresponding to a long read from the synthetic data** The red boxes highlight transient current spikes within the squiggle. The orange lines indicate the threshold ($T = 4$) to remove spikes in our analysis of the paper. Note that the squiggle in this figure is from an entire read.

1.3 Junction squiggle preprocessing

We occasionally observe transient current spikes in junction squiggles which are a few current measurements far away from a typical range of squiggle current measurements (between -3 and 3 for squiggles normalised using Tombo; see https://nanoporetech.github.io/tombo/resquiggle.html for details on the normalisation procedure in Tombo). Supplementary Fig. S2 visualises a squiggle corresponding to a long read from the synthetic data (see section 3 in the main text or supplementary section 2.1 for details of the data), showing a typical range of normalised current measurements, as well as examples of the spikes. NanoSplicer removes those spikes from junction squiggles by eliminating current measurements whose absolute values are bigger than a user-specified threshold T. In the analysis of this paper, we used $T = 4$ to remove spikes from junction squiggles which are normalised by Tombo.

1.4 Dynamic time warping (DTW) in NanoSplicer

In NanoSplicer, we adapt DTW to align a junction squiggle to each of its candidate squiggles. DTW is an efficient algorithm for aligning two sequences which may vary in speed. Formal backgrounds on DTW can be found in Keogh and Ratanamahatana (2005). Here, supplementary

section 1.4.1 presents our implementation of DTW in NanoSplicer which places restrictions on standard DTW, and supplementary section 1.4.2 details these restrictions.

1.4.1 Align junction squiggle to candidate squiggles using DTW

NanoSplicer aligns a junction squiggle to each of its candidate squiggles using DTW. For each alignment, NanoSplicer treats the junction squiggle as observations from a model that has the means and standard deviations from each candidate squiggle as parameters. Then, we use a negative log-likelihood as a measure of distance in DTW. Let $\boldsymbol{x} = (x_1, \ldots, x_K)$ be the junction squiggle with length K, where x_k is the k-th current measurement. Let $\boldsymbol{c}^m = (\mu_i^m, \sigma_i^m)_{i=1}^{L_m}$ represent the m-th candidate squiggle, where μ_i^m and σ_i^m are the mean and standard deviation of the current level for the i-th k-mer in its junction motif with length L^m. Note that for simplicity, the main text describes "the mean and standard deviation for each nucleotide" instead of "for k-mer"; see supplementary section 1.2.3 for the definition of k-mer and the relationship between a nucleotide and a k-mer. Our implementation of DTW for the alignment is the same for all M candidate squiggles, so we will describe it for a single candidate squiggle and drop the superscript ^{m} for the rest of supplementary section 1.4.

Let $\boldsymbol{D} = [d_{i,k}]$ represent an $L \times K$ distance matrix, where $d_{i,k}$ is a distance between x_k and (μ_i, σ_i) defined by

$$d_{i,k} = -\log f(x_k|\mu_i, \sigma_i), \tag{1}$$

where f is the probability density function of the modified normal distributions (see supplementary section 1.5 for the definition). Let $\boldsymbol{a} = (a_1, \ldots, a_K)$ denote an alignment between $\boldsymbol{x}$ and $\boldsymbol{c}$, where a_k indicates the index of the mean-standard deviation in $\boldsymbol{c}$ where x_k is aligned. For example, $a_k = j$ indicates that x_k and (μ_j, σ_j) are aligned. Using DTW, NanoSplicer finds the alignment that minimises a distance between $\boldsymbol{x}$ and $\boldsymbol{c}$, defined by $\sum_{k=1}^{K} d_{a_k,k}$, among alignments satisfying the restrictions described in the following section 1.4.2.

1.4.2 DTW restrictions in NanoSplicer

Here, we detail the restrictions that are imposed on alignments when NanoSplicer searches for the alignment with minimum distance between $\boldsymbol{x}$ and $\boldsymbol{c}$. Supplementary Fig. S3 shows examples of a valid alignment (satisfying all restrictions) and an invalid alignment (breaching the restrictions).

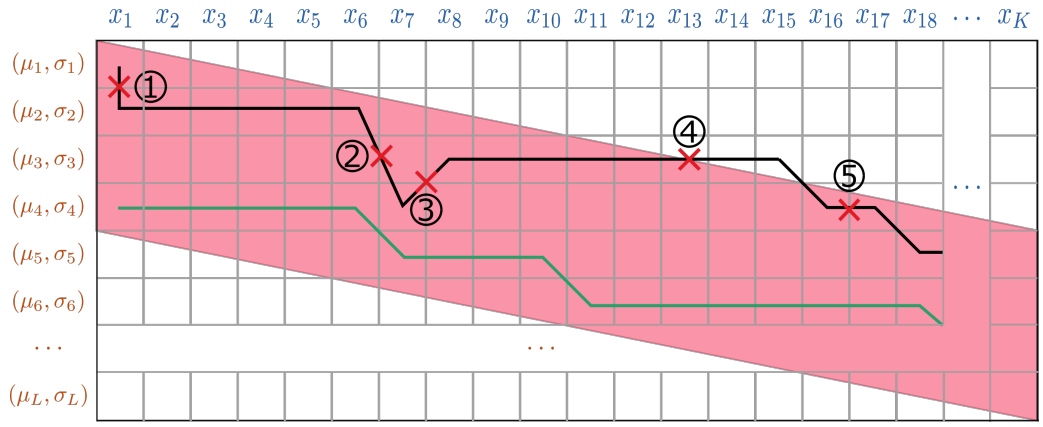

Fig. S3: **Examples of valid and invalid alignments in NanoSplicer** Figure shows an $L \times K$ matrix having junction squiggle $x_1, \ldots, x_K$ in its column and candidate squiggle $(\mu_1, \sigma_1), \ldots, (\mu_L, \sigma_L)$ in its row, respectively. An alignment is represented by a path on the matrix. Specifically, a path passing the grid at i-th row and j-th column indicates that (μ_i, σ_i) and x_j are aligned. The diagonal pink band is the area where valid paths are restricted to be inside. The green path shows a valid path in NanoSplicer DTW implementation. The black line represents an invalid path, and the red "X"s show where each of the restrictions described in supplementary section 1.4.2 is breached: 1) x_1 is aligned to multiple elements in the candidate squiggle; 2) The continuity is breached; 3) The monotonicity is breached; 4) A path is not allowed to go out of the pink band; 5) The minimum dwell time t ($t = 4$ in this example) is not satisfied as the number of current measurements aligned to (μ_4, σ_4) is less than t ($= 4$).

- **No warping in the junction squiggle:** We treat each current measurement of the junction squiggle as an observation from a model that has means and standard deviations of each candidate squiggle as parameters. Each single observation has only one mean-standard deviation in the model. Thus, each measurement in $\boldsymbol{x}$ is aligned to only one mean-standard deviation in the candidate squiggle.

- **Monotonicity and continuity in the candidate squiggle:** An alignment $\boldsymbol{a} = (a_1, \ldots, a_K)$ is valid only if $a_1, \ldots, a_K$ are monotonically and continuously ordered. That is, $a_k \leq a_{k+1}$ and $a_{k+1} - a_k \leq 1$, for all $k \in \{1, \ldots, K-1\}$.

- **Warping band:** An alignment $\boldsymbol{a} = (a_1, \ldots, a_K)$ can be represented by a path $(a_1, 1), \ldots, (a_K, K)$ in an $L \times K$ matrix, where L is the length of the candidate squiggle. Paths are constrained to be within a diagonal band with bandwidth $w \times L$, where $w \in (0, 1]$ is a user-specified value. See supplementary Fig. S3 for examples of paths and band. Specifically, (a_k, k) has to be within the band for all $k \in \{1, \ldots, K\}$. In our paper, we used $w = 0.4$. The motivations of this warping band restriction include:

  - The junction squiggle alignment is expected to cover most nucleotides in the candidate junction motif. Thus, we prevent current measurements in the junction squiggle from being aligned to only a small proportion of the candidate squiggle.

  - In practice, the start and end of the junction squiggle may not perfectly match that of the candidate squiggles. Thus, we include more nucleotides at each side of the candidate junction motif as a buffer (see supplementary section 1.2.2 for more details), and then allow the method to search for the best start and end positions of the junction squiggle alignment among that near the two ends of the candidate squiggle (i.e., path does not have to start at $(1, 1)$ or end at (L, K) on the matrix.).

  - It speeds up the DTW algorithm which searches for the best alignment (see Keogh and Ratanamahatana (2005) for more explanation).

- **Minimum dwell time:** Very short dwell times (i.e., the duration of a translocation event) may not reflect typical translocation events, potentially causing misleading results in NanoSplicer. Thus, NanoSplicer restricts the number of current measurements in $x_1, \ldots, x_K$

aligned to (μ_j, σ_j) to ones bigger than or equal to t for all $j \in 1, \ldots, L$. We used $t = 4$ in the analysis of the paper.

1.5 Modified normal distributions

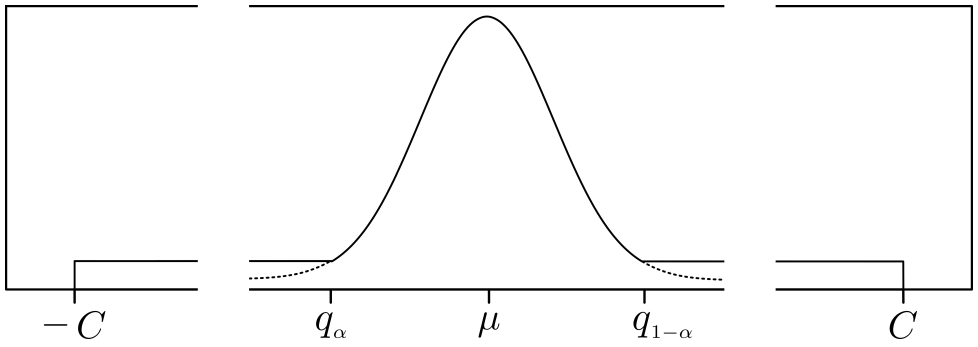

Fig. S4: **Modified normal distribution** The probability density function of the modified normal distribution has flat tails.

