Limitations of Genomic Analysis on Novel Species

Luis Contreras-Orendain

Advisor: Sara Mathieson

Computer Science Senior Thesis

Spring 2021

Abstract

For widely studied species such as humans, fruit flies, and mice, there are many sequenced genomes, but for novel species, only a few or a singular processed genome is available. Being able to study novel species is important to understand their environmental impact, in the case of invasive species, or their genetic relation to other species but they pose the greatest difficulty to study. Genetic sequences are created using assemblers and assembling the genome for a diploid species is a computationally complex task which is why diploid assemblers create a phased collapsed genome that contains similar genetic information. Applications of Pairwise Sequential Markovian Coalescent (PSMC) modeling for population size inference and Phylogenetic tree generation for building a species family tree become more difficult with novel species and it is not clear how to proceed. Other tools exist, such as Read Mapping and NCBI's Blast, that provide the initial steps to the first two tools mentioned but are not well integrated with them. As a proof of concept, we applied Read Mapping with PSMC analysis and Blast with Phylogenetic tree construction on the novel species, the Spotted Lanternfly, to investigate the feasibility of these tools on a phased genome. At least for this genome, our analysis shows there to be significant limitations due to computational run time and with the processed output of our pipeline. More work is needed to better integrate various tools to analyze novel species.

Contents

Introduction

Genomic Analysis Techniques

Understanding how a genome is constructed is important in recognizing the difficulties in performing genomic analysis on a novel species. Raw genetic material is extracted and processed in a lab, creating sequences of random lengths called *reads*. The assembler takes these reads as input, processed them to be even lengths, and inserts them into a data structure called *a de Bruijn graph*. After handling bubbles and repeated nodes in the graph, the genome is assembled by traversing the graph [1]. Many species of interest are diploid, contain genetic information from both parents, and it should be assumed that a diploid assembler would output two genomes, one for each parent. However, diploid assemblers create a *phased* genome where the two parent genomes are collapsed onto each other to reduce the complexity of the already complex task [2].

With processed genomes, Pairwise Sequential Markovian Coalescent (PSMC) modeling can be applied that compares two or more genomes for instances of variation, *coalescent events*, that can be interpreted for population size over time. An example of PSMC is shown in Figure 3 on Neanderthal DNA with respect to other humans. PSMC has been shown to work with a high level of accuracy, an example being the extinction date for Neanderthals in the Figure agreeing with archaeological evidence [3]. With novel species, specifically diploid, applying PSMC to them would not be an issue since there are two genomes within the diploid genome. However, because of the phasing nature of diploid assemblers, there is only one and PSMC becomes inapplicable to novel species in this format.

As a workaround to PSMC's minimum genome requirements, the abundant number of reads used in the assembly process is leveraged in *Read Mapping* to calculate variation cites. Read Mapping takes each read and finds where in the genome they overlap completely. With several reads overlapping a specific region as in Fig. 1, significant variation is noted whereas very minor or no variation is ignored. Read

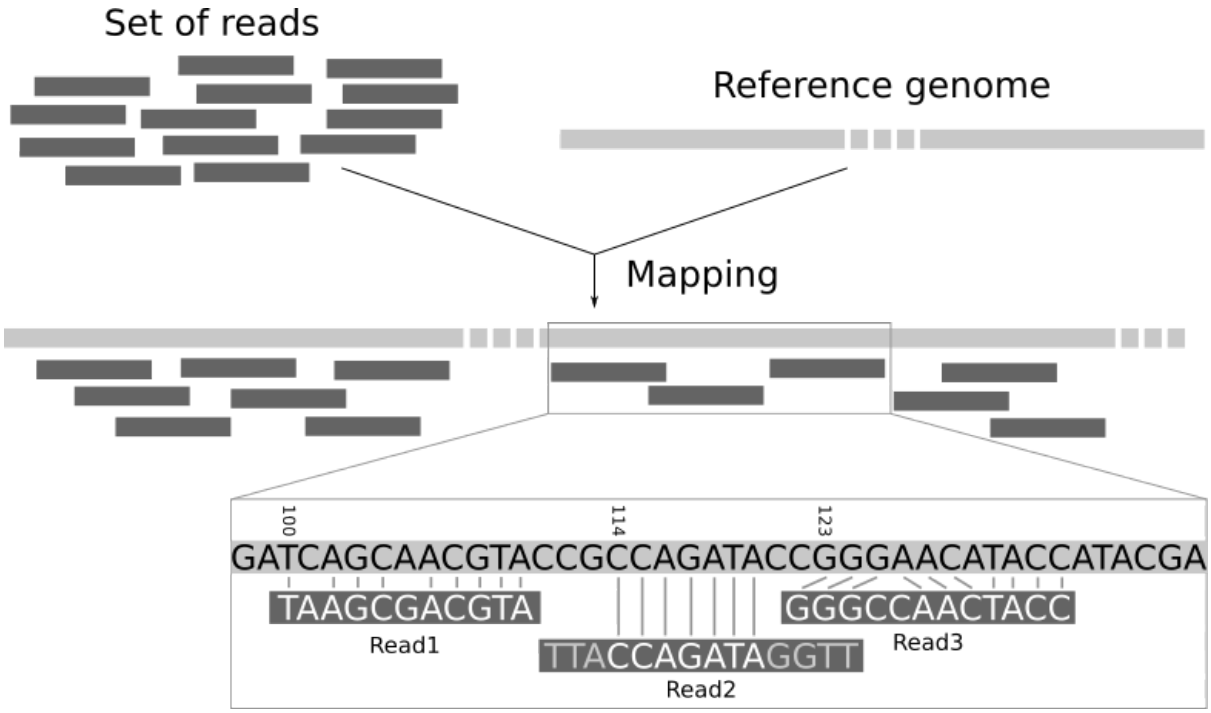


Figure 1: A big picture overview of Mapping a set of Reads to a Reference Genome [4].

Mapping thus performs a similar function as PSMC's multi-genome comparison and the variation cites can be used to analyze population size over time.

Another method of analyzing novel species is figuring out a measure of how different that species is to others. This is done using the Neighbor-Joining (NJ) algorithm to create Phylogenetic Trees. To find these species to perform Phylogenetic Tree generation, the National Center for Biotechnology Information's (NCBI) Basic local Alignment Search Tool (Blast) is used which searches its vast nucleotide database for genomes that are comparable to the target genome and returns genomes that share sufficient overlap. With those genomes, the similar species are identified.

Using a genome from each species, they are compared and given a rating of how close they match. The NJ algorithm then greedily selects the two genomes with the highest comparison rating and joins them together to a node. The algorithm will treat that node as a representation of the two genomes and carries its own comparison rating. It continues until a tree is created where one can visually mark ancestor species, sibling species, neighboring species, etc. This representation is useful in finding similar species

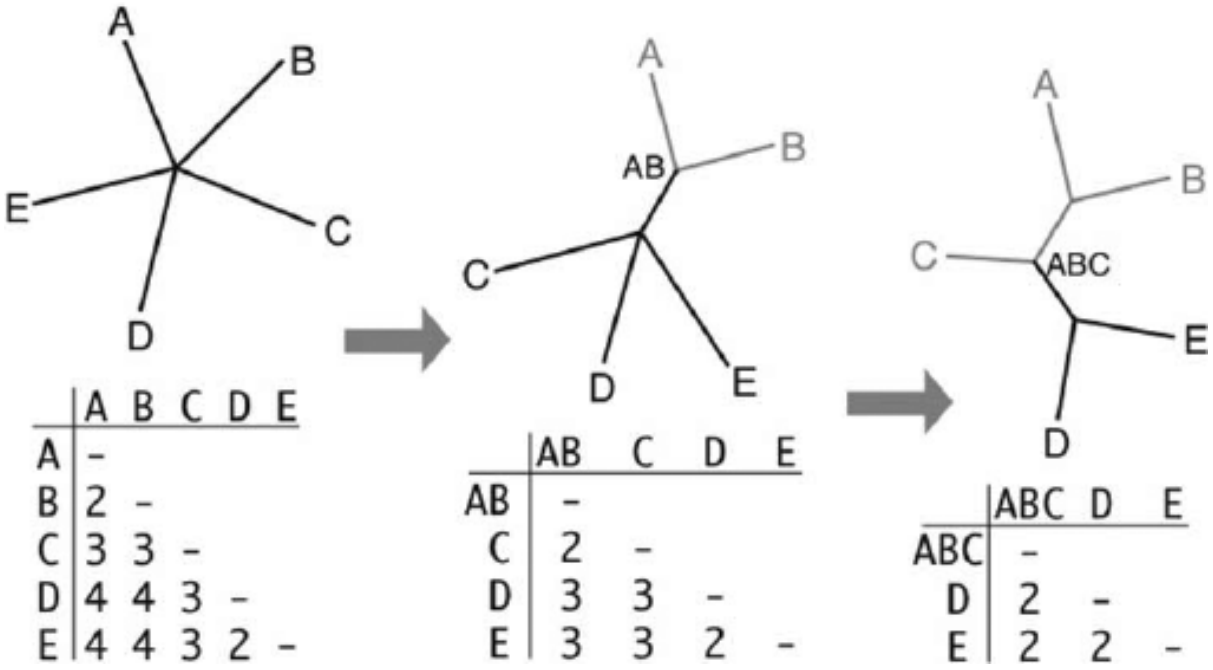


Figure 2: An example application of Neighbor Joining to an unrooted tree. In the tables below, it picks the two closest nodes (like A and B) to link up together into a new node (node AB). It will continue until all nodes have been paired and the tree is rooted [5].

that are better studied than the novel species. The information can be carried over and applied to the novel species to the extent that they are similar.

Novel Species of Interest: Spotted Lanternfly

Spotted Lanternflies (SLF) were first spotted in Eastern Pennsylvania back in 2014, being transported from South Korea on a trading ship [6]. Their invasiveness has the potential to cripple many local ecosystems and damage many businesses. Their ability to adapt to new food sources when they have over-consumed gives them the quality to be a very dangerous invasive species. The more they spread, the more they cause this negative feedback loop to seek more plant species to devour. Figuring out how to control them is imperative. Currently, the best traps being employed are developed from traps used on a different insect, as they both share one key behavior: they always tend to move upwards [6]. These traps utilize this behavior to ensnare them as they move up a tree and immobilize them until they starve themselves to death.

