BIOINFORMATICS ORIGINAL PAPER Vol. 26 no. 12 2010, pages 1500–1505
doi:10.1093/bioinformatics/btq206

Gene expression

Advance Access publication April 21, 2010

Supersplat—spliced RNA-seq alignment

Douglas W. Bryant Jr[1,2], Rongkun Shen[1,†], Henry D. Priest[1], Weng-Keen Wong[2] and Todd C. Mockler[1,*]

[1]Department of Botany and Plant Pathology and Center for Genome Research and Biocomputing and [2]Department of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331, USA

Associate Editor: Ivo Hofacker

ABSTRACT

Motivation: High-throughput sequencing technologies have recently made deep interrogation of expressed transcript sequences practical, both economically and temporally. Identification of intron/exon boundaries is an essential part of genome annotation, yet remains a challenge. Here, we present supersplat, a method for unbiased splice-junction discovery through empirical RNA-seq data.

Results: Using a genomic reference and RNA-seq high-throughput sequencing datasets, supersplat empirically identifies potential splice junctions at a rate of ∼11.4 million reads per hour. We further benchmark the performance of the algorithm by mapping Illumina RNA-seq reads to identify introns in the genome of the reference dicot plant *Arabidopsis thaliana* and we demonstrate the utility of supersplat for *de novo* empirical annotation of splice junctions using the reference monocot plant *Brachypodium distachyon*.

Availability: Implemented in C++, supersplat source code and binaries are freely available on the web at http://mocklerlab-tools.cgrb.oregonstate.edu/

Contact: tmockler@cgrb.oregonstate.edu

Received on September 23, 2009; revised on April 9, 2010; accepted on April 16, 2010

1 INTRODUCTION

Recent advancements in high-throughput sequencing (HTS) technologies (reviewed in Fox *et al.*, 2009; Shendure and Ji, 2008) have made deep interrogation of expressed transcript sequences both economically and temporally practical, resulting in massive quantities of sequence information using the RNA-seq approach (Wang *et al.*, 2009). Extracting comprehensible genic models from this sea of data depends upon the identification of intron/exon boundaries.

One current method used to identify intron/exon boundaries, Q-PALMA (De Bona *et al.*, 2008), utilizes a machine learning algorithm to identify splice junctions, training a large margin classifier on known splice junctions from the genome of interest. This method depends upon the availability of previously known splice junctions on which to train the algorithm, and, when finding novel potential splice junctions, is biased toward those which are similar to its training data. In scoring novel potential splice junctions, the algorithm is biased toward canonical terminal dinucleotides, scoring those which conform to these biases higher than ones that do not. While, in general, these biases may prove to be correct, many potential splice junctions which do not conform to these rules threaten to remain unidentified.

A second method, TopHat (Trapnell *et al.*, 2009), works by first creating exonic coverage islands from short-reads and then, based on canonical intron terminal dinucleotides (ITDN), which exist in these islands, identifies possible splices between neighboring exons. Like QPALMA, TopHat is strongly built around the idea of canonical ITDN, resulting in similar issues to QPALMA. Further, since TopHat bases its predictions on coverage islands, a sufficient number of short RNA-seq reads must be used as input such that reliable exon islands may be identified.

Here, we present our algorithm implemented in C++, supersplat, which identifies all locations in a genomic reference sequence that indicate potential introns supported by empirical transcript evidence derived from RNA-seq data. Our approach does not include bias for canonical ITDN, but rather finds every potential splice junction supported by empirical evidence, doing so in a straightforward, transparent manner and guided only by user-provided values for minimum intron size and maximum intron size, and by the minimum number of matching short-read nucleotides allowed on each flanking exon. Further, any number of short reads may be used as input, since supersplat does not need or attempt to generate exon coverage islands.

2 ALGORITHM

2.1 Definition

Supersplat begins by loading the input reads, and their reverse complements, into a hash table as key-value pairs, storing the nucleotide sequence of each read as keys and the number of occurrences of each read as corresponding values. This limits the amount of system memory required to store the input reads to a single copy of each unique sequence. Next, the input reference sequences are read and processed.

For each reference sequence and starting at the reference sequence's first base, supersplat builds an index that holds the location of every encountered k-mer, where k ranges from the minimum read chunk size, c, to an upper length, i, both of which are specified by the user. This location index is stored in a hash table as key-value pairs, where the key is the encountered k-mer and the value is a sorted list of the reference-specific, one-indexed locations where the k-mer was found, illustrated in Figure 1.

