

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

Installing Mauve

Installing and Running Mauve

Whether Mauve will be running under Windows, Linux, Mac OS X, or another operating system, it requires Java version 1.4 or later. Many systems already have Java 1.4 installed, so this will not be a concern. The Windows version of Mauve includes the Java installer for 32-bit windows systems that don't already have it. On other systems Java may need to be installed separately.

Downloading

Download the latest version of Mauve for your operating system from <http://gel.ahabs.wisc.edu/mauve/download.php>.

Installation on Windows

Installing

Run the Mauve installer that has been downloaded. Click the "Next" button several times. If you do not already have Java 1.4 or later and your computer runs 32-bit Windows, the Mauve installer will offer to start the Java installer for you. If your computer uses 64-bit Windows, you will need to download and install the 64-bit Java Runtime Environment from [Sun's Java web site](#).

Running

After installation, Mauve will appear in your system's Start Menu. To run Mauve simply select it from the Start Menu. Alternatively, the command-line `mauveAligner` and `progressiveMauve` programs may be run from the directory where Mauve was installed. See the chapter on command-line execution for more details.

Installation on Mac OS X

Installing

Download the Mauve disk image and copy the Mauve application to your "Applications" folder or another folder of your choice.

Running

Easy. Double-click the Mauve application.

Installation on Linux

Prerequisites

The Linux version is compiled for 32-bit and 64-bit x86 architecture and will only run on such hardware. Mauve can be built from source code to run on other architectures. Mauve requires the Sun Java runtime environment to work properly. The GNU `gcj` java runtime environment, which is the default environment on linux distributions such as Fedora, Red Hat and others, may work but is not supported. Please download and install a [Sun java runtime environment](#) for your system.

Configuring

If java is not in the executable path, the Mauve script may need to be edited to point to the Java installation. Once downloaded and uncompressed, edit the file `Mauve` by changing the variable `JAVA_CMD` to contain the full path to the java executable.

Running

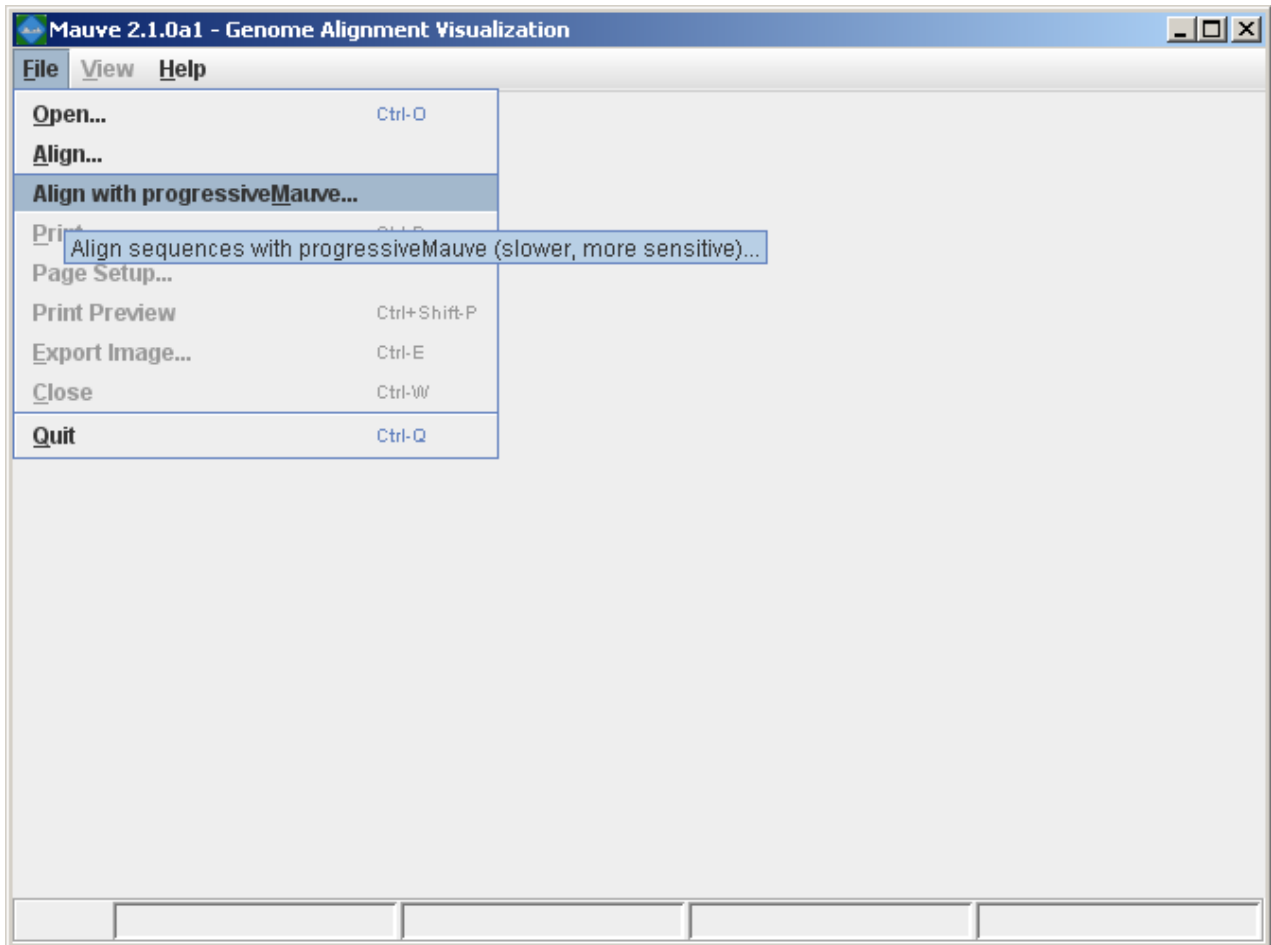
Simply run `./Mauve` from within the Mauve directory to start the Mauve Java GUI. Alternatively, the command-line alignment programs `mauveAligner` and `progressiveMauve` can be used and provide more flexibility than the Mauve GUI. You may want to add the directory containing Mauve to your executable path.

Other Unix-like operating systems

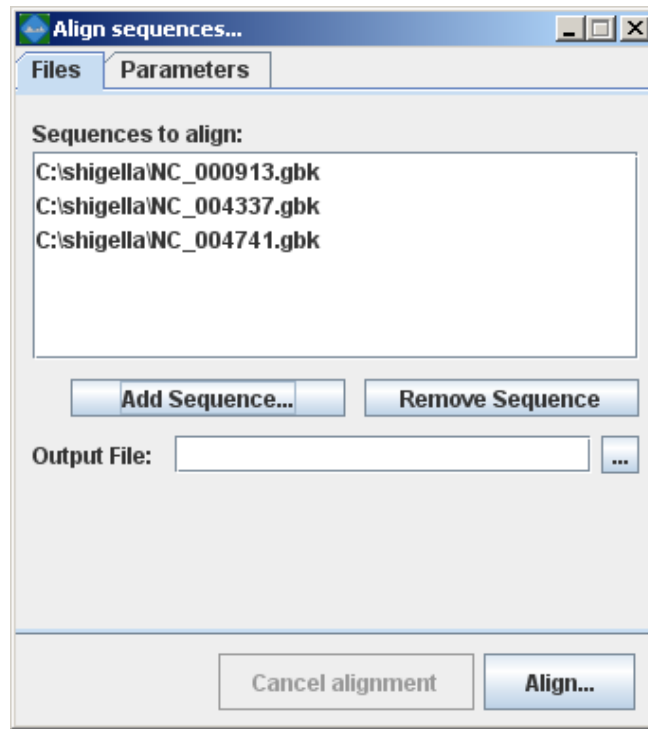
Although we only provide pre-compiled binary distributions of Mauve for Windows, Linux, and Mac OS X, we do make the source code for the Mauve GUI and `mauveAligner` available. If the `mauveAligner` source code can be compiled on a system, then the compiled binary can be used as a drop-in replacement for the `mauveAligner` binary distributed with the Linux version of Mauve. Mauve should then behave just as it does under Linux.

Constructing a Genome Alignment

Mauve provides a simple user interface for aligning genomes. Once Mauve has started up, simply select "Align..." or "Align with progressiveMauve..." from the "File" menu.



Mauve then presents the alignment dialog box:



The following sections describe the various entry fields in the dialog box.

Using the alignment dialog box

Sequence file input formats

Genome sequence files can be given to Mauve in any of FastA, Multi-FastA, GenBank flat file, or raw formats. Mauve deduces the file format based on the file name extension. By default, a file with any unrecognized file name extension is assumed to be in (Multi-)FastA format. Extensions recognized by Mauve are .gbk (GenBank), and .raw (raw sequence data). Note that some very old versions of Mauve (up to 20040219) also assume that .txt files are in raw format. FastA and GenBank format files with the genome of your organism can usually be downloaded from NCBI at <ftp://ftp.ncbi.nih.gov/genomes/>. The .fna files are in FastA format and the .gbk files are in GenBank format. For genomes with plasmids, multiple .fna or .gbk files may need to be downloaded and concatenated into a single file.

When an individual file contains several sequence entries, they will be concatenated and the whole concatenated sequence will be aligned to the sequences in the other files. This behavior allows multi-chromosomal genomes to be aligned by including all chromosomes in a single sequence file. Similarly, incomplete genomes that consist of several sequence contigs can be aligned, though beware that incorrectly ordered contigs will appear as genome rearrangements in the Mauve viewer.

Alternatively a single Multi-FastA file containing all the genomes to align can be specified. In this case Mauve assumes one genome per Multi-FastA sequence entry and will align the entries to each other.

Selecting Genomes to align

The top entry area lists the sequence file(s) containing the genomes that will be aligned. To add a sequence file, click the "Add sequence..." button and select the file to add. The Windows version of Mauve supports Drag-and-drop, allowing sequence files to be added by dragging them in from the windows explorer.

Setting an output location

The location where Mauve stores its alignment results can be set using the "File output:" text entry field. If left blank, Mauve will prompt for an output file location.

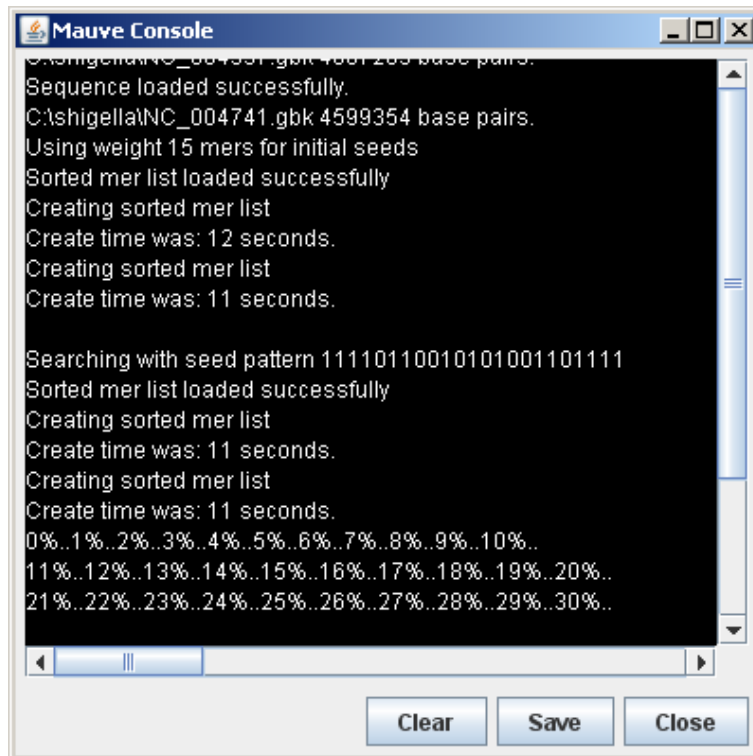
Setting custom alignment parameters

The alignment parameters are different between original Mauve (mauveAligner) and Progressive Mauve, and are discussed in more

detail in a later section.

Computing the alignment

Once the genome sequences have been loaded, click the "Align..." button to start the alignment. A console dialog will appear and will show the progress being made towards completing the alignment.

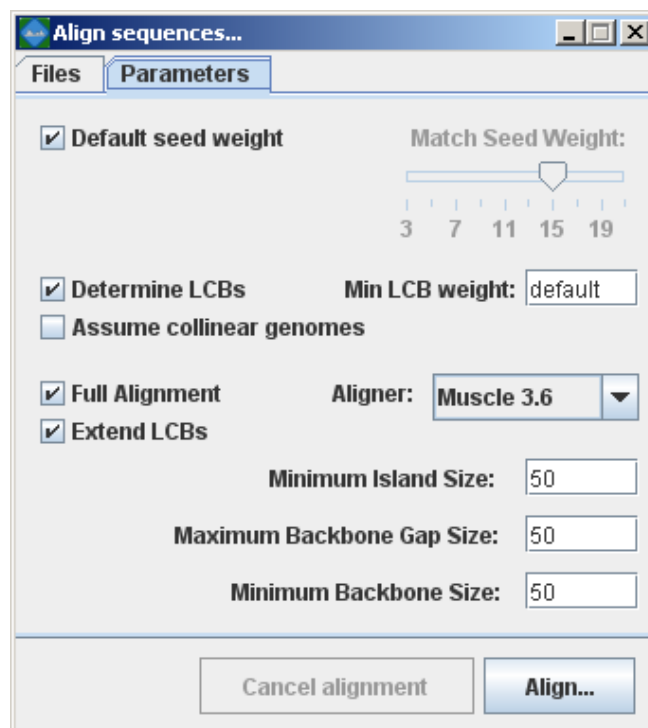


Cancel a running alignment

As of version 2.1.0, it is possible to cancel a running alignment by returning to the "Align sequences..." dialog box and clicking the "Cancel alignment" button. Alternatively, quitting the Mauve program should also kill the running alignment. If all else fails, the running aligner (mauveAligner or progressiveMauve) can be killed using the Windows task manager, the Mac OS X process inspector, or the 'kill' command in unix.

Original Mauve (mauveAligner) alignment parameters

By default, Mauve selects a set of alignment parameters that are appropriate for aligning closely related genomes with moderate to high amounts of genome rearrangement. However, some alignment parameters can (and should!) be adjusted to change Mauve's behavior. For example, the default value for Minimum LCB weight is often too low and should be replaced with a manually chosen value. When aligning more divergent genomes, the seed size can be reduced to find additional alignment anchors and achieve greater alignment coverage over the genomes. Another option disables the full alignment process, allowing Mauve to quickly generate a comparative picture of genome organization.



