2022

SUBFAMILY CLUSTERING USING LABEL UNCERTAINTY (FOR TRANSPOSABLE ELEMENT FAMILIES)

Audrey M. Shingleton
University of Montana, Missoula

SUBFAMILY CLUSTERING USING LABEL UNCERTAINTY (FOR TRANSPOSABLE ELEMENT FAMILIES)

By

Audrey Shingleton

Bachelor of Science, The University of Washington, Seattle, WA, 2020

Thesis

presented in partial fulfillment of the requirements
for the degree of

Master of Science
in Computer Science

The University of Montana
Missoula, MT

Spring 2022

Approved by:

Ashby Kinch Ph.D., Dean
Graduate School

Travis Wheeler Ph.D., Chair
Computer Science

Jesse Johnson Ph.D.
Computer Science

Robert Hubley, Senior Software Engineer
Institute for Systems Biology, Seattle, WA

Shingleton, Audrey M.S., May 2022 Computer Science

SCULU: Subfamily Clustering Using Label Uncertainty (for transposable element families)

Chairperson: Travis Wheeler

Biological sequence annotation is typically performed by aligning a sequence to a database of known sequence elements. For transposable elements, these known sequences represent subfamily consensus sequences. When many of the subfamily models in the database are highly similar to each other, a sequence belonging to one subfamily can easily be mistaken as belonging to another, causing non-reproducible subfamily annotation. Because annotation with subfamilies is expected to give some amount of insight into a sequence's evolutionary history, it is important that such annotation be reproducible. Here, we present our software tool, SCULU, which builds upon our previously-described methods for computing annotation confidence, and uses those confidence estimates to find and collapse pairs of subfamilies that have a high risk of annotation collision. The result is a reduced set of subfamilies, with increased expected subfamily annotation reliability.

ACKNOWLEDGMENTS

I'd like to start by thanking my advisor, Travis Wheeler, for his immense amount of support and guidance during my time in the Wheeler lab. His time spent helping to develop methods, edit and revise papers, and thoughtfully answer questions was invaluable. I'd also like to thank George Lesica for helping to develop my skills in software development, my professors for continuing to further my interest in computer science, as well as friends, family and fellow lab members for their support. Finally, I'd like to express my gratitude for the NIH grant U24 HG010136 (NHGRI) that funded my research.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 1 INTRODUCTION

Bioinformatics background

Genomes are comprised of genetic material that functions as instructions for how an organism will grow and behave. It is stored as a string of nucleotides (DNA), with an alphabet of the letters 'A', 'C', 'T', and 'G'. These strings can be of varying lengths. For instance the human genome is approximately 3 billion nucleotides long, whereas small viral genomes may only be a few thousand nucleotides in length. Understanding the genetic material of an organism gives insight into evolutionary history. Organisms that are related will have DNA sequences with more similarity than those less related. An important part of gaining this insight relies on recognizing features in the genome. This process is known as genome annotation.

Genome annotation assigns labels to nucleotide sequences. It is typically based on comparing these sequences to a database of known sequence elements in a process called sequence alignment. Scores are assigned to the aligned sequences, corresponding to the strength of the match, and labels are assigned to the highest scoring matches.

TE subfamilies

Transposable elements (TEs) are mobile genetic elements that often replicate, leaving behind an extensive trail of copies that result in wide-spread interspersed repetitive regions throughout a genome (for reviews, see [1, 2]). For the purposes of genome annotation, the remnant copies found in a genome are typically organized into *families* within a library of TE families, such as Repbase [3] or Dfam [4]. In such a library, each family represents a collection of instances resulting from a distinct history. A family history may involve gradual mutation of an active ("master") TE such that copies are all slightly different from each other at their time of creation; such familes are often represented by a single element in a TE library, which captures a mixture of ancestral forms.

Alternatively, a family history may be punctuated, with a master producing a large number of identical copies at each of several lineage bursts; it is common for a library to break such a family into *subfamilies*, with each representing the remnants of such a replication burst.