In our NanoSplicer model in section 2.4.2 in the main text, we used modified normal distributions which have flat tails, making our methods robust to measurements that match none of the M candidate squiggles. This section describes the modified normal distribution with μ, σ, C, and α as parameters. The probability density function of the modified normal distribution can be represented by:

$$f(y|\mu, \sigma, C, \alpha) \propto \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2}, & \text{if } y \text{ in } [q_\alpha, q_{1-\alpha}], \\ \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{q_\alpha-\mu}{\sigma}\right)^2}, & \text{if } y \text{ in [-C,}q_\alpha\text{) or (}q_{1-\alpha}\text{,C]}, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where q_α and $q_{1-\alpha}$ are the α and $1 - \alpha$ quantiles of the normal distribution with mean μ and standard deviation σ. Supplementary Fig. S4 shows the shape of this density function. In the analysis of this paper, we used $\alpha = 0.01$ (see supplementary section 2.9.2 for the performance assessment with different values of α) and $C = 4$ to ensure all observed current measurements and their summary values are between $-C$ and C.

1.6 Junction squiggle segmentation

Section 2.4.1 in the main text describes NanoSplicer procedure for junction squiggle segmentation, supplementary Fig. S5 provides a toy example, and supplementary Fig. S6A visualises the segments of a junction squiggle for 3 candidates. Here, we will describe our procedure to compute the candidate squiggles and alignments of the summary $\boldsymbol{y}$ (denoted by $\boldsymbol{c}_s^1, \ldots, \boldsymbol{c}_s^M$ and $\boldsymbol{A}_s$, respectively) from that of $\boldsymbol{x}$ (supplementary sections 1.6.1 and 1.6.2) and our practical implementation of the segmentation (supplementary section 1.6.3).

1.6.1 Candidate squiggles for the summary from junction squiggle segments

The standard deviations of the summary values are expected to be smaller than that of the original current measurements. Thus, we estimate standard deviations of the summary values from squiggle data, and use them to predict candidate squiggles, $\boldsymbol{c}_s^1, \ldots, \boldsymbol{c}_s^M$, for the summary which have been used as model parameters in the NanoSplicer model. A key idea to estimate standard deviations of the summary is to assume that summary values from the same squiggle (i.e., the same read) share the same standard deviation and to estimate the shared standard deviation by using multiple summary values from an entire squiggle. Specifically, for each (entire) squiggle, we already have its alignment to its basecalled read (from the "resquiggle" tool in Tombo during the step of obtaining junction squiggles; see section 2.1 in the main text). Let L denote the number of nucleotides in the basecalled read. We first partition current measurements in the (entire) squiggle into L sections by using their alignments to the read (i.e., consecutive measurements aligned to the same nucleotide belong to the same section). Then, we compute compute the summary $\tilde{\boldsymbol{y}} = (\tilde{y}_1, \ldots, \tilde{y}_L)$, where $\tilde{y}_i$ summarises information in the squiggle at its i-th section. The standard deviation of the summary values from this squiggle

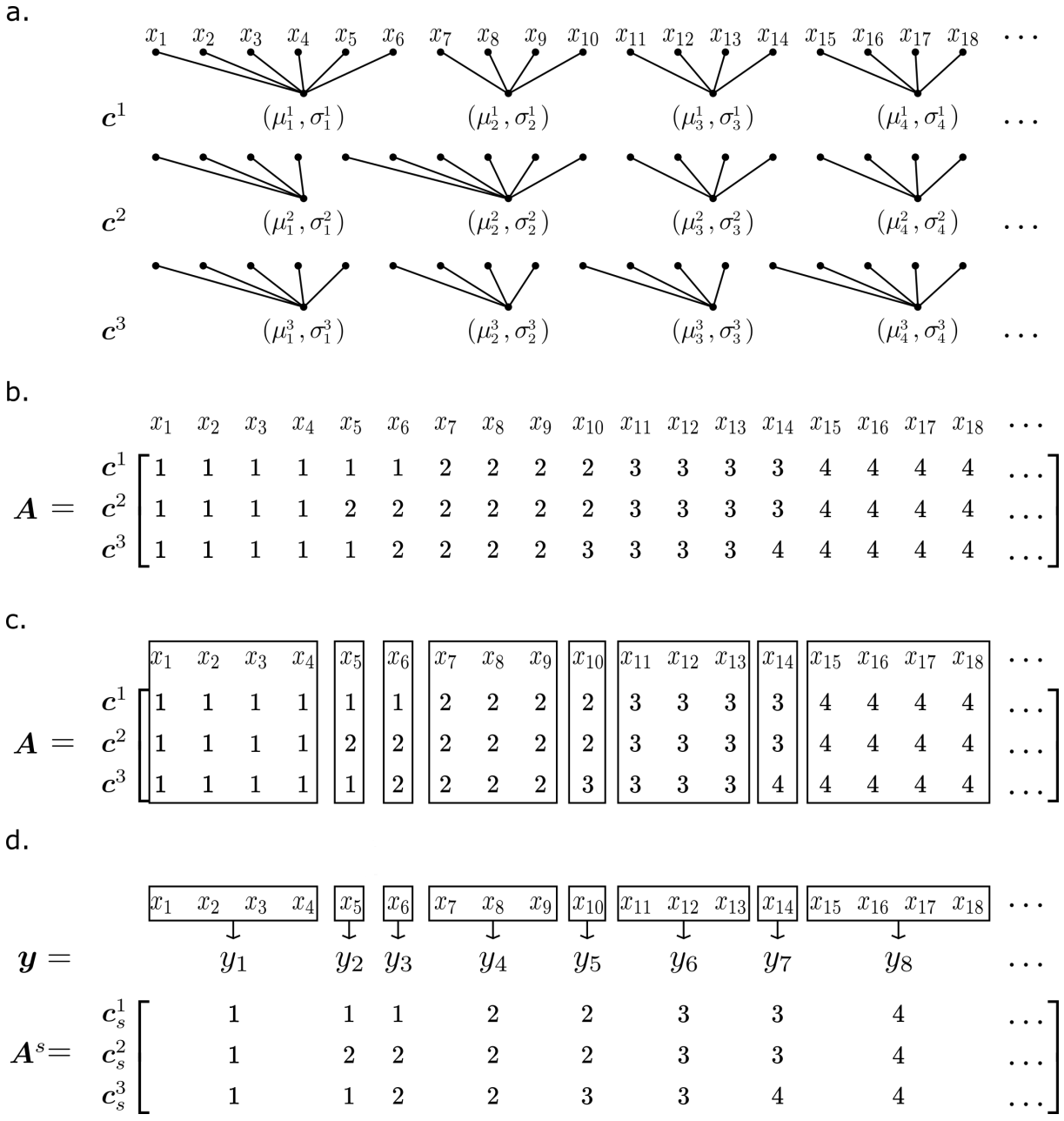

Fig. S5: **Toy example of alignments for junction squiggle, junction squiggle segmentation, and alignments for segmented junction squiggle** a: Example of alignments between a junction squiggle $\boldsymbol{x} = (x_1, x_2, \ldots)$ and 3 candidate squiggles $\boldsymbol{c}^1, \boldsymbol{c}^2, \boldsymbol{c}^3$. b: The alignments are represented by a matrix $\boldsymbol{A}$. c: Segments are defined by consecutive current measurements whose columns in $\boldsymbol{A}$ are the same. d: Current measurements in the junction squiggle are summarised into $\boldsymbol{y}$. The candidate squiggles of $\boldsymbol{y}$ are denoted by $\boldsymbol{c}_s^1, \boldsymbol{c}_s^2, \boldsymbol{c}_s^2$ and the matrix $\boldsymbol{A}^s$ represents the alignments between $\boldsymbol{y}$ and the candidate squiggles $\boldsymbol{c}_s^1, \boldsymbol{c}_s^2, \boldsymbol{c}_s^2$; see Supplementary sections 1.6.1 and 1.6.2 for details.

can be estimated by

$$\hat{\sigma}_s = \sqrt{\frac{\sum_{i=1}^{L}(\tilde{y}_i - \mu_i)^2}{L-1}}, \tag{3}$$

where μ_i is the mean of $\tilde{y}_i$ which can be obtained from the "expected current level model" in Tombo.

1.6.2 Alignments for the summary from junction squiggle segments

The M alignments between the junction squiggle $\boldsymbol{x} = (x_1, \ldots, x_K)$ and the M candidate squiggles $\boldsymbol{c}^1, \ldots, \boldsymbol{c}^M$ are represented by an $M \times K$ matrix $\boldsymbol{A} = [a_{mk}]$, where a_{mk} indicates the index of the mean-standard deviation in $\boldsymbol{c}^m$ where x_k is aligned; supplementary Fig. S5 provides a toy example. Removing redundant columns in $\boldsymbol{A}$ results in an $M \times N$ matrix $\boldsymbol{A}_s = [a^s_{mn}]$, where a^s_{mn} indicates the index of the mean-standard deviation in $\boldsymbol{c}^m_s$ where y_n (or n-th segment) is aligned; see supplementary Fig. S5 for an example.

1.6.3 Practical implementation of junction squiggle segementation

Segments with only a small number of measurements yield less stable summary values. However, NanoSplicer model does not incorporate the uncertainty of the summary, and so these summary values possibly cause less optimal performance of NanoSplicer. Thus, we filter out summary values that are computed by $\leq s$ current measurements - we used $s = 3$ in the analysis of the paper.

As the number of candidates increases, the number of segments will likely increase as well, as each candidate will contain unique mean-standard deviation elements that require further segmentation. This often leads to increasing numbers of segments with $\leq s$ measurements (see supplementary Fig. S6A). Thus, in practice, we also implement a modified procedure that uses a relaxed definition of segments to improve performance. From the M candidate squiggles $\boldsymbol{c}^1, \ldots, \boldsymbol{c}^M$, we first pick one as a baseline, denoted by $\boldsymbol{c}^b$; we chose one from the mapped

splice junction in the analysis of the paper. Then, we construct candidate-specific segments for each candidate $\boldsymbol{c}^m$ by applying the original segment definition (see section 2.4.1 in the main text) to only two candidates $\boldsymbol{c}^b$ and $\boldsymbol{c}^m$; see supplementary Fig. S6B for an example. For each candidate $\boldsymbol{c}^m$, we can compute the candidate-specific summary $\boldsymbol{y}^m$ using the candidate-specific segments. Then, we approximate the assignment probability in equation (3) of the main text using candidate-specific summaries, $\boldsymbol{y}^1, \ldots, \boldsymbol{y}^M$. Specifically, equation (3) in the main text can be written as:

$$\mathsf{P}(z = m|\boldsymbol{y}, \Theta) \propto \frac{\mathsf{P}(\boldsymbol{y}|z = m, \boldsymbol{c}_s^m, \boldsymbol{A}_s)}{\mathsf{P}(\boldsymbol{y}|z = b, \boldsymbol{c}_s^b, \boldsymbol{A}_s)} \mathsf{P}(z = m), \tag{4}$$

where $\Theta = (\boldsymbol{c}_s^1, \ldots, \boldsymbol{c}_s^M, \boldsymbol{A}_s)$. We then approximate the likelihood ratio $\frac{\mathsf{P}(\boldsymbol{y}|z=m,\boldsymbol{c}_s^m,\boldsymbol{A}_s)}{\mathsf{P}(\boldsymbol{y}|z=b,\boldsymbol{c}_s^b,\boldsymbol{A}_s)}$ using $\frac{\mathsf{P}(\boldsymbol{y}^m|z=m,\boldsymbol{c}_s^m,\boldsymbol{A}_{m,b}^s)}{\mathsf{P}(\boldsymbol{y}^m|z=b,\boldsymbol{c}_s^b,\boldsymbol{A}_{m,b}^s)}$. Here, $\boldsymbol{A}_{m,b}^s$ represents alignments of $\boldsymbol{y}^m$ to $\boldsymbol{c}_s^m$ and $\boldsymbol{c}_s^b$.

