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. Conventional seed-and-extend algorithms struggle to accurately identify these poly(A) tails. All existing tools that we are aware of focus exclusively on the identification and trimming of poly(A) tails, failing to provide the detailed information needed for studying the polyadenylation process.

**Results:** To address this problem we have created SCOPE++, a tool for finding the precise border of 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.

**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 *polyadenylated tail length variability* have recently been identified as critical mechanisms in gene experession and regulation {Proudfoot:2011ke}{Xing:2011cn}{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 polyadenylated (poly(A)) tails in transcriptome data. While finding a significantly long stretch of adenine bases in a sequence transcript is not difficult, the challenge deepens when you try to account for sequence modifications that could obscure the tail sequence (e.g. base-call errors, the effect of processes such as RNA editing, or sequencing artifacts). Tools such as SeqClean {Anonymous:SLVrguZz}, TrimEst {Rice:2000wr}, or SeqTrim {Falgueras:2010bf} are able to effectively remove poly(A) tails via truncation, but cannot recover the detailed information needed in studying issues related to length variation. We instead require a tool 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, cleaves the sequence at a *poly(A) site*, and 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:2011cn}. 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 gene {Proudfoot:2011ke}{Xing:2011cn}{DiGiammartino:2011ez}. It is clear that APA is an important regulator in eukaryote 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 query 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) or 3’-oligouridylation (i.e., poly(U) tails) have been shown to affect mRNA degradation and are common in many eukaryotic species{Choi:2012ip}{Rice:2000wr}. 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}.

Data relevant to the role of polyadenylation in gene regulation is best collected from sequenced mRNA and cDNA sequences. 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, frustrating traditional seed-and-extend algorithms. Certain tools, such as SeqClean {Anonymous:SLVrguZz}, TrimEST {Rice:2000wr}, and SeqTrim {Falgueras:2010bf} are able to reliably remove such tails by identifying one end and truncating, and are hence useful for those interested in obfuscated coding sequence. 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 for 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 tail length, alternative expression, and their role 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 with potentially imperfect poly(A) tracts to a predefined Hidden Markov Model (HMM) topology. (For interested readers unfamiliar with HMM, a useful bioinformatics-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.

**HMM Topology**

SCOPE++ identifies poly(A) tails using a variable-state Hidden Markov Model {Durbin:1998wq} conforming to the topology illustrated in Figure 1, 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 we see the HMM topology, and in Figure 2 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. Computational experiments indicated that using four such states (split two and two) achieves a reasonable balance (data not shown).

**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 otherwords, we required 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 estimated the remaining parameters by taking a sampling of the fragments, determining the approximate location of the poly(A) tails. Specifically: we employed 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 purposes of this discussion we will confine ourselves to discussing poly(A) searches.) We identified a window *WCenter*with the highest such score and, if the score exceeds a pre-determined threshold, we take this as a potential subsequence of a poly(A) tail. The boundary windows, Wleft and Wright, are then selected by finding the windows with scores at the threshold value surrounding *WCenter*, leaving us with a window continuing a high A content and thus labeled a putative tail. We illustrated the scoring in Figure 3(a) with an example using *M=8,* and the choice of putative sequences in Figure 3(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 estimated the HMM parameters: transition probabilities are based on mean tail length and the assumption that the length is geometricly distributed, 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}, an algorithm 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 tat our initial estimate of the model if fairly close to that found by an application of Baum-Welch, hence the algortihm is generally not worth the text time.

Once the above is completed, the remainder of the algorithm consists of independent applications to each sequence of the Viterbi algorithm, used to find an optimized fit of the sequence to the model {Durbin:1998wq}. While Viterbi can be time consuming in many application, in this situation that is not the case. Specifically: 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 allow us to reduce that to *O(mn)* time. And as the number of states employed is frequently fixed, this is further reduced to runtime linear in the size of the sequence fragment.

Benchmark Sets: Given the lack of benchmarks sets on which to compare tools, we have developed a “semi-synthetic” benchmark set: namely, we have taken a set of real sequences with human-identidied tails and artifically removed the tails of all impurities (that is: conver all non-adenine bases to adenine). The result are synthetic sequence with characteristics highly correlated to real data. While the human annotation is likely the subject of human error, by cleaning them of impurities we have essentially turned the human-identified tails into actual (simulated) tails that should be identified by our tool. The non-tail portions of our synthetic fragments are direct copies of actual biological sequence, hence preserving and hidden 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.

To test the quality of results against other tools we required that the benchmark sets contain sequenced transcript fragments in which the exact location of the tail is known – allowing us to calculate tool error. However, as no such sets were available, we generated *partially* *synthetic* benchmarks for as follows. Using six different data sets of sequenced cDNA sequences from three different organisms (human, arabidopsis, and chlymydominus [CITE] ), for each data set we randomly picked 500 squences and annotated them for poly(A) tails based on human inspection. For each sequence on which a biologist found a sequence with a high concentration of As, we then “purified” the sequence by replacing all non-A bases with As – thus giving us a possibly synthetic pure poly(A) tail any tool should be able to find at the position be believe to contain a real poly(A) tail. Finally, we simulated base-call errors in the poly(A) tail by replacing each A base with another with probability *p*. At the end, we have a real coding sequence with an embedded poly(A) based on a human estimated annotation and artifical errors. This allowed us to compute result quality over a model that should be very close to reality. Granted that human inspection is prone to transcription errors and other mistakes, this step of validation can be used to estimate the distribution of poly(A) tail lengths and positions required to build a model for simulating poly(A) tails.

