SCOPE++: Sequence Classification Of homoPolymer Emissions

**James T. Morton1, Patricia Abrudan2, Nathanial Figueroa1, and Chun Liang1,2** §**, John E. Karro1,3,4** §

1Department of 1Computer Science and Software Engineering, 2Biology, 3Microbiology, and 4Statistics, Miami University, Oxford, Ohio, USA.

§Corresponding authors

Email addresses:

JTM: [mortonjt@miamiOH.edu](mailto:mortonjt@miamiOH.edu)

PA: [abrudapa@miamiOH.edu](mailto:abrudapa@miamiOH.edu)

NF: [figuernd@miamiOH.edu](mailto:figuernd@miamiOH.edu)

CL: [liangc@miamiOH.edu](mailto:liangc@miamiOH.edu)

JEK: [karroje@miamiOH.edu](mailto:karroje@miamiOH.edu)

# Abstract

**Background:** mRNA polyadenylation, the addition of a poly(A) tail to the 3’-end of pre-mRNA, is a process critical to gene expression and regulation in eukaryotes. To understand the molecular mechanisms governing polyadenylation and other relevant biological processes, it is important to identify these poly(A) tails accurately in transcriptome sequencing data and differentiate them from artificial adapter sequences added in the sequencing process. But the annotation of these tails is complicated by the presence of sequencing errors and post-transcriptional modifications. While determining that a tail is present in a given transcript fragment, these obfuscations make the problem of boundary identification a challenge; conventional seed-and-extend algorithms struggle to accurately identify these poly(A) tail end-points. Further, all existing tools that we are aware of focus exclusively on the trimming of poly(A) tails, failing to provide the detailed information needed for studying the polyadenylation process.

**Results:** We have created SCOPE++, a tool for finding the precise border of poly(A) tails and other homopolymers in raw mRNA sequence reads. Based on a Hidden Markov Model (HMM) approach, SCOPE++ accurately identifies specific homopolymer sequences in error-prone EST/cDNA data or RNA-Seq data at a speed appropriate for large sequence sets.

**Conclusions:** We demonstrate that our tool can precisely identify poly(A) tails with near perfect accuracy at the speed required for high-throughput applications, providing a valuable resource for polyadenylation research.

# Background

*Alternative polyadenylation* (APA) and *poly(A) tail length variability* have recently been identified as critical mechanisms in gene expression and regulation {Proudfoot:2011ke, Xing:2010cn, DiGiammartino:2011ez, Mayr:2009fd, Wu:2011eu, Choi:2012ip}. But conducting studies on their exact role in the regulation process is complicated by the challenge of collecting precise information on the presence and characteristics of poly(A) tails in transcriptome data. While finding a significantly long stretch of adenine bases in a transcript sequence is not difficult, the challenge deepens when you try to account for sequence modifications that could obscure the tail sequence purity (e.g. base-call errors, the effect of processes such as RNA editing, or sequencing artifacts). Tools such as SeqClean {Anonymous:EwaMJS6L}, TrimEst {Rice:2000wr}, or SeqTrim {Falgueras:2010bf} are able to effectively remove poly(A) tails via truncation, but cannot recover the detailed information needed when studying issues related to length variation. Thus a tool is required that is able to identify poly(A) tail boundaries and length, and is robust to disruptions in the homopolymer sequence.