Once a reference sequence has been indexed supersplat iterates through all unique reads identifying those which can, while satisfying user-specified conditions, be partitioned and matched exactly against the reference, thereby identifying potential splice junctions. Each m-base long read is partitioned in all possible two-chunk configurations, with chunk one starting at the

*To whom correspondence should be addressed.
†Present address: Vollum Institute, Oregon Health & Sciences University, Portland, OR 97239, USA

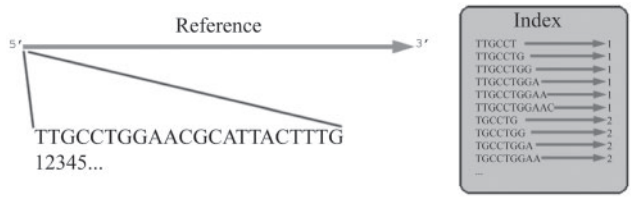

Fig. 1. Supersplat indexes a reference by starting at the first base in the reference sequence and stepping through the sequence, one base at a time. For each such stepping, b, supersplat stores each k-mer which begins at position b, where k ranges between the minimum read chunk size, c, and the MICS, i, both of which are specified by the user. In this figure's example, c is 6 and i is 11. Supersplat starts building the index by storing the first six bases of the reference, starting at the beginning of the reference, location 1, as a 6mer in the index, and associates that 6mer with a list of locations, which presently contains only location 1. Supersplat then stores the first seven bases of the reference as a 7mer in the index, and associates that 7mer with a list of locations, containing location 1. This continues until supersplat stores the first 11 bases of the reference as an 11mer, and associates that 11mer with a list of locations, containing location 1. Now that supersplat has reached $k = i = 11$, supersplat steps to the next base of the reference sequence, location 2. Supersplat now stores the first six bases of the reference, starting at reference location 2, as a 6mer in the index, and associates that 6mer with a list of locations, containing location 2. This process repeats until supersplat has indexed the entire reference sequence in this way.

minimum chunk length, c, growing iteratively by one base until chunk one is of length $m - c$, and chunk two starting at length $m - c$, shrinking iteratively by one base until chunk two is of length c.

For each such partitioning supersplat retrieves from the location index location-list one ($LL1$), corresponding to the exact k-mer represented by chunk one, and location-list two ($LL2$), corresponding to the exact k-mer represented by chunk two. If one of the chunks is longer than the largest k-mer indexed by supersplat, supersplat retrieves the location-list corresponding to the first i bases of that chunk.

Once $LL1$ and $LL2$ have been retrieved, for each element of $LL1$, $LL1_{0:k}$, supersplat iterates over all elements of $LL2$, $LL2_{0:l}$, comparing the locations of each element pair. If it is found that the minimum intron size, n, and the maximum intron size, x, both of which are user-specified, are satisfied by any such pair, supersplat ensures an exact match of chunks one and two to the reference sequence if necessary and, if a match is verified, marks the bounded genomic sequence as a possible intron.

This process repeats for each possible partitioning of each read and over each reference sequence. All possible splice junctions are output to a file by default, with an option to output only canonical ITDN matches if desired.

2.2 Complexity analysis

Let G be the reference sequence length, E_{LL1} be the number of elements in location-list one, E_{LL2} be the number of elements in location-list two, N_{reads} be the number of input reads, and suppose all input reads are of length R. The supersplat algorithm begins by indexing both the reference sequence and the input reads in the creation of two hash tables, the reference index and the reads index. These two caching structures are created in time $O(G)$ and $O(N_{\text{reads}})$, respectively.

Most of supersplat's processing time is spent performing iterative pairwise comparisons between location-lists, dwarfing the amount of time spent generating the indices. This iterative pairwise comparisons algorithm has overall time complexity of $O(N_{\text{reads}} * R * E_{LL1} * E_{LL2})$. The inner term $E_{LL1} * E_{LL2}$ is an upper bound; in reality, all possible pairs of elements in $LL1$ and $LL2$ need not be explored due to two optimizations. First, location-lists are limited in size by the length of the chunk being queried. If the size of one chunk is short, which results in a large number of genomic matches and thus a location-list that contains a large number of entries, its paired chunk is long, which results in the second location-list containing a relatively few number of entries. Second, location-lists are always sorted in increasing order of reference positions as a byproduct of the way in which the reference index is constructed. As a result of the location-lists being sorted, once a comparison has been made between elements of $LL1$ and $LL2$ where the distance between these two elements is greater than the maximum allowed intron size, all remaining elements in $LL2$ can be skipped for the current element of $LL1$.

