A tandem simulation framework for predicting mapping quality

Ben Langmead[1,2, 3]

[1]Department of Computer Science, Whiting School of Engineering, Johns Hopkins University

[2]Center for Computational Biology, McKusick-Nathans Institute of Genetic Medicine, JHU

[3]Department of Biostatistics, Bloomberg School of Public Health, JHU

Read alignment is the first step in most sequencing data analyses. It is also a source of errors and interpretability problems. Repetitive genomes, algorithmic shortcuts, and genetic variation impede the aligner's ability to find a read's true point of origin. Aligners therefore report a mapping quality: the probability the reported point of origin for a read is incorrect. However, there is no established method for calculating mapping quality in a general way. We describe an accurate, aligner-agnostic framework for predicting mapping qualities that works by simulating a set of tandem reads, similar to the input reads in important ways, but for which the true point of origin is known. Alignments of tandem reads are used to build a model for predicting mapping quality, which is then applied to the input-read alignments. The model is automatically tailored to the alignment scenario at hand, allowing it to make accurate mapping-quality predictions across a range of scenarios. We implement this approach in a software tool called Qtip, which is accurate, low-overhead, and compatible with popular read aligners. Qtip is open source software available from `https://github.com/BenLangmead/qtip`.

1 Introduction

Read alignment is often the first task when analyzing sequencing data. This is the process of determining each read's point of origin with respect to a reference genome. Much prior work is concerned with making read aligners computationally efficient[1]. That said, a read's point of origin can be ambiguous, and the reported alignments can be incorrect[2]. Repetitive genomes, sequencing errors and genetic differences contribute to the problem. In addition to being efficient, aligners must accurately characterize the uncertainty associated with each alignment, as first proposed in the seminal MAQ study[2] which coined the term "mapping quality." Aligners have methods for predicting mapping quality, which is reported in the MAPQ field of the SAM/BAM format[3]. These methods are generally quite ad hoc, not well described in research literature or software manuals.

We introduce the *tandem simulation* framework for predicting mapping qualities in a manner that is agnostic to the aligner and parameters used. We introduce a tool called Qtip that implements the framework. Qtip operates alongside and in cooperation with an aligner like Bowtie 2[4]; the term "tandem simulation" refers to this cooperation. After observing the input reads and alignments, Qtip trains an ensemble tree model for predicting mapping qualities. Training uses simulated "tandem" reads, which are randomly drawn from the genome but crafted in a way that mimics statistical properties of the input reads, including their length, quality, gap, and edit distributions. The aligner itself must be modified to report feature data for the model, but alignment algorithms need not be changed. We implemented changes for the Bowtie 2[4], BWA-MEM[5] and SNAP[6] aligners. Qtip works with any aligner that outputs feature data in a special SAM field; it is not limited to the tools adapted for this study.

We demonstrate Qtip's predictions are superior to those made by the read aligners themselves, both on average and for most specific MAPQ cutoffs tested. We use simulation experiments to show this for various read aligners (Bowtie 2, BWA-MEM , SNAP), alignment settings (read lengths, alignment parameters, species), and accuracy criteria.

We also perform a variant-calling experiment to show the improved mapping qualities can benefit downstream analysis. To our knowledge this is the first description of a general technique for characterizing alignment uncertainty, applicable across software tools and alignment settings. Qtip is open source software available from `https://github.com/BenLangmead/qtip` and experiments can be reproduced using the software at `https://github.com/BenLangmead/qtip-experiments/tree/v1.6.1`.

2 Background

Alignment errors. Given a sequencing read and reference genome, a read aligner like Bowtie 2[4], BWA-MEM[5] or SNAP[6] searches for the read's highest-scoring alignment to a substring of the reference. Alignment score measures the degree of similarity between the strings, with a higher score indicating fewer mismatches and gaps. If more than one alignment has maximal score, one is chosen arbitrarily. Though many aligners can be configured to report more than one alignment per read, we assume here that just one is reported, as is common. If the reported alignment does not correspond to the read's true origin, the alignment is *incorrect*, and we call this an *alignment error*. Incorrect alignments lead to interpretation problems later on[7,8].

Aligners use heuristics — computational shortcuts — to limit effort expended. Heuristics affect which alignments can and cannot be found, shaping what errors the aligner might make. Supplementary Material section S.1 outlines the heuristics used in Bowtie 2. While heuristics can cause valid alignments to be missed, this is a reasonable compromise since searching without heuristics is too slow in practice.

Alignment errors belong to three categories, as defined in the MAQ study[2]:

1. The read is reported to have originated from a locus in the reference genome, but actually originates from a sequence not included in the reference
2. No alignment to the reference is found, but the read actually originates from some

locus in the reference

3. An alignment to locus L_r in the reference is reported, but the read actually originates from a different locus in the reference, L_t

Category-1 errors can be caused by contaminating DNA, or from the use of an incomplete or inappropriate reference genome sequence, e.g., a draft assembly or mismatched strain. Category-2 errors can occur when the alignment at L_t falls below the minimum similarity threshold (S_{min}), or when the alignment at L_t is missed due to alignment heuristics. Category-3 errors are caused by a combination of repetitive DNA, sequencing errors, genetic differences between subject and reference, and alignment heuristics.

Category-3 errors and the related idea of "multi mappers," reads that align equally well to many loci, are discussed in prior studies[9,10]. Category-3 errors are also the most numerous. We confirm this here by using the Mason simulator to construct several collections of synthetic Illumina-like reads from the human genome and from several contaminating genomes. We simulated both 100-nt and 250-nt reads, and both unpaired and paired-end reads. We also constructed a similar collection using mouse as the source genome. We aligned the reads with Bowtie 2 v2.3.0 and observed alignment error rates ranging from 1.8–8% depending on read length and paired-end status. 96.3–99.7% of the errors were of category 3. 0.2–3.9% of the errors were of category 2. Very few (<0.15%) were of category 1. This experiment is detailed in Supplementary Material section S.2 and Supplementary Table S1.

Here we focus on the task of predicting mapping qualities for aligned reads in light of category-3 errors. Category-2 errors are not considered, since no mapping quality prediction is needed in those cases. Although category-1 errors affect mapping quality prediction, we assume they are rare enough to be ignored. In principle, category-1 errors could be included in our model, e.g. by assuming a global prevalence of category-1 error and scaling predictions accordingly, or by including contamination in the tandem simulation.

Mapping quality. While searching for alignments, aligners uncover information that can be used to predict whether a given alignment is correct. For instance, if the aligner discovers that a read aligns equally well to several copies of a repeat, its confidence that the selected alignment is correct will be low. If the aligner discovers that a read aligns perfectly to one locus and very poorly to a few others, confidence will be higher.

Confidence is measured as the probability p that the reported alignment is correct. Let the *mapping quality* $q = -10 \cdot \log_{10}(1 - p)$. Higher values for p (or q) indicate higher confidence. The widely used SAM/BAM format[3] requires that q, rounded to the nearest integer, be reported in the `MAPQ` field of each alignment. We therefore seek a method that predicts q (or equivalently, p) accurately across a range of alignment scenarios.

Mapping quality measures something distinct from alignment score. High alignment score indicates high sequence similarity (few mismatches and gaps) between read and reference. It does not imply high mapping quality. For instance, consider a read that aligns with no gaps or mismatches to two distinct loci in the reference. Alignment score is high because there are no gaps or mismatches, but there is only a 50% chance of choosing the correct alignment ($q \leq 3$). Other measures that do not take genomic repeats into account, such as E-values[11], are also poor proxies for mapping quality.

Related work. The MAQ study[2] describes sources of alignment error and presents a model for predicting q. Model inputs include quality-weighted alignment scores for best and second-best alignments, and the total number of alignments found that have the same score as the second-best alignment. Successors to MAQ, such as BWA[12], BWA-SW[13] and BWA-MEM[5] use more complex functions to predict q, including for paired-end alignments. In BWA-MEM q is a function not just of the best and second-best alignment scores, but also information about whether and how seeds — substrings of the read — match the genome. Qtip uses similar data to train its model, but, because predictions are made in a separate tool running alongside, the aligner aligner author is freed from writing this function, which can be difficult. Qtip's approach also works for both local and end-to-end alignments, whereas BWA-SW and BWA-MEM are strictly local aligners.

ARDEN[14] uses a mutated "decoy" version of the reference genome to estimate the aggregate prevalence of category-3 errors. However, ARDEN does not attempt to predict q for individual alignments, and so cannot be used in the scenarios we describe. LoQuM[15] uses a set of simulated training alignments and a logistic regression model to predict new qs for an already-aligned dataset. Unlike Qtip, though, LoQuM does not predict q from scratch; rather, it "recalibrates" q using the aligner-reported mapping quality as an input, along with other inputs derived from the alignment.

MOSAIK[16] is a read aligner that uses a neural network model to predict q. The user must train the model ahead of time, by supplying it with simulated reads annotated with their true point of origin. Features used in the model include the alignment scores of the best and second-best alignments, read sequence entropy, and the number of potential mapping locations. Training involves generating a simulated dataset, but this is not done automatically. To produce training data, the user must set up a simulation experiment that must presumably mimic the real data in important ways. Tandem simulation works similarly to MOSAIK's approach, building a model from simulated reads, but without requiring the user to conduct the training.

Tandem simulation also has similarities to a previous method for allele-specific expression proposed by Hodgkinson et al[17]. In that method, RNA sequencing reads are aligned to a reference genome and allelic ratios are computed at heterozygous sites. The method then simulates a high-coverage "null" dataset with two key properties. First, the genome from which the reads are simulated is customized to include non-reference alleles detected by a separate genetic assay of the same individual. Second, when the simulator creates reads that overlap heterozygous variants, the two alleles are sampled in a perfect 50/50 ratio. Null reads are aligned to the original reference genome using the same read aligner and parameters as in the initial alignment step, much like the alignment of tandem reads in our framework. Allelic ratios derived from null alignments are used to normalize the allelic ratios derived from the original reads, reducing alignment bias. While the method by Hodgkinson et al and our method have distinct goals, they are alike in their use of a newly simulated dataset to improve results from an initial alignment.

3 Methods

Tandem simulation. The user provides a collection of n input reads: $R = r_0, r_1, ..., r_{n-1}$. The user also specifies a read aligner, parameters for the read aligner, a reference genome in FASTA format, and any other files required by the aligner, such as an index. The tandem simulation framework (Figure 1) aligns the input reads to the reference genome and predicts a mapping quality q_i for each aligned read. In step 1, input reads are aligned to the reference genome using the specified aligner and parameters. In step 2, the SAM-formatted[3] alignments are parsed and an *input model*, capturing information about the input reads and their alignments, is built. In step 3, the input model and reference genome are used to simulate a new set of reads, called *tandem* reads since they originate from tandem simulation. Each tandem read is from a random location in the genome and is labeled with its true point of origin. In step 4, tandem reads are aligned to the reference genome using the same aligner and parameters as in step 1. In step 5, the alignments produced in step 4 are parsed and converted to *training records*. Because the true point of origin is known, each training record can be labeled as correct or incorrect. In step 6, a model is trained on the records from step 5. In step 7, SAM alignments from step 1 are parsed. For each aligned read, a *test record*, like the training record from step 5, is constructed. Based on the test record, the trained model is applied to predict q_i. The alignment's SAM record is then re-written substituting q_i in the MAPQ field.

Importantly, the mapping quality model trained in step 6 is tailored to the alignment scenario at hand. The same aligner and alignment parameters from step 1 are used again

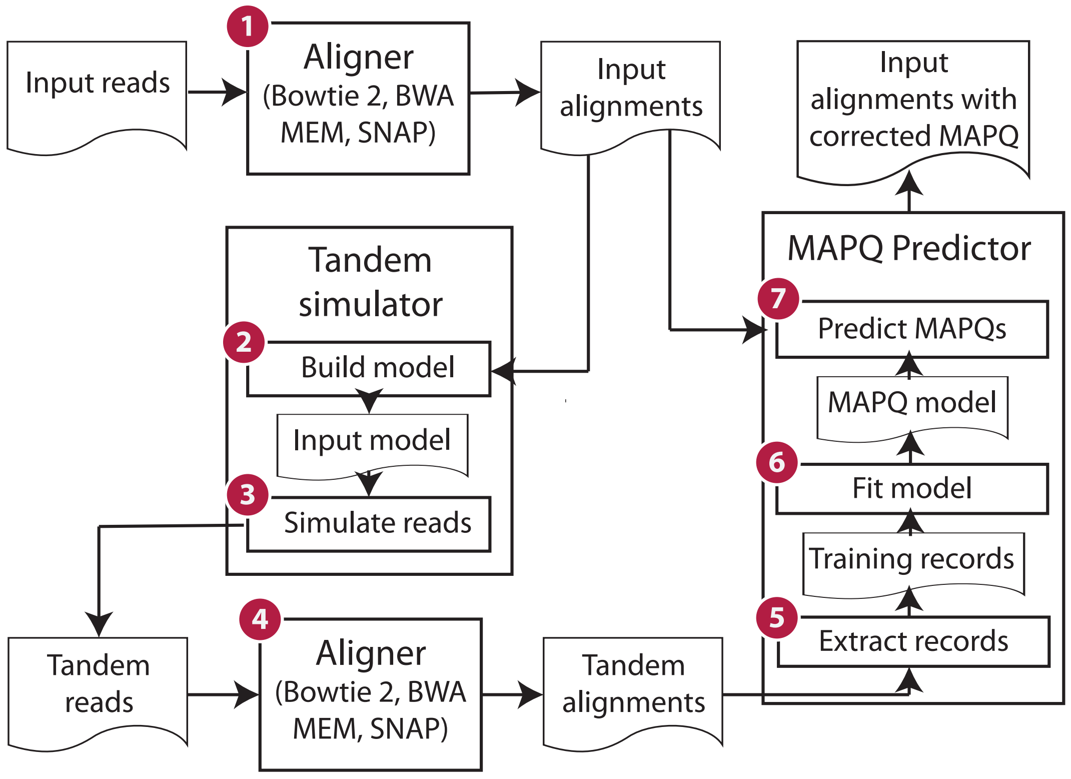

Figure 1: Stages of the Qtip pipeline, highlighting computation steps and intermediate results. Numbers denote ordering of steps and arrows denote flow of data. The input (upper left) is a collection of sequencing reads and the ultimate output (upper right) is SAM file containing alignments, where each aligned read's MAPQ field is set according to Qtip's prediction.

in step 4, and the tandem reads generated in step 3 mimic the input reads. This is in contrast to the strategy of MOSAIK[16], which requires the user to specify a trained neural network at alignment time. MOSAIK's strategy is less convenient and, if the user-specified training set is a poor match for the input reads, can yield poor predictions.

