MALIGN version 2.7
Parallel version 1.5

by Ward Wheeler and David Gladstein
Documentation by Daniel Janies and Ward Wheeler
Department of Invertebrates
American Museum of Natural History
Voice (212) 769-5754
Fax (212) 769-5233
wheeler@amnh.org

I. Introduction

A. The problem
B. Heuristic alignment
C. Multiple "multiple alignment" topologies
D. Parsimony as optimality criterion
E. Comparison to other methods

II. Interface

A. Stdin Stdout interface
B. Command line interface
C. Program prompting

III. Input file formats

A. Data file
    i. Nucleic acids - GenBank format
    ii. Amino acid sequences - GenBank format
    iii. Nucleic and amino acid sequences - Block format
B. Parameter file
C. Groups file

IV. Setting the parameters

A. Gap costs
B. Change cost
C. Complex costs
D. Maximum Cost

V. Setting the alignment cost function

A. Column
B. Cladogram based costs

VI. Setting the alignment construction procedure

A. Alignment procedures
    i. Pairwise
    ii. Tree-based heuristic method
B. Orders of Sequence Addition
C. Branch Swapping
D. Random Alignments

VII. Specification of alignment order

VIII. Output formats

A. ASCII

Typographical convention:

`courier =` example of commands and output

I. Introduction

The multiple alignment procedures implemented in MALIGN are close parallels to those used to search for minimum length cladograms. The logic and program commands will be clear to those familiar with programs such as Hennig86, NONA, and PAUP. Alignment topologies are constructed via different sequence addition sequences and improved through branch swapping. Each alignment topology yields a multiple alignment and cladograms are constructed from that alignment. The cost of the most parsimonious cladogram is then assigned as the multiple alignment cost.

This document describes the underlying concepts and practical usage of MALIGN, a multiple sequence alignment program. Although MALIGN can be used to process amino acid data (by conversion to nucleotide triplets), this program is designed to analyze nucleic acid sequences. Therefore, most of the discussion and examples refer to DNA / RNA data although they pertain to both sequence types.

A. The problem

The first step in any systematic or evolutionary study is to establish provisional homology statements. Sequence alignment represents this step for analysis of biological molecules.

The kernel of sequence alignment is the well-known dynamic programming algorithm of Needleman and Wunsch (1970). In this procedure, transformation and gap costs are established, and sequences are aligned via the insertion of gaps. The Needleman and Wunsch procedure employs an optimality criterion such that a maximum score represents the best alignment.

Sankoff and Cedergren (1983) have proposed the use of phylogenetic trees within an n-dimensional Needleman-Wunsch framework. Their method requires that the cladogram of relationships is known a priori and alignments of sequences are performed in the order prescribed by the phylogenetic relationships of the taxa from which the sequences were derived. If this knowledge is not at hand, the alignment would be repeated for all (or some heuristic subset) of phylogenetic relationships. Any search of this type involving more than a few taxa would take a long time.

MALIGN employs an optimality criterion of minimal cost of phylogenetic tree to choose the best alignment. These concepts are treated in detail below in the sections: D. Parsimony as optimality criterion and E. Comparison to other methods.

Minimization occurs by searching for the least costly path through a matrix determined by the sequences and costs associated with accepting nucleotide mismatches or inserting gaps (insertion-deletion events). This matrix implicitly contains all possible alignments. In the case of two sequences (lengths A and B), a matrix of AxB elements is required to lay out the possible paths. If the sequences are very long, this matrix will be very large and the computational effort required to find the solution commensurably intense. The Needleman-Wunsch algorithm was specified for two sequences and can, in principle, be extended to any number of sequences. However, the addition of sequences opens an immense computational problem. Generally, m sequences of length N bases will require N^m elements of storage. For example, If three sequences of length N are to be aligned, a cube of N^3 elements is constructed (i.e., if N = 300, 27 million elements will be contained in this cube). Only the most impoverished of data sets can be aligned in this exact fashion.

This computational problem can be circumvented by aligning the sequences in a series of pairs and then amalgamating the paired alignments by the insertion of extra gaps. This approach will yield a multiple alignment. However there are two problems. First, pairwise alignments are sensitive to the order in which sequences are added. Although the Needleman-Wunsch procedure is commutative, it is not associative e.g., ((A + B) + C) ≠ ((A + C) + B)). Different alignment orders yield different multiple alignments, hence different homology statements. Second, when this approach departs from pairwise alignments the link to a Needleman-Wunsch optimality criterion of the alignment is lost. In this situation, how does one compare alignments and define an optimal alignment?

MALIGN offers solutions to the two problems of order dependency and optimality through use of heuristic searches, multiple alignment topologies, and parsimony.

B. Heuristic alignment

Since it is intractable to construct the optimal n-dimensional multiple alignment matrix, all the algorithms discussed and implemented here are heuristic attempts at this elusive goal. In order to increase the chances of achieving a solution close to optimal, large computational effort is required. MALIGN offers commands from the quick and extremely dirty "pairwise" (`pair`) to more aggressive search strategies involving branch swapping and rerooting (e.g., `build`).

C. Multiple "multiple alignment" topologies

There may be several alignments that yield equally parsimonious cladograms. MALIGN enables the discovery of these ambiguities in alignment.

D. Parsimony as optimality criterion

Sequence alignment is a procedure by which we can recognize and describe potential homology among nucleotide (or amino acid) positions. A logical means to assess the quality of these statements is required. In MALIGN the best alignment is the one that produces the shortest phylogenetic tree. The minimum number of steps or changes required by an alignment is the length of the most parsimonious branching diagram for these sequences. Since this is the same criterion used to determine the relative merits of the cladograms derived from these sequences, it is logical to extend this criterion to the alignments themselves. Hence, the alignment which yields the most parsimonious cladogram(s) is the best (there may, of course be several such alignments).

