RESEARCH SUMMARY

Splam: a splice-junction recognizer for cleaner RNA-seq alignments

By Kuan-Hao Chao

Aug 15, 2024 · 9 min read · Updated Jul 1, 2026

Abstract

Splam was motivated by a gap between splice-site prediction and transcriptome assembly. SpliceAI showed that deep learning could recognize splice signals, but its canonical-transcript framing left alternative isoform junctions underrepresented for the RNA-seq cleanup problem we faced in CHESS-scale transcript assembly. Splam treats each candidate intron as a paired donor-acceptor recognition problem, scores local junction sequence, and uses low-scoring junctions to remove weak spliced alignments before transcript assembly.

A common RNA-seq workflow aligns millions of short reads to a reference genome. Wherever a read spans an intron (a stretch a cell splices out of the transcript, joining the two flanking exons), the aligner records a splice junction — evidence for the boundary where pre-mRNA was spliced. Alignment-derived junctions are not all equally trustworthy: alignment errors, sequencing artifacts, and noisy transcription or splicing can produce low-confidence junctions that inflate false introns and distort transcript assembly. This was the practical setting behind Splam (Chao et al., 2024).

The CHESS 3 project (Varabyou et al., 2023) made that problem concrete. CHESS builds human gene catalogs from large collections of RNA-seq data. At that scale, spurious spliced alignments — reads mapped across a splice junction that may not be real — create false-positive junction signals, make the input to transcriptome assembly (reconstructing each gene's transcripts from the aligned reads) noisy, and can push assemblers toward transcript models built on weak splice evidence.

At the same time, SpliceAI (Jaganathan et al., 2019) had already shown something powerful: a deep neural network could recognize splice signals directly from sequence. It inspired the way I thought about the problem. But our use case was different from SpliceAI's original one. SpliceAI was trained and evaluated around one canonical protein-coding transcript per gene, while RNA-seq alignments contain junctions from many isoforms (the alternative transcripts one gene produces by combining its exons in different ways). For transcriptome cleanup, alternative junctions are not an edge case; they are part of the signal we want to preserve.

That led to a complementary problem, not an easier one. Detecting splice junctions is itself hard. Splam asks a downstream question: once an annotation or a spliced aligner proposes a donor-acceptor pair (the two ends of an intron — the donor at its 5′ start, the acceptor at its 3′ end), can a deep-learning recognizer use the local sequence around that candidate to decide whether the junction has credible splice signals, and can that score help remove spurious junctions before transcript assembly?

Splam is the algorithm I built to answer that question.

Why build a junction recognizer?

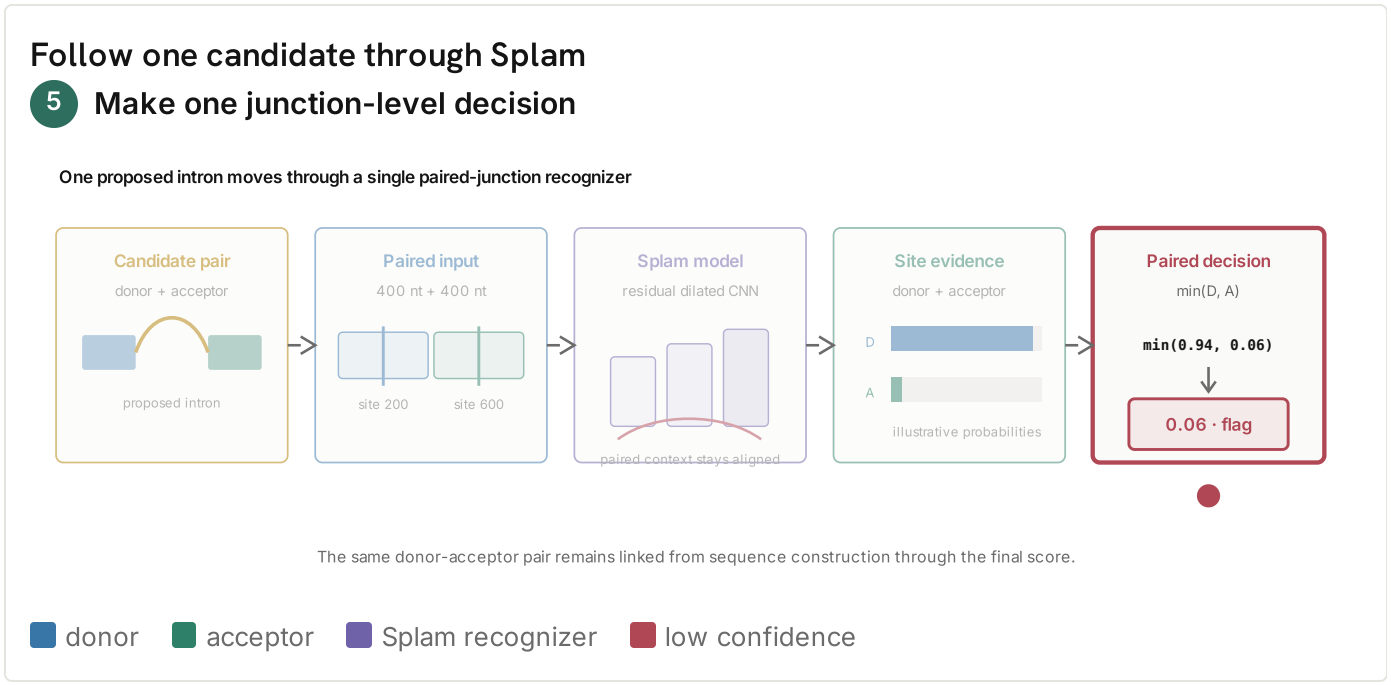
Splam was designed for candidate-junction recognition. In this pipeline, the input is a proposed intron from an annotation or an RNA-seq alignment, and the goal is to score whether that donor-acceptor pair looks like a plausible splice junction before the read is handed to a transcriptome assembler.