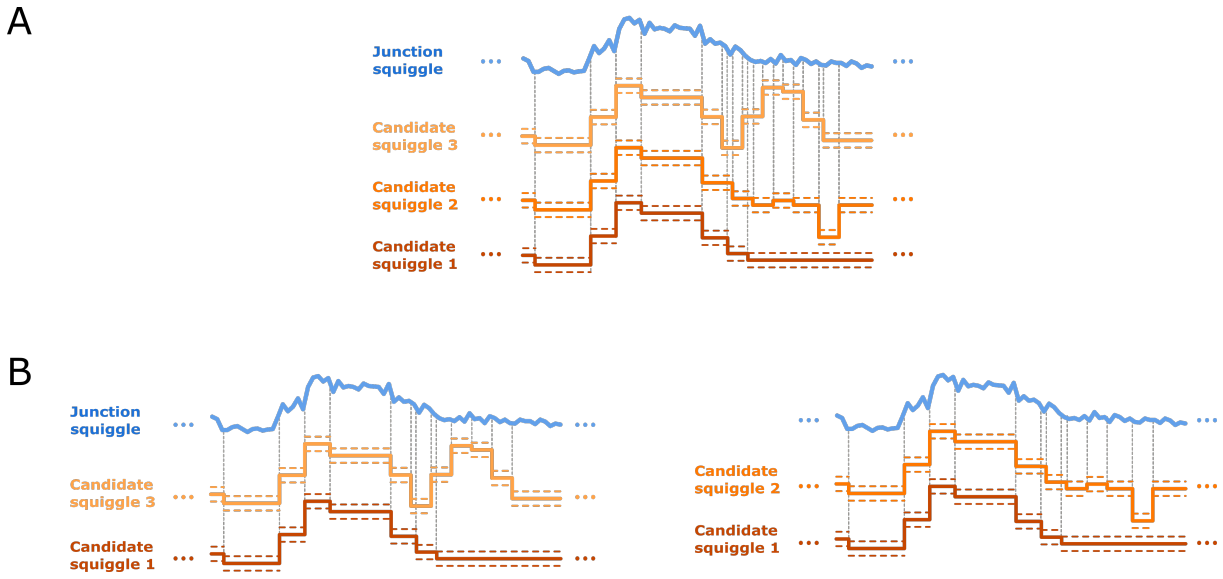

Fig. S6: **Junction squiggle segmentation** A: Example of alignments between a junction squiggle and 3 candidate squiggles. Each current measurement in the junction squiggle (blue) is vertically aligned with its corresponding mean-standard deviation in each candidate squiggle. The black dotted lines are the boundaries of the segments. B: Left and right panels show candidate-specific segments for candidates 3 and 2, respectively (candidate 1 used as a "base candidate").

1.7 Squiggle information quality

The NanoSplicer model assumes that the junction squiggle corresponds to the location of the JWR, however Tombo can potentially align the JWR to the incorrect squiggle location. Therefore, we add a step to filter out junction squiggles that do not have a high quality alignment to any candidate squiggle, suggesting they emanate from an off-target read subsequence. First we measure the alignment quality between a junction squiggle and each of its candidate squiggles using the average log likelihood over the nucleotides of the candidate squiggle. Then, we compute the maximum of these alignment qualities across M candidates, referred to as *squiggle information quality (SIQ)*. In practice, we restrict our splice junction identification to JWRs with SIQ bigger than a threshold to ensure their junction squiggles have high quality alignments at least one of their candidate squiggles. Here, we detail each step to computing SIQ.

Step 1: Measure the alignment quality between a junction squiggle and each of its candidate squiggles. Let $\boldsymbol{x}$ and $\boldsymbol{c}^m$ denote the junction squiggle and the m-th candidate squiggle with length L_m, respectively. We first partition current measurements in $\boldsymbol{x}$ into L_m sections using the alignment between $\boldsymbol{x}$ and $\boldsymbol{c}^m$ (i.e., consecutive measurements aligned to the same mean-standard deviation belong to the same section). Then, we compute the summary $\boldsymbol{y}^m = (y_1^m, \ldots, y_{L_m}^m)$, where y_n^m summarises information in $\boldsymbol{x}$ at its n-th section. Then, we measure the alignment quality between $\boldsymbol{x}$ and $\boldsymbol{c}^m$ using

$$S_m = \frac{\sum_{n=1}^{L_m} \log f(y_n^m | \mu_i^m, \sigma_i^m)}{L_m}, \tag{5}$$

where f is the probability density function of the modified normal distribution (see supplementary section 1.5), and μ_i^m, σ_i^m are the mean and standard deviation in the candidate squiggle $\boldsymbol{c}_s^m$ aligned to y_n^m (see supplementary section 1.6.1 for details on the candidate squiggle $\boldsymbol{c}_s^m$ of the summary values).

Step 2: Compute the maximum of the alignment qualities across M candidates. Step 1

provides $S_1, \ldots, S_M$ that measure the alignment quality between the junction squiggle and the M candidate squiggles. Then, we define squiggle information quality (SIQ) by

$$\text{SIQ} = \max\{S_1, \ldots, S_M\}, \tag{6}$$

and we restrict our splice junction identification to JWRs with SIQ bigger than a threshold to ensure their junction squiggles have high quality alignments at least one of their candidate squiggles.

1.8 Prior mixing proportion as a function of nucleotide composition near splice sites

It has been reported that among the canonical Eukaryotic splice junctions (i.e. "GT-AG" splice junctions), "GTC" or "GTT" is less frequent than "GTA" or "GTG" as intron patterns right after 5' splice sites; "AAG" or "GAG" is less frequent than "CAG" or "TAG" as the 3 nucleotides proceeding 3' splice sites (Irimia and Roy, 2008). For example, it's been reported that the relative frequency of "GTA" or "GTG" at the 5' splice site is larger than 94% in human, mouse, *S. cervisiae* and *S. pombe* (Ast, 2004). Also, our empirical analysis of RefSeq (O'Leary *et al.*, 2016) reference genome annotation (release GRCh38.p13) shows that the relative frequency of "GTA" or "GTG" at the 5' splice site and the relative frequency of "CAG" or "TAG" at the 3' splice site are 94.15% and 93.43%, respectively. Minimap2 (Li, 2018) provides an option to exploit these different frequencies of nucleotide composition near splice sites. NanoSplicer also enables users to exploit that information by modelling the prior mixing proportion as a function of nucleotide composition near splice sites.

In our biological RNA data analysis (see section 4 in the main text), we incorporated this information as follows. Let "GT^+" and "GT^-" denote the more frequent (i.e."GTA" or "GTG") and less frequent (i.e. "GTC" or "GTT") intron patterns at 5' splice sites, respectively; similarly, "AG^+" and "AG^-" denote the more and less frequent ones at 3' splice sites. Then, we model

the mixing proportion as

$$\mathsf{P}(Z=m) \propto \begin{cases} h & \text{if the } m\text{-th candidate junction is non-canonical,} \\ 1 & \text{if the } m\text{-th candidate junction has "GT}^{-}\text{-AG}^{-}\text{"}, \\ r & \text{if the } m\text{-th candidate junction has "GT}^{+}\text{-AG}^{-}\text{" or "GT}^{-}\text{-AG}^{+}\text{"}, \\ r^2 & \text{if the } m\text{-th candidate junction has "GT}^{+}\text{-AG}^{+}\text{"}. \end{cases} \tag{7}$$

We used $r = 9$ in our biological RNA data analysis, corresponding to a priori assumption that 1) GT^{+} at the 5' site and AG^{+} at the 3' site are nine times more likely to be splice sites than GT^{-} and AG^{-}, respectively; and 2) these propensities at the 5' and 3' splice sites are independent; see supplementary section 2.9.3 for NanoSplicer performance assessments for difference choices of r. We used $h = 1$ in the biological RNA data analysis, a priori assuming that non-canonical junctions have the same propensity to be a splice junction as canonical ones with "GT^{-}-AG^{-}".

The values for these hyperparameters (h and r) or the function for the prior mixing proportions can be modified for different datasets (e.g., datasets from different organisms) based on prior knowledge on the general propensity of a candidate to be a splice junction. Ultimately, methods that learn the prior mixing proportions from the data at hand may provide a better performance (left for future work).

2 Supplementary result

2.1 Data description

Synthetic RNA nanopore sequencing dataset: The synthetic sequins dataset was generated by Dong *et al.* (2021). The original dataset (GSE151984) contains duplicate libraries of each sequin mix (A and B), for a total of 4 cDNA libraries. The two sequin mixes contain the same transcripts but at differing concentrations. Libraries were constructed with the cDNA-PCR (SQK-PCS109) sequencing kit and PCR Barcoding (SQK-PBK004) kit using the standard protocol. The final pooled library was sequenced on a R9.4.1 MinION flow cell. In our analysis, we only used reads from one of the sequin mix A libraries. The *fast5* files containing the raw

nanopore squiggles are available from the European Nucleotide Archive (ENA) under accession ERS7273757.

Biological short-read sequencing dataset: We used the dataset generated by Holik *et al.* (2017). The original dataset contains 40 samples from two lung cancer cell lines (NCI-H1975 and HCC827). We utilised one of the samples (Accession: GSM1564311) from the NCI-H1975 cell line that was prepared using the Illumina TruSeq RNA v2 kit (Poly-A mRNA) followed by sequencing on an Illumina HiSeq 2500.

