– Independent Work Report Spring, 2019 –

Simulating Domain-Level Evolution in Zinc Fingers

William Lin
Adviser: Mona Singh, Chaitanya Aluru

Abstract

This paper describes the steps by which a new domain-level simulation of zinc fingers was developed. The simulation combines the use of multiple Python packages for phylogenetic tree creation and sequence evolution, as well as a unique representation of domain-level events (including duplications, losses, and speciations) in order to simulate domain evolution patterns as well as sequence level evolution of protein domains and the linker regions which surround them. The resulting orthogroups generated by the simulation were finally compared to actual zinc-finger orthologous groups in order to verify the simulation's accuracy. This simulation aims to provide new insight regarding the evolutionary mechanisms of zinc finger protein domain evolution and addresses a void in currently available domain-level simulation models.

1. Introduction

1.1. Gene Families and Zinc Fingers

Gene families are considered a characteristic feature of higher-level organisms [10]. There exist many gene families that are conserved and diversified across multiple lineages of mammals, including humans. One such gene family is that of the Cys2Hys2 Zinc Finger (or C2H2 zinc finger for short), more generally known as zinc fingers. Zinc finger protein domains are small protein structural motifs which are characterized by the presence of one or more stabilizing zinc ions; zinc fingers code for a class of transcription factors, or key regulatory proteins, which control gene expression by activating or repressing associated genes.

Zinc fingers are particularly notable as they are the most numerous class of transcription factors in many multicellular animals. Within the human genome, there are approximately 700 members of

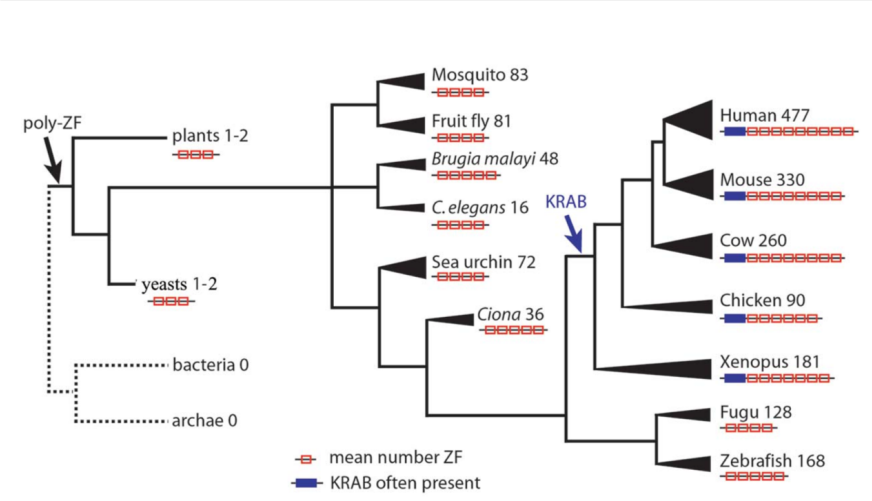

Figure 1: Diversification of zinc fingers in animals.

the zinc finger gene family of transcriptional repressors (Figure 2) [10]. Zinc finger proteins in the C2H2 zinc finger superfamily alone account for more than half of the transcription factors in the human genome (Figure 2) but are not nearly as frequent or diverse in other eukaryotes evolutionarily distant from humans (they are similarly common in other primates) [17]. As shown in Figure 1 [10], there have been multiple expansions in zinc fingers throughout history (we believe zinc fingers have undergone rapid expansions in primates on chromosome 19). The number of zinc-finger domains per protein within eukaryotes increases as organisms get closer to humans and decreases in more distantly related organisms. As a result of this relative abundance, the information appears to suggest an overall trend of evolution towards higher mammals, as indicated through fingers.

Previous research suggests that gene families like Zinc fingers evolved through a series of repeated gene duplications and functional divergence. The evolution of regulatory genes like zinc fingers has important functional importance, as changes in these genes directly impact networks of other effector genes (molecules which bind to and regulate other proteins) and have effects on a species' physiology and development. Since the regulation of zinc fingers carries functional importance for humans and is a main source of transcriptional difference between humans and other mammals [11], understanding the genomic organizations of how zinc fingers evolved is extremely valuable.

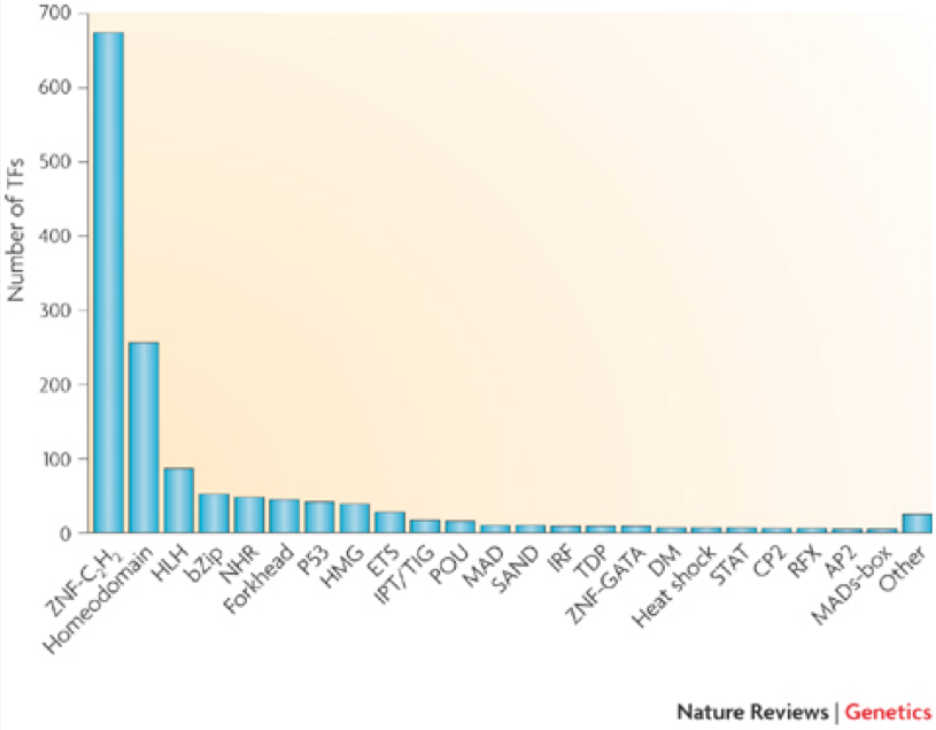

Figure 2: Transcription factors frequencies within humans.

1.2. Motivation and Goal

The goal of this project is to create a simulation which can faithfully simulate domain level evolution for critical sequences of zinc fingers. By creating a viable simulation model, we are able to further examine the overarching evolutionary trends discussed in 1.1 regarding the fundamental importance of zinc-fingers. Currently, there exist multiple gene family simulations, but limited research regarding domain level evolution within organisms due to complications which arise when researching functional domains. Domains are functional units whose function are strongly dependent on conservation of certain portions of their sequence. Therefore standard sequence evolution methods are not applicable. In addition, several classes of domains including zinc fingers exhibit complicated multidomain duplication and loss events which must also be taken into account. As a result, domain evolution is a more constrained problem than standard sequence evolution. There is currently no end to end solution for simulating domain level evolution, and this independent work seeks to address this.