The task of splitting family instances into subfamilies often boils down to clustering sequences into groups such that there is high similarity within a group, and differentiation between groups. This often entails identification of shared co-segregating mutations [5, 6, 7], but can also involve reconstruction of ancestral relationships [8], inference of similarity networks [9], or tree-based entropy measures [10].

Because annotation with subfamilies is often presumed to provide a sense of a sequence's historical context, it is important that such annotation be reproducible. We recently evaluated the reliability of genome annotation with common subfamilies using RepeatMasker and the currently-available Repbase library of TE family/sub-family consensus sequences for human TEs. To assess reliability, we considered natural replicate copies (i) resulting from segmental duplications in the human genome, and (ii) based on homologous copies shared by human and chimpanzeečitediscordance. We found that more than 10% of replicates found in segmental duplications are annotated as belonging to different subfamilies, both in young families (Alu) and old (MIR, L1). A similar level of discordant classification was observed in homologous TE instances shared by human and chimp.

One cause of high rates of non-reproducible subfamily annotation is that subfamily models are so similar to each other that their matching sets are highly intersected: a sequence properly belonging to one subfamily can easily be mistaken as belonging to another. When subfamily models are particularly similar, it may take only a few random mutations to flip adjudication preference from one subfamily to the other. We found that nearly half of all problematic Alu annotations fall into this category (the others may be due to forces such as recombination and homologous gene conversion).

Here, we present a new method designed to increase subfamily annotation reliability, implemented in software called SCULU. As input, SCULU is provided with consensus sequences for a collection of subfamilies belonging to a single primary family, along with a set of instances for each

subfamily. It identifies subfamilies with high risk for annotation collision, and merges them; the result is a reduced set of subfamilies, with increased expected subfamily annotation reliability. We anticipate that SCULU will prove useful as a post-processing step in largely-automated TE curation pipelines, but it also supports downstream manual curation of subfamilies by producing all subfamily merging information, allowing a manual curator to override some automated decisions.

SCULU's approach is based on our previously-published method for computing annotation confidence in the face of multiple competing annotations [11, 12], and it is released as open source software at `https://github.com/TravisWheelerLab/sculu`. We demonstrate the utility of SCULU by applying it to (i) human Alu subfamilies and to (ii) members of the metulj family found in multiple Heliconius (butterfly) genomes [13].

CHAPTER 2 METHODS

The guiding principle of SCULU is that subfamilies should be reliably separable - if a sequence properly belongs to one subfamily, then it should be very unlikely to be assigned to some other subfamily due to common chance events. SCULU achieves this goal by identifying pairs of subfamilies for which separable assignment is unreliable; it merges such pairs into a single subfamily, repeating until convergence. SCULU identifies unreliably-separable pairs empirically, by aligning instances of each subfamily to all of the subfamily consensus sequences, and computing a score-based estimate of annotation confidence to each instance. If a large number of instances indicate low confidence in separation of two subfamilies, they are merged. Below, we provide details of that process.

Confidence of subfamily assignment for a single sequence

Alignment of all sampled subfamily instances to all subfamily consensus sequences is performed using cross_match[14] which supports complexity adjusted scoring and the custom scoring matrices used to align and annotated TE instances in RepeatMasker. The cross_match tool produces an alignment score for each aligned instance-consensus pair; this score corresponds to the odds ratio of (i) the probability of observing the sequence t if it is homologous to the subfamily q_i vs (ii) the probability of observing t under a random (non-homology) model of sequence composition. Specifically, the score is a scaled logarithm of that odds ratio. Due to the fact that these alignment scores correspond to probabilities, we are able to compute a measure of confidence that a specific sequence belongs to a specific subfamily. The confidence in a particular assignment can be computed as a function of its alignment score relative to the scores of all competing candidate annotations for that sequence (see [12] for derivation).

$$\text{Conf}(q_i|t) = \frac{2^{score(t,q_i)/\lambda}}{\sum_j 2^{score(t,q_i)/\lambda}} \tag{2.1}$$

An extreme example of the consequence of this method of computing annotation confidence is that if two candidate subfamily annotations produce alignments to the target sequence with the same score, then the subfamilies will evenly split confidence (for example, if no other subfamilies produce an alignment, then the confidence for each will be 50%). The ability to compute these confidence values allows SCULU to assign a measure of confidence to each possible subfamily assignment. The result is that, for each TE family instance, instead of giving a single classification based on the subfamily with the best alignment score, SCULU can produce a listing of confidences for possible annotations from all subfamilies it might have been assigned to.