The assignment of cladogram length to alignment cost is complicated by the necessity of gap penalties when sequences do not line up precisely. If a cost is not assigned to the insertion of gaps, a trivial (cost equal to zero) alignment will result. In the trivial alignment one sequence would have gaps at each position where the other sequence has nucleotides.

Furthermore, to create an entirely logical and consistent parsimony-based alignment, the same cost function that is used in the Needleman-Wunsch procedure must be used in the construction of cladograms from these alignments (see sample MALIGN output files appropriate for various phylogenetic software). Gaps are considered a fifth state (after A,C,G,T/U). The cost of transformation between a gap and the other states in cladogram construction is determined by the parameters of the

alignment. When phylogenetic cost commands are specified in MALIGN such a weighting is performed.

E. Comparison to other methods

Several methods have been presented to align multiple sequences. In each approach, an alignment topology is specified to allow the ordered accumulation of aligned sequences into a multiple alignment. An alignment topology is simply a heuristic to accrete sequences into a multiple alignment. The methods differ in how the topology is determined and how the topology interacts with the actual alignment.

Alignment order affects pylogenetic results (Lake, 1991). Some advocate using known phylogenies to guide alignments (Mindell, 1991) however preconceived notions of relationships will bias the analysis. Alignments could be repeated for all possible phylogenetic relationships. However, this problem is computationally intractable and hence some subset of phylogenetic relationships are typically used to guide an alignment.

There are various ways of: 1) creating alignment topology or topologies from the sequences, 2) producing multiple alignments from guide trees, 3) determining the relative quality of the alignments produced (if more than one alignment is produced).

These methods can be summarized by their implementations in software. Here methods are discussed briefly. These methods are discussed in detail in a review in preparation.

Feng and Doolittle (1987, 1990; adapted by Higgins and Sharp, 1988, 1989 Higgins et al., 1992, Thompson et al., 1994) start with pairwise alignments. A distance measure is calculated from these pairwise alignments and a Fitch- Margoliash (distance) (Fitch and Margoliash, 1967) tree determined for these distances. The phylogenetic topology yielded by the Fitch-Margoliash analysis determines the order of alignment. This process is taken one step further by Konings et al. (1987) and Hein (1989a, b, 1990).

Clustal X (Thompson, et. al., 1994) uses pairwise similarity to construct an alignment topology. This topology is used for progressive addition of groups of similar sequences into an alignment. Consensus sequences that incorporate only bases present in all sequences and preserve gaps inserted in pairwise alignments are used to store alignments at internal nodes. Only a single multiple alignment is constructed.

In TreeAlign (Hein, 1989a, b) pairwise distances are used to construct an alignment topology and an initial alignment with observed sequences at the edges of the tree. The alignment topology is converted to a parsimony tree and used to direct an alignment algorithm. During alignment, potential ancestral sequences are created at each node in the alignment topology using parsimony. Although a parsimony score is attached to the alignment topology no other trees are constructed for comparison. The alignment topology is subjected to nearest neighbor interchanges and the most parsimonious tree is used to align sequences.

II. Interface

There are three ways in which the user may communicate with MALIGN:

1) UNIX style interaction through the use of standard input and output.
2) Command line specification.
3) Program prompting for input and output files.

A. Stdin Stdout interface

The UNIX style interface is typed at the command line. Input and output files are specified through the use of the less-than (<) and greater-than (>) signs. The parameter file and optional groups file precede the input file which precedes the output, such as follows:

```
MALIGN parameter_file < input_file > output_file
```

or if a groups file is specified:

```
MALIGN parameter_file groups_file < input_file > output_file
```

When this interface is used, no additional commands can be specified other than those provided in the parameter file.

Shell scripts can be written to invoke several runs of MALIGN. Scripts are useful to explore the effects of variable parameters on the alignment of one or more datasets. The appendix includes several examples of shell scripts for MALIGN.

B. Command line interface

The command line interface operates like any command line driven program. Commands are specified when the program is invoked. Commands may be specified in any order. Commands are identical to those used in parameter files with a few exceptions. The differences (from file specifications) are in the commands 1) `input`, which specifies, strangely enough, the input file, 2) `output`, which precedes the name of the output file, 3) "groups," which specifies the groups file, and 4) `parameters`, which specifies a file from which further commands may be garnered. Usage of the command line is as follows:

```
MALIGN input input_file output output_file parameters param_file
```

C. Program prompting

The third means of interface is a series of prompts. For the UNIX and DOS machines, the name of the executable is typed ("MALIGN," or some variation) and the prompts follow. For example:

```
djanies on argus /users/djanies/crinoids 26 # malign

MALIGN version 2.7
by Ward Wheeler and David Gladstein
Department of Invertebrates
American Museum of Natural History
Phone (212) 769-5754
FAX   (212) 769-5233
e-mail wheeler@amnh.org
Copyright 1991-1995 all rights reserved

     What is the name of the sequence containing file?

     test

     What is the name of output file?

     test.out

     What is the name of the parameter file?

     param
```

III. Input file formats

A. Data file

Two input file formats are legible by MALIGN. The first is a modified, series of the sequence Genbank files. Genbank comments must be removed. The name and nucleotide data is all that is needed for each sequence. The single file contains all the sequences to be aligned. This is the default input file format.

i. Nucleic acids - GenBank format

This format is quite simple. The file contains a line with the sequence name (<=20 characters, without spaces, so "A._bii" not "A. bii"). The sequence follows in strings of ten bases, six to a line. Base numbers precede the 6X10 strings on each line. An additional carriage return follows after the last base of the sequence. All bases should be upper case, and IUPAC ambiguity codes are accepted. The number of sequences and their maximum length is limited to 32,767.