Polyadenylation is a post-transcriptional process in which the 3’-end of a pre-mRNA is cleaved and replaced with a poly(A) tail to form a mature mRNA. Specifically, the polyadenylation protein complex binds to poly(A) signals, then cleaves the sequence at a *poly(A) site*, and finally collaborates with the poly(A) polymerase to perform non-templated adenine addition a few bases downstream of the appropriate poly(A) signals {Proudfoot:2011ke}. The poly(A) tail at the 3’ end of the mRNA is the hallmark of mRNA maturation, and also serves as a regulatory signal that is critical for mRNA nucleus-to-cytoplasm transportation, mRNA stability, and protein translation {Proudfoot:2011ke, Xing:2010cn}. Recent research suggests that many eukaryotic genes employ *alternative polyadenylation* (APA), in which multiple distinctive poly(A) sites are utilized to create different transcript isoforms from the same genes {Proudfoot:2011ke, Xing:2010cn, DiGiammartino:2011ez}. It is clear that APA is an important regulator in eukaryotic gene expression and regulation. For example, 3’-UTR shortening by APA appears to be highly active in cancer cells {Mayr:2009fd}. To increase our understanding of the underlying molecular and biological mechanisms governing polyadenylation and other relevant processes, RNA-seq data is continually being generated to aid in annotating the junctions of the 3’-UTR and the poly(A) tail {Wu:2011eu, Ozsolak:2010jm}. Moreover, 3’-end tagging (i.e., addition of non-templated U or C/U-rich tags) and 3’-oligouridylation (i.e., poly(U) tails) have been shown to affect mRNA degradation and are common in many eukaryotic species {Choi:2012ip, Morozov:2012bf}. Studies have also shown that the length of the poly(A) tail has a direct effect on mRNA stability, and mRNAs with short poly(A) tails can be stored in cytoplasm and reactivated later for translation by a re-polyadenylation process that elongates the tail lengths {Bonaiti:2006ik, Mendez:2001ds}.

We would like to extract data on poly(A) tail characteristics from the aforementioned RNA-seq data that is now so plentiful. However, the precise identification of poly(A) tails embedded within those sequences is a challenge, as the search for a contiguous sequence of adenine bases is complicated by the potential obfuscation of the sequence pattern by base-call errors and the presence of sequencing-induced artificial sequences (e.g., adapters, linkers and primers) added near the poly(A) tails. Such modifications, as well as those from poorly understood biological processes (such as RNA editing and non-templated nucleotide addition {Cheng:2001jl, JIN:2004ih}), can disguise the characteristic adenine sequence and result in impurity in the poly(A) tails. Certain tools, such as SeqClean {Anonymous:EwaMJS6L}, TrimEST {Rice:2000wr}, and SeqTrim {Falgueras:2010bf}, are able to reliably remove such tails by identifying one end of the tail and truncating it. But they are not able to provide the information needed to study the polyadenylation process itself.

In this paper, we introduce SCOPE++, an open-source software tool employing a Hidden Markov Model approach for the precise identification of the boundaries and length of poly(A) tails and other homopolymers in sequence reads. SCOPE++ runs at a speed appropriate for Next Generation Sequencing output sizes, with a capability to self-tailor its computational model to the characteristics of a given dataset through the use of machine learning algorithms. This makes it possible to precisely study poly(A) tail length and boundaries, and their roles in regulating gene expression. In particular, our tool is designed to accurately detect poly(A) tails of lower purity, where tail boundaries will be difficult to identify using conventional algorithms.

# Methods

SCOPE++ identifies poly(A) tails through the alignment of sequence reads to a predefined Hidden Markov Model (HMM) topology using the Viterbi algorithm. (For the interested reader unfamiliar with HMMs, the Viterbi algorithm, or Baum-Welch Training, a useful bioinformatics-oriented overview is presented in *Durbin et al.* {Durbin:1998wq}, with a more technical review in *Rabiner* {Rabiner:1986jk}). Employing sliding windows to initialize HMM parameter values tailored to the dataset, SCOPE++ utilizes the Viterbi algorithm to approximate the most likely position of a poly(A) tail within any given fragment. {Durbin:1998wq}

**HMM Topology**

SCOPE++ identifies poly(A) tails using a variable-state Hidden Markov Model {Durbin:1998wq} conforming to the topology illustrated in Figure 1(a), allowing for the identification of both perfect and imperfect poly(A) tails in raw sequence reads. Using sliding windows along a random sampling of the input as training data for initial parameter values (optionally coupled with Baum-Welch training for HMM parameter optimization), SCOPE++ utilizes the Viterbi algorithm to approximate the most likely position of a poly(A) tail within any given fragment. In Figure 1(a) we see the HMM topology, and in Figure 1(b) the decomposition of a fragment. In the alignment of the fragment each base will be assigned to either: the poly(A) state (thus indicating it is a member of a poly(A) tail); a background state (thus indicating it is not); or one of the intermediate A states. These middle states serve as tail boundary bases, which we require to be error free. Increasing the number of such states can increase the precision of boundary identification, at the cost of sensitivity as base-call errors become more likely to appear within the defined end region. Experiments indicated that using four such states (split two and two) achieves a reasonable balance.

