A Genetic Origin for Male-Male Courtship in Drosophila: A Comprehensive Genomic Comparison and Alignment Algorithm

Jordan Howe

GSAlign Background:

GSAlign is an algorithm that works to perform a sequence alignment between two genomes, encountering differences between the sequences in the forms of single nucleotide polymorphisms (SNPs), insertions, and deletions (Leach & Blair, 2014). This algorithm is also unique because it is specialized to deal with intra-species genomes, or a collection of individuals within the same species. This is important because it is more finely tuned and able to pick up slight differences that more broad algorithms may not be able to detect. While the aims of my project were not to detect intra-species differences, the intra-species specificity of the algorithm is great to capture the slight differences that can occur between closely related *Drosophila* species.

The GSAlign algorithm uses a variety of functions, but it primarily works by separating the reads into two groups via a divide and conquer method: those with differences and those without. From here, the algorithm takes on three primary functions: (1) Local Maximal Exact Match (LMEM) identification, (2) similar region identification, and (3) alignment processing. A LMEM is a common string between the two genomes examined that begins at a specific location in the sequence. Upon LMEM identification, they are converted into pairs of identical fragments, one from the reference and one from the query. These pairs are clustered into groups based off of regions of similarity between the two sequences. After this, gaps are identified between the two pairs, which represent insertions, deletions, or SNPs. To construct a full alignment, GSAlign closes these gaps and outputs an alignment file as well as a vcf file with all variants.

With this general overview of the functions in mind, one can now venture into the specifics of the algorithm. GSAlign primarily uses Burrows-Wheeler transform in order to generate an array with genome sequence P and its reverse complement, P’. This works by taking the sequence string, T, with length L, and continuously rearranging it to create a new string, BWT(T). The last base in each rearrangement is then selected and used to create a new sequence, the BWT. This is used to find the specific sequence or pattern, S, in genome Q, the second genome, via a backwards search through the BWT array to locate the occurrences of S in text T. With this process, the LMEMs are found using P and P’. These are denoted with a tuple as a simple pair (i1,i2,j1,j2), with i1 and i2 representing the starting and ending positions of the sequence on P, and j1 and j2 representing the starting and ending positions of the sequence on Q. If P[i1,i2] = Q[j1,j2], it is a match, and if they are not equal, the match does not extend. There are also some user-inputted parameters to edit the number of LMEMs that are returned, such as a user-inputted length k that represents the minimum LMEM length and a threshold f that represents the maximum number of LMEM occurrences. This procedure continues until the end of genome Q.

Phase two of the GSAlign functions is the identification of similar regions. After the simple pairs are collected, they are split into regions based on their homology. For each pair, the position difference (i1-j1) is determined and used to group together pairs with similar position differences. Using a maximum difference threshold, each simple pair is examined. All simple pairs that have a position difference under the maximum threshold (default = 25) are clustered together. After re-sorting the simple pairs by their positions in sequence Q, some pairs may no longer be co-linear with their adjacent pairs, indicating outliers. These outliers are identified by whether the position difference from the adjacent sequences is greater than 5. If there are multiple occurrences of the same sequence, then the one with the smallest position difference is used. For every two adjacent simple pairs sa=(ia,1, ia,2, ja,1, ja,2) and sb=(ib,1, ib,2, jb,1, jb,2), the gap between them, denoted as gap(Sa, Sb), is calculated as jb,1 - ja,2. If this gap is greater than 300 BP, then the simple pair sb is the break point of this similar region of pairs. This clustering algorithm continues until it runs through all simple pairs. For pairs in the same cluster, If ia,1 $\leq$ ib,1 $\leq$ ia,2 or ja,1 $\leq$ jb,1 $\leq$ ja,2, then sa and sb overlap, and the overlapping fragment is removed from the smaller pair. After removing overlaps, gaps are filled by inserting normal pairs (ir, ir+1, jr, jr+1). A normal pair is represented as a 4-tuple (i1, i2, j1, j2) where P[i1, i2] $\neq$ Q[j1, j2].