To work with the tandem simulation framework, the read aligner must report the feature data — how well the read aligned, what other alignments were found, what heuristics were used, etc — used to train and apply the model. This requires modifications to the alignment software. The modifications are not complex and do not affect efficiency or accuracy of the aligner. However, making appropriate modifications requires knowledge of how the aligner works and of which intermediate alignment results constitute informative features. For this study, we adapted three tools: Bowtie 2 v2.3.0, BWA-MEM v0.7.15 and SNAP v1.0beta.18. Supplementary Material section S.3 provides links to our modifications and details about the modifications made and how features were chosen.

We chose these three aligners both because of their popularity and because they together support a breadth of alignment scenarios. For example, Bowtie 2 and BWA-MEM support local alignment, Bowtie 2 and SNAP support end-to-end alignment, and all three tools support both unpaired and paired-end alignment. Also, all three tools produce their own mapping quality predictions. We will evaluate Qtip's mapping quality predictions in all of these modes, comparing Qtip's predictions to those made by the aligners.

Read and alignment categories. When predicting mapping quality, Qtip uses a different model depending on whether the alignment is unpaired (*unp*), paired-end and concordantly aligned (*conc*), paired-end and discordantly aligned (*disc*), or paired-end with the opposite end having failed to align (*bad-end*). Qtip trains each model with alignments of the same category. Qtip parameters control the minimum number of tandem reads or pairs of each category to generate, either as a fraction of the number of input reads or as an absolute number. The default number of training reads/pairs generated for each category is $45 \cdot \sqrt{x}$ where x is the number of input alignments of that category. Both the scaling factor and the function are configurable via Qtip's `--sim-function` and `--sim-factor` parameters. Qtip enforces a minimum of 30,000 tandem reads in the case of the *conc* and *unp* categories and 10,000 in the case of the *disc* and *bad-end* categories.

Input model and simulation of tandem reads. The input model built in step 2 of Qtip (Figure 1) captures information about the input reads and alignments. Qtip uses this to simulate new tandem reads that are from random genomic locations but are similar to the input reads in key ways, mimicking their read length distribution, quality strings, and patterns of gaps and mismatches. Tandem paired-end reads additionally mimic the input's fragment length distribution and relative orientation of the two ends.

To accomplish this, Qtip takes the following approach. For each aligned unpaired read, a corresponding *template* is created. The template is a data structure consisting of the strand aligned to, the read's quality string, and the pattern of mismatches and gaps in the alignment as defined by the `CIGAR` and `MD:Z` SAM fields. For each aligned pair, the template additionally stores the pair's inferred fragment length and a flag indicating which end aligned upstream with respect to the genome. Since templates for large datasets can quickly exhaust memory, Qtip uses reservoir sampling keep a configurable-sized subsample of the templates. The default sample size is 10,000.

In step 3, Qtip uses the input model to simulate tandem reads. To simulate an unpaired tandem read, Qtip randomly draws an unpaired template, with replacement and uniform probability, from those collected in step 2. A new read is constructed from the template by (a) drawing an appropriate-length substring from the reference genome uniformly at random, (b) possibly reverse-complementing it, according to the template strand, (c) mutating the extracted sequence according to the template pattern of mismatches and gaps, and (d) setting the new read's quality string equal to the template's. The simulated read's point of origin is encoded in the read name, allowing later steps to check whether an alignment is correct. The process for simulating a paired tandem read is similar, with fragment length determined by the template. More details are given in Supplementary Material section S.4.

Importantly, some aspects of the input data are hard to mimic. For example, errors made by 454 and Ion Torrent sequencing technologies often manifest as spurious extensions or retractions of homopolymers[18]. Since genome substrings are matched with templates randomly, homopolymer errors in the template will often fail to line up with homopolymers in the substring. Other aspects of the input data are not as difficult to mimic, but happen not to be captured in Qtip's simulation. For example, if a dataset is enriched or depleted for reads drawn from a particular genomic feature (e.g., coding regions), Qtip's simulation, which draws reads uniformly at random from across the genome, will not exhibit that pattern. While we demonstrate Qtip performs well despite these deficiencies, they nonetheless illustrate that it is difficult to construct tandem reads that truly mimic input reads in all ways. We return to this in the Discussion section.

Mapping quality model. Given training records derived from tandem reads aligned in step 4, we train a model in steps 5 and 6 that is later used to predict qs for the input alignments. Qtip trains separate models for each alignment category: *unp*, *conc*, *disc*, and *bad-end*. The particular features used to train a model vary depending on the alignment category and read aligner. We briefly summarize them here, but more details are provided in Supplementary Material section S.3.

These features are included regardless of aligner or alignment category: (a) the alignment score of the best alignment, (b) the difference between the alignment score of the best alignment and that of the second-best alignment if one was found, (c) the length of the aligned read, (d) the sum of the base qualities of the aligned bases, and (e) the sum of the base qualities of the soft-clipped bases. In the case of a concordantly-aligned paired, the inferred fragment length (from the SAM TLEN field) is also included as a feature.

By default Qtip uses an implementation of random forests[19] from the scikit-learn[20] library to model and predict mapping qualities. The random forest consists of many decision trees, each trained on a bootstrap sample of the training (tandem) data. Each tree contributes a vote as to the probability the given alignment is correct, and the final prediction is an average over the votes. This model has many advantages. For example, it is invariant under scaling transformations of features. Also, training is very efficient, which is important since models are tailored to the scenario at hand, and must be rebuilt anew each time Qtip runs. Details on the model are in Supplementary Material section S.5.

3.1 Results.

3.1.1 Experimental conditions.

Simulations were conducted using Mason v0.1.2[21], or a different simulator where indicated. We ran Qtip v1.6.1 in combination with the Bowtie 2 v2.3.0, BWA-MEM v0.7.15, and SNAP v1.0beta.18. Experiments were performed on nodes of the Maryland Advanced Research Computing Center; each node is an Intel Haswell system with two 12-core processors (2.5GHz) and 128GB RAM.

All read aligners were run in their default reporting modes. In other words, all aligners report up to one "best" alignment per read. Reads that fail to align are excluded from the analysis. We used the GRCh38 assembly with some short sequences filtered out (see Supplementary Material section S.6) as our human reference. Qtip ran on Python v2.7.12 and used scikit-learn v0.18. Experiments can be reproduced using software and documentation at `https://github.com/BenLangmead/qtip-experiments/tree/v1.6.1`.

3.1.2 Plots and measures.

Let A be a vector of n alignments $a_0, a_1, ..., a_{n-1}$. Let correct(a_i) equal 1 if a_i is correct, 0 otherwise. Let incorrect(a_i) equal $1 - \text{correct}(a_i)$. An alignment is considered correct if the leftmost base involved in the alignment is within 30 nucleotides of the leftmost base in the simulated substring, with appropriate adjustments for soft clipping. Let $Q = q_0, q_1, ..., q_{n-1}$ be mapping qualities corresponding to $a_0, a_1, ..., a_{n-1}$, as predicted by the read aligner, and let $P = p_0, p_1, ..., p_{n-1}$ be the corresponding correctness probabilities, using the relationship that $q = -10 \cdot \log_{10}(1-p)$. Let Q' and P' be defined similarly, but for the mapping qualities predicted by Qtip.

We define plots (CID, CSED) and measures (RCA, RCE) that characterize how Qtip's predictions (Q') compare to the aligner's (Q). CID and RCA capture how well Q' *ranks* alignments from most to least likely to be correct relative to Q. CID and RCA are invariant under monotonic transformations of P and P'; they are concerned only with how well alignments are ranked, not with probabilities per se. CSED and RCE capture how closely P' matches the the true correctness relative to P; i.e. CSED and RCE are concerned with how well P' and P fit their probabilistic interpretation.

Cumulative Incorrect Difference (CID): Let $\hat{A}$ be A sorted in descending order by Q, and likewise for $\hat{A}'$ and Q'. The *cumulative incorrect vector* C is the vector $c_0, c_1, ..., c_{n-1}$ such that $c_i = \sum_{j=0}^{i} \text{incorrect}(\hat{a}_j)$ [1]. C' is defined similarly for $\hat{A}'$. Let D be the element-wise difference $C' - C$. When $d_i < 0$, Qtip's mapping qualities yield a better segregation of correct from incorrect alignments about the "pivot" i. When $d_i > 0$, the aligner's mapping qualities give the better segregation. A "CID plot" draws a line representing the d_i's (vertical axis) for $i = 0$ to $n-1$ (horizontal axis), and we judge Qtip's efficacy according to the line's tendency to stay below $y = 0$.

Cumulative Squared Error Difference (CSED): Let $\hat{A}$ and $\hat{P}$ be A and P sorted in descending order by P, and likewise for $\hat{A}'$ and $\hat{P}'$. The *cumulative squared error vector* E is the vector $e_0, e_1, ..., e_{n-1}$ such that $e_i = \sum_{j=0}^{i} (\text{correct}(\hat{a}_j) - \hat{p}_j)^2$, with E' defined similarly for $\hat{A}'$ and $\hat{P}'$ [2]. Let S be the element-wise difference $E' - E$. When $s_i < 0$, Qtip's mapping qualities yield lower squared error up to the ith alignment. The "CSED plot" draws a line representing the s_i's (vertical axis) for $i = 0$ to $n-1$ (horizontal axis). Like for the CID plot, we judge Qtip's efficacy according to the line's tendency to stay below $y = 0$.

[1] For a group of alignments sharing the same Q, the penalty is averaged across the group's elements in C and C'. I.e. if , $\hat{a}_k, \hat{a}_{k+1}, ..., \hat{a}_l$ is a maximal stretch of alignments sharing the same quality, then $c_i = c_{i-1} + \sum_{j=k}^{l} \text{incorrect}(\hat{a}_j)/(l-k+1)$ for $k \le i \le l$.

[2] For a group of alignments sharing the same Q, the corresponding elements of E and E' equal the mean squared error of the group. I.e. if , $\hat{a}_k, \hat{a}_{k+1}, ..., \hat{a}_l$ is a maximal stretch of alignments sharing the same quality, then $e_i = e_{i-1} + \sum_{j=k}^{l} (\text{correct}(\hat{a}_j) - \hat{p}_j)^2/(l-k+1)$ for $k \le i \le l$.

Relative change in area under CID (RCA): is defined as $(\sum_{i=0}^{n-1} c'_i - \sum_{i=0}^{n-1} c_i)/\sum_{i=0}^{n-1} c_i$. Negative values indicate a better overall ranking is achieved using Qtip's predictions.

Relative change in sum of squared errors (RCE): is defined as $(\text{SSE}(P') - \text{SSE}(P))/\text{SSE}(P)$, where $\text{SSE}(P) = \sum_{i=0}^{n-1}(correct(a_i) - p_i)^2$. Negative values indicate that Qtip's predictions yield lower total squared error.

The distinction between the rank-based (CID, RCA) and probabilistic (CSED, RCE) metrics relates to how downstream tools, e.g. variant callers, use mapping qualities. Freebayes[22] and the Genome Analysis Toolkit (GATK)[23] ignore an alignment if its mapping quality is below a threshold. In this case, CID and RCA are relevant since as directly evaluate how well various thresholds separate correct from incorrect alignments. Other methods, such as the consensus genotype calling method described in the MAQ study[2], interpret a mapping quality as a probability. Alignments are weighted according to their probability, with no alignments excluded. Here, CSED and RCE are relevant since they directly evaluate how well the probabilities match actual correctness status.

We note that the problem of evaluating and plotting relative quality of two sets of mapping-quality predictions is not specifically addressed in past studies. ROC-like plots are used for the related task of comparing aligners[4,5], where the axes represent false and true positives and a line follows points corresponding to increasingly permissive mapping-quality thresholds. But the two-dimensionality of these plots makes it hard to find comparable points, points on two curves where the threshold allows same number of alignments. A similar problem exists for comparisons examining particular thresholds (≥ 10, ≥ 20, etc); for two sets of predictions the thresholds might allow very different numbers of alignments, impeding interpretation. CID and CSED plots are inspired by Accuracy versus Drop Rate (ADR) plots[24] and are related to ROC-like plots, except: (a) two separate lines are represented more concisely as a single line showing the difference, and (b) at a given horizontal point, thresholds allowing the same number of alignments (having the same "drop rate") are compared.