By just noting one key behavior, innovative traps have been used that mostly affect SLFs and leave others relatively unharmed. Using PSMC modeling on an SLF

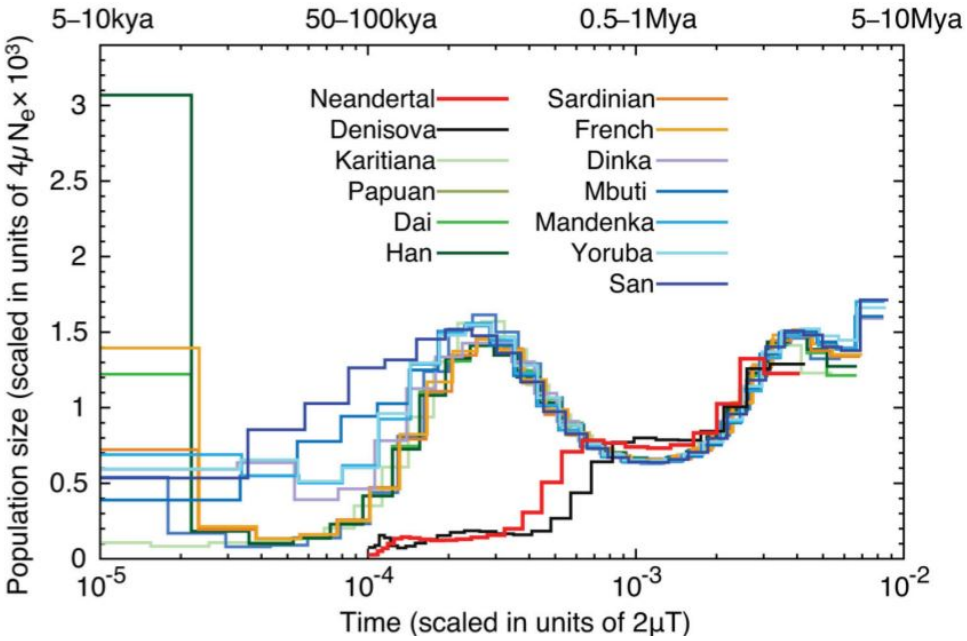


Figure 3: A PSMC application to show population sizes over time of various hominid populations. Note the time scale flows backwards. The red Neanderthal and black Denisova lines reach zero around 50-100kya which is corroborated with archaeological data [3].

genome would corroborate the initial sighting and growth of the SLF. Should multiple genomes be available for SLFs in regions of interest (such as China, South Korea, United States), a comparison of their PSMC output would illustrate their population size variation over time and indicate when there were migration events, agreeing with current timescales. Population growth or decay would also corroborate the rate at which SLFs acquire new sources of nutrition and provide a measure of how destructive it is being in its environment. Using Phylogenetic trees, other species like the SLF can be analyzed for similar behaviors and for discovering possible prevention techniques.

For widely studied species, many tools exist to analyze it from a genomic perspective. Much of the literature is focused on using the abundant data available for certain species. As such, many genomic analysis tools are tailored towards utilizing sufficient data. Issues arise when trying to analyze novel species using the same tools beginning from the moment the genome is assembled to using analysis algorithms that is not accounted for well in the literature. I will explore the currently available analysis tools that would be ideal to study novel species but can't be used in their current form. I

will delve into potential supplemental methods that could allow those tools to be usable on novel species and postulate ways to combine them.

Literature Review

Assembling a Genome

DNA cannot be extracted from an organism in its entirety because of the sequencing process. Instead, short chunks, called 'reads,' are extracted and must be assembled into the genome using tools called Assembler Algorithms. The common method assemblers use is taking the short reads that are variable in length and constructs them into *k*-mers , lengths of DNA that are fixed at the input value *k*, using them in a graph data structure to form into contiguous sequences (contigs). This method is called *de novo*, Latin for 'from new', because they do not use another genome to aid in the assembly process. They start from scratch, and why the name de novo is applied. *Velvet* is an example of a de novo genome assemblers.

Velvet contains a set of assembler algorithms that improved on previous algorithms by utilizing high amounts of short reads (25-50 base pairs in length) that would cover the end-result genome well [1]. *Velvet* will output the set of contiguous sequences (contigs) that represent the genome covered by the multitude of reads. Since the reads significantly cover the final genome, when they are converted to *k*-mers , there will be a significant amount of overlap among them since they will represent the same region in the genome. To handle this overlap and reconcile the *k*-mers into the genome-representing contigs, the *k*-mers are read into a de Bruijn graph. Each node in the graph contains information about several reads, displaying only the last nucleotide base in the read. For example, the group {TAGAC, AGACT, GACTG} would display 'CTG' and are grouped because they all overlap. They would be connected to another node with the group {ACTGA, CTGAT, TGATT, GATTG} and display 'ATTG.' They are connected since the last in the first group, 'GACTG', overlaps with the first in the second group, 'ACTGA' with the 'ACTG' portion. This example is shown in the graph in Figure 4. Once *Velvet* has constructed the graph, it will carry out the next two stages out of 4: correcting errors, and resolving repeats. Error analysis is done by using topological

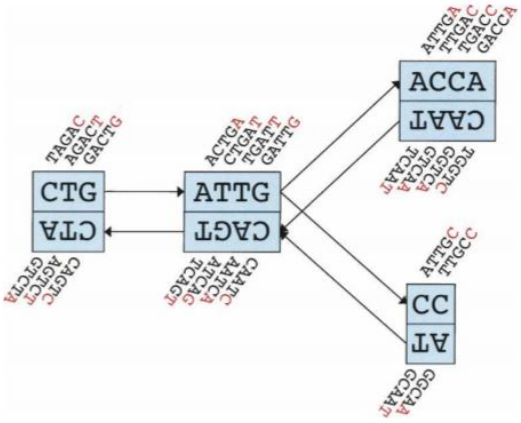


Figure 4: Example section of a de Bruijn graph used in *Velvet* assembler algorithms [1]. Each node has a sequence that represents the last nucleotide base of a in a group and is connected to other nodes that overlap with the node sequence.

analysis on the graph where the data creates strange features in the graph. Repetition is resolved by separating paths that share overlap, marking those nodes and eliminating them.

Each stage of *Velvet* has different algorithms and computational requirements. Constructing and using the de Bruijn graph is one of the main obstacles since there are so many *k*-mers that must be inserted and connected [1]. As an example, a simple bacteria could have a de Bruijn graph that is 2 Gb in size. Once it is constructed, bubbles are handled using the Tour Bus algorithm, which utilizes Dijkstra's Algorithm. An example of a bubble is shown in (A) in Fig. 5. The Tour Bus algorithm will identify a bubble as starting in the same node (node B) and ending up in the same node (node E) by taking different paths (following C-D or C"-D'), and so B-C-D-E-C"-D' is a bubble. The algorithm aligns the inner nodes, since they would be very similar to form the bubble, and merges them together, effectively removing the bubble. The algorithm continues until there are no more bubbles in the graph. Since it is based on Dijkstra's algorithm, Tour Bus has a time complexity of $O(N\log(N))$ with N being the number of nodes in the de Bruijn graph, which is dependent on the number of *k*-mers [1].

Besides bubbles, repeating nodes in the graph is another issue. Repeats are handled using the Breadcrumb algorithms and is dependent on paired-end reads. These reads have been sequenced in the forward and backward direction, producing very two

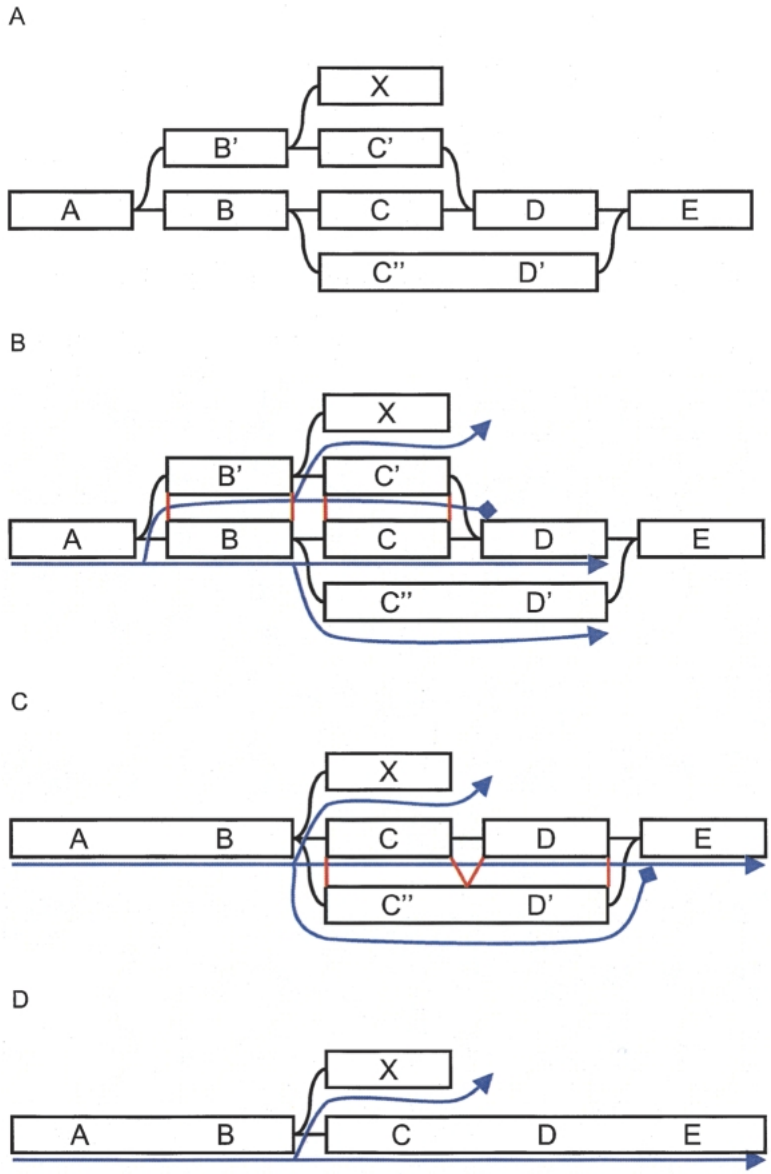


Figure 5: Application of the Tour Bus Algorithm on part of a de Bruijn graph to remove the two bubbles containing the nodes A-B-B'-C'-D and B-C-C"-D-D'-E. [1]

similar reads (denoted here as read A, B) and where the name paired-end comes from. They are very useful in repetition analysis. A user cutoff is then used to select nodes longer than that cutoff that have only a few A or B reads from paired-end reads. Of those reads, the algorithm flags all the other nodes that contains its counterpart and marks them with 'breadcrumbs.' These breadcrumbs are used to create a subset of the graph which is easier to analyze. Starting in one of the long nodes, the algorithm follows the flagged nodes and tries to go as far as possible to another long nodes. In this best case, the two long nodes are merged. Otherwise, the algorithm continues with the next