2. Background and Related Work

2.1. Protein Domains

Some protein classes, like zinc fingers, are particularly unique, as they are composed of subunits named domains. Domains can be defined as segments of proteins with distinct structure, function, and evolutionary history [8]. These protein domains often combine with one another in order to form larger, multi-domain proteins, and in vertebrates like humans, proteins from the same gene family can be found in repeated, adjacent patterns. In many families, including C2H2 zinc fingers, domains are mainly found in these regular structures, and examples of these structures include repeated β-sheets or solenoids [5]. Domain structures, particularly longer domain repeats, are especially common in multicellular species [6]. Previous research suggests that protein classes with the highest rates of repeats carry out important functions in eukaryotes and also serve to increase genetic variability amongst eukaryotes [14]. Domain repeats can be observed interacting with other bodily proteins and ligands. Even amongst well-defined and conserved protein domains, sequence conservation is often low, and variation in sequences and domain repeat length allows for domain repeats to both bind to a plethora of different partners and perform highly diverse functions.

Currently, the evolutionary mechanisms by which protein domains, including those of Zinc fingers, diversified in animals is relatively unknown. Domain repeats are thought to have formed as a result of tandem duplications, where segments are duplicated and directly inserted next to their origin [8]. Other hypothesized explanations for domain repeats include but are not limited to recombination of intron (coding) regions, or DNA "slippage" which occurs as a result of DNA hairpins [9]. Domain duplication has been shown to commonly occur during evolution [13], and as a result of the aforementioned high frequency and functional importance of domain-repeat coding genes in animals, understanding the periods in evolutionary history in which these events occured can help us understand why and how zinc finger repertoires differ so drastically across species.

2.2. Current Status of Protein Domain Research

Previous research regarding protein domains can be broken down into three major categories: identification and clustering, domain evolution dynamics, and reconciliation based approaches.

Identifying and Clustering: There currently exist a few options for researchers investigating protein domains to identify and cluster them. HMMER is a widely used software package which searches sequence databases for homologs, or sequences with shared ancestry, and makes sequence alignments. HMMER software (the latest version of which is 3.2.1) is used in conjunction with a profile database such as Pfam; HMMER compares the profile to the input sequence, and sequences which score higher in HMMER are considered to be homologous. With HMMER, researchers are able to easily identify and analyze protein domains.

Domain Evolution Dynamics: There is currently limited research regarding the genomic organization of protein domains and how they evolved. Researchers have investigated the amount of protein domains that are duplicated over time and based on sequence similarity data from alignments, have attempted to provide initial insights regarding the evolutionary behavior of these conserved domains [8]. Figure 3 visualizes one previous investigation during which researchers extracted proteins domains from the sequence and calculated pairwise similarities between the discovered domains. Alignments between all these domain pairs reveals “diagonals" of long sequences of domains sharing extremely high similarity, all of which indicate block duplication.

Based on these "heat maps" (as shown in Figure 3), important inferences are able to be drawn regarding the protein domain’s evolutionary history. These inferences have revealed that in regions of high domain repeats, duplications are more common than deletions and in some protein domains like zinc fingers successfully identified patterns of repetition including tandem domain duplications.

Reconciliation-Based Approaches: There have been no previous attempts of reconciliation-based approaches in order to recreate the evolutionary histories of protein domains, but there have been some early attempts to apply reconciliation-based approaches with genes. Reconciliation based approaches are rooted in the incongruency between a gene tree and corresponding species tree which arise as a result of evolutionary events such as gene duplication and gene loss. Reconciliation-based

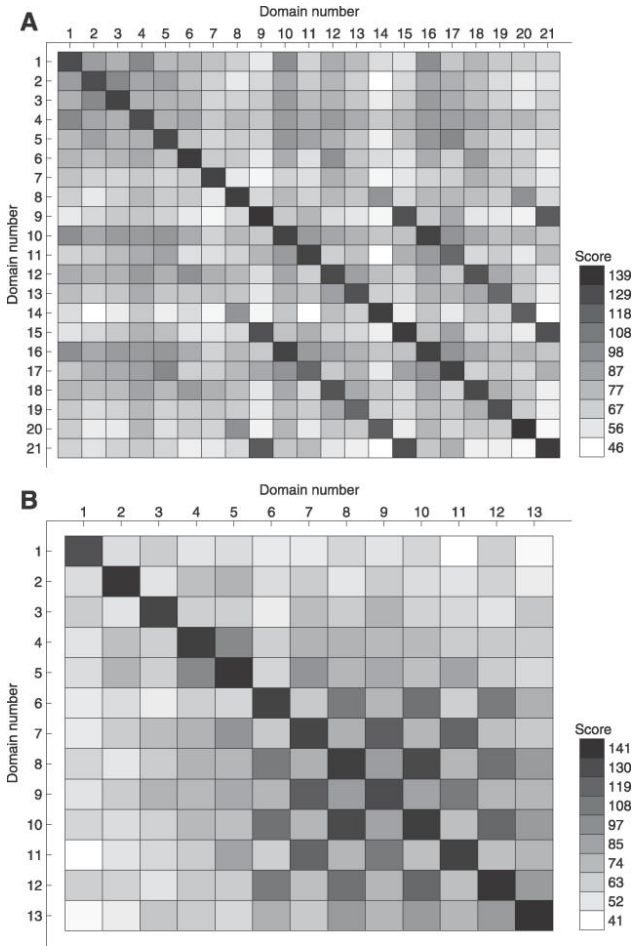

Figure 3: Alignment scores between sequences.

approaches aim to simulate duplication, loss, and lateral gene transfers (otherwise known as DTL-scenarios) using dynamic programming algorithms. Though powerful, these reconciliation-based approaches are unable to perform rigorous phylogenetic access; these approaches are able to detect that block duplications occured at some point in the past, but are unable to reveal when, the organism in which they occured, and the event history prior to the duplication.

2.3. Related Algorithms

A valuable first step in understanding these mechanisms is to create a simulation which can reproduce specific aspects of zinc finger evolution. The previously described attempts to research protein domains heavily focus on developing algorithms to infer phylogenetic histories. The goal of this independent work is to go one step further, developing a viable simulation model to develop a "ground truth" to compare against real data. By writing an accurate simulation, we will be able to

learn more about the mechanisms by which domain-level evolution occurs, particularly within zinc fingers. There currently exist multiple tools which provide incomplete aspects of this framework. Many of these tools accurately simulate gene family evolution, but they do not take into account the previously described additional constraints introduced by domain level events.

GenPhyloData: GenPhyloData is a suite of tools which creates simulated phylogenetic trees for families of genes. GenPhyloData creates simulated phylogenetic trees using a birth-death process and can also generate gene family trees based on a species tree which accounts for gene duplication, gene loss, and lateral gene transfer events [15]. However, these simulations are constrained to the gene level, and do not provide any valuable insights for evolution at the domain level as it occurs in zinc fingers. In addition, GenPhyloData is unable to adequately simulate tandem duplications, which as previously explained are commonly observed in zinc fingers.