Confidence-based separability of two subfamilies

Suppose that we have a collection of TE family instances that are aligned to the consensus sequences for subfamily A and subfamily B. For each sequence in the collection, we can compute the confidence of assignment to subfamilies A and B.

One mechanism for recognizing failure to discriminate between the two subfamilies would be to identify the fraction of sequences that did not have sufficiently-strong separation between their confidences. Let $C_t(A)$ be the computed confidence that sequence t belongs to subfamily A, and $C_t(B)$ be the confidence that t belongs to subfamily B. If one of these confidence values is not higher than the other by some threshold p, then this suggests that these subfamilies may not be discernible. For example, and without loss of generality, suppose we set $k = \frac{1}{3}$ and that $C_t(A) \cdot k > C_t(B)$ (so the confidence of assignment to subfamily A is more than 3x the confidence of subfamily B) - we may say that sequence t is separable because it's assignment to one subfamily is quite confident. We say that t has a 'clear winner'. Meanwhile, some other sequence s may show that $C_s(A) \cdot k < C_s(B)$ - in this case, we say that s does not have a clear winner, and call this an 'uncertain pair'. After computing confidence values for each TE instance in our collection, if some threshold fraction of these instances do not exhibit a clear winner, then we have reason to merge subfamilies A and B.

Confidence-based separability for a collection of subfamilies

We now extend the approach to assume alignments to a collection of subfamilies. In this scenario, each TE instance will align to many subfamily consensus sequences, yielding a collection of confidence values. In order to determine if a pair of subfamilies has sufficiently strong separation, we define a threshold based on the maximum confidence value from the collection.

Suppose there are n alignments for subfamilies $(a_1, a_2, ..., a_n)$ to a TE instance t, and let $M_t = \max_{1..n} C_t(a_i)$. If no other confidence values for t are above a defined fractional threshold $p \cdot M_t$, we say that t has a clear winner. On the other hand, if some other subfamily presents a confidence greater than $p \cdot M_t$, then sequence t is said to have unclear annotation. All subfamilies exceeding this confidence threshold are added to an 'unclear winner set' for t, such that each pair of subfamilies in the set can be said to be an 'uncertain pair' for t.

Subfamily pair independence

To determine if a subfamily pair should be merged, SCULU computes a measure of the independence between the subfamilies in this pair. Let w_i be the number of times subfamily i is a clear winner over all test sequences, and let u_{ij} be the number of times subfamily i is in an uncertain pair with subfamily j. SCULU computes the independence of subfamily pair (i, j) as:

$$\text{Independence}(i, j) = \frac{w_i}{w_i + u_{ij}} \tag{2.2}$$

If subfamily i is in no uncertain pairs with subfamily j, then Independence(i, j) will be 1; conversely, if subfamily i is not a clear winner for any TE instance, and shares an uncertain pair with j on at least one instance, the Independence(i, j) will be 0.

Collapsing unreliable subfamilies

SCULU computes the pairwise independence value for each pair of uncertain subfamilies, and produces a list of pairs sorted in ascending independence value; ties are broken by uncertain pair count u_{ij} (descending). In the first merging stage, the pair with lowest independence (and in case

of ties, largest u_{ij}) is marked to-be-merged. SCULU proceeds along the sorted list, identifying all additional pairs (i', j') for which neither i' nor j' have yet been marked to-be-merged, and adding them to the to-be-merged set. Traversal of the ordered list stops when Independence(i, j) rises above a parameterized threshold. All resulting to-be-merged pairs are collapsed into a new subfamily, in a procedure described in the next paragraph. After this set is collapsed, which concludes a single iteration in SCULU, the new subfamily elements are each aligned to the full set of TE instances, then confidence and Independence values are computed for all new pairs. The process is repeated (sorting, collapsing, aligning, ...) until no pairs with below-threshold Independence remain. The result is a proposed collection of more discernible subfamilies based on these results.