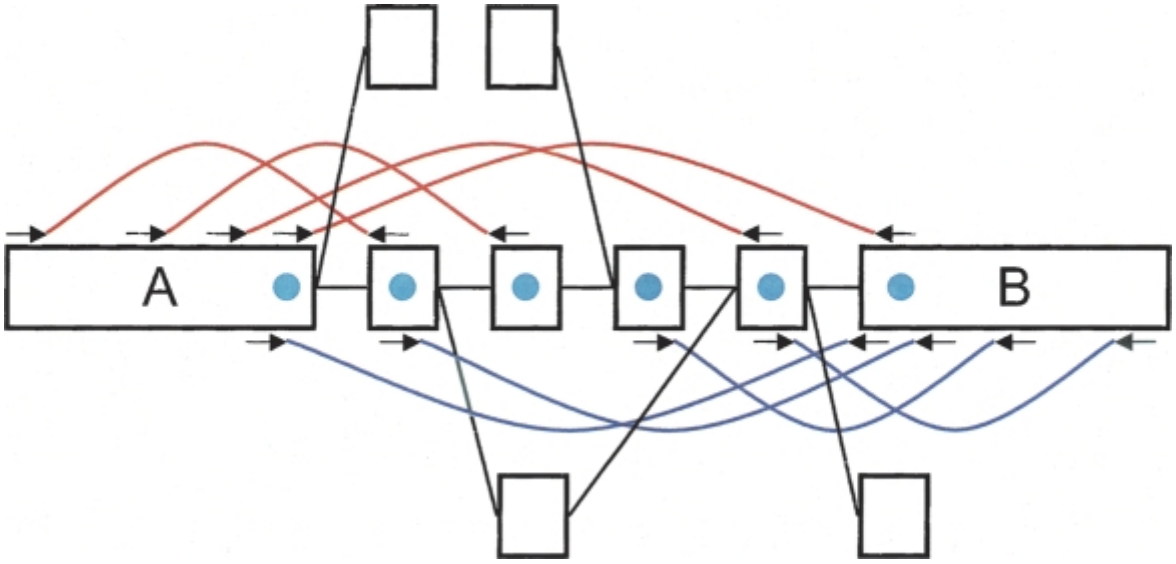


Figure 6: Example application of the Breadcrumb Algorithm. Nodes *A* and *B* are contigs that are connected by several reads indicated by the red and blue arcs. These contigs are connected through other contigs which are marked by the blue circle (breadcrumb). A subgraph is created using the breadcrumbs that is easier to explore and reduce repeats.

path or node until all possible paths and nodes have been processed [1]. In comparison to other assemblers at the time it was released, *Velvet* uses more storage space in order to improve how errors in the graph are dealt with, thus speeding it up and improving efficiency [1].

In using graph traversal to sequence the genome, *Velvet* will not produce a true diploid genome. That was not one of the main considerations in selecting data structures or algorithms. *Velvet* is best suited for Haploid genomes. For a diploid assembler that utilized similar methods, there is the *Falcon* assembler. *Falcon* still creates a de Bruijn graph using k-mers constructed from reads as nodes and forms the connections based on the way those k-mers overlap [2]. As it iterates through the graph to do the assembly, *Falcon* removes bubbles, which it considers as 'haplotigs.' Haplotigs are contigs that contain haplotype information, which is inherited from only one parent. One of the haplotigs that make up the bubble is removed and what remains is called the primary contig. This primary contig will contain information from both haplotypes because of overlapping and the non-overlapping bubble regions are returned as haplotigs. The bubble removal and partial assembly process is shown in Fig. 7.

With these alterations, the *Falcon* assembler captures the diploid nature of the species being sequenced and does not lose most of the information like *Velvet* does.

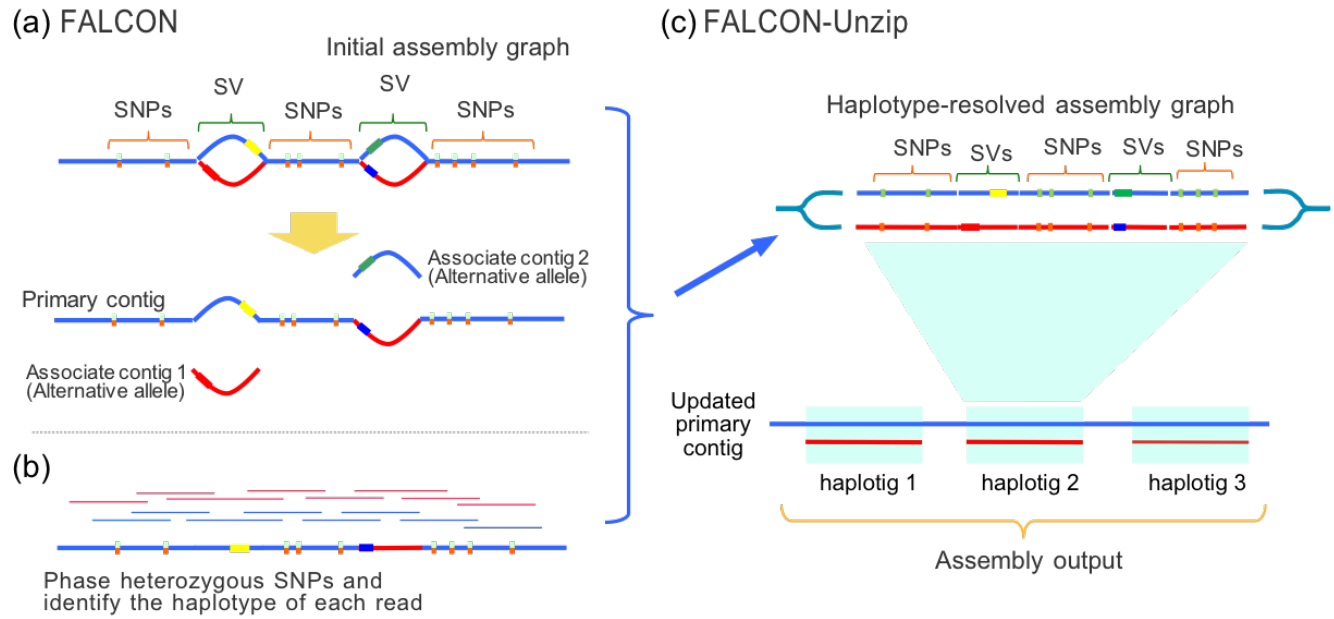


Figure 7: Falcon assembler collapsing the diploid genome into a phased genome, removing areas of structural variation (SV) into separate haplotigs. [2]

Inferring population size from the genome using PSMC

One initial step to start exploring a novel species, after its genome has been assembled, is an application of Pairwise Sequentially Markovian Coalescent (PSMC) modeling. PSMC uses the genome to correlate population size changes over time. PSMC is based on coalescent theory, a model of how gene variation could have occurred due to a common ancestor and is implemented using the Hidden Markov Models (HMM) data structure [7].

A Hidden Markov Model is a statistical model that assumes a set of 'hidden states' in which a state Y depends on a previous state X. There are also observable states corresponding to a specific time, and the hidden states correspond to that observable state. In Fig. 9, the location of S_l and S_{l+1} would represent observable states and the i squares represent the hidden states. An example coalescent event happening at a specific

observable state S_l could cause the hidden state to go from i to $i+1$ at observable state S_{l+1}.

For PSMC more specifically, observable states represent sections in the diploid genome and hidden states correspond to pre-defined Time to Most Recent Common Ancestor (TMRCA) categories, such as 5, 10, 15 years ago. PSMC is given the diploid genome as input. With each of the two genomes in the diploid pair, they are given to an HMM constructed in this way. As the PSMC-HMM proceeds through the genome, it will encounter coalescent events at observable states causing it to move to different hidden states. Tracking the path the HMM takes, a plot of coalescent events over time can be created as in Fig. 3.

Applying PSMC to multiple genomes of the same species can yield statistics on the number of coalescent events at a specific point in time. One such application is shown in the heat map in Fig. 8 where the average timeline is in red. The heat map helps indicate that the PSMC data is accurate and not overly affected by minor variations. If the TMRCA is big for a given coalescent event, this indicates the population size at that event is big, it had to go further back in time to find a common ancestor. Similarly, a low TMRCA indicates a small population size. Using these correlations can include extinction events, demonstrate bottle necking, and show changes in population over time as well as determining if two related genomes diverse or converge to each other [7].

PSMC is an especially useful tool, especially in terms of supporting the time scale of events as predicted using other tools. For example, the PSMC application on Neanderthal data in Fig. 3 shows their extinction event at 50-100 thousand years ago. This result agrees with what archaeologists and other researchers have proposed. These findings also suggest that several gene flow events (when two separate populations managed to mate with each other) happened between different populations of human species, which is suggested by lower levels of coalescent events [3]. Since there are lower levels of coalescent events in both populations of human species, it indicates that there were fewer common ancestors and thus the population must have been bigger.

These findings aid in understanding how humans evolved over time and what factors lead to that evolution, and the same methodology can be applied to other species of interest to note inter-relatedness.

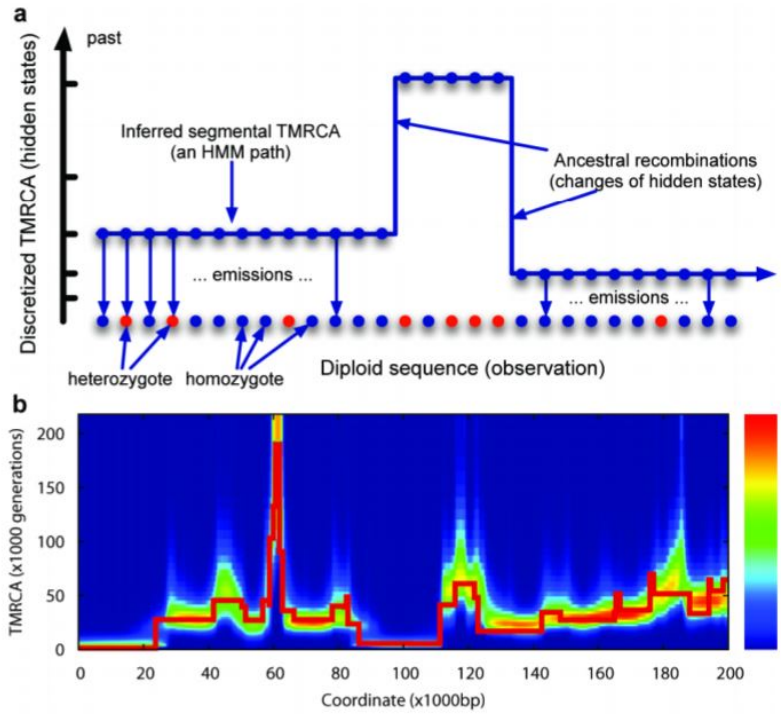


Figure 8: Illustration of PSMC with relation to the HMM. As the model moves through observations in the genome, a coalescent event cause the model to change in hidden states (a). With the transition to certain hidden state, the time to most recent common ancestor can be inferred with which the population size can then be inferred (b) [7].