Pyvolve: Pyvolve is an open source Python module which is specifically designed in order to simulate gene-level evolution based on phylogenetic trees using continuous-time Markov models for sequence evolution [16]. Pyvolve outputs its simulated sequences in groups of partitions, the evolution of each of which can be modeled using substitution matrices (examples of which include JTT, WAG, and LG). Pyvolve, however, does not take into account the presence of functional domains within a sequence. If one were to attempt to simulate evolution of amino acid sequences using Pyvolve, the library may erroneously overwrite conserved sequences of amino acids which define protein sequences. These erroneous revisions do not accurately reflect true evolutionary events, as modifications are highly conserved sequences of protein repeats may result in serious negative effects on an organism's physiology.

REvolver: REvolver is a software package with simulates sequence level evolution while conserving some evolutionary stable sequence characteristics, such as functional domains [12]. REvolver is the closest currently available software for simulating domain evolution but does not adequately simulate zinc finger evolution. REvolver utilizies HMMER 3.0 software to identify functional domains, but as research conducted by advisor Chaitanya Aluru indicates, HMMER 3.0 is quite bad at recognizing short domains like zinc fingers. As a result, REvolver suffers from similar

issues to Pyvolve, failing to preserve short functional domains within an amino acid sequence.

Due to the nature of domain evolution, we hope to develop a model which tightly integrates aspects of phylogeny generation and sequence generation. However, in these aforementioned packages, the constraints of both are not completely addressed; there is currently no end to end solution for simulating domain level evolution which fully addresses the difficulties associated with simulating functional domains. Developing a package which addresses both factors as well as provides domain-level simulation will address these concerns.

3. Approach

The main goal of this independent work is to generate sequence groups which resemble and simulate the evolution of actual zinc fingers. We seek to match certain attributes of real sequences and compare generated data to existing orthologous groups of zinc finger proteins. The main approach of this project was to develop software which, when provided domain history and gene history, generated a group of sequences from a root sequence which later was compared to current sequence data in order to verify the simulation accuracy.

We broke down the challenge of simulating domain-level evolution into two main components: macro-scale evolution and micro-scale evolution. In the case of domain-level evolution of zinc fingers, macro-scale evolution refers to generating a group of orthologous sequences from a starting root sequence based on speciation, tandem duplication, and loss events. Micro-scale evolution specifically refers to the constantly occurring sequence evolution (random mutations, insertions, and deletions) of both linker sequences between functional domains as well as the functional domains themselves. This will properly address the domain scale evolution at all levels, ensuring accurate simulation to avoid the constraints associated with functional domain modeling. The design of the simulation draws inspiration from the reconciliation-based approaches in Section 2.2. In previous models of gene family simulations, reconciliations can be defined as a mapping of nodes from a gene tree G to the nodes of a corresponding species tree S, where G and S are rooted, full binary trees. By using such a mapping, reconciliations can be formalized and adequately represent duplications,

losses, and lateral gene transfers. For genes, DTL-scenarios can be defined as follows:

- Each gene tree vertex is assigned exactly one event: a gene speciation, a duplication, or a lateral gene transfer.
- Each host tree vertex is mapped to one ore more guest tree vertexes, and there exists a host tree vertex for each mapped guest tree vertex.
- Every vertex of the gene tree is mapped into the species tree in a way that is consistent with the previous points and with the temporal order implicitly represented by the trees.

In our current new model of domain family simulations, we redefine reconciliations as a mapping of nodes from a host tree (representing gene-level events) and a guest tree (representing domain-level events). In addition, we adjust our definition to reflect domain specific events as follows (which we will henceforth refer to as a DL-scenario):

- Each guest tree vertex is mapped to exactly one event: a domain duplication, loss, or speciation.
- Each host tree vertex is mapped to one ore more guest tree vertexes, and there exists a host tree vertex for each mapped guest tree vertex.
- Each vertex of the guest tree is mapped into the host tree in a way that is consistent with the temporal structure represented by the trees.

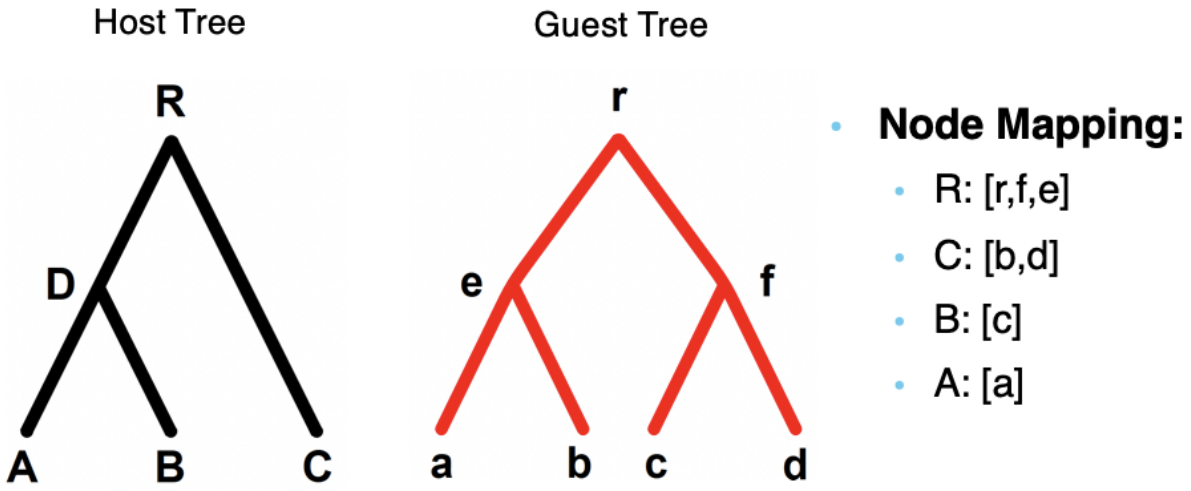

Figure 4: Example of a host tree, guest tree, and mapping between nodes [4]

Figure 4 provides a visual example of the DL-scenario which serves as a basis for the novel simulation. Based on the phylogenies provided by the rooted guest tree and host tree, we can then reconstruct the evolutionary history of zinc fingers at both the domain level and the gene level.

4. Implementation

Function Name and Parameters	Description
def findLinkers(starts, ends, startingSequence)	Returns information about linker start positions, end positions, and sequences
def reconstructSequence(starts, ends, sequences, linkerStarts, linkerEnds, linkerSequences)	Returns a complete amino acid sequence based on linker and domain sequences
def probability(score)	Returns probability of amino acid based on score in HMM file
def mutateDomain(domain,hmmfile,branchlength)	Returns protein domain mutated as per probabilities calculated from HMM file
def domainEvolution(host, guest, hmmfile, nodemap, sequence)	Returns orthogroup of sequences as well as domain bookkeeping information

Table 1: Simulation API

Table 1 details the API that was developed for the simulation model. As described in the approach, the implementation for this simulation software was broken down into three separate phases, namely the birth-death process in order to generate sample phylogenetic trees, domain tracking, and sequence generation for domain and non-domain evolution.