The following sections describe the various entry fields in the dialog box.

Parameter descriptions:

Match seed weight

The seed size parameter sets the minimum weight of the seed pattern used to generate local multiple alignments (matches) during the first pass of anchoring the alignment. When aligning divergent genomes or aligning more genomes simultaneously, lower seed weights may provide better sensitivity. However, because Mauve also requires the matching seeds must be unique in each genome, setting this value too low will reduce sensitivity.

Default seed weight

Setting this option will allow Mauve to select an initial match seed weight that is appropriate for the length of sequences being aligned. The default seed size for 1MB genomes is typically around 11, is around 15 for 5MB genomes, and continues to grow with the size of the genomes being aligned. The defaults may be conservative (too large), especially when aligning more divergent genomes. On the other hand, higher seed weights reduce noisy matching and can lead to better alignment in some cases.

Min LCB Weight

The LCB weight sets the minimum number of matching nucleotides identified in a collinear region for that region to be considered true homology versus random similarity. Mauve uses an algorithm called greedy breakpoint elimination to compute a set of Locally Collinear Blocks (LCBs) that have the given minimum weight. By default an LCB weight of 3 times the seed size will be used. The default value is often too low, however, and this value should be set manually. The procedure to determine a reasonable value for the Min LCB Weight usually involves constructing an initial alignment with the default value, and then using the LCB weight slider in the Mauve GUI (see the next section) to find a weight that eliminates all spurious rearrangements. The sequences can then be realigned using the manually determined weight value.

Determine LCBs

If this option is disabled Mauve will simply identify matches (local multiple alignments) among the genomes. See the description of match generation in the command-line interface chapter.

Extend LCBs

Controls whether mauveAligner will attempt to extend the range of existing LCBs and search for additional LCBs. For some datasets, LCB extension can be very time-consuming and may not offer much improvement in the alignment.

Assume collinear genomes

Select this option if it is certain that there are no rearrangements among the genomes to be aligned. Using this option when aligning

collinear genomes can result in improved alignment accuracy.

Aligner

Sets whether mauveAligner will utilize MUSCLE or ClustalW to compute global alignments between alignment anchors. MUSCLE is the preferred choice, since it is faster and more accurate.

Island and Backbone sizes

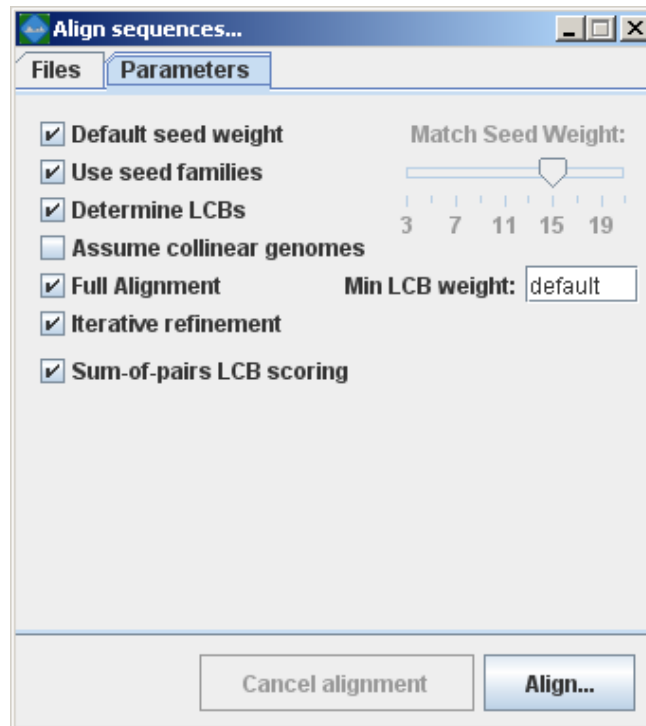
An island is a region of the alignment where one genome has a sequence element that one or more others lack. This parameter sets the alignment gap size used to calculate islands and backbone segments. See the description of the .islands and .backbone files below for more information.

Full alignment

Selecting the "Full alignment" option causes Mauve to perform a recursive anchor search and a full gapped alignment of the genome sequences using the ClustalW or MUSCLE progressive alignment method. If not selected, Mauve will identify LCBs but will not do a recursive anchor search or progressive alignment.

Progressive Mauve alignment parameters

By default, Mauve selects a set of alignment parameters that are appropriate for aligning closely related genomes with moderate to high amounts of genome rearrangement. However, some alignment parameters can (and should!) be adjusted to change Mauve's behavior. For example, the default value for Minimum LCB weight is often too low and should be replaced with a manually chosen value. When aligning



Match seed weight

As with mauveAligner, the seed size parameter sets the minimum weight of the seed pattern used to generate local multiple alignments (matches) during the first pass of anchoring the alignment. When aligning divergent genomes or aligning more genomes simultaneously, lower seed weights may provide better sensitivity. However, because Mauve also requires the matching seeds must to be unique in each genome, setting this value too low will reduce sensitivity.

Default seed weight

Setting this option will allow Mauve to select an initial match seed weight that is appropriate for the length of sequences being aligned. The default seed size for 1MB genomes is typically around 11, is around 15 for 5MB genomes, and continues to grow with the size of the genomes being aligned. The defaults may be conservative (too large), especially when aligning more divergent genomes. On the other hand, higher seed weights reduce noisy matching and can lead to better alignment in some cases.

Use seed families

Setting this option causes progressiveMauve to search for matches using three spaced seed patterns instead of just one. Using three patterns can greatly improve sensitivity on diverged genomes, and permits using a higher seed weight for closely related organisms without loss of sensitivity. Using seed families requires minor additional compute time in most cases.

Determine LCBs

If this option is disabled Mauve will simply identify matches (local multiple alignments) among the genomes. See the description of match generation in the command-line interface chapter.

Assume collinear genomes

Select this option if it is certain that there are no rearrangements among the genomes to be aligned. Using this option when aligning collinear genomes can speed up progressiveMauve's running time.

Full alignment and Iterative Refinement

Selecting the "Full alignment" option causes Progressive Mauve to perform a recursive anchor search and a full gapped alignment of the genome sequences using MUSCLE. If not selected, Progressive Mauve will identify alignment anchors cluster them into LCBs finish the alignment. The "Iterative Refinement" option applies MUSCLE to refine the initial alignment, often improving the initial alignment. As MUSCLE performs tree-independent iterative refinement, this option should be used to avoid biasing phylogenetic inference with a single guide tree.

Sum-of-pairs LCB scoring

This option selects whether breakpoint penalties are applied among all pairs of extant sequences, or whether penalties are applied to inferred ancestral gene-ordering. Since Progressive Mauve does not accurately infer ancestral gene orders, and even if it did, it does not infer ancestral genome content, this option should be considered "experimental" and should only be disabled when aligning collinear genomes.

Using the Alignment Viewer

The display layout

The alignment display is organized into one horizontal "panel" per input genome sequence. Each genome's panel contains the name of the genome sequence, a scale showing the sequence coordinates for that genome, and a single black horizontal center line. Colored block outlines appear above and possibly below the center line. Each of these block outlines surrounds a region of the genome sequence that aligned to part of another genome, and is presumably homologous and internally free from genomic rearrangement. When a block lies above the center line the aligned region is in the forward orientation relative to the first genome sequence. Blocks below the center line indicate regions that align in the reverse complement (inverse) orientation. Regions outside blocks lack detectable homology among the input genomes. Inside each block Mauve draws a similarity profile of the genome sequence. The height of the similarity profile corresponds to the average level of conservation in that region of the genome sequence. Areas that are completely white were not aligned and probably contain sequence elements specific to a particular genome. The height of the similarity profile is calculated to be inversely proportional to the average alignment column entropy over a region of the alignment.

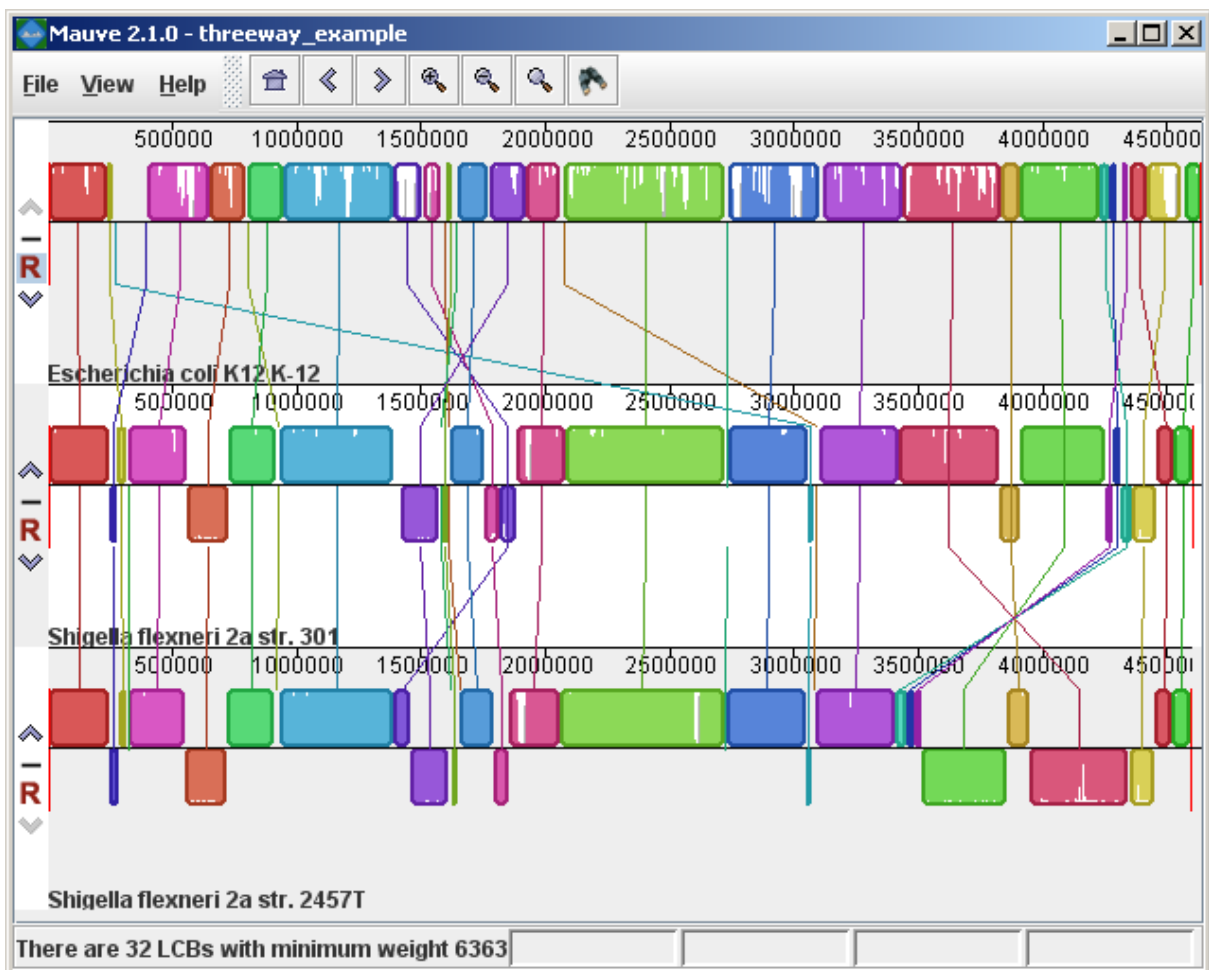


Figure 1 (above) shows an alignment of *E. coli* K12 MG1655, *S. flexneri* 2a 301, and *S. flexneri* 2457T. Notice how inverted regions in the *S. flexneri* are clearly depicted as blocks below a genome's center line. These three genomes were taken from the NCBI FTP site and aligned with Progressive Mauve using default parameters, as described in the previous section.

In Figure 1, colored blocks in the first genome are connected by lines to similarly colored blocks in the second and third genomes. These lines indicate which regions in each genome are homologous. Notice the crossing "X" pattern of lines, which happen to occur in the vicinity of the predicted origin and terminus of replication in these organisms. When viewing genomes with complex rearrangements, the LCB connecting lines can be confusing and they can be hidden (or made visible again) by typing **Shift+L** (pressing shift and L simultaneously) or using the "View" menu.

In the standard color scheme, the region of sequence covered by a colored block is entirely collinear and homologous among the genomes. The boundaries of colored blocks usually indicate the breakpoints of genome rearrangement, unless sequence has been gained or lost in the breakpoint region.