Biological RNA nanopore sequencing dataset: To assess the performance of NanoSplicer, we used a subset of a dataset generated by Matthew Ritchie's Lab at the Walter and Eliza Hall Institute of Medical Research, Australia. The original dataset contains 6 libraries from two lung cancer cell lines (NCI-H1975 and HCC827). Libraries were constructed with the cDNA-PCR (SQK-PCS109) sequencing and PCR Barcoding (SQK-PBK004) kits using the standard protocol. The final libraries were sequenced on a PromethION flow cell (FLO-PRO002). We utilised a portion of the reads mapping to chromosome 1 from one of the NCI-H1975 libraries. The squiggles (*fast5* files) of the reads included in our data analysis are available from ENA (Accession: ERS7273453).

2.2 Nanopore data : Basecalling of raw signals (squiggles)

We basecalled squiggles (*fast5* files) described in supplementary section 2.1 using Guppy 3.6.1. When running Guppy, we used configuration files *dna_r9.4.1_450bps_hac.cfg* for the synthetic RNA dataset and *dna_r9.4.1_450bps_hac_prom.cfg* for biological RNA dataset to get *fastq* files. Using Guppy, we filtered out low quality reads and demultiplex barcoded reads and trimmed adaptor sequences with the following command line parameters: *"--calib_detect --qscore_filtering --barcode_kits "SQK-PBK004" --trim_barcodes"*.

2.3 Nanopore data : Mapping of basecalled reads

2.3.1 Mapping nanopore reads to a reference genome (splice-aware)

NanoSplicer requires mapped nanopore reads as input. We used minimap2 to map nanopore reads to a reference genome (both synthetic and biological data).

Synthetic RNA nanopore reads: Using minimap2 (version 2.17-r943-dirty), we mapped the synthetic RNA reads to the sequin genome v2.4 with the following set of parameters: *"-ax splice -t 32 -k 14 -ub –eqx –secondary=no –sam-hit-only –splice-flank=no"*. We activated *"-eqx"* option in minimap2 to output the CIGAR string with mismatch information. By default, the CIGAR string in the output BAM file uses "M" to indicate a match or mismatch. With *"-eqx"*, the matches and mismatches are denoted as "=" and "X", respectively. We used the mismatch information when calculating the junction alignment qualitiy (see supplementary section 2.5). We deactivated the "splice flank" sequence preference option (*"–splice-flank=no"*) as this preference is not present in sequins; see supplementary section 1.8 for details on the intron sequence preference.

Biological RNA nanopore reads: Using minimap2 (verson 2.17-r943-dirty), we mapped the biological RNA reads to the human GRCh38 assembly (release GRCh38.p13). We ran minimap2 with the following set of parameters: *"-ax splice -t 32 -k 14 -ub –eqx –secondary=no –sam-hit-only"*. The setting is identical to what we used in the synthetic RNA data above, except for the removal of *"–splice-flank=no"*.

2.3.2 Mapping (synthetic) nanopore reads to the sequin isoforms

In our synthetic data analysis (section 3 in the main text), we defined a ground truth splice junction for each JWR by choosing one among the known 745 sequin splice junctions. To identify a 1-1 correspondence between each read and a sequin isoform, we mapped the reads to the sequin isoforms using minimap2. We ran minimap2 with command *"minimap2 -ax map-*

ont -t 32 –secondary=no –sam-hit-only", which is the setting for mapping nanopore reads to a reference without splicing. With the parameter "–secondary=no", the resulting BAM file did not contain any secondary alignments. Furthermore, to remove the potential ambiguity, we filtered out the reads with supplementary alignment(s) to a different isoform.

2.4 Short read data: pre-processing, mapping, and identifying splice junctions supported by the mapped short reads

In short-read sequencing data, base qualities near the ends of reads tend to be poor. All leading and trailing bases below a Phred quality score of 20 were removed using Trimmomatic v0.36 (Bolger *et al.*, 2014). We mapped the short reads to GRCh38 using STAR v2.7.3a (Dobin *et al.*, 2013) with the following parameters: "--outSAMtype BAM SortedByCoordinate --twopassMode Basic". To improve the mapping quality, we provided the RefSeq (O'Leary *et al.*, 2016) reference genome annotation (release GRCh38.p13) to guide the mapping. We used pysam v0.15.2 (https://github.com/pysam-developers/pysam) to find splice junctions supported by the mapped short reads, as well as the number of reads mapped to each of them.

2.5 Junction alignment quality (JAQ)

Basecalling errors can result in low quality alignments between the JWR and the reference genome. Therefore we hypothesized that the initial mapping would perform poorly for JWRs with high basecalling errors and that the advantage of NanoSplicer will be greatest for these. To test this, we adapted the definition of read accuracy in Rang *et al.* (2018) and defined a *junction alignment quality* (JAQ) by

$$\text{JAQ} := \frac{\sum(\text{matched bases in an alignment})}{\sum(\text{matched/mismatched/inserted/deleted bases in an alignment})}. \tag{8}$$

We obtained the number of matched, mismatched, inserted or deleted bases from the CIGAR string in the mapping result (SAM/BAM file). In the CIGAR string from splice-aware mappers,

the label for reference skipping (i.e. the mapped introns) is different from the deletions, and so the reference skipping is not counted in either numerator or denominator in equation 8. Only the mismatches, insertions or deletions near the mapped splice sites are counted. In the analysis of the paper, we used 25 bases (including mismatched, inserted and deleted bases) preceding the mapped 5' splice site and 25 bases after the mapped 3' splice site as bases in an alignment to compute JAQ.

2.6 Non-analysable JWRs

Fig. S7: **NanoSplicer fails to obtain candidate junction motifs when the mapped exon is too short.** NanoSplicer can not construct candidate junction motifs if the mapped exons do not overhang candidate junction motif start/end positions.

NanoSplicer failed to run on 0.44% of all JWRs in the synthetic dataset and 0.31% in the biological dataset due to the following two reasons. Note that these JWRs were omitted when we calculated the accuracy of the initial mapping in the main text.

1. 0.36% and 0.26% of the total JWRs in the synthetic and biological datasets, respectively: Mapped exons were too short, so NanoSplicer could not construct candidate junction motifs (supplementary Fig. S7). These JWRs are potentially related to spurious splice junction mapping. Only 31.6% and 56.3% of these JWRs (the synthetic and biological datasets, respectively) are initially mapped to true splice junctions.

2. 0.08% and 0.05% of the total JWRs in the synthetic and biological datasets, respectively: Tombo failed to align squiggles to reads, so NanoSplicer could not locate junction squig-

gles. These JWRs may have low squiggle quality. 86.0% and 92.5% of these JWRs (the synthetic and biological datasets, respectively) are initially mapped to true splice junctions. The accuracy for these JWRs are lower than that for JWRs for which we successfully ran NanoSplicer (93.4% and 94.5% for the synthetic and biological datasets, respectively).

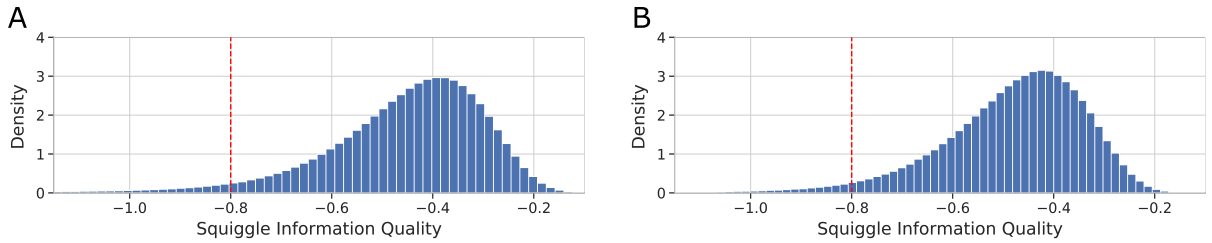

Fig. S8: **Empirical distributions of SIQ.** To choose a suitable threshold for SIQ, we first computed SIQ for all JWRs included in our analysis (3,477,172 and 2,073,181 JWRs for synthetic and biological data, respectively), and construct an empirical distribution of SIQ by pooling all SIQ values (A and B panels for synthetic and biological data, respectively). Then, assuming that most JWRs are well aligned to correct squiggle locations, we choose -0.8 as a SIQ threshold that identifies junction squiggles whose SIQ values much smaller than a majority of SIQs in the distribution (4.8% and 4.4% of all JWRs $<$ -0.8 for synthetic and biological data, respectively). In supplementary section S2.5.1, we assess NanoSplicer performance for different choices of a SIQ threshold and discuss how the choice of thresholds involves trade-offs between accuracy and the ability to identify splice junctions.

2.7 Contribution of SIQ, assignment probability and NanoSplicer correction to the accuracy improvement

In our analysis in section 3 [section 4] in the main text, NanoSplicer identified splice junctions for 3,058,814 JWRs [1,902,248 JWRs] whose SIQ $>$ -0.8 and strongest assignment probability $>$ 0.8; supplementary Fig. S8 shows empirical distributions of SIQ. So the accuracy of NanoSplicer is evaluated on this smaller set while we used all JWRs to compute the accuracy of initial mapping. Thus, multiple factors could contribute to the increased accuracy of NanoSplicer. These include the ability of SIQ and assignment probability to identify wrongly

mapped JWRs, as well as NanoSplicer correction of the initial mapping results. To investigate the contributions of these factors, we compute the accuracy of the initial mapping on two sets of JWRs: 3,309,817 JWRs [1,981,590 JWRs] after SIQ filtering (SIQ $>$ -0.8) and 3,058,814 JWRs [1,902,248 JWRs] after SIQ and strongest assignment probability filtering (SIQ $>$ -0.8 and strongest assignment probability $>$ 0.8). Supplementary Fig. S9 shows the accuracy for JWRs with different ranges of junction alignment quality. Supplementary Table S1 shows the number of JWRs in each junction alignment quality bin for all JWRs, JWRs after SIQ filtering, and JWRs after SIQ and strongest assignment probability filtering, including completely missed JWRs.