4.1. Birth-Death Process

The implementation of this simulation was written in Python 2.7 for multiple reasons. Python features robust libraries for managing tree based data structures (ETE3), as well as easy-to use libraries for common computational biology tasks such as sequence alignments and sequence evolution (Pyvolve). ETE3 is a Python framework which provides a rich API for the annotation, analysis, and visualization of trees. ETE3 allows us to create, search, and modify our guest trees and host trees as Python objects. For example, Figure 5 provides examples of sample host and guest trees annotated by node label, branch length, and node events, which were created using ETE3 and subsequently used in the testing of simulation.

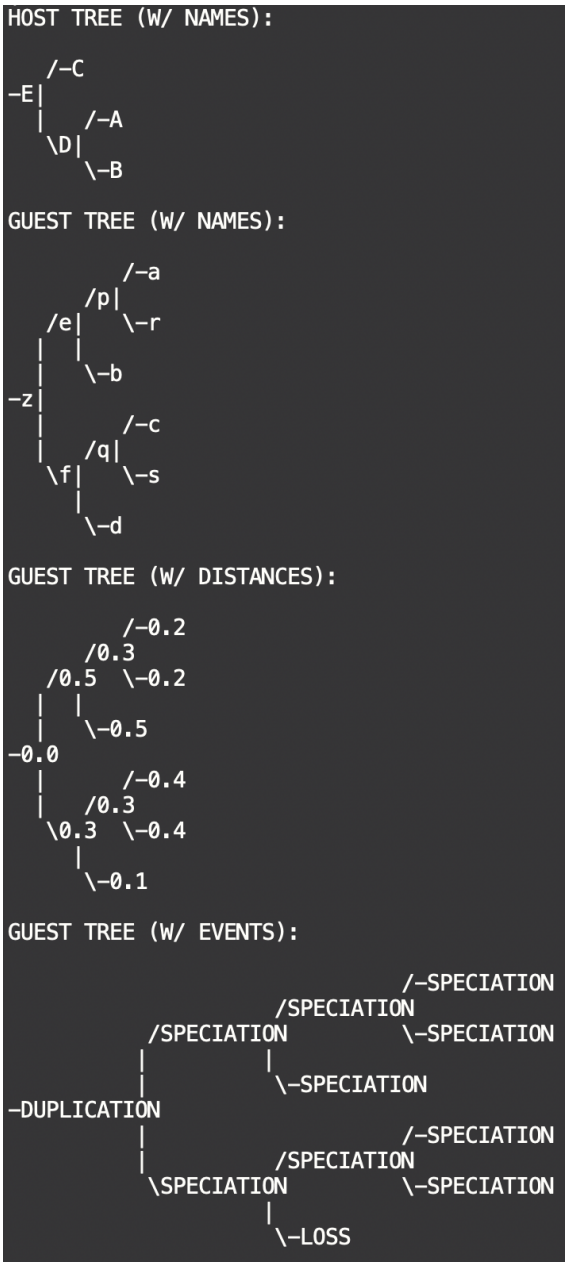

Figure 5: Example of a ETE3 structure host tree

The host trees and guest trees used for parts of the analysis in Section 5 were developed by advisor Chaitanya Aluru according to the birth-death models of diversification, a continuous-time Markov process which tracks how individuals change over time based on a given birth and death rate. Using the birth-death paradigm, we are able to simulate multidomain duplications. Birth-death models are particularly useful when performing phylogenetic analysis; birth-death models intuitively represent events where births reflect duplications (number of "individuals" or species increases by one) and

deaths reflect losses (number of "individuals" or species decreases by one) [1]. As a result, we can further apply these birth-death models to replicate domain-level events where duplications correspond to domain duplications and deaths correspond to domain losses. In our case, instead of increasing population size by 1, we increase by a number drawn from an exponential distribution with increasingly small chances of drawing larger duplication sizes.

The current software suite includes a host tree generating function which, when provided a birth rate, death rate, and tree height, randomly generates a tree topology and assigns branch lengths to all of the nodes of the tree such that the sum of the branch lengths from any root to leaf is equal to the tree height. In order to create a guest tree from the host tree, the topology of a guest tree is then created within each node of the host tree based on the host tree topology, duplication rates, and average distances which occur between events. For each pair of host trees and guest trees, a mapping is then generated, as previously described in Section 3.

4.2. Domain Tracking

The simulation function for evolution of domain-specific events directly requires the host tree, guest tree, and HMM profile for protein domain under investigation (in this case zinc fingers), and a root starting sequence (representative of an ancestral amino acid sequence containing zinc finger domains). For testing purposes, we initially utilized a randomly generated C2H2 zinc finger protein sequence. In order to make a realistic root sequence, we randomly sampled zinc finger proteins from existing sequences. A prefix and suffix region were taken from a single zinc finger protein, and the desired number of domains were then taken one at a time from randomly drawn proteins. These proteins were taken from separate orthogroups to ensure no recent evolutionary history between domains on the randomly sampled protein. Linker regions were retained between domains in order to generate a realistic zinc finger sequence. *hmmsearch* was utilized to search for a zinc finger protein profile HMM, and the output was parsed using *grep* in order to output three domain bookkeeping lists: start positions, end positions, and sequences of zinc fingers within the sequence. Given the two tree structures, we traverse them as follows.

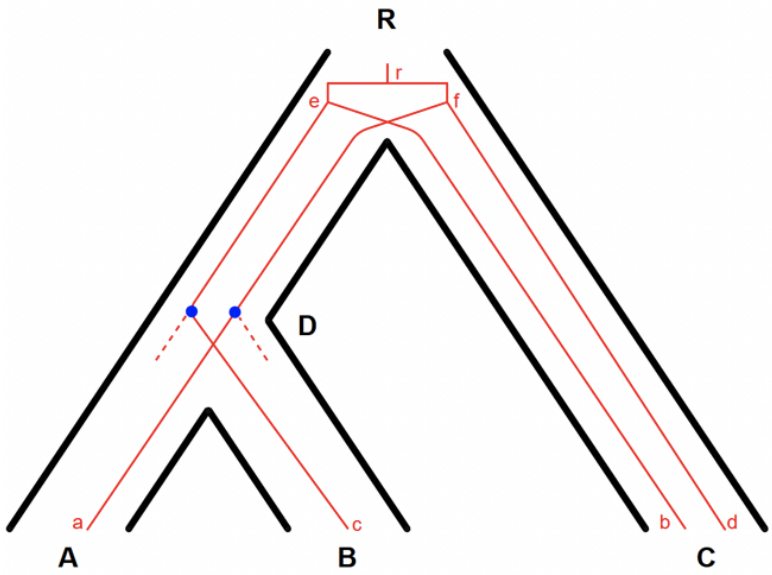

Figure 6: Simultaneous traversal of both the host tree and the guest tree [4]

The "merged" tree in Figure 6 reflects both the host tree (in black) and guest tree (in red) specified in Figure 4 and directly visualizes the mapping between the two. The simulation performs a post-order traversal of each of the host nodes. Given the node mapping and the guest tree, when the simulation reaches a new host node, it first generates a subtree containing only the mapped nodes of a host node. For instance, host node R directly maps to guest tree nodes r, e, and f. Each of these newly generated subtrees represents a series of speciation, duplication, and loss events. In Figure 6, we can observe that within node R, there exists a duplication event where the zinc finger represented by node r duplicates, and zinc fingers e and f undergo speciation events.