The space complexity of the supersplat algorithm is linear in the number of reads, the reference size and the maximum index chunk size (MICS) parameter, resulting in the space complexity $O(N_{\text{reads}} * G * MICS)$.

3 RESULTS

3.1 Performance

Real-world performance of supersplat was tested with a set of 3 690 882 unique *Arabidopsis thaliana* Illumina RNA-seq reads (short read archive accession: SRA009031) (Filichkin *et al.*, 2010), each of which was known to map to an annotated (TAIR8) splice junction location on the *Arabidopsis* reference genome. To benchmark supersplat, the algorithm's primary performance parameter, MICS, was incremented iteratively by one from 9 to 18. For each such iteration, the total run time from reference indexing to final output, as well as maximum memory (RAM) usage, was recorded. Benchmarking was performed on a 3.0 GHz Intel Xeon processor with 32 GB memory. For all tests, the minimum intron size, n, was set to 40; the maximum intron size, x, was set to 5000; and the minimum chunk size, c, was set to 6, while the MICS, i, was varied. Performance results are shown in Figure 2. From these results, we see that as the MICS increases, runtimes decrease and RAM usage increases. Between MICS values of 9 and 15 each stepping decreases runtime by about a factor of 4, after which yields diminishing returns. For runtimes on these data, a MICS value of 15 was optimal, resulting in a runtime of 19.4 CPU minutes and with maximum RAM usage of ∼18.8 GB. This indicates an average of ∼190 252 reads mapped per CPU minute, or ∼11.4 million reads mapped per CPU hour.

3.2 Empirical annotation of splice junctions in *Brachypodium distachyon*

To demonstrate the utility of supersplat for *de novo* discovery of splice junctions, we mapped Illumina RNA-seq reads (short read archive accession: SRA010177) to the genome of the model grass *B.distachyon*. For this analysis we used ∼10.2 Gb (∼289 million 32mer reads) of Illumina transcript sequence generated using the RNA-seq approach (Fox *et al.*, 2009; The International Brachypodium Initiative, 2010). We used ELAND (A.J.Cox, unpublished data) to identify all 32mer Illumina reads that aligned anywhere in the *Brachypodium* genome with up to two mismatches. This step eliminated ∼79 million reads from further analysis. The remaining ∼210 million reads were filtered using DUST (Morgulis *et al.*, 2006) to remove reads containing low-complexity stretches, leaving ∼150 million reads that were aligned to the *Brachypodium* genome assemblies using supersplat.

Potential novel splice junctions predicted by supersplat were filtered to retain only those supported by at least two distinct independent RNA-seq reads with different overlap lengths on each side of the predicted intron (i.e. the portions of the reads aligning

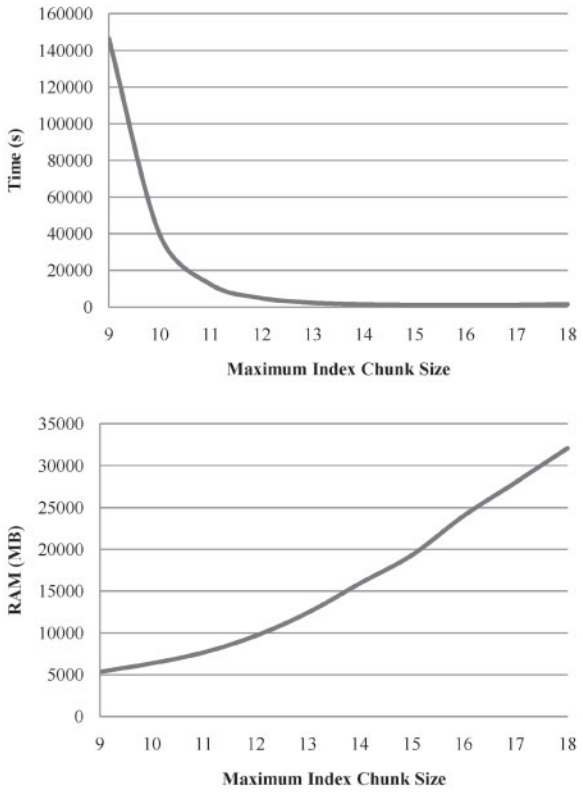

Fig. 2. By increasing the maximum index size, the exhaustive genome-to-reads comparisons are reduced resulting in shorter runtimes. This same increase correlates with an increase in peak RAM usage as a result of larger lookup tables.