For example:

```
SequenceA
   1 CAGCAGCACG CAAATTACCC ACTCCCGGCA CGGGAGGGTA GTGACGAAAA
ATAACAATAC
  61 CCGTC
```

```
SequenceB
   1 CAGGCACGCA AATTACCCAC TCCCGGCAGA GGTAGTGACA AAAAATAACG
ATACGGGACT
  61 CCGTCAC

SequenceC
   1 GGCACGGAGG TAGTGACGAA AAATAACGAT ACGGGACTCA TCCGAGGCCC
CGTAATCGGA

SequenceD
   1 AAATTACCCA CTCCCGGCAC GGAGGTAGTG ACGAAAAATA ACGATACGGG ACTCA

SequenceE
   1 GAGGTAGTGA CGAAAAATAA CAATACAGGA CTCATATCCG AGGCCCTGTA ATT
```

Sequences whose names are followed by certain characters will be modified as follows:

`taxon_name!` reversed and complemented

`taxon_name=` reversed

`taxon_name+` complemented

`taxon_name*` presumed to be protein sequence and converted to nucleotide triplets

The modification characters (!, +, =, and *) are removed from sequence names, hence will not be included in output and should not be included in any groups file.

ii. Amino acid sequences - GenBank format

This format is identical to that of nucleotide sequences with the exception that instead of A's, C's, G's, and T's, single letter IUPAC amino acid codes are used. For example:

```
SequenceF
      1 ILAVEELVI SLIVES

SequenceG*
      1 AAYVTTTCC KKYK
```

The amino acid codes are then converted to triplet representation using IUPAC nucleotide ambiguity codes. This is accomplished automatically when the sequence contains an element that is not a nucleotide code. Since there is great overlap between the nucleotide and amino acid codes, an asterisk may be placed at the end of a sequence name to designate it as protein and force the conversion (this asterisk is used only in the data file - not a groups file). This asterisk is not included when the alignments are output.

The conversion used is the universal code with IUPAC

Amino Acid	Single Letter Code	Nucleotide Representation
Alanine	A	GCN
Arginine	R	MGN
Asparagine	N	AAY
Aspartic Acid	D	GAY
Cysteine	C	TGY
Glutamine	Q	CAR
Glutamic Acid	E	GAR
Glycine	G	GGN
Histidine	H	CAY
Isoleucine	I	AAH
Leucine	L	CTN
Lysine	K	AAR
Methionine	M	ATG
Phenylalanine	F	TTY
Proline	P	CCN
Serine	S	TCN
Threonine	T	ACN
Tryptophan	W	TGG
Tyrosine	Y	TAY
Valine	V	GTN

Since this scheme is anything but universal, because mitochondrial sequences don't use the same codes. Codes may be reassigned through the newcodes command. For example:

```
newcodes M ATR
newcodes 0 NNN
```

These lines reassign new triplets based on user definitions. The first line changes the representation of methionine to the codon ATR (ATA and ATG) wherever it occurs, as in some mitochondrial codes.

Since there may be protein sequences with several different codes (yeast tryptophan = TGR) in a single data set or amino acid ambiguities it is possible to include the digits 0-9 in protein sequences. These digits are also assigned triplet representations through newcodes. In order for MALIGN to read the following data:

```
SequenceF  1  AGVT000123 99IIJMXV3
```

The user would have to assign codes to the digits 0 1 2 3 9 via the newcodes command.

iii. Nucleic and amino acid sequences - Block format

The second input file format is more like the representations of aligned sequences where the sequence names are displayed to the left of some number of bases (usually 50-100) and repeated for all sequences in the block. This input format

requires specifying the command `inalign` since the input sequences may have already been aligned.

The same coding and recoding features are present for this input format, only the presentation of the sequences differs. As opposed to the Genbank format where each sequence is present in its entirety before the nucleotides of the next sequence, the Block format interleaves the sequences. For example:

```
3 2
SequenceA ACCTC--GAC
SequenceB ACGTCAAGA-
SequenceC ACGTCAAAAT

SequenceA GGTCTCT
SequenceB --TCTCT
SequenceC GGCCCCT
```

The first line of the file contains two integers (e.g., 3 and 2). The first is the number of sequences and the second the number of lines per sequence. Gaps may be included in the sequences, but are stripped out in preparation for alignment. This format is similar to the sequence input format used by PAUP and many other programs. It is identical to the `interleave` output format command that can be specified for MALIGN.

B. Parameter file

The parameters for MALIGN may be presented to the program from a file. They can be upper or lower case or any combination. Many commands can be abbreviated to a minimum number of characters as defined below in the commands summary (section XVII). An example of a parameter file is provided below:

```
internal 3
leading 2
trailing 2
changecost 2
interleave
hen86
hengap
keeptrees 100
keepaligns 100
build
aspr
score 2
spr
newcodes X YYY
newcodes 0 AGT
linelength 60
time
phylotime
matrix 0 2 1 2 2 0 2 1 1 2 0 2 2 1 2 0
coding 2
```

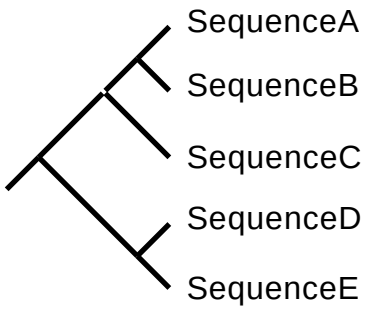
If the command requires an argument it follows immediately after a space and must be an integer. Carriage returns are optional in a file (but of course forbidden in any command line activity).

C. Groups file

If the user wishes to specify an alignment order, such as a "known" phylogeny, a groups file is required. The hierarchy of sequences is specified by the sequence names in standard nested parenthesis notation. For example the following groups file:

```
(((SequenceA SequenceB),SequenceC)(SequenceD SequenceE))
```

Specifies the alignment topology:



In the groups file, sequence names may be separated by spaces, commas, or confusing combinations of both. The name of the groups file may follow the parameter file in a UNIX-like interface, after the command `group filename` in a command line, or when prompted for example:

```
Will you specify alignment groups (y/n)?
y
What is the name of this groups file?
filename
```

MALIGN can use a groups file to determine the order of alignment completely. In this case, the user makes the assumption that he has a priori knowledge of alignment order such as that suggested by a partially resolved hypothesis of relationship (e.g., a bird might be aligned with a crocodile before either is joined to a snake). Another assumption might be that it is appropriate to first align together certain functional class of sequences (such as α-type globins) in a molecular evolutionary analysis across different loci. In both cases, the groups file implies a directed branching diagram for the alignment order which may or may not be reasonable to employ.