The topology of the HMM is fairly simple, but requires the fragment conform to a certain template form. Specifically, any embedded tail must have *x* error-free adenine bases at either end (where *x=2* in Figure 1(a)), and the tolerance of non-adenine tail bases will be dictated by probabilities assigned in the poly(A) state. There is the possibility of a non-tail adenine sitting by chance close to the tail and thus being incorrectly annotated as part of that tail. However, there is virtually no way to distinguish such an aberrant base using only sequence information (or even establishing whether or not there actually an error) without wet-lab conformation, hence any tool will likely suffer from this. Two homopolymers sitting in close proximity could be merged (with the intervening bases labeled as errors within the tail). However, the occurrence of several consecutive non-A bases will result in a poor fit to the model, and hence are unlikely to be accepted. In practice, the probability of two homopolymers actually occurring close enough to each other be a problem appears to be very small.

**Parameter Estimation**

Starting with this fixed topology and a set of fragments, we needed to estimate several parameters for the HMM which may be dependent on the characteristics of the biological data or the sequencing processes. Specifically, we needed transition probabilities (the probability of moving between the homopolymer and background regions in the fragments), and emission probabilities (the base distribution for each fragment type). Note that the probabilities involving the end-segment states (the “A” states) are fixed: each of these states must transition to the next state with a probability of 1. In other words, we require that *x* bases at each end of the tail be adenine(using a user specified *x* with a default value of 2).

Given a large set of input data and the fixed HMM topology, we estimate the remaining parameters by taking a sampling of the fragments, quickly determining the approximate location of the poly(A) tails, and using that data for the basis of the estimation. Specifically: we apply a sliding-window scoring filter of user-specified width *M* to a random sampling of the input data. Each window is scored by calculating the difference in the number of adenine and non-adenine bases. (Or thymine v. non-thymine: the tool is able to search for poly(T) tails as well, though for the purpose of this discussion we will confine ourselves to discussing poly(A) searches.) As illustrated in Figure 2(a), we identify a window *WCenter*with the highest such score and, if the score is larger than a pre-determined threshold, we treat this as a potential subsequence of a poly(A) tail. We then find the first window to the left, *WLeft*, whose score is equal to the threshold value, and identify a *WRight* in a symmetric fashion. Finally, we merge together all windows to form one contiguous sequence representing a hypothetical poly(A) tail to be used for parameter training (see Figure 2(b)). We note that this method intentionally tends towards the over-estimation of the actual boundaries, allowing us to compensate for the presence of base-call errors at the ends of the poly(A) tail. From these putative sets we can then estimate the HMM parameters: transition probabilities are based on mean tail length with the assumption a geometrical distribution, while emission probabilities are sampled directly from the contents of the putative fragments.

Following this, estimates can be further refined using the Baum-Welch algorithm {Durbin:1998wq}, which is designed to optimally fit an HMM model to a given training dataset. In practice, we find that use of Baum-Welch increases runtime with a negligible improvement in results. It appears that our initial estimate of the model parameters are fairly close to the estimates returned by Baum-Welch, hence the algorithm is not worth the extra time.

Once we have HMM parameter estimates, SCOPE++ independently applies the Viterbi algorithm {Durbin:1998wq} to each sequence, getting back the best fit of the sequence to our model (and thus an assignment of each base to one of the characterizing states). While the Viterbi algorithm generally takes *O(mn2)* time (where *m* is the length of the sequence and *n* is the number of states in the model), the structure of our model allows us to reduce that to *O(mn)* time. As the number of states employed is fixed, in practice the runtime is linear with respect to the size of the sequence fragment.