3.1.3 Simulation experiments.

We conducted simulation experiments to show how Qtip's mapping quality predictions compare to those made by read aligners themselves. We vary several experimental conditions, including (a) read length, (b) aligner parameterization, (c) reference genome, (d) read alignment tool, and (e) read simulator. The simulator encodes the read's true point of origin in the read name, allowing Qtip to check later whether an alignment is correct.

Simulated samples. We used Mason to simulate five Illumina-like samples with unpaired reads of length 50, 100, 150, 250 and 500 respectively. We simulated five paired-end samples with the same lengths, with most fragment lengths between $2L-4L$ nt where L is the read length. We simulated 4 million reads/pairs for each sample. Aligners were configured to consider fragment lengths in the $2L-4L$ range as concordant. Thus, most simulated pairs aligned concordantly (consistent with paired-end constraints) whereas some aligned discordantly. Simulator commands, and implications for fragment lengths, are discussed in Supplementary Material section S.6. Alignment commands are in Supplementary Material section S.7.

Varying read length. We used Qtip together with Bowtie 2 to align and predict mapping qualities for each Mason-simulated sample. We rounded Qtip's predictions to the nearest integer per the SAM/BAM format. For each alignment, we parsed the aligner-predicted and Qtip-predicted mapping qualities, as well as the read's true point of origin with respect to the reference as provided by Mason. We then calculated RCA and RCE (Table 1) and plotted CSED (Figure 2). For clarity, CSED y values were transformed with $y_{plot} = \text{sign}(y_{orig}) \cdot \log_{10}(|y_{orig}|+1)$.

To measure variability of Qtip's predictions, we repeated each experiment 10 times starting from step 2 (building input model) onward, seeding the pseudo-random number

		End-to-end				Local			
		RCA		RCE		RCA		RCE	
	Read length	mean	sd	mean	sd	mean	sd	mean	sd
Unpaired	50	-9.03	0.26	-24.61	0.19	-2.57	0.54	-16.05	0.43
	100	-7.43	1.82	-18.53	2.61	-11.09	1.36	-29.21	1.06
	150	-9.11	1.15	-16.22	0.62	-15.70	2.13	-29.92	0.95
	250	-16.82	2.01	-19.97	0.44	-23.12	1.34	-29.44	0.49
	500	-33.77	0.60	-27.26	0.37	-36.89	3.10	-30.61	1.96
Paired	50	-31.08	0.35	-52.55	0.26	-64.01	0.09	-48.58	0.17
	100	-34.93	1.00	-54.32	0.19	-63.81	0.43	-52.34	0.72
	150	-37.93	1.39	-54.60	0.17	-64.37	0.50	-56.52	0.82
	250	-47.62	0.35	-57.65	0.15	-67.05	0.69	-60.88	1.78
	500	-59.70	1.41	-65.17	0.22	-71.69	0.36	-72.28	1.90

Table 1: Relative change in area under CID (RCA) and relative change in sum of squared error (RCE) when running Qtip and Bowtie 2 on Mason-simulated Illumina-like samples of various lengths. Relative change is expressed as a percent. Each sample consists of 4 million reads/pairs. Samples are either unpaired and paired-end, and Bowtie 2 is run in either end-to-end or local alignment mode as indicated. Results are means and standard deviations over 10 random trials, repeated starting from the input modeling step.

generator differently in each of the 10 trials. The RCA/RCE tables describe all 10 trials whereas, for clarity, the CSED plot describes only the first trial.

Qtip's mapping qualities are overall superior to those predicted by Bowtie 2, as indicated by the negative RCAs and RCEs (Table 1). This is true across all samples tested, and in both end-to-end and local alignment mode. The improvement is larger for samples with longer reads and for paired-end samples. Variability is modest overall but somewhat higher for longer reads. See Discussion for further comments on variability.

However, there are portions of the CSED plots (Figure 2) where the plot rises above $y = 0$, indicating that for some cutoffs the aligner-reported mapping qualities exhibit lower cumulative squared error than those predicted by Qtip. This can be observed for the unpaired experiments, particularly for 50 nt reads. However, Qtip's superior predictions at other q thresholds – especially the low thresholds depicted at the right-hand side of the plots – help bring overall RCE below zero in all cases. For paired-end samples, CSEDs show Qtip's predictions are superior at nearly all q thresholds.

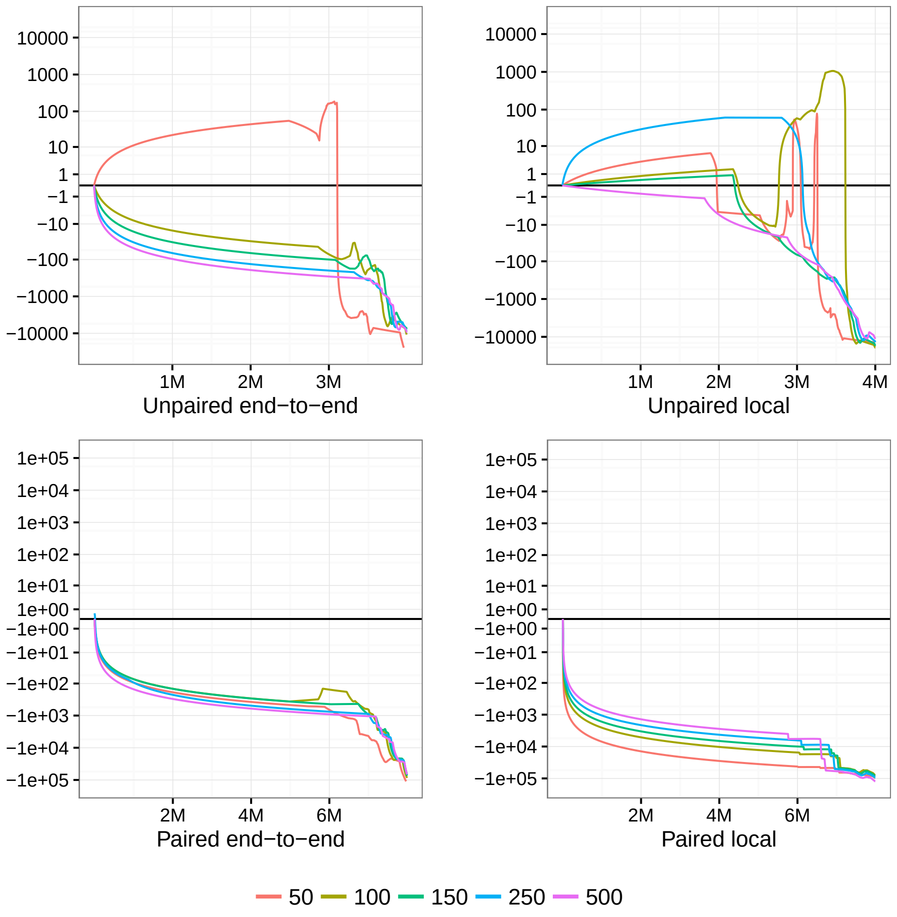

Figure 2: Cumulative squared-error difference (CSED) plot from running Qtip and Bowtie 2 on Mason-simulated Illumina-like samples of various lengths. Each sample consists of 4 million reads or pairs. The horizontal axis indicates cumulative number of reads/ends passing the cutoff, with the left-hand extreme corresponding to a high mapping-quality cutoff and right-hand extreme corresponding to a low cutoff. Results for unpaired samples are on top, paired on bottom. Bowtie 2 is run in its (default) end-to-end alignment mode in the case of the left-hand plots, and in local alignment mode in the case of the right-hand plots.

Varying reference genome and alignment tool. To study how reference genomes of varying length and repetitiveness influence Qtip's predictions, we experimented with three species: Human, Mouse, and *Zea mays*. The human GRCh38 reference is 3.10 Gbp long (2.95 Gbp excluding Ns) with about 50% of the genome annotated as repetitive according to RepeatMasker. The mouse GRCm38 reference is 2.73 Gbp long (2.65 Gbp excluding Ns), with about 44% of the genome annotated as repetitive according to Repeat-

Masker. *Zea mays* AGPv4 is 2.13 Gbps long (2.10 Gbp excluding Ns). Though no official RepeatMasker annotation is available, past studies report that 85% of the genome consists of transposable element sequences[25], making it the most repetitive of the genomes tested. *Zea mays* was chosen because it is substantially more repetitive than the human and mouse references. We used the Mason-simulated 100 nt and 250 nt samples, both unpaired and paired-end. For compatibility with the Mason simulator, some short sequences were filtered out of all three assemblies (see Supplementary Material section S.6).

At the same time, to study how Qtip's predictions compare to those produced by various read aligners, we tested three aligners – Bowtie 2, BWA-MEM and SNAP– with each genome. The changes made to each aligner in order to work with Qtip are detailed in Supplementary Material section S.3. We calculated RCA and RCE for the 10 trials and plotted CSED for the first trial.

Qtip-predicted mapping qualities are superior in nearly all scenarios, as indicated by negative RCAs and RCEs (Table 2). The sole exception is the SNAP 250-nt paired human experiment, which has mean RCE of 4.35% (but mean RCA of -51.32%). Variability of RCAs and RCEs across trials is generally modest, but tool dependent. BWA-MEM 's standard deviations are remarkably small, ranging up to only 0.42. Bowtie 2's range up to 2.61 and SNAP's up to 4.44. See Discussion for further comments on variability.

CSED curves (Figure 3) again show that for some cutoffs, aligner-reported mapping qualities are superior in terms of minimizing cumulative squared error, i.e. where the CSED rises above $y = 0$. Qtip's mapping qualities seem to perform worse for many cutoffs in the BWA-MEM unpaired experiments, especially for *Zea mays*. But in all bases, Qtip's qualities perform better at lower cutoffs. Qtip's mapping qualities perform particularly well for the Bowtie 2 *Zea mays* experiments, and for all the paired-end experiments.

Other simulation experiments. We also conducted simulation experiments varying the sensitivity level of the aligner, described in Supplementary Material section S.8, and vary-

			100 nt				250 nt			
			RCA		RCE		RCA		RCE	
			mean	sd	mean	sd	mean	sd	mean	sd
Unpaired	Human	Bowtie 2	-7.43	1.82	-18.53	2.61	-16.82	2.01	-19.97	0.44
		BWA-MEM	-15.96	0.20	-47.46	0.19	-16.52	0.28	-51.40	0.13
		SNAP	-19.58	0.18	-36.58	0.47	-14.74	0.27	-25.47	0.32
	Mouse	Bowtie 2	-5.60	0.24	-17.19	0.45	-7.05	0.33	-17.73	0.37
		BWA-MEM	-13.69	0.05	-46.50	0.08	-16.80	0.08	-51.40	0.13
		SNAP	-9.02	0.17	-31.07	0.33	-10.61	0.20	-31.78	0.43
	Zea mays	Bowtie 2	-6.63	0.32	-19.56	0.25	-17.09	0.38	-25.89	0.44
		BWA-MEM	-19.23	0.05	-58.74	0.07	-25.17	0.11	-67.15	0.08
		SNAP	-13.02	0.24	-38.48	0.53	-24.01	0.43	-53.80	0.42
Paired	Human	Bowtie 2	-34.93	1.00	-54.32	0.19	-47.93	0.26	-57.77	0.19
		BWA-MEM	-14.16	0.12	-41.29	0.22	-12.32	0.42	-42.73	0.28
		SNAP	-51.36	0.98	-11.16	1.12	-51.32	1.45	4.34	2.29
	Mouse	Bowtie 2	-22.67	0.22	-43.87	0.21	-31.07	0.18	-49.77	0.23
		BWA-MEM	-11.84	0.11	-36.11	0.32	-13.25	0.20	-39.82	0.20
		SNAP	-29.90	0.67	-17.04	0.76	-30.16	0.30	-15.79	0.47
	Zea mays	Bowtie 2	-31.92	0.18	-52.15	0.17	-57.90	0.13	-76.08	0.19
		BWA-MEM	-17.02	0.13	-47.42	0.14	-21.35	0.24	-56.47	0.16
		SNAP	-36.28	0.55	-17.08	0.79	-26.45	4.44	-20.05	0.52

Table 2: Relative change in area under CID (RCA) and relative change in sum of squared error (RCE) for various aligners and reference genomes, expressed as percent change. The experiments used 100 nt or 250 nt reads, and unpaired or paired-end reads, as indicated. Results are means and standard deviations over 10 random trials, repeated starting from the input modeling step.