The Backbone color scheme

When an alignment has been computed with Progressive Mauve, a display mode is available that colors regions conserved among all genomes differently than regions conserved among subsets of the genomes. We term regions conserved among all genomes as "backbone," which are drawn in mauve color. Applying the color mode to the alignment of three *E. coli* and *Shigella*, and disabling LCB outlines in the "View->Style" menu yields:

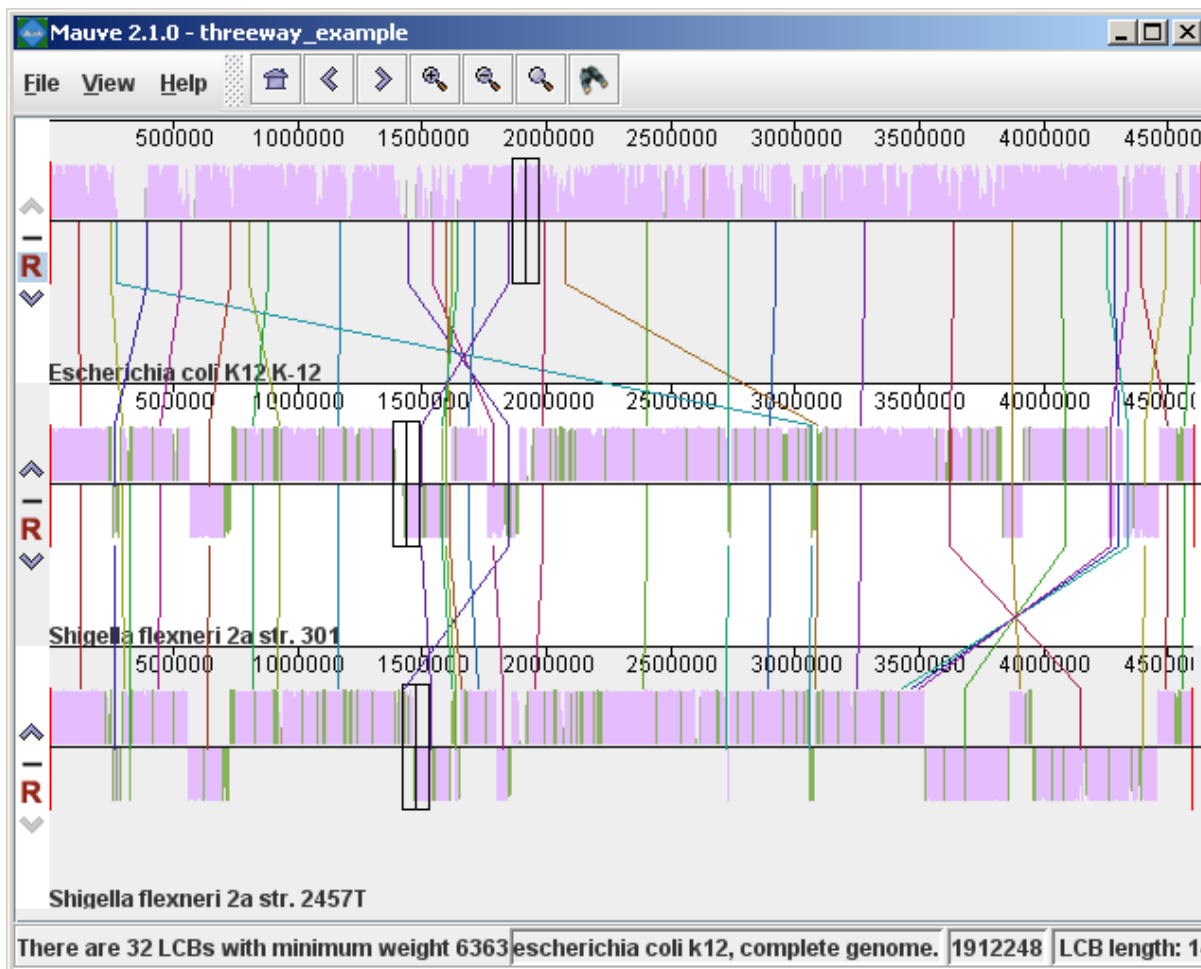


Figure 2 (above) shows the same alignment of *E. coli* K12 MG1655, *S. flexneri* 2a 301, and *S. flexneri* 2457T, but displayed with different style settings. Parts of the similarity plot which are colored mauve are conserved among all three genomes, while portions colored green are segments conserved only among the *Shigella flexneri*. Segments conserved among *E. coli* and *S. flexneri* 2457T are colored brown, but are too small to notice in a whole-genome view (see below). The black box follows the mouse cursor and highlights the homologous site in each genome. Clicking the similarity plot will line up orthologous regions

Zooming in on Annotated Features

If the aligned genome sequences were in GenBank files containing annotated features Mauve will display the annotated features next to the sequence similarity profiles. For efficiency reasons, annotated features appear only when less than 1Mbp of sequence is being displayed. To zoom in on the alignment, either use the magnifying glass buttons from the toolbar, or use the keyboard shortcuts Ctrl+Up to zoom in (press Control and up arrow key simultaneously) and Ctrl+Down to zoom out. Once viewing less than 1Mbp of sequence, annotated CDS features show up as white boxes, tRNAs are green, rRNAs are red, and misc_RNA features are blue. Mauve displays the /product qualifier when the mouse cursor is held over a feature. When a feature is clicked, Mauve shows a detailed listing of feature qualifiers in a popup window, in addition to a menu that links to the NCBI Entrez protein entry corresponding to the gene.

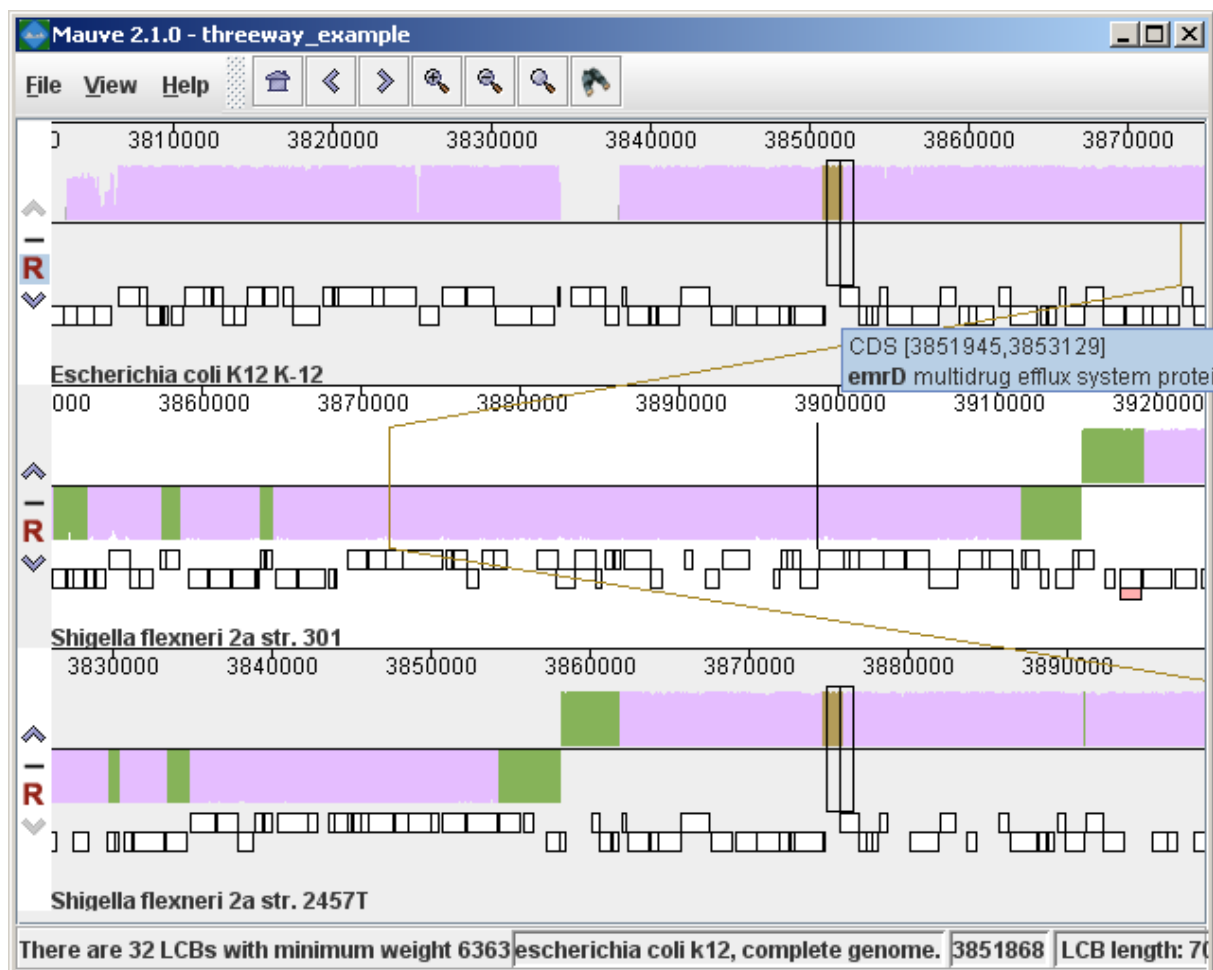


Figure 3 (above) shows the same alignment zoomed in on a region 3.85Mbp into the *E. coli* K12 genome where the gene *emrD* is encoded. Annotated genes are shown as white boxes, with genes transcribed from the reverse strand shifted downward. The mouse cursor (shown as a black line) is hovering over the *emrD* CDS in *E. coli* K12. The similarity plot for the N-terminal end of the gene and the upstream region is colored brown, indicating that the segment of DNA is missing from the *S. flexneri* 2a 301 isolate. The segmental deletion could potentially have consequences for gene regulation in this *S. flexneri* 2a 301.

Navigating the display

Zooming and shifting the display

The alignment display is interactive, providing the ability to zoom in on particular regions and shift the display to the left and right. Navigating through the alignment visualization can be accomplished by using the magnifying glass and arrow control buttons on the toolbar immediately above the display. Alternatively, keyboard shortcuts allow rapid movement through the alignment display. The keystrokes Ctrl+up arrow and Ctrl+down arrow zoom the display in and out, while Ctrl+left arrow and Ctrl+right arrow shift left and right, respectively. Holding down the "shift" key with Ctrl+left and Ctrl+right accelerates the shifting leftward and rightward.

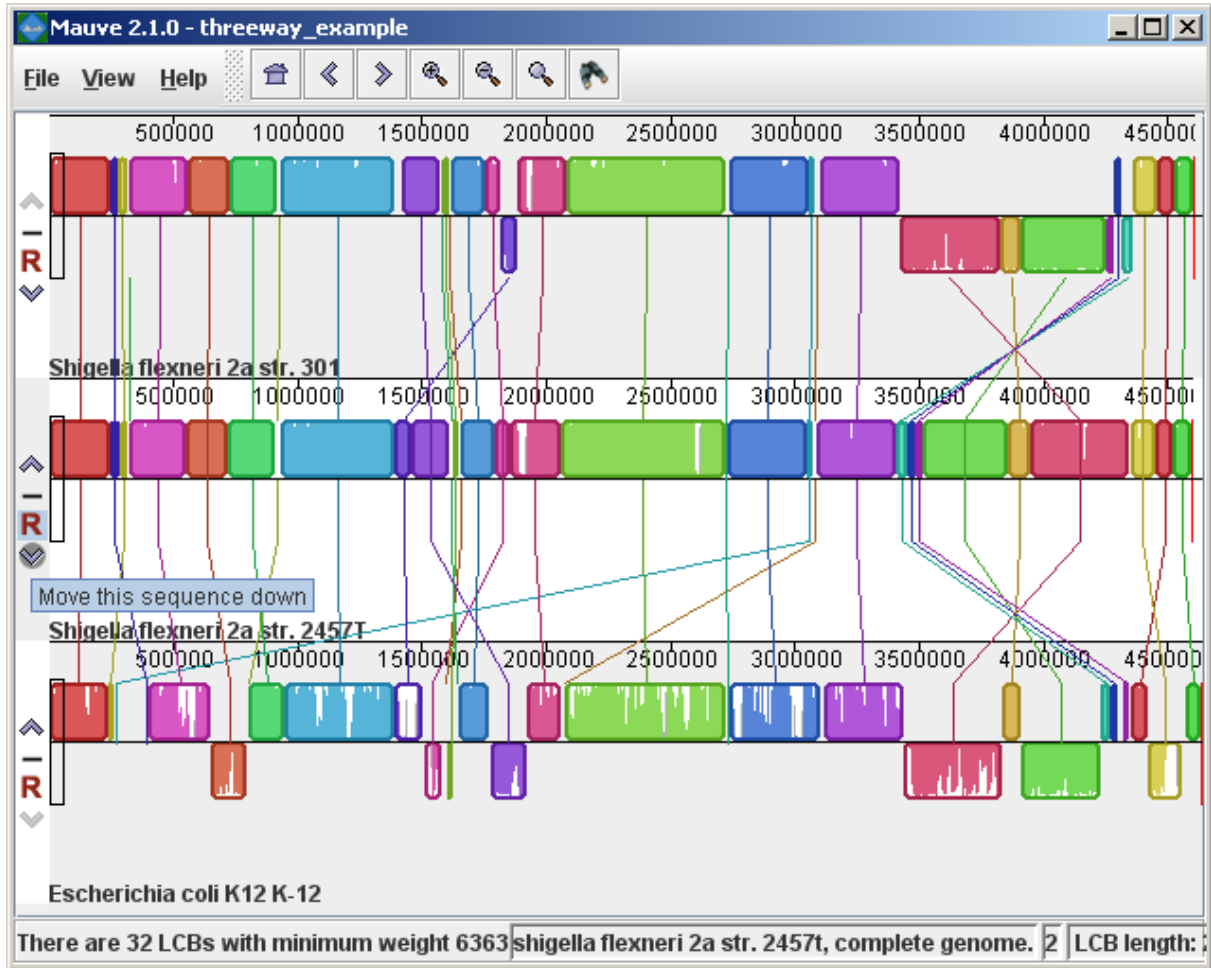
Mouse control

When moving the mouse over the alignment display Mauve will highlight the aligned and homologous sites of each genome with a black vertical bar. By clicking the mouse in the sequence similarity plot, Mauve will line up the orthologous sites of each genome. Finally, a portion of the alignment can be selected by clicking and dragging over the similarity plot while holding down the "shift" key.