IV. Setting the parameters

A. Gap costs

The user may specify several gap costs: those for internal, leading, and trailing gaps as well as those for extra (length >1) gaps and those which potentially maintain reading frames.

The leading and trailing gap costs are separate commands to allow for sequences of unequal lengths. These would usually be less than the internal gaps (between the first and last bases). If these values are set too low (relative to the internal gap and change costs), a trivial alignment will result with sequences lined up with long strings of gaps. The default setting is one less than the internal gap cost. The maximum gap cost is 32767 (Alignment costs are represented internally by a 15 bit integer. This limits most of the numerical parameters from 0 to 2^{15}-1). A reasonable starting value might be 3 for internal gaps and 2 for the leading and trailing.

The default gap costs in MALIGN are based on independent change at each position. Hence, a gap of five bases is five times as costly as one which is at a single position. This can be modified in two ways. If you desire more complex gaps use extragpaps. A separate cost for gaps which follow existing gaps can be specified with the command `extragap n` to modify the cost of long (>1) gaps.

Costs may be set that result in all gaps being equally costly no matter what their length by specifying the command `extragaps 0.`

If extragaps is unspecified then all gaps are specified by the command `internal.`

Alternatively, additional gaps can be more costly than the initial insertion, by specifying the command:

`internal 1 extragaps 2`. This means that the opening gap costs 1 and extragaps cost 2.

`internal 2 extragaps 1.` This makes the extragaps cheaper than internals.

A facility exists to allow the determination of gap costs that may result in the maintenance of coding regions. Gaps whose lengths are multiples of three can be set to cost less than those which are not through the command `coding n.` In this case n is the cost of the ultimate gap in the multiple of length 3, 6, 9 etc... The internal gap cost is invoked for gaps which might break up a reading frame.

One cannot specify both coding and extragaps. Also, coding and extragap commands are ignored for leading and trailing gaps which are specified by the commands `leading n` and `trailing n.`

B. Change cost

The change cost specifies the cost of a nucleotide change as opposed to a gap. The maximum change cost is 32767.

C. Complex costs

Complex character transformation models may be specified via the `matrix` command. The sixteen entries of the matrix (ACGT x ACGT) specify the costs of transformations between each of the four nucleotide bases.

```
A->A  A->C  A->G  A->T
C->A  C->C  C->G  C->T
G->A  G->C  G->G  G->T
T->A  T->C  T->G  T->T
```

These costs are applied in both the alignment and cladogram cost stages of the analysis. This matrix would specify a transversion: transition cost ratio of 5:1.

```
matrix
0 5 1 5
5 0 5 1
1 5 0 5
5 1 5 0
```

the matrix must not include carriage returns if is part of a command line. In the command line a space is required between each cost, for example the command:

```
matrix 0 5 1 5 5 0 5 1 1 5 0 5 5 1 5 0
```

is identical to the one in the preceding example.

These types of characters, first proposed by Sankoff and Rousseau (1975; more explicitly directed towards sequences in Sankoff and Cedergren, 1983) and implemented as by Swofford in PAUP, permit the user to specify more precise cost functions than a simple gap/change model would allow. Matrices must be both symmetric (aij = aji) and metric (must obey the triangle inequality).

D. Maximum Cost

The maximum alignment cost is 2^{31}-1, thus gap and change costs should be chosen with this in mind.

V. Setting the alignment cost function

A. Column

Command `score 0` is the quickest alignment cost. This cost is simply a tally over all positions of the cost per aligned position. The cost of a column (aligned position) is calculated by subtracting one from the number of states multiplied by the change cost. The gap cost is then added to this if one or several of the sequences show a gap. This is the minimum amount of change (and cost) which can be exhibited at this position. Obviously, simultaneous minimization of all positions is extremely unlikely to be possible. This problem is ameliorated by the commands that follow.

B. Cladogram based costs

Heuristic cladogram lengths -- Commands `score 1` and `score 2` find a heuristic solution to the problem of finding the shortest tree derivable from the alignment presented. This is done in two ways: 1) a single pass finding a single cladogram, `score 1,` or 2) a more extensive search, `score 2,` which may find multiple (up to `keeptrees` or all with `maxtrees`) equally parsimonious cladograms. Multiple tree searches are generally better able to find shorter arrangements.

Branch swapping may be performed during cladogram construction with the commands `spr` and `tbr`. Although these commands are time consuming, they often yields shorter cladograms. The command `treerandorders` reorders taxa during heuristic cladogram construction to avoid local minima.

Heuristic searches may be improved by randomly reordering the taxa. In this way local optima may be avoided. The command `treerandorders n` will perform this reordering and re-searching n times.

VI. Setting the alignment construction procedure

Since MALIGN accretes sequences in a stepwise manner, a method must be specified to determine the order of alignment. Four commands (`pair, quick, build, and randalignment`) are available to choose orders and branch swapping analogous to that of cladogram construction may be performed. Any and all of these may be performed sequentially to yield the best set of alignments possible.

A. Alignment procedures

i. Pairwise

The command `pair` is most rapid of the multiple alignment commands. This procedure begins with all pairwise alignments. The least costly pair (determined by the `score n` command) is then aligned with all the remaining sequences. The least costly of these (and the unused original pairs) is aligned with the remaining and so on. This method performs n^2-2n pairwise alignments to generate a single multiple alignment. This procedure is similar to that of UPGMA and as such shares its benefits (speed) and its shortcomings (unlikely to be very good).

ii. Tree-based heuristic method

The command `build` provide more complete, but more laborious search procedure based on heuristic tree searches. In this method, alignments are constructed by adding taxa and trying each of the possible addition points and alignment orders available for those sequences much in the way heuristic cladogram construction algorithms work. The ith sequence is aligned in each of the 2i-3 orders (the alignment topology is affected by the placement of the root). The best (and all equally costly) alignments are kept up to the value set by `keepaligns` or all with `maxaligns`. The command `quick` works the same way as build but limits the consideration to a single alignment topology at any stage. The command `reorder` will perform pairwise alignments such that more similar sequences are added first.