The algorithm moves on to the third and final stage, the alignment processing, which further classifies the normal pairs into three types: I, II, and III. If the pair has equal size and has less mismatches than the threshold value, it is type I, indicating only substitutions and no change to the reading frame. Type II is classified as one fragment being a null string and the other one containing at least one base, indicating an insertion/deletion event. All other pairs are classified as type III, requiring gapped alignment. Lastly, all pairs are concatenated to result in the final alignment.

Replicate Code:

In order to replicate the functions of GSAlign, a python script was written. This involved file inputs of fasta files from two species. These fasta files consist of data extracted via a python script from Kegg and NCBI entries. These entries were obtained using a python script with API in order to extract the entire recorded genome of four *Drosophila* species: *Drosophila* *melanogaster*, serving as a model genome, *Drosophila simulans*, *Drosophila virilis,* and *Drosophila grimshawi*. These species were selected based off of the extent of their male-male courtship (MMC) behavior, with *D. virilis* and *D. grimshawi* exhibiting MMC to a lesser extent than *D. melanogaster* and *D. simulans* (Suvanto et al, 1994 and McRobert and Tompkins, 1988). It is hypothesized that the species that exhibit MMC will be more similar to each other than species that do not exhibit MMC, and vice versa, since MMC primarily is a result of misidentification due to minimal differences between the male and female sexes. These differences would manifest in increased gaps between the sequences, whether those gaps are SNPs, insertions, or deletions, between species that undergo MMC and species that do not. For the species that enact the same level of MMC, there would be fewer gaps between the sequences, reflecting increased genomic similarity. Since sexual differentiation occurs in genes on the X chromosome, this could be especially pronounced for those gene sequences.

In a separate python script, the genomic information was extracted into a tab-separated value (tsv) file, with each entry corresponding to a different gene. These gene entries were extracted using a separate Python script into a file in Fasta format, with each entry also corresponding to a gene, with one file per species. The species were input into the main script in a pairwise manner in order to compare all of the sequences. Two imports were used: argparse in order to create command line arguments to run the script, and SeqIO from Biopython in order to parse the input files and create the output file.

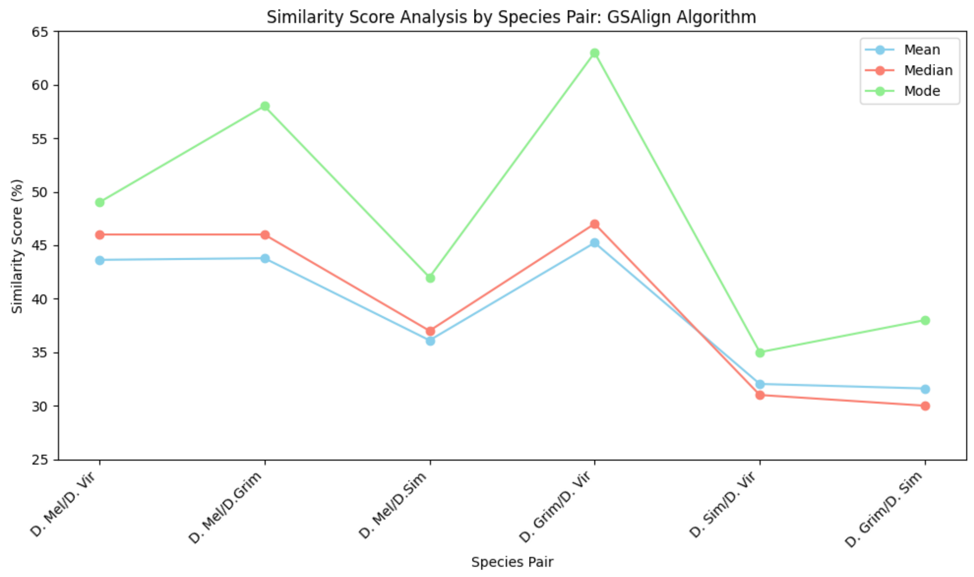
The script was organized with Object Oriented Programming (OOP), which uses encapsulation of functions into classes and inheritance from a parent class to its subclasses. Two classes were used: GeneEntry, to extract the sequences from the files, and GSAlign to perform the alignment algorithm. This class had four objects: the gene ID, gene sequence, organism name, and the gene name. This was done for labelling purposes, since the goal was to not only get an alignment, but also determine which genes in particular were similar and which were less similar, as those on the X chromosome are the ones which are in focus. The organism's name was also important, since we want to ensure that an organism is not being compared to itself. Next, the parse_fasta_files function was utilized in order to extract all of this information from each input file for sequence comparison. The entries were then returned for each respective organism.