For each host node, given the guest subtree nodes to which it is mapped, we identify all of the possible guest subtree roots within the nodes by checking if each of the node's parents are in the list. In the case of Figure 6, there is one root r for node R, while e and f are both not roots as r is in the mapping of R. After identifying all of the roots, we traverse each of the guest subtree(s) within each host node based on evolutionary distances (which are assigned as the branch lengths of the tree) using a shortest path algorithm. Whenever we encounter new branches, we add their branch lengths to a list data structure storing all of the distances of the next nodes. At any point, the next guest tree node we are traversing to is that with the shortest evolutionary distance away from the root node.

After iterating over a guest subtree node, we remove it from the list and traverse to the node with shortest distance.

Throughout the guest subtree traversal, we keep track of the start positions, end positions, and sequences of zinc fingers which change as new domains are added or lost throughout the events during guest subtree traversal. Each guest subtree node can either be a duplication, loss, or speciation event. In the case of a duplication, a new domain is copied directly after a node and its trailing linker sequence; a new entry is added to each of the three domain bookkeeping lists. In the case of a loss, the current domain is removed; all entries of the domain are removed from the domain bookkeeping lists. In the case of a speciation event, the three domain bookkeeping lists are unchanged. At the end of traversing each guest subtree, the final state of each of the three domain bookkeeping lists, are mapped to each of the host nodes.

Based on the initial start positions and end positions of the identified zinc finger protein domains, we are able to also deduce the positions of the linker sequences, or short peptide sequences that occur between protein domains. When traversing the tree as mentioned in Section 4.2, domain duplications and deletions are also reflected in changes to the positions of linker sequences. Therefore, we also maintain bookkeeping lists of the start positions, end positions, and sequences of each of the linkers.

The original amino acid sequence is then reconstructed from the start positions, end positions, and sequences of both the linker sequences and the zinc finger protein domains. The resulting reconstructed sequences at the end of traversing each guest subtrees are finally mapped to the host node. The overall output of the simulation is therefore the sequences of the orthogroup, the leaves of the host tree which represent the most recent descendants of the ancestral root sequence.

4.3. Sequence Generation: Domain and Non-Domain Evolution

In addition to domain-level evolution events, there is constant sequence-level evolution which occurs amongst domains and the linker sequences surrounding them when we traverse the guest trees. For example, in Figure 6, the original linker sequences surrounding node r continue mutating until the guest subtree is completely traversed. In addition, even though nodes e and f are duplicated from

the same starting domain, the domains e and f have different sequences as a result of constantly ongoing sequence level evolution.

Using the Python module Pyvolve, we evolve non-domain sequences of linkers simultaneously as we traverse the host and guest trees. The substitution model we selected for amino acid substitutions for Pyvolve was JTT, a versatile and currently popular choice for research [7]. Pyvolve evolved the linker sequences based on the branch length of the sequence inputted; as the simulation iterates through the guest subtrees, when it traverses to another node, Pyvolve evolves the sequence based on the length of the sequence on which it is traveling.

Unlike sequence evolution for non-domains, sequence evolution for domains is a bit more complicated. In the basic sequence evolution model that Pyvolve utilizes, all bases are treated as equally likely to be mutated because standard substitution matrices are used to handle their evolution. In domain-sequences however, this is not true, as specific amino acids in zinc finger protein domains are highly conserved. In order to tackle this issue, we confer with HMM models. Based on the previously downloaded HMM models from Pfam, we observe that specific amino acids are more likely to mutate and evolve than others.

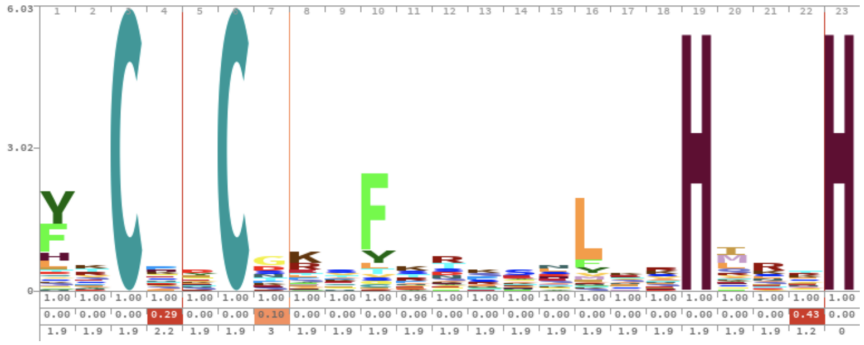

Figure 7: C2H2 zinc finger HMM logo

HMM logos provide an easy to understand graphic representation of an HMM file and reveal the conservation pattern of a set of sequences [18]. The size of each letter at each position indicates the degree to which that sequence is conserved at that location. In C2H2 zinc fingers, one can observe that positions 3, 6, 19, and 23 to highly conserved, while there are varying levels of sequence

variation for the remaining positions [3]. We utilized the information stored in the C2H2 zinc finger HMM file in order to reflect this conservation pattern.

HMM	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	
	m->m	m->i	m->d	i->m	i->i	d->m	d->d	b->m	m->e												
	0	*	-13531																		
1	-4836	323	-1381	-6544	2942	-2963	1855	-1152	-2263	-404	-1245	-2818	-6341	-1949	-5940	-1921	-1257	-1841	-565	3539	1
-	-210	-336	220	-37	-342	377	73	-669	210	-511	-732	404	423	70	27	439	205	-414	-161	-176	
E	0	-12489	-13531	-894	-1115	-701	-1378	0	*												
2	-1305	-948	-1084	213	-1422	-5478	-783	-868	1395	-2844	-591	-528	122	1128	1125	-60	1150	682	-5992	2	2
-	-210	-336	220	-37	-342	377	73	-669	210	-511	-732	404	423	70	27	439	205	-414	-161	-176	
E	0	-12489	-13531	-894	-1115	-701	-1378	*	*												
3	-10990	6043	-10297	-10753	-10799	-9316	-9950	-12017	-10863	-11296	-11274	-10574	-9703	-10734	-10338	-11426	-10995	-11708	-9350	-10779	3
-	-210	-336	220	-37	-342	377	73	-669	210	-511	-732	404	423	70	27	439	205	-414	-161	-176	
E	0	-12489	-13531	-894	-1115	-701	-1378	*	*												

Figure 8: C2H2 zinc finger HMM file

Within the HMM file, the rows of the HMM correspond to different positions, and the columns of the HMM file correspond to all possible amino acids. For each position, the first row of scores represents the probabilities that parameterize the HMM. The higher the score, the more likely that amino acids are to occur at that position. In Figure 8, column C is the only non-negative value at position three (6043), indicating that it is extremely conserved.