(see the next section for the ramifications of this and other commands). The command `reorder` can be used with either `quick` or `build`.

B. Orders of Sequence Addition

The order in which sequences are added into the alignment search can affect the end solution. This is true for the tree based heuristic searches only. The results of pair searches are unchanged. In order to address this problem, several means of adding sequences are available.

Sequences may be added in the order in which they appear in the input data file. This is quick and easy. The sequences may also be added in order of decreasing similarity (cost of pairwise alignment, most similar sequences first) from most to least similar sequences. The `pair` command is simple but has two shortcomings. In the first place, (n2-n)/2 pairwise alignments are performed solely for the purpose of ordering the input data. This is computationally costly. The second problem is in the potential for getting stuck in local optima. Branch swapping may not solve this problem. Any single addition order can lead to a result which is not globally optimal. This is one of the most severe problems of non-exact solutions.

Random addition orders, `randorders` n can address both of these shortcomings. The randomization is rapid and when a number (n) of orders are tried, and realigned local optima may be escaped. This is related to the problem of islands of trees shown by Ramsköld and Werdelin (1991) and Maddison (1991). There are too many addition sequences to explore them all. Random addition orders is a means of expanding the search space but it still a small fraction of the total search space. The shortest of many random addition orders alignments is often shorter than those constructed by methods based on pairwise similarity.

C. Branch Swapping -- commands aspr arrt atbr.

The swapping options will attempt to improve multiple alignment by finding alignment topologies that yield shorter cladograms. Subtree pruning and regrafting (command `aspr`) branch swap on alignment topology. Reroot alignment topology (command `arrt`) to search for better alignment topologies. Tree bisection and reconnection (command `atbr`) branch swap on alignment topology.

D. Random Alignments

The command `randaligns` n will generate n random alignment topologies, keeping the single or multiple best. The results can be subjected to swapping.

VII. Specification of alignment order

With the inclusion of a groups file (section III C), partial or complete alignment orders may be specified. The various alignment commands (with the exception of the `pair` command) will build a multiple alignment subject to the conditions specified in the file. This will greatly speed up the `quick, build,` and `exact` searches by removing whole families of alignments from consideration. The random alignments `randaligns` command, however, is actually slowed down. Since not all random alignment orders will match the groups, many more are generated until an acceptable order is found.

Using this method alignments can be performed in the order specified by expected phylogenetic relationships among the sequences. This can have two less desirable properties. First, the alignments generated are almost always less parsimonious (or generate cladograms which are) than those that are constructed without these groups specifications. The only situation in which this would not occur would be if the user had stumbled on a better solution than the heuristics had. A second problem is that by requiring certain sequence groups, searches can result in grossly, and obviously local alignments.

The use of the groups file can be restricted to the addition segment of the alignment search with the command `start`. This command causes MALIGN to use the groups file to create the initial multiple alignment(s) and then abandon the constraints during branch swapping. `Start` can be useful in specifying starting points for alignment or in continuing analyses performed without swapping.

VIII. Output formats

There are several output formats any or all of which can be specified. In each case the order of the output taxa is by default that of the alignment itself. Taxa in the alignment can be output in the same order as the input file by specifying the command `outorder input`.

A. ASCII

If the command `ascii` is specified, aligned sequences are output as ASCII strings on a single line. The name of the sequence and the sequence itself are separate lines.

B. Interleaved

When `interleaved` is specified, sequences are presented in batches of `linelength` bases with the sequences interdigitated to facilitate comparison. This corresponds to the data part of the NEXUS format `interleaved`. The top of the file contains the number of sequences and number of lines per sequence.

C. Dot

If the command `dot` is specified, output is identical to that of `interleaved` with the exception that the alignment is output such that all bases that match that of the first taxon in each column of the matrix are replaced with a dot. Only base mismatches, gaps, and missing data are shown with characters.

D. Hen86

If the command hen86 is specified, bases are recoded for input to Farris' Hennig86. A becomes `0`, C becomes `1`, G becomes `2`, T/U becomes 3, and gaps become ? (Sequence ambiguities are recoded as ?). The command `nona` truncates taxon names in the Hennig86 file to 10 characters.

E. Hengap

The command hengap is Identical to the command hen86 except that new characters are created for the presence (`1`) and absence (`0`) of gaps. Leading and trailing gaps remain missing (?). A line is included in the file to tell Hennig86 to make the characters nonadditive (`cc-.`) and apply the same weights to the change and gap characters used in the alignment procedure. Phylogenetic analysis of the aligned

sequences in only consistent when these weights are applied. Unless the nogap command is specified, MALIGN employs these relative weights uniformly over both sequence alignment and cladogram construction.

F. Nogap

Instead of using gap weights and characters in addition to change weights, nogap specifies that gaps be treated as missing data in phylogenetic reconstruction specified by the `score` command (see section V). nogap introduces inconsistencies into the analysis in that the cost function is treating gaps one way in cladogram construction and another in alignment. As a result, the `exact` command performs an exhaustive search (as opposed to branch and bound) when nogap is specified.

G. Paup

Output specified by the command paup is identical to `interleaved` but includes the commands for DNA characters in the data block of the NEXUS file (with U's converted to T's).

H. Pdot

The command `pdot` specifies output format in NEXUS file. The output is identical to that specified by paup except that bases that match that of the first taxon are replaced by a dot (i.e., "match first" command in NEXUS).

I. Linesize

`Linesize` n determines the number of aligned positions to be displayed per line for the output format commands are hen86, `interleaved`, `dot`, and hengap.