The next class was the GSAlign class, which performed the alignment functions. The first function initializes the GSAlign object, then a main function follows to begin the alignment. It calls to the parse_fasta_files function, extracts the sequence, and iterates through the sequences. It also ensures that comparisons are only made between species and that a species is not compared to itself. It also includes a function to write the gene comparison information and similarity scores to the output file as they are processed. Next, there was a function to compute the similarity score for each gene pair, using Ukkonen’s distance as a simple yet quick algorithm. It follows a parsimony algorithm, with the minimum distance between two sequences being the minimum number of operations (substitutions, insertions, deletions) required to transform one sequence into another. It iterates over one sequence, then for each base in the second sequence, it calculates the cost of transforming that base in sequence 2 to the corresponding base in sequence 1, then appends these values to the current matrix. Upon completion of the iterations, it returns the last item in the current row, representing the distance between the two sequences. The compute_similarity method then uses this distance to calculate the similarity score, dividing the distance by the maximum sequence length for that gene comparison, multiplying by 100, then subtracting from 100. The subtraction step is performed to reflect that more similar sequences will result in a lower quotient for the division step, even though the similarity score would be higher.

Next, the code has some steps to undergo the Burrows-Wheeler transform that is essential in the GSAlign algorithm. This is done primarily with the find_exact_matches function, which initializes a list to store match indices, creates a search loop that continues as long as the top index of the BWT is less than or equal to the bottom, and retrieves the last symbol in the sequence to conduct the BWT. The top and bottom indices are updated to narrow down the search by counting the occurrences of the extracted symbol. Upon completion of the loop, the matches are sent. This method is called to in the run_bwt_alignment function, which takes two gene sequences, performs a BWT, then returns the alignment score by counting the number of matches with the find_exact_matches function. The BWT array is made in the next function, generating a suffix array "sa", concatenates the BWT sequence based off the indices of the array, then returns the BWT sequence. Finally, the main function calls the parser arguments and all the functions, returning an output file of each pairwise comparison alignment between the genes and their similarity scores.

Results:

For the self-made algorithm, only alignment scores between specific gene sequences were extracted. For the GSAlign algorithm, a sequence alignment, similarity scores, and VCF files of variants were all extracted when running the functionalities. A reason for this discrepancy was the sheer complexity of the code, as running comparisons for such large files that encompassed the entire recorded, functional genome for these *Drosophila* species was very time consuming and took a lot of computational memory. Therefore, some complexity and data was sacrificed in order to get a functioning, complete dataset for similarity scores as a means of comparison.



