

## 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.