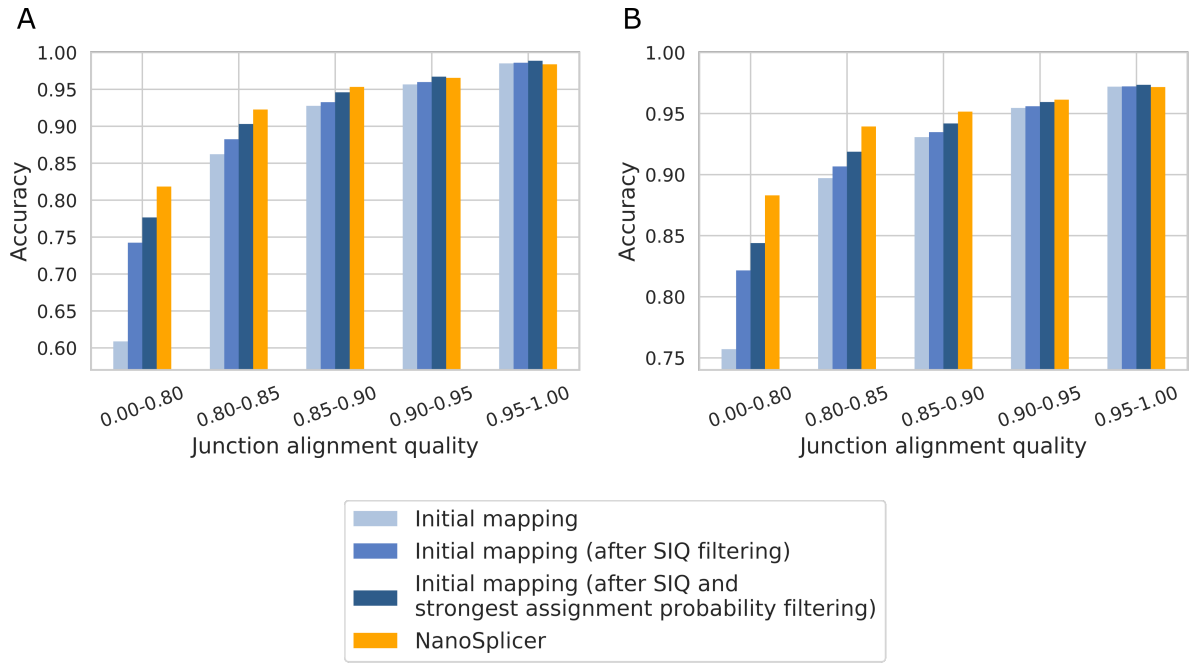

Fig. S9: **Contribution of SIQ, assignment probability and NanoSplicer correction to the accuracy improvement.** A: Accuracy of NanoSplicer and the initial mapping in the synthetic RNA dataset. B: Accuracy of NanoSplicer and the initial mapping in the biological RNA dataset.

	Junction alignment quality	All JWRs	JWRs with SIQ>-0.8	JWRs with SIQ>-0.8 & strongest assginment probability >0.8
Synthetic RNA dataset	0-0.8	249,137 (75,421)	173,931 (28,092)	146,834 (20,875)
	0.8-0.85	195,453 (12,250)	177,351 (7,611)	156,565 (6,182)
	0.85-0.9	598,889 (18,704)	570,466 (14,048)	515,537 (11,844)
	0.9-0.95	737,720 (13,460)	718,506 (11,194)	663,355 (9,656)
	0.95-1	1,695,973 (14,694)	1,669,563 (12,883)	1,576,523 (12,107)
Biological RNA dataset	0-0.8	106,590 (17,736)	80,316 (7,947)	72,979 (6,400)
	0.8-0.85	105,206 (5,233)	94,738 (3,912)	88,417 (3,328)
	0.85-0.9	355,046 (12,610)	335,690 (10,842)	317,948 (9,735)
	0.9-0.95	456,355 (13,129)	442,042 (12,253)	423,569 (11,416)
	0.95-1	1,049,984 (24,335)	1,028,804 (23,635)	999,335 (22,624)

Table S1: **Number of JWRs in each junction alignment quality (JAQ) bin.** The table presents the number of JWRs in each junction alignment quality (JAQ) bin for all JWRs, JWRs after SIQ filtering, and JWRs after SIQ and strongest assignment probability filtering. The numbers in the bracket represent the number of "completely missed JWRs"

2.8 JWRs filtered by SIQ and/or assignment probability are enriched in completely missed JWRs

Completely missed JWRs are JWRs whose mapped splice junctions are far away (> 10 nucleotides away in the analysis of the paper) from any true splice junctions, so their true splice junctions are not included as candidate junctions. Here, we assess how well SIQ and/or assignment probability in NanoSplicer recognise and filter out these JWRs. Supplementary Table S2 shows the proportions of JWRs and completely missed JWRs after SIQ and/or assignment probability filtering (these proportions have been computed from the numbers in supplementary Table S1). For each proportion, we performed the hypothesis testing for $H_0 : p_1 = p_2$ versus $H_1 : p_1 < p_2$, where p_1 is the proportion of completely missed JWRs after filtering and p_2 is the proportion of non-completely missed JWRs after filtering. For most cases, the SIQ and/or assignment probability filtered out more JWRs for completely missed JWRs than non-completely missed JWRs (p-value $< 10^{-15}$ for all cases except for the case with "*" in supplementary Table S2 which has p-value $= 0.013$), supporting that SIQ and/or assignment probability filtering help identify JWRs without true junctions as candidates.

	Junction alignment quality	JWRs with SIQ>-0.8 (out of all JWRs)	JWRs with strongest assignment probability >0.8 (out of all JWRs with SIQ>-0.8)	JWRs with SIQ>-0.8 & strongest assignment probability >0.8 (out of all JWRs)
Synthetic RNA dataset	0-0.8	69.81%(37.25%)	84.42%(74.31%)	58.94%(27.68%)
	0.8-0.85	90.74%(62.13%)	88.28%(81.22%)	80.10%(50.47%)
	0.85-0.9	95.25%(75.11%)	90.37%(84.31%)	86.08%(63.32%)
	0.9-0.95	97.40%(83.16%)	92.32%(86.26%)	89.92%(71.74%)
	0.95-1	98.44%(87.68%)	94.43%(93.98%)*	92.96%(82.39%)
Biological RNA dataset	0-0.8	75.35%(44.81%)	90.86%(80.53%)	68.47%(36.08%)
	0.8-0.85	90.05%(74.76%)	93.33%(85.07%)	84.04%(63.60%)
	0.85-0.9	94.55%(85.98%)	94.72%(89.79%)	89.55%(77.20%)
	0.9-0.95	96.86%(93.33%)	95.82%(93.17%)	92.81%(86.95%)
	0.95-1	97.98%(97.12%)	97.14%(95.72%)	95.18%(92.97%)

Table S2: **Proportions of JWRs and completely missed JWRs after SIQ and/or assignment probability filtering.** This table shows the proportions of JWRs after SIQ filtering (first column) and JWRs after SIQ and assignment probability filtering (among all JWRs or JWRs who pass SIQ filtering; the third and second columns). The numbers in brackets are those proportions for "completely missed JWRs".

2.9 Assessing NanoSplicer performance for different choices of SIQ and strongest assignment probability thresholds, quantiles in the modified normal distribution, and a hyperparameter in the mixing proportions

In our analysis in the paper, we chose -0.8 as a SIQ threshold, 0.8 as a strongest assignment probability threshold (sections 3 and 4 in the main text), $\alpha = 0.01$ for quantiles in the modified normal distributions (supplementary section 1.5), and $r = 9$ as a hyperparameter in the mixing proportions (supplementary section 1.8). Here we assess NanoSplicer performance for different choices of those values using randomly selected 100K JWRs from the synthetic data and biological data (biological data has been used for the hyperparameter in the mixing proportions).

2.9.1 Squiggle information quality (SIQ) and strongest assignment probability threshold

We used randomly selected 100K JWRs from the synthetic data to assess NanoSplicer performance for different choices of SIQ and strongest assignment probability thresholds. Supplementary Fig. S10 C & D show accuracy of NanoSplicer and the initial mapping for varying SIQ threshold values (-0.9, -0.8, -0.7, -0.6) at a single strongest assignment probability threshold (0.8) and for varying strongest assignment probability threshold values (0.7, 0.8, 0.9) at a single

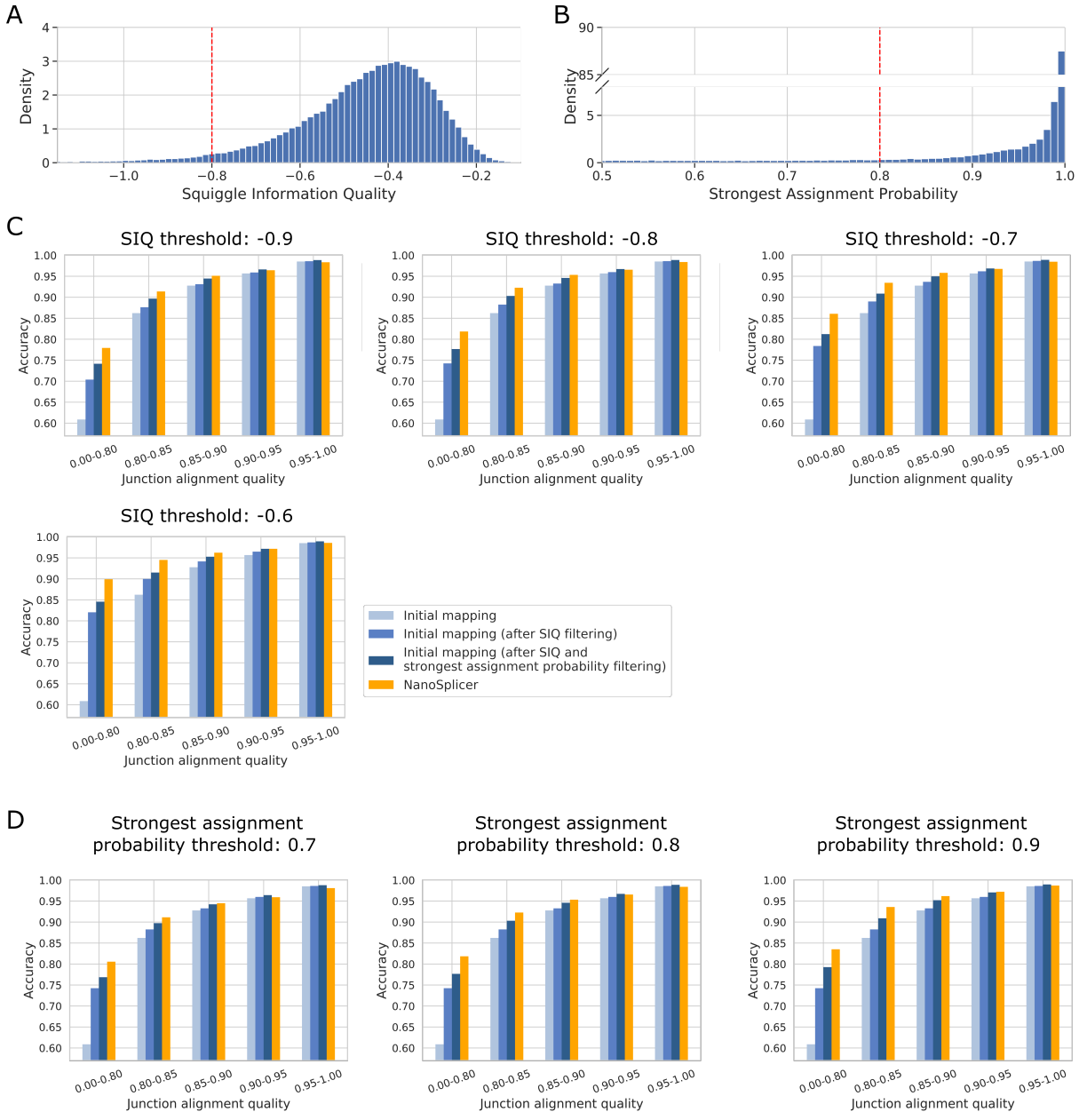

Fig. S10: **Assessment of NanoSplicer performance for different choices of SIQ and strongest assignment probability thresholds.** Distributions of SIQ values (A) and strongest assignment probabilities (B) from randomly selected 100K JWRs from the synthetic data. The red dashed lines indicate the threshold values used in our analysis of the main text. C: Accuracy of NanoSplicer and the initial mapping for varying SIQ threshold values (-0.9, -0.8, -0.7, -0.6) at a single strongest assignment probability threshold (0.8). D: Accuracy of NanoSplicer and the initial mapping for varying strongest assignment probability threshold values (0.7, 0.8, 0.9) at a single SIQ threshold (-0.8).