Three requirements followed from that. First, the algorithm had to respect isoforms. The Splam paper tested junctions from MANE (Morales et al., 2022) and alternative RefSeq annotations (O'Leary et al., 2016), and the alternative-junction benchmark is where the canonical-transcript framing mattered most. Second, the algorithm had to be useful after alignment, where the input is already a proposed donor-acceptor pair and the practical goal is to remove weak spliced alignments without throwing away real junctions. Third, the algorithm had to model the biological unit. SpliceAI predicts donor and acceptor sites as positions in sequence, whereas a splice junction is defined by a pair: a donor and an acceptor that belong to the same intron.

That is why Splam is not an exhaustive chromosome-wide splice-site scanner. It is a compact splice-junction recognition algorithm, focused on the candidate junction as the unit, designed to drop into an RNA-seq pipeline between spliced alignment and transcript assembly. It was my first splice-site deep-learning project — its ideas later grew into OpenSpliceAI.

The algorithm

Splam starts with a candidate intron: a proposed donor site and a proposed acceptor site, usually coming from an annotation or an RNA-seq alignment. It builds a fixed-length representation using exactly **800 nucleotides**: a 400 nt window centered on the donor and a 400 nt window centered on the acceptor. For short introns, those windows may overlap or use `N` padding; negative-strand candidates are reverse-complemented before the donor and acceptor contexts are concatenated. The full candidate-to-score path is one connected workflow.



The resulting one-hot input has shape **4 × 800**. Splam projects it to 64 feature channels, then processes it with a deep residual convolutional network containing **20 residual units** in five groups of four. Every unit contains two grouped, dilated convolutions with batch normalization and Leaky ReLU, plus a shortcut connection. Across the five groups, kernel widths are 11, 11, 11, 21, and 21, while dilation rates increase through 1, 5, 10, 15, and 20. This widens the sequence context without pooling away nucleotide resolution. Group outputs are projected and added before a final three-channel softmax. The complete model has **651,715 parameters**.