However, as more TMRCA categories are defined to improve accuracy or increasing the number of coalescent events states will increase the time complexity of running PSMC, scaling exponentially due to the vast number of possibilities that can occur in the HMM. [8]. The overall running time of an HMM is $O(NM)$ with N being the number of hidden states and M being the number of observable states. If they are similar enough, this becomes quadratic in time which becomes computationally infeasible for large quantities. Luckily, there is a linear-time implementation in the number of TMRCA categories.

The HMM can be thought of as a matrix with dimensions based on the amount of TMRCA categories and the number of coalescent states. Algorithms perform calculations on that matrix, having to iterate through it to perform the state transitions. This is

why we get that quadratic time complexity. This is reduced to a linear time complexity by noting several key things. Calculating the transition probability from one state to another repeats which can be captured in explicit formulas for given states [8]. Additionally, the matrix has a few symmetrical properties caused by the coalescent data the HMM is being applied on which can be exploited to create other explicit formulas. Several other mathematical improvements are made that bring the run time down to linear time such as inserting intermediary hidden states between observable states. A visual representation of these improvements is seen in Figure 9. Each interval in the HMM is augmented with three floating states that capture the dependence of recombination and coalescent events. It could be the case that certain events should not occur during a specific transition in hidden states, so these floating states handle that by transitioning the probability to another separate hidden state. With these improvements, algorithms that depend on coalescent theory, such as PSMC, see vast improvements and frees up resources to be used in other assumptions the models make to make it computationally feasible [8].

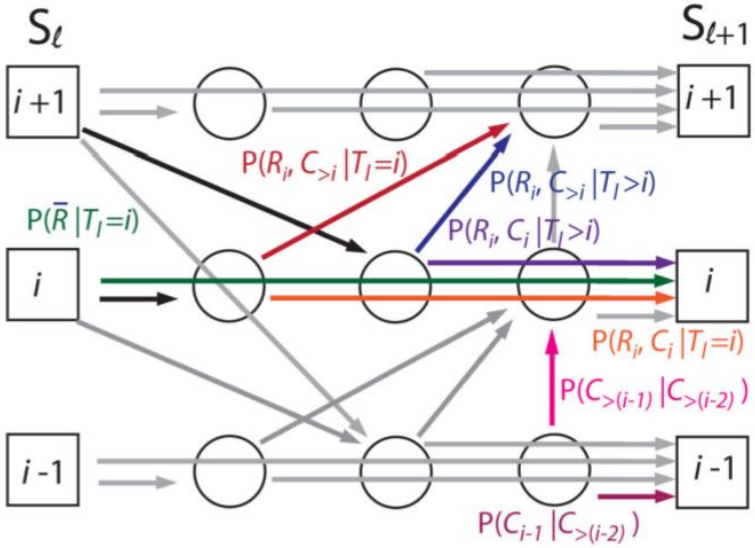


Figure 9: HMM example that has been augmented for PSMC with the hidden states being i nodes and the observable states being S_ℓ [8]. The intermediary circles represent the augmentation and capture the recombination and coalescent probabilities.

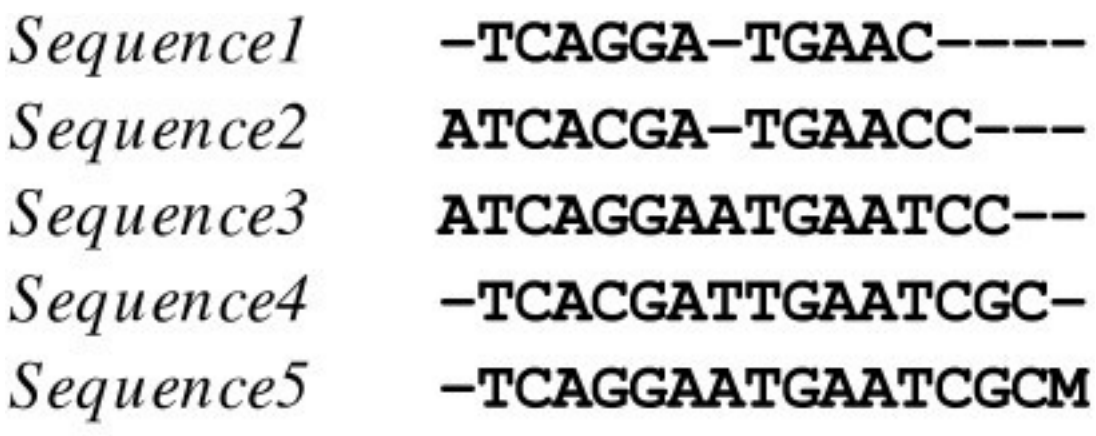


Figure 10: Multiple sequence alignment example where several reads are mapped onto each other. The dashed lines are added in order to get a more exact overlap [10].

Phylogenetic Trees

In addition to using PSMC to analyze population size changes over time, situating a species with relation to other neighboring species is also important in considering how species diverge from each other. Using evolutionary distance-data based on coalescent events, a tree is created using the Neighbor-Joining algorithm that minimizes the total length between neighbors and their common ancestors. Doing so displays a tree of ancestors to children and how closely they are related genetically as shown in Figure 11 [9]. More recently, this process has been done using two algorithms: ClustalW and RapidNJ.

ClustalW is a multiple sequence alignment program for DNA or proteins. There are four steps to do this alignment: Pairwise Sequence Alignment; Neighbor-Joining (NJ) Method; Guide Tree Construction; and Global Alignment Generation [11]. In the first step, a distance matrix is computed containing a measure of how close one sequence is to another. The distance matrix is used to create an unrooted tree using the Neighbor-Joining method. We'll discuss NJ more with RapidNJ. The tree is then used as a guide with which RapidNJ can create structure and a root.

The overall time-complexity for ClustalW is $O(N^2)$ as given by the NJ method. ClustalW outputs distance data based on those alignments stored in the distance matrix which is then used by RapidNJ. It is especially suited for large datasets, where the stan-

dard NJ method becomes less effective because of the time-complexity, by implementing a search heuristic that speeds up selecting which two sequences to connect to a node.

NJ joins sequences and nodes together using a greedy approach until all sequences have been added to the tree. The NJ algorithm selects the two sequences i and j with the smallest branch length by using

$$Q(i,j) = D(i,j) - u(i) - u(j) \tag{1}$$

and creates a node connecting them together. $D(i,j)$ is the distance between node i and node j and $u(i)$ is the average distance to the other nodes in the tree given as

$$u(l) = \sum_{k=0}^{r-1} D(l,k)/(r-2) \tag{2}$$

where r is the number of nodes left to be processed. A new node ij is created and connected to nodes i and j. The distance from those two sequences to the new node is calculated and set to the node value to be used in future iterations of the algorithm [12]. The NJ algorithm continues iterating through all the sequences in such a greedy fashion until no new sequences are left to be added to the tree. RapidNJ improves on this by decreasing the time to select the two shortest distance nodes to join using two additional matrix data structures storing the sorted information from the distance array and a map to go from that sorted information back to the location in the distance matrix. The entries in these arrays are only calculated if they need to be and RapidNJ avoids searching the entire list of combinations to find the two nodes with the shortest distance.

Along with showing those distance relationships, phylogenetic trees allow us to learn more about the evolution of a species, to make predictions about novel species, to learn about the diversification of that species [13]. For example, we could have a new organism that we would like to know more about. Situating it in a phylogenetic tree using its genome will show us which organisms it is similar to and by how much. The information known about the other organism can then be transferred and applied to the new organism, subject to how much they overlap.

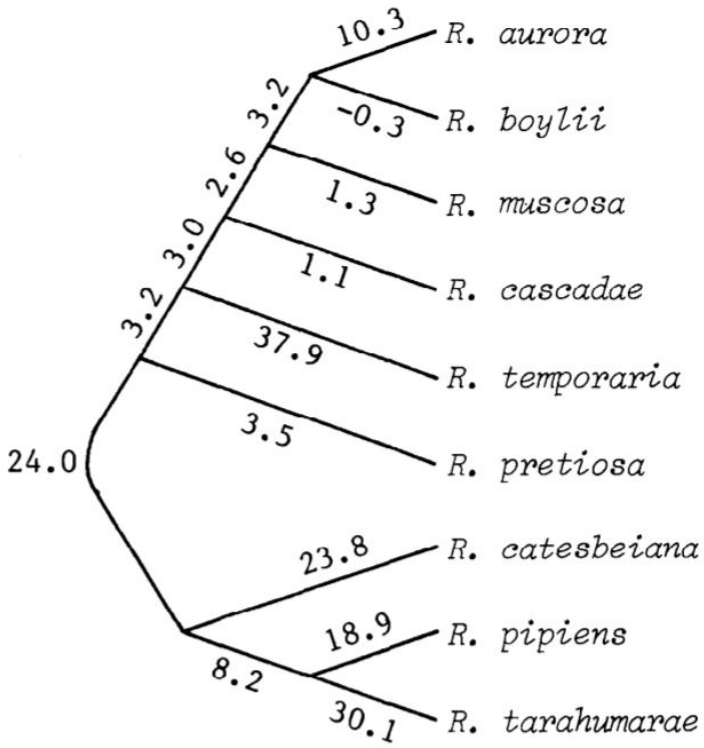


Figure 11: Phylogenetic Tree example for immunological data. In this example, the numbers indicate the measure of support for that connection in the tree, with higher numbers indicating more support [9]. They do not indicate the distance between nodes.

Limitations of Genomic analysis with novel species

Assemblers have been applied to a wide range of species for genomic analysis. One of the most popular studied animals is the common fruit fly with a plethora of sequenced genomes available for further study. Given the organism's simplicity and the relative size of its genome, studying the fruit fly using genomic analysis algorithms is very feasible and is the ideal. It is also very computationally feasible to check the correctness across many assemblers as well as being able to quickly implement and test improvements to the algorithms [14]. However, on the other end of the spectrum of data availability, species that are new to genetic study, so called novel species, often only have a couple or a singular assembled genome. The algorithms described, such as PSMC and Phylogenetic Trees work well for species that have at least two sequenced genomes. This poses a difficulty in analyzing novel species.