SIQ threshold (-0.8), respectively. The NanoSplicer performance is robust to difference choices of the thresholds. More stringent threshold values (larger threshold values) lead to higher accuracy, especially for JWRs with low junction alignment quality (JAQ). However, more stringent threshold values reduce the number of JWRs for which NanoSplicer is able to identify splice junctions (see Supplementary Fig. S10 A & B). For example, 92.9% of JWRs with JAQ $\leq$ 0.9 have SIQ $>$ -0.9, while only 64.3% of these JWRs have SIQ $>$ -0.6.

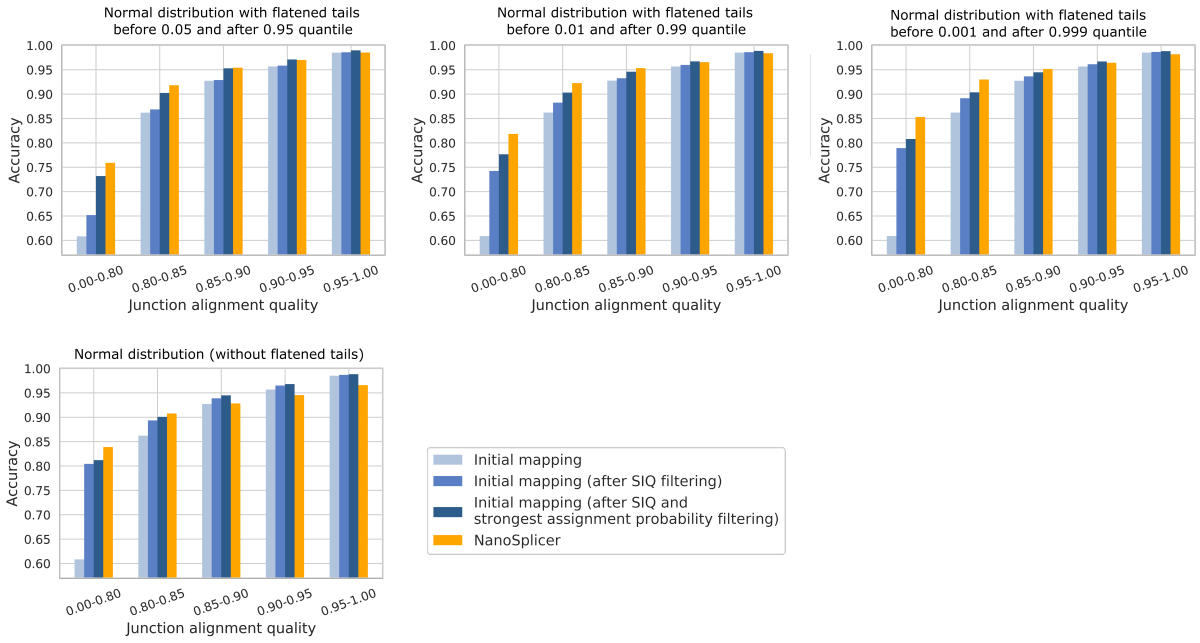

Fig. S11: **Assessment of NanoSplicer performance for different choices of α.** The α has been used to define quantiles in the modified normal distributions.

2.9.2 α for quantiles in the modified normal distributions

Our NanoSplicer model (section 2.4.2 in the main text) uses the modified normal distributions which have flat tails, making our methods robust to measurements that match none of the M candidate squiggles. Supplementary section 1.5 introduces details of these distributions which have α as a parameter to define q_α and $q_{1-\alpha}$ in the distributions. Here, we used the randomly selected 100K JWRs from the synthetic data to assess NanoSplicer performance for different

choices of α. Supplementary Fig. S11 show the accuracy of NanoSplicer and the initial mapping for varying α values (0.05, 0.01, 0.001, 0). Note that the modified normal distributions with $\alpha = 0$ are equivalent to normal distributions. The use of the modified normal distributions ($\alpha > 0$) helps improve NanoSplicer correction of the initial mapping results, particularly for JWRs with JAQ > 0.8. Among $\alpha > 0$, smaller α values lead to higher accuracy, particularly for JWRs with JAQ ≤ 0.85.

2.9.3 Hyperparameter r in the mixing proportions

In our biological RNA data analysis, we used $r = 9$ in the prior mixing proportions, corresponding to a priori assumption that 1) GT^+ at the 5' site and AG^+ at the 3' site are 9 (=r) times more likely to be splice sites than GT^- and AG^-, respectively; and 2) these propensities at the 5' and 3' splice sites are independent (see supplementary section 1.8 for details). Here, we assess NanoSplicer performance for different choices of r using randomly selected 100K JWRs from the biological data. Supplementary Fig. S12 shows accuracy of NanoSplicer and the initial mapping for varying r values between 1 to 96. Note that $r = 1$ corresponds to a priori assumption that nucleotide composition near splice sites is not related to the propensity of a candidate to be a splice junction. Modelling a prior mixing proportion as a function of nucleotide composition near splice sites ($r > 1$) helps improve NanoSplicer performance. NanoSplicer performance is robust to difference values of r - at all values of r between 1 to 96, NanoSplicer accuracy is higher than that of the initial mapping (94.6%).

2.10 More examples of splice junction correction with NanoSplicer

To demonstrate how squiggles can provide extra information to identify splice junctions, Supplementary Fig. S13 A1 & A2 and Fig. S13 B1 & B2 show examples of splice junction correction with NanoSplicer from the synthetic and biological datasets, respectively.

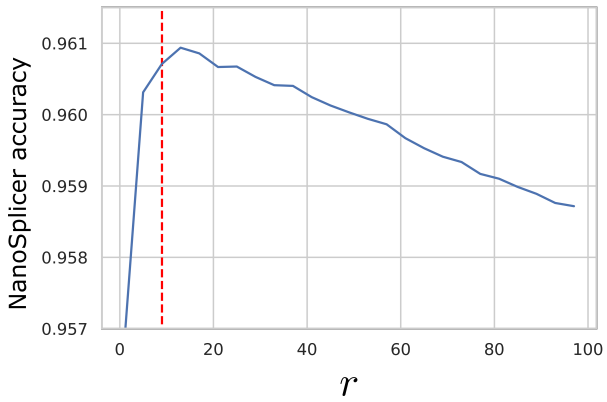

Fig. S12: **Assessment of NanoSplicer performance for different choices of a hyperparameter r in the mixing proportions.** The red dashed line indicates $r = 9$ which has been used in our analysis of the main text.

2.10.1 Additional example in synthetic RNA data

In the reference genome in Supplementary Fig. S13A1, there are two "GT" 5' splice motifs right next to each other, leading to two candidate splice junctions. Based on the ground truth of the synthetic RNA, two exonic bases "GT" preceding the 5' splice site were deleted from the nanopore read, resulting in the JWR initially mapped to the wrong 5' splice site with a 2-base shift. We compared the junction squiggle of this JWR to the candidate squiggles obtained from the true splice junction and the splice junction matching the initial mapping. Supplementary Fig. S13A2 shows the alignments between the junction squiggle and the two candidate squiggles respectively. The squiggle from the true candidate splice junction is visually a closer match to the junction squiggle. NanoSplicer quantified this squiggle similarity, giving an assignment probability of 0.99 to the true candidate for this JWR.

2.10.2 Additional example in biological RNA data

In the reference genome in Supplementary Fig. S13B1, there are two "AG" 3' splice motifs 4 bases apart at the 3' splice site, leading to two candidate splice junctions. One of the candidate splice junctions is supported by the short read data and is assumed to be the true one. However,

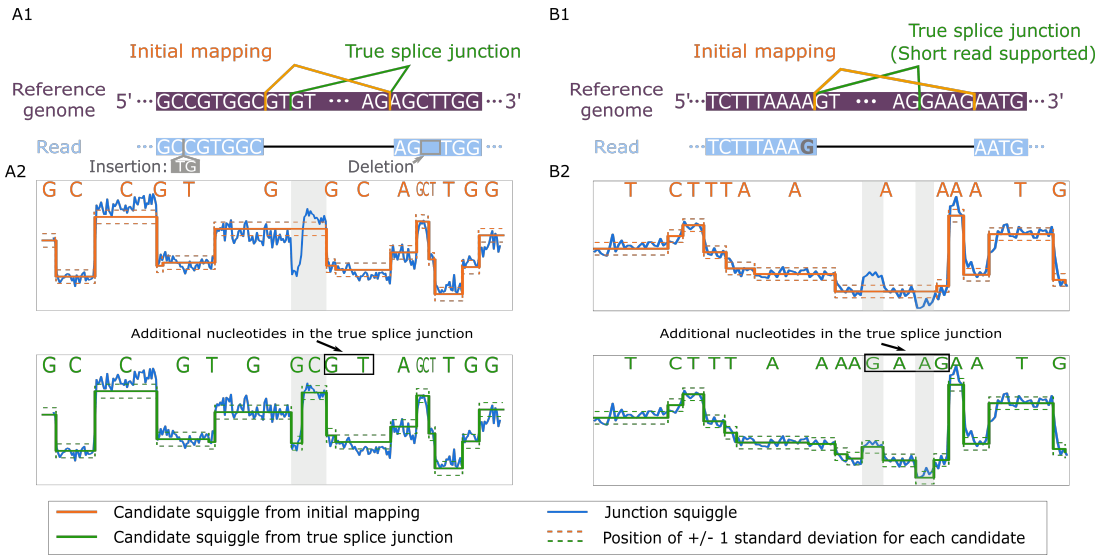

Fig. S13: **More examples of NanoSplicer correcting wrongly mapped JWRs** A1 & A2: Synthetic RNA data, B1 & B2: Biological data. A1 & B1: JWR mapping. Reference genome sequence shown in purple. Green line shows the location of the known ground truth splice junction. The mapped nanopore read (blue) shows the basecalled nucleotides of the JWR and how they were aligned to the reference genome. Mismatched nucleotide is highlighted by dark grey. Orange line shows the splice junction identified by the initial mapping of the JWR. Insertion - basecalled nucleotides in read that were not part of genome alignment. Deletion - nucleotides in the reference genome that are not part of the read. A2 & B2: Alignments between the junction squiggle for the JWR (blue) and the corresponding candidate squiggles from A1 & B1 (orange and green). Each junction squiggle current measurement is vertically aligned with its assigned mean-standard deviation in the candidate squiggle. The junction motifs for each candidate are shown at the top of each panel. Each nucleotide in the motifs is aligned with its corresponding squiggle position. Panels focus on the junction squiggle areas that distinguish between the candidates (grey background; the absolute difference in log likelihood between the two candidate models is bigger than 1.35; see Supplementary section 2.11 for details).