Benchmark Sets: There is a lack of beachmark sets on which to test our tools; there has been some work on experimentally identifying tails {Subtelny:2014id, Chang:2014dh}, but the information produced by the studies describes tail length *distribution*, not the individual tail positions needed for testing. Hence we have developed a “semi-synthetic” benchmark set: namely, we have taken a set of real sequences with human-identified tails and artificially cleaned the tails of all impurities (that is: we convert all non-adenine bases to adenine). The result is a set of synthetic sequences with characteristics highly correlated to real data. While the human annotation is subject to human error, by cleaning them of impurities we have essentially turned the human-identified tails into simulated tails that we would expect a tool to identify. The non-tail portions of our synthetic fragments are direct copies of actual biological sequences, hence containing within them any sequence characteristics that might have an effect on the performance of the tool. We can then introduce simulated base-call error at a controlled rate, providing large datasets with known tail locations on which to test and compare different tools.

# Results

In assessing the quality of SCOPE++ results we look at two metrics: its ability to correctly identify poly(A) tails (sensitivity), and its ability to find the tail boundaries (precision). (Specificity is not addressed, as all tools concerned are new perfect in avoiding false positive – not a difficult task for any tool, and not worth discussion here). We assess sensitivity with the standard formula, the ratio of true positives to actual positives. To assess a tool’s precision, we measure result quality in three ways: *% correct*, *average trim error* and *sum of squares error*. *% correct* is the fraction of tails with correctly identified end points. *Average trim error* is the distance between estimated and actual boundaries (with positive values indicating the estimation is extended past the actual boundary – including bases it should not be), averaged over all incorrect sequence. *Sum of squares error* is the average of the square of trim error. The last reflects the fact that the seriousness of boundary error increases super-linearly with the error (e.g. being off by four bases is more than twice as problematic as being off by two bases), given the effect of such error on downstream analysis.

First we test whether the complexity of an HMM approach is warranted (given the apparently simple nature of the problem) by comparing SCOPE++ against a basic string search algorithm. Using our semi-synthetic dataset (see Methods), we compare against an algorithm that searches for maximal substrings *s* such that: (1) *s* is no shorter than a fixed value *m*, and (2) the fraction of non-A bases in *s* is no greater than a fixed value *p*. We find that while this basic algorithm can effectively detect the presence of poly(A) tails, it cannot match SCOPE++ in boundary detection accuracy. Experiments indicate that an effective parameter assignment is *m=10* and *p=0.15*, and with those values we find both tools have near-perfect sensitivity (and specificity) for data subjected to a base-call error rate ranging from 0 to 0.1 errors per base. But the simple algorithm suffers in boundary precision. With a 0% base-call rate (i.e. perfect tails), the simple tool is able to correctly identify boundaries less than 25% of the time, and over-extends the boundaries by an average of 8 bases; under these conditions SCOPE++ returns perfect results. Bumping the error rate improves the simple tail results slightly, as more errors in the tail prevent over-extension. But even at a 5% error rate the tool can only identify 34% of the boundaries correctly, and it over-extends by an average of 4.5 bases. SCOPE++ correctly identifies end points 73% of the time, and those boundaries it misses are short by an average of 0.6 bases. In short: the simple algorithm is useful if we are merely interested in determining if a poly(A) tail is present, but the complexity of SCOPE++ is required if precision is important.

Next we test whether the HMM is an appropriate structural match for the biologically-dictated structure of the sequences. To test this we compare SCOPE++ as described (using *x=2*, thus requiring each tail start and end with 2 As) against a basic two state HMM (containing a “poly(A)” state and a “non-poly(A)” state). Using the same training procedure for the simplified HMM that we use for SCOPE++ (see Methods), the tool tends to assign all bases to the non-poly(A) state. The simplified HMM is able to identify only about 10% of the fragments having tails (as opposed to SCOPE++’s 100%), even when dealing with perfect tails: while tail identification is simple, it is not that simple. We conclude from this that the complexity of our model is in some way describing the appropriate biologically-dictated sequence structure – at least beyond what the trivial HMM can describe.