XII. Diagnosing predetermined alignments

It is possible to input to MALIGN a preexisting alignment such as the one below and determine its cost. This can be done through the use of the commands `inalign` and `costonly`. The command, `inalign`, specifies that the input sequences are already aligned and in the `interleaved` format**.** This is one way to diagnose the efficacy of hand job alignments.

```
3 2
Sequence1 ACCTC--GAC
Sequence2 ACGTCAAGA-
Sequence3 ACGTCAAAAT

Sequence1 GGTCTCT
Sequence2 --TCTCT
Sequence3 GGCCCCT
```

The `costonly` command tells the program to determine the cost of the alignment by the means specified in the parameter file or by the commands specified from the command line. As with all other commands, `inalign` and `costonly` may be included in a parameter file, the command line or both.

The utility of these commands comes in two areas. The first is the comparison of received alignments to those generated by MALIGN. These alignments may be more or less costly than those generated by the program. A second utility is in the hand modification of MALIGN generated alignments. Alignments may be edited by

hand (as with a word processor) to produce new alignments which may appear more sensible than those produced automatically. The `costonly` command allows the assessment of relative optimality of the hand-modified to heuristic alignments in the same way that manually swapped cladograms can be compared to those generated by parsimony programs for goodness-of-fit. Hand job alignments may be less costly than those produced by MALIGN, but MALIGN is agnostic as to the relationships of the sequences, the hand of the editor is rarely so impartial. A single better alignment implies that there are other superior arrangements which may be radically different in phylogenetic implication, caveat editor.

XIII. Exploring alternate alignment paths

When an alignment is performed, each cell of the alignment matrix is linked to a cell above and/or its left. By backtracking through these connections, the alignment is accomplished along a path through the matrix, usually about the diagonal. In a pairwise alignment, each cell may be connected to one of three other cells. These are the cells above, to the left, and diagonally above and left of the current cell. The optimal connection is made by choosing among these cells on the basis of cost. The lowest cost connection is made and becomes part of the alignment path. The connection from the cells above and to the right imply the insertion of gaps in one or the other sequence and have an associated cost. The diagonal path implies a sequence match or mismatch with its concurrent benefit or cost.

It is possible for two or more of the connections to offer equally costly alignment paths. This can occur where the insertion of a gap or sequence mismatch are equally costly, or in situations with extremely gappy alignments. These equally costly paths yield different alignments which may have different phylogenetic repercussions. The multiple paths are a source of sequence alignment ambiguity which can be explored with MALIGN.

When equally costly connections present themselves, MALIGN will choose the diagonal (match or mismatch) path by default. This behavior can be modulated with a series of commands specify alternate means of choice. Gaps can be favored by three commands, `prefright, prefdown,` and `contig`. The right and down preferences specify that all other things being equal they will prefer a gap in one or the other sequence. The command `contig` will prefer to insert a gap over a match or mismatch when a gap already exists in one of the two sequences (or alignments). This tends to create alignments with fewer gap locations but contiguous gaps made or more positions. The commands `prefright` and `prefdown` also produce fewer, longer gaps but in a more arbitrary manner. The diagonal (default) preference can be respecified in a series of commands by `prefdiagonal`.

Although in a pairwise context these alternate paths are equally costly, when multiplied down the heuristic alignment topology the grand alignment may not be equally costly to that produce under default conditions. `contig` may appear to yield the most parsimonious and (visually pleasing) alignments, this is a crude generalization and counter examples exist.

In addition to these gap-preferring deterministic commands, it is possible to make random choices. The command `prefrandom` will make a random (or more properly psuedorandom) choice among equally costly alignment paths. A final command, `discontig`, is basically the opposite of `contig. Discontig` will favor gaps only if the aligned sequences do not already have a gap. This will break up

insertion/deletion stretches into multiple gaps. It doubtful if any one would really want to do this, but its there for symmetry if nothing else.

Additional commands `prefmatch`, (equal to `prefdiagonal`) `prefshorter`, and `preflonger` may also be specified. `prefshorter` will favor the insertion of gaps in shorter sequences while `preflonger` the opposite.

When these commands are used with a groups file (section VII), alternate alignments based on the same alignment order can be generated. This can help the investigator to examine the robust and sensitive segments of an alignment.

XIV. Reporting options

`noerror` Suppresses non-fatal error messages.

`taciturn` Suppresses the display of progress information.

`silent` Suppresses both error and progress reporting.

`printintermediate` If more than one alignment procedure is specified (such as `build` and `exact`), `printintermediate` causes MALIGN to print intermediate results.

XV. Miscellaneous yet important topics

A. Memory considerations

As mentioned above, pairwise alignment requires the creation of a matrix of NxM elements, where N and M are the lengths of the two sequences to be aligned. MALIGN uses four bytes to store each of these elements. Hence, the total memory requirement is approximately eight times the square of the longest sequence plus the size of the executable file.

If the memory capacity of the computer is exceeded, the error message "....assertion failure" will appear and cause the program to exit. The `keeptrees` and `keepalignments` commands can be lowered to allow longer sequences to be analyzed.

The command `lowmem` can be used to minimally allocate memory. This may alleviate some memory problems (but can create others - don't use it if it misbehaves). This command has a computational overhead of roughly 5%. `showmem` can be used to report the size of memory allocation attempts. Since this is reported before allocation, the amount of memory required to avoid failure can be determined.

B. Text control

If input data, parameter, and/or groups files are created or modified with word processors, it is crucial to make sure that the resulting files are saved as text only. Some word processing programs, regardless of platform, may insert illegal, misplaced, and sometimes invisible characters that do not conform to formats required by MALIGN. Means of controlling text are to use text editors suited to UNIX compilers and/or word processors that show all characters (e.g., "show paragraph" command in MSWord or "show codes" command in WordPerfect). Often it is useful to use a combination of programs to root out formatting errors.

