Finding and evaluating RNA motifs with CMfinder

Joyce Zhou
Advised by Larry Ruzzo

1 Abstract

Non-coding RNA (ncRNA) is transcribed RNA that does not code for protein, but instead folds into a functional shape. Many ncRNA scans have been conducted within the last 20 years with a variety of tools, with special focus on 3' and 5' UTR regions of genes. However, full-genome scans remain computationally expensive and limit their search sequences to individual alignment blocks. This has the potential of concealing ncRNA by truncating it at alignment block boundaries.

We implemented a new tool `blockmerger` for combining adjacent multiple alignment blocks, excluding species which cannot be combined. We used this merge tool with CMfinder to search for ncRNA in four regions of the human genome of varying lengths and found many promising candidate motifs, some of which reflect results previously found, and many of which span alignment block boundaries.

We also designed and evaluated six different methods to estimate K-L divergence between covariance models. Using one of these divergence estimation methods, we clustered a small subset of our ncRNA candidates with Spectral Clustering.

2 Introduction

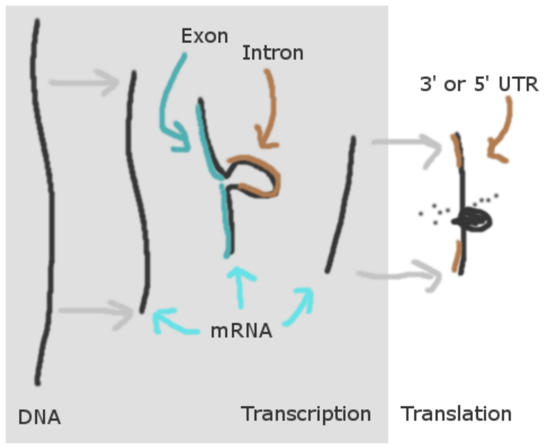
We introduce the concept and importance of non-coding RNA (ncRNA), as well as different methods that have been used to try to find new ncRNA genes. We also explain the motivations of this thesis.

2.1 Background

The "central dogma" in biology describes the process of protein synthesis, wherein each gene in an organism's DNA is responsible for encoding one protein. According to the central dogma, messenger RNA (mRNA) is transcribed from the DNA of a gene and then translated into protein.

However, not every base of a messenger RNA is directly translated. There are significant 5' and 3' untranslated regions (UTR) located before and after the codons that denote where the protein-coding region start and stop. Furthermore, mRNA is spliced to remove introns before it is translated [1]. About 90% of either strand of the entire human genome is transcribed into RNA somewhere in the body, but known proteins account for less than 2% of the genome [7]. The large amount of otherwise unused RNA transcripts has the potential to be very useful.

An example diagram of RNA splicing and common ncRNA locations outside of protein-coding region:



Because RNA is single-stranded, it is able to form base pairs with itself, which can form functional structures. Thus, functional segments of RNA transcripts are considered "non-coding RNA" (ncRNA) or "ncRNA genes". Similarlystructured ncRNAs can be grouped into ncRNA "families", akin to protein families.

Many ncRNAs are important in the mechanisms of gene (both RNA and protein) expression regulation. There are a couple historically very well-known examples of ncRNA families, such as transfer RNA (tRNA) and ribosomal RNA (rRNA), which are directly involved in the process of protein translation. More recently, large ncRNA families such as microRNAs and riboswitches were respectively found to regulate the transcription and translation of mRNA itself [4]. As of 2017, the Rfam database for ncRNA families contains over 2600 unique families and counting [9].

ncRNA functionality generally comes from its secondary structure (the folded shape, which comes from a set of base pairings) instead of primary structure (the RNA sequence itself), so the secondary structure is conserved more than the primary structure [7]. Compensating base changes are a key example of this secondary structure conservation [7]. For example, a C-G base pair in a mouse ncRNA could be replaced by an A-U base pair in homologous monkey ncRNA, which would preserve the secondary structure but alter the primary structure.

An example of sequence v.s. structure conservation:

Original	`GACAAACUUGUC`
	`<<<<<-->>>>>`
Protein Same	`GACAAACUGGUG`
	`-<<-<-->->>-`
Structure Same	`UAGACACGUCUA`
	`<<<<<-->>>>>`

Therefore, in contrast to protein discovery methods which can take advantage of sequence conservation (similarity between the DNA of different species) to find regions of interest [20], ncRNA discovery algorithms largely rely on structural conservation (similarity between the potential folded structure of an RNA transcript from the DNA of different species) [7]. However, most comparative sequence analysis (such as the MULTIZ multi-way sequence alignments) focuses on sequence conservation, which has the possibility of misaligning structurally conserved elements, as they may have low sequence similarity [7, 25]. Because alignment tools such as MULTIZ generate sets of aligned conserved sequences in alignment blocks, structurally conserved elements which should be aligned in different species may end up skewed across different alignment blocks.

The approaches of ncRNA discovery algorithms can be grouped into three major categories: "align-first", "fold-first", and "joint" [7]. The "align-first" approach establishes a shared structure based on alignment of sequences, but can fail if there is too little or too much sequence conservation [7, 25]. The "fold-first" approach aligns sequences based on potential structures for each of them [7]. The "joint" approach aims to use both sequence and structural similarities, but can be computationally expensive [7].

For our investigation, we use `CMfinder`, which takes the "fold-first" approach and is designed to work with a potentially unrelated set of input sequences [25].

2.2 Related Work

Some previous major ncRNA discovery algorithms based on sequence alignments include EvoFold [15], [18], and RNAz [8]. Algorithms based on structure alignments include FOLDALIGN [21] and `CMfinder`.

We chose to focus on `CMfinder` because of its functionality on unaligned sequences [25], as well as to be able to compare our results with previous `CMfinder` scans.

Previous scans for ncRNA elements using `CMfinder` have investigated: the 5' upstream sequences of homologous genes in Firmicute prokaryotic bacteria [26], the 5' upstream sequences of homologous genes in all fully sequenced bacteria [23], and the full genomes of several vertebrates used in the MULTIZ 17-way alignment, segmented into individual alignment blocks [19].

The `CMfinder` package also includes a program called pscore which evaluates the quality of a given motif. Previous scans have used cutoffs on pscore to select interesting motifs likely to be true positives. We can also evaluate the accuracy of that cutoff by generating randomized sequences and calculating a false discovery rate (FDR). For example, the vertebrate scan used a pscore cutoff of 50 and found a FDR ranging from 20-50% depending on the G-C content of the sequence [19].

Previous motif grouping efforts largely consist of the Rfam ncRNA database [9] and Infernal software package [14]. Families of motifs are identified by a single CM. Instances of a given motif are identified when scoring that CM against a given sequence exceeds the gathering cutoff [9].

2.3 Motivation

In previous vertebrate scans for RNA motifs, `CMfinder` was run with individual alignment blocks as input. This means that motifs discovered would be limited to those within a single alignment block, as no inputs cross the border of a alignment block.

In this particular investigation, we wanted to investigate specific areas of the genome in greater detail. Because they are much smaller than previous search spaces, we could also afford the computational power to examine them in greater detail than in the previous large ncRNA scans.

Because of the sliding-window approach we used (more accurately sliding-alignment-block), we were also interested in evaluating the differences between similar motifs found in overlapping windows.