Having justified the idea behind the general approach, we now move to a comparison against existing tools. For this we looked at the TrimPoly module of SeqClean {Anonymous:EwaMJS6L} and TrimEST {Rice:2000wr}; SeqTrim {Falgueras:2010bf} was not used due to its slower runtime. Table 1 displays the average sensitivity and specificity of SCOPE++, TrimPoly {Anonymous:EwaMJS6L}, and TrimEST {Rice:2000wr} over the six different semi-synthetic data sets (described in Methods). For each simulation we start with our benchmark dataset of 500 “cleaned” sequences, and for each sequence we randomly generated 200 simulated sequences by stochastically introducing error into the poly(A) tail, as well as 200 sequences without tails (formed by randomly sampling from the non-tail portion of the sequence). Altogether, this gives us 10,000 tail-containing simulated fragments derived directly from actual fragments and 10,000 tail-omitted fragments based on similar base distributions. Using this set, we observe a almost perfect sensitivity and specificity in both the SCOPE++ and TrimPoly tools.

However, when looking at the *precision* at which boundaries can be identified, we see a different story. We find that SCOPE++ identifies the correct boundaries in a significantly higher number of tails than the other tools (using the 3% simulated error rate, SCOPE++ correctly identified 77% of the sequence boundaries, as opposed to 28% for TrimPoly), and has a significantly smaller average sum-of-squares error rate. In short, SCOPE++ is considerably more precise.

We also look at the performances of SCOPE++ and TrimPoly as functions of both tail length and the length of adjacent sequencing-adapter sequence (a portion of the fragment added downstream of the tail as an artifact of the sequencing process). We find that, while TrimPoly performs better for identifying very short poly(A) tails (< 20 bp), it is highly sensitive to the length of any adapter remaining on the fragment; SCOPE++ is completely robust to such interference. By augmenting the length of tails or adjacent sequencing-adapter fragments of real data in simulation, we can examine the effects of tail length and adapter length on sensitivity (see Figure 3(a)). We observe a significant decrease in the sensitivity of TrimPoly as the adapter length grows beyond the length of the tail – a factor having no effect on the sensitivity of SCOPE++. In Figure 3(b) we plot sensitivity as a function of tail length for fixed adapter segment length. Once again, while SCOPE++ can handle any such values, TrimPoly suffers in the presence of adapter segments longer than 250 bp – falling to a sensitivity of less than 0.4. In Figure 4 we verify this assertion with real data [5]. Figure 4(a) shows the number of fragments identified as containing a tail and an adapter of length ≥ m (with adapter lengths for TrimPoly identified sequences determined by an ad-hoc post processing scan), while in Figure 4(b) we see the same information as a percentage of the total identified (i.e. the sensitivity). The findings are consistent with the simulation: as the tails shift deeper into the fragment, the relative ability of TrimPoly to identify those tails diminishes significantly.

**Large Dataset Validation**

In order to provide some validation on actual data, we set SCOPE++ to find poly(T) tails and ran it against a 17 GB Arabidopsis dataset {Wu:2011eu}. Developed using poly(T) tag sequencing, it has been estimated that about 60% of the reads within this set contain poly(T) tails; our tool discovered them in 59.6% of all reads; see Table 2 for details n the results. A quick search of the set dataset for sequences containing short homopolymers returns only 126 clear false negatives, all of which were present in low quality regions and probably would have been discarded anyway.

# Availability and Requirements Our software can be found at <https://github.com/mortonjt/SCOPE>, with code to be distributed as open source. SCOPE++ was implemented in C++11, and has been tested using the gnu g++ compiler (v. 4.6) on both OS X and Ubuntu Linux.

# Conclusions

Here we have presented SCOPE++, a novel approach for identifying imperfect homopolymers when end-point precision is important. We have compared SCOPE++ to simpler tools, and in doing so established the necessity of addressing a seemly simple problem with a complex approach. Unlike other tools, SCOPE++ is able to identify end boundaries at both ends of a tail (as opposed to simple trimming at the inside end) and is able to train on data-specific model parameters. Our tests have shown that the tool performs on par with existing poly(A) trimming tools in terms of speed, while showing considerably more precision in terms of identifying end-point boundaries and sensitivity in identifying tails buried further away from the fragment ends.