XVII. Command summary:

Conventions:

`courier = command`

`courier bold` is the necessary and sufficient portion of the command, any portion of the rest of the command is optional, for example when using **`change`**`cost, only change` must be specified but `changecost will work.`

`courier italics bold` = argument required buy certain commands

`alignfile` ***`filename`*** A file which contains only `infile` commands. This is to simplify alignment of alignments or use diverse weighting schemes for various regions See also `infile`.

`arrt` Reroot alignment topology to search for better alignment topologies.

`ascii` Output in block style format.

`aspr` SPR (subtree pruning and regrafting) branch swap on alignment topology.

`atbr` TPR (tree bisection and reconnection) branch swap on alignment topology.

`build` Wagner type heuristic for multiple topologies (see also quick).

`change``cost` **`n`** Assigns cost n of nucleotide substitution.

`clados` Output in Hennig86/Clados format.

`coding n` Assigns cost n to third gap in stretches of gaps divisible by three.

contig If the cost is the same in the context of the Needleman-Wunch procedure, contig favors gap insertion to base mismatch. contig tends to insert gaps in groups rather than scattered single gaps.

costonly Measures cost of an alignment provided by the user.

discontig If the cost is the same in the context of the Needleman-Wunch, procedure, discon favors base mismatch over gap insertion. Discontig tends to insert scattered single gaps rather than groups of gaps.

dot Outputs alignment such that all bases that match that of the first taxon in each column of the matrix are replaced with a dot. Only base mismatches, gaps, and missing data are shown with characters.

expand Converts gaps to X (e.g., adjusts X_ to XX, or X_X to XXX).

extragap **n** Cost n of second gap in a series of gaps. Important to keep reading frame of coding sequences.

ftbr Do single tree tbr before multiple tree tbr.

grainmax Parallel processing command, set grain size to maximum.

grainmin Parallel processing command, set grain size to minimum.

grains n Parallel processing command, set grain size to number n.

group Set group file. A group file specifies alignment order defined by a series of nested sets of sequences in standard parenthesis notation with or without commas e.g., (((SequenceA SequenceB),SequenceC) (SequenceD SequenceE)). The group file is used in the command line after the parameter file e.g., MALIGN parameter_file groups_file < input_file > output_file

hen86 output format Hennig86/NONA/CLADOS

hengap Output format is Hennig86/NONA/CLADOS with gaps coded as a matrix of characters and a separate matrix of gaps. Identical to hen86 except that new characters are created for the presence ("1") and absence ("0") of gaps. Leading and trailing gaps remain missing ("?"). A line is included in the file to tell Hennig86 to make the characters nonadditive (cc-.) and apply the same weights to the change and gap characters used in the alignment procedure. Subsequent analysis of the aligned sequences is methodologically consistent when these weights are applied, for example, in phylogenetic algorithms. Unless the nogap command is specified, MALIGN employs these relative weights uniformly over both sequence alignment and cladogram construction.

inalign Specifies that data is being input in interleaved or prealigned format. Sequences are presented in batches of bases whose length is specified by the command `linelength.` The sequences are interdigitated to facilitate comparison. This corresponds to the data part of the NEXUS format "interleaved." The top of the file must contain two integers separated by a space. The first integer represents the number of sequences and the second integer represents the number of lines per sequence.

infile *type weight filename* A command, often included in a file called by `alignfile`, specifying: type of data, weight to be applied to that set of sequences, filename containing the data, which causes the program to input all the aligned sequences in the file as a single object. This object is then aligned to the contents of other `infile` commands and arguments containing single or several sequences via the normal heuristic commands in MALIGN. Costs correspond to the product of weight specified alignment costs for that particular set of sequences in cladogram reconstruction (i.e., [weight * (gap+ change costs)].

For example a command line may include the following command:

```
alignfile filename
```

this command will call the file named filename who's contents are below:

```
infile a 1 sequencesX
infile n 1 sequencesY
infile m 1 teeth
```

`infile a 1 sequencesX` specifies that this set of sequences should be aligned and weighted 1.

`infile n 1 sequencesY` specifies that this set of sequences should not be aligned but should be weighted 1.

`infile m 1 teeth` specifies that this file is a morphological data set and should be weighted 1.

`infile` ***`m`*** m specifies a morphology file. See also `morph`.

`infile a` a specifies an alignment or set of interleaved sequences to be aligned as a single object.

`infile` ***`n`*** specifies that these

`interl``eaved` Output format in which aligned sequences output "linesize" at a time.

`intern``al` ***`n`*** Internal n specifies the cost of an internal gap insertion (0-32767).

`keepa``ligns` ***`n`*** Specifies maximum number,. n, of equally costly alignments to keep under `build` (0-32767).

`keept``rees` ***`n`*** Keeptrees, specifies maximum number, n, of equally costly cladograms to keep and subject to branch swapping under `score 2` if specified (0-32767).

`leadi``ng` ***`n`*** Leading, cost n of an insertion of a leading gap (0-32767).

`linel``ength` ***`n`*** Specifies linelength n for interleave format (0-32767).

`low``mem` Decreases memory usage at the cost of execution. Uses `iter` but minimally allocates memory. This can save a huge amount of memory (at least a factor of ten) but carries a 5% overhead in computational effort.

matrix ***n n n n n n n n n n n n n n n n*** Sankoff (step) Matrix. Sixteen arguments specify complex character transformation weights in 4 of 4 rows ACGT x ACGT matrix. The matrix must be symmetrical. For example:

if TV = 2
TS = 1

and the stepmatrix is:

```
  a c g t
a 0 2 1 2
c 2 0 2 1
g 1 2 0 2
t 2 1 2 0
```

The appropriate MALIGN command line is:

```
matr 0 2 1 2 2 0 2 1 1 2 0 2 2 1 2 0
```

Note: A matrix of 5x5 to incorporate gap costs is used in NEXUS files For example:

usertype clustal stepmatrix = 5

```
   A C G T -
[A] 2 1 2 4
[C] 2 2 1 4
[G] 1 2 2 4
[T] 2 1 2 4
[-] 4 4 4 4
```