Besides the limited data availability, another consideration when dealing with novel species is the method of assembling the available diploid genome(s). The *Falcon* assembler is an example of this predicament. In the assembly process shown in Fig. 7, *Falcon* essentially collapses the two parts of the diploid genome from the parents into one continuous genome [2]. A true diploid assembler would produce two genomes for each part of the diploid genome. PSMC that needs two genomes to compare, so having one phased genome is not sufficient. An actual diploid genome, that contains a genome for each of the two parents, can be used with PSMC since one part of the diploid genome is compared to the other.

There also exists the case where it is difficult to place the novel species in relation with other species. Creating a Phylogenetic trees finds that relation and describes it graphically, but the genomes of related species are needed to use with ClustalW and RapidNJ. Finding those genomes becomes an issue when not a lot is known about the species to begin with, so creating a Phylogenetic tree becomes more difficult.

Supplemental tools for use on Novel Species

At PSMC's core, it is comparing each half of the diploid genome with the other for cites of variation. These variations are called single-nucleotide variants (SNV) because the genomes differ at specific sites. For example, a subset of one of the diploid genome could be 'ACG' and the other half could be 'TGA' where *G* and *A* do not match since *A* always pairs with *T*, *C* always pairs with *G*.

Another method of computing SNVs is called Read Mapping, with an example pipeline shown in Fig. 12. For a novel species with a phased genome produced by an assembler such as *Falcon*, Read Mapping can be applied on the phased genome to produce all of the SNV cites. An example of a Read Mapping pipeline consists of using the Burrows-Wheeler Aligner (BWA) software package along with the BCFtools utilities. BWA first creates an index of the reference genome (in this case, the phased genome) that increasing efficiency in terms of searching the genome during alignment later on [15]. Using the index and the reads initially used to create the reference genome,

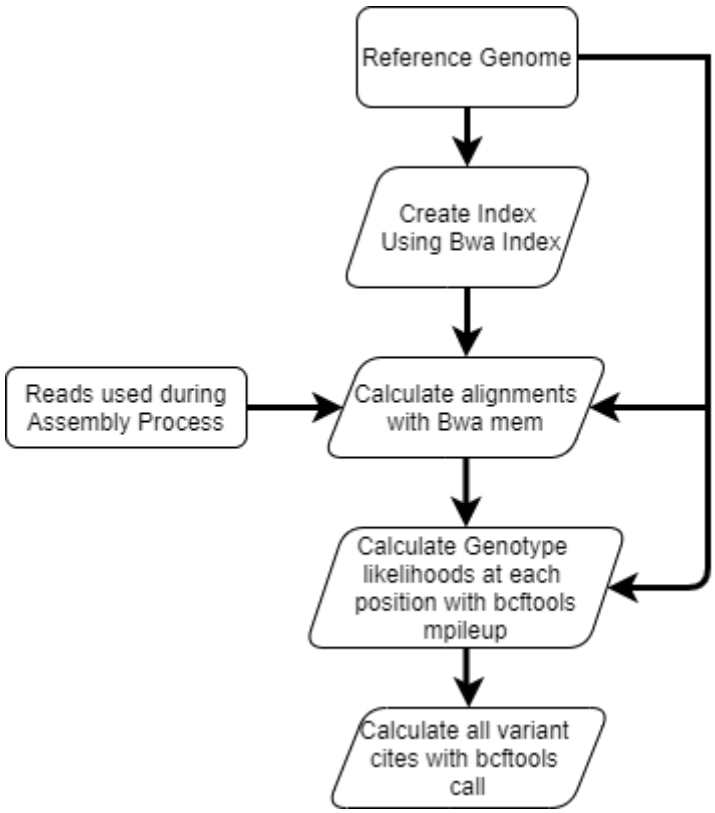


Figure 12: Read Mapping pipeline that incorporates the available reference genomes and reads in the BWA and BCFtools functions. Created using https://app.diagrams.net/

BWA then aligns the reads to the reference genome using the index to speed up the process. This process doesn't match the entire read to its part of the genome, as we see with Read2 in Fig. 1, because there may be any number of insertions, deletions, or mismatches. Those variations then go on to be the SNVs that would be parsed using BCFtools.

BCFtools is a set of utilities for parsing and manipulating variants in specific file formats. SNVS are stored in Variant Call Format (VCF) and a binary version called BCF [16]. BCFtools mpileup is applied on the alignment from BWA which produces a VCF or BCF file containing a summary of the coverage of the mapped reads to the reference genome. Using BCFtools call completes the pipeline to output all the genomic variant cites in the reference genome.

Another very popular tool is call Basic Local Alignment Search Tool (Blast) by the National Center for Biotechnology Information (NCBI). NCBI hosts a wide array of databases, including a nucleotide (nt) database that would be of most interest with

novel species. Using Blast with a reference genome compares the reference genome to the many genomes within the nucleotide database and returns those genomes that have a high local alignment percentage as well as what species they are [17]. With any new sequence, using Blast is a good tool to find out more about it based on similarities with other species.

Problem Statement

In the east coast of the United States, a novel species first appeared in 2014 in Pennsylvania. The Spotted Lanternfy (SLF) arrived from Southeast Asia in a shipping container or some other form of cargo, as some have theorized [18]. It is extremely invasive and spread quickly once it was first detected. Moreover, it extremely difficult to control using conventional traps. One such trap has been effective, called the circle trap, and is utilized by analogy from another species that has a similar upward moving behavior as the SLF[1].

Given the pressing environmental issues and the fact that almost nothing is known about the Spotted Lanternfly, we propose an application of these genomic tools to create a PSMC plot and a Phylogenetic tree. This application will also highlight drawbacks in those algorithms with the current data available. To do this proposal, we'll also use the Read Mapping pipeline and Blast. Currently, PSMC and Phylogenetic tree generation do not incorporate some way to handle reduced input data. Creating a pipeline that utilizes the reference genome and the reads with those tools would be optimal.

[1] https://extension.psu.edu/how-to-build-a-new-style-spotted-lanternfly-circle-trap

Methods

The information gathered using genomic analysis tools would be most abundant with novel species. Yet, novel species carry lots of inherent issues such as limited starting data and limited working knowledge about them. Two tools, PSMC and Phylogenetic Trees, are best suited for the task of analyzing novel species as they already work with limited data. Using other supplemental tools, such as Blast and Read Mapping, extends the use of those tools to also be capable of handling species with little starting data. Here, we describe the implementation of those tools and how such a pipeline would work for a novel species such as the Spotted Lanternfly.

For these implementations, we utilized a machine equipped with an Intel i7-9700 CPU with 8 cores, with 64 Gb of RAM, and an Nvidia Quadro P2000 GPU. Many of these tools benefit from multi-threaded operations, so having 8 cores in the CPU vastly improves the running time of these operations. They also benefit from having a large amount of RAM to work with since that is faster to access than if it were stored in a Hard Disk Drive.

The Spotted Lanternfly that we will be using to test the implementation was downloaded using NCBI's SRA Toolkit, which is a command line tool that allows us to download from their databases using a reference called an accession number. The Phased Genome assembled by Kingan et al is available on a USDA database and must be downloaded manually [19]. The reads used to assemble the data is available under SRA accession number SRR9001434 [18]. These were downloaded locally, with the genome being 2.29 Gb in size and the reads being 263.71 Gb in size.

Measuring Variant cites using Read Mapping

The tools BWA and BCFtools are the most popular in the bioinformatics community [15][16]. They are constantly updated and both are open-source tools. Additionally, BWA and BCFtools are coded in C which provides greater efficiency when working with large datasets.

Bwa was installed from their Github[2] page. This creates a folder called BWA. We enter into that directory in a terminal and run 'make' to compile together all of the C files. BCFtools was installed similarly from their Github[3]. Bcftools utilizes a separate software tool called htslib that is also installed by cloning their Github[4] page. After running 'make' in each of the two created folders, the last step of the installation is adding the path to the files to the PATH variable so that the commands can be run from anywhere.

Creating the index of the reference genome was done by running `bwa index /Path/To/Ref_genome`. Read alignment was performed by running `bwa mem -t 8 /Path/To/Reads`. BWA index offered only two parameters, one for naming the output file and the other for choosing the algorithm type to create the index. Since our reference genome is bigger than 2 Gb, we chose to not select an Algorithm type as the alternative was a linear-time algorithm that works best with genomes smaller than 2 Gb. BWA mem had many parameters to affect the alignment process but all were left to pre-set values. Further tests can affect parameter choice to influence algorithm performance. We did choose the number of threads to use to be 8 which decreased the running time of the alignment and best utilized the resources of our machine.

Using the alignment file and the reference genome, we then turn to BCFtools to calculate genotype likelihoods at each position. We did this by running `bcftools mpileup -t 8 -o output_file -Ou -f Path/To/Reference_Genome Path/To/Alignment_File`. Not many parameters were chosen besides using 8 threads to perform the calculation. There are four file types that can be used: compressed and uncompressed BCF/VCF file types. The parameter -O(b/u/z/v) selects which file type, respectively.