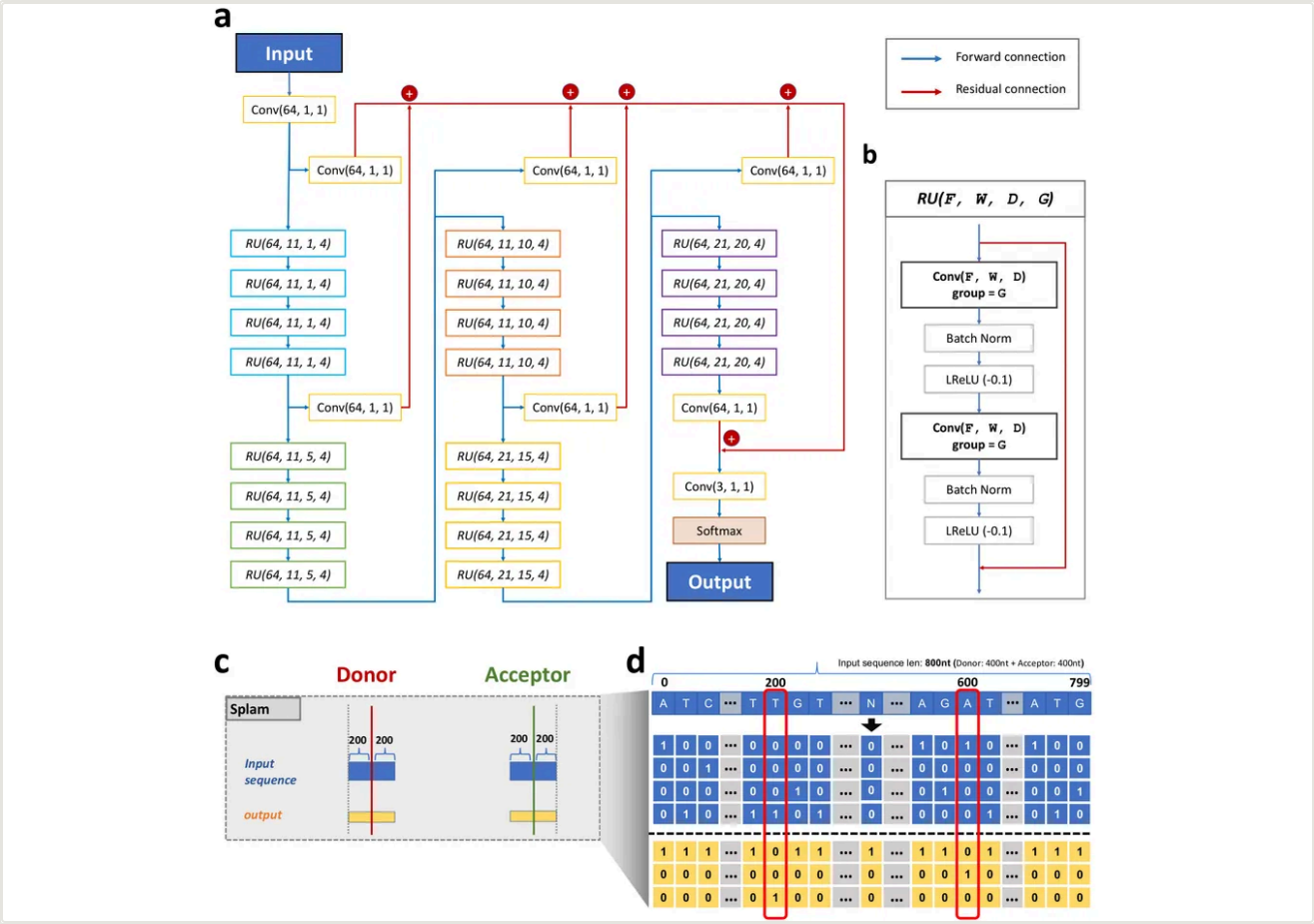


Figure 1. The Splam algorithm. **(a)** A deep stack of dilated residual units whose growing dilation widens the receptive field, ending in a per-base softmax over donor / acceptor / neither. **(b)** The residual unit — grouped dilated convolutions with batch norm and leaky ReLU. **(c, d)** Splam reads a donor and its acceptor together as one 800 nt input (200 bp flanking each site), scoring the junction as a pair.

The network returns a **3 × 800** array: donor, acceptor, and neither probabilities at every input position. Splam reads the donor probability at position 200 and the acceptor probability at position 600. The junction score is the smaller of those two values. That minimum is important: a confident donor paired with a weak acceptor remains a weak junction rather than being rescued by the stronger end.

Two choices set the algorithm apart from SpliceAI. It uses a short, biologically motivated window because the design tests whether enough splice-recognition information is concentrated near the donor, acceptor, branch point (a conserved intron site just upstream of the acceptor), and nearby regulatory motifs. And it scores the donor and acceptor together, mirroring the paired nature of an intron. This design is meant for evaluating putative splice junctions from annotation or RNA-seq alignments; it is not an exhaustive chromosome-wide search over every possible donor-acceptor pair. We also trained the neural predictor on splice junctions drawn from multiple isoforms in annotated protein-coding genes, so alternative junctions were present during training rather than treated as mistakes.