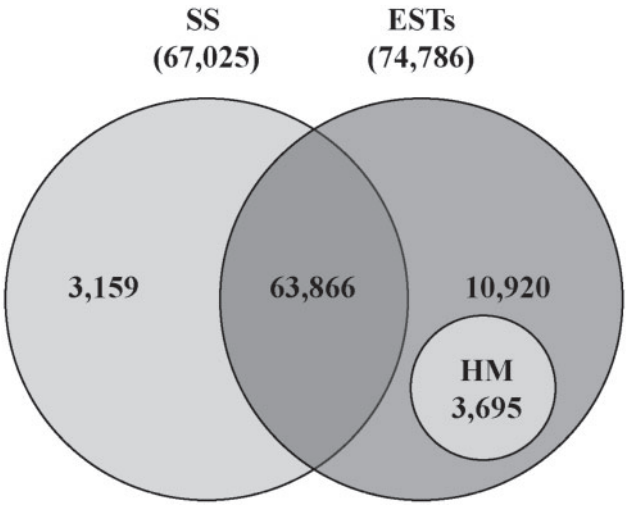

Fig. 3. A Venn diagram showing the comparison of supersplat predicted *Brachypodium* GT-AG introns against BradiV1.0 annotated GT-AG introns verified by *Brachypodium* ESTs. The 67 025 *Brachypodium* GT-AG introns (set SS) predicted by supersplat were supported by 1.55 million RNA-seq reads. The 74 786 BradiV1.0 annotated GT-AG introns (set ESTs) were verified by alignment of 2.29 million 454 reads and 128 000 Sanger reads. The 3695 introns in set HM are supersplat false negative introns that were missed by supersplat due to the minimum chunk size of 6 used in this analysis but verified as being supported by the RNA-seq data using HashMatch (Filichkin *et al.*, 2010).

to the predicted exons), reads mapping to only a single genomic location, a minimum overlap length of 6 bases on one exon, additional support of at least one microread matching each of the predicted flanking exonic sequences, a minimum predicted intron length of 20 and a maximum predicted intron length of 4000. This analysis, using *ad hoc* filters designed to reduce false discoveries, identified a total of 1.55 million RNA-seq reads supporting 67 025 introns containing canonical GT-AG terminal dinucleotides. These intron predictions are publicly available and presented in the *Brachypodium* community genome database and viewer found at http://www.brachybase.org. Among all 67 025 GT-AG introns predicted by supersplat in this experiment, 63 866 (95.3%) were independently validated (Fig. 3) by BradiV1.0 annotated introns verified by *Brachypodium* ESTs.

An example is presented in Figure 4, which depicts an empirically annotated *Brachypodium* gene encoding a protein similar to an annotated hypothetical protein in rice. In this example, the filtered supersplat results predicted 14 out of 15 introns depicted in the empirical TAU (H.D.Priest *et al*., unpublished data) models. The one intron of this gene not predicted by supersplat was inferred from other Sanger and 454 EST data (data not shown).

4 DISCUSSION

Supersplat aligns RNA-seq data to a reference genome as a gapped alignment in order to empirically define locations representing potential introns. Unlike the other comparable tools, Q-PALMA and TopHat, supersplat is not inherently biased in favor of canonical ITDN, but instead by default exhaustively identifies every potential splice junction supported by the input data. In another study (Filichkin *et al.*, 2010), we found that following conservative filtering of supersplat output we were able to independently validate ∼91% and ∼86%, respectively, of canonical and non-canonical predicted introns that were tested by RT-PCR and Sanger sequencing. Supersplat does provide a canonical ITDN option, which has been incorporated because, as in our *Brachypodium* example, in some cases users may prefer to only mine their RNA-seq data for introns containing the most common ITDNs rather than the far less-common non-canonical ITDNs. Other user-provided parameters that limit the supersplat alignment algorithm are the minimum and maximum allowable intron sizes and the minimum overlap of a read on a flanking exon.