Finally, the variant cites are calculated by running `BCFtools call -c -variants-only -Oz -o output_file Path/To/mpileup/Output`. The pa-

[2]https://github.com/lh3/bwa
[3]git://github.com/samtools/bcftools.git
[4]git://github.com/samtools/htslib.git

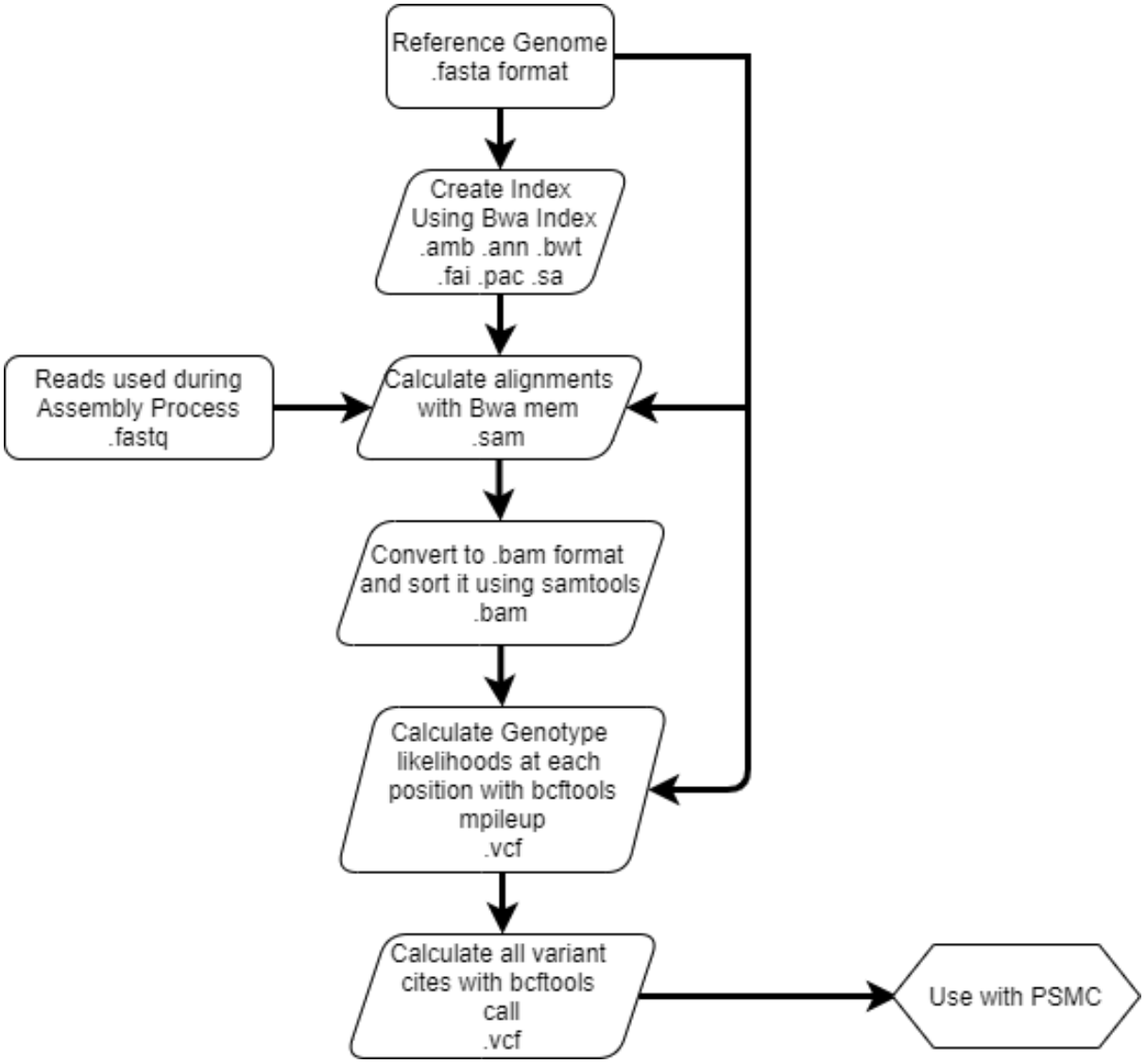


Figure 13: More detailed workflow to do Read Mapping using BWA and BCFtools software packages, showing the various file formats that are outputted. Also shows connection to PSMC. Created using https://app.diagrams.net/

rameter $-c$ chooses the type of consensus calling method (consensus calling being another way of saying what is the most likely nucleotide at each location in the genome).

With the output of BCFtools call, we have a file that tells us where all the variation cites are in the reference genome which we can use when applying PSMC to the genome.

Using Read Mapping in PSMC

PSMC is installed from the Github[5] repository. The installation is similar to BWA and BCFtools: run `make; (cd utils; make)` in the PSMC directory. Afterwards, the directory is added to the PATH variable. The process for producing a population size history plot is as follows: The consensus call file is converted to a PSMC specific format by running `utils/fq2psmcfa call_output > diploid.psmcfa`.

The HMM that PSMC utilizes is created, and calculations performed on, by running `psmc -N25 -t15 -r5 -p "4+25*2+4+6" -o diploid.psmc diploid.psmcfa`. The $-N$ parameter indicates. The $-t$ parameter gives the upper limit of the TMRCA, which in this case is 15, and is the number of hidden states in the HMM. The $-r$ command gives the mutation rate per generation which changes the scale of the population size plot and shifts it. The $-p$ parameter species the time intervals, representing the observable states in the HMM, and each interval has it's own population-size parameter that affects how it infers population-size. In this example, the parameters are separated by addition symbols and multiplication (such as X*Y) means that the next X parameters will span Y time intervals [20].

PSMC can be fine-tuned with the mutation rate parameter and the time interval parameters. Increasing the number of time intervals will give a more in-depth look at the number of coalescent events but will also increase the affect of genetic variations [20]. These parameters are manually worked with until the final PSMC output looks stabilized.

The population size history is inferred by running the command `utils/psmc2history.pl diploid.psmc | utils/history2ms.pl > ms-cmd.sh` and visualized as a plot by running `utils/psmc_plot.pl diploid diploid.psmc`

[5] https://github.com/lh3/psmc

Phylogenetic Tree Recreation Using Mitochondrial DNA

In the work done by Jeong et al [21], they created a Phylogenetic tree for the Spotted Lanternfly and several other species and species groups using their mitochondrial genomes. The genomes were downloaded from the NCBI Genbank database using the command line tool Entrez Direct[6] (Edirect). The species name and accession numbers are shown in Table 1. Each was downloaded by running the command `esearch -db nucleotide -query "[Accession Number]" | efetch -format fasta > /Path/To/Output/File.fasta`.

Once they have been downloaded, all of the mitochondrial genomes were concatenated into one fasta file to give as input into ClustalW. Both ClustalW and RapidNJ were utilized within Galaxy, an open source web-based platform for biomedical research [22]. Running these algorithms on Galaxy is similar to running on the command line with the added benefit of a user interface and an explanation of what all the parameters do. They can also be installed locally and run through the command line[7][8].

Name	Accession Number	Size (in Kb)
Delphacidae	MH352481	15.540
Ricania speculum	KX371891	15.036
Ricania marginalis	JN242415	15.985
Sivaloka damnosus	FJ360694	16.265
Geisha distinctissima	FJ230961	16265
Betatropis formosana	MH324927	16.456
Aphaena discolor nigrotibiata	MN025523	15.420
Aphaena (Callidepsa) amabilis	MN025522	16.529
Lycorma delicatula	EU909203	16.236
Lycorma delicatula	FJ456942	15.693
Lycorma delicatula	MN607209	16.103
Pyrops (Laternaria) candelaria	FJ006724	16.315

Table 1: Species used to recreate the Phylogenetic tree generated by Jeong et al [21].

[6]Installation: https://www.ncbi.nlm.nih.gov/books/NBK179288#chapter6.Getting_Started
[7]http://www.clustal.org/download/current/
[8]https://github.com/somme89/rapidNJ

Utilizing Blast to find similar species

NCBI's Blast was downloaded and installed from their website[9]. Like with BWA and BCFtools, we add the path to the Blast directory to the Path variable which allows us to run Blast from anywhere on our machine. Additionally, the nucleotide (nt) database was downloaded by running `perl update_blastdb.pl nt`. The size of this database is 395 Mb. Performing a Blast search using a reference genome is done with the command `blastn -db nt -query /Path/To/Genome/File -perc_identity 90 -outfmt 5 -max_target_seqs -out /Path/To/Output/File.xml`. The three parameters, perc_identity, outfmt, and max_target_seqs, were not used initially to see the full output but it is recommended to include these. Using these parameters vastly limits the amount of data to those that are the closest aligned. Perc_identity stands for percent identity and is a measure of how much the blasted sequence overlaps with the database sequence. Blast will only output overlaps bigger than this parameter. The next parameter describes the output format with 5 standing for xml output format. There are many post-Blast processing tools that work best with xml format rather than the standard text output format. The last parameter limits how many matched sequences to output. Using these parameters provides a better overview of the entire genome and will make it easier to create a Phylogenetic tree.

From the output of Blast we can select closely overlapping species for further investigation such as in creating a phylogenetic tree. The genomes are downloaded from Genbank using the accession number given by Blast. A Phylogenetic tree is created using the species after performing some preprocessing on the outputted matches. Among full genome matches, Blast will also match with RNA and mitochondrial genomes but the goal is to study the relationship hierarchy to other species.

[9] https://www.ncbi.nlm.nih.gov/books/NBK52640/

Results

Studying novel species in a genomic context is a difficult problem due to the lack of data in comparison to non-novel species. The two methods discussed, Read Mapping into PSMC and Blast into Phylogenetic Tree generation, are a good starting point into exploring them more and we applied them to the Spotted Lanternfly phased genome. The results were less than we hoped for due to complications with that genome, or possibly with the parameters that were given to one of the algorithms.

Read Mapping Issues

Various issues occurred while running through the pipeline. The first one that occurs is after running BWA mem in that the output file (in .bam format) is not sorted and needs to be for BCFtools mpileup. This was done using the samtools sort command. Samtools is another software package similar to BCFtools. The other issue was run time. As seen in Table 2, all of the files that the algorithms are working with are huge and took several days to terminate in the case of BWA mem, BCFtools mpileup.

The final output of the pipeline, the output for BCFtools call, contains information on only a subset of the initial genome. All of the variants detailed in that file are 'indels', locations of insertions or deletions, and it was not able to find any SNPs or SNVs.

Outputs	Size (Mb)
SLF Genome	2,289.6
SLF Reads	263,713.1
BWA mem	190,919.3
BCFtools mpileup	9,778.6
BCFtools call	0.742
Blast	369.7

Table 2: Size of various files outputted while running through the Read Mapping pipeline. The size of the files affects the overall running time, with some of the algorithms taking days to finish. Note, these sizes are for the uncompressed versions of the output files.

Because the read mapping output file was problematic, it was not possible to convert the consensus file (VCF format) into the PSMC specific file (psmcfa format).

There were two produced errors: "substr outside of string at /Path/To/vcfutils.pl" and "Use of uninitialized value in lc at /Path/To/vcfutils.pl."