Collapsing of a pair (i, j) of subfamilies is achieved by (1) performing a multiple sequence alignment (MSA) of all instances of subfamilies i and j, using the MSA tool MAFFT[15]. A consensus sequence for the subfamily is computed based on the merged MSA (see next section). Then the pair of consensus sequences for i and j is removed from the library, and replaced with the consensus sequence for the merged subfamily. The resulting consensus sequences are then handed back into the sequence alignment approach that was used to produce our collection of candidate alignments in the first place. To minimize rework, all alignments not involving subfamilies i and j are retained, all alignments involving i or j are removed, and alignments are introduced for the new subfamily.

Computing the consensus for a merged subfamily

The input to SCULU consists of two related collections: (i) a set of instances of each subfamily, and (ii) a consensus sequence for each subfamily. The consensus sequence serves as the representative of each subfamily during the process of computing annotation confidence/independence. When merging two subfamilies, SCULU must produce a consensus sequence for the new subfamily. It does this by computing an MSA of the collected instances of the two merged subfamilies, then producing a consensus based on that MSA. A simple mechanism for computing a consensus is to represent a MSA column with the most frequently occurring letter in that column. This approach can produce

biased results if the MSA contains a cluster of highly-related sequences; to overcome this, we implemented a method that uses the *hmmbuild* tool in HMMER [16] to compute a profile that applies Henikoff position-based weights [17] to overcome such bias, then produces the consensus using the tool *hmmemit*.

However, in the case of transposable element subfamilies, subfamily instances have been subjected to long-standing neutral mutations, and in particular show signs of rampant CpG deamination, in which 'CG' dinucleotide pairs mutate to 'TG' with exaggerated frequency. The result is that a naive consensus calling may place a 'T' in a column that was originally the 'C' of a 'CG' dinucleotide. To overcome this, SCULU by default produces consensus sequences using the Linup tool within RepeatMasker [7], which specifically accounts for these mutations. Because the choice of 'T' or 'C' at CpG sites will influence alignment scores in an essentially random fashion, SCULU also by default ignores all alignment score contributed by positions aligned to 'CG' dinucleotides in the consensus.

Some families end in a poly-A tail, and variable-length Poly-A tails in subfamily consensus sequences can lead to length-biased annotation confidence. To overcome this, SCULU also includes the option to require that all subfamily consensus sequences share identical-length poly-A tails.

CHAPTER 3 RESULTS

To evaluate the efficacy of SCULU, we applied our methods to three different collections of subfamily consensus sequences using our default merging parameters of $k = \frac{1}{3}$ and Independence$(i, j) <$ 50% except where noted in the text. These collections consisted of (i) 45 Alu subfamilies as present in Repbase, (ii) 195 Alu subfamilies produced by CoSeg [5], and (iii) 2,483 metulj subfamilies from Heliconiine butterflies [13]. In all cases, a multiple sequence alignment (MSA) of instances of each subfamily member is provided as input to SCULU. All tests were performed on a 3.2GHz Intel Xeon E5-2630, with 32 cores and 128 Gb RAM.

For both sets of Alu subfamilies, we sampled the 50 longest sequence instances from each subfamily as our initial test set for alignment. Due to the size of the metulj dataset, we sampled only the 10 longest sequence instances from each subfamily for our test set. The metulj subfamily MSAs present a surprising challenge, in that some instances appear in multiple subfamily MSAs. Since we wish each test sequence to be unique, duplicate instances were removed. All alignments were performed using using the alignment software cross_match, along with with a custom nucleotide scoring matrix used in RepeatMasker. 25p41g 3.1:

Table 3.1: Nucleotide alignment scoring matrix 25p41g.