Reordering sequences and changing the reference genome

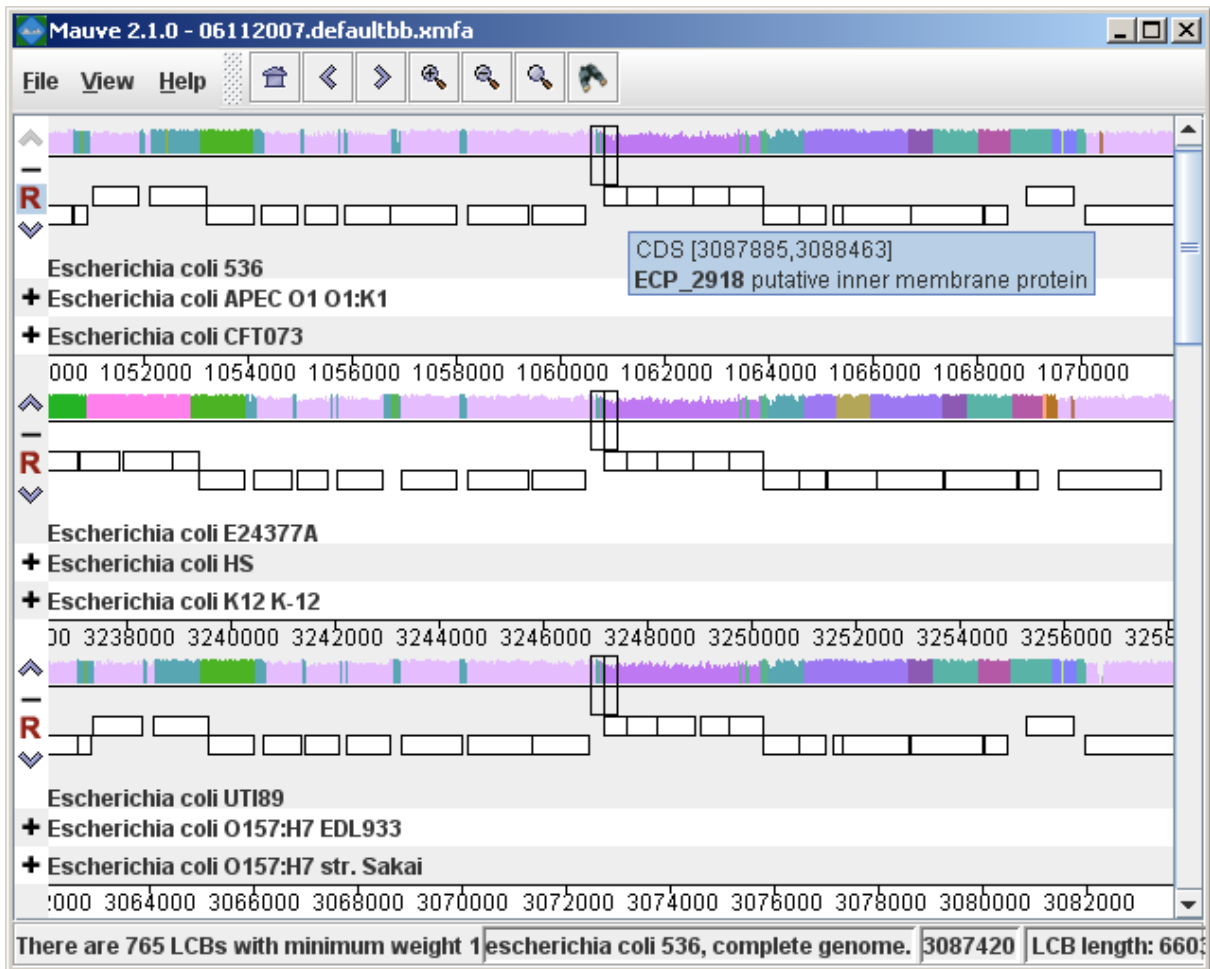
When viewing alignments of many genomes, it is often desirable to reorder the genome sequences. This can be accomplished by

clicking the up and down arrow buttons to the left of each genome. By default, Mauve uses the first genome sequence as the reference genome for assigning a reference orientation to inverted blocks. The reference genome can be changed by clicking the "R" button for the desired reference sequence, also to the left of the genome sequence display. The following figure shows the alignment of three *E. coli* and *Shigella* after it has been reordered and the reference was changed:



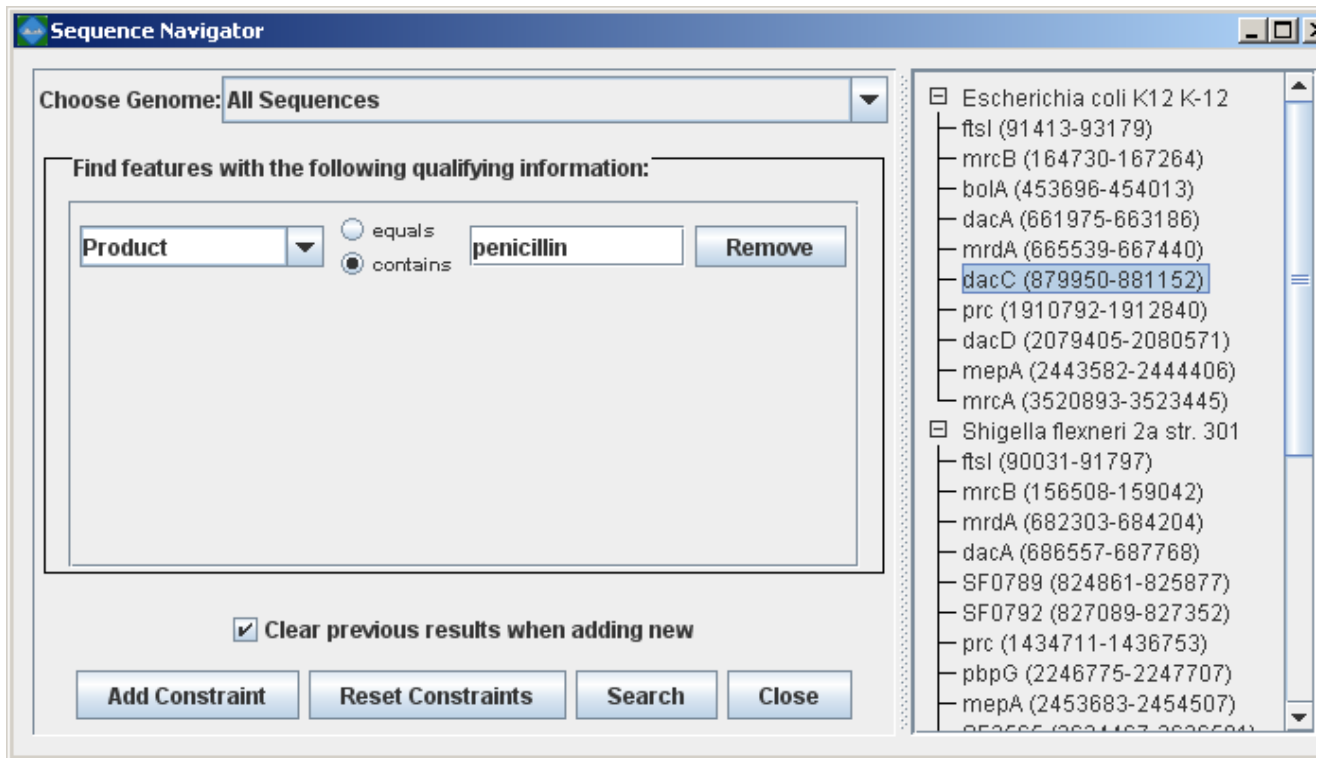
Hiding sequences

When viewing an alignment with a large number of genome sequences, it can be helpful to simplify the view by restricting the display to genomes of interest. This can be accomplished by clicking the minus button at the left of the display. For example, here is a view of an alignment of 21 genomes, with many of them hidden from view.

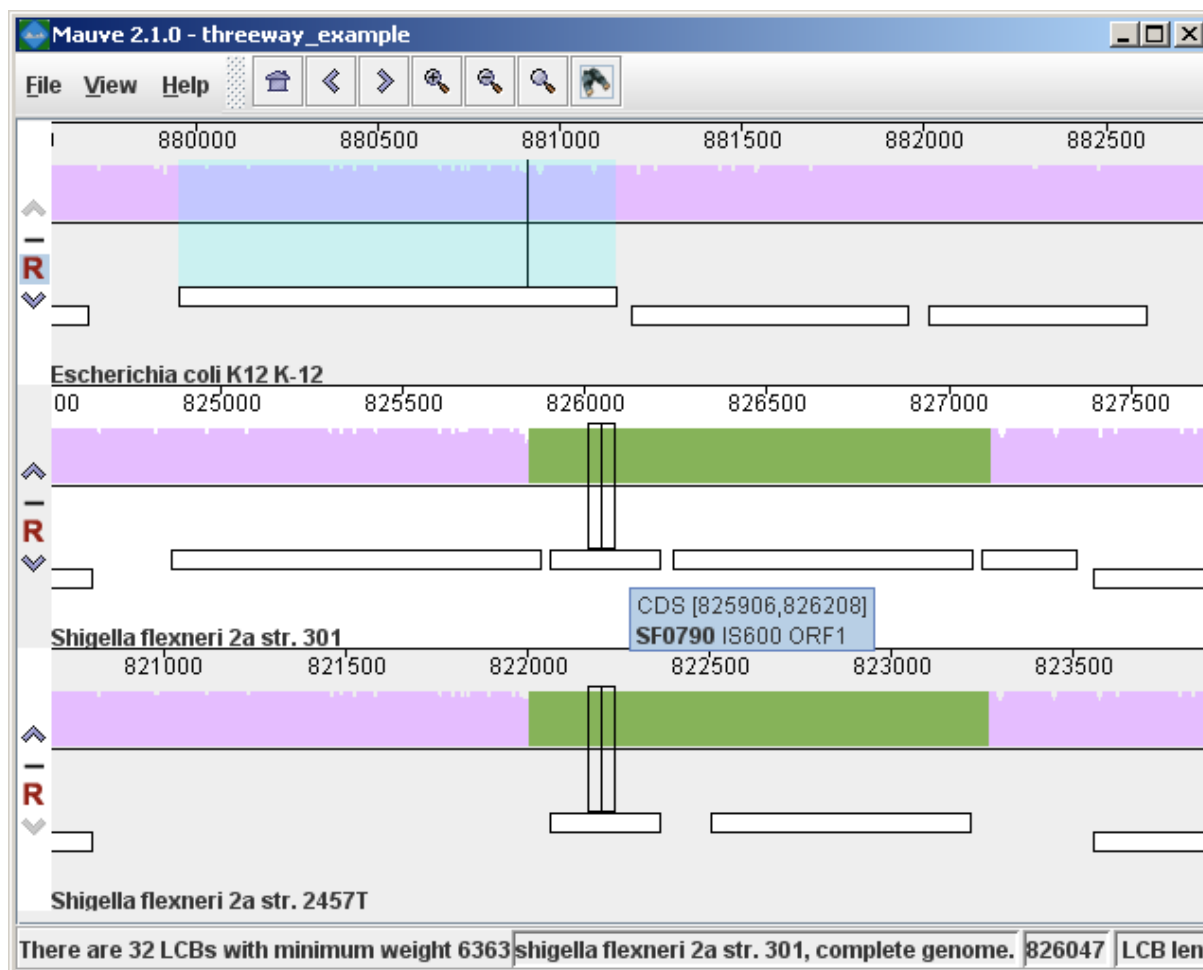


Searching for annotated features

Every good molecular biologist has a favorite gene. Since version 2.0.0, Mauve has a convenient interface to make finding your favorite gene easy. Annotation can be searched by by gene name, product description, amino acid sequence, and other information. The search feature can be activated by clicking on the binoculars button in the toolbar, or through the "View->Go To->Find Features" menu item. Mauve will then present a search interface window:



In the present example, the alignment of three *E. coli* and *Shigella* has been searched for genes that encode products related to penicillin. The search results for each genome are shown at right. A particular gene, *dacC* has been selected. When *dacC* is clicked, the display refocuses itself onto the region surrounding the *dacC* gene, which is highlighted in light blue:



Interestingly, the penicillin binding protein encoded by *dacC* appears to have been pseudogenized in the *Shigella* genomes. Inspection of the surrounding region reveals two CDS annotated as IS600, a well known transposable element that has colonized the *Shigella*. The IS600 has inserted itself near the end of the *dacC* gene.

Creating publication quality graphics with Mauve

Although Mauve offers a means to export an image file of the current display, using the "File->Export Image..." menu item, the exported images are raster graphics and may not be ideal for journal publication. To create publication quality EPS or PDF graphics with Mauve, it is better to print directly to a postscript file or a PDF. This is done by selecting "File->Print" and then choosing a PDF or postscript output from the print dialog. On Mac OS X, there is an option in the print dialog to "Save to PDF" which can easily be used. Under Windows, there is no built in PDF renderer, although several third-parties sell software with this feature. Adobe Acrobat includes a PDF renderer, and companies such as [FoxIt](#) and [pdf995](#) offer free PDF rendering software. Another option is to use the "Print to file..." checkbox in the windows print dialog. If a postscript printer driver has been installed, the resulting file will be in postscript (EPS) format. A postscript printer driver can be installed in Windows without actually connecting a real printer, by going to the Printers control panel, and choosing "Add printer." Select a "local printer attached to this computer" and do not "Automatically detect and install my Plug and Play printer". Click Next. Select "Use the following port" and select "FILE: (Print to File)". Click Next. When prompted for a printer model, select Apple Color LW 12/660 PS. Click Next. Choose a printer name and you're done.

Once a postscript or PDF file has been created, it is often desirable to edit the file. The best program to do so is Adobe Illustrator, as it has been designed for editing vector graphics files such as postscript and PDF. Alternative graphics editing software such as Xara Xtreme or the open-source Inkscape (version 0.46 or later contains PDF import) may be worth investigating.

While it may seem like a hassle to process Mauve images in vector formats like PDF or EPS, your effort will be rewarded with crisp print quality that far exceeds a grainy and pixelated screenshot. Moreover, Mauve graphics in PDF or EPS format can be scaled to print in large multi-page formats or on posters without loss of visual quality.

Exporting sets of positionally orthologous features (genes, CDS, tRNA, and so on)

Beginning with version 2.3.0, Mauve has the ability to identify sets of positionally orthologous sequence features and export a list of such annotated features. Annotated features can be CDS, gene, rRNA, tRNA, or misc_RNA. With an alignment generated by progressiveMauve, the orthologs can be exported by selecting "Tools->Export->Export Positional Orthologs" from the menu. A window will then appear requesting the location to save the ortholog output file(s) along with some configurable parameters for the ortholog identification algorithm. One parameter is the permissible range of nucleotide identity for ortholog sequences. To be called as orthologs, a pair of annotated features must have an average pairwise nucleotide identity within the specified range. The nucleotide identity is calculated only over conserved regions shared by the pair of annotated features, not including large gaps. A second parameter is the "coverage" of the pair of annotated features in the alignment. The coverage is calculated as the fraction of a feature that is aligned to the other feature. If X% of the length of feature A is aligned to feature B, and Y% of the length of B is aligned to feature A, and X and Y are both within the user-specified range of allowable coverage, then A and B will be considered orthologous.

Once Mauve has identified all pairs of positionally orthologous features in the input genomes, it then applies transitivity to the ortholog predictions. If A is found homologous to B, and B to C, then A must also be homologous to C, even if A and C do not meet the nucleotide identity or coverage requirements specified above. It has been argued in the literature that the evolutionary relationship of orthology can not be applied transitively. This is true in general, and for this reason one should consider positionally orthologous features inferred by Mauve as just that: inferences, which could be incorrect in some cases. Frequently homologous genes located in the same chromosomal context will indeed be orthologs and so inference of additional orthologs by transitive application of orthology will yield correct results. But occasionally evolutionary processes such as gene conversion with duplicate gene copies may create a situation where the positional homolog is a paralog. For this reason we suggest that users of the ortholog export feature exercise caution when interpreting the evolutionary history of positional orthologs generated by Mauve.

In addition to parameters controlling the ortholog inference procedure, the export window contains an option to output a file of multiple alignments corresponding to each ortholog set. The `/locus_tag` qualifier of GenBank annotations is used to give names for orthologous features in the ortholog output files.