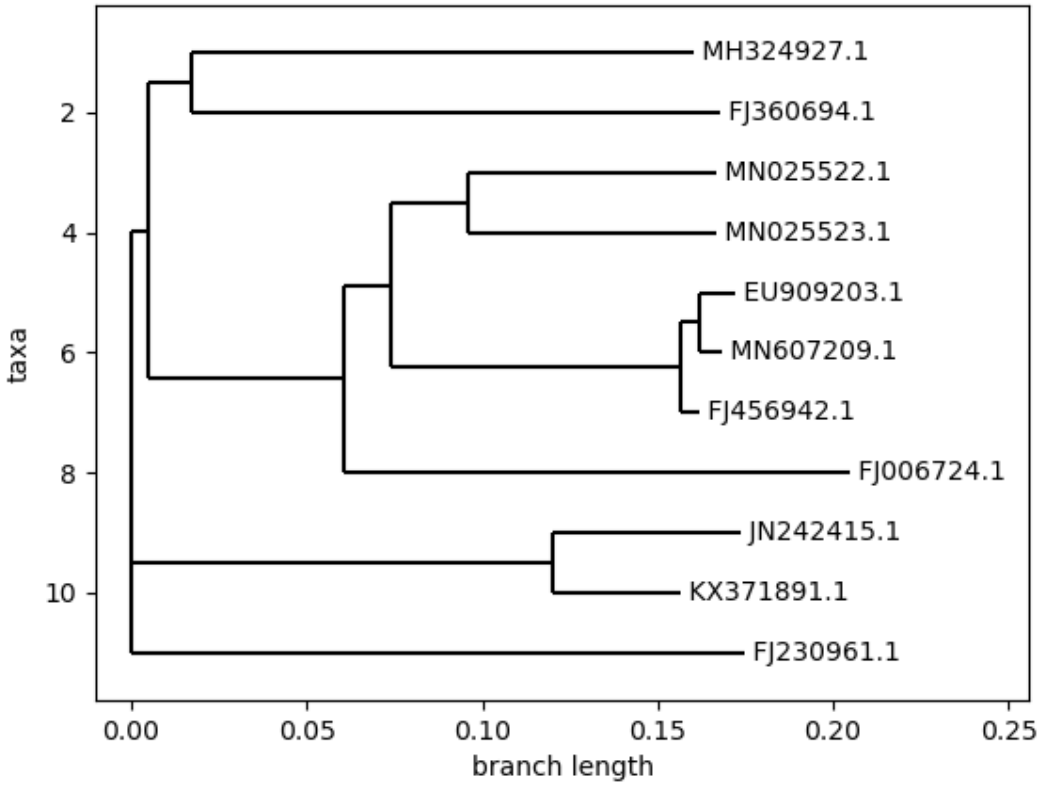
Recreating the Mitochondrial Phylogenetic Tree for the SLF

We recreated a Phylogenetic tree shown in the study by Jeong et al [23] using the same 11 species, listed in Table 1. ClustalW produced a measure of the overlap between those 11 genomes which was used by RapidNJ to create the tree shown in Fig. 14a. This tree is structurally similar to the tree in Fig. 14 with some branches having moved around due to some variation in the tree generation algorithm. The three Spotted Lanternfly genomes (Accession Numbers EU909203, MN607209, FJ456942) are in the middle of the tree and are branched very closely together.

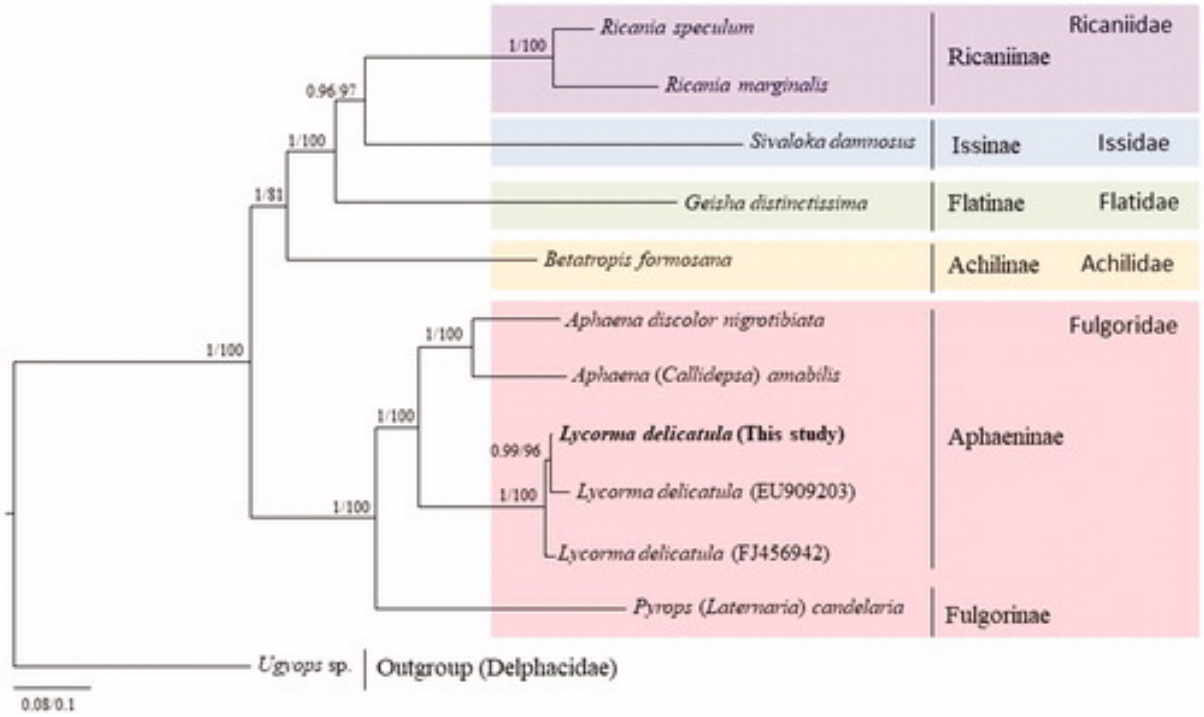
The time to do these calculations and create a tree with the 11 species was relatively quick and produce a more detailed tree.

Using Blast Output for Phylogenetic Tree Creation

The Spotted Lanternfly genome is 2.3 Gb in size and contains 2927 contigs. After running Blast for almost a week with the parameters previously described, Blast produced matching sequences from the nucleotide database for only 48 of those contigs. Attached to each listed genome is a score in bits that can be used to further refine the number of outputted matching genomes. There were 209,704 matched sequences with the average number of matches to contig being 4,368. There were 2,452 unique matches to genomic sequences. 458 of those matches are to mRNA sequences. The rest of the matched sequences were to different parts of the same species genome. Of the remaining matches, 424 are to sequenced genomes with 51 unique nuclear genomes. Some of the most common matched species were Bombus terrestris (buff-tailed bumblebee), Myrmica ruginodis (a species of ant), Caligus rogercresseyi (a parasite), Mellicta athalia (a type of butterfly), Melitaea cinxia (a type of butterfly), Autographa gamma (a type of moth), Deilephila porcellus (a type of Hawkmoth), Lymantria monacha (a type of moth), and several other anthropod species. A large percentage of the matched sequences were



(a) Replication of Phylogenetic tree using ClustalW and RapidNJ using only 11 main species. Spotted Lanternfly accession codes are EU909203, MN607209, and FJ456942.



(b) Phylogenetic Tree produced by Jeong et al [23]

Figure 14: Comparison of Phylogenetic Trees using two different data sets based on the Spotted Lanternfly

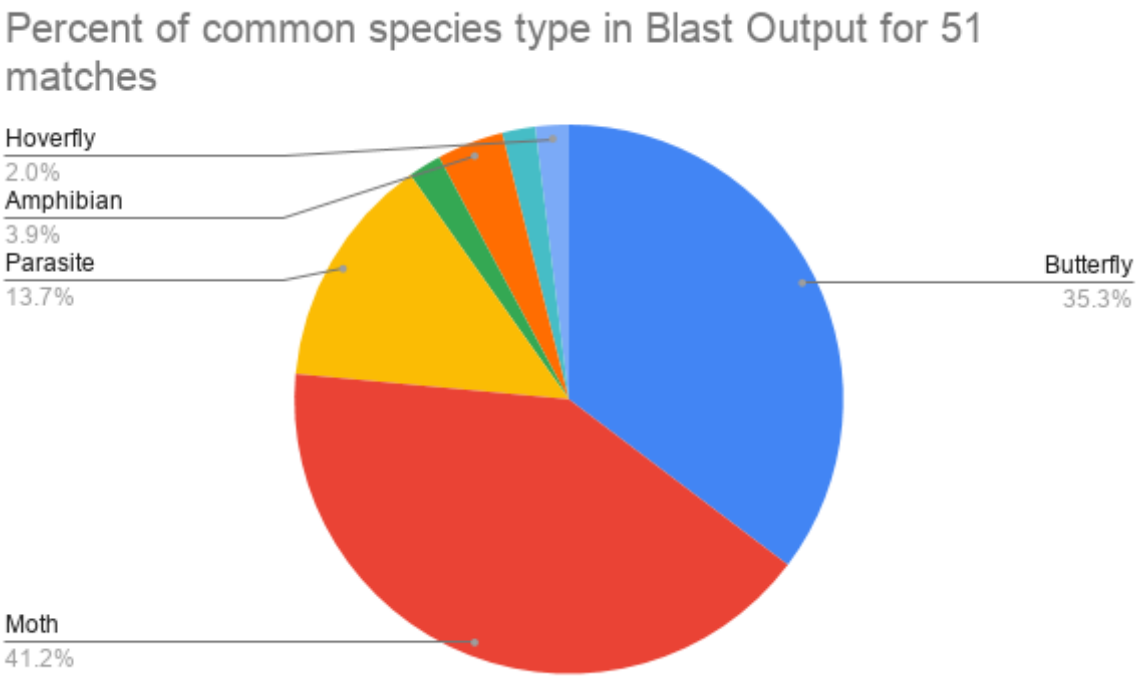


Figure 15: Percentage of overarching common species type found in the Blast output as applied to the Spotted Lanternfly genome.

moths or butterflies, as shown in Fig. 15. These genomes can then be used with further Phylogenetic Tree analysis as previously described.

The 51 unique nuclear genomes produced by Blast were downloaded from Genbank and concatenated into one fasta file, along with the SLF genome. As a result of the file size, ClustalW crashed every time. Limiting the SLF genome to one contig and reducing the number of matched genomes to a smaller number such that the file size was a bit smaller than 1 Gb still crashed the algorithm.

Discussion

There are many difficulties when working with novel species as a result of the limited data available for study but also do to the method of producing that data. Diploid assemblers producing a phased genome instead of a true diploid genome is a major obstacle when using conventional genomic analysis tools. However, it is still possible to produce results that allow us to further explore the novel species.

Read Mapping was used to circumvent the need for having a true diploid genome as input to PSMC. However, in trying to use the that to calculate the variant locations, we found that all of the variants were either insertions or deletions. There should have been some amount of SNVs and the fact that there were none is alarming. There are a variety of reasons this could be the case. The first lies with the parameters we chose when running the BWA or BCFtools algorithms. Using parameters that refine the output more or checking whether the algorithms were used properly in the first place would be viable choices in moving forward. The second could be whether the available reads were actually used to assemble the available genome. Those two not matching up would produce the high number of indels present in the final output. Until this issue is resolved, we can't move forward with PSMC analysis.