the nanopore read in supplementary Fig. S13B1 was mapped to the other candidate splice junction due to basecalling errors near the splice junction (the *"A"* and *"GAAG"* preceding and after the true splice junction were basecalled as only *"G"* in the nanopore read). We compared the junction squiggle of this JWR to the squiggles obtained from both candidate splice junctions. Supplementary Fig. S13B2 shows the alignments between the junction squiggle and the two candidate squiggles respectively. The shape of the squiggle for the true candidate is clearly a better match for the junction squiggle. NanoSplicer quantified this squiggle similarity, leading to an assignment probability of 0.999 to the true candidate for this JWR.

2.11 Identification of junction squiggle areas that distinguish between two candidates

In Fig. 3 in the main text and Fig. S13 for examples of NanoSplicer correcting wrongly mapped JWRs, we highlighted junction squiggle areas that distinguish between two candidates using grey background. We identified those areas as follows. Let y_n represent a summary value at the n-th segment of the junction squiggle. Suppose y_n is aligned to (μ^t, σ^t) and (μ^o, σ^o) in the candidate squiggles $\boldsymbol{c}_s^t$ and $\boldsymbol{c}_s^o$, respectively. Summary values from the same squiggle (i.e., the same read) share the same standard deviation, so $\sigma^t = \sigma^o = \sigma$. Assuming that $\boldsymbol{c}_s^t$ is the candidate squiggle identified by NanoSplicer, we denote μ^t by $\mu^o + \tilde{\mu}$. Then, $y_n \sim \mathcal{N}^*(\mu^o, \sigma)$ for the candidate squiggle $\boldsymbol{c}_s^o$ and $y_n \sim \mathcal{N}^*(\mu^o + \tilde{\mu}, \sigma)$ for the candidate squiggle $\boldsymbol{c}_s^t$, where $\mathcal{N}^*$ denotes the modified normal distributions. Then, the n-th segment is junction squiggle areas that distinguish between the two candidates if

$$-2\left[\log f(y_n|\mu^o, \sigma) - \log f(y_n|\mu^o + \tilde{\mu}, \sigma)\right] > \chi^2_{1,\alpha}, \tag{9}$$

where f is the probability density function of the modified normal distributions (see Supplementary section 1.5 for details) and $\chi^2_{1,\alpha}$ represents the $1 - \alpha$ quantile for the χ^2 distribution with 1 degree of freedom. We used $\alpha = 0.1$ in all figures of this paper.

2.12 Run time

The analysis pipeline includes following steps: 1. Identify the locations of JWRs from mapped reads; 2. Calculate junction alignment quality (JAQ) for each JWR; 3. Perform splice junction identification for each JWR. In steps 2 and 3, the entire analysis is parallelizable because the analysis for each JWR is independent. Multiprocessing is available in NanoSplicer (https://github.com/shimlab/NanoSplicer). Table S3 shows the run time for each step in our analysis of synthetic and biological datasets.

	Identify the location of JWRs	**Calculate junction alignment quality for all JWRs**	**Identify splice junctions for all JWRs**
Synthetic RNA dataset	∼2 mins	∼7.8 CPU hours (∼16 mins with 32 CPUs)	∼7151 CPU hours (∼124 hours with 72 CPUs)
Biological RNA dataset	∼1.5 mins	∼2.5 CPU hours (∼6 mins with 32 CPUs)	∼2007 CPU hours (∼35 hours with 72 CPUs)

Table S3: **Run time** The numbers in the table represent NanoSplicer run time for analysis of 3,492,373 and 2,216,054 JWRs from 1,528,017 and 758,330 mapped reads in the synthetic and biological RNA datasets, respectively. When multiprocessing option is available, the numbers in brackets represent NanoSplicer run time with the number of CPUs used in our analysis, and the numbers outside the brackets are the total CPUs hours.

2.13 Example of StringTie2 replacing rarer splice junctions with those from more highly expressed isoforms

Methods like StringTie2 (Kovaka *et al.*, 2019) use information from other reads (e.g., nearby splice junctions supported by high read counts) to guide splice junction correction. A limitation with this approach is that it can lead to the the replacement of rarer splice junctions with those from more highly expressed isoforms, causing less abundant junctions to go undetected. To present an example illustrating this limitation, we ran StringTie2 on the mapped long reads from the lung cancer cell line RNA nanopore sequencing dataset with default settings (using command *"stringtie -L"*); see supplementary section 2.1 for details of the dataset and supplementary sections 2.2 and 2.3 for the details of the basecalling and mapping.

Supplementary Fig. S16 A to C shows an example of StringTie2 replacing rarer splice junctions with those from more highly expressed isoforms. In this example, there are two different

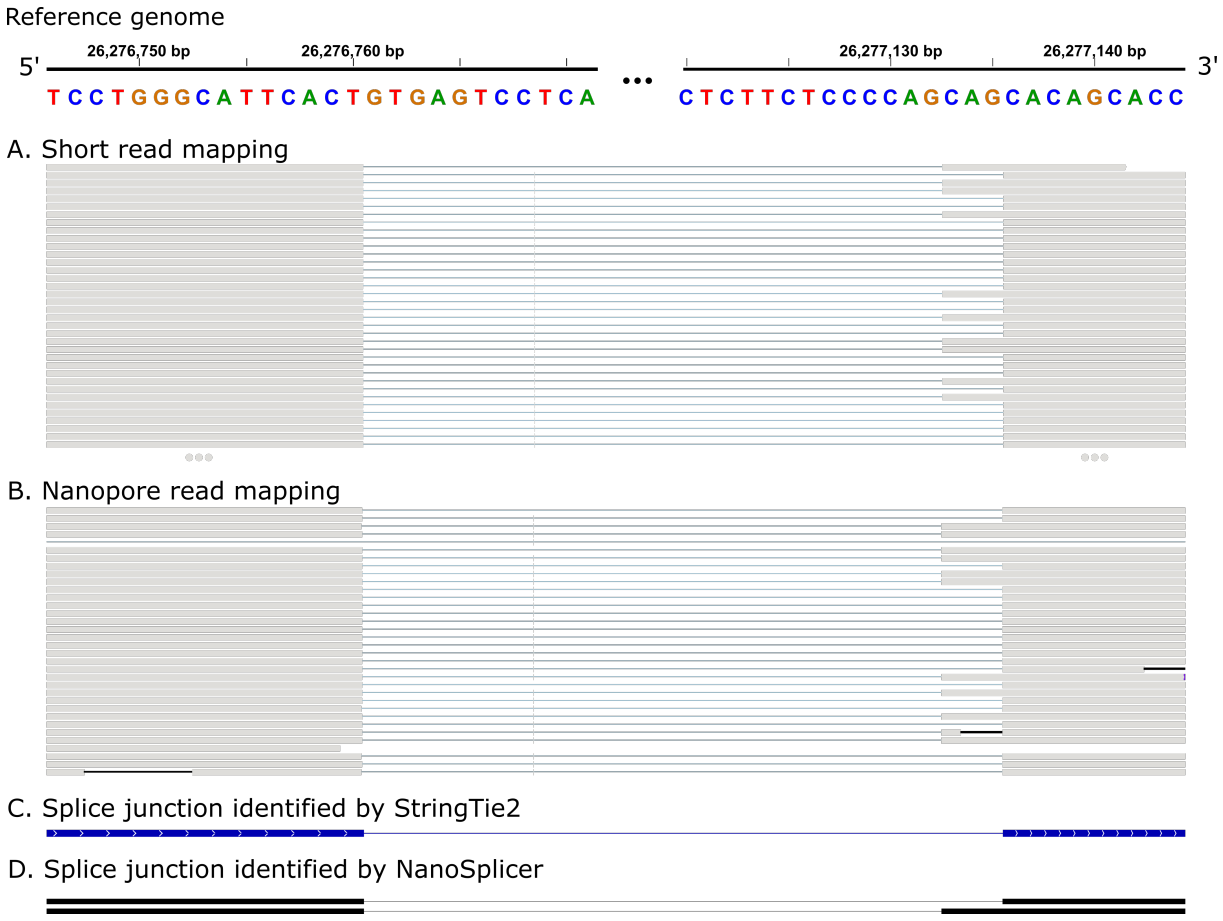

Fig. S14: **Example of StringTie2 replacing rarer splice junctions with those from more highly expressed isoforms.** The genome region in the figure is 26,276,746-26,277,144 bp on chromosome 1 in the human genome (GRCh38.p13). See supplementary section 2.1 for the detailed description of the sequencing data used in this figure. Note that this figure has been created by simplifying a screenshot of the Integrative Genomics Viewer (IGV) (Robinson *et al.*, 2011). A. Short read mapping result at the corresponding region. Note that only a proportion of the mapped short reads is displayed. B. Long read mapping result at the corresponding region. C. Splice junction for all isoforms assembled by StringTie2 using the long reads. D. Splice junctions identified by NanoSplicer using the long reads.

splice junctions supported by 20 and 63 short reads, respectively; see supplementary section 2.1 for the details of the short reads which are generated from the same lung cancer cell line. Short reads are expected to provide accurate information on the locations of splice junctions, supporting that these two splice junctions are real. The 3' splice site in the minor splice junction (one

supported by 20 short reads) is different in 3 nucleotides towards the upstream of the major one (supported by 63 short reads); see supplementary Fig. S16A. The initial mapping of long reads support both splice junctions (21 and 11 long reads mapped to the major and minor splice junctions, respectively). However, all isoforms assembled by StringTie2 using these mapped long reads support only the major splice junction, leading to the minor splice junction undetected (supplementary Fig. S16C).