# Results

In assessing the quality of SCOPE++ results we looked at three metrics: its ability to correctly identify poly(A) tails (sensitivity), its ability to avoid incorrectly identifying a sequence as containing a tail (specificity), and its precision in identifying tail boundaries. To assess a tool’s precision in boundary estimation, we measured result quality in two ways: *% correct* and *sum of squares error*. The first is the fraction of tails with correctly-identified end points, while the second is computed by looking at the average square of the boundary identification error. We used sum-of-squares error to reflect that the seriousness of boundary error increases super-linearly with the error (e.g. being off by four bases is more than twice as bad as being off by two bases), given the effect of such error on downstream analysis.

For comparison we looked at the TrimPoly module of SeqClean {Anonymous:SLVrguZz} and TrimEST {Rice:2000wr}; SeqTrim {Falgueras:2010bf} was not used due to its slower runtime. In Table 1 we saw the average sensitivity and specificity of SCOPE++, TrimPoly, and TrimEST {Rice:2000wr} over the six different semi-synthetic data sets (described in Methods). For each simulation we started with our benchmark dataset of 500 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 by randomly sampling from the non-tail portion of the sequence. Altogether, this gives us 10,000 tail-containing fragments derived directly from actual fragments and 10,000 tail-omitted fragments based on similar base distributions. On these results we saw 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 found that SCOPE++ identified the correct boundaries a significantly higher number of tails (averaging 77% of the dataset, as opposed to 28% of for TrimPoly on data with a 3% error rate), and has a significantly smaller average sum-of-squares error rate. In short, SCOPE++ is considerably more precise.

We also looked at the performance of SCOPE++ and TrimPoly as a function of both tail length and the length of adjacent sequencing adapter sequence (a portion of the fragment added downsteam 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 – while SCOPE++ is completely robust to such interference. By augmenting the length of tails or adjacent sequencing adapters fragments of real data in simulation, we can examine the effects of tail length and adapter length on sensitivity (see Figure 4(a)). We observed 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 significance of SCOPE++. In Figure 4(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 5 we verified this assertion with real data – data set ERR125556 [15]. In Figure 5(a) we saw the number of fragments identified as containing a tail and an adapater of length ≥ m (with adapter lengths for TrimPoly identified sequences determined by an ad-hoc post processing scan), while in the second we saw the same information as a percentage of the total identified. The findings were consistent with the simulation: as the tails shift deeper into the fragment, the relative ability of TrimPoly to identify them diminishes significantly. In Figure 5(b) we looked at the sensitivity of each tool, as a function of tail length, for fixed values of adapter tail length. Again, SCOPE++ returns near-perfect results, while TrimPoly’s sensitivity is substantially impacted by longer adapter segments.

**Large Dataset Validation**

To verify the simulated analysis of our tool, SCOPE++ was run against a 17 GB Arabidopsis dataset [CITE: Arabidopsis]. The Arabidopsis data set was developed using poly(T) tag sequencing and it was estimated that around 60% of the reads contained poly(T) tails. Our tool discovered that 59.6% of all of the reads contained poly(T) tails, which was very close to the initial estimate. A quick search 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 trashed 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. In comparison to other tools, SCOPE++ has several advantages, including the ability to identify both end boundaries and to tailor/train its model parameters to fit different sequence technologies. From our tests, we have verified that SCOPE++ performs on par with existing poly(A) trimming tools in terms of speed, while showing considerably more accuracy in terms of identifying end-point boundaries and predicting tails buried further away from the fragment ends. As a result, we conclude that SCOPE++ is the best tool for poly(A) studies, where accurate positional and boundary information of poly(A) tails are mandatory.

# Authors' contributions

JM designed, developed and implemented most of the software algorithms. PA conducted human validation on the tests. NF created found the optimal parameters of the software. JK coordinated this project and conducted all of the simulation tests and performed a statistical analysis on the results. LC coordinated this project 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).

# Figures

Figure : A generalization of the HMM topology used for identification. The number of A states between the background and poly(A) states can be varied 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.

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

Figure : (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 : Application of tools to the Triticum aestivum ERR125556 fragment set {Anonymous:JztaqS8u}. 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:SLVrguZz}, 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 [CITE: Conifergdb]), we remove all sequencing errors within the human identified poly(A) or poly(T) tails, then introduce simulated error into those tails to allow the testing of sensitivity and boundary condition 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.

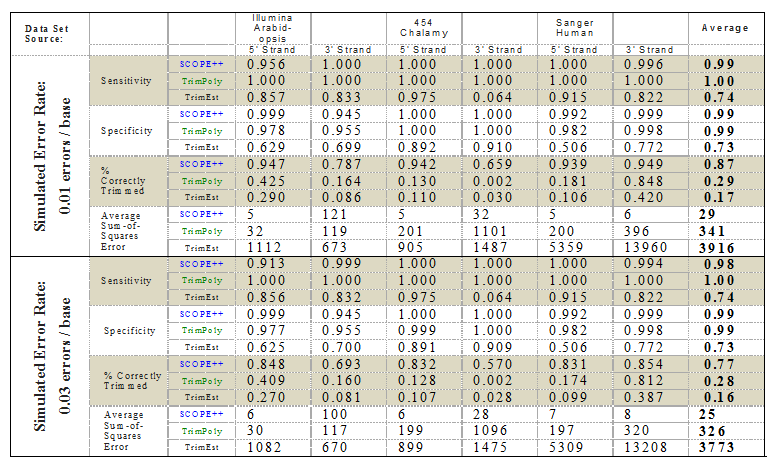


Table 2: In SCOPE++ trim option, poly(A) tails and poly(T) tails can be simultaneously trimmed. The table below lists all possible arrangements of poly(A) and poly(T) tails within a single read and the frequencies in which they appear in the actual dataset {Anonymous:9FrKu31h}.

|  |  |  |
| --- | --- | --- |
| 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% |