Blast was able to find many overlapping genomes within its nucleotide database and we use that data to create a Phylogenetic tree containing the one phased genome for the Spotted Lanternfly. Of particular note in Fig. 15 is that a majority of the matched species were types of moths and butterflies. Recalling the mitochondrial tree by Jeong et al [23] in Fig. 14(b), the family names listed on the right (Ricaniidae, Issidae, Flatidae, Achilidae, and Fulgoridae) are all covering various planthopper species. On the other hand, all of the moth and butterfly species produced by blast were part of the Lepidoptera Order. In the taxonomic order given by Kingdom-Phylum-Class-Order-Family-Genus-Species[10], the various planthopper families belong to the Hemiptera order. The two are united by the Class Insecta. This taxonomic order in the Blast output and

[10]https://www.biosciencewriters.com/Species-Taxonomy-Nomenclature.aspx

the mitochondrial genomes tell us that Spotted Lanternflies are more closely related to moths and butterflies.

There are various caveats to this, however. The first is that I am only using the partial Blast output and opting to choose only the matched sequences that correspond to a genome rather than to RNA or mitochondrial data. I also created those statistics using unique species names rather than weighting the percentages by the number of similar matches. That would potentially change the chart in Fig. 15. Because only a subset of the Spotted Lanternfly genome was Blasted, the matched sequences could lack the representation of grasshopper species or other arthropod species. Arthropods encompass all invertebrate animals with a segmented body, exoskeleton, and jointed appendages. This would include not only the moths, butterflies, and grasshoppers listed above, but also other types of insects and spiders. Blast could produce matches to insect or spider species that weren't found because of only a subset of the SLF genome being blasted.

When trying to create a Phylogenetic tree using the subset of unique species genomes, we ran into issues when running ClustalW. Either running the $>$ 1 Gb file in Galaxy or in the command line or other online sites, none would terminate or return errors if they did. By producing bigger picture statistics, like in Fig. 15, we are still able to capture some of the structure we would have seen in the Phylogenetic tree. Using a different mutli-sequence aligner that is better suited for bigger datasets could be used to get around this issue related to ClustalW.

Conclusion

In trying to study novel species that have only one assembled diploid genome using conventional tools, we ran into several issues not made clear in the literature. These issues were centered around the assembly process for the genome and we found that most if not all diploid assemblers create a phased (collapsed) genome instead of a true diploid sequence. Many of the conventional algorithms, such as PSMC and NJ trees, require a true diploid genome or multiple phased genomes. To circumvent these issues, we investigated various tools that would provide similar information albeit at a longer processing time. These were Read Mapping using the initial assembly reads to find variant cites to feed into PSMC, and Blast to find similar genomes by comparison with a large genomic database to be utilized with Phylogenetic Trees. In applying these tools to the novel species, the Spotted Lanternfly, processing time became an issue as a result of large file sizes. We also ran into an unresolved issue while doing PSMC. More work is needed to further streamline and debug this process to better research novel species.

Future Work

To better analyze novel species using these tools, there are several things that could be done. One extension is to repeat the read mapping process with more refined parameters that both improve the run-time of those algorithms and improve the final results. Alternatively, repeating these processes using a subset of the genome and/or the reads to reduce the workload on the read mapping algorithms.

In terms of using the Blast output, we were not able to create a Phylogenetic tree. Reducing the number of matches from 51 to a manageable number could be done to use ClustalW and produce a Phylogenetic tree. Once that is possible, an alternative to better analyze the novel species genome would be creating a Phylogenetic tree contig by contig rather than holistically. This would identify other species that are similar in certain areas and dissimilar in others, providing further use in identifying closely related species for additional study.

As another alternative to PSMC and Phylogenetic trees, performing bootstrapping to create multiple fictitious genomes would remove the limitations caused by the limited number of genomes for a novel species. Bootstrapping takes random samplings of the genome to create a fictitious genome of the same size. However, more research is needed to see what algorithms could be used and to what extent.

Acknowledgements

I would like to especially thank my advisor, Sara Mathieson, for their immense help throughout this project, especially when it came to the biology aspect. I would also like to thank my Chesick advisor, Ted Brzinski, for supporting and encouraging me throughout.

References

[1] D. R. Zerbino and E. Birney. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5):821–829, February 2008.

[2] Chen-Shan Chin, Paul Peluso, Fritz J Sedlazeck, Maria Nattestad, Gregory T Concepcion, Alicia Clum, Christopher Dunn, Ronan O'Malley, Rosa Figueroa-Balderas, Abraham Morales-Cruz, Grant R Cramer, Massimo Delledonne, Chongyuan Luo, Joseph R Ecker, Dario Cantu, David R Rank, and Michael C Schatz. Phased diploid genome assembly with single-molecule real-time sequencing. *Nat Methods*, 13(12):1050–1054, December 2016.

[3] Kay Prüfer, Fernando Racimo, Nick Patterson, Flora Jay, Sriram Sankararaman, Susanna Sawyer, Anja Heinze, Gabriel Renaud, Peter H. Sudmant, Cesare de Filippo, Heng Li, Swapan Mallick, Michael Dannemann, Qiaomei Fu, Martin Kircher, Martin Kuhlwilm, Michael Lachmann, Matthias Meyer, Matthias Ongyerth, Michael Siebauer, Christoph Theunert, Arti Tandon, Priya Moorjani, Joseph Pickrell, James C. Mullikin, Samuel H. Vohr, Richard E. Green, Ines Hellmann, Philip L. F. Johnson, Hélène Blanche, Howard Cann, Jacob O. Kitzman, Jay Shendure, Evan E. Eichler, Ed S. Lein, Trygve E. Bakken, Liubov V. Golovanova, Vladimir B. Doronichev, Michael V. Shunkov, Anatoli P. Derevianko, Bence Viola, Montgomery Slatkin, David Reich, Janet Kelso, and Svante Pääbo. The complete genome sequence of a Neanderthal from the Altai Mountains. *Nature*, 505(7481):43–49, January 2014.

[4] Joachim Wolff, Bérénice Batut, and Helena Rasche. Mapping (galaxy training materials), 03 2021.

[5] Michael Dunn, Niclas Burenhult, Nicole Kruspe, Sylvia Tufvesson, and Neele Becker. Aslian linguistic prehistory. *Diachronica*, 28(3):291–323, October 2011.

[6] Julie M Urban. Perspective: shedding light on spotted lanternfly impacts in the USA. *Pest Management Science*, 76(1):10–17, January 2020.

[7] Heng Li and Richard Durbin. Inference of human population history from individual whole-genome sequences. *Nature*, 475(7357):493–496, July 2011.

[8] Kelley Harris, Sara Sheehan, John A. Kamm, and Yun S. Song. Decoding Coalescent Hidden Markov Models in Linear Time. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, and Roded Sharan, editors, *Research in Computational Molecular Biology*, volume 8394, pages 100–114. Springer International Publishing, Cham, 2014. Series Title: Lecture Notes in Computer Science.

[9] The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, July 1987.

[10] Kelvin Li, Eli Venter, Shibu Yooseph, Timothy B Stockwell, Lance D Eckerle, Mark R Denison, David J Spiro, and Barbara A Methé. ANDES: Statistical tools for the ANalyses of DEep sequencing. *BMC Research Notes*, 3(1), July 2010.

[11] M.A. Larkin, G. Blackshields, N.P. Brown, R. Chenna, P.A. McGettigan, H. McWilliam, F. Valentin, I.M. Wallace, A. Wilm, R. Lopez, J.D. Thompson, T.J. Gibson, and D.G. Higgins. Clustal W and Clustal X version 2.0. *Bioinformatics*, 23(21):2947–2948, November 2007.

[12] Keith A Crandall, Jens Lagergren, and WABI International Workshop on Algorithms in Biomatics. *Algorithms in bioinformatics: 8th international workshop, WABI 2008, Karlsruhe, Germany, September 15-19, 2008 : proceedings*. Springer, Berlin, 2008. OCLC: 255376488.

[13] Phylogenetic systematics, a.k.a. evolutionary trees : Using trees. `https://evolution.berkeley.edu/evolibrary/article/phylogenetics_09`.

[14] E. W. Myers. A Whole-Genome Assembly of Drosophila. *Science*, 287(5461):2196–2204, March 2000.

[15] H. Li and R. Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760, July 2009.

[16] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, August 2009.

[17] Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L Madden. BLAST+: architecture and applications. *BMC Bioinformatics*, 10(1):421, 2009.

[18] Sarah B Kingan, Julie Urban, Christine C Lambert, Primo Baybayan, Anna K Childers, Brad Coates, Brian Scheffler, Kevin Hackett, Jonas Korlach, and Scott M Geib. A high-quality genome assembly from a single, field-collected spotted lanternfly (Lycorma delicatula) using the PacBio Sequel II system. *GigaScience*, 8(10):giz122, October 2019.

[19] Sarah Kingan, Julie Urban, Christine Lambert, Primo Baybayan, Anna Childers, Brad Coates, Brian Scheffler, Kevin Hackett, Jonas Korlach, and Scott M. Geib. Data from: A High-Quality Genome Assembly from a Single, Field-collected Spotted Lanternfly (Lycorma delicatula) using the PacBio Sequel II System, 2019. type: dataset.

[20] Krystyna Nadachowska-Brzyska, Reto Burri, Linnéa Smeds, and Hans Ellegren. PSMC analysis of effective population sizes in molecular ecology and its application to black-and-white ficedula flycatchers. *Molecular Ecology*, 25(5):1058–1072, February 2016.

[21] Na Ra Jeong, Min Jee Kim, Wonhoon Lee, Gwan-Seok Lee, and Iksoo Kim. Complete mitochondrial genome of the spotted lanternfly, *Lycorma delicatula* White,

1845 (Hemiptera: Fulgoridae). *Mitochondrial DNA Part B*, 5(1):370–372, January 2020.

[22] Enis Afgan, Dannon Baker, Bérénice Batut, Marius van den Beek, Dave Bouvier, Martin Čech, John Chilton, Dave Clements, Nate Coraor, Björn A Grüning, Aysam Guerler, Jennifer Hillman-Jackson, Saskia Hiltemann, Vahid Jalili, Helena Rasche, Nicola Soranzo, Jeremy Goecks, James Taylor, Anton Nekrutenko, and Daniel Blankenberg. The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Research*, 46(W1):W537–W544, May 2018.

[23] Na Ra Jeong, Min Jee Kim, Wonhoon Lee, Gwan-Seok Lee, and Iksoo Kim. Complete mitochondrial genome of the spotted lanternfly, *Lycorma delicatula* White, 1845 (Hemiptera: Fulgoridae). *Mitochondrial DNA Part B*, 5(1):370–372, January 2020.