Supersplat's exhaustive and unbiased approach comes at the cost of large unwieldy output files, which can reach the size of tens of gigabytes for large sets of RNA-seq data. In particular, reads matching repetitive sequences or reads containing low-complexity stretches can match in numerous places in a reference genome as false spliced alignments. Users should, therefore, carefully determine appropriate criteria for prefiltering potentially problematic data prior to running supersplat. For example, as described for our empirical annotation of splice junctions in *B.distachyon*, reads that are likely to represent exonic data by virtue of their alignment to the target genome over their entire length should be removed prior to running supersplat. In addition, it is a wise precaution to filter out low-quality reads, low-complexity reads and highly repeated reads likely to result in numerous

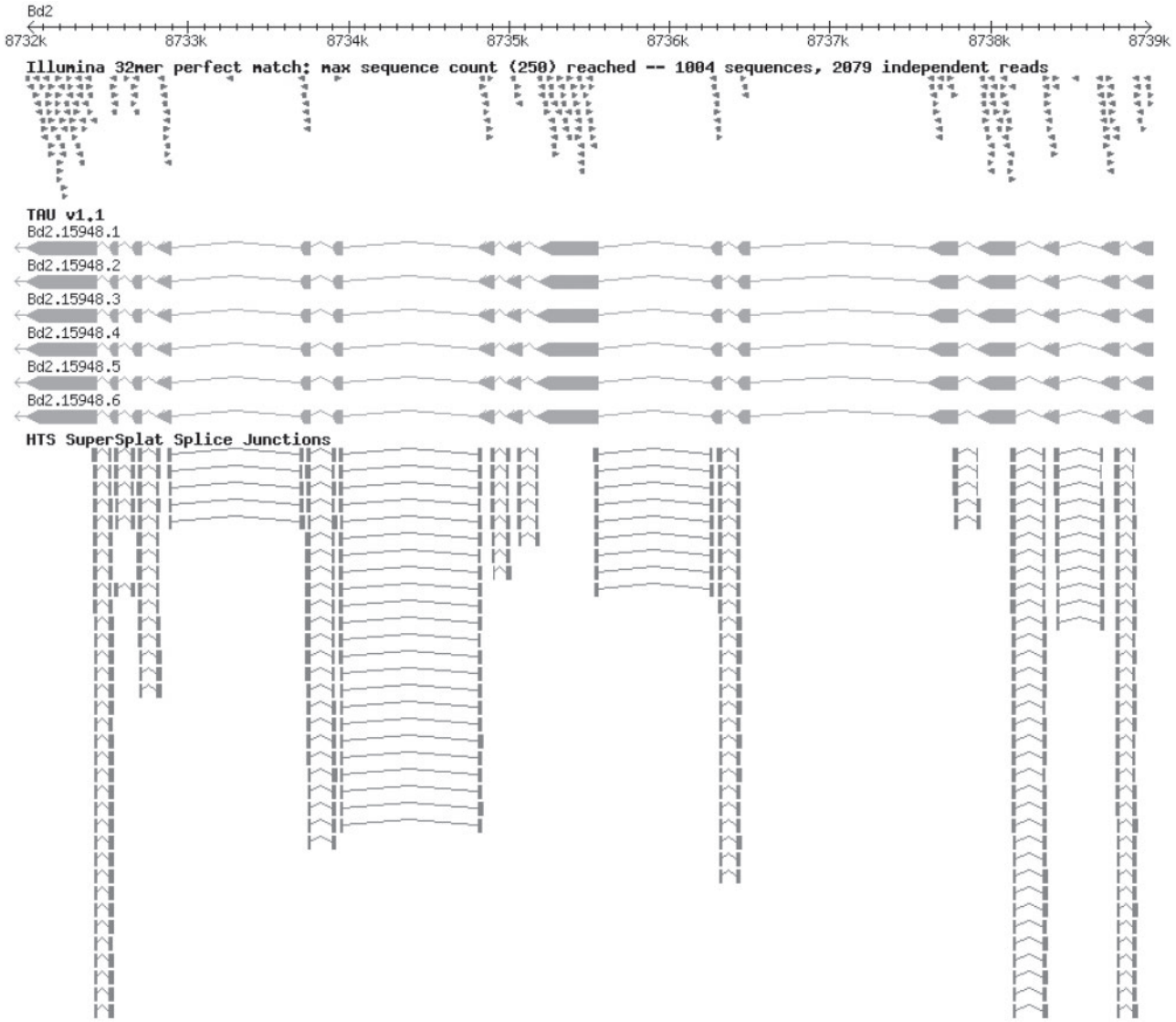

Fig. 4. An example of filtered supersplat output displayed in GBrowse v1.69 at BrachyBase (http://www.brachybase.org). The 'Illumina 32mer perfect match' track represents the distribution of perfectly matching 32 nt Illumina HTS RNA-seq reads over the region. 'HTS SuperSplat Splice Junctions' are Illumina reads aligned using supersplat specifically to identify putative introns. The 'TAU v1.1' track depicts empirical transcription unit models derived from transcript data, including the splice junctions predicted by supersplat.