The hopes of this investigation were that we might reproduce some results of the previous vertebrate scan. We also hoped to find new ncRNA motifs that extend upon those previously found, or were previously missed due to crossing block boundaries.

3 Methods

Primarily, we extended the approach of the previous `CMfinder` vertebrate scan by considering synteny blocks in our multi-way alignment. We also performed analysis of the quality scores of our results, and finally explored the overlapping distinct motifs generated by our synteny block approach.

3.1 Block Merging

To perform block merging in support of finding functional ncRNA segments, we developed a Java executable `blockmerge.jar` and will describe its functionality as follows.

Multi-way alignment software such as MULTIZ produces a Multiple Alignment Format (MAF) file, which describes groups of aligned sequences (alignment block), their coordinates in the species genome that they originate from, and any gaps in between each adjacent alignment block with reference to some reference species.

It is possible for sequences from the same species in adjacent alignment blocks to have gaps in between them, to be on opposite strands from each other and thus inverted, to be out of order, or to be on entirely different or unknown chromosomes. A set of contiguous alignment blocks that have none of these discontinuities are known as “syntenic blocks” [3].

In the large MULTIZ multi-way alignments that we use, it is possible for no adjacent alignment blocks to be completely syntenic due to a small number of sequences being discontinuous. In fact, MULTIZ already takes this into account to produce alignment blocks that are as large as possible. However, not all of the species used in the alignment necessarily contain a motif, so motif-containing sequences may be inadvertently separated into separate alignment blocks. Not all of the species of a 100-way alignment are significant or helpful when searching for ncRNA that exists in human DNA; distant species may not even have homologous ncRNA.

Furthermore, some species have sequences that have been mapped to “scaffolds” instead of full chromosomes, which means that there is a section of genome that the mapping is unsure about where it belongs. Again, there may be unnecessary breaks between alignment blocks because of this scaffold notation, as different scaffolds might actually belong to the same chromosome.

Therefore, we started by establishing a set of rules for defining syntenic blocks that were relevant for finding functional ncRNA. These rules apply once for each species in a pair of alignment blocks, and then only the species that fit all the requirements across all of the blocks to be merged are included in the resulting merged block. For clarity, the species being tested for merging is called S, the first alignment block is called A, and the second alignment block is called B.

1. As a sanity check, check that S is in both blocks A and B.
2. Check that S is on the same chromosome in both blocks A and B. Alternatively, it can be on a scaffold in either or both blocks A and B.
3. Check that S is on the same strand in both blocks A and B.
4. Check that the sections of S in A and the sections of S in B are not disordered. B must be down-strand of A.
5. Check that the gap between the sequence from S in A and the sequence from S in B does not exceed some given threshold.

We also developed two different approaches to perform window-shifting on alignment blocks that have their individual advantages.

The first approach (called `block` by the program) takes in a parameter `numBlocksPerOutput` and produces merged blocks that each contain a fixed number of original blocks, with sequences chosen according to the listed synteny rules. For example, if we merge 5 blocks per output, and an input MAF file has 1000 individual alignment blocks, then we will output 995 output files which each contain the merged sequences from contiguous sets of 5 alignment blocks at a time. The Nth output file will start at the Nth original alignment block.

The advantage of this method is that it is very simple and produces an even overlap of all of the regions of the alignment. Overlapping CMs found by using `block` merges are more evenly distributed and more accurately reflect potential CM variation within an area.

However, often the multiple alignment contains several alignment blocks in a row that are extremely short. When this happens, `block` ends up producing extremely short alignment blocks, which can be as short as 10 bases. This is much shorter than the lengths we expect CMs to be!

The second approach (called `fillblock` by the program) addresses this issue by using a length range instead. It takes in two parameters `minOutputLength` and `maxOutputLength` and produces merged blocks where the reference species sequence length is within the threshold values. Each output merged block can contain any number of original blocks.

`fillblock` starts by adding blocks to the list of blocks-to-merge until the reference is at least as long as the threshold minimum sequence length. `fillblock` then alternates between merging that list and adding the next block to the list until the reference exceeds the threshold maximum. When the reference exceeds maximum, blocks are removed from the beginning of the list until the reference is within the expected range again and we return to merging. When there are no more new alignment blocks to be added, we remove blocks from the beginning of the list and merge until the reference is under the minimum threshold again.

The `fillblock` approach addresses the previous issue of extremely short alignment blocks: in the 100-way alignment especially, there are often many alignment blocks as short as 4 bases clustered together. As a result, the `block` approach can create extremely short merged blocks, which we want to avoid. Because `fillblock` focuses on output sequence length instead of capping the number of blocks used, it avoids this issue.

However, the downside of `fillblock` is that it often produces output blocks that entirely contain each other and there is much more overlapping between merges than in `block`.

3.2 Score Quality

To evaluate the quality of each motif output by CMfinder, we scored them using RNAphylo [24, 22]. We filtered out motifs with an RNA-partition score less than 50 for inclusion in the final track display, based on the threshold used in a previous study [19].

We were interested in evaluating the quality of this score threshold. To do this, we ran a control test using randomized multiple sequence alignments.

Randomized multiple sequence alignments were generated with SISSIz [6] based on the already-merged alignment blocks. SISSIz roughly conserves sequence diversity and maintains local conservation and gap patterns but because it regenerates sequences, ncRNA structures are destroyed [6]. This provides a good null model for us to compare against.

To evaluate the quality of the score threshold, we randomized the merged blocks and used them as input to CMfinder. Resulting found motifs were then each scored through the same pipeline as the non-randomized motifs. All resulting scores of both pipelines were graphed.

3.3 Comparing Motifs

`CMfinder` produces several potential motifs from a single FASTA input, including short simple motifs and longer combined motifs. This means that even within a single input, there can be several overlapping motifs that differ slightly.

Because we use merged blocks as input to `CMfinder`, the nature of the outputs from `CMfinder` means that we get a lot of motif hits that are overlapping and may be directly redundant. Here's how we evaluated them.

3.3.1 Overlap Comparison

To do a basic evaluation and comparison of overlapping motif results, we wrote several Java executables `consensusoverlap.jar` and `pagegenerator.jar` and will describe their functionality as follows.

`pagegenerator.jar` is an executable that produces a visualization of an individual motif output produced by `CMfinder`. It converts the multiple alignment and individual quality scores associated with the motif into an easily viewable HTML page used in the track hub.

`consensusoverlap.jar` is an executable that reads all the motifs produced within a given region of the reference sequence (in this case, the human genome version hg38) and creates "clusters" of motifs based on how they overlap. A cluster of motifs starts with a single motif and is built by adding motifs not already in the group that overlap some motif in the group. The program visualizes these clusters by displaying them as consensus folding structures positioned at the appropriate coordinates in the reference genome, with the reference sequence also displayed to show how inserts and deletes are included in the consensus structure.

`consensusoverlap.jar` also creates a neat visualization of the quality scores of the motif for each species sequence used in the alignments, which can reveal patterns in how different variations of motifs might be represented in different species.

3.3.2 CM Divergence Scoring

We were also interested in grouping the motifs into clusters based on their direct similarity, regardless of position in reference sequence.