FREQS	A:	0.295		C:	0.205		G:	0.205		T	0.295
A	R	G	C	Y	T	K	M	S	W	N	X
8	1	-6	-13	-14	-15	-11	-2	-10	-3	-1	-27
2	2	1	-13	-13	-14	-6	-5	-5	-5	-1	-27
-2	3	10	-13	-13	-13	-1	-8	-1	-8	-1	-27
-13	-13	-13	10	3	-2	-8	-1	-1	-8	-1	-27
-14	-13	-13	1	2	2	-5	-6	-5	-5	-1	-27
-15	-14	-13	-6	1	8	-2	-11	-10	-3	-1	-27
-9	-5	-1	-9	-6	-2	-1	-9	-5	-5	-1	-27
-2	-6	-9	-1	-5	-9	-9	-1	-5	-5	-1	-27
-8	-4	-1	-1	-4	-8	-4	-4	-1	-8	-1	-27
-3	-6	-10	-10	-6	-3	-6	-6	-10	-3	-1	-27
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-27
-27	-27	-27	-27	-27	-27	-27	-27	-27	-27	-27	-27

Repbase Alus

The Alu family is a primate-specific family of non-autonomous transposable elements with more than 1 million copies making up roughly 10% the human genome. They have been extensively studied, and have been the subject of a variety of attempts at devising reasonable subfamily sets. One such effort [5] served as the basis of the CoSeg software now used within RepeatMasker, and produced a collection of 195 Alu subfamilies. The Repbase database of transposable elements [3] contains 45 Alu subfamilies, with the reduction resulting from extensive manual curation [personal communication, Arian Smit]. Because this Repbase Alu set is the result of manual curation that was in part intended to remove subfamilies with questionable distinctiveness, little overlap between families is expected. Even so, our earlier analysis [11] found that $\sim 14\%$ of Alu biological replicates showed discordant annotation across replicates, meaning that some cross-contamination is evident. We applied SCULU to this set, to gain insight into its merging patterns. With default settings, we find that SCULU merges 11 pairs of subfamilies. In practice, we do not expect that all of these subfamilies should be removed from the Repbase dataset, because they may be included specifically for reasons that may override the desire for annotation independence; even so, this analysis highlights the fact that SCULU generally conserves most subfamilies, and effectively identifies a few that demonstrate risk of cross-matching behavior.

Considering the merges in order, the first merged subfamily pair is (AluY, AluYm1). We sampled 50 instances from each subfamily, so the expectation is that each subfamily will be a clear winner 50 times over all of the test sequences. However, AluY was never a clear winner, and was part of an uncertain winner set 131 times. AluYm1 was a part of 102 uncertain winner sets out of those 131. Equation 2.2 indicates and Independence(AluY,AluYm1) = 0. Meanwhile, AluYm1 was a clear winner 19 times, so that Independence(AluY,AluYm1) = 19/(19+102) = 0.157. Note that the Independence values are not symmetric. In this case, both subfamilies have sufficient evidence of being unreliable with each other making this a reasonable merge.

The second Repbase Alu subfamily merge selected by SCULU is AluSg and AluSz. AluSg was a clear winner two times, and was part of 38 unclear winner sets. Meanwhile, AluSz was a clear winner

35 times, and was part of an unclear winner set 156 times, AluSg and AluSz were both found in the same unclear winner set 22 times. As a result, Independence(AluSg,AluSz) = 2/(2+22) = 0.083 and Independence(AluSz,AluSg) = 35/(35+22) = 0.614. Though Independence(AluSz,AluSg) is safely above the merge threshold, the same is not true of Independence(AluSg,AluSz). Essentially, this analysis calls for AluSg to be merged *into* AluSz, since AluSg is not sufficiently independent of AluSz.

We find it instructive to explore the unclear winner sets for each of the merged subfamilies 3.2. AluSg was part of 38 unclear winner sets; for 10 sequences, only one other subfamily was in contention for a confidence annotation, but in some cases many more subfamilies were within the threshold confidence fraction:

Table 3.2: Distribution of the unclear winner set sizes containing AluSg.

Uncertain winner set size	Counts
2	10
3	6
4	9
5	3
6	6
12	1
13	1
19	1
20	1

Meanwhile, AluSz was part of 156 unclear winner sets, mostly with sizes of 2-4 3.3:

Table 3.3: Distribution of the unclear winner set sizes containing AluSz.