spurious alignments. In the event that an annotation exists for the genome of interest, reads matching annotated splice junctions can be filtered out of the input in order to focus the supersplat analysis only on the identification of potential novel introns. Sensible selection of runtime options and post-processing steps are good precautions to control false discoveries. For example, users may want to choose reasonable limits for minimum and maximum intron lengths guided by prior data. In addition, as described in our examples, supersplat output can be filtered to retain only those intron predictions supported by some minimum chunk size, multiple independent overlapping RNA-seq reads, introns supported by reads mapping to only a single genomic location, validation by RNA-seq data from independent biological replicates, multiple different overlap lengths for the read fragments on the flanking exons and additional transcript evidence supporting the predicted exons. Despite these precautions factors such as read lengths, sequencing error rates, target genome complexity and gene duplications contribute to the likelihood of false discoveries with supersplat. Some of these issues will no doubt be resolved by improvements to HTS technologies such as increased read lengths, reduced error rates and routine use of paired-end reads.

Using our test set of reads matching known *A.thaliana* splice junctions, we performed an analysis of supersplat's precision while focusing on two of these standard filters independently, including minimum chunk size and number of overlapping reads. Precision, also known as positive predicted value (PPV), is defined as true positives (TP) divided by the sum of TP and false positives (FP), PPV = TP/(TP + FP). It is worth noting that an algorithm's PPV can be skewed by generating only a small number of very cautiously

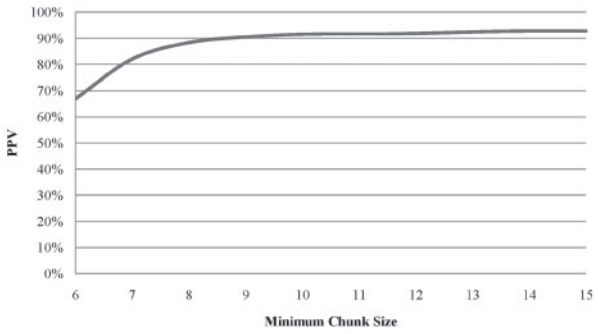

Fig. 5. PPV versus minimum chunk size. As minimum chunk size is varied from 6 to 15 the precision of supersplat rapidly approaches and exceeds 90%. Here, the PPV denominator, TP + FP, ranges over 360 237 (minimum chunk size of 6) to 260 495 (minimum chunk size of 15).

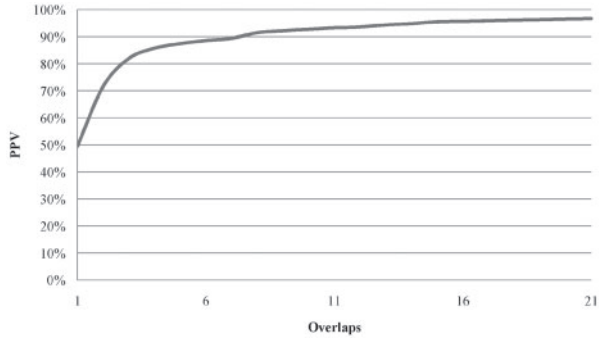

Fig. 6. PPV versus number of reads overlapping each splice junction. As the number of overlapping reads is varied from 1 to 21, the precision of supersplat rapidly approaches and exceeds 90%, reaching 97% with 21 overlapping reads. Here, the PPV denominator, TP + FP, ranges over 244 782 (single read) to 124 219 (21 overlapping reads).

declared positives, resulting in a very small but highly confident output set. Supersplat, in contrast, generates exhaustive output that is not filtered according to confidence. As a result the PPVs presented here are computed using large denominators ensuring that this metric is an accurate reflection of supersplat's performance.

