

Introduction to Mauve

Genomes evolve

Over the course of evolution, genomes can undergo many small and large-scale changes. Local changes such as nucleotide substitution and indels have been observed in comparative studies of individual genes. Recombination, however, can cause large-scale changes such as gene loss, duplication, rearrangement, and horizontal transfer. Understanding the rates and patterns of each type of change is expected to yield insights into various biological processes.

Genome Alignment - A means for comparison

The advent of genome sequencing provides the data needed to characterize rates and patterns of genome evolution. Genome alignments can identify evolutionary changes in the DNA by aligning homologous regions of sequence. Mauve is a software package that attempts to align orthologous and xenologous regions among two or more genome sequences that have undergone both local and large-scale changes.

Locally Collinear Blocks

Because recombination can cause genome rearrangements, orthologous regions of one genome may be reordered or inverted relative to another genome. During the alignment process, Mauve identifies conserved segments that appear to be internally free from genome rearrangements. Such regions are referred to as Locally Collinear Blocks (LCBs).

The original Mauve algorithm

This section gives a brief working overview of the Mauve algorithm. For more detail, please see the paper [Mauve: Multiple Alignment of Conserved Genomic Sequence with Rearrangements](#) that appears in [Genome Research](#).

Anchored alignment

Like many other genome alignment methods, Mauve uses the anchored alignment technique to rapidly align genomes. Unlike most genome alignment methods however, Mauve allows the order of alignment anchors to be rearranged in each genome permitting identification of genome rearrangements.

Finding anchors

The current Mauve release uses inexact, ungapped matches as alignment anchors. The inexact matches are found using a seed-and-extend method, where each seed match conforms to a pattern of matching nucleotides. The inexact seed matching paradigm has previously been described by [Ma. et. al 2002](#). Mauve extends the pairwise seed matching paradigm to match multiple sequences simultaneously under the constraint that any seed must be unique in each genome. In order to be considered an anchor, the fully extended match must cover some minimum number of nucleotides. The minimum match length is user-definable, and Mauve has a default value that automatically adjusts for the size of genomes being aligned.

Selecting Locally Collinear Blocks (LCBs)

Among the initial set of anchors identified by Mauve, some will be random matches that should not be included in the correct genome alignment. Such random matches typically appear as small, insignificant genome rearrangements. In order to identify and discard such matches, Mauve requires that each collinear region of the alignment meets a "minimum weight" criteria. The weight of an LCB is defined as the sum of the lengths of matches in that LCB. Mauve removes matches composing low-weight LCBs from the set of alignment anchors before completing the alignment. The minimum LCB weight is a user-definable parameter, and by default Mauve chooses this value to be 3 times the minimum match size.

Finishing the alignment

Once a set of alignment anchors that define Locally Collinear Blocks have been determined, Mauve can complete a gapped global alignment of each LCB. Mauve applies the ClustalW progressive global alignment algorithm to each LCB.

Limitations of the original Mauve algorithm

- * Mauve works best on closely-related organisms (like *E. coli* and *Salmonella* or *Y. pestis*)
- * Large regions shared by subsets of the genomes are not aligned

- * Rearranged regions shared by subsets of the genomes will not be identified
- * The minimum LCB weight must be manually determined for accurate estimation of genomic rearrangement
- * It has difficulty aligning genomes with vast amounts of segmental duplication

The Progressive Mauve algorithm: addressing limitations of the original algorithm

Comparative genomics has revealed that closely-related bacteria often have highly divergent gene content. While the original Mauve algorithm could align regions conserved among all organisms, the portion of the genome conserved among all taxa (the core genome) shrinks as more taxa are added to the analysis. As such, the original Mauve algorithm did not scale well to large numbers of taxa because it could not align regions conserved among subsets of the genomes under study. Progressive Mauve employs a different algorithmic approach to scoring alignments that allows alignment of segments conserved among subsets of taxa. The Progressive Mauve algorithm has been described in Aaron Darling's Ph.D. Thesis, and is also the subject of a manuscript under review. A brief overview is given here.