`CMfinder` basically works by performing expectation maximization on covariance models (CMs) to fit them to any motifs that the set of input sequences may contain. The output of `CMfinder` consists of several motifs, which are represented as multiple sequence alignments with a consensus folding structures [22, 25].

Instead of directly comparing the multiple sequence alignments, we compare the CMs that they each represent.

CMs are a type of model that is designed to represent a ncRNA, containing information about its structure and probability of each base being in a particular position [5]. A CM is essentially a tree of nodes, where each node represents a single unpaired base, two paired bases, or a branch in the structure. Each node also contains information about the probability of bases being inserted, deleted, or mismatched at that position [5]. A CM can be described as a probability distribution over all possible RNA sequences, because each node in the tree contains probabilities of certain bases in the sequence being emitted.

The CM that `CMfinder` discovers are first recovered from the output multiple sequence alignment by using the `cmbuild` tool from the Infernal v1.1.2 package [14]. These CM files can then be directly compared using a score system based on the idea of Kullback-Leibler divergence (K-L divergence), which is a measure of difference between two probability distributions [13].

For two discrete distributions P and Q and probabilities of emission $P(x)$ and $Q(x)$, the K-L divergence is [13]:

$$D_{\text{KL}}(P\|Q) = \sum_{x \in X} P(x) \log(\frac{P(x)}{Q(x)})$$

However, estimating the K-L divergence with a guaranteed error range is impossible for some discrete probability distributions [2], and CMs are discrete distributions. Therefore, we implemented six methods to estimate an approximate divergence score and empirically evaluated their variance.

For each of these methods, we start with a set of sampled sequences. The `cmemit` tool in Infernal samples a sequence from a given CM and the `cmalign` tool aligns the given CM with the given sequence [14].

`cmalign` outputs a bit score for each aligned CM/sequence pair [14]:

$$\text{bit score} = \log_2(\frac{P_{\text{CM}}(x)}{P_{null}(x)})$$

If we align the same sequence x to two different CMs P and Q, then we have two bit score outputs P-bitscore and Q-bitscore. The log likelihood can be computed:

$$\begin{gathered} P\text{-bitscore} - Q\text{-bitscore} \\ = \log_2(\frac{P_P(x)}{P_{null}(x)}) - \log_2(\frac{P_Q(x)}{P_{null}(x)}) \\ = \log_2(\frac{P_P(x)}{P_Q(x)}) \end{gathered}$$

The six methods we implemented use this basic log-likelihood calculation but vary in two characteristics: sampling and correction. We named each method based on the combination of sampling and correction methods used.

In the following sections we describe computing an estimate of the K-L divergence score $D_{\text{KL}}(P\|Q)$ using the CMs P and Q.

We implemented two different approaches to sampling sequences from the CMs P and Q.

1. Sample n sequences only from CM P. This approach is also referred to as “P-only” or “`Po`”.

2. Sample $\frac{n}{2}$ sequences from P and $\frac{n}{2}$ sequences from Q. This approach is also referred to as “P and Q” or “`PQ`”.

We implemented three different approaches to aggregating the log-likelihood scores calculated for each sample sequence.

1. Directly average all n log-likelihood scores. The summation is:

$$\sum_{i=1}^{n} \frac{1}{n} \log(\frac{P_P(x_i)}{P_Q(x_i)})$$

This approach is also referred to as "Average" or "AV".

2. Weight all log-likelihood scores with a rough estimate of $P_P(x)$. In this approach, we estimate with $2^{P\text{-bitscore}_i} = \frac{P_P(x_i)}{P_{null}(x_i)}$. The summation is:

$$\frac{\sum_{i=1}^{n} 2^{P\text{-bitscore}_i} \log(\frac{P_P(x_i)}{P_Q(x_i)})}{\sum_{i=1}^{n} 2^{P\text{-bitscore}_i}}$$

This approach is also referred to as "2 exponential bitscore" or "2P".

3. Weight all log-likelihood scores with a rough estimate of $P_P(x)$, adjusted to remove the $P_{null}(x)$ factor. In this approach, we estimate

$$P_{null}(x) \approx (\frac{1}{4})^{(l-L)} e^{-(\frac{l}{L})}$$

where l is the actual length of x and L is the average (expected) length of sequences emitted from the CM that x was emitted from. The summation is:

$$\frac{\sum_{i=1}^{n} 2^{P\text{-bitscore}_i} (\frac{1}{4})^{(l_i - L_i)} e^{-(\frac{l_i}{L_i})} \log(\frac{P_P(x_i)}{P_Q(x_i)})}{\sum_{i=1}^{n} 2^{P\text{-bitscore}_i} (\frac{1}{4})^{(l_i - _iL)} e^{-(\frac{l_i}{L_i})}}$$

This approach is also referred to as "2 exponential P bitscore adjusted by N" or "PN".

Each method is a combination of a sampling method and correction method, in total creating six different divergence estimates ("PoAV", "PQAV", "Po2P", "PQ2P", "PoPN", "PQPN") for a given pair of CMs.

3.3.3 CM Clustering

To cluster all of the CMs found, we can score each CM against every other CM. This gives us a score matrix M where

$$M_{ij} \approx D_{\text{KL}}(\text{CM}_i \| \text{CM}_j)$$

To get a symmetrical distance matrix S, we can add $S = D + D^T$ so that

$$S_{ij} = D_{ij} + D_{ji}$$

$$\approx D_{\text{KL}}(\text{CM}_i \| \text{CM}_j) + D_{\text{KL}}(\text{CM}_j \| \text{CM}_i)$$

We use Spectral Clustering [27] method of clustering graph nodes (as implemented in Scikit-learn [16]), which requires an affinity matrix for the nodes of the graph to be clustered. In an affinity matrix, a higher value between two nodes represents that the nodes are closer together. To compute the affinity matrix A, we use the standard deviation s of matrix S and compute $A = e^{\frac{-D}{s}}$ so that

$$A_{ij} = e^{\frac{-D_{ij}}{s}}$$

We interpret our matrix as a complete graph where each node represents a CM and each edge represents the distance (or affinity) between each CM. Estimated divergence scores are converted into affinity scores, so higher edge weights mean that the two CMs are more similar and lower edge weights mean that two CMs are more different. Spectral Clustering groups every node in this graph into k clusters using the edge weights in an affinity matrix, where k is a parameter passed to the algorithm.

To generate visualizations of the distance matrix, we use a Hierarchical Clustering algorithm and group elements of the matrix by tree branch.

3.4 Pipeline

The overall pipeline used to process input for CMfinder and evaluate results will now be described.

We started by downloading the MULTIZ 100-way alignment MAF files for these regions we were interested in:

chr1:	149,844,498	-	149,849,024
chr2:	218,255,319	-	218,257,366
chr5:	140,072,857	-	140,108,630
chr12:	62,602,752	-	62,622,213

Alignment blocks for each of these regions were merged with `blockmerge.jar`, producing sets of FASTA files representing each merged block.

Each FASTA file was passed into `CMfinder` to find motifs, and output candidate motifs in Stockholm format were scored by `RNAPhylo` (also known as `pscore`) [24].

Candidate motifs were filtered by score and assembled into `.bed` and `.bb` (big BED) formats [10] for inclusion in the track hub display using `trackgenerator.jar` and `rerunTracksForDirs.sh`.