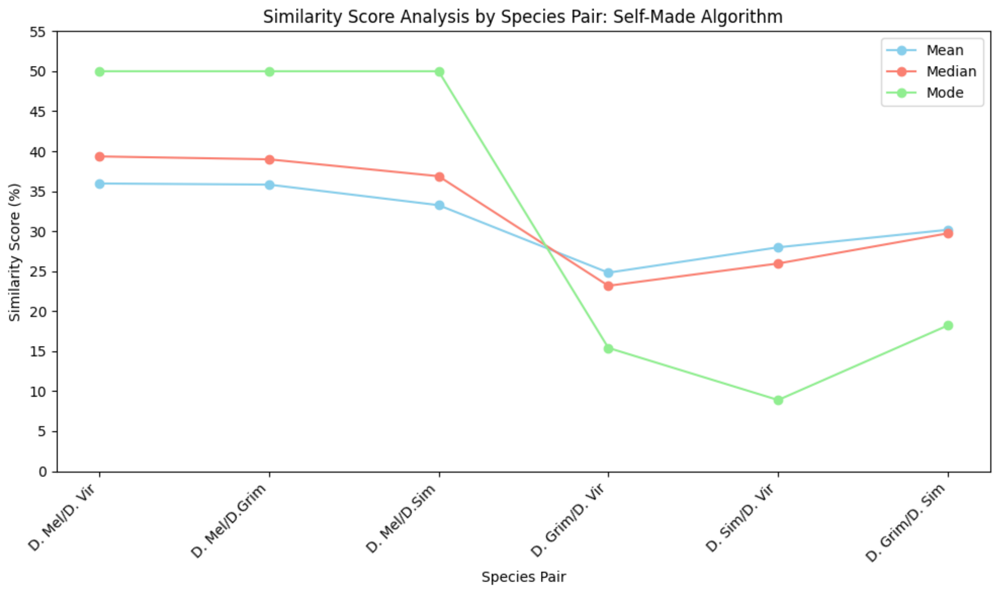


Figure 1: Similarity score analyses by species pair for the self-made algorithm (top) and the GSAlign algorithm (bottom). The mean, median, and mode similarity scores are plotted in blue, red, and green respectively.

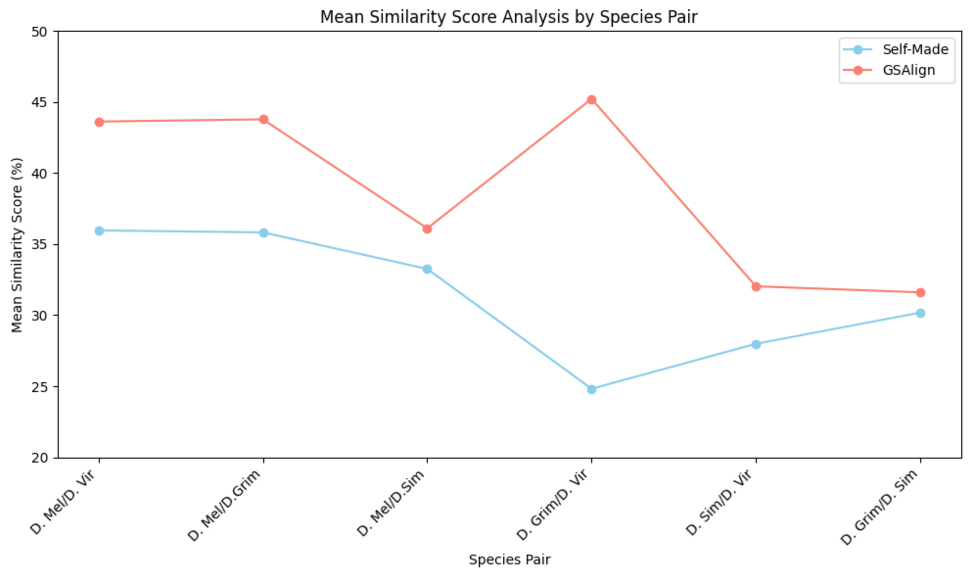


Figure 2: Plot of mean similarity scores for the self-made algorithm (blue) and the GSAlign algorithm (red). Trends were relatively similar with the exception of the *D. grimshawi/D. virlis* species pair.