The success of SCOPE++ also indicates some potential for generalizing the approach to related problems, such as the identification of low-complexity repeats or simple artifacts that might need to be weeded out of a sequence in a precise manner. While HMM approaches are frequently time consuming and can be over-kill for this sort of problem, in cases where we can keep the complexity of the model fairly low and the number of states small, the same techniques might work to keep the runtime within reason.

# Authors' contributions

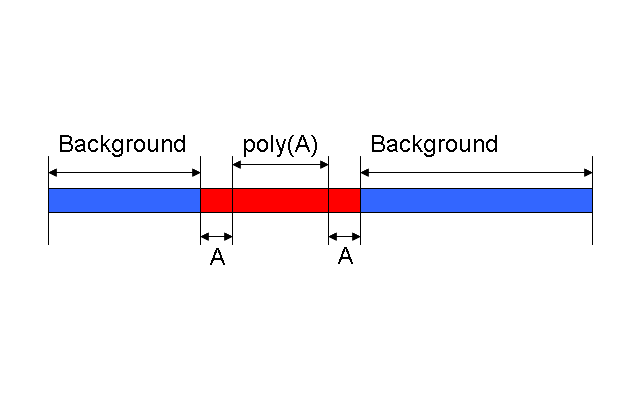
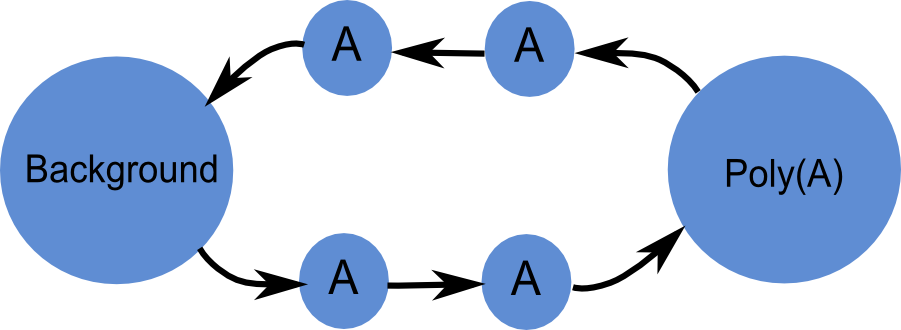
JK and LC designed, coordinated and managed the whole project. JM designed, developed and implemented most of the software algorithms. PA conducted human validation on the tests. NF developed software for finding the optimal parameters of SCOPE++. JK contributed to the algorithm design and conducted all of the simulation tests and performed a statistical analysis of the results. LC contributed to software testing and conducted human validation on the results. All authors read and approved the final manuscript.

# Acknowledgements

Funding: This project was funded partially by the National Science Foundation (No. O953215 to JK) and the NIH-AREA (1R15GM94732-1 A1 to CL).

**Data deposition**: The sequence reported in this paper has been deposited in the National Center for Biotechnology Information Short Reads Archive (accession no. [SRA028410](http://www.pnas.org/external-ref?link_type=GEN&access_num=SRA028410)).

# Figures



(a) (b)

Figure : (Left) A generalization of the HMM topology used for identification. The number of A states between the background and poly(A) states can vary depending on user priorities, with a larger number of states leading to more precise boundary identification but loss of overall sensitivity. Using a total of four states (split two and two) achieves a reasonable balance in practice. (Right) Example of a sequence read containing an identified poly(A) tail that has been divided into four types of segment: background (bases not part of the poly(A) tail), upstream tail-ends (tail bases within *x* of the upstream tail boundary), interior tail bases (further than at least *x* bases away from either boundary), and down-stream tail-ends.

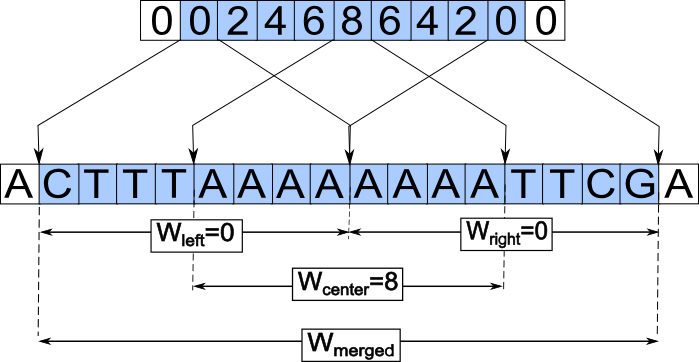


Figure 2: The sliding window scoring scheme used to provide an estimate of the HMM parameters. (a) Illustration of score calculation. (b) Finding *Wcenter*.