To evaluate the null models and score threshold, merged blocks were randomly regenerated using SISSIz to break any structured ncRNA, then passed through the same CMfinder pipeline. The scores of all the null-model candidate motifs were collected using `listScores.sh`. Graphs were generated in Python, the code for which is available as a Jupyter notebook `thresholding.ipynb`.

We wrote `KLDivergenceEstimate.sh` to generate individual divergence estimate scores between each pair of input CMs. To perform CM divergence scoring across an entire CM input set, we use `klmatrixgenerator.jar` in combination with this script to assemble the entire distance matrix. Clustering is then performed using the `scikit-learn` package in Python [16]. This code is available as a Jupyter notebook `cmcluster.ipynb`.

4 Results

We will now discuss some of the results that we obtained in this investigation.

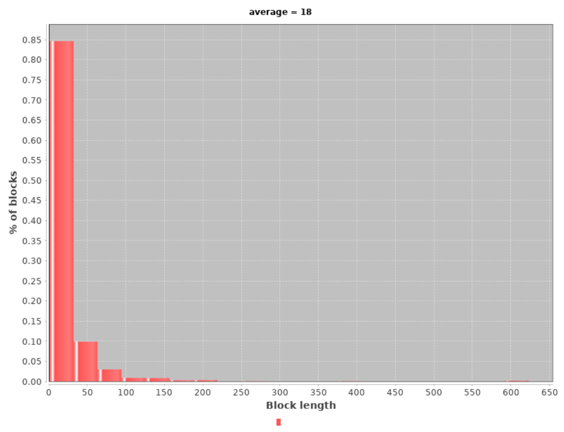
All of these results can also be viewed using a track hub [17] in the UCSC genome browser [11]: `http://students.washington.edu/jyzhou15/trackHub/hub.txt`

All graphics are also uploaded for viewing at: `http://students.washington.edu/jyzhou15/trackHub/`

4.1 Merged-block Motifs

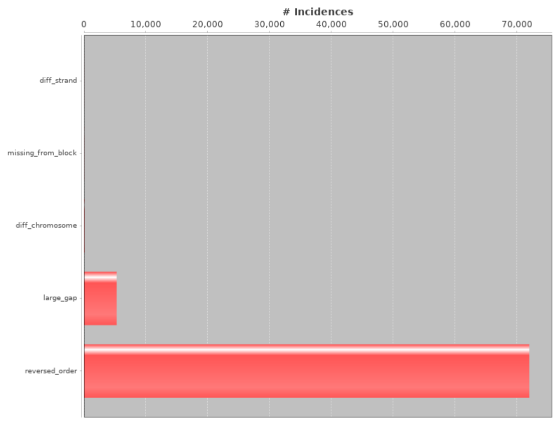
We start by examining the process and results of merging alignment blocks. In particular, we compare the results of merging using a 10-block `block` merge vs. a 75-400 base length `fillblock` merge. All of the results here were taken from merge results from chromosome 5 in the MULTIZ 100-way alignment, as this is the longest region of interest.

There were 1981 total alignment blocks in our region of interest. The input alignment blocks were generally very short, with almost 85% of them being under 30 bases long. They ranged from 1 base to 600 bases in length. The average reference sequence length for these alignment blocks was 18 bases, which is much shorter than many ncRNAs. It is worth noting that these blocks are almost definitely shorter than MULTIZ 17-way alignment blocks used in the previous vertebrate scan, as they include more species and fragment more often.



The `block` (N-block) merge method with N=10 was used on this input set of alignment blocks, producing 1971 merged blocks. We tracked how often each species contained two adjacent sequences that were not mergeable and which merging rules they broke. We also examined the merge block lengths,

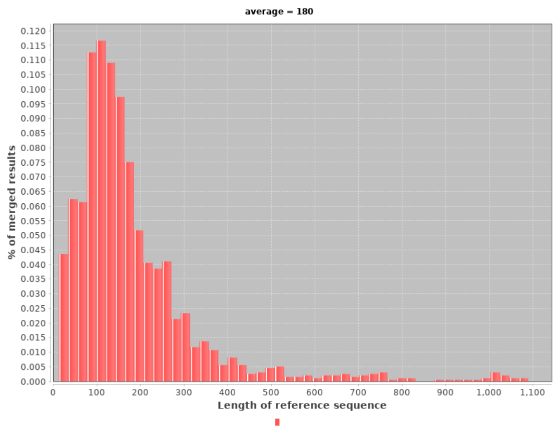
Most species used in the alignment had at least one junction where they could not be merged. In general, species more distant from human such as birds and fish had more merge failures. Species evolutionarily closer to human such as “ponAbe2” (oranguetan) “panTro4” (chimp) had very few merge failures. Interestingly, “gadMor1” (Atlantic cod) also had unusually few merge failures. The most common reason by far for merge failures was because sequences were out of order, although large gaps in between each block were also a major cause.



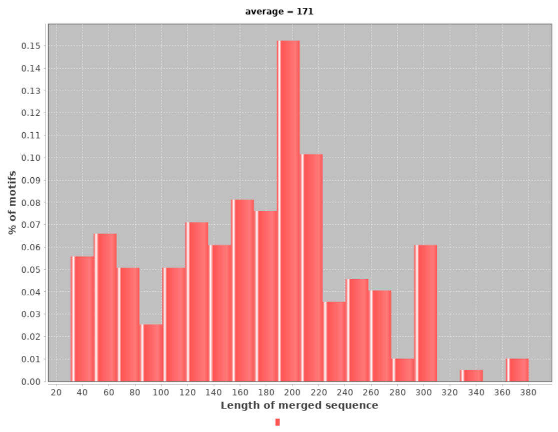
Note that the causes are sorted in order from least to most common, from top to bottom.

Refer to appendix A.1 for a graph on disjoint species with detail.

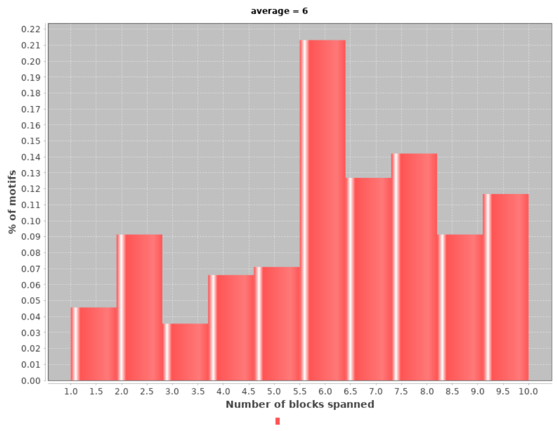
10-block merged alignment blocks varied widely in length from fewer than 10 bases to over 1000 bases, averaging 180 bases in length. Here the downsides of n-block merging are apparent, with both extremely short merged sequences and extremely long sequences that are unlikely to be helpful for finding ncRNA.



In fact, the sequences in which motifs were found ranged from 30 to 400 bases in length. The average reference sequence length in all motifs found was 171 bases when we filter out motifs that do not include human DNA and motifs that score under 50 on the RNAphylo RNA-posterior scoring.