What it showed

The advantage shows up most clearly on alternative junctions. On held-out human junctions, Splam reached 96.1% junction-level accuracy at a score threshold of 0.8 on the Test40K benchmark. On MANE splice sites, Splam and SpliceAI were nearly identical. The separation appeared on alternative splice junctions from RefSeq but not MANE, the exact setting that motivated training on multiple isoforms: Splam produced fewer false negatives despite using much less sequence. In the paper's threshold-0.1 disagreement analysis, Splam correctly recovered **2,609** true splice junctions that SpliceAI labeled negative, while SpliceAI recovered **202** that Splam labeled negative.

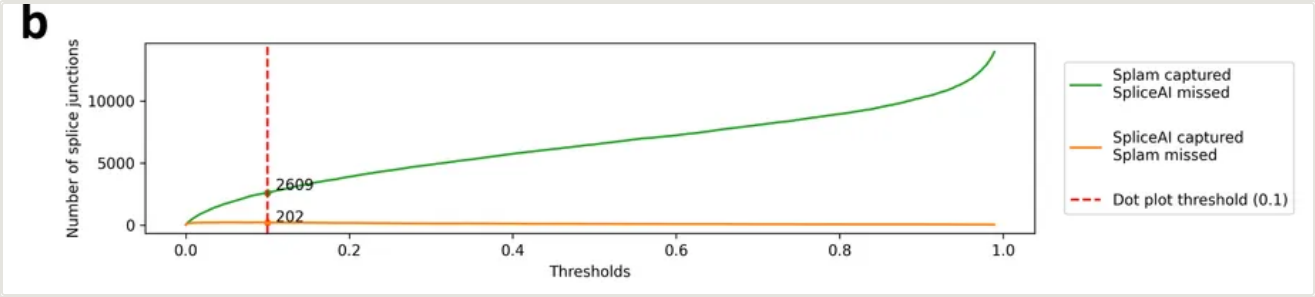
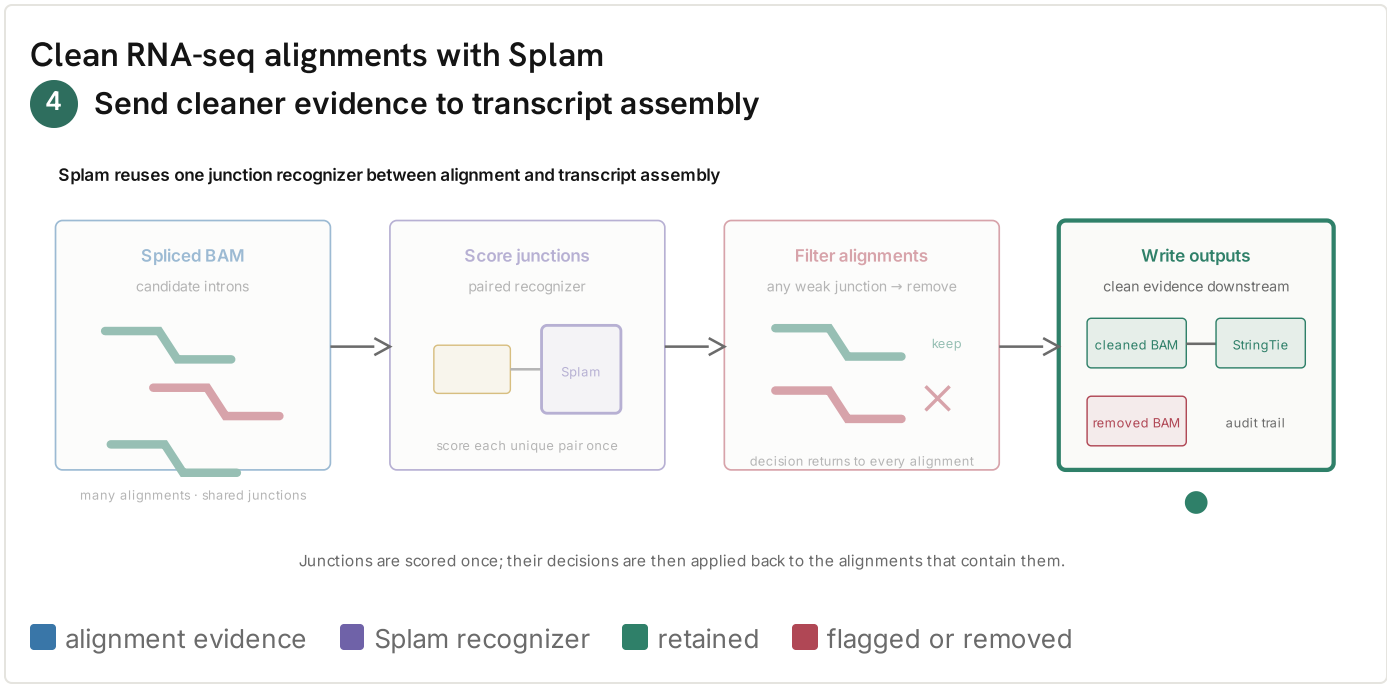