Supplementary Fig. S16D shows splice junctions identified by NanoSplicer using the same dataset (see main text section 4 for the details). NanoSplicer identified both splice junctions (the major and minor splice junctions are identified at 19 and 7 JWRs, respectively; NanoSplicer did not provide identification results for 6 JWRs due to their poor squiggle information quality; see main text section 4 for the details). This supports that NanoSplicer has a good potential for identifying rare splice junctions as its performance is not affected by other reads. Note that NanoSplicer identifies splice junctions only for JWRs whose squiggles are informative (26 JWRs out of 32 in this example), limiting the potential usage of its output in splice junction "quantification" (see main text section 5 for more discussion). Thus, in the paper, we restricted our analysis to JWRs that have at most one nearby ground truth (see main text sections 3 and 4) as this analysis does not require splice junction quantifications, and we could not perform a quantification comparison between NanoSplicer and StringTie2.

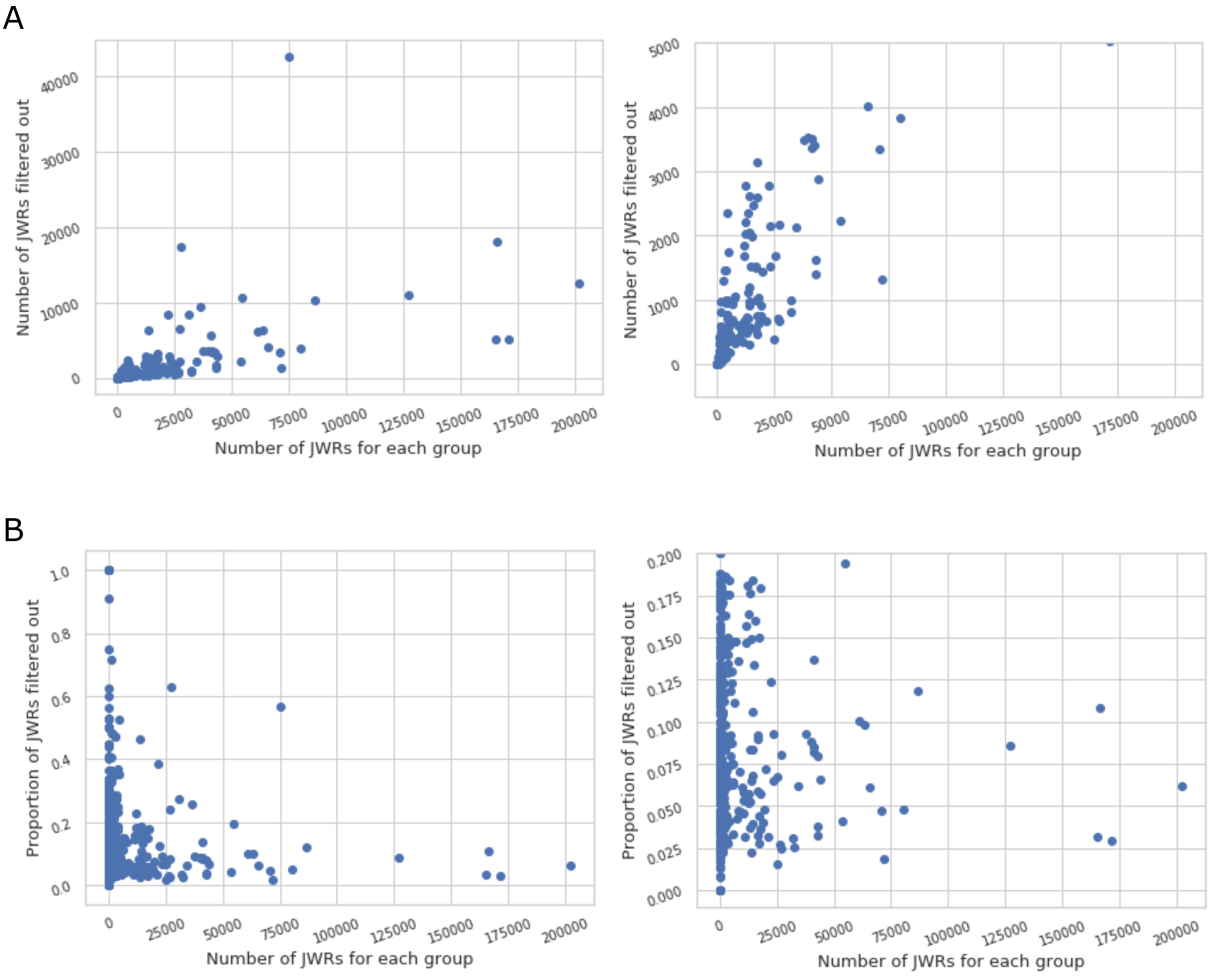

Fig. S15: **JWRs filtered out by SIQ and strongest assignment probability are not a random subset of all JWRs.** In the synthetic data analysis (section 3 in the main text), we identified 650 unique sequins splice junctions as ground truths for 3,342,643 JWRs (there are 3,477,172 in total but we identified ground truth for 3,342,643 JWRs; the remaining JRWs are completely missed JWRs). We grouped the JWRs based on their ground truths, leading to 650 groups of JWRs. JWRs in each group share the same ground truth. For each group, we computed the total number of JWRs, the number of JWRs filtered out (SIQ $\leq$ -0.8 or strongest assignment probability $\leq$ 0.8), the proportion of the filtered out JWRs. The top two panels (A) show a scatter plot of the number of JWRs filtered out vs the total number of JWRs (with different limits of the y-axis). The bottom two panels (B) show a scatter plot of the proportion of the filtered out JWRs vs the total number of JWRs (with different limits of the y-axis). If JWRs filtered out by SIQ and strongest assignment probability are a random subset of all JWRs, we would expect the proportion of the filtered out JWRs to be very similar across the groups of JWRs. However, the plots in B do not show this pattern.

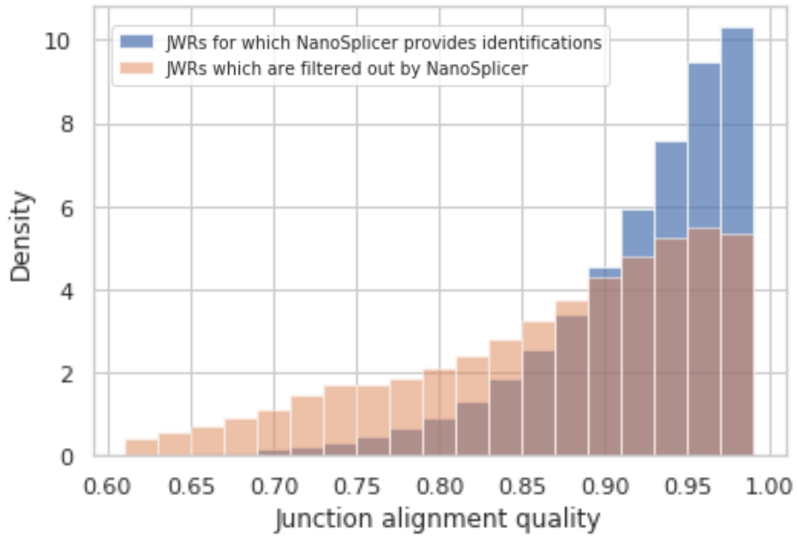

Fig. S16: **Distributions of JAQ for JWRs filtered out / remaining.** In our synthetic data analysis (section 3 in the main text), 3, 418,358 JWRs are filtered out by NanoSplicer. Figure shows two distributions of junction alignment quality (JAQ) for the 418,358 JWRs filtered out and the remaining 3,058,814 JWRs.

References

Ast, G. (2004). How did alternative splicing evolve? *Nature Reviews Genetics*, **5**(10), 773–782.

Bolger, A. M., Lohse, M., and Usadel, B. (2014). Trimmomatic: a flexible trimmer for illumina sequence data. *Bioinformatics*, **30**(15), 2114–2120.

Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., and Gingeras, T. R. (2013). Star: ultrafast universal rna-seq aligner. *Bioinformatics*, **29**(1), 15–21.

Dong, X., Tian, L., Gouil, Q., Kariyawasam, H., Su, S., De Paoli-Iseppi, R., Prawer, Y. D. J., Clark, M. B., Breslin, K., Iminitoff, M., *et al.* (2021). The long and the short of it: unlocking nanopore long-read rna sequencing data with short-read differential expression analysis tools. *NAR genomics and bioinformatics*, **3**(2), lqab028.

Holik, A. Z., Law, C. W., Liu, R., Wang, Z., Wang, W., Ahn, J., Asselin-Labat, M.-L., Smyth, G. K., and Ritchie, M. E. (2017). Rna-seq mixology: designing realistic control experiments to compare protocols and analysis methods. *Nucleic acids research*, **45**(5), e30–e30.

Irimia, M. and Roy, S. W. (2008). Evolutionary convergence on highly-conserved 3 intron structures in intron-poor eukaryotes and insights into the ancestral eukaryotic genome. *PLoS Genet*, **4**(8), e1000148.

Keogh, E. and Ratanamahatana, C. A. (2005). Exact indexing of dynamic time warping. *Knowledge and information systems*, **7**(3), 358–386.

Kovaka, S., Zimin, A. V., Pertea, G. M., Razaghi, R., Salzberg, S. L., and Pertea, M. (2019). Transcriptome assembly from long-read rna-seq alignments with stringtie2. *Genome biology*, **20**(1), 1–13.

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

O'Leary, N. A., Wright, M. W., Brister, J. R., Ciufo, S., Haddad, D., McVeigh, R., Rajput, B., Robbertse, B., Smith-White, B., Ako-Adjei, D., *et al.* (2016). Reference sequence (refseq) database at ncbi: current status, taxonomic expansion, and functional annotation. *Nucleic acids research*, **44**(D1), D733–D745.

Rang, F. J., Kloosterman, W. P., and de Ridder, J. (2018). From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy. *Genome biology*, **19**(1), 1–11.

Robinson, J. T., Thorvaldsdóttir, H., Winckler, W., Guttman, M., Lander, E. S., Getz, G., and Mesirov, J. P. (2011). Integrative genomics viewer. *Nature biotechnology*, **29**(1), 24–26.