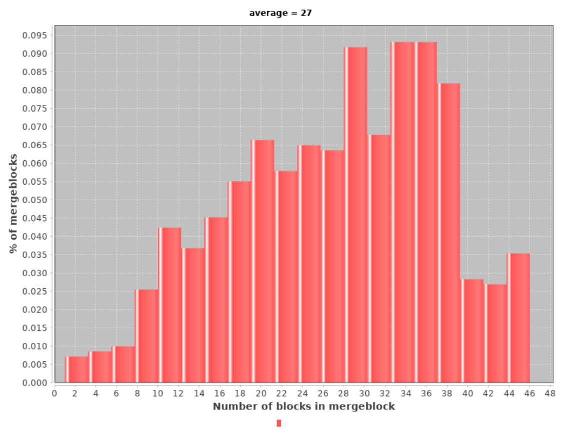
Most post-filter motifs spanned over 5 alignment blocks, showing that merging blocks is effective.



For a graph pertaining to species composition in the motifs found, refer to appendix A.2.

We also used the `fillblock` merge method (with a reference sequence length range of 75-400 bases) on the same region, producing 709 merged blocks.

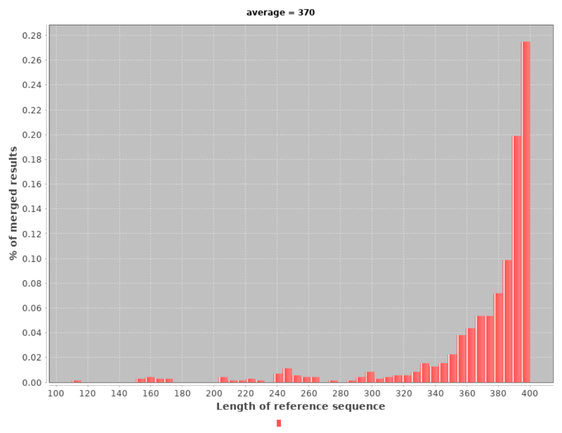
As expected, `fillblock` uses a varying number of alignment blocks in each merge, but stays within the reference sequence length threshold for all results. Output merged blocks contained anywhere from 2 to 46 original alignment blocks, with an average of 27 alignment blocks in each output. This indicates that we are making use of more short alignment blocks, which N-block merging tends to exclude.



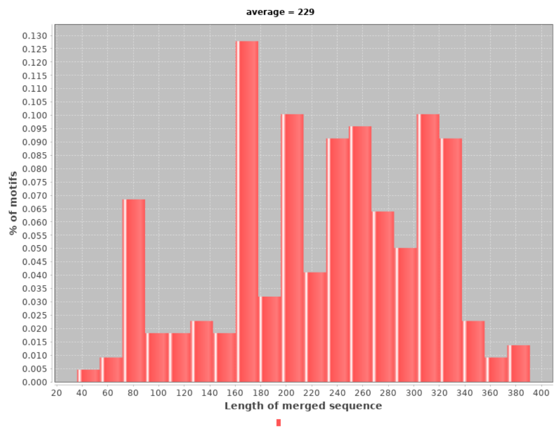
The species representation in `fillblock` output is approximately the same as in N-block merges, but with significantly fewer overall failed attempts. This is because there were fewer overall attempts at merging, as many attempts exceeded the overall length limit.

For detail about `fillblock` merge failures across species and across causes refer to appendices A.3 and A.4.

In contrast to the 10-block merge, which had a very wide range of merge lengths, `fillblock` succeeds in merging sequences that have lengths close to the upper threshold. Over 50% of merged block outputs have a reference sequence with a length of 375-400 bases, which is right under the upper length threshold of 400 bases.



Motifs found using merged alignments from `fillblock` tended to come from longer sequences, averaging 229 bases. The species distribution is very similar to that of the 10-block merge. The average number of blocks that a motif spans is 16, which is unsurprising given that the average number of blocks per output merge is also higher.

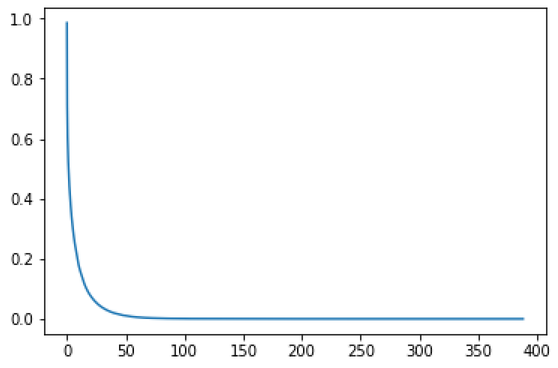


For more detailed graphs refer to appendices A.5 and A.6.

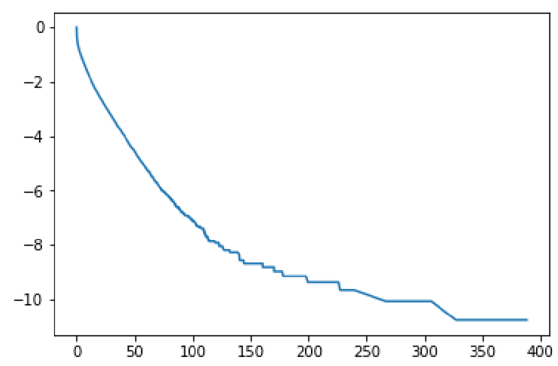
Overall, we found that block merging was successful at allowing us to discover motifs separated across multiple alignment blocks.

4.2 Score Threshold

We graphed the false-positive rate as score threshold increases. This graph includes data with all variations of block-merging configurations and in all regions of the genome we investigated.



The threshold that we used has a very low false-positive rate, under 1%. This is the same data on a natural logarithmic scale:



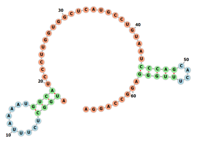
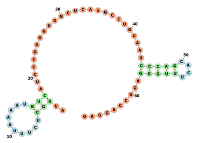
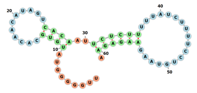
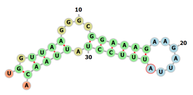
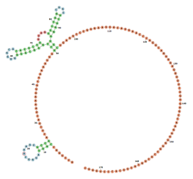
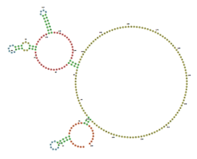
The number of motifs with scores higher than 200 was relatively few, which is why the false positive rate starts appearing discontinuous at higher score thresholds. It is also worth noting that there were about 1.5 times as many motif results in the original data than in the null model data.

4.3 CM Clustering

We clustered a small group of motifs taken from the chromosome 5 `fillblock` results. These motifs were handpicked to evaluate the clustering effectiveness. We used the `PQ2P` score estimate for all clustering, as this score was extremely consistent with `Po2P` and had both a relatively low empirical variance and high range. Scores ranged from below 5 to over 200.

First we examine the hand-picked group of motifs itself. Motifs in this set were specifically chosen to represent a wide range of ncRNA secondary structures and to be easily grouped by hand. Two of these motifs were near-identical, only differing in which merged alignment block and species sequences they were found with. All of these motifs are referred to here according to the number of the merged block they originate from, and each can be directly viewed on the track hub and cluster overlap reference pages.