Keyboard control reference

Function	Keystroke
Zoom in	Ctrl + Up
Zoom out	Ctrl + Down
Scroll display left	Ctrl + Left
Scroll display right	Ctrl + Right
Large left scroll	Shift + Ctrl + Left
Large right scroll	Shift + Ctrl + Right
Open an existing alignment file	Ctrl + O
Print the current view	Ctrl + P
Set the page layout for printing	Ctrl + Shift + P
Export the current view as an image	Ctrl + E
Close the current alignment window	Ctrl + W
Quit Mauve	Ctrl + Q
Show or hide LCB connecting lines	Shift + L
Show or hide LCB outlines	q

Other Color Schemes

When the "Full Alignment" option has been disabled, several color schemes become available. These color schemes are not available when viewing a full alignment with a similarity profile display.

LCB Color

Each locally collinear block (LCB) is assigned a unique color.

Multiplicity Color

Matches are colored differently based on their *Multiplicity*, where Multiplicity is defined as the number of matching genomes. e.g. A match with multiplicity=5 indicates that 5 genomes contain that homologous region of sequence.

Multiplicity Type Color

Matches are colored differently based on which genomes they match in, or their *multiplicity type*. For example, regions that match in genomes A, B, and D would be colored differently than regions matching in B, C, and D, even though the matches have the same multiplicity. The normalized version of multiplicity type color mode makes the colors used more distinguishable by eliminating unused multiplicity types from the color palette.

Offset Color

Matches are colored based on their *generalized offset*. The generalized offset is a number that summarizes the relative positioning of the match in each genome. Matches with similar generalized offsets will generally be collinear with each other. Thus regions of potential collinearity can be visualized using this color mode without actually performing LCB detection. This can be especially useful when comparing a large set of diverse genomes because the current implementation of LCB detection doesn't identify rearranged regions that exist in only a subset of the genomes being aligned. With offset color, such regions will appear as a continuously colored region in the Mauve viewer.

Mauve Output File Formats

When a genome alignment is created, Mauve creates several output files containing data related to the alignment. Two of these files, the `.mauve` and `.alignment` files actually contain the alignment in two different formats. The other files contain auxiliary information such as the genome phylogenetic guide tree that was used for alignment, an identity matrix for the genomes, the location of *backbone* -- regions conserved among all genomes, and the locations of *islands* -- regions where one or a subset of the genomes has a unique sequence element.

The following sections describe the information contained by each of these files and their associated file formats.

The `.alignment` file and the XMFA file format

The `.alignment` file contains the complete genome alignment generated by Mauve in the eXtended Multi-FastA (XMFA) file format. This standard file format is also used by other genome alignment systems that align sequences with rearrangements. The XMFA file format supports the storage of several collinear sub-alignments, each separated with an `=` sign, that constitute a single genome alignment. Each sub-alignment consists of one FastA format sequence entry per genome where the entry's defline gives the strand (orientation) and location in the genome of the sequence in the alignment.

The general structure of the file format as described by its author ([Michael Brudno](#)) is as follows:

```
>seq_num:start1-end1 ± comments (sequence name, etc.)
AC-TG-NAC--TG
AC-TG-NACTGTG
...

> seq_num:startN-endN ± comments (sequence name, etc.)
AC-TG-NAC--TG
AC-TG-NACTGTG
...
= comments, and optional field-value pairs, i.e. score=12345

> seq_num:start1-end1 ± comments (sequence name, etc.)
AC-TG-NAC--TG
AC-TG-NACTGTG
...

> seq_num:startN-endN ± comments (sequence name, etc.)
AC-TG-NAC--TG
AC-TG-NACTGTG
...
= comments, and optional field-value pairs, i.e. score=12345
```

Non-standard XMFA formatting used by the Mauve GUI

The Java-based Mauve alignment viewer requires some non-standard formatting of the XMFA file in order to display it. Most importantly, the Mauve viewer requires that every nucleotide of the input genome sequences be recorded exactly once in the XMFA file. Thus, nucleotides in one genome can not to multiple sites in a second genome. For segments of a genome that did not align and were outside LCBs, these segments must be given as ungapped singleton entries in the XMFA:

```
> seq_num:startN-endN ± comments
ACTCAGGTTATCG...
=
```

A second non-standard formatting requirement is that the first LCB entry in the XMFA list all input genome sequences, even if they did not align in that LCB. Genomes that have no sequence in that LCB are given using 0-0 as the coordinate range. For example, in an alignment of four genomes where only two were aligned in the first LCB, the initial LCB might look like:

```
> 1:0-0 +
```

```
>2:1-377 +
ACGA---TAAAATTCCC...

>3:1-422 -
ACTACCCTACAATTGGC...

>4:0-0 +
=
```

The Mauve alignment file format

The .mauve or .mln file also contains a representation of the genome alignment. Instead of including every aligned nucleotide as in the XMFA format, the Mauve alignment format stores the coordinates of large exactly matching regions to save space. For similar genomes the Mauve alignment format saves a significant amount of disk space over the XMFA format.

The mauve alignment format begins with a single line stating the revision of the file format, followed by several lines describing the sequences that were aligned. Using the alignment of three *Salmonella* genomes as an example:

```
FormatVersion      4
SequenceCount      3
Sequence0File      D:S_typhi.fas
Sequence0Length    4809037
Sequence1File      D:S_typhi2.fas
Sequence1Length    4791961
Sequence2File      D:S_typhimurium.fas
Sequence2Length    4857432
IntervalCount      69
```

Currently Mauve uses version 4 of its alignment file format. The next line contains the token SequenceCount which is used to specify the number of sequences aligned. Two lines are then given for each sequence, the first specifying the location of the original sequence file and the second specifying the sequence length in nucleotides. The final line contains the token IntervalCount which specifies the number of locally collinear blocks that were found among the aligned genomes.

The remainder of the file contains a number of Interval definitions, each of which specifies an LCB-- one collinear region of aligned sequence. Together, these LCBs make up a complete genome alignment with rearrangements. Here is an example interval definition:

```
Interval 3
153 292618 -2687311      294793
GappedAlignment
7 292771 -2687304      294946
GCCTGCG
GCCTGCG
CCATGTC
53 292778 -2687251      294953
GappedAlignment
1 292831 -2687250      295006
A
A
G
127 292832 -2687123      295007
```

Each LCB begins with an Interval token specifying the relative position of the LCB within the first genome aligned (the reference genome). Subsequent lines specify the actual alignment. When constructing an alignment, Mauve chooses a set of multi-MUMs (exactly matching regions present in each genome aligned) to anchor its alignment with. Each Interval definition records the position of these multi-MUM anchors in addition to alignments of the regions between anchors that were calculated using Clustal-W. In the example above, the line 153 292618 -2687311 294793 records a multi-MUM of length 153, with a left-end at 292,618 in *S_typhi*, on the opposite strand at position 2,687,311 in *S_typhi2*, and at 294,793 in *S_typhimurium*.

The next 5 lines in the example above give an alignment of inexact matching sequence generated by ClustalW. The token ClustalResult indicates that the following lines belong to such an alignment. The next line gives the total length of the (possibly

gapped) alignment and the left-end of the Clustal alignment in each genome. Finally, the next three lines (one per sequence aligned) record the actual alignment.

Each Interval records one or more of the multi-MUM and Clustal alignment entries which, when strung together, can specify a complete alignment over the region spanned by the LCB.

The islands file

The .islands file contains a tab-delimited text listing of genomic islands found in the alignment. Each island represents a region of the alignment where one or more genomes have a sequence element that one or more others lack. In the current Mauve implementation, an island is defined by the genome coordinates of one sequence where another genome contains a gap of length n or longer in that part of the alignment. The length of gaps that constitute islands can be set with the Minimum Island Size field of the Align sequences dialog box. For example, if n was defined as 5 for the following alignment:

```
Genome 0: ACACGTTTCGCTTCGAAA
Genome 1: ACAC-----TTCGAA-
Genome 2: ATACGATCGCTTCGTAA
```

We would say that genomes 0 and 2 have an island at positions 5 through 10. Each line of the .islands file records a single island in the form: GenomeA # <tab> leftA <tab> rightA <tab> GenomeB # <tab> leftB <tab> rightB. So in the .islands file, our example islands would be recorded as:

```
0 4 11 1 4 5
1 4 5 2 4 11
```

The first line records that in Genome #0 nucleotides 4 through 11 align with nucleotides 4 through 5 in Genome #1. Similarly, the second line records that nucleotides 4 and 5 of Genome #1 align with nucleotides 4 through 11 of Genome #2. In both cases the island length is 6 and can be calculated as $\text{absolute}(\text{rightA} - \text{leftA}) - (\text{rightB} - \text{leftB})$. Note that negative left and right values indicate the inverse orientation (the opposite strand).

The original Mauve backbone file

The .backbone file records regions of the alignment where sequence is conserved among *all* of the genomes being aligned. The current Mauve implementation defines a conserved region as an area of the alignment at least x nucleotides long that contains no gaps as long or longer than y nucleotides. When using the Align sequences window to perform an alignment, the values of x and y are fixed to the minimum island size. These values can be set explicitly using the command-line mauveAligner application.

Each line of the .backbone file records a single conserved segment. Left and right end coordinates of the conserved segment are given for each genome sequence. For example, the line:

```
22256      22371    20147    20299    22255    22370
```

Would indicate that nucleotides 22,256 through 22,371 from the first genome are conserved in the second and third genomes from 20,145 through 20,299, and 22,255 through 22,370 respectively. Two entries exist per genome, and each entry is tab-delimited. Negative valued coordinates indicate an inverted region (on the opposite strand).

The Progressive Mauve backbone file

Progressive Mauve utilizes a revised backbone file format which reflects its ability to align regions conserved among subsets of the genomes under study. A short example of the backbone file format is:

```
seq_0_leftend  seq_0_rightend  seq_1_leftend  seq_1_rightend  seq_2_leftend  seq_2_rightend
1              15378           1              15377           1              15377
16728         19795           15378          18446           15378          18445
0              0               18447          18668           18446          18667
19796         20566           18669          19439           18668          19438
```

The first line is a header line, indicating the information contained by each column. Each subsequent line corresponds to a segment of DNA conserved among two or more genomes. Thus, the first line indicates that the segment between coordinates 1 and 15378 in the first genome is homologous to the segment between coordinates 1 and 15377 of the second and third genomes. Similarly, the second line indicates that the segment [16728-19795] in the first genome is homologous to [15378-18446] in the second genome

and [15378-18445] in the third genome.

An island exists in the first genome between [15379-16727], and its existence is given in the backbone file as a lack of any line containing that segment. The seq_0_rightend column skips from 15378 on line 1 to 16728 on line two. By default, the rows of the backbone file are sorted on the seq_0_leftend column (absolute value). To infer islands in seq 0, we can thus simply compare the rightend of one line to the leftend from the subsequent line. The data can be trivially processed to observe islands in other genomes using a spreadsheet program like OpenOffice Calc or MS Excel. Simply sort the rows on the absolute value of the column for a sequence (e.g. seq_2_leftend) and then compare right-end to left-end on the subsequent line.

An island (or subset backbone) also exists in the second and third genomes. The existence of the subset backbone is given on the third line, where seq_0_leftend and seq_0_rightend both have zero values to indicate that the first genome lacks any detectable homology to the segments [18447-18668] in the second genome and [18446-18667] in the third genome.

The guide tree file

The guide tree is the standard Newick tree file format. A decent description of the Newick tree file format can be read here: <http://evolution.genetics.washington.edu/phylip/newicktree.html>

The identity matrix file

This is tab-delimited text where rows and columns are genomes in the order input to the aligner. Identity scores range between 0 and 1, where 0 indicates that no identical homologous nucleotides were found, and 1 indicates that every homologous nucleotide was identical.

The permutation matrix file

This is a tab-delimited text file that records the order and orientation that each LCB occurs in the aligned genomes. The permutation matrix file can be used to infer phylogenetic rearrangement history using tools such as BADGER, GRAPPA, MGR, and others. This file is generated by adding the command-line option --permutation-matrix-output=<filename> when running mauveAligner. An example of a file with three genomes and seven LCBs follows:

```
0  1  2  3  4  5  6
1  2  3 -6 -5 -4  0
0  1  2  3 -6 -5 -4
```

Each genome is recorded on a single line, with lines ordered according to the order of input genomes. The LCB arrangements are recorded on each line, with the first genome used as a reference genome to assign numeric identifiers to LCBs. A minus sign (-) indicates that a block is inverted relative to the reference genome.

The LCB boundary file

This is another tab-delimited file that complements the permutation matrix file with information about the LCB boundaries. This file can be used, for example, to derive the lengths of blocks and by extension, the lengths of genome rearrangements predicted by a rearrangement history reconstruction algorithm. An example of the file format for three genomes follows:

```
#seq0_leftend  seq0_rightend  seq1_leftend  seq1_rightend  seq2_leftend  seq2_rightend
1              11936          250           14147          1             11951
13856         54361         16067         56756         -3398531      -3439222
57092         120151        3854873       3917932       -41024        -106027
...
```

The file starts with a header line defining the content of each column. Each subsequent row defines the left- and right-side LCB boundaries for one LCB. Thus, the first row indicates that nucleotides 1-11936 from genome 0 correspond to nucleotides 250-14147 from genome 1 and nucleotides 1-11951 from genome 2.

The SNP file

This tab-delimited file can be created from alignments using Mauve version 2.3.0 and later. For every polymorphic site in an alignment, the SNP file records the nucleotides present in each genome at that site, along with the sequence coordinates of the site

in each genome. An example on three genomes is as follows:

SNP pattern	sequence_1	sequence_2	sequence_3
AAT	5276590 5246627 394		
TTC	5276784 5246821 588		
AAC	5277418 5247455 1222		
MAA	5278225 5248262 2030		
AAC	5282804 5252841 6609		

The first column lists the SNP pattern with sequences ordered the same as when input for alignment. The top of the XMFA file can be used as a reference for sequence filenames. Each subsequent column contains coordinates for each of the genomes in turn. Thus each line lists a SNP pattern and the location of that SNP in each genome. Note that IUPAC nucleotide ambiguity codes are considered as possible SNPs -- hence the M in the fourth example SNP above.