Uncertain winner set size	Counts
2	63
3	49
4	28
5	6
6	6
8	1
12	1
13	1
19	1

While SCULU suggests a few merges for Repbase Alu subfamilies, most pairs are not merged; we investigate an example in which SCULU choose to reject the merge of the 'uncertain pair' (AluYi6_4d, AluYf1). AluYi6_4d was a clear winner 46 times and was found to be in an 'unclear winner set' 55 times. This means that when AluYi6_4d showed high enough confidence for a test sequence, it was the clear winner of that sequence about half of the time. The specific pair (AluYi6_4d, AluYf1) shared an unclear winner set 33 times, so that Independence(AluYi6_4d, AluYf1) = 46/(46+33) = 0.582. For the inverse relation, AluYf1 was a clear winner 118 times (more than double the expected 50 from its representative instances), so that Independence(AluYf1,AluYi6_4d) = 118/(118+33) = 0.781. Since both relations exceed the independence threshold, the pair is not merged.

The runtime of SCULU on this dataset was 10 minutes and 28 seconds and required 2 iterations

to reduce the set of 45 subfamilies to 33. The following are the selected merges (in order) 3.4:

Table 3.4: Repbase Alu merging statistics.

Subfamily pair	w_i	w_j	u_ij	Independence(i,j)	Independence(j,i)
AluY,AluYm1	0	19	102	0	0.157
AluSg,AluSz	2	35	22	0.083	0.614
AluSq2,AluSq	7	53	55	0.113	0.491
AluSx,AluSx1	14	23	77	0.154	0.230
AluYk3,AluYh3	18	48	87	0.171	0.356
AluYh7,AluYh9	15	16	46	0.246	0.258
AluSx3,AluSx4	8	24	22	0.267	0.522
AluSg4,AluSg7	10	50	24	0.294	0.676
AluYe6,AluYe5	20	47	32	0.385	0.595
AluJr,AluJo	20	27	27	0.426	0.500
AluYk2,AluYk4	20	18	23	0.474	0.535
AluYk11,AluYj4	1	267	5	0.167	0.982

SCULU's default merging parameters are $k = \frac{1}{3}$ and Independence$(i, j) < 50\%$. We explored the impact of altering these parameters, and found that the merging results are fairly resilient to modest changes 3.5:

Table 3.5: Repbase Alu subfamily output set sizes.

		Independence		
		50%	60%	70%
k	1/4	31	30	27
	1/3	33	30	27
	1/2	36	34	31

CoSeg Alus

We also sought to understand how SCULU would perform on the original set of 195 subfamilies identified by CoSeg. With default settings, SCULU reduced the number of subfamilies to 124. Each Repbase subfamily maps to a distinct CoSeg subfamily, except in cases mirroring the merges identified in the previous section. The runtime of SCULU on this dataset was 11 minutes and 27 seconds and 2 iterations were required to reduce the set of 195 subfamilies to 124. Merging results were largely impervious to parameter changes - increasing the Independence value threshold did not provide significantly more merges, nor did changes in k (the parameter that influences the counts of the 'uncertain pairs') 3.6:

Table 3.6: Coseg Alu output set sizes.

CoSeg Alu subfamily output set sizes

		Independence		
		50%	60%	70%
k	1/4	117	111	107
	1/3	124	117	110
	1/2	133	126	120

Metulj

We also evaluated the performance of SCULU on a dataset with many more subfamilies, and with much less analytical history: the metulj subfamilies from Heliconius (butterfly) genomes [13]. The metulj family represents a highly-abundant class of SINE transposable elements, and has been clustered into a remarkably-large 2,493 subfamiles. One suggestion that the subfamilies are not entirely independent of each other is that the distributed dataset of subfamily instances contains numerous cases of redundancy, in which a single instance is included in the MSAs of multiple subfamilies. With default parameters, SCULU collapses the initial set down to 334 independent subfamilies. While this is still quite a large subfamily set, SCULU automates a massive amount of merging, making it much easier for this set to be further reduced with manual input if needed. In the resulting set, 221 of the initial subfamilies were not merged with any other subfamilies, meaning that a mere 9% of the initial metulj subfamilies were considered to be reliably separable on their own. Of the subfamilies that were merged, a majority only required a single merge in order to meet our standards of Independence. However, in some cases SCULU merged an extremely large amount of subfamilies together to meet our thresholds - in the most extreme case, one merged subfamily proposed by SCULU consisted of 908 of the initial subfamilies, about 36%. The table below shows

the distribution of the number of merges for each subfamily in the resulting output set 3.7:

Table 3.7: Distribution of the number of merges in an output subfamily.