These were the motifs chosen (visualized using Forna [12]), the clusterings I predicted they would fall into, and how they were actually clustered with $N = 2$ and $N = 3$ clusters:

#	Shape	Pre	$N = 2$	$N = 3$
“288”:		A	A	A
“290”:		A	A	A
“511”:		A	B	B
“516”:		B	B	B
“278”:		C	B	C
“544”:		C	B	B

For a visualization of the distance matrix, refer to appendix B.1

Our approach for clustering can produce unintuitive results for more ambigious groupings such as 278 and 544, which have moderately similar structures but are grouped separately with higher N. However, this does confirm that this clustering method works well for similar motifs such as 288 and 290.

However, a big downside of this clustering approach is that it runs in $\mathcal{O}(n^2)$ time with a very high constant factor, so it gets very computationally expensive for a larger set of motifs.

Furthermore, because Spectral Clustering takes the number of clusters expected as a constant input, there may be deceptive groupings if the number of clusters is incorrect. Spectral Clustering does not provide any form of dendrogram for its clusters, so we are unable to easily modify the number of clusters after they have already been made.

5 Conclusion

`blockmerger` allows us to adapt larger multiple alignments with shorter alignment blocks for use in applications that may require extended input, such as but not limited to CMfinder and other ncRNA scanning algorithms. Using `blockmerger`, we found many promising ncRNA candidates.

K-L divergence estimation methods allow us to summarize the similarity of covariance models and visualize their relationships differently from simple family groupings. Using `klmatrixgenerator`, we created distance matrices for CMs themselves and clustered them.

5.1 Future Steps

One method of merging blocks that we did not implement due to complexity was including a fixed number of bases in each merged block and shifting the merging window by a fixed number of bases on each merge. Given more time, it might be helpful to see how further controlling the input sequence length affects the distribution of motifs found. This would also allow us to control both motif overlap and merge sequence length at the same time, which neither N-`block` nor `fillblock` does.

Another improvement that could be made to `blockmerger` is to recognize different species genomes when inserting bases between alignment blocks. Many alignment blocks have several unknown bases between known sequences contained in the alignment, and currently `blockmerger` acknowledges them as "N" (unknown) bases. This may affect the quality of our results, as many programs (including CMfinder and SISSIz) interpret "N"s randomly.

It would be beneficial to compare all of the motifs we found against Rfam (and therefore previous ncRNA scans) to check if we found novel ncRNA motifs or expanded upon previously known motifs. The track hub already allows for a cursory comparison, however, having more detailed information about how our results match up is always helpful.

It might also be beneficial to explore a wider range of parameters for CMfinder: for this investigation, we used the same parameters as the previous vertebrate scan. However, this does not take into account potential differences in input length.

We were also curious why the `PN` (null-probability-corrected) scoring system had an unusually high variance across multiple runs on the scale of 40-50 for both sampling methods. We predicted that this system would have less variance than `2P`, but this was not the case. This disparity is worth looking into more thoroughly. In general, further evaluation of divergence scoring should be done. There may be more complex but theoretically accurate estimators in literature which are applicable [2].

In regards to clustering covariance models, it would be very interesting to perform a more thorough test of the system using Rfam's already-known families of ncRNA as a quality measure. The current quality test of clustering is entirely based on individual judgement and is not very good, while Rfam performs its motif grouping based on a separate scoring and ranking system.

Finally, it would be worthwhile to explore different methods of clustering that do not have the same weaknesses as Spectral Clustering. For instance, Hierarchical Agglomerative Clustering may allow more control over individual cluster counts, as well as provide insight into which clusters are closer together.

References

[1] Arnold J. Berk. "Discovery of RNA splicing and genes in pieces". In: *Proceedings of the National Academy of Sciences* 113 (4 Jan. 2016), pp. 801–805. DOI: `10.1073/pnas.1525084113`.

[2] Yuheng Bu et al. “Estimation of KL Divergence: Optimal Minimax Rate”. In: *IEEE Transactions on Information Theory* 64 (4 Feb. 2018), pp. 2648–2674. DOI: 10.1109/TIT.2018.2805844.

[3] Julien Y. Dutheil, Sylvain Gaillard, and Eva H. Stukenbrock. “MafFilter: a highly flexible and extensible multiple genome alignment files processor”. In: *BMC Genomics* 15 (1 Jan. 2014), p. 53. DOI: 10.1186/1471-2164-15-53.

[4] Sean R. Eddy. “Non-coding RNA genes and the modern RNA world”. In: *Nature Review Genetics* 2 (Dec. 2001), pp. 919–929. DOI: 10.1038/35103511.

[5] Sean R. Eddy and Richard Durbin. “RNA sequence analysis using covariance models”. In: *Nucleic Acids Research* 22 (11 June 1994), pp. 2079–2088. DOI: 10.1093/nar/22.11.2079.

[6] Tanja Gesell and Stefan Washietl. “Dinucleotide controlled null models for comparative RNA gene prediction”. In: *BMC Bioinformatics* 9 (1 May 2008), p. 248. DOI: 10.1186/1471-2105-9-248.

[7] Jan Gorodkin et al. “*De novo* prediction of structured RNAs from genomic sequences”. In: *Trends in Biotechnology* 28 (1 Nov. 2009), pp. 9–19. DOI: 10.1016/j.tibtech.2009.09.006.

[8] Andreas R. Gruber et al. “RNAz 2.0: Improved Noncoding RNA Detection”. In: *Biocomputing* (2010), pp. 69–79. DOI: 10.1142/9789814295291_0009.

[9] Ioanna Kalvari et al. “Rfam 13.0: shifting to a genome-centric resource for noncoding RNA families”. In: *Nucleic Acids Research* 46 (D1 Nov. 2017), pp. D335–D342. DOI: 10.1093/nar/gkx1038.

[10] W. James Kent et al. “BigWig and BigBed: enabling browsing of large distributed datasets”. In: *Bioinformatics* 26 (17 Sept. 2010), pp. 2204–2207. DOI: 10.1093/bioinformatics/btq351.

[11] W. James Kent et al. “The Human Genome Browser at UCSC”. In: *Genome Research* 12 (6 May 2002), pp. 996–1006. DOI: 10.1101/gr.229102.

[12] Peter Kerpedjiev, Stefan Hammer, and Ivo L. Hofacker. “Forna (force-directed RNA): Simple and effective online RNA secondary structure diagrams”. In: *Bioinformatics* 31 (20 Oct. 2015), pp. 3377–3379. DOI: 10.1093/bioinformatics/btv372.

[13] S. Kullback and R. A. Leibler. “On Information and Sufficiency”. In: *The Annals of Mathematical Statistics* 11 (1 1951), pp. 79–86. DOI: 10.1214/aoms/1177729694.

[14] Eric P. Nawrocki and Sean R. Eddy. “Infernal 1.1: 100-fold Faster RNA Homology Searches”. In: *Bioinformatics* 29 (22 Nov. 2013), pp. 2933–2935. DOI: 10.1093/bioinformatics/btt509.

[15] Jakob Skou Pedersen et al. “Identification and Classification of Conserved RNA Secondary Structures in the Human Genome”. In: *PLoS Computational Biology* 2 (4 Apr. 2006), e33. DOI: 10.1371/journal.pcbi.0020033.