Figure 2. Catching alternative junctions. **(b)** Across score thresholds, Splam recovers more true splice junctions that SpliceAI labels negative (green) than the reverse case (orange). At the 0.1 threshold highlighted in the paper, the counts are 2,609 versus 202.

There is a tradeoff worth naming. In that same comparison, Splam had more false positives than SpliceAI at the selected threshold: 145 versus 16. The paper argues that, for this use case, false negatives on annotated MANE or RefSeq junctions are more likely to be genuine errors than false positives among splice-site-like unannotated sequences, which may include low-level or incompletely annotated splicing. That is the reasoning behind prioritizing recovery of true junctions while still using score thresholds to filter low-confidence alignments.

Because Splam scores a junction as a pair, its donor and acceptor scores also stay more consistent with each other. The algorithm generalized without retraining in the paper's tests on chimpanzee, mouse, and Arabidopsis thaliana: at a threshold of 0.8, junction recall was 91%, 87%, and 80%, compared with 57%, 47%, and 20% for SpliceAI on the same species.

And it cleans up alignments. This closes the loop back to the CHESS-scale problem. In the workflow evaluated in the paper, Splam sits after spliced alignment and before transcript assembly: it pulls each alignment's splice junctions out of the BAM file (the standard binary file of aligned reads), scores them, and removes alignments containing spurious low-scoring junctions — handing the assembler a cleaner file. The benchmark placed this step after HISAT2 alignment (Kim et al., 2019) and before StringTie assembly (Pertea et al., 2015), while the same cleanup slot also fits spliced aligners such as STAR (Dobin et al., 2013).