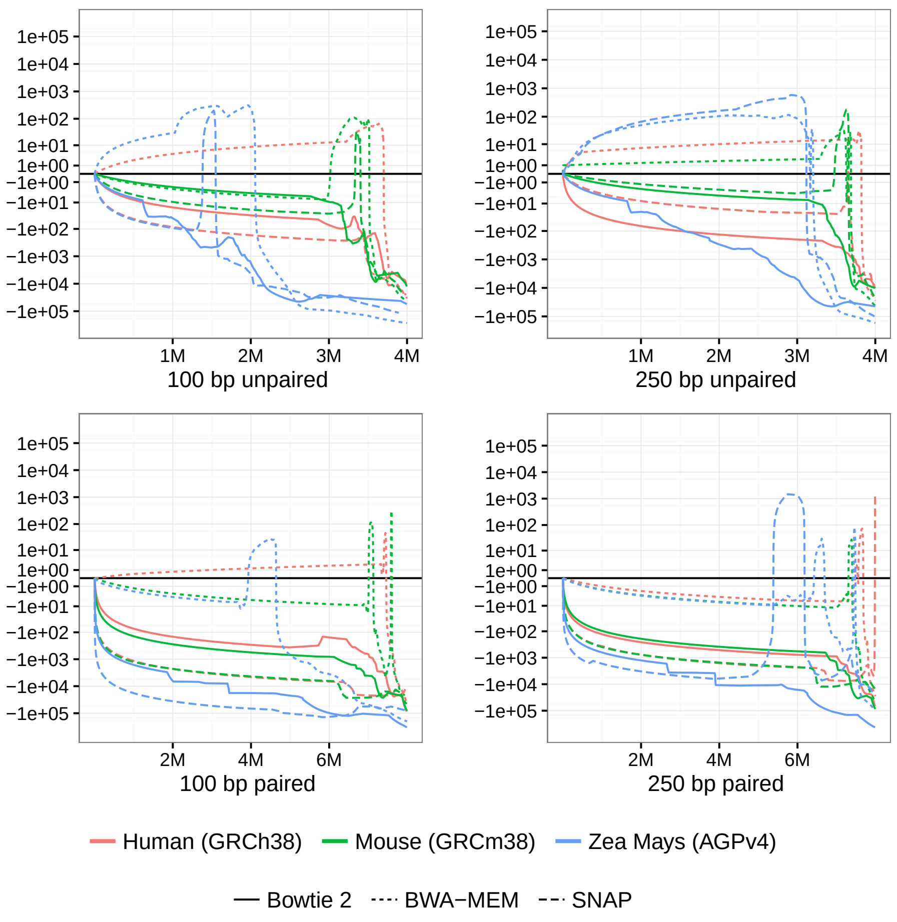

Figure 3: Cumulative squared-error difference (CSED) plot from running Qtip with various reference genomes and read aligners. The input reads are Mason-simulated Illumina-like 100 nt (left) and 250 nt (right) samples, each consisting of 4 million reads/pairs. The horizontal axis indicates cumulative number of reads/ends passing the cutoff, with the left-hand extreme corresponding to a high mapping-quality cutoff and right-hand extreme corresponding to a low cutoff. Results for unpaired samples are shown on top, and paired on bottom.

ing the software tool used to generate the simulated reads, described in Supplementary Material section S.9.

3.1.4 Variant calling

To demonstrate Qtip's effect on downstream results, we evaluated variant calling accuracy with and without Qtip's predictions. We used paired-end human 100 x 100 Illumina HiSeq reads from the Platinum Genomes project[26] (ERR194147) and gold-standard Platinum variants[26] for the sequenced individual (NA12878). The Platinum variants are high-confidence pedigree-validated calls supported by multiple bioinformatics pipelines and sequencing technologies. The analysis is limited to areas of the genome called with high confidence by Platinum Genomes.

We used Freebayes v1.1.0[22] to call single-nucleotide variants (SNVs) once for the alignments with the original mapping qualities and again for the same alignments but with Qtip-predicted mapping qualities. Following standard practice[27], we filtered out variant calls with read depth greater than 4 poisson standard deviations above the mean. We defined a true positive as an SNV call made from ERR194147 data that matched a platinum call, a false positive as a call made from ERR194147 that did not match any platinum call, and a false negative as a platinum call that did not match any ERR194147 call. We calculated F_β for various βs. F_1 ($\beta = 1$) is the typical F1-score, related to the harmonic mean of precision and recall. Setting $\beta > 1$ gives recall more weight than precision and setting $\beta < 1$ gives precision more weight than recall. We tried values of β ranging from 0.25 to 4 to cover a range of precision-recall tradeoffs. Further details on alignment and variant calling are given in Supplementary Material section S.10.

Like other variant callers and downstream tools, Freebayes thresholds on mapping quality (Q) to eliminate some alignments prior to variant calling, eliminating alignments with $Q < 1$ by default. Since we are concerned with overall accuracy of mapping qualities and not with any particular threshold, we re-ran Freebayes with various integer Q thresholds: 0–12, 15, 20 and 30. Freebayes also associates a genotype quality value with each called variant, given in the VCF file's `QUAL` field. We used the `vcfroc` tool from `vcflib` (`https://github.com/vcflib/vcflib`) to evaluate all possible `QUAL` thresholds for all possible Q thresholds, ultimately selecting Q and `QUAL` thresholds maximizing F_β.

Results are presented in Table 3. For all βs examined, Qtip-predicted mapping qualities yielded superior F_β scores. For all but the lowest two values of β tested, this results in the Qtip predictions yielding more true positives and fewer false positives than the original predictions. For the lowest βs, it results in the Qtip predictions yielding much more than 20,000 more true positives at the cost of around 1,000-1,500 more false positives. Notably, these improvements are achieved simply by changing the mapping qualities; the alignments are the same and the variant caller has not been modified or tuned in any way. We also note that Qtip's improved performance is obtained using a smaller range of mapping quality values. Qtip-predicted mapping qualities in this experiment ranged from 0 to 35, whereas Bowtie 2 mapping qualities ranged from 0 to 42.

	Original			Qtip			Δ (Qtip − Orig)		
β	F_β	QUAL	Q	F_β	QUAL	Q	F_β	TP	FP
0.250	0.9925	194	2	0.9925	201	3	+7.0e-05	+24,937	+1,503
0.333	0.9906	150	3	0.9909	167	3	+3.6e-04	+20,620	+1,085
0.500	0.9872	87	3	0.9882	115	4	+9.6e-04	+16,127	-15
0.750	0.9843	10.6	3	0.9856	56.6	4	+1.3e-03	+2,173	-6,130
1.000	0.9835	0.0158	3	0.9847	4.52	4	+1.2e-03	+5,837	-2,498
1.500	0.9846	1.79e-06	3	0.9857	0.000417	5	+1.1e-03	+4,733	-2,399
2.000	0.9860	5.06e-08	3	0.9872	6.44e-06	5	+1.1e-03	+3,956	-4,926
3.000	0.9880	1.16e-09	3	0.9890	1.64e-07	5	+1.0e-03	+2,476	-14,478
4.000	0.9892	1.06e-10	3	0.9901	5.26e-09	5	+8.8e-04	+2,510	-13,918

Table 3: Single-nucleotide variant (SNV) F_β scores for various βs with original mapping qualities and with Qtip-generated qualities. Paired-end reads from ERR194147, a female, were aligned with Bowtie 2 together with Qtip. SNV variants were called with Freebayes for chromosomes 1–22 and X. Variant-quality (QUAL) and mapping-quality (Q) cutoffs yielding greatest F_β score are reported. Platinum variants were used as the true callset. Before calculating F_β, calls outside Platinum Genomes high-confidence regions were excluded. Rightmost columns show differences in F_β, number of true positive SNVs, and number of false positive SNVs.

3.1.5 Efficiency and overhead.

A drawback of the tandem simulation framework is that steps 2–7 (see Figure 1) require memory and computation beyond what is required by the aligner itself. Steps 2 and 7 examine and compute over all the input alignments and steps 3–6 compute over the tandem reads. Qtip's ensemble tree models can be trained quickly (step 6), but substantial computation is required to make a prediction for each input alignment (step 7).

			Time (minutes)			Peak memory (gigabytes)		
			Time	+Qtip	% inc	Memory	+Qtip	% inc
ERR050082	Unpaired	Bowtie 2	24.68	26.33	6.67	3.26	3.49	7.09
		BWA-MEM	22.18	23.75	7.08	7.53	7.79	3.40
		SNAP	12.13	13.75	13.32	29.26	29.51	0.85
	Paired	Bowtie 2	63.22	67.08	6.10	3.27	3.58	9.63
		BWA-MEM	57.93	63.38	9.41	7.87	8.26	4.87
		SNAP	11.28	14.42	27.69	30.21	30.55	1.14
ERR050083	Unpaired	Bowtie 2	24.33	25.97	6.73	3.26	3.49	7.10
		BWA-MEM	24.60	26.08	6.03	7.75	8.01	3.33
		SNAP	12.23	13.73	12.31	29.26	29.51	0.85
	Paired	Bowtie 2	69.67	73.63	5.70	3.27	3.58	9.57
		BWA-MEM	61.58	67.52	9.62	7.92	8.31	4.83
		SNAP	11.95	14.68	22.86	30.21	30.55	1.14

Table 4: Overhead of the Qtip tool. Measured as the increase in running time (left) and peak memory footprint (right) from when the aligner runs by itself ("Time") to when the aligner runs in combination with Qtip ("+Qtip"). "% inc" columns give the percent increase. Times are in minutes and memory footprints are in gigabytes.

We measured Qtip overhead when analyzing public datasets ERR050082 & ERR050083. Specifically, we measured how running time and peak memory footprint grew when Qtip ran alongside the aligner, versus when the aligner ran by itself. Running-time overhead is modest for Bowtie 2 and BWA-MEM , ranging from 5% to 10% (Table 4). For SNAP, running-time overhead is larger, 12% to 14% for unpaired and 23% to 28% for paired-end alignment. Peak memory footprint added by Qtip was 200-400 megabytes in all cases, substantially smaller than the footprint of the aligners themselves which must keep a copy of the reference genome index in memory. In the case of SNAP, peak memory footprint increased by less than 1.15%. For BWA-MEM the increase was always less than 5% and for Bowtie 2 less than 10%.

4 Discussion

We presented the tandem simulation framework and the Qtip software tool implementing the framework. To date, strategies for predicting mapping qualities have either been *ad hoc* or have required the user to prepare training data tailored to the scenario at hand. Qtip runs alongside a read aligner and builds an input model, simulates tandem reads, aligns those using the same aligner and parameters, then uses the trained model to predict mapping qualities. The model and training data are produced automatically and are tailored to the scenario at hand. While Qtip adds overhead to the read alignment process, it reasonable, with time overhead in the 6–28% range and memory overhead in the 1–10% range. This framework, its improved predictions and the evaluation performed here should make authors of downstream software tools more confident that mapping qualities can be treated as the probabilities they claim to be, and to integrate those probabilities into their models rather than simply thresholding.

Qtip's predictions are accurate in various scenarios: various read lengths, unpaired or paired reads, various alignment tools & parameters, etc. We defined novel measures (RCA & RCE) and plots (CID & CSED) for evaluating and plotting mapping-quality predictions. The framework is easy to adapt to other aligners; the aligner must be modified to output feature data in an extra SAM field. Nor is it difficult to add new features to an already-adapted read aligner. Since Qtip's ensemble tree model is scale-agnostic, scaling guesswork it not necessary when adding a feature.

This framework is also applicable to specialized alignment settings, such as spliced RNA-seq alignment. In that case, a nuanced notion of correctness is needed; we care not only where an alignment lands on the reference but also whether it includes the correct splice junctions. There is room for improvement in the area of predicting mapping qualities for spliced alignments. Popular tools use simplistic prediction functions drawing quality values from a small range of possibilities. TopHat and STAR report mapping quality of either 0 or 255 (repetitive versus unique), and TopHat 2 reports 0, 3 or 50 depending on the number of alignments found. Qtip's more principled approach would produce a fuller spectrum of values, potentially with large downstream benefits.

Tandem simulation works to the degree that tandem reads can be sampled "from the same distribution" as input reads. In reality, sampling from the same distribution is not possible. Qtip mimics some aspects of the input data but not others. Homopolymer extensions/retractions are not captured, for example, creating a fundamental difference between tandem and input reads. A tradeoff exists here: Qtip's simple model mimics some aspects of the input without sacrificing efficiency, whereas a more complex and less efficient model could improve accuracy by mimicking more aspects. A task for future work is to measure various points in this tradeoff space, and to define measures for characterizing how and to what extent a set of tandem reads differs from the input reads.

A related question for future work is whether Qtip's sampling scheme can be improved. An importance sampling scheme, for example, might favor tandem reads originating from more difficult-to-predict portions of the sample space. These "important" reads might be reads originating from repetitive elements, or with certain patterns of mismatches and gaps. The scheme, together with appropriate weighting during model training, could achieve comparable accuracy while reducing the number of tandem reads required, thus reducing overhead. It could also reduce the greater prediction variability we see in experiments involving longer reads and more repetitive genomes.

Author Contributions BL designed the method and wrote the software and manuscript.

Acknowledgments We thank Rafael Irizarry for helpful design discussions in the early stages of the project. We thank Abhinav Nellore and Michael Schatz for helpful comments on manuscript drafts. BL was supported by NSF grant IIS-1349906 to BL, NIH/NIGMS grant R01GM118568 to BL, and NIH/NHGRI grant R01HG005220 to Rafael Irizarry.

Literature cited

1. Reinert, K., Langmead, B., Weese, D. & Evers, D. J. Alignment of Next-Generation Sequencing Reads. *Annual review of genomics and human genetics* **16,** 133–151 (2015).
2. Li, H., Ruan, J. & Durbin, R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome research* **18,** 1851–1858 (2008).
3. Li, H. *et al.* The sequence alignment/map format and SAMtools. *Bioinformatics* **25,** 2078–2079 (2009).
4. Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nature methods* **9,** 357–359 (2012).
5. Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv:1303.3997* (2013).
6. Zaharia, M. *et al.* Faster and more accurate sequence alignment with SNAP. *arXiv preprint arXiv:1111.5572* (2011).
7. Pickrell, J. K., Gilad, Y. & Pritchard, J. K. Comment on Widespread RNA and DNA Sequence Differences in the Human Transcriptome. *Science* **335,** 1302 (2012).
8. Treangen, T. J. & Salzberg, S. L. Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nature Reviews Genetics* **13,** 36–46 (2011).
9. Taub, M., Lipson, D., Speed, T. P., *et al.* Methods for allocating ambiguous short-reads. *Communications in Information & Systems* **10,** 69–82 (2010).
10. Treangen, T. J. & Salzberg, S. L. Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nature Reviews Genetics* **13,** 36–46 (2012).
11. Karlin, S. & Altschul, S. F. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences* **87,** 2264–2268 (1990).

12. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics* **25,** 1754–1760 (2009).

13. Li, H. & Durbin, R. Fast and accurate long-read alignment with Burrows–Wheeler transform. *Bioinformatics* **26,** 589–595 (2010).

14. Giese, S. H., Zickmann, F. & Renard, B. Y. Specificity control for read alignments using an artificial reference genome-guided false discovery rate. *Bioinformatics* **30,** 9–16 (2014).

15. Ruffalo, M., Koyutürk, M., Ray, S. & LaFramboise, T. Accurate estimation of short read mapping quality for next-generation genome sequencing. *Bioinformatics* **28,** i349–i355 (2012).

16. Lee, W.-P. *et al.* MOSAIK: A hash-based algorithm for accurate next-generation sequencing short-read mapping. *PloS one* **9,** e90581 (2014).

17. Hodgkinson, A., Grenier, J.-C., Gbeha, E. & Awadalla, P. A haplotype-based normalization technique for the analysis and detection of allele specific expression. *BMC bioinformatics* **17,** 364 (2016).

18. Loman, N. J. *et al.* Performance comparison of benchtop high-throughput sequencing platforms. *Nature biotechnology* **30,** 434–439 (2012).

19. Breiman, L. Random forests. *Machine learning* **45,** 5–32 (2001).

20. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).

21. Holtgrewe, M. Mason–a read simulator for second generation sequencing data. *Technical Report FU Berlin* (2010).

22. Garrison, E. & Marth, G. Haplotype-based variant detection from short-read sequencing. *arXiv preprint arXiv:1207.3907* (2012).

23. McKenna, A. *et al.* The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome research* **20,** 1297–1303 (2010).

24. Lin, S. *et al.* Validation and extension of an empirical Bayes method for SNP calling on Affymetrix microarrays. *Genome biology* **9,** R63 (2008).

25. Schnable, P. S. *et al.* The B73 maize genome: complexity, diversity, and dynamics. *Science* **326,** 1112–1115 (Nov. 2009).

26. Eberle, M. A. *et al.* A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree. *Genome Research* **27,** 157–164 (2017).

27. Li, H. Toward better understanding of artifacts in variant calling from high-coverage samples. *Bioinformatics* **30,** 2843–2851 (Oct. 2014).

Supplement for: A tandem simulation framework for predicting mapping quality

Ben Langmead

S.1 Alignment heuristics in Bowtie 2

We elaborate our discussion of alignment heuristics with a few observations about how Bowtie 2 searches for alignments.

1. Only alignments at or above an *alignment score threshold* S_{min} are considered. These are called *valid*.
2. *Local similarity filters* require that one or more substrings of the read to align with few or no differences. For instance, a filter might require that alignments contain a length-k stretch of nucleotides that match the reference exactly. For long enough k, this disqualifies otherwise valid alignments.
3. *Repeat filters* avoid the extensive and mostly unproductive work performed when many candidates pass the local similarity filter. Say we take a 20 nt read then use a fast index lookup to find all reference locations where the first 10 nt of the read occur exactly. Say there are 100,000 such places. Instead of examining each candidate, Bowtie 2 will examine a subset and ignore the rest.
4. *Early stopping* halts alignment work once it seems to have become unproductive. For instance, the aligner might stop searching once a certain number of consecutive index queries have failed to yield a valid alignment.

S.2 Measuring prevalence of alignment errors

We defined three categories of alignment error:

1. The read is reported to have originated from a locus in the reference genome, but actually originates from a DNA sequence not included in the reference
2. No alignment to the reference is found, but the read actually originates from some locus in the reference
3. An alignment to locus L_r in the reference is reported, but the read actually originates from a different locus in the reference, L_t

To measure prevalence of these errors in a realistic setting, we used Mason[1] to simulate 100,000 100-bp unpaired Illumina-like reads derived mostly from the human genome (GRCh38), but with 7% of reads derived from contaminating genomes: 1% each from mouse (GRCm38), the bacteria *Propionibacterium acnes*, *Mycolpasma Hominis*, *Mycolpasma fermentans*, *Acholeplasma laidlawii*, and *Mycolpasma hyorhinis*, and the fungus *Malassezia globosa*. We arrived at this array of contaminants after surveying studies on sequencing data contamination[2–4]. We simulated both unpaired and paired-end samples, and we simulated samples with 100-nt and 250-nt reads. We ran Mason as described in the Supplementary Material section S.6 section. We then ran Bowtie 2 v2.3.0 as described in the Supplementary Material section S.7 section to align the reads. We also repeated these experiments with the GRCm38 genome as the target (93% of reads) and GRCh38 as the contaminant (1% of reads) genome. For each aligned sample, we counted errors of the three categories and tabulated them as a percentage of the simulated reads and as a percentage of the errors (Supplementary Table S1).

As expected, longer reads yield fewer errors than shorter reads and paired-end reads yield fewer errors than unpaired reads. A large majority of the errors, ranging from 96.3-99.7%, are of category 3. The next largest, ranging from 0.2-3.9% of errors, are of category 2. These numbers could be reduced using more sensitive alignment parameters. Category

			Category 1		Category 2		Category 3	
			Reads	Errors	Reads	Errors	Reads	Errors
Human	100	Unpaired	0.006	0.144	0.159	3.567	4.287	96.289
		Paired	0.004	0.136	0.079	2.657	2.888	97.206
	250	Unpaired	0.002	0.100	0.065	2.713	2.325	97.187
		Paired	0.002	0.094	0.073	3.902	1.794	96.004
Mouse	100	Unpaired	0.005	0.064	0.065	0.822	7.809	99.114
		Paired	0.003	0.056	0.029	0.502	5.760	99.442
	250	Unpaired	0.003	0.053	0.012	0.238	4.895	99.709
		Paired	0.002	0.051	0.013	0.340	3.839	99.609

Table S1: Prevalence of errors of different categories when analyzing Mason-simulated samples using Bowtie 2. Columns labeled Reads give the percent of all the reads, both correctly and incorrectly aligned, in the labeled error category. Columns labeled Errors give the percent of the errors in the labeled category.

1 errors are rare, ranging from 0.05-0.14% of errors. The prevalence of category 3 errors underscores the need for better methods for predicting mapping quality, such as those presented in this study. The rarity of category 1 errors justifies our decision in this study to ignore these when predicting mapping qualities.

S.3 Modifications to read aligners

Three alignment tools – Bowtie 2, BWA-MEM , and SNAP– were modified to output feature data needed by the tandem simulation framework. To make the modifications, we studied the source code of each tool and identified data that could be relevant to the mapping quality prediction task. We started by studying how the tool itself predicts mapping quality, since the inputs to those routines are candidate features. In all three cases, we added more features beyond these, usually with the goal of capturing more information about the repetitiveness of the read and how many of its "seeds" either failed to align or aligned repetitively.

Here we examine each tool, discussing (a) how the tool predicts mapping quality, (b) which features we re-used from that process, and (c) which additional features we added.

Bowtie 2 Bowtie 2's mapping quality calculation uses a lookup table, with the lookup keys being functions of (a) the best alignment's score, (b) the second-best alignment's score, (c) whether end-to-end or local alignment were used, (d) whether the read aligned concordantly in a pair. We preserve (a) and (b) in the features we elect to report from Bowtie 2. We do not need to report either (c) or (d) because these follow naturally from how Qtip works. In the case of (c), the training data is tailored to match the alignment parameters used to align the input data; the training data will always match the input data in terms of the alignment mode used. In the case of (d), Qtip trains separate models for separate alignment categories (*conc*, *disc*, *unp*, *bad-end*), and always uses the appropriate model for prediction.

Following are the unpaired features we output in the modified version of Bowtie 2:

- Best alignment score
- Difference between the best and second-best alignment score, or NA if no second-best alignment was found

- Difference between the best and second-best edit distance, or NA if no second-best alignment was found
- Number of seeds for which Bowtie 2 sought an alignment (*attempted* seeds) divided by the number of nucleotides in the read
- Fraction of attempted seeds that aligned uniquely (to exactly one place in the reference)
- Same as previous but limited to seeds that aligned in the same orientation as the overall reported alignment
- Fraction of attempted seeds that aligned repetitively (to more than one place in the reference)
- Same as previous but limited to seeds that aligned in the same orientation as the overall reported alignment
- Mean number of alignments found for any attempted seed
- Same as previous but limited to seeds that aligned in the same orientation as the overall reported alignment

Following are the paired-end features we output in the modified version of Bowtie 2:

- Sum of the alignment scores for both ends of the best paired-end alignment
- Difference between the best and second-best paired-end alignment score (i.e. the sum of the scores of the two ends), or NA if no second-best alignment was found
- Difference between the best and second-best paired-end edit distance (i.e. the sum of the edit distances of the two ends), or NA if no second-best alignment was found

BWA-MEM. BWA-MEM 's mapping quality calculation depends on, among other factors, (a) the best alignment's score, (b) the second-best alignment's score, (c) the number of alignments "tied" for second-best, (d) the number of aligned bases (i.e. excluding any soft-clipped bases), (e) the percent identity, summarized across the aligned bases, (f) the *seed coverage*, which roughly equals the mean number of times a read base is covered by an aligned seed. With the exception of percent identity, we use these features and others in Qtip.

Following are the unpaired features we output in the modified version of BWA-MEM :

- Best alignment score
- Edit distance of the best alignment
- Difference between the best and second-best alignment score, or NA if no second-best alignment was found
- Number of alignments "tied" for second-best
- Fraction of seeds that aligned repetitively
- Seed coverage

Following are the paired-end features we output in the modified version of BWA-MEM :

- Difference between the best and second-best paired-end alignment score (i.e. the sum of the scores of the two ends), or NA if no second-best alignment was found
- Sum of alignment scores of the two ends of the best paired-end alignment
- Number of paired-end alignments "tied" for second-best
- Fraction of seeds from either end of the pair that aligned repetitively
- Seed coverage over both ends of the pair

SNAP. For a given read, SNAP assigns a probability to each alignment found for that read. SNAP's procedure for calculating the best alignment's mapping quality mostly consists of dividing the best alignment's probability by the sum of the probabilities of all the other alignments found. The probability of a given alignment is function of the length of the read, the read's quality values, and a collection of parameters giving the prior probability of a variant, gap opening, gap extension, etc. We use these probabilities and other features in Qtip.

Following are the unpaired features we output in the modified version of SNAP:

- Edit distance of the best alignment
- Difference between best and second-best alignment's edit distance, or NA if no second-best alignment was found
- Probability of the best alignment divided by the total probability of all the alignments found
- Number of seeds ignored because they matched in the genome too many times
- Minimum number of times a matching seed matched in the genome
- Number of seeds that failed to match the genome
- Mean number of matches per seed

Following are the paired-end features we output in the modified version of SNAP:

- Difference between the best and second-best paired-end edit distance (i.e. the sum of the scores of the two ends), or NA if no second-best alignment was found
- Total number of seeds in the pair ignored because they matched in the genome too many times
- Minimum number of times a matching seed matched in the genome across both ends of the pair
- Number of seeds that failed to match the genome across both ends of the pair
- Mean number of matches per seed across all seeds in the pair

Qtip parses feature data from the extra `ZT:Z` SAM field. So once we identified a feature, we enable Qtip to use that feature simply by printing it as an additional value in the `ZT:Z` field. This field contains comma-separated numeric data. The following is an example of an alignment produced by BWA-MEM with the `ZT:Z` field. The `ZT:Z` field is not printed for unaligned reads, and Qtip will ignore these.

Qtip also ignores the small number of chimeric alignments reported by BWA-MEM. Proper handling of chimeric alignments is future work.

In addition to the features discussed above, which are reported by the read aligners in the `ZT:Z` field, Qtip adds a few additional features, specifically:

- Length of the read or end
- Sum of the base qualities of the aligned bases in the read or end
- Sum of the base qualities of the soft-clipped bases in the read or end

Finally, for a given end of a paired-end alignment, whether concordant or discordant, feature data for the opposite end is included in the record for the given end. This is in addition to the feature data described above. I.e., in the record for a given end of a paired-end read, you will find aligner-specific unpaired feature data, aligner-specific paired-end feature data, generic features, and both aligner-specific and generic features *for the opposite end*.

When the training and test data are compiled within Qtip, columns in which all elements are equal are eliminated. E.g., if the input consists of reads all of the same length, then the read length will be identical for all records and the corresponding column will be dropped from the training and test matrices. Similarly, if the input consists of reads aligned in end-to-end alignment mode (Bowtie 2's default), then the sum of the base qualities of the soft-clipped bases will always be 0, and the column corresponding to that feature will again be dropped.

Availability of aligner modifications. Bowtie 2 v2.3.0 has the appropriate feature-printing code built in; earlier versions of Bowtie 2 do not have this code. The `--mapq-extra` parameter instructs Bowtie 2 v2.3.0 to print feature data in the `ZT:Z` extra SAM field. To run this version of Bowtie 2 in combination with Qtip, use Qtip's `--bt2-exe` parameter.

Qtip comes with a source-code patch that causes BWA-MEM to output feature data for use with Qtip. To create a modified version of BWA-MEM , download the BWA-MEM v0.7.15 source archive from `https://sourceforge.net/projects/bio-bwa/files/`. Once extracted, change to the BWA-MEM v0.7.15 source directory and run:

```
patch -p1 < /path/to/qtip/software/bwa/bwa_conc_flags_0.7.15.patch
```

Then build the binaries as usual. To run this version of BWA-MEM in combination with Qtip, use Qtip's `--bwa-exe` parameter. The modified BWA-MEM will print the feature data in the `ZT:Z` extra SAM field.