Based on the scores for each of the amino acids, we can generate a probability distribution of amino acids for each position in order to determine which are the most likely to occur. We utilize an equation from the HMMER 2.3.2 documentation in order to calculate the probability from the scores in an HMM save file [2].

$$P(\text{Amino Acid}) = N \cdot 2^{\frac{score}{Intscale}}$$

Intscale is a constant defined to be 1000, and N is a constant defined as the null model probability. Since N is not provided, we rederived the null model probability used by HMMER 2.3.2 given that the probability that position 3 is amino acid C is 1 with reasonably certainty.

$$P(D) = N \cdot 2^{\frac{score}{Intscale}}$$

$$1 = N \cdot 2^{\frac{6043}{1000}}$$

$$N \approx 0.015166$$

After formalizing the probability equation, we calculate the probabilities for each amino acid, creating a probability distribution at each position. The expected number of amino acids e which undergo mutations (with replacement) can then be calculated by multiplying branch length by sequence length. In order to select which positions to mutate, the information content of each position is calculated using the equation $-\sum_{\forall i} p_i \cdot log_2(p_i)$, summing over a single probability distribution for a given position i. The e positions with the highest information content are then mutated as per the probability distribution. These calculations allow for more constrained domain-sequence evolution with accurately takes into account HMM based evolution and conserved sequences. By breaking down the simulation problem by weaving together "host" level events and "guest" level events, the simulation is able to accurately simulate domain-level evolution while allowing for the presence of sequence level evolution.

5. Evaluation and Results

```
**********
HOST NODE: E
HOST NODE CHILDREN: [Tree node 'C' (0x1a29d48f9), Tree node 'D' (0x1a29d48f5)]
GUEST ROOTS: [Tree node 'z' (0x1a29d48f1)]
INPUT POSITIONS: {}
INPUT SEQUENCES: ['FECKECGKTFSRSTHLIEHQRTH']
INPUT STARTS: ['FECKECGKTFSRSTHLIEHQRTH']
INPUT ENDS: ['FECKECGKTFSRSTHLIEHQRTH']
MAPPING: [Tree node 'e' (0x1a29d6c0d), Tree node 'f' (0x1a29d6c15), Tree node 'z' (0x1a29d48f1)]
GUEST ROOTS: [Tree node 'z' (0x1a29d48f1)]
----------
CURRENTLY EXAMINING GUEST ROOT: z
CURRENT CLOSEST NODE: z
EVENT TITLE: DUPLICATION
POSITIONS: {Tree node 'e' (0x1a29d6c0d): 0, Tree node 'f' (0x1a29d6c15): 1}
STARTS: [130, 158]
ENDS: [152, 180]
CURRENT SEQUENCES: ['FCCEECGITFMRSTHLTEHYRTH', 'FCCKEGGKKFSRPTRLIEHYRTH']
----------
CURRENT CLOSEST NODE: f
EVENT TITLE: SPECIATION
POSITIONS: {Tree node 'e' (0x1a29d6c0d): 0, Tree node 'f' (0x1a29d6c15): 1}
STARTS: [130, 158]
ENDS: [152, 180]
CURRENT SEQUENCES: ['FCCEECHITFMRSTHLTVHYRTH', 'FCCKFGGKKFSRPTRLIEWYRTH']
----------
CURRENT CLOSEST NODE: e
EVENT TITLE: SPECIATION
POSITIONS: {Tree node 'e' (0x1a29d6c0d): 0, Tree node 'f' (0x1a29d6c15): 1}
STARTS: [130, 158]
ENDS: [152, 180]
CURRENT SEQUENCES: ['FCCEECHITFMRSTHLTVHYRTH', 'FCCKFGGKKFSRPTRLIEWYRTH']
FINISHED EXAMINING GUEST ROOT: z
----------
**********
FINAL HOST NODE DOMAIN SEQUENCES: ['FCCEECHITFMRSTHLTVHYRTH', 'FCCKFGGKKFSRPTRLIEWYRTH']
FINAL AMINO ACID SEQUENCE: VPSLRGVMSEETCNAEGQINRQAPPQIEKKNNFEKTEVRSMCVSLTEISPSERGPEATMFSIISNLETQSKASKGD
RSPNFRQNSVDNRELKYQNFVCQKKPCKFKSCRNAFRYESDEFIHNCLHDGESLFCCEECHITFMRSTHLTVHYRTHIESLQGELNEHLGNSQVQVSRHTPAP
TQSKMCLEKTKIKTMFVTLNQIAPGERGPTSNEFTLNNAIDTQQQICKGDRGPSFRMNSREDSEFRHQSFGHKNTPCKFSGCGNSFRYDSDLCITSRIYEGEK
SFCCKFGGKKFSRPTRLIEWYRTHTVDKTHTTE
```

Figure 9: Results during intermediate steps of the simulation

Figure 9 details the preliminary results which were achieved during the intermediate steps of the simulation. As we are iterating through both the host nodes and the guest subtrees, information regarding the current state of the simulation is outputted to the terminal. Traversals over a host node are bounded by asterisks (*), and traversals over guest nodes are bounded by hyphens (-). During each step of the traversal, information about the host node (namely its name, its children, the guest subtree roots it contains, the current lists of of the positions, starts, ends, and sequences used for bookkeeping the locations of functional domains within the amino acid sequence) are stored as features of the node.

At its conclusion, the simulation algorithm outputs a list representing the orthogroup, sequences at each of the leaves of the host tree which represent the descendant sequences of the original root sequence, as well as a dictionary containing the three aforementioned bookkeeping lists (start positions, end positions, and sequences of zinc finger protein domains) for each of the leaf nodes of the host tree. In order to verify the validity of the simulation results, we can compare these outputs against actual data from real orthogroup zinc finger sequences. This real data was drawn from OrthoDB.org, an online catalog of orthologous protein-coding genes, and the specific dataset was that of a clade of placental mammals known as Eutheria.

Host trees were generated using the birth-death process with a duplication rate of 0.3, a loss rate of 0.1, and an average tree height of 0.3. Guest trees were generated using a sigmoid function for duplication rate, and an exponential function for duplication size. We generated 500 pairs of randomized corresponding host and guest trees. Since these trees are randomly generated, there were often some bizarre instances where the algorithm would output degenerate tree structures where the resulting sequence similarities whose pairwise sequence similarities would equal one. The output of the randomized tree generation was screened as to not consider eventual simulation model based on erroneous tree structures; all cases where no domains remained after the evolutionary process were removed from sample results. Based on these randomly generated host and guest trees pairs, the simulation was run using the standard C2H2 zinc finger hmm model and the same randomly generated root sequence for each of these 500 pairs of trees.