The orthologs file

The orthologs file is a tab-delimited file that can be created from progressiveMauve alignments using Mauve version 2.3.0 and later. The ortholog file lists groups of annotated and unannotated genes that are predicted to be *positionally* orthologous by whole-genome multiple alignment. Each row in the file lists a group of orthologous genes, along with the index of the genome from which the gene derives, the name of the gene (if given in the annotation file) and its sequence coordinates in the global coordinate system of that genome. Entries within a line are tab-delimited and colon-delimited. An example for 4 genomes follows:

```
0:Z03:2818-3750 1:c04:3512-4444 2::2801-3733 3:ECSE_03:2800-3732
0:Z04:3751-5037 1:c05:4445-5731 2::3734-5020 3:ECSE_04:3733-5019
0:Z05:5251-5547 1:c07:5945-6241 1:c08:6021-6269 2::5234-5530 3:ECSE_05:5233-5529
0:Z06:5700-6476 1:c10:6301-7077 3:ECSE_06:5682-6458
```

The first line lists a group of four orthologous genes, with one gene coming from each genome. In the first entry 0:Z03:2818-3750, the leading 0 refers to the genome's index, with indices assigned in the order the genomes were input for alignment. Thus genome 0 is the first genome, 1 is the second, and so on. The next part, Z03, refers to the locus_tag identifier for the annotated gene. The third colon-delimited part refers to the coordinate range of the annotated gene. The remainder of the line lists out genes in the other three genomes found to be positionally orthologous. In the case of genome 2 we have an entry 2::2801-3733. In this case there was no gene annotated in the region, but a region was found to be positionally orthologous, and so the coordinates of that region are listed without a locus_tag.

The third line in the example highlights a situation where multiple annotated genes in one genome are found to be orthologous to a single gene in other genomes. In this case, two overlapping genes were annotated in genome 1, and each of those genes individually was predicted to be positionally orthologous to the corresponding genes in other genomes. Since they overlap and are orthologous to the same genes in other genomes, they are considered a group of positional orthologs. Thus, any group of positional orthologs may contain multiple genes from a single genome.

The fourth line in the example illustrates the situation where one of the genomes does not have a positional ortholog of the genes. In this case, genome 2 lacks a region predicted to be positionally orthologous.

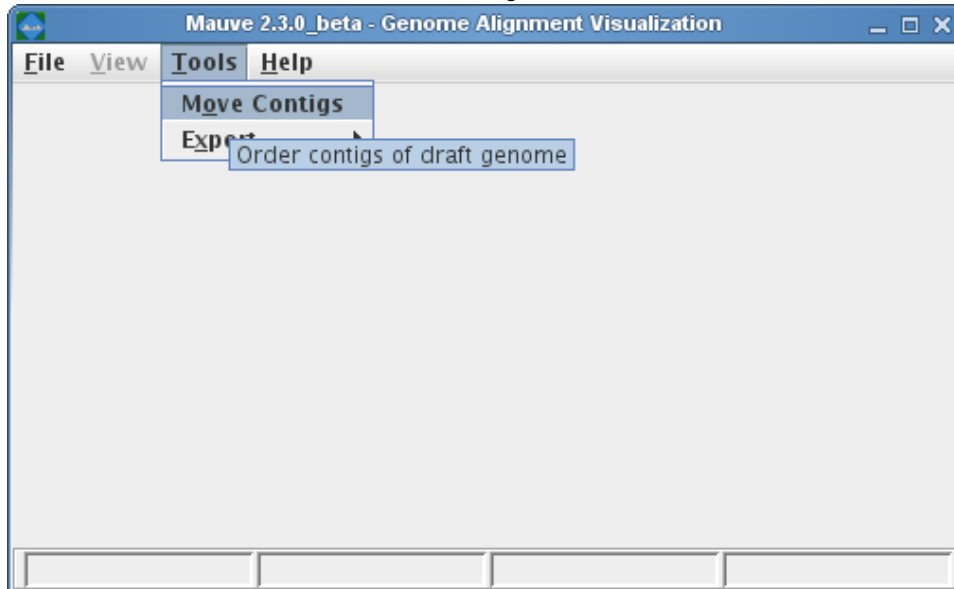
Reordering contigs in draft genomes

When to use Mauve Contig Mover (MCM)

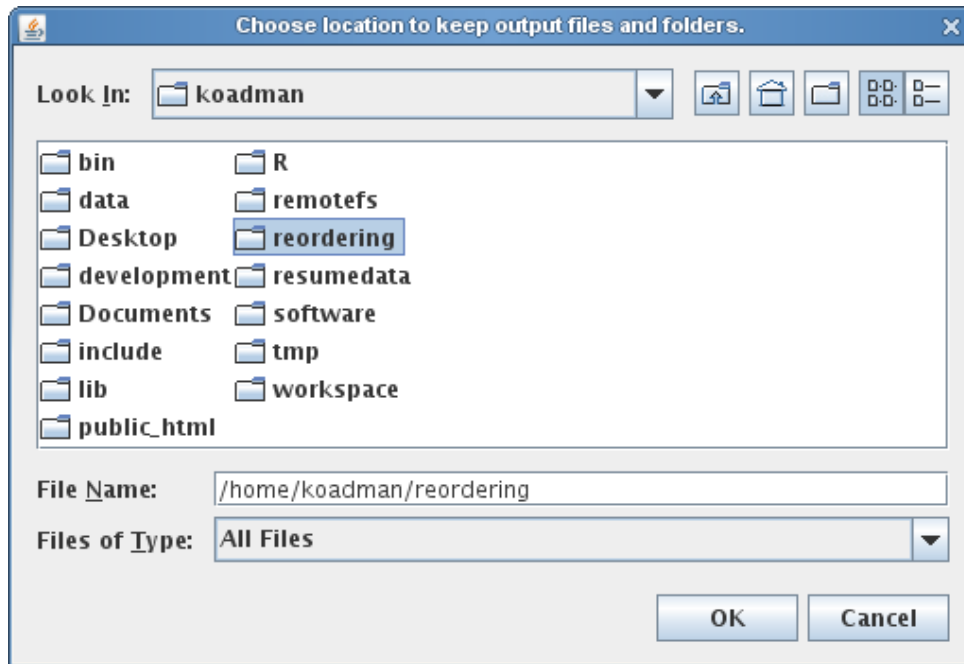
The Mauve Contig Mover (MCM) can be used to order a draft genome relative to a related reference genome. The functionality of this software module has been described in [Rissman et al. 2009](#), a publication in *Bioinformatics*. The Mauve Contig Mover can ease a comparative study between draft and reference sequences by ordering draft contigs according to the reference genome. In many cases, true rearrangements in the draft relative to the reference can be identified. The quality of the reorder is limited by the distance between the sequences, as indicated by the amount of shared gene content among the two organisms. A more distant reference will usually yield fewer ordered draft genome contigs, and may also induce erroneous placements of draft contigs. In addition to ordering contigs, MCM also orient them in the most likely orientation, and, if annotated sequence features are specified in an input file (e.g. with GenBank format input for the draft), MCM will output adjusted coordinates ranges for the features.

Using Mauve Contig Mover

Mauve Contig Mover can be launched from the Tools->Order Contigs Menu of The Mauve Viewer.



Once the Mauve Contig Mover has been launched, it starts by requesting the user to specify an output directory. MCM will create a series of mauve alignments in the output directory, structured into several subfolders, so creating a new, empty output folder is often best to minimize clutter. The output directory selection is shown below.



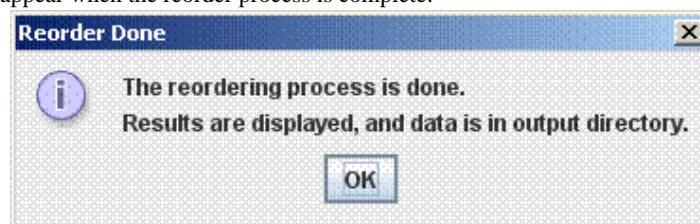
Once the output directory has been set, a window similar to the Progressive Mauve Alignment Window appears. Using the Progressive Mauve alignment window is described in more detail in the section "[Constructing a genome alignment](#)", but when used for reordering contigs, the following additional notes and constraints apply:

1. Only two sequences should be entered. The first must always be the reference, the second the draft genome to reorder. The first may also be a draft, but only the second will be reordered.
2. The reference genome may be in any of the allowable file formats, the draft must either be in a fasta or genbank file. If the draft is in genbank format, a unique identifier must be specified in either the LOCUS tag of each contig.
3. The alignment parameters have already been adjusted for what is generally best for draft genomes. This may depend on the draft and reference, and can be adjusted if needed.

The Reordering Process

The reordering will begin when the start button is pressed. It is an iterative process, and may take anywhere from a half hour to several hours. It may be cancelled at any point (intermediate results will be viewable). If it is canceled after the first reorder the reordered draft genome will be available in a Multi-FastA file in the corresponding output directory, although an alignment of the canceled ordering step will not be present. If the ordering process is not manually ended, it will terminate when it finds an order has repeated. Sometimes the order will cycle through several possibilities; this indicates it cannot determine which of them is most likely. Alignment parameters may be changed before reorder starts or any time between alignments.

The following message will appear when the reorder process is complete:



The MCM Output Files

MCM will output a series of folders called alignment1-alignmentX, representing each iteration of the reorder. Each has the standard Mauve alignment files, as described in the section [Mauve Output File Formats](#). Each folder also has an additional file called name_of_genome_contigs.tab, where "name_of_genome" is the draft genome's name. This file is included for ease of interpreting reorder results, and also acts as an index to the fasta as the contig orders and orientations change (even if the draft was originally input as a genbank, after the first alignment, it will be converted to a fasta with annotation information preserved in a file described below). The file is divided into 3 sections, each containing a list of contigs. The data for each contig includes its label (name), its location in the genome (numbered in pseudocoordinates from the first to last contig; these coordinates can be entered into the View->Go To->Sequence Position menu option to jump to that contig using the Mauve Alignment Viewer), and whether it

is oriented the same as originally input, or was complemented. The three sections are described below:

1. **Contigs to reverse:** This section contains contigs whose order is reversed with respect to the previous iteration. Note that contigs in this section may be oriented the same as originally input, this can be determined from the forward or complement designation.
2. **Ordered Contigs:** This is a list of all the contigs in the order and orientation they appear in the fasta for the draft of this iteration of the reorder. Since these include all the contigs in the original input, those with no ordering information (no aligned region) will be clustered at the end. These will appear as contigs with no LCBs at the end of the draft genome.
3. **Contigs with Conflicting Order information:** This is a list of contigs containing LCBs suggesting multiple possible locations. These may be of interest to verify positioning, or to look at points of potential rearrangement or misassembly.

If the draft was input as an annotated genbank file, a second file will appear in each alignment folder called `name_of_genome_features.tab`. This file will contain a line for each annotation, information about its current orientation and location (which will change if the contig is inverted), coordinates from the previous iteration (indicating relative orientation), and whether it is reversed from the original input. It will also have a label field used to identify each feature. This will be gotten from the annotation, as checked in the following order: `db_xref`, `label`, `gene`, and `locus_tag`.

Thus, the folder with the highest numbered alignment contains a fasta and one (or possibly two) descriptor files representing the final order of the draft genome.

Reordering contigs from the command-line (batch mode)

In situations where it is necessary to order contigs in a large number of draft genomes it is often more desirable to automate the process using command-line interfaces and scripts. Mauve Contig Mover supports command-line operation through the Mauve Java JAR file.

Given a reference genome file called "reference.gbk" and a draft genome called "draft.fasta", one would invoke the reorder program with the following syntax:

```
java -Xmx500m -cp Mauve.jar org.gel.mauve.contigs.ContigOrderer -output results_dir -ref reference.gbk -draft draft.fasta
```

The file `Mauve.jar` is part of the Mauve distribution. On windows systems it can usually be found in `C:\Program Files\Mauve\XMauve.jar` where `X` is the version of Mauve. On Mac OS X it is located inside the Mauve application. For example, if Mauve has been placed in the OS X applications folder, `Mauve.jar` can be found at `/Applications/Mauve.app/Contents/Resources/Java/Mauve.jar`. On Linux, `Mauve.jar` is simply at the top level of the tar.gz archive. In the above example command, it will be necessary to specify the full path to the `Mauve.jar` file.

Using mauveAligner from the command-line

The mauveAligner and progressiveMauve alignment algorithms have been implemented as command-line programs included with the downloadable Mauve software. When run from the command-line, these programs provide options not yet available in the graphical interface.

Where to find mauveAligner and progressiveMauve

mauveAligner and progressiveMauve are included with the Mauve software. On Windows they are located in the directory where Mauve was installed, usually `C:\Program Files\Mauve`. On 64-bit Windows platforms, the binaries in the `win64` subdirectory may be used instead. On Mac OS X the mauveAligner and progressiveMauve binaries are packaged within the Mauve application. Relative to Mauve.app the paths will be `Mauve.app/Contents/MacOS/mauveAligner` and `Mauve.app/Contents/MacOS/progressiveMauve`. The binaries can be copied to a more convenient location. However, if using Mauve versions 2.0.0 or earlier, be sure to also copy the `muscle_aed` binary to the same location. On Linux the mauveAligner and progressiveMauve binaries are in the top level directory of the software distribution, with 64-bit binaries in the `linux-x64` subdirectory.

Alternatively, mauveAligner and progressiveMauve can be compiled from source for your particular platform. See the [Mauve Developer's Guide](#) for more details.

mauveAligner arguments and examples

Usage:

```
mauveAligner [options] <seq1 filename> <sml1 filename> ... <seqN filename> <smlN filename>
or
mauveAligner [options] <Multi-FastA file>
```

Each entry like `<seq filename>` denotes the name of a sequence file on disk that contains a genome to align. Sequence files can be in FastA, Multi-FastA, or GenBank flat file format. If an individual file contains several sequence entries, they will be concatenated and the whole concatenated sequence will be aligned to the sequences in the other files.

Each sequence must have a corresponding Sorted Mer List (SML) file name given. If the SML file does not exist, mauveAligner will create it. Alternatively, mauveAligner can be given the name of a Multi-FastA file containing all the genomes to align. In this case mauveAligner assumes one genome per Multi-FastA sequence entry and will align the entries to each other.

Example 1. Aligning *E. coli* and *Salmonella* genomes stored in separate GenBank files:

```
mauveAligner --output=ec_sa.mauve --output-alignment=ec_vs_sa.alignment ecoli.gbk
ecoli.gbk.sml salmonella.gbk salmonella.gbk.sml
```

Example 2. Aligning several poxvirus genomes stored in a single Multi-FastA file:

```
mauveAligner --output=poxvirus.mauve --output-alignment=poxvirus.alignment
all_poxvirus.mfa
```

Example 3. Generating initial alignment anchors without actually aligning:

```
mauveAligner --no-recursion --no-gapped-alignment -o poxvirus.mauve all_poxvirus.mfa
```

Example 4. Aligning a group of Hepatitis C virii known to have no rearrangements:

```
mauveAligner --collinear --output=hcv.mauve --output-alignment=hcv.alignment hcv.mfa
```

Example 5. Aligning two yeast genomes, increasing the weight threshold to avoid spurious LCBs:

```
mauveAligner --weight=300 --output=two_yeast.mauve
--output-alignment=two_yeast.alignment S_cerevisiae.gbk S_cerevisiae.gbk.sml
S_bayanus.gbk S_bayanus.gbk.sml
```

Many other command line options can be given to the mauveAligner binary. A full listing follows.