Qtip comes with a source-code patch that causes SNAP to output feature data for use with Qtip. To create a modified version of SNAP, download the SNAP v1.0beta.18 source archive from `http://snap.cs.berkeley.edu`. Once extracted, change to the SNAP v1.0beta.18 source directory and run:

```
patch -p1 < /path/to/qtip/software/snap/snap_features_patch.diff
```

Then build the binaries as usual. To run this version of SNAP in combination with Qtip, use Qtip's `--snap-exe` parameter. The modified SNAP will print the feature data in the `ZT:Z` extra SAM field.

S.4 Input modeling and tandem simulation

Scanning the input. The construct the input model, Qtip using reservoir sampling to obtain an appropriate-sized subset of the input alignments. It does this over the course of a single scan through the input alignments. For paired-end alignments, Qtip's scan matches up the two ends so that they can be considered together here. Rads that fail to align are ignored, as are non-primary alignments,. During the scan, Qtip distinguishes between the four different categories of alignment: *conc*, *disc*, *unp* and *bad-end*. A separate reservoir sampler is used for each category. The maximum number of alignments sampled in a category can be configured with Qtip's `--input-model-size` parameter (default: 30,000).

Templates. Every category of template contains at least the following fields:

- Score
- Read length
- Strand
- Quality string
- Edit transcript

Alignment score is copied from the `AS:i` SAM field. Read length is easily calculated from the SAM/BAM `SEQ` field. Strand is inferred from the SAM `FLAG` field. Quality string is copied from the SAM texttttQUAL field.

Edit transcript is inferred from a different combination of SAM fields depending on the aligner used. For Bowtie 2 and BWA-MEM , it is inferred from the `CIGAR` and `MD:Z` fields. SNAP does not report the `MD:Z` field, and the CIGAR field alone is not always sufficient to determine where mismatches are located. However, SNAP can optionally print a CIGAR string that does contain sufficient information to determine where mismatches are located (the `-=` parameter), so when running SNAP from Qtip, that parameter is always specified.

When that parameter is specified, the edit transcript can be inferred from the CIGAR field alone.

The 5 fields listed in the previous subsection are sufficient for representing an unpaired read template. A bad-end template includes two additional fields:

- Whether aligned end is end 1 or 2
- Length of the (unaligned) opposite end

These are inferred from the SAM `FLAG` and `SEQ` fields for the opposite end.

A *conc* template contains all of the fields listed above plus these additional fields:

- Score of the opposite end
- Sum of scores of the two ends
- Strand of opposite end
- Quality string of opposite end
- Edit transcript of opposite end
- Whether end 1 aligns to the left of end 2 with respect to the reference
- Fragment length

Most of these additional fields are inferred straightforwardly from the SAM records for the two ends as described previously. Which end aligns upstream is determined by examining the SAM `POS` field for the two records. Fragment length is inferred from the SAM `TLEN` field.

A *disc* template is identical to a *conc* template. Note, though, that the fragment length field may not be interpretable for a discordantly aligned pair, since the two discordantly-aligned ends can align very far apart or even to different chromosomes.

Handling soft clipping. Input alignments with soft-clipped bases (`S` CIGAR operation) pose a problem for our templates: because the soft-clipped bases failed to align, we do not know how to represent them in the edit transcript. Simply excluding those bases from the edit transcript – effectively trimming them from the tandem read – is not desirable. We would like the tandem reads to mimic the input reads in all important ways, including in the prevalence of soft-clipping.

One solution is to, for each instance of soft clipping, extract the corresponding stretch of reference bases and perform post-facto alignment of the soft-clipped bases. The alignment would likely be poor (hence the soft clipping), but would yield informative values for the edit transcript. This is slow in practice, however, since it requires that we extract a arbitrary stretch of the reference genome – requiring disk head movement in many cases – for each instance of soft clipping.

Instead, Qtip represents soft-clipped bases with special `S` characters in the edit transcript. When simulating the read, Qtip simply inserts a random base into each soft-clipped position. Advantages of this approach are: (a) it is fast and simple, requiring no additional alignment, and (b) random bases are a reasonable proxy for soft-clipped bases since' they are unlikely to align well to the reference, and so will likely be clipped. The disadvantage is that there are no guarantees about how successfully the random bases will act as a proxy for the soft-clipped bases. For example, the random bases could, by coincidence, match the reference genome well enough that they align without any soft clipping, or with less soft clipping than was needed for the corresponding input read. This causes the tandem reads to depart somewhat from the character of the input reads, though Qtip's good performance on local read aligners suggests that the effect is likely minor.

Simulating from templates. Qtip uses the input models to create the tandem read set in the following way. The user specifies the location of the reference genome in FASTA format to Qtip. Qtip iterates through overlapping windows (*chunks*) of the reference genome. Each chunk consists of $B + O$ contiguous bases, where the base length B is 100,000 by default. We discuss later how the overlap length O is determined. The O bases at the end of the $B - O$ sized chunk will appear again at the beginning of the following chunk, except in cases where, for example, we reach the end of a chromosome.

Overlaps between chunks are to ensure all possible read starting positions are equally probable. When a read is simulated, the read's leftmost base must fall within the first B bases of the current chunk. The read's leftmost base may not fall within the final O bases (though those bases will appear again at the beginning of the following chunk).

For each chunk and each alignment category, Qtip calculates the number of reads to simulate according to a binomial draw with $n = B$, the number of possible starting positions for the fragment, and $p = \frac{G}{n}$ where G is the genome length and n is the number of reads of that category we aim to simulate in total. G is estimated from the FASTA file sizes, and n is set according to the `--sim-*` arguments passed to Qtip.

Each unpaired read is then simulated by (1) picking a template of the appropriate type uniformly at random from among those sampled, (2) picking a leftmost read position from among the first B positions of the chunk uniformly at random, (3) extracting an appropriate-length substring of the reference genome starting at the chosen position, (4) mutating the extracted reference substring according to the strand and edit transcript information in the template, (5) setting the read name equal to a special string that encodes the read's true point of origin, (6) setting the read quality string equal to the quality string in the template, and finally (7) printing the read to a FASTQ file in preparation for the tandem alignment step.

A bad-end read is simulated similarly except that a dummy opposite end is synthesized by generating a random nucleotide sequence. The dummy opposite end is virtually guaranteed not to align, mimicking the situation for the input pair.

Paired-end reads are simulated similarly except that when extracting the substring from the reference genome, a substring corresponding to the entire fragment is extracted. In the case of a *conc* pair, fragment length is dictated by the length given in the template. In the case of a *disc* pair, fragment length is set to a pre-determined value larger than the largest concordant fragment length simulated. This is configurable via Qtip's `--max-allowed-fraglen` parameter. Once a fragment substring is extracted, read sequences are extracted from either end of the fragment, choosing the upstream end and the orientations according to the template.

In order to simulate a long fragment, the entire fragment must be present in some chunk returned by the FASTA parser. To guarantee this will be the case, Qtip sets the overlap length O equal to the maximum fragment or read length in any of the input models.

Aligning tandem reads. After tandem reads are simulated, they are aligned (step 4 in Figure 1) using the same read aligner and the same alignment parameters as were used to align the input reads in step 1. However, circumstances can cause the tandem alignments to depart from the template.

First, a tandem read may simply fail to align, decreasing the number of Qtip's training examples by one. This is relatively rare, since the templates themselves correspond to reads that aligned successfully, and are therefore unlikely to run seriously afoul of the aligner heuristics. However, the tandem read might be extracted from a qualitatively different genomic region than the original read. For example, the tandem read might be drawn from a repeat, whereas the original read was not. This increases the chance that, e.g., the aligner might stop early before finding a valid alignment for the tandem read.

Second, the tandem read might align successfully, but with a different alignment score and/or edit transcript from those in the template. E.g., if the tandem read is drawn from a repetitive portion of the genome, the edits induced by the transcript might cause the read to align with higher alignment score elsewhere. Alternately, the aligner might fail to align the read to its true point of origin and instead align it such that its score is lower than the template score. We so no strong evidence that Qtip's predictions are systematically biased in either direction, either producing tandem alignments with overall higher or overall lower scores than the input alignments.

Discussion. We described a method for deriving input models from aligned reads. We also described a method for using the input models, together with the reference genome, to simulate a collection of tandem reads that mimic the input reads in key ways. The method has several advantages: (a) it is simple, not requiring an external read simulator, (b) input models are built during a single pass over the input alignments, (c) reads are simulated during a single pass over the reference genome, (d) since the input alignments are scanned one- or two-at-a-time, and since only a single chunk of the reference genome considered at a time during simulation, the added memory footprint is low.

It should also be noted that, because read sequences are drawn randomly from across the genome and are matched with templates randomly, the tandem reads taken as a whole are not consistent with any particular genome sequence. That is, the reads will not generally agree with each on, e.g., where any SNPs or other variants are located with respect to the reference genome. But since the read aligners themselves consider each read independently and one at a time, this should not affect the performance of the mapping quality model trained in step 6.

S.5 Mapping-quality prediction model

Random forest model Qtip uses the scikit-learn Python module[5] to build its mapping quality prediction model. Specifically Qtip uses scikit-learn's `RandomForestRegressor`, a model consisting of an ensemble of decision trees. Each tree is trained on a bootstrap sample of the training data, and each tree contributes a "vote" as to the probability the given alignment is correct. The final prediction is an average of all trees' votes.

Random forests have many advantages in this setting. First, decision tree training is not sensitive to how features are scaled. When adding a new feature, an aligner author can simply report the data as-is, without first having to consider whether log scaling, or any other monotone scaling, is needed to bring the feature into agreement with other features. This is a major advantage over scale-sensitive models like support vector machines or neural networks.

Second, random forests are efficient. They have few hyperparameters, making the training process simpler and faster. E.g., the SNAP aligner takes over 2 hours to align the paired-end ERR050083 dataset, whereas the resulting random forests take about 2 minutes to train. This is a key advantage over, e.g., neural networks. Having said that, the prediction step – step 7 from Figure 1 – is the single biggest contributor to Qtip's computational overhead. We plan to explore whether using a different random forest implementation could increase prediction throughput.

Finally, after a `RandomForestRegressor` model is fit, it optionally reports feature importances. A feature's importance is essentially the degree to which splitting on the feature helps to separate positive from negative training examples.

Hyperparameters Parameters governing the size, shape and splitting behavior of the random forest, often called hyperparameters, have to be chosen at model fitting time. For this model, the relevant parameters are:

- Number of decision trees in the forest. Set to 30 by default in Qtip, but adjustable using the `--num-trees` parameter.
- Maximum number of leaf nodes per decision tree. Set to 35 by default in Qtip, but adjustable using the `--max-leaf-nodes` parameter.
- Maximum number of features to split on at each internal decision tree node. This is expressed as the fraction of the total number of features, and it's allowed to vary from 0.1 to 0.45 by default. It is adjustable using the `--max-features` parameter.

As mentioned, the `--max-features` hyperparameter is not fixed. In fact, all three hyperparameters can be configured to take values in a range; e.g., specifying the parameter `--num-trees 30,35,40,45,50` will allow `--num-trees` to take on any of the specified values. For each hyperparameter allowed to vary in this way, a particular value is chosen at model-training time via a hyperparameter fitting procedure that performs hill-climbing with a configurable numeric tolerance (`--optimization-tolerance`). The hyperparameters selected are those that yield the best out-of-bag score after training. The out-of-bag score is calculated and by the `RandomForestRegressor` model. As an alternative to the out-of-bag score, cross validation can be used instead (via Qtip's `--no-oob` parameter).

Alternate models Qtip can optionally be configured to use the `ExtraTreesRegressor` class, which implements a variant on random forests called extremely randomized trees[6]. The hyperparameters are defined, configured, and fit in exactly the same manner as for the `RandomForestRegressor`.

Forming the training and test matrices The test matrix is a matrix where each row is an input alignment and each column is a feature. Elements contain feature data. In the case of a paired-end alignment, the two ends appear as separate rows in the matrix. The training matrix is similar, except that the rows are tandem alignments rather than input alignments. For each alignment category, Qtip builds a training matrix and uses it to train the model, as shown in step 6 of Figure 1. In step 7, Qtip uses the model to predict mapping quality for the input alignments.

As mentioned, the columns of the matrices are features and the rows are alignments. Qtip might simplify the matrix in certain ways. For example, if all elements in a column are identical in the training matrix, that column is removed from both the training and test matrices. Also, if two columns of the training matrix are identical, the rightmost of the two columns is removed from both the training and test matrices.

To maintain a low memory footprint, matrices are handled such that only a single fixed-sized block of consecutive rows needs to be present in memory at a time. For example, when Qtip parses test records derived from the input SAM file, the records are read in blocks, with only one block residing in memory at a time. By default, a single block contains at most 250K records. This is configurable via Qtip's `--max-rows` option. Qtip then uses the model to predict mapping qualities for the block and writes the corresponding mapping qualities to disk before deallocating the block and moving on.

Missing values As discussed in Supplementary Material section S.3, some feature data can be "missing." For example, an important feature used in Qtip is the difference between the score of the best and second-best alignments found by the aligner. But if the aligner fails to find a second-best alignment, this feature will take the value `NA`, meaning missing or "not available." Qtip deals with `NA` values by automatically setting all of the `NA` values in the column equal to $x+1$ where x is the maximum non-`NA` value in the column. The fact that `NA`s are set to be slightly larger than the largest non-`NA` value often suggests particular way of ordering the data values for a feature. For example, in the case of the difference between the best and second-best score, it makes sense for larger differences to be represented by larger numbers, so that when an `NA` is replaced by $x+1$, this is interpreted by the model, sensibly, as a large difference.