[16] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[17] Brian J. Raney et al. “Track data hubs enable visualization of user-defined genome-wide annotations on the UCSC Genome Browser”. In: *Bioinformatics* 30 (7 Apr. 2014), pp. 1003–1005. DOI: 10.1093/bioinformatics/btt637.

[18] Elena Rivas and Sean R. Eddy. “Noncoding RNA gene detection using comparative sequence analysis”. In: *BMC Bioinformatics* 2 (8 Oct. 2001). DOI: 10.1186/1471-2105-2-8.

[19] Stefan E. Seemann et al. “The identification and functional annotation of RNA structures conserved in vertebrates”. In: *Genome Research* 27 (8 May 2017), pp. 1371–1383. DOI: 10.1101/gr.208652.116.

[20] Gary D. Stormo. “Gene-Finding Approaches for Eukaryotes”. In: *Genome Research* 10 (4 Apr. 2000), pp. 394–397. DOI: 10.1101/gr.10.4.394.

[21] Daniel Sundfeld et al. “Foldalign 2.5: multithreaded implementation for pairwise structural RNA alignment”. In: *Bioinformatics* 32 (8 Apr. 2016), pp. 1238–1240. DOI: `10.1093/bioinformatics/btv748`.

[22] Zasha Weinberg. *CMfinder package 0.4.1.15 user manual*. Sept. 2018. URL: `https://downloads.sourceforge.net/project/weinberg-cmfinder/cmfinder-0.4.1.15-user-manual.pdf`.

[23] Zasha Weinberg et al. “Identification of 22 candidate structured RNAs in bacteria using the CMfinder comparative genomics pipeline”. In: *Nucleic Acids Research* 35 (14 June 2007), pp. 4809–4819. DOI: `10.1093/nar/gkm487`.

[24] Zizhen Yao. “Genome scale search of noncoding RNAs: Bacteria to Vertebrates”. PhD thesis. University of Washington, 2008.

[25] Zizhen Yao, Zasha Weinberg, and Walter L. Ruzzo. “CMfinder - a covariance model based RNA motif finding algorithm”. In: *Bioinformatics* 22 (4 Feb. 2006), pp. 445–452. DOI: `10.1093/bioinformatics/btk008`.

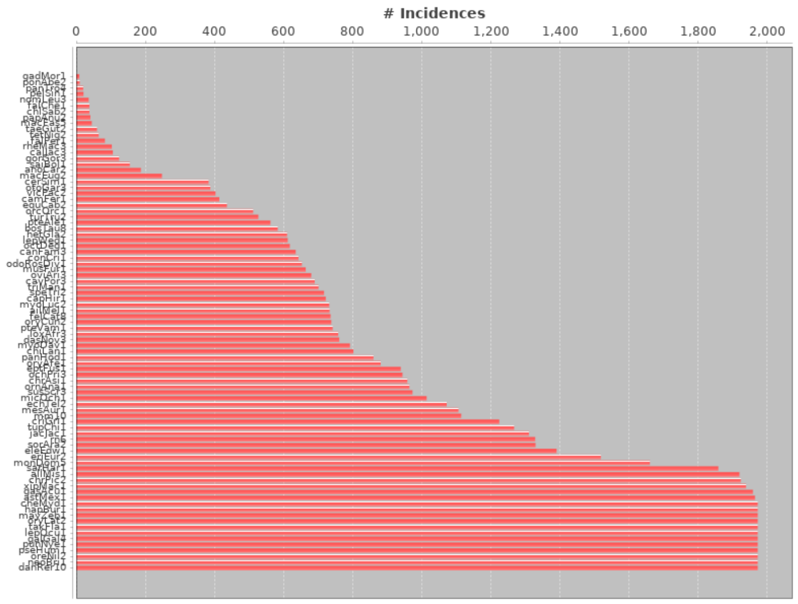
[26] Zizhen Yao et al. “A Computational Pipeline for High-Throughput Discovery of *cis*-Regulatory Noncoding RNA in Prokaryotes”. In: *PLoS Computational Biology* 3 (7 July 2007), e126. DOI: `10.1371/journal.pcbi.0030126`.

[27] Stella X. Yu and Jianbo Shi. “Multiclass Spectral Clustering”. In: *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*. ICCV ’03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 313–. ISBN: 0-7695-1950-4. URL: `http://dl.acm.org/citation.cfm?id=946247.946658`.

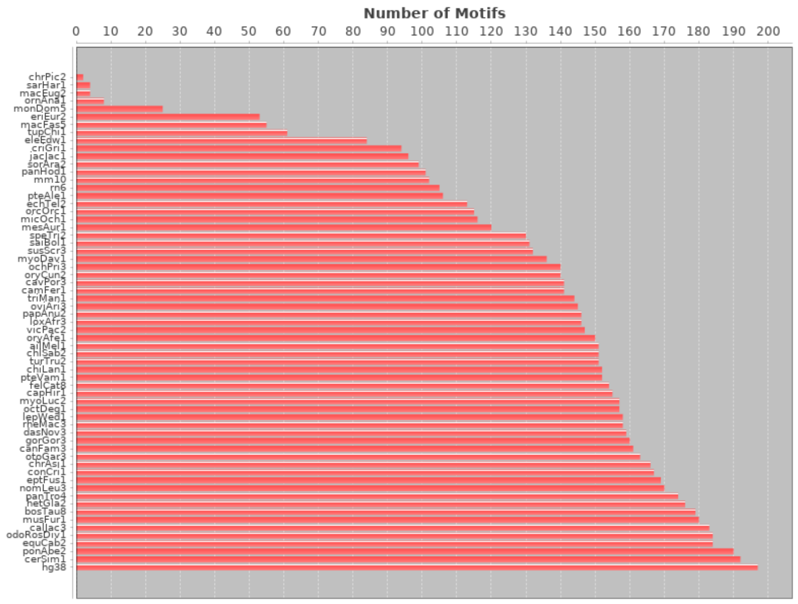
Appendix

A Block-merge graphs (all on chr5)

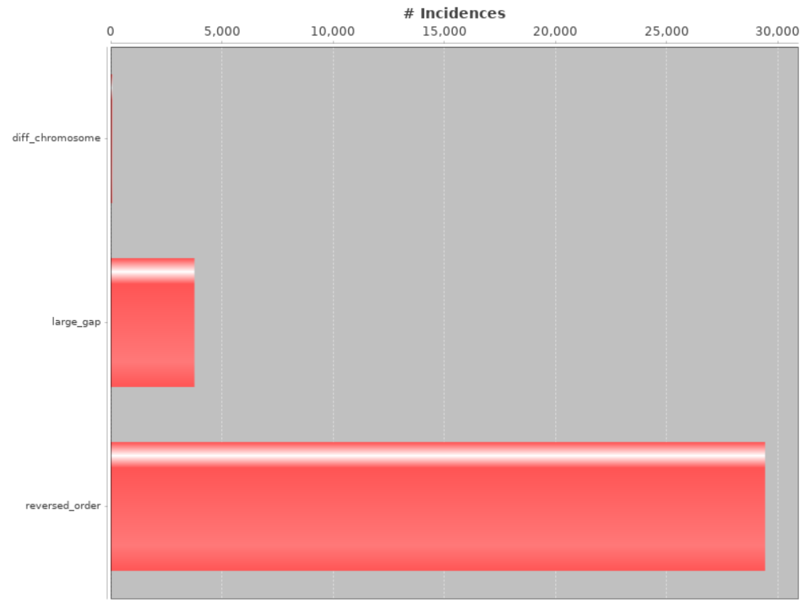
A.1 Breakdown of instances when merging two block sequences failed across species for a 10-block merge