Figure 3: (a) Plot of tool sensitivity as a function of the ratio of adapter segment length to tail length: while SCOPE++ is robust to this ratio, TrimPoly sensitivity deteriorates as the length of the adapter segment becomes longer than that of the tail.



Figure 4: Application of tools to the Triticum aestivum ERR125556 fragment set {Anonymous:j6lilaHr}. With 140,000 tail-containing sequences identified, (a) shows the number of sequences with adapter segments of length ≥ m that are identified by SCOPE++, TrimPoly, or both. In (b) we see the percentage of those sequences that were identified only by SCOPE++ or TrimPoly.

**Table 1:** Results of tests on data-derived simulated sets using SCOPE++, the TrimPoly component of SeqClean {Anonymous:EwaMJS6L}, and TrimEst {Rice:2000wr}; SeqTrim {Falgueras:2010bf} was too slow for testing at the necessary scale. Starting with a set of 500 human-annotated sequenced transcript fragments (Illumina-sequenced Arabidopsis, 454-sequences Chalamydomons, and Sanger-sequence Human [21]), we remove all sequencing errors within the human identified poly(A) or poly(T) tails, then introduce simulated errors into those tails to allow for the testing of sensitivity and boundary identification at a controlled error rate. For specificity we generate sequences using the base distribution of the non-tail segments of the sequences in the set. Applying each tool to the set, we then have a reference solution that allows us to compute result qualities.

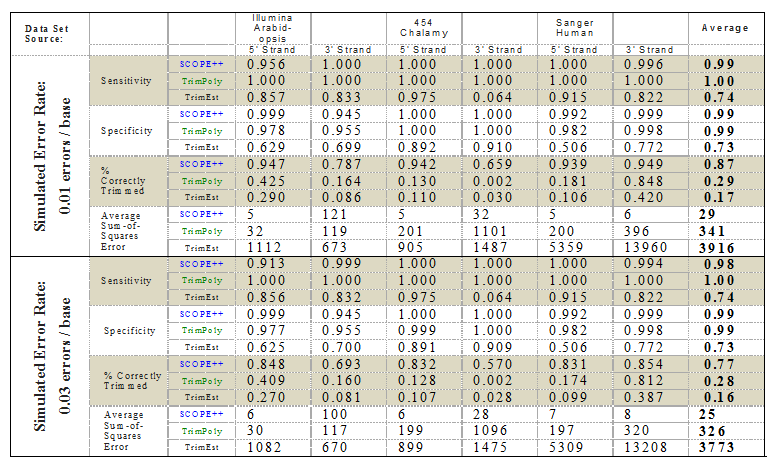


Table 2: With the SCOPE++ trim option, poly(A) tails and poly(T) tails can be simultaneously trimmed. The table below lists all possible arrangements of poly(A) and poly(T) tails within a single read and the frequencies in which they appear in the actual dataset {Anonymous:YKtSyhnz}.

|  |  |  |
| --- | --- | --- |
| Trim type | Number identified | Percent identified |
| [TTTT.....................]: | 45577831 | 53.1196% |
| [.....................TTTT]: | 679815 | 0.792303% |
| [AAAA.....................]: | 15734 | 0.0183375% |
| [.....................AAAA]: | 83200 | 0.096967% |
| [TTTT............TTTT]: | 2864767 | 3.3388% |
| [AAAA............AAAA]: | 340 | 0.000396259% |
| [TTTT............AAAA]: | 2733703 | 3.18605% |
| [AAAA…….....TTTT]: | 1059 | 0.00123423% |
| No homopolymers | 33845914 | 39.44636558% |

{papers2\_bibliography}

[21]

Conifergdb: http://bioinfolab.miamioh.edu/bioinfolab/index.php