S.6 Simulations

Mason. To generate unpaired samples we ran Mason v0.1.2 with these parameters:

```
mason illumina -hn 2 -i -s (seed) -sq -n (read_len) -N 4000000 \
      -o (output_fastq) (genome_fasta)
```

`illumina` causes Mason to simulate Illumina-like reads. `-hn 2` causes Mason to simulate 2 haplotypes. `-i` causes Mason to include information about the read's true point of origin in the read name, which is useful later for determining which alignments are correct. `-s` sets the pseudo-random seed, which we set arbitrarily for our experiments. `-sq` causes Mason to simulate base qualities. `-n` specifies the read length. `-N` specifies the number of reads to simulate. `-o` specifies the output file.

To generate paired-end samples we ran Mason with these parameters:

```
mason illumina -hn 2 -i -s (seed) -sq -mp -rn 2 -ll (mean) -le (stddev) \
        -n (read_len) -N 4000000 -o (output_fastq) (genome_fasta)
```

`illumina` causes Mason to simulate Illumina-like reads. `-hn 2` causes Mason to simulate 2 haplotypes. `-i` causes Mason to include information about the read's true point of origin in the read name, which is useful later for determining which alignments are correct. `-s` sets the pseudo-random seed, which we set arbitrarily for our experiments. `-sq` causes Mason to simulate base qualities. `-mp` causes Mason to simulate paired-end reads. `-rn 2` enables "one-based slash suffix" read naming. `-ll` and `-le` are discussed below. `-n` specifies the read length. `-N` specifies the number of reads to simulate. `-o` specifies the output file.

For the paired-end samples, we ran Mason with parameters `-ll 3L -le L`. These establish a mean ($3L$) and standard deviation (L) for the fragment length distribution. This means that some simulated fragments will fall outside the range $2L - 4L$, despite the fact that this is the range of fragment lengths that the aligners will consider "concordant"

(Supplementary Material section S.7). This is desirable; for our paired-end samples, we would like the input to include reads that yield all three alignment categories: *conc*, *disc* and *bad-end*.

Mason v0.1.2 cannot handle very short reference sequences. To work around this, we removed all reference sequences shorter than 10,000 bases at the outset. We filtered out 90 short sequences from the GRCh38 assembly, totaling 229,926 nt. We filtered out 1 1,976-nt-long sequence from the GRCm38 assembly. And we filtered out 5 short sequences from the AGPv4 assembly, totaling 37,762 nt.

When simulating paired-end reads, Mason v0.1.2 would sometimes generate pairs where the end that one would expect to find upstream would actually be found downstream. For example, in a typical Illumina paired-end protocols, the two ends of a fragment originating from the forward reference strand would align with (a) end 1 upstream of end 2, (b) end 1 aligning in the forward orientation, and (c) end 2 aligning in the reverse orientation. But for around 0.10–0.15% of the pairs in a Mason-simulated sample, we found that the upstream/downstream relationship between the ends was reversed. These were filtered out prior to our experiments.

wgsim. To generate unpaired samples we ran wgsim with these parameters:

```
wgsim -S (seed) -1 (read_len) -2 (read_len) -N 4000000 \
        (genome_fasta) (output_fastq)
```

`-S` sets the pseudo-random seed, which we set arbitrarily for our experiments. `-1` and `-2` specify the read lengths for either end of a paired-end read. `-N` specifies the number of reads to simulate. The last two parameters specify the reference genome FASTA file and the output FASTQ file.

To generate paired-end samples we ran wgsim with these parameters:

```
wgsim -S (seed) -d (mean) -s (stddev) -1 (read_len) -2 (read_len) \
        -N 4000000 (genome_fasta) (output_fastq_1) (output_fastq_2)
```

`-S` sets the pseudo-random seed, which we set arbitrarily for our experiments. `-d` sets the mean fragment length and `-s` sets the fragment length standard deviation. `-1` and `-2` specify the read lengths for either end of a paired-end read. `-N` specifies the number of reads to simulate. The last two parameters specify the reference genome FASTA file and the output FASTQ files.

Art. To generate unpaired samples we ran Art with these parameters:

```
art_illumina -sam -na -rs (seed) -l (read_len) -f (fold_coverage) \
        -i (genome_fasta) -o (output_fastq)
```

`-sam` enables SAM output. Information about a simulated read's point of origin is encoded in the SAM file. `-na` suppresses the ALN alignment file. `-rs` sets the pseudo-random seed, which we set arbitrarily for our experiments. `-l` specifies the read length. `-f` specifies the fold coverage. We set this to be high enough that we are eventually able to obtain 4 million reads or pairs per sample. `-i` specifies the reference genome FASTA file. `-o` specifies the output file.

To generate paired-end samples we ran Art with these parameters:

```
art_illumina -sam -na -p -rs (seed) -m (mean) -s (stddev) -l (read_len) \
        -f (fold_coverage) -i (genome_fasta) -o (output_fastq)
```

`-sam` enables SAM output. Information about a simulated read's point of origin is encoded in the SAM file. `-na` suppresses the ALN alignment file. `-rs` sets the pseudo-random seed, which we set arbitrarily for our experiments. `-m` specifies the mean fragment length. `-s` specifies the standard deviation. `-l` specifies the read length. `-f` specifies the fold coverage. We set this to be high enough that we are eventually able to obtain 4 million reads or pairs per sample. `-i` specifies the reference genome FASTA file. `-o` specifies the output file.

S.7 Read alignment

Bowtie 2. For paired-end samples, we ran Bowtie 2 with the `-I` and `-X` parameters to set the minimum and maximum concordant fragment lengths. We specified `-I 2L -X 4L` where `L` corresponds to the `-ll` and `-le` parameters specified when running Mason. For experiments where we varied sensitivity level, we also specified a sensitivity parameter: `--very-fast`, `--fast`, `--sensitive` or `--very-sensitive`. For experiments where Bowtie 2 was run in local alignment mode, we also specified the `--local` parameter. Additionally, Qtip always specifies the `--mapq-extra` option when running Bowtie 2, which enables feature-reporting features. When running Qtip together with Bowtie 2, the user may also specify additional Bowtie 2 parameters on the command line.

BWA-MEM. For paired-end samples, we ran BWA-MEM with the `-I` option to set the minimum and maximum concordant fragment lengths. BWA-MEM's default behavior, unlike either Bowtie 2 or SNAP, is to infer the fragment length distribution over a sample of the input reads. By specifying `-I` we disabled that automatic inference. This was to ensure that the read aligners worked similarly and, in particular, that all agreed on which fragment lengths should be considered concordant. The `-I` parameter sets the mean and standard deviation of the fragment length distribution. BWA-MEM consider any fragment length within 4 standard deviations of the mean to be concordant. Thus, we set `-I 3L,L/4`, achieving the same result as we did above with Bowtie 2's `-I` and `-X` parameters. When running Qtip together with BWA-MEM , the user may also specify additional BWA-MEM parameters on the command line.

SNAP. For paired-end samples, we ran SNAP with the `-s` parameter to set the minimum and maximum concordant fragment lengths. We specified `-s L 3L` indicating that the distance between the leftmost base of the upstream made and the leftmost base of the downstream made is allowed to vary from L to $3L$, achieving the same result as we did above with Bowtie 2's `-I` and `-X` parameters. Additionally, Qtip always sets SNAP's `-=` parameter for reasons described in Supplementary Material section S.4. When running Qtip together with SNAP, the user may also specify additional SNAP parameters on the command line.

S.8 Varying aligner sensitivity

We used Qtip together with Bowtie 2 to align and predict mapping qualities for the Mason-simulated 100 nt and 250 nt unpaired and paired-end samples. We repeated each experiment four times, varying Bowtie 2's sensitivity parameter: `--very-fast`, `--fast`, `--sensitive` or `--very-sensitive`. We also tried Qtip both in its (default) end-to-end alignment mode and in its local alignment mode. We calculated RCA and RCE for all 10 trials of all experiments (Table S2) and plotted CSED for the first trial of the experiments that used the 100 nt reads (Figure S1).

Qtip-predicted mapping qualities are superior overall to Bowtie 2's in every scenario tested, as indicated by the negative RCAs and RCEs (Table S2). Variability of RCAs and RCEs across trials is generally modest, but higher for `--very-sensitive` mode. Standard deviations range up to 5.92; see Discussion for further comments on variability.

CSED curves (Figure S1) again show that for some ranges of cutoffs, aligner-reported mapping qualities are better at minimizing cumulative squared error, corresponding to the portions of the CID curves that rise above $y = 0$. This is most clearly observed for unpaired `--very-sensitive` alignment. But Qtip provides superior predictions for the vast majority of cutoffs, including nearly all cutoffs for the paired-end experiments.

			End-to-end				Local			
			RCA		RCE		RCA		RCE	
	Length	Sensitivity	mean	sd	mean	sd	mean	sd	mean	sd
Unpaired	100	–very-fast	-9.55	1.04	-20.73	1.13	-17.06	0.37	-47.04	0.23
		–fast	-8.33	0.28	-19.96	0.26	-13.66	0.37	-36.52	0.42
		–sensitive	-7.62	1.03	-18.91	1.05	-11.68	1.23	-29.21	1.78
		–very-sensitive	-7.15	1.46	-19.10	1.16	-8.88	1.75	-22.07	2.85
	250	–very-fast	-25.26	0.77	-32.32	0.43	-32.44	1.59	-52.37	0.40
		–fast	-18.75	1.63	-24.21	0.71	-24.16	1.40	-35.51	0.60
		–sensitive	-15.53	2.67	-19.07	1.43	-19.83	4.30	-27.41	3.15
		–very-sensitive	-14.72	5.92	-16.80	3.59	-17.35	4.69	-21.53	1.51
Paired	100	–very-fast	-32.56	1.00	-50.48	0.26	-67.65	0.21	-49.77	0.63
		–fast	-33.69	0.51	-51.75	0.22	-66.36	0.12	-51.48	0.51
		–sensitive	-35.05	0.65	-54.19	0.12	-63.84	0.28	-52.11	0.61
		–very-sensitive	-37.23	0.36	-56.57	0.13	-60.65	0.25	-54.02	0.54
	250	–very-fast	-46.44	0.45	-57.22	0.25	-71.51	0.60	-62.20	1.86
		–fast	-45.04	3.03	-58.18	0.19	-69.35	0.34	-63.67	1.36
		–sensitive	-47.18	0.89	-57.79	0.20	-67.04	0.79	-62.40	1.14
		–very-sensitive	-47.73	1.14	-57.84	0.31	-63.48	1.07	-62.34	1.56

Table S2: Relative change in area under CID (RCA) and relative change in sum of squared error (RCE) when running Qtip and Bowtie 2 at various levels of sensitivity on Mason-simulated Illumina-like samples. Relative change is expressed as a percent. Each simulated sample consists of 4 million reads or pairs. Bowtie 2 is run in either end-to-end or local alignment mode as indicated. Results are means and standard deviations over 10 random trials, repeated starting from the input modeling step.

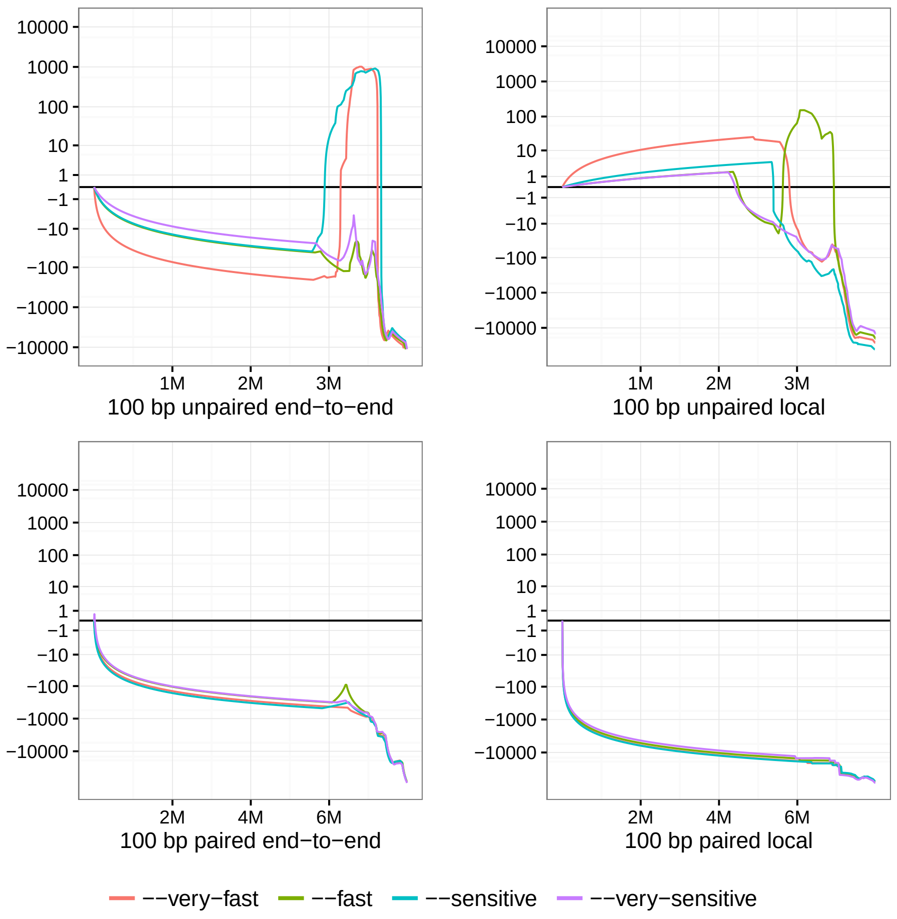

Figure S1: Cumulative squared-error difference (CSED) plot from running Qtip and Bowtie 2 and varying Bowtie 2's sensitivity level. The input reads are Mason-simulated Illumina-like 100 nt samples, each consisting of 4 million reads/pairs. The horizontal axis indicates cumulative number of reads/ends passing the cutoff, with the left-hand extreme corresponding to a high mapping-quality cutoff and right-hand extreme corresponding to a low cutoff. Results for unpaired samples are shown on top, and paired on bottom. Bowtie 2 is run in its (default) end-to-end mode in the case of the left-hand plots, and in local mode in the case of the right-hand plots.