Merge count	Frequency
1	221
2	99
4	2
10	1
12	1
18	1
22	1
40	1
68	1
76	1
82	1
190	1
284	1
346	1
908	1

The metulj data exemplifies the value in SCULU's approach of collapsing multiple non-overlapping

'uncertain pairs' in each iteration, rather than performing only a single merge per iteration. Given n subfamilies and m test sequences, SCULU will read in $n \cdot m$ alignments. For each test sequence, up to $\frac{n^2}{2}$ independence value calculations are performed. As a single independence calculation takes $\mathcal{O}(1)$ work, each iteration of SCULU will perform $\mathcal{O}(m\dot{n}^2)$ work, regardless of the number of pairs that need to be merged. Since the amount of work per iteration is high, it is beneficial to make multiple merges per iteration, to reduce the number of times SCULU must read in new alignments and calculate new Independence values. With the multi-merging approach implemented in SCULU, only 73 iterations were required, rather than the 2,149 that would be required for a naive merging approach 3.8.

Table 3.8: Distribution of the number of pairs merged per iteration.

Pairs merged per iteration	Number of iterations
1-10	49
11-50	12
51-150	8
151-300	2
300+	2

The runtime of SCULU on this dataset was 41 hours, 21 minutes and 47 seconds and required 73 iterations to reduce the set of 2,483 subfamilies to 334 3.9.

Table 3.9: Metulj subfamily output set sizes.

		Independence	
		50%	70%
k	1/4	278	256
	1/3	334	275

CHAPTER 4 DISCUSSION

We have demonstrated the efficacy of SCULU on multiple collections of subfamily consensus sequences, and have shown that it produces reasonable results for extremely large collections of subfamilies. The output subfamily set sizes and proposed merges are fairly stable with parameter changes.

The methods in SCULU are designed to work for subfamilies that can be represented by at least 10 near full length sequence instances. This makes it suitable for relatively young subfamilies, but may not be immediately applicable to families whose representatives are mostly fragments.

SCULU proves to be a useful tool as an initial first pass for automating the process of reducing subfamily sets based on annotation confidence. Users can then use the results and accept or reject the proposed merges. For instance, even if a subfamily pair was considered to be unreliable by SCULU, it may still be useful to keep them separate given expert knowledge of a family's evolutionary history.

While the development of SCULU was motivated by TE-specific subfamily classification, and all experiments are constrained to this domain, we note that the method is sufficiently general that it may prove useful in curating other hierarchically clustered datasets as well, such as exploring the seperability of families sharing the same clan in the Pfam [18] database of protein domains or the Rfam [19] database for non-coding RNA families.

BIBLIOGRAPHY

[1] G. Bourque, K. H. Burns, M. Gehring, V. Gorbunova, A. Seluanov, M. Hammell, M. Imbeault, Z. Izsvák, H. L. Levin, T. S. Macfarlan *et al.*, "Ten things you should know about transposable elements," *Genome biology*, vol. 19, no. 1, pp. 1–12, 2018.

[2] J. N. Wells and C. Feschotte, "A field guide to eukaryotic transposable elements," *Annual review of genetics*, vol. 54, pp. 539–561, 2020.

[3] W. Bao, K. K. Kojima, and O. Kohany, "Repbase update, a database of repetitive elements in eukaryotic genomes," *Mobile Dna*, vol. 6, no. 1, p. 11, 2015.

[4] J. Storer, R. Hubley, J. Rosen, T. J. Wheeler, and A. F. Smit, "The dfam community resource of transposable element families, sequence models, and genome annotations," *Mobile DNA*, vol. 12, no. 1, pp. 1–14, 2021.