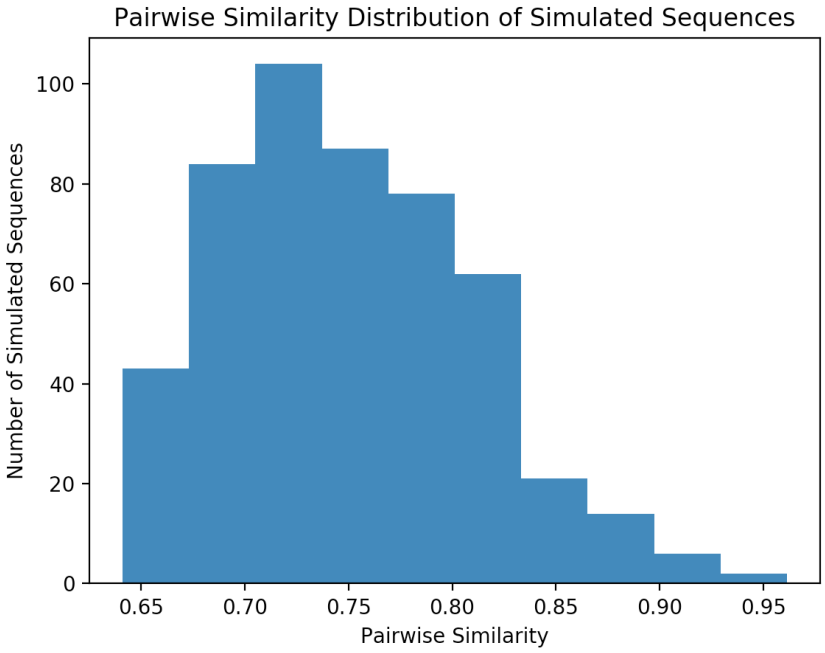

Figure 10: Distribution of pairwise similarities between simulated sequences

We first analyzed the pairwise similarity of the simulated sequences within the orthogroups with one another. The distribution of these average pairwise similarities for each groups of sequences is shown in Figure 10. The distribution is a bell-shaped distribution which is slightly positively skewed (the mean is greater than the median for this distribution). The average pairwise similarity of simulated sequences was 0.7511 with a variance of 0.00353. We first calculated the average pairwise similarity of each orthogroup, and averaged over all of the simulated orthogroups.

We perform the same analysis regarding pairwise similarities of the sequences in the orthogroup using the Eutheria dataset acquired from OrthoDB. This new analysis reveals similar values, and the distribution of values can be observed in Figure 11. The average pairwise similarity of orthogroup sequences is 0.64706 with a variance of 0.05136. The pairwise similarities for sequences are approximately centered around the same values, and the minor differences between the graphs can be attributed to randomization in constructing the analysis, namely in the branch lengths generated by the randomized trees as well as the specific OrthoDB test set selected for this analysis. Thus, we can reasonably conclude that the simulation, given its constraints, can accurately generate realistic sequences.

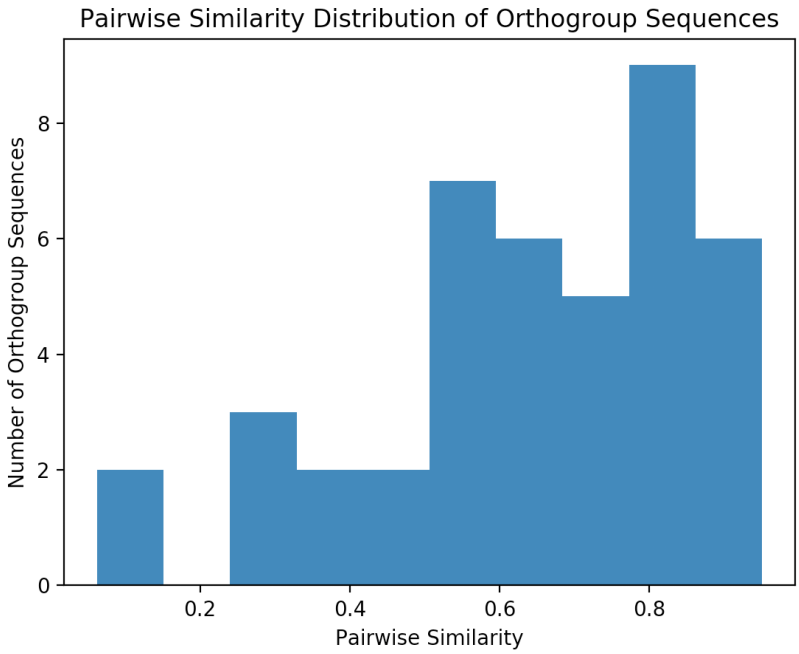

Figure 11: Distribution of pairwise similarities between OrthoDB sequences

In addition, we also calculated the average pairwise similarity between the zinc finger domain sequences within each of the orthogroups. The distribution of these average pairwise similarities are shown in Figure 12. The distribution is positively skewed (to a large spike in pairwise similarity

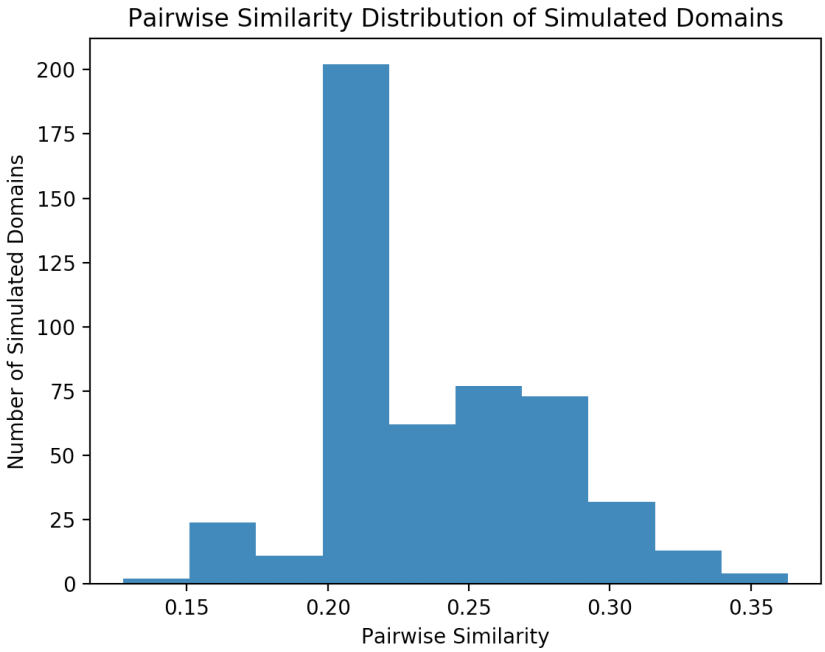

Figure 12: Distribution of pairwise similarities between simulated domains

around 0.20). The average pairwise similarity between randomly simulated domains is 0.23578, with a variance of 0.00017. Again, we perform the same analysis for zinc finger domains but using the Eutheria datset acquired from OrthoDB, determining an average pairwise similarity of 0.50271 with a variance of 0.01144.