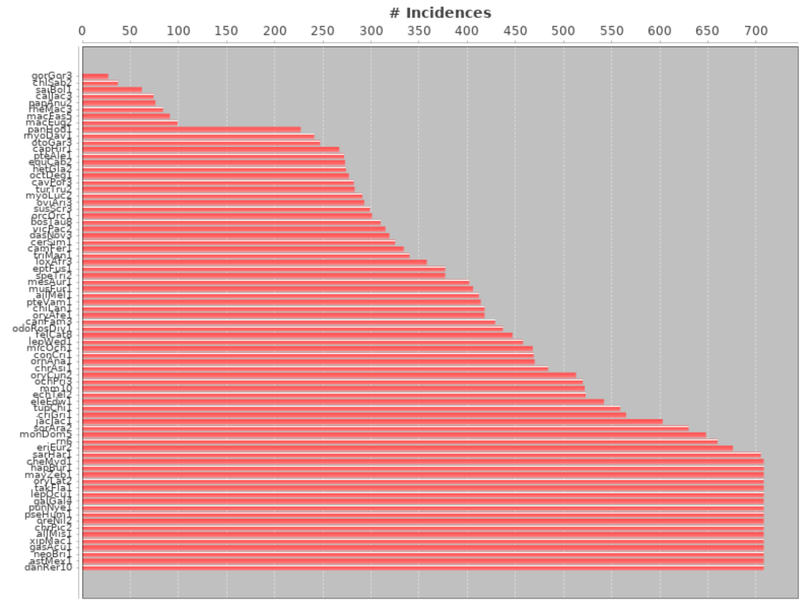
A.2 Species that post-filter motifs were found in for a 10-block merge



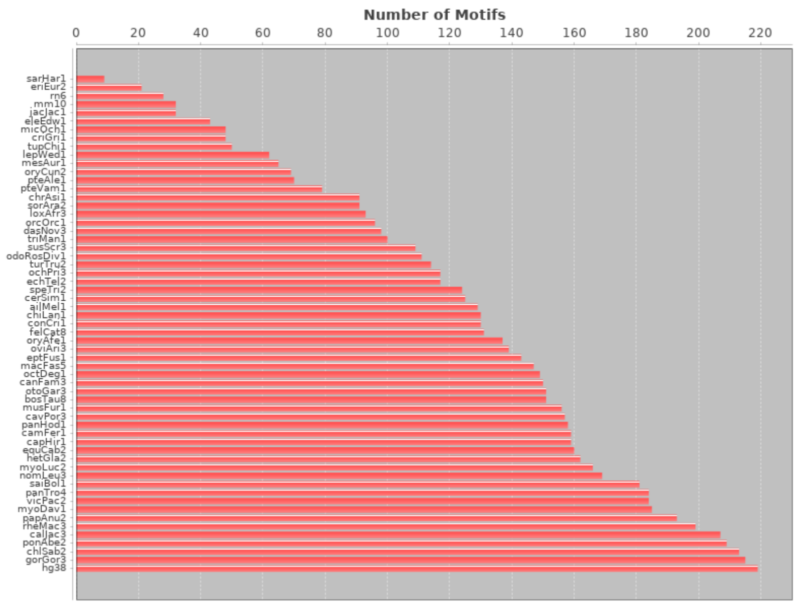
A.3 Breakdown of instances when merging two block sequences failed across causes for a `fillblock` merge



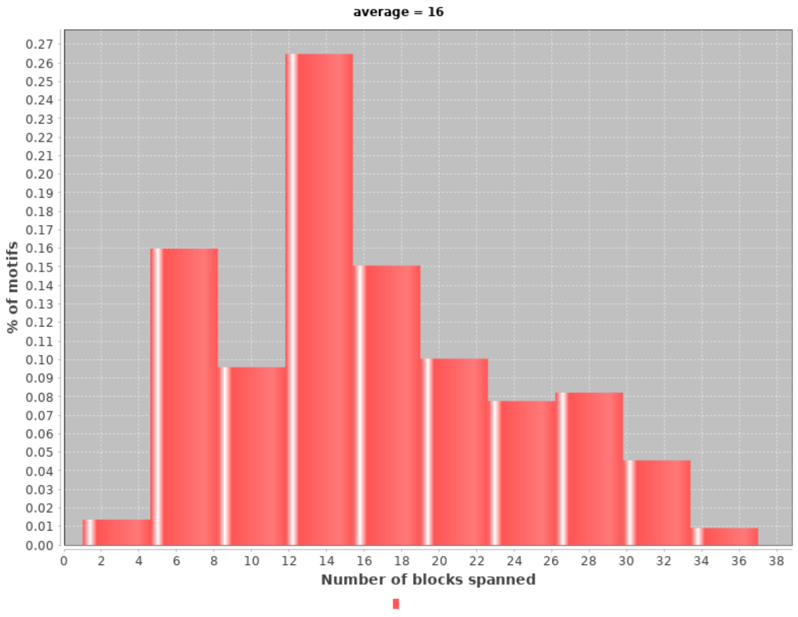
A.4 Breakdown of instances when merging two block sequences failed across species for a `fillblock` merge



A.5 Species that post-filter motifs were found in for a `fillblock` merge

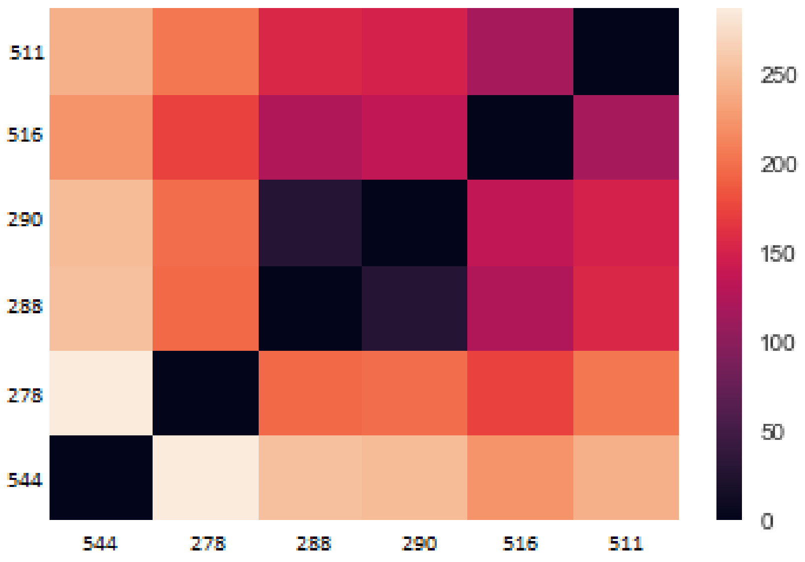


A.6 Number of blocks that post-filter motifs spanned for a `fillblock` merge



B CM clustering graphs (all from chr5, `fillblock`)

B.1 Distance matrix for the chosen subset of CMs


511
516
290
288
278
544
544
278
288
290
516
511
250
200
150
100
50
0