In our analysis, a true positive prediction is one that correctly identifies a location in the genome at which there exists The *Arabidopsis* Information Resource (TAIR)-annotated splice junction. A FP prediction is one that incorrectly identifies a location in the genome at which there is no such TAIR annotated splice junction. PPV was calculated as minimum chunk size was varied from 6 to 15 with results shown in Figure 5, filtering in this case for intron predictions in the supersplat output which had the minimum chunk size shown. From these results, we see that even at a minimum chunk size of six, the precision of supersplat is nearly 70%. As minimum chunk size increases this precision value rapidly approaches and exceeds 90%.

PPV was then calculated as the number of reads overlapping each splice junction was varied from 1 to 21 with results shown in Figure 6. In this case, we filtered for intron predictions in the supersplat output that had the respective number of intron overlaps shown. From these results, we see with six overlapping reads a PPV of 90%, with PPV reaching 97% at 21 overlapping reads.

Runtime performance of supersplat is closely tied to how deeply the genomic reference is indexed, dictated by the MICS value. Supersplat repeatedly queries its index for the genomic locations of various sized k-mers, which represent potential short read fragment alignments. Since the probability of any specific k-mer existing at a particular genomic location, under the assumption that all bases occur with equal probability, is 0.25^k, as k increases the probability of any specific k-mer occurring decreases. Thus we expect, on average, that the list of all locations in a genomic reference sequence of a specific k-mer to be longer by a factor of four than a similar list of a specific $(k + 1)$-mer. As supersplat processes short reads from its input, it repeatedly iterates over these location lists. As the MICS value increases the lengths of these lists decrease, reducing runtimes by about a factor of four for each increase in the MICS value. When the MICS value becomes sufficiently large, which for these data was around 15, any particular MICS-sized k-mer occurs so rarely in the genomic reference sequence that further increases in the MICS value will yield little to no decrease in runtime.

Identification of reads spanning splice junctions is essential for RNA-seq-based studies of alternative splicing (Filichkin *et al.*, 2010; Sultan *et al.*, 2008) and for assembly of empirical transcription unit models from RNA-seq datasets using tools such as G-Mo.R-Se (Denoeud *et al.*, 2008), Cufflinks http://cufflinks.cbcb.umd.edu/ or TAU (H.D.Priest *et al.*, unpublished data). As demonstrated, supersplat is an effective algorithm for mining potential splice junction reads from RNA-seq data and its exhaustive search of the potential splice junction sequence space can uncover many previously unknown splice junctions given sufficiently deep transcriptome data.

ACKNOWLEDGEMENTS

We thank Dr Sergei Filichkin, Samuel Fox, Mark Dasenko and Steve Drake for assistance with Illumina sequencing, and Chris Sullivan and Scott Givan for assistance with bioinformatics and visualization.

Funding: Oregon State University startup funds (to T.C.M.); National Science Foundation Plant Genome (grant DBI 0605240, partially); Department of Energy Plant Feedstock Genomics for Bioenergy (grant DE-FG02-08ER64630); Oregon State Agricultural Research Foundation (grant ARF4435 to T.C.M.); Computational and Genome Biology Initiative Fellowship from Oregon State University (to H.D.P.).

Conflict of Interest: none declared.

REFERENCES

Denoeud,F. *et al.* (2008) Annotating genomes with massive-scale RNA sequencing. *Genome Biol.*, **9**, R175.

De Bona,F. *et al.* (2008) Optimal spliced alignments of short sequence reads. *BMC Bioinformatics*, **24**, i174.

Filichkin,S.A. *et al.* (2010) Genome wide mapping of alternative splicing in Arabidopsis thaliana. *Genome Res.*, **20**, 45–58.

Fox,S. *et al.* (2009) Applications of ultra high throughput sequencing in plants. *Plant Syst. Biol.*, **553**, 79–108.

Morgulis,A. *et al.* (2006) A fast and symmetric DUST implementation to mask low-complexity DNA sequences. *J. Comput. Biol.*, **13**, 1028–1040.

Shendure,J. and Ji,H. (2008) Next-generation DNA sequencing. *Nat. Biotechnol.*, **26**, 1135–1145.

Sultan,M. *et al.* (2008) A global view of gene activity and alternative splicing by deep sequencing of the human transcriptome. *Science*, **321**, 956–960.

The International Brachypodium Initiative (2010) Genome sequencing and analysis of the model grass Brachypodium distachyon. *Nature*, **463**, 763–768.

Trapnell,C. *et al.* (2009) TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, **25**, 1105–1111.

Wang,Z. *et al.* (2009) RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet.*, **10**, 57–63.