S.9 Varying read simulator

Different read simulators make different decisions about how to emulate the sequencing process and its errors and biases. To study how this impacts Qtip's predictions, we ran a series of experiments using Mason[1] and the popular wgsim (`https://github.com/lh3/wgsim`) and Art[7] simulators. For all three, we simulated 100 nt and 250 nt samples, both unpaired and paired-end. See Supplementary Material section S.6 for the commands used and details about simulated fragment lengths. We then used Qtip together with Bowtie 2 to align and predict mapping qualities. We calculated RCA and RCE for all 10 trials of all experiments (S3) and plotted CID for the first trial (S2). Overall, the results show the choice of simulator does not have a large effect on the outcome of the experiment. Differences in prediction performance on the paired-end samples may be due in part to the different fragment length distributions generated by the simulators, as discussed in Supplementary Material section S.6.

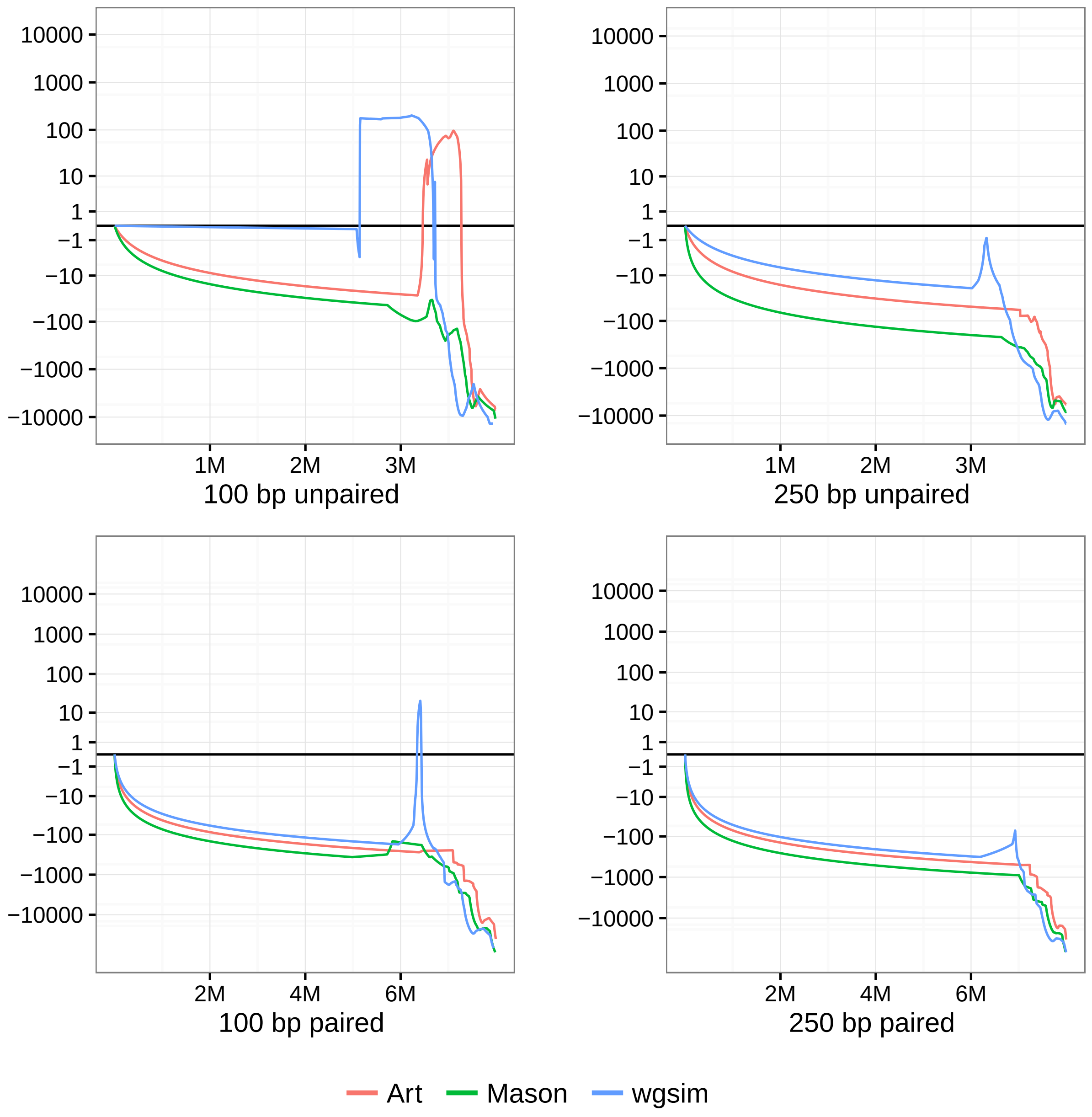

Figure S2: Cumulative squared-error difference (CSED) plot from running Qtip and Bowtie 2 on samples simulated from three different read simulators: Mason, wgsim and Art. Each sample consists of 4 million reads or pairs. The horizontal axis indicates cumulative number of reads/ends passing the cutoff, with the left-hand extreme corresponding to a high quality cutoff and right-hand extreme to a low cutoff.

			RCA		RCE	
			mean	sd	mean	sd
Unpaired	100 nt	Art	-6.26	0.35	-15.87	0.66
		Mason	-7.43	1.82	-18.53	2.61
		wgsim	-9.06	0.25	-19.60	0.90
	250 nt	Art	-12.41	0.84	-19.83	0.53
		Mason	-16.82	2.01	-19.97	0.44
		wgsim	-18.52	0.32	-26.93	0.50
Paired	100 nt	Art	-26.02	0.83	-41.98	0.26
		Mason	-34.93	1.00	-54.32	0.19
		wgsim	-26.37	0.84	-46.00	0.19
	250 nt	Art	-40.37	0.36	-47.21	0.30
		Mason	-47.93	0.26	-57.77	0.19
		wgsim	-39.18	1.10	-54.41	0.21

Table S3: Relative change in area under CID (RCA) and relative change in sum of squared error (RCE) for samples simulated by various simulators. Bowtie 2 end-to-end alignment was used in all cases. Relative change is expressed as a percent. The experiments used 100 nt or 250 nt reads, and unpaired or paired-end reads, as indicated. Results are means and standard deviations over 10 random trials, repeated starting from the input modeling step.

S.10 Variant calling experiment

Alignment: We used Qtip v1.6.1 together with Bowtie 2 v2.3.0 to align the paired-end sequencing reads from ERR194147. The Qtip command used was:

```
qtip \
    --ref ${FASTA_PATH}/hg38.fa \
    --m1 ERR194147_1.fastq --m2 ERR194147_2.fastq \
    --index ${INDEX_PATH}/hg38.fa \
    --bt2-exe ${BOWTIE2_PATH}/bowtie2 \
    --keep-intermediates \
    --output-directory ${ODIR} \
    --write-orig-mapq \
    --write-precise-mapq \
    --temp-directory ${TEMP} \
    -- ${BOWTIE2_ARGS}
```

The Bowtie 2 arguments used (inserted where `${BOWTIE2_ARGS}` appears above) are: `-I 0 -X 550 -t -p 24`. The first two parameters set minimum and maximum fragment length to 0 and 550 respectively. The `-t` parameter enables extra timing output. The `-p 24 --reorder` parameters run Bowtie 2 with 24 threads while keeping output alignments in an order corresponding to the input reads.

With these parameters, the output directory from Qtip contains both an `input.sam` file with the original mapping qualities, and a `final.sam` file with the Qtip-predicted mapping qualities for the same alignments.

Variant calling: After converting the SAM output from the alignment run to sorted BAM using sambamba[8], we then used Freebayes v1.1.0 to call variants:

```
freebayes \
    -X -u --haplotype-length 0 \
    -f ${FASTA_PATH}/hg38.fa \
    --min-mapping-quality ${MIN_MAPQ} \
    -v ${OUTPUT_VCF} \
    ${INPUT_BAM}
```

The `-X -u` arguments ensure Freebayes calls only SNVs and indels, though the indels calls were not studied here. The `--haplotype-length 0` argument ensures each SNV appears as a separate call in the VCF file, even in cases where nearby SNVs can be phased into a haplotype block. This helps make the SNVs more comparable between the Freebayes output and the Platinum calls. `-f` specifies the reference FASTA file. `--min--mapping-quality` specifies the minimum mapping quality an alignment must have to be included in the variant calling analysis. We run Freebayes repeatedly setting different values for `${MIN_MAPQ}`. We also tried running Freebayes in three other modes affecting the mapping quality threshold: (a) its default mode, (b) `--standard-filters` mode, and (c) `--use-mapping-quality`. But for none of the βs tested did the cutoffs enforced by any of those three modes yield maximal F_β. In fact, the thresholds yielding maximal F_β were always in the range 2–5.

The above command was run twice for every mapping quality threshold: once with `${INPUT_BAM}` is set to the sorted BAM derived from the `input.sam` file, and once with with `${INPUT_BAM}` set to the sorted BAM derived from `input.sam`.

Variant filtering: We applied two filters to the variant calls output by Freebayes. First, we eliminated variant calls with spuriously high coverage, as suggested in prior work[9]. Spuriously high coverage tends to indicate the presence of a copy number change that confounds variant calling. This was accomplished with a simple `awk` script that removed VCF records with depth (as indicated by the `DP` field of the `INFO` column) at least four poisson standard deviations above the mean. Mean depth of coverage was calculated to be 52.79 using `samtools flatstat`, so the filter removed variants with depth greater than 82. This script, and all others used for these experiments, can be found at `https://github.com/BenLangmead/qtip-experiments/tree/v1.6.1`.

Next, we filtered the Freebayes output to eliminate variant calls outside genomics regions deemed "high confidence" by the Platinum Genomes[10] project. We used the `vcfintersect` tool from the `vcflib` (`https://github.com/vcflib/vcflib`) collection; the specific command was:

```
vcfintersect -b ${PLATINUM_BED} ${VCF}
```

Where `${VCF}` was the file produced by Freebayes and `${PLATINUM_BED}` was a version of the Platinum Genomes high-confidence region `bed` file obtained from the Platinum Genomes FTP site (`ftp://ussd-ftp.illumina.com/2016-1.0/hg38/small_variants/`). We first preprocessed the `${PLATINUM_BED}` file to retain only regions in chromosomes 1–22 and X.

F scores: To obtain F-scores we compiled two `roc` files, one for the variant calls using the aligner-predicted mapping qualities and one for the calls predicted by Qtip. A `roc` file summarizes the number of true positives, false positives, and false negatives at all possible `QUAL` cutoffs. The `QUAL` field of the VCF file gives Freebayes' "genotype confidence." The ROCs were obtained by running `vcfroc`, a tool in the `vcflib` (`https://github.com/vcflib/vcflib`) collection. The `vcfroc` command was:

```
vcfroc \
    --truth-vcf ${PLATINUM_VCF} \
    --reference ${FASTA_PATH}/hg38.fa \
    ${VCF_FN} > ${ROC_FN}
```

Finding the maximal F scores: All of the above experiments were carried out for all mapping-quality thresholds and for both the original and the Qtip-predicted mapping qualities. For the original mapping qualities and given a particular β, we calculated maximal F_β by evaluating F_β:

$$F_\beta = (1+\beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

For every possible combination of mapping-quality and genotype-quality threshold. We then reported the maximal F_β.

Literature cited in supplement

1. Holtgrewe, M. Mason–a read simulator for second generation sequencing data. *Technical Report FU Berlin* (2010).
2. Erlwein, O. *et al.* DNA extraction columns contaminated with murine sequences. *PLoS One* **6,** e23484 (2011).
3. Lusk, R. W. Diverse and widespread contamination evident in the unmapped depths of high throughput sequencing data. *PloS one* **9,** e110808 (2014).
4. Olarerin-George, A. O. & Hogenesch, J. B. Assessing the prevalence of mycoplasma contamination in cell culture via a survey of NCBI's RNA-seq archive. *Nucleic acids research* **43,** 2535–2542 (2015).
5. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).
6. Geurts, P., Ernst, D. & Wehenkel, L. Extremely randomized trees. *Machine learning* **63,** 3–42 (2006).
7. Huang, W., Li, L., Myers, J. R. & Marth, G. T. ART: a next-generation sequencing read simulator. *Bioinformatics* **28,** 593–594 (2012).
8. Tarasov, A., Vilella, A. J., Cuppen, E., Nijman, I. J. & Prins, P. Sambamba: fast processing of NGS alignment formats. *Bioinformatics* **31,** 2032–2034 (2015).
9. Li, H. Toward better understanding of artifacts in variant calling from high-coverage samples. *Bioinformatics* **30,** 2843–2851 (Oct. 2014).
10. Eberle, M. A. *et al.* A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree. *Genome Research* **27,** 157–164 (2017).