General options:

```
--output=<file> Output file name. Prints to screen by default

--mums Find MUMs only, do not attempt to determine locally collinear blocks (LCBs)

--no-recursion Don't perform recursive anchor identification (implies --no-gapped-alignment)

--no-lcb-extension If determining LCBs, don't attempt to extend the LCBs

--seed-size=<number> Initial seed match size, default is log2( average seq. length )

--max-extension-iterations=<number> Limit LCB extensions to this number of attempts, default is 4

--eliminate-inclusions Eliminate linked inclusions in subset matches.

--weight=<number> Minimum LCB weight in base pairs per sequence

--match-input=<file> Use specified match file instead of searching for matches

--lcb-match-input Indicates that the match input file contains matches that have been clustered into LCBs

--lcb-input=<file> Use specified lcb file instead of constructing LCBs (skips LCB generation)

--scratch-path=<path> For large genomes, use a directory for storage of temporary data. Should be given two or more
times to with different paths.

--id-matrix=<file> Generate LCB stats and write them to the specified file

--island-size=<number> Find islands larger than the given number

--island-output=<file> Output islands the given file (requires --island-size)

--backbone-size=<number> Find stretches of backbone longer than the given number of b.p.

--max-backbone-gap=<number> Allow backbone to be interrupted by gaps up to this length in b.p.

--backbone-output=<file> Output islands the given file (requires --island-size)

--coverage-output=<file> Output a coverage list to the specified file (- for stdout)

--repeats Generates a repeat map. Only one sequence can be specified

--output-guide-tree=<file> Write out the guide tree used for CLUSTALW alignment to the designated file

--collinear Assume that input sequences are collinear--they have no rearrangements
```

Gapped alignment controls:

```
--no-gapped-alignment Don't perform a gapped alignment

--gapped-aligner=<muscle|clustal> Set the gapped alignment algorithm (Default is muscle)

--max-gapped-aligner-length=<number> Maximum number of base pairs to attempt aligning with the gapped aligner

--min-recursive-gap-length=<number> Minimum size of gaps that Mauve will perform recursive MUM anchoring on
```

(Default is 200)

Signed permutation matrix options:

`--permutation-matrix-output=<file>` Write out the LCBs as a signed permutation matrix to the given file

`--permutation-matrix-min-weight=<number>` A permutation matrix will be written for every set of LCBs with weight between this value and the value of `--weight`

Alignment output options:

`--alignment-output-dir=<directory>` Outputs a set of alignment files (one per LCB) to a given directory

`--alignment-output-format=<directory>` Selects the output format for `--alignment-output-dir`

`--output-alignment=<file>` Write out an XMFA format alignment to the designated file

mauveAligner Execution modes

MUM (Match) generation

Generates a list of unique and ungapped local multiple alignments shared by two or more of the sequences. Alignments among subsets of the input genomes are included unless the `--nway-filter` option is used. See the MUM file format reference.

Use the `--mums` command line option to invoke match generation without alignment. Other command line options that can be used in combination with `--mums` are `--seed-size` to specify a minimum weight for matches, and `--eliminate-inclusions` to remove overlapping regions among different matches. Acceptable values for seed size are odd numbers in the range of 5 to 31. The algorithm to remove overlapping regions gives preference to matches with higher multiplicity, where multiplicity is defined as the number of genomes that a match occurs in. When two overlapping matches have identical multiplicity, the shorter of the two matches is truncated to eliminate the overlap.

Breakpoint elimination

Performs greedy breakpoint elimination to determine Locally Collinear Blocks (LCBs). The resulting LCBs will all be above some minimum weight. Invoked when full alignment is disabled by adding `--no-gapped-alignment`.

Full alignment

Performs a complete alignment of the conserved regions among the input genome sequences using ClustalW progressive dynamic programming. This is the default execution mode.

Using progressiveMauve from the command-line

progressiveMauve also offers many command-line options to provide detailed control over alignment parameters. This section gives a reference of the parameters and some examples regarding their usage.

Command-line paramter reference

`--apply-backbone=<file>` Read an existing sequence alignment in XMFA format and apply backbone statistics to it

`--disable-backbone` Disable backbone detection

`--mums` Find MUMs only, do not attempt to determine locally collinear blocks (LCBs)

`--seed-weight=<number>` Use the specified seed weight for calculating initial anchors

`--output=<file>` Output file name. Prints to screen by default

`--backbone-output=<file>` Backbone output file name (optional).

`--match-input=<file>` Use specified match file instead of searching for matches

`--input-id-matrix=<file>` An identity matrix describing similarity among all pairs of input sequences/alignments

`--max-gapped-aligner-length=<number>` Maximum number of base pairs to attempt aligning with the gapped aligner

`--input-guide-tree=<file>` A phylogenetic guide tree in NEWICK format that describes the order in which sequences will be aligned

`--output-guide-tree=<file>` Write out the guide tree used for alignment to a file

`--version` Display software version information

`--debug` Run in debug mode (perform internal consistency checks--very slow)

`--scratch-path-1=<path>` Designate a path that can be used for temporary data storage. Two or more paths should be specified.

`--scratch-path-2=<path>` Designate a path that can be used for temporary data storage. Two or more paths should be specified.

`--collinear` Assume that input sequences are collinear--they have no rearrangements

`--scoring-scheme=<ancestral | sp_ancestral | sp>` Selects the anchoring score function. Default is extant sum-of-pairs (sp).

`--no-weight-scaling` Don't scale LCB weights by conservation distance and breakpoint distance

`--max-breakpoint-distance-scale=<number [0,1]>` Set the maximum weight scaling by breakpoint distance. Defaults to 0.9

`--conservation-distance-scale=<number [0,1]>` Scale conservation distances by this amount. Defaults to 1

`--skip-refinement` Do not perform iterative refinement

`--skip-gapped-alignment` Do not perform gapped alignment

`--bp-dist-estimate-min-score=<number>` Minimum LCB score for estimating pairwise breakpoint distance

`--mem-clean` Set this to true when debugging memory allocations

```
--gap-open=<number> Gap open penalty
--gap-extend=<number> Gap extend penalty
--substitution-matrix=<file> Nucleotide substitution matrix in NCBI format
--weight=<number> Minimum pairwise LCB score
--min-scaled-penalty=<number> Minimum breakpoint penalty after scaling the penalty by expected divergence
--hmm-p-go-homologous=<number> Probability of transitioning from the unrelated to the homologous state [0.0001]
--hmm-p-go-unrelated=<number> Probability of transitioning from the homologous to the unrelated state [0.000001]
--seed-family Use a family of spaced seeds to improve sensitivity
```

Examples of progressiveMauve usage

Example 1. Align three genomes from the input files genome_1.gbk, genome_2.gbk, and genome_3.gbk, saving the output to a file called threeway.xmfa

```
progressiveMauve --output=threeway.xmfa genome_1.gbk genome_2.gbk genome_3.gbk
```

Example 2. Align the same three genomes but also save the guide tree and produce a backbone file

```
progressiveMauve --output=threeway.xmfa --output-guide-tree=threeway.tree
--backbone-output=threeway.backbone genome_1.gbk genome_2.gbk genome_3.gbk
```

Example 3. Align the same three genomes, but do not detect forced alignment of unrelated sequence and do not create a backbone file

```
progressiveMauve --output=threeway_no_backbone.xmfa --disable-backbone genome_1.gbk
genome_2.gbk genome_3.gbk
```

Example 4. Detect forced alignment of unrelated sequence in the alignment produced in Example 3. Use custom Homology HMM transition parameters. Save a backbone file.

```
progressiveMauve --apply-backbone=threeway_no_backbone.xmfa --output=threeway.xmfa
--backbone-output=threeway.backbone --hmm-p-go-homologous=0.001
--hmm-p-go-unrelated=0.000005
```

Example 5. Compute ungapped local-multiple alignments among the input sequences and save them to a file called threeway.mums

```
progressiveMauve --mums --output=threeway.mums genome_1.gbk genome_2.gbk genome_3.gbk
```

Example 6. Compute an alignment of the same three genomes, using previously computed local-multiple alignments

```
progressiveMauve --match-input=threeway.mums --output=threeway.xmfa genome_1.gbk
genome_2.gbk genome_3.gbk
```

Example 7. Set a custom breakpoint penalty to cope with genomes where default penalty does not work. The default penalty can be extracted from the program's textual output, in this hypothetical example, the default penalty will be 100000.

```
progressiveMauve --output=threeway.xmfa --weight=50000 genome_1.gbk genome_2.gbk
genome_3.gbk
```

Example 8. Set a minimum scaled breakpoint penalty to cope with the case where most genomes are aligned correctly, but manual inspection reveals that a divergent genome has too many predicted rearrangements.

```
progressiveMauve --min-scaled-penalty=5000 --output=threeway.xmfa genome_1.gbk
```

```
genome_2.gbk genome_3.gbk
```

Example 9. Globally align a set of collinear virus genomes that reside in a single FastA file. Use seed families to improve anchoring sensitivity in regions below 70% sequence identity.

```
progressiveMauve --collinear --seed-family --disable-backbone --output=virus.xmfa  
all_virii.fasta
```

Mauve Version History

Version 2.3.1 (2009-11-11)

New features:

- * Compatibility with older versions of Mac OS X

Bugs fixed:

- * SNP output now handles mixed case sequence files and gap characters
- * Aligner memory usage has been streamlined, resulting in reduced memory requirements for alignments with many genomes
- * Fix for bug that prevented reordering GenBank-format draft genomes on the command-line
- * Fixed a crash when generating mums
- * Report an error when sequences containing gaps are used as input
- * Remove extra newlines in the ortholog alignment output
- * Fixed an OpenJDK rectangle drawing bug

Version 2.3.0 (2009-06-16)

Major new features:

- * Reordering contigs of draft genomes, published as Rissman et al 2009
- * Export a list of Single Nucleotide Polymorphisms present in an alignment (thanks to Meg Woolfit for the suggestion)
- * Export a file listing the genome arrangement as a signed permutation
- * Export a list of positionally orthologous CDS, tRNAs, or misc_RNAs (thanks to Elizabeth Skippington for the suggestion)
- * Export a file containing alignments of orthologous CDS, including alignments of unannotated regions predicted to be orthologous (thanks to Sam Sheppard for the idea)

Other new features:

- * On Linux it is now possible to launch Mauve from outside the Mauve directory (thx Dongying Wu)
- * Option to disable drawing the mouse cursor highlighting orthologous regions so that figures for manuscripts can be rendered (thanks to Andrew Kropinski for the idea)

Bugs fixed:

- * An alignment would complete successfully but the viewer would fail to load it because the alignment contained invalid data in the backbone file. This bug was very commonly encountered in 2.2.0.
- * An alignment with progressiveMauve would crash with an error like "cga doesn't fit" or another error message, when run on multicore computers. This was a race condition in progressiveMauve's parallelism.

Known issues:

- * Parallelism has been disabled for all supported platforms to avoid a race condition bug.

Version 2.2.0 (2008-06-23)

New features:

- * New menu option to disable the red contig boundary lines
- * Progressive aligner sensitivity/accuracy improvements

Bugs fixed:

- * progressiveMauve was not sensitive enough to small rearrangements, although it did have high predictive value
- * progressiveMauve was not properly extending LCBs in the 2.1.x releases
- * progressiveMauve did not work on 64-bit linux (indefinite waiting message)
- * Versions 2.1.x did not render vector graphics when printing to PDF or EPS

- * seed families are too memory intensive and are now disabled by default
- * segmentation fault (crash) on linux/mac due to a bug in c++ sort functions
- * update check causes a stall during GUI launch when the network is inaccessible
- * building from source was difficult

Known issues:

- * Mauve 2.2.0 for Macintosh can not run in parallel on more than one CPU
- * Additional bugs can be found at http://sourceforge.net/tracker/?group_id=181544&atid=897627

Version 2.1.1 (2007-11-27)

Bugs fixed:

- * The alignment programs mauveAligner and progressiveMauve did not work on Mac due to a missing shared library (libgomp.dylib)
- * progressiveMauve would occasionally crash during alignment scoring and greedy breakpoint elimination due to a math bug
- * The sequence navigator did not properly highlight feature search results when launched from the menu. It did work when launched from the toolbar binoculars button.

Version 2.1.0 (2007-11-20)

New features:

- * A graphical interface to reorder and hide genomes from the display, and to change the reference genome
- * A new graphical interface to change the viewing style for genome comparisons
- * Parallelized alignment algorithms that run faster on multi-core CPUs like the Intel Core Duo and the AMD Opteron
- * 64-bit implementations for Windows and Mac OS X. 64-bit machines with enough RAM can align 20 or more bacterial genomes
- * Seed family searches. Several seed patterns can be used simultaneously to improve sensitivity on heavily diverged genomes

Bugs fixed:

- * Several progressiveMauve alignment crashes
- * A mauveAligner bug whereby the aligner never finished on some datasets
- * Many more. For a full list of known and fixed bugs please see http://sourceforge.net/tracker/?group_id=181544&atid=897627

Known issues:

- * Mauve 2.1.0 for Macintosh requires OS X version 10.4 or later
- * A bug exists which can crash Mauve on Linux and Mac OS X during the "recursive anchor search."
- * Mauve does not work with GNU java, which is the default java in some Linux distributions
- * Sun JRE 1.6.0 for Linux [has a bug](#) that breaks alignment loading (it's not my fault!). It may be necessary to manually open the alignment file using "File->Open" after making the alignment.

Version 2.0.0 (2007-01-26)

Mauve 2.0 includes an accurate new alignment algorithm called **Progressive Mauve**. The progressive alignment algorithm has been described in Aaron Darling's Ph.D. thesis, available from http://gel.ahabs.wisc.edu/~darling/darling_thesis.pdf

Progressive Mauve features:

- * **Substantially improved sensitivity on divergent genomes.** Progressive Mauve can "climb a phylogenetic ladder" such that a pair of genomes with > 1 substitution per site can be aligned if phylogenetic intermediate genomes are included in the alignment
- * **Segments that are present in subsets of the genomes will be aligned.** The original Mauve aligned only segments conserved in all genomes.
- * **A configurable substitution matrix dictates the LCB and alignment scores.** The substitution matrix gives Mauve additional power in distinguishing true homology from random sequence similarity. In the original algorithm LCBs were scored based on the sum of multi-MUM lengths
- * **A random-walk homology test identifies conserved "backbone" segments.** Random walk statistics similar to those used in Gapped-BLAST are used to identify pairwise conserved segments. Pairwise homologies are then combined by transitive homology relationships to identify segments conserved in two or more genomes.