[5] A. L. Price, E. Eskin, and P. A. Pevzner, "Whole-genome analysis of alu repeat elements reveals complex evolutionary history," *Genome research*, vol. 14, no. 11, pp. 2245–2252, 2004.

[6] Hubley, R. and Smit, AFA., "Coseg - a program to identify repeat subfamilies using significant co-segregating (2-3 bp) mutations," 2017. [Online]. Available: http://www.repeatmasker.org/RMBlast.html

[7] J. M. Storer, R. Hubley, J. Rosen, and A. F. Smit, "Curation guidelines for de novo generated transposable element families," *Current Protocols*, vol. 1, no. 6, p. e154, 2021.

[8] A. C. Wacholder, C. Cox, T. J. Meyer, R. P. Ruggiero, V. Vemulapalli, A. Damert, L. Carbone,

and D. D. Pollock, "Inference of transposable element ancestry," *PLoS genetics*, vol. 10, no. 8, p. e1004482, 2014.

[9] O. Levy, B. A. Knisbacher, E. Y. Levanon, and S. Havlin, "Integrating networks and comparative genomics reveals retroelement proliferation dynamics in hominid genomes," *Science advances*, vol. 3, no. 10, p. e1701256, 2017.

[10] K. Sjolander, "Phylogenetic inference in protein superfamilies: analysis of sh2 domains," in *Proc. Int. Conf. Intell. Syst. Mol. Biol*, vol. 6, 1998, pp. 165–174.

[11] K. M. Carey, G. Patterson, and T. J. Wheeler, "Transposable element subfamily annotation has a reproducibility problem," *Mobile DNA*, vol. 12, no. 1, pp. 1–9, 2021.

[12] K. M. Carey, R. Hubley, G. T. Lesica, D. Olson, J. W. Roddy, J. Rosen, A. Shingleton, A. F. Smit, and T. J. Wheeler, "Polya: a tool for adjudicating competing annotations of biological sequences," *bioRxiv*, 2021.

[13] D. A. Ray, J. R. Grimshaw, M. K. Halsey, J. M. Korstian, A. B. Osmanski, K. A. Sullivan, K. A. Wolf, H. Reddy, N. Foley, R. D. Stevens *et al.*, "Simultaneous te analysis of 19 heliconiine butterflies yields novel insights into rapid te-based genome diversification and multiple sine births and deaths," *Genome biology and evolution*, vol. 11, no. 8, pp. 2162–2177, 2019.

[14] Green, P., "Phrap and cross match." [Online]. Available: http://www.phrap.org/phredphrap/phrap.html

[15] K. Katoh and D. M. Standley, "Mafft multiple sequence alignment software version 7: improvements in performance and usability," *Molecular biology and evolution*, vol. 30, no. 4, pp. 772–780, 2013.

[16] R. D. Finn, J. Clements, W. Arndt, B. L. Miller, T. J. Wheeler, F. Schreiber, A. Bateman, and S. R. Eddy, "Hmmer web server: 2015 update," *Nucleic acids research*, vol. 43, no. W1, pp. W30–W38, 2015.

[17] S. Henikoff and J. G. Henikoff, "Position-based sequence weights," *Journal of molecular biology*, vol. 243, no. 4, pp. 574–578, 1994.

[18] S. El-Gebali, J. Mistry, A. Bateman, S. R. Eddy, A. Luciani, S. C. Potter, M. Qureshi, L. J. Richardson, G. A. Salazar, A. Smart *et al.*, "The Pfam protein families database in 2019," *Nucleic acids research*, vol. 47, no. D1, pp. D427–D432, 2019.

[19] E. P. Nawrocki, S. W. Burge, A. Bateman, J. Daub, R. Y. Eberhardt, S. R. Eddy, E. W. Floden, P. P. Gardner, T. A. Jones, J. Tate *et al.*, "Rfam 12.0: updates to the rna families database," *Nucleic acids research*, vol. 43, no. D1, pp. D130–D137, 2015.