The trends in the similarity scores were relatively similar between the self-made code and the GSAlign code. The GSAlign similarity scores were output as decimals, while in the self-made code, they were output as percentages, so for better visualization and conceptualization of the data, the GSAlign similarity scores were multiplied by 100 in order to better visualize them as percentages. These scores tended to be slightly higher than those from the self-made algorithm, as seen in figure 1. This could be due to the way that exact matches were calculated in terms of complexity of the code, as the self-made code relied on the simpler Ukkonen edit distance to calculate the similarity score in order to reduce complexity. In addition, while the majority of the trends for similarity scores for the species pairs were consistent between methods, one differed between the two. The *D. grimshawi*/*D. virilis* pair had a much higher similarity score using GSAlign than with the self-made algorithm. This could be due to the nature of the sequences, as both species contain large amounts of repeats of microsatellite DNA, which is thought to have brought about evolution of the species (Brittain, et. al, 2014). These repeats may have been accounted for in the more complex GSAlign algorithm, but not the self-made one, resulting in this discrepancy.

This discrepancy generates ambiguity in the results, since it was hypothesized that the species pairs that undergo MMC would be more closely related, and that the species pairs that do not undergo MMC would not be as closely related. The species that undergo MMC are *D. melanogaster* and *D. simulans*, while those that do not undergo MMC are *D. virilis* and *D. grimshawi* (Suvanto et al, 1994 and McRobert and Tompkins, 1988). For the GSAlign results, *D. grimshawi*/*D. virilis* are the most closely related species pair, proving the hypothesis, however, the opposite holds true for the self-made algorithm. Taking into account the other results, the next two species pairs that are most closely related are *D. melanogaster/D. virilis* and *D. melanogaster/D. grimshawi*. This result is constant in terms of trends and values for both methods. With this result in mind, one could most likely disprove the hypothesis and accept the null.

This conclusion, however, does neglect to factor in one variable: availability of records for the species. *D. melanogaster* is often used as a model organism for many genomic studies and therefore is more heavily sequenced and examined than other species. Because of this, there are simply more gene entries for this species than the others listed. In addition, the sequencing and construction of the genomes such as identifying sequence locations and gene/protein functions was most likely done using the genome of *D. melanogaster* as a sort of scaffold. This would increase the sequence similarity between any species and *D. melanogaster*, since any gene that is in the records of another species would also very likely be present in *D. melanogaster*. This can be seen where the similarity scores between *D. melanogaster* and other species are much higher than between non-*melanogaster* species. Therefore, this could potentially call into question the validity of the similarities described above between *D. melanogaster* and *D. virilis* and between *D. melanogaster* and *D. grimshawi*, emphasizing the need for comprehensive datasets for a variety of species.

With this in mind, one can begin to reconsider the highest similarity score for the species pair *D. virilis* and *D. grimshawi* from the GSAlign algorithm. While the GSAlign algorithm was followed in the self-made code to the greatest possible extent, it is possible that the concessions made in favor of efficiency and memory usage failed to capture the fine intricacies between genomic comparisons such as that of between this species pair. Therefore, the acceptance of the result is up to the interpreter, but as the author of the self-made code, I would choose to accept the GSAlign result and validate the hypothesis that species that do not undergo MMC will be more closely related and have higher similarity scores. In order to better validate this result, more data would be needed on the prevalence of MMC in other species, as well as more detailed genomic data for lesser-known *Drosophila* species in order to perform a more thorough analysis.

Works Cited

Brittain, A., Stroebele, E., & Erives, A. (2014, June 30). Microsatellite Repeat Instability Fuels Evolution of Embryonic Enhancers in Hawaiian Drosophila. PLoS Genetics.

Leach, M. D., & Blair, S. S. (2014). GSAlign: An Alignment Tool for Comparative Genomic Sequences. PLoS ONE, 9(6), e100947.

McRobert, S. P., & Tompkins, L. (1988). Two consequences of homosexual courtship performed by drosophila melanogaster and drosophila affinis males. Evolution, 42(5), 1093.

Suvanto, L., Hoikkala, A., and O. Liimatainen, J. (1994). Secondary Courtship Songs and Inhibitory Songs of Drosophila virilis-Group Males. *Behavior Genetics*, 24(1), 85-94.