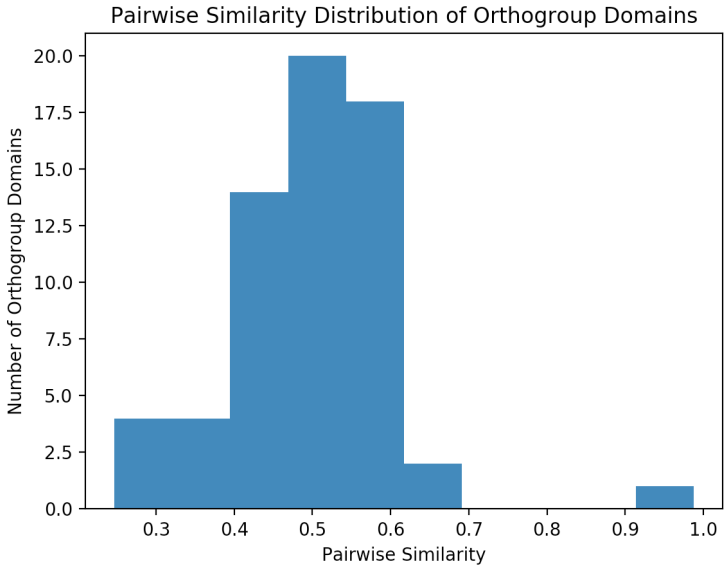

Figure 13: Distribution of pairwise similarities between OrthoDB domains

When examining the distributions of simulated pairwise similarities and OrthoDB pairwise similarities for domains, we can interpret the results by examining them through a stochastic viewpoint. Consider the case of DNA. DNA accumulates random mutations as over time as a result of a plethora of reasons (copying errors, UV radiation damage). This process can be modeled as a set of random events, in this case mutations across a sequence, quite similar to the balls and bins problem, a classic probability theory problem where balls are placed in random bins. We can similarly apply this stochastic model to understanding our results. In domain regions of amino acid sequences, specific functional positions are more highly conserved than others. Any mutations to those functional positions would kill an organism, and therefore among surviving organisms, there must be a large degree of conservation at those positions. Instead of being a random "balls and bins" process, these conservation processes are accurately reflected in the HMM file, which is translated by the simulation model by the similarity distribution. Therefore, there exists a "threshold" value of

conserved amino acids which are preserved during a protein domain (like zinc finger's) evolutionary history; this is revealed in both of the distributions in the form of a large proportion of orthogroup domains grouped around the same pairwise similarity. In the simulation model, this directly explains the large uptick in the distribution of pairwise similarity of simulated domains around 0.2, and in the case of the OrthoDB dataset, the relatively clustering of pairwise similarities around 0.5. Based on these assessments, we can conclude that this simulation model is able to create realistic internally similar orthologous groups from on a valid zinc finger protein sequence.

6. Conclusion

The simulation model developed in this independent work provides a new tool for investigating the procedures by which zinc finger protein domains evolve, supplementing previous algorithms for inferring phylogenetic histories with a model which accurately considers sequence conservation, as well as host and domain level events which occur during the phylogeny. The data from the pairwise sequence similarities do show that the simulation can generate accurate orthologous groups for zinc fingers when provided a host tree and guest tree topology, as well as an HMM file. In addition, by examining the clustering of data in the the pairwise sequence similarities for domains, we observe patterns consistent with evolutionary conservation of specific amino acids within the protein domain. Thus, the simulation model developed during this independent work is able to faithfully simulate domain level evolution for critical sequences of zinc fingers. This simulation is an important step towards building and testing new types of algorithms which can reveal more information about zinc fingers. The future goals of this independent work are to expand the protein domain simulation beyond zinc fingers to other functionally important transcription factors and protein domains whose structure or evolutionary behavior differs from that of zinc fingers. In addition, the current simulation model only accounts for duplication, loss, and speciation events, and adding support for more complex domain behavior, including but not limited to lateral domain transfers, domain merges, and domain splits, would further increase the accuracy of the simulated orthogroup results.

7. Acknowledgements

I would like to thank my advisors Mona Singh and Chaitanya Aluru for their thoughtful advice and help throughout this semester and my independent work. Thank you for consistently challenging me throughout the semester with interesting research questions!

I pledge my honor that this paper represents my own work in accordance with University regulations.

References

[1] "Birth Death Process," https://lukejharmon.github.io/pcm/chapter10_birthdeath/#section-10.2-the-birth-death-model4, accessed: 2019-04-12.

[2] "HMMER Documentation," http://eddylab.org/software/hmmer3/3.1b2/Userguide.pdf, accessed: 2019-04-10.

[3] "Pfam HMM Logo," https://pfam.xfam.org/family/PF00096#tabview=tab4, accessed: 2019-04-06.

[4] "Reconciliation Diagrams," Accessed from Chaitanya Aluru, accessed: 2019-04-19.

[5] M. A. Andrade, C. Perez-Iratxeta, and C. P. Ponting, "Protein repeats: structures, functions, and evolution," *Journal of structural biology*, vol. 134, no. 2-3, pp. 117–131, 2001.

[6] G. Apic, J. Gough, and S. A. Teichmann, "Domain combinations in archaeal, eubacterial and eukaryotic proteomes," *Journal of molecular biology*, vol. 310, no. 2, pp. 311–325, 2001.

[7] M. Arenas, "Trends in substitution models of molecular evolution," *Frontiers in genetics*, vol. 6, p. 319, 2015.

[8] Å. K. Björklund, D. Ekman, and A. Elofsson, "Expansion of protein domain repeats," *PLoS computational biology*, vol. 2, no. 8, p. e114, 2006.

[9] P. Djian, "Evolution of simple repeats in dna and their relation to human disease," *Cell*, vol. 94, no. 2, pp. 155–160, 1998.

[10] R. O. Emerson and J. H. Thomas, "Adaptive evolution in zinc finger transcription factors," *PLoS genetics*, vol. 5, no. 1, p. e1000325, 2009.

[11] S. Huntley *et al.*, "A comprehensive catalog of human krab-associated zinc finger genes: insights into the evolutionary history of a large family of transcriptional repressors," *Genome research*, vol. 16, no. 5, pp. 669–677, 2006.

[12] T. Koestler, A. v. Haeseler, and I. Ebersberger, "Revolver: modeling sequence evolution under domain constraints," *Molecular biology and evolution*, vol. 29, no. 9, pp. 2133–2145, 2012.

[13] C. Looman *et al.*, "Krab zinc finger proteins: an analysis of the molecular mechanisms governing their increase in numbers and complexity during evolution," *Molecular biology and evolution*, vol. 19, no. 12, pp. 2118–2130, 2002.

[14] E. M. Marcotte *et al.*, "A census of protein repeats," *Journal of molecular biology*, vol. 293, no. 1, pp. 151–160, 1999.

[15] J. Sjöstrand *et al.*, "Genphylodata: realistic simulation of gene family evolution," *BMC bioinformatics*, vol. 14, no. 1, p. 209, 2013.

[16] S. J. Spielman and C. O. Wilke, "Pyvolve: a flexible python module for simulating sequences along phylogenies," *PloS one*, vol. 10, no. 9, p. e0139047, 2015.

[17] J. M. Vaquerizas *et al.*, "A census of human transcription factors: function, expression and evolution," *Nature Reviews Genetics*, vol. 10, no. 4, p. 252, 2009.

[18] T. J. Wheeler, J. Clements, and R. D. Finn, "Skylign: a tool for creating informative, interactive logos representing sequence alignments and profile hidden markov models," *BMC bioinformatics*, vol. 15, no. 1, p. 7, 2014.