Finding initial local multiple alignments

Progressive Mauve elaborates on the original algorithm for finding local multiple alignments. Instead of using a single seed pattern for match filtration, Progressive Mauve uses a combination of three seed patterns for improved sensitivity. The palindromic seed patterns have been described in Darling et al. 2006 "Procrastination leads to efficient filtration for local multiple alignment"

Seed matches which represent a unique subsequence shared by two or more input genomes are subjected to ungapped extension until the seed pattern no longer matches. The result is an ungapped local multiple alignment with at most one component from each of the input genome sequences.

Computing a pairwise genome content distance matrix and guide tree

Progressive Mauve builds up genome alignments progressively according to a guide tree. The guide tree is computed based on an estimate of the shared gene content among each pair of input genomes. For a pair of input genomes, g_x and g_y , shared gene content is estimated by counting the number of nucleotides in g_x and g_y aligned to each other in the initial set of local multiple alignments. The count is normalized to a similarity value between 0 and 1 by dividing by the average size of g_x and g_y . The similarity value is subtracted from 1 to arrive at a distance estimate. Neighbor joining is then applied to the matrix of distance estimates to yield a guide tree topology. Note that the guide tree *is not* intended to be a phylogeny indicative of the genealogy of input genomes. It is merely a computational crutch for progressive genome alignment. Also note that alignments are later refined independently of a single guide tree topology to avoid biasing later phylogenetic inference.

Computing a pairwise breakpoint distance matrix

Prior to alignment, Progressive Mauve attempts to compute a conservative estimate of the number of rearrangement breakpoints among any pair of genomes. For each pair of genomes, pairwise alignments are created from the local-multiple alignments and the pairwise alignments are subjected to greedy breakpoint elimination. The breakpoint penalty used for greedy breakpoint elimination is set high for closely related genomes and scaled downward according to the estimate of genomic content distance. Because the breakpoint penalty is high, the resulting set of locally collinear blocks represent robustly supported segmental homology, and a conservative estimate of the breakpoint distance can be made on this basis. The conservative estimate of breakpoint distance is used later during progressive alignment to scale breakpoint penalties.

Progressive genome alignment

A genome alignment is progressively built up according to the guide tree. At each step of the progressive genome alignment, alignment anchors are selected from the initial set of local multiple alignments. Anchors are selected so that they maximize a Sum-of-pairs scoring scheme which applies a penalty for predicting breakpoints among any pair of genomes. Because rates of genomic rearrangement are highly variable, especially in some bacterial pathogens, some genomes may be expected to exhibit greater rearrangement than others. As such, a single choice of scoring penalty is unlikely to yield accurate alignments for all genomes. To cope with this phenomenon, Progressive Mauve scales the breakpoint penalty according to the expected level of sequence divergence and the number of well-supported genomic rearrangements among the pair of input genomes. These scaling values are taken from the distance matrices computed earlier in the algorithm.

Anchored alignment

Once anchors have been computed at a node in the guide tree, a global alignment is computed on the basis of the anchors. Given a set of anchors among two genomes, a genome and an alignment, or a pair of alignments, a modified MUSCLE global alignment

algorithm is applied to compute an anchored profile-profile alignment. MUSCLE is then used to perform tree-independent iterative refinement on the global genome alignment.

Rejecting alignment of unrelated sequence

Although we compute a global alignment among sequences, genomes often contain lineage-specific sequence and are thus not globally related. The global alignment will often contain forced alignment of unrelated sequence. A simple hidden Markov model structure is used to detect forced alignment of unrelated sequence, which are then removed from the alignment.

Strengths of the Progressive Mauve algorithm

- * It can be applied to a much larger number of genomes than the original Mauve algorithm
- * It can align more divergent genomes than the original algorithm. Genomes with less than 50% nucleotide identity are often alignable
- * Manual adjustment of the alignment scoring parameters is usually not necessary
- * It aligns regions conserved among subsets of the input genomes

Limitations of the Progressive Mauve algorithm

- * It is substantially slower than the original Mauve algorithm
- * It consumes more memory than the original Mauve algorithm
- * Manual adjustment of the breakpoint penalty may still be required