- * **Progressive Mauve can scale to hundreds of genomes.** We have successfully aligned over a hundred poxvirus genomes, hundreds of Hepatitis C Virus genomes, and the genomes of over thirty enterobacteria.

Anna Rissman has contributed a new annotation search interface that tremendously improves browsing genomes with annotated sequence features. The new search interface can:

- * Jump directly to a particular numerical coordinate in a particular genome
- * Jump directly to an annotated feature of a particular name
- * Search for and list annotated features by any part of their name or qualifying value. Searchable fields include things like CDS name, product description, GO terms, database x-refs, and amino acid subsequences

Other new features:

- * The windows installer supports installations by users without administrative privileges and also supports multiple concurrent installations
- * The windows installer includes Java 6 for machines without an existing Java installation
- * Alignments from Progressive Mauve can be viewed in "backbone color" mode, which colorizes segments based on the combination of genomes in which they are conserved.

Bugs fixed:

- * Infinite looping during LCB extension should be fixed. LCB extension may still be slow for some datasets, with questionable benefit. It can be disabled in the Parameters tab of the "Align sequences.." window.
- * Several bugs were fixed that would cause sporadic crashes in MUSCLE 3.6.

Known bugs:

- * Bug tracking has moved to the sourceforge web site. Please use the following address to report bugs:
http://sourceforge.net/tracker/?group_id=181544&atid=897627

Version 1.3.0 (2006-05-24)

New features:

- * Rearrangement history analysis by generalized transposition/block interchange, code courtesy of Mike Tsai. Use the green/blue/black toolbar button to perform an analysis. For details on the inference algorithm see Yancopoulos, S., Attie, O and Friedberg, R. "Efficient sorting of genomic permutations by translocation, inversion and block interchange" *Bioinformatics* 2005 21(16):3340-3346
- * Rearrangement history through the GRIMM/MGR server now supported, use the red/yellow/black toolbar button.
- * No more limit on the number of input sequences. This feature comes at the cost of increased memory consumption. Just because the limit has been removed doesn't mean that the alignment algorithm will do a good job handling a large number of sequences.
- * The mauveAligner command-line tool has the new option --lcb-match-input to extract matches from an existing alignment and use them as anchors when computing a new alignment.
- * MUSCLE has been upgraded to version 3.6.
- * The Mac OS X version is now a Universal binary with support for both Intel and PowerPC macs.

Bugs fixed:

- * MUSCLE on Windows now runs faster and uses less memory.
- * On unix, Mauve will search \$PATH for mauveAligner.
- * Eliminated the obnoxious behavior where Mauve would repeatedly pop up the console window during the alignment process
- * Fixed bugs when viewing alignments after shifting the LCB minimum weight slider.

Version 1.2.3 (2005-10-20)

New features:

- * Display each component of multi-part features (e.g. a multiple exon gene) as a linked series of boxes
- * Display repeat_region annotations from GenBank files
- * Improved load time for previously viewed alignments using a disk-based alignment cache
- * Added a help menu item that will clear the on-disk alignment cache

- * Colorized the similarity plots, grey was starting to depress our entire lab.
- * Support for highlighting arbitrary regions of the sequence by holding the shift key while clicking and dragging on the display
- * Faster nucleotide display when zoomed in closely

Bug fixes:

- * Fixed GUI interface for setting the minimum seed size (Thanks to Bill Bruno and Todd Treangen)
- * Fixed mouseover, highlighting, and display alignment in the LCB display mode (used when the Full Alignment option is disabled)

Version 1.2.2 (2005-06-11)

New features:

- * Ability to set the reference genome
- * Displays gene name or locus tag for annotated features

Bug fixes:

- * Uses less memory for MUSCLE alignments by restricting the size of segments aligned by MUSCLE
- * Alignment editor shows original sequence coordinates when an aligned character is clicked
- * Handles GenBank files with a large number of sequence entries (e.g. C. hominis)

Version 1.2.1 (2005-05-05)

New features:

- * Experimental support for alignment editing using Cinema-MX Right click on an LCB to trigger the edit menu
- * Annotated features are shown in both forward and reverse complement orientation
- * When clicking an annotated feature, a database xref popup menu shows options for viewing the feature in known databases. Currently supported databases are Entrez Protein and ASAPdb. Please contact us about support for other databases.
- * Choice of either MUSCLE or Clustal in the GUI or on the command line with the `--gapped-aligner` switch

Bug fixes:

- * Database xref speed improvement
- * Features not shown when viewing large regions to improve speed
- * Fixed an off-by-one bug that crashed the viewer when loading Salmonella alignments
- * Eliminated extraneous MUSCLE warning messages on the Mac OS X console.

Known issues:

- * In some cases MUSCLE on Windows can consume lots of system resources. If you are experiencing problems we suggest using Clustal for the time being.

Version 1.2.0 (2005-04-20)

New features:

- * **Image export.** Mauve can export the current display view in JPEG, TIF, and PNG formats at a user-defined resolution.
- * **MUSCLE Alignment.** Mauve now calculates gapped alignments using the [MUSCLE](#) DNA alignment method. Previously Clustal-W was used to calculate gapped alignments. MUSCLE provides significantly improved speed and accuracy over Clustal-W
- * **Collinear genomes.** When it is known that the genomes being aligned have no rearrangements, select the Assume collinear genomes option in the "Align sequences" window to tell Mauve that it should find the single best LCB. On the command line, this behavior can be invoked with the `--collinear` switch.

Bug fixes:

- * Fixed a memory leak in LCB extension that could quickly consume all system memory.
- * The 1.1.0 Windows release used a large amount of memory because it was compiled for use with 160 sequences instead of the

usual 16.

- * Fixed a bug parsing GenBank sequences which had extra newlines in them

Known issues:

- * On Mac OS X the muscle aligner generates lots of warning messages about not being able to open /proc/meminfo. These messages can be ignored and will be eliminated in future releases.

Version 1.1.0 (2005-03-30)

New features:

- * Mauve can launch a web browser to show a web database entry for annotated features that have a /db_xref qualifier in their GenBank entry. Currently ASAPdb and NCBI's PubGene databases are supported. Other databases may be added upon request.
- * Mauve bundles a text console display for informational and error messages. It can be viewed by selecting Show console from the Help menu.
- * Alignment generation has been multithreaded so that alignments can be generated while another alignment is being viewed.
- * A Print Preview and Page Layout dialog boxes provide greater control over printing.

Bug fixes:

- * A bug in mauveAligner which resulting in a "bad alignment" error message has been fixed.
- * Printing has been fixed to provide accurate high resolution images. Output is vectorized so images printed to PDF documents (e.g. using Acrobat or PDF995) can be easily embedded in publications and have a high quality look.
- * Screen space is used more economically when viewing a large number of sequences, allowing at least six genomes to display on a 1024x768 screen. Previously only three genomes would fit on a screen of that size.
- * additional fixed bugs can be found in the Mantis tracker

Version 1.0.1 (2005-03-08)

Bug fixes:

- * Unable to align genomes on Mac OS X 10.3 (due to a missing library)
- * Unable to run Mauve under Linux (Mauve was not set as executable)
- * Unable to load genome annotations after constructing an alignment when the source GenBank files reside in a different directory than the alignment output
- * Lower memory profile. Mauve now uses less memory when reading alignments, making the large Drosophila alignments available on the web site viewable on computers with over 1GB RAM.

Version 1.0 (2005-03-01)

New features:

- * Display of annotated features such as CDS, rRNA, tRNA and misc_features from GenBank format files
- * Full support for Mac OS X
- * Display of chromosomal boundaries and contigs of incomplete genomes
- * Display of contig/chromosome name and sequence coordinates
- * Source code released under the GNU General Public License. Mauve is now free software and can be modified and redistributed freely (in accordance with the GPL).

2004-12-16

Bug fixes:

- * Inter-LCB islands were (still) incorrect and are now fixed. This bug was originally reported by Val Burland in Dec. 2003 but didn't get completely fixed until now.
- * Vertical only display resize fixed
- * Fixed a bug in the LCB info display when the LCB slider had been adjusted

New Features:

- * Sequence similarity browsing in the display
- * More sensitive alignments by using inexact seeds. This means Mauve can be used on more taxa simultaneously and also on more divergent taxa.
- * Display alignment in the similarity browser can be accomplished with a single mouse click

2004-07-08

Bug fixes:

- * Brought PDF documentation up-to-date

2004-06-30

Bug fixes:

- * Fixed several missing icon bugs in linux and windows releases
- * Mauve now requires Java 1.4 to function correctly, a warning message is displayed if Mauve detects an older version of Java

New features:

- * Automatic checking for new releases, and on Windows, downloading the new release and starting the installer
- * Releases now include a User Guide in PDF format and a link to the online guide

2004-05-18

Bug fixes:

- * Fixed a bug that would result in unreasonably large print jobs

New features:

- * Implemented interactive LCB weight adjustment. This feature allows dynamic adjustment of the LCB weight within Mauve using a slider control. As the minimum LCB weight is changed the display updates with the corresponding set of LCBs.
- * Added toolbar buttons for basic navigational controls such as scrolling and zooming the display. Also added a color mode selector.
- * Added keyboard shortcuts for open (ctrl+o), close (ctrl+w), print (ctrl+p), and quit (ctrl+q).

Important changes:

- * The up and down arrows no longer zoom in and out, instead use ctrl+up and ctrl+down

2004-04-26

Bug fixes:

- * Fixed a bug that allowed users to zoom in too far
- * Fixed a bug that kept the screen from clearing after selecting close from the File menu
- * Fixed a bug that would cause the display to become unaligned after using the context menu to align the display and subsequently zooming out.
- * Fixed a bug that raised an exception when no matching regions were found among the sequences being aligned
- * Changed the .txt file extension to default to FastA file format instead of raw sequence data
- * Fixed a subtle bug that would occasionally cause a "horrible error" to be reported

New features:

- * Improved the Align sequences dialog interface to give the user an easy set of default values with optional control over the alignment parameters. Also added an interface to set the backbone parameters.
- * Added the ability to change ordering of genomes in the display using drag and drop. To enter and leave reordering mode press the h key. When in reordering mode genomes can be dragged and dropped into the desired order.

2004-02-19

Bug fixes:

- * Fixed a bug in the island coordinate output that caused the end positions of inverted regions to be huge (wrong) numbers. Removed obsolete -d and --stats command line options. Added a --id-matrix or -t command line option to print out an identity matrix. Mauve now creates an identity matrix by default.

2003-12-09

Bug fixes:

- * Fixed another bug in the LCB extension code that occasionally resulted in incorrect alignments.

2003-12-06

Bug fixes:

- * Fixed a serious bug that caused the first sequence to be misaligned in regions with numerous gaps.

2003-11-04

Bug fixes:

- * Fixed a minor file format problem with XMFA (eXtended Multi-FastA) output. Fixed a bug with reverse complement backbone determination.

2003-10-17

Bug fixes:

- * Updated the GUI interface to match new command line options for the aligner. The GUI was behaving badly in the 2003-10-08 release. Full alignment is now selected by default, if no LCB weight is specified one will be chosen automatically.

2003-10-08

Bug fixes:

- * Final aligner debugging completed using thousands of test cases. Numerous bugs were squashed. The aligner should be (but isn't) rock solid now. Removed file I/O from the ClustalW code, resulting in vast speedups during recursive alignment. One more bug to kill.

2003-09-13

Bug fixes:

- * Debugged the LCB extension code, hopefully eliminating strange crashes from the previous release.

2003-09-08

New Features:

- * Added LCB extension code, allows Mauve to search for additional MUM anchors off the ends of LCBs. Increased the minimum anchor size to 5 b.p. to reduce spurious anchoring.

2003-08-15

New Features:

- * Added print menu item to the File menu - Clean-ups for Linux - Mauve for Linux is ready!

2003-08-13

Bug fixes:

- * Additional bugfixes for the recursive anchoring and alignment process. Will this software ever be bug free?

2003-08-02

Bug fixes:

- * Fixed some serious bugs in the production of gapped alignments from Mauve style MUM alignments. This bug affected island predictions and backbone predictions of previous releases.

New features:

- * Implemented Multi-FastA sequence input. If a single sequence file is given Mauve assumes it is a Multi-FastA file where each sequence entry corresponds to a sequence to align.

2003-07-31

New features:

- * Implemented backbone identification -- backbone is any region of sequence conserved among all genomes under study. Mauve's definition of backbone is any region longer than x base pairs that does not contain any gaps longer than x base pairs. x is specified as the island size in the "Align Sequences..." window.

2003-07-30

Bug fixes:

- * Fixed a segfault in SML binary search

New features:

- * Added end-gap penalties to clustal alignments

2003-07-28

Bug fixes:

- * Fixed a few problems with Clustal Alignment -- more to come

2003-07-25

Bug fixes:

- * Changed the default maximum Clustal alignment size to 9000 - Fixed numerous bugs related to recursive alignment and ClustalW alignment

New features:

- * Now outputs alignments in Extended Multi-FastA (xmfa) format

2003-06-12

Bug fixes:

- * Don't write a gapped alignment unless the user requests a Full alignment

2003-05-08

Bug fixes:

- * bugfixes for displaying large (> 2 G.B.) sequences

New features:

- * new ruler label style for easier reading

2003-05-05

Bug fixes:

- * Fixed DragNDrop for align sequences window

2003-04-28

Bug fixes:

- * Fixed Align sequences window disappearance

New features:

- * Added clustalW format alignment output (command line mauveAligner only)

2003-04-26

Bug fixes:

- * Fixed printing (broken in previous release)

New features:

- * Supports drag and drop of alignments to the alignment window and of sequence files to the sequences list

2003-04-21

Bug fixes:

- * Fix mouse click accuracy problem when zoomed in
- * don't clear alignment window after alignment

New features:

- * mauveAligner finds interLCB islands
- * copy the File chooser view to new alignment windows

2003-04-04

New features:

- * Have mauveAligner output islands, tree, and alignment
- * Implement high-resolution printing hack

2003-04-03

New features:

- * Add LCB color lines to link LCBs

2003-02-21

New features:

- * Initial Release - Packaged into an installer for windows