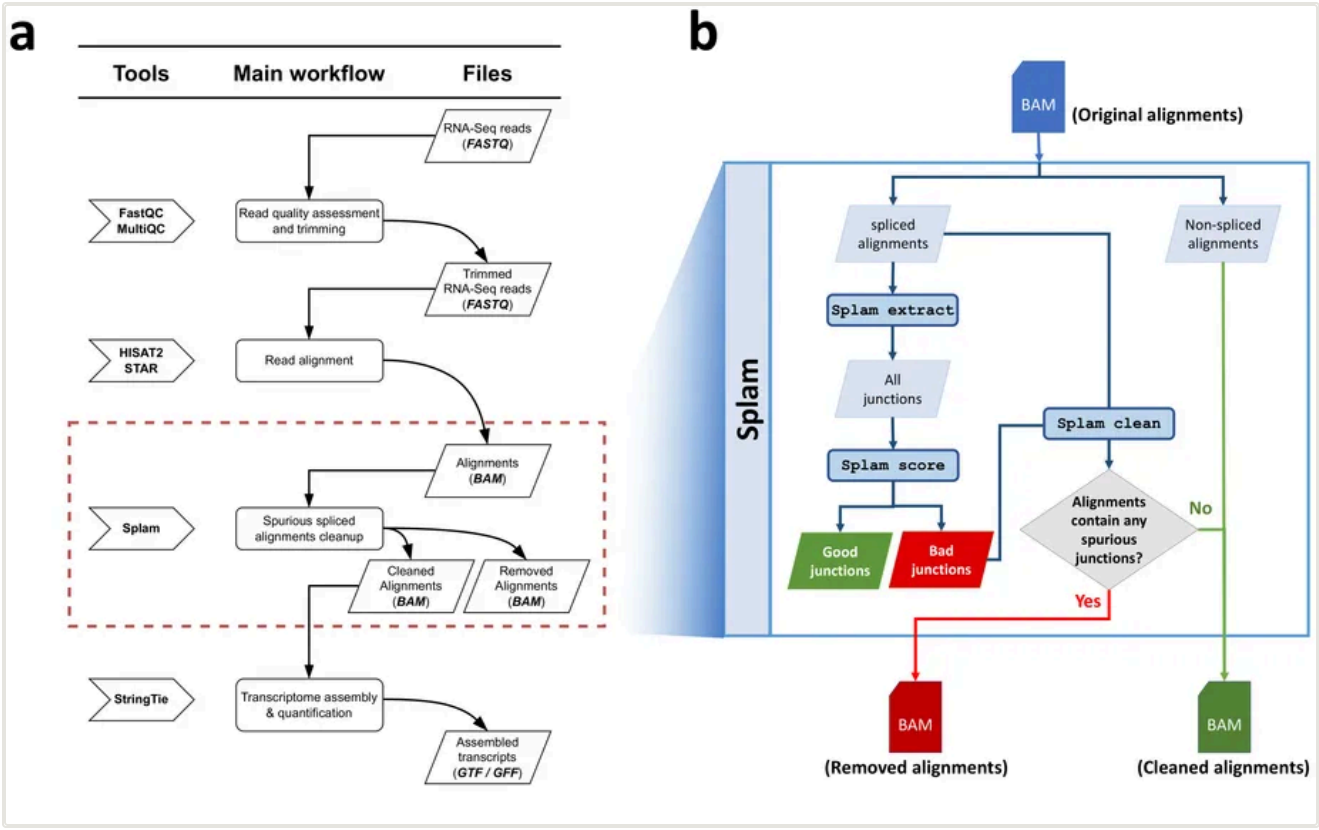


Figure 3. Splam as a pipeline step. **(a)** After reads are aligned (HISAT2/STAR), Splam cleans the spurious spliced alignments before transcript assembly (StringTie). **(b)** Internally, Splam extracts each alignment's junctions, scores them as good or bad, and writes out a cleaned BAM alongside a separate file of the removed alignments.

The payoff in the paper is concrete and bounded. In 20 human RNA-seq samples aligned with HISAT2, Splam cleaning raised intron precision while keeping RefSeq intron recall nearly unchanged: +14 to +25 percentage points of intron precision in 10 poly(A)-capture samples with less than 1% recall loss, and +28 to +37 points in 10 rRNA-depletion samples with a 0.7 to 1.0% recall drop. When the cleaned BAMs were assembled with StringTie, transcript precision improved in both library types; in the rRNA-depletion samples, the assemblies also gained about 500 to 600 RefSeq-matching transcripts.

The bigger picture

Splam is a focused algorithm with a clear job: evaluate candidate splice junctions and use those scores to improve RNA-seq alignments. It came from combining the inspiration of SpliceAI with a practical transcriptome assembly problem: too many spurious splice junctions make downstream assembly noisy, but overly aggressive filtering risks losing real isoform structure.

The lessons I carried forward are the design choices supported by these benchmarks. In the tested candidate-junction setting, local sequence around both splice sites was sufficient for strong recognition performance. Modeling the donor-acceptor pair made the two ends of an intron more consistent, and training on multiple annotated isoforms improved recovery of alternative junctions. Those same ideas shaped my later splicing work, including OpenSpliceAI, the open, retrainable reimplementation of SpliceAI.

Splam is free and open source, with code and documentation online for applying the same junction-scoring and alignment-cleaning workflow.

Read the paper in Genome Biology, browse the code, work through the documentation, or watch the talk. Splam was built with Alan Mao, Steven Salzberg, and Mihaela Pertea at Johns Hopkins.

References

1. Chao, K.-H., Mao, A., Salzberg, S. L., and Pertea, M. Splam: a deep-learning-based splice site predictor that improves spliced alignments. Genome Biology (2024). doi:10.1186/s13059-024-03379-4
2. Varabyou, A. et al. CHESS 3: an improved, comprehensive catalog of human genes and transcripts based on large-scale expression data, phylogenetic analysis, and protein structure. Genome Biology (2023). doi:10.1186/s13059-023-03088-4
3. Jaganathan, K. et al. Predicting splicing from primary sequence with deep learning. Cell (2019). doi:10.1016/j.cell.2018.12.015
4. Morales, J. et al. A joint NCBI and EMBL-EBI transcript set for clinical genomics and research. Nature (2022). doi:10.1038/s41586-022-04558-8

5. O'Leary, N. A. et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. Nucleic Acids Research (2016). doi:10.1093/nar/gkv1189
6. Kim, D. et al. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. Nature Biotechnology (2019). doi:10.1038/s41587-019-0201-4
7. Pertea, M. et al. StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. Nature Biotechnology (2015). doi:10.1038/nbt.3122
8. Dobin, A. et al. STAR: ultrafast universal RNA-seq aligner. Bioinformatics (2013). doi:10.1093/bioinformatics/bts635