However gaps are specified by the commands `internal, leading, and trailing` in MALIGN.

maxaligns Dynamically reallocates alignment buffer to hold all discovered alignments (within memory limits).

morph Specifies that input file is morphological matrix. See `infile m`.

newccodes *nnn X* Specifies a new amino acid triplet code nnn for single letter amino acid code X.

noalign Specifies that these sequences are prealigned and are not to be aligned. See `infile n.` The alignment will be accepted and the specified costs of gaps and changes will be applied.

noerr Suppresses the reporting of errors to the screen.

nogap Treat gaps as missing data under phylogenetic alignment cost command. (By default gaps are included as characters as with hengap output).

nolead Cost of leading or trailing gaps are not considered in costs of alignment.

nolow Turns off `lowmem` command. Considers only a narrow band surrounding the diagonal path in the NW matrix.

nona Output format in which taxon names are limited to 10 characters.

noquick Turns off quick.

outorder Taxa in presentation of results are in the same order as in the input file (this command overrides the default which is taxa are presented in the order in which they were aligned).

pair Creates a single multiple alignment by joining the best pair of sequences or previous alignments.

param Command line command that takes commands from a parameter file.

paup Specifies output format in NEXUS file.

pdot Specifies Output format in NEXUS file in which bases that match that of the first taxon are replaced by a dot (i.e., match first command).

phylotime Reports time spent reconstructing phylogenetic trees (not total time of alignment).

prefdiagonal Specifies preference for diagonal path in NW matrix (i.e., prefers match/mismatch over gap when equally costly alignment paths are possible; this is the default).

prefdown Specifies preference for downward path in NW matrix (i.e., prefers gap over match/mismatch when equally costly alignment paths are possible).

preflonger Specifies preference to insert a gap in the longer sequence in NW matrix when equally costly alignment paths are possible.

prefrandom Specifies no preference when equally costly alignment paths are possible. Gap or match/mismatch will be chosen at random.

prefshorter Specifies preference to insert a gap in the shorter sequence in NW matrix when equally costly alignment paths are possible.

printintermediate Prints out intermediate results when several alignment commands are specified (e.g., random addition sequences; see `randorder`).

quick Wagner type build for alignments limiting consideration to single trees holds through swapping.

randalign *n* Performs n alignments with random alignment topologies.

randorder*n* Randomly reorders sequences n times and repeats alignment each time to avoid local minima.

randtree *n* Generates random trees in tree search; see also `treerandorders` for tree reconstruction commands

reorder Orders addition sequence for heursitic alignment search based on pairwise similarity such that more similar sequences are added first.

report Writes alignment parameter information to output file.

score *n* Specifies the cost regime to be applied to multiple alignment. Many users will be familiar with the old score arguments thus these are presented below (even though the old score arguments don't work anymore).

new argument		old arguments
0	Uses non-topological, column score.	0
1	Cladogram search keeping only a single tree.	3,7
2	Search on multiple trees.	4,8

showmem Displays memory in the number of bytes required.

silent No reporting whatsoever.

spr SPR branch swapping on cladogram search.

start Uses group file for initial alignment construction. Then abandons the initial alignment tree and performs unconstrained branch swapping.

tacit Scant reporting via suppression of MALIGN progress information.

tbr TBR branch swapping on cladogram search.

time Displays execution time in seconds.

trail Cost of a trailing gap insertion (0-32767).

treerandorders Reorders taxa during heuristic cladogram construction to avoid local minima.

worst Orders addition sequence for heursitic alignment search based on pairwise similarity such that least similar sequences are added first.

Acknowledgments

We would like to acknowledge the assistance of Rob DeSalle, John Gatesy, Ranhy Bang, Cheryl Hayashi, Paul Vrana, Michelle Barcia, Jeffrey Groth, Charles Ray,

Alfreid Vogler, Michael Whiting, James Carpenter, Norman Platnick, Michael Novacek, Norman Carlin, Elise Broach, David Swofford, Zoe Wheeler, and Harrison Wheeler in the preparation of this program.

XXIII. References

Carpenter, J. M. 1988. Choosing among equally parsimonious cladograms. Cladistics 4:291-296.

Farris, S. J. 1969. A successive approximations approach to character weighting. Syst. Zool. 18:374-385.

Farris, J. S. 1981. Distance data in phylogenetic analysis. in V. A. Funk and D. R. Brooks, eds. Advances in Cladistics: Proceedings of the First Meeting of the Willi Hennig Society. New York: New York Botanical Garden. pp. 3-22.

Farris, J. S. 1985. Distance data revisited. Cladistics 1:67-85.

Farris, J. S. 1988. Hennig86 version 1.5, Program and documentation. Port Jervis, NY.

Feng, D. and R. F. Doolittle. 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. Evol. 25:351-360.

Feng, D. and R. F. Doolittle. 1990. Progressive alignment and phylogenetic tree construction of protein sequences. in R. F. Doolittle ed. Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences. Meth. Enzomol. 183:375-387.

Fitch, W. M. and E. Margoliash. 1967. The construction of phylogenetic trees. Science 155:279- 284.

Gatesy, J., R. DeSalle, and W. Wheeler. 1993. Alignment-ambiguous nucleotide sites and the exclusion of systematic data. Mol. Phyl. Evol., 2:152-157.

Goloboff, P. 1993. Estimating character weights during pre-search. Cladistics, in press.

Hein, J. 1989a. A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when a phylogeny is given. Mol. Biol. Evol. 6:649-668.

Hein, 1989b. A tree reconstruction method that is economical in the number of pairwise comparisons used. Mol. Biol. Evol. 6:669-684.

Hein, J. 1990. Unified approach to alignment and phylogenies. in R. F. Doolittle ed. Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences. Meth. Enzomol. 183:626-644.

Hendy, M. D. and D. Penny. 1982. Branch and bound algorithms to determine minimal evolutionary trees. Math. Biosci. 59:277-290.

Higgins, D. G. and P. M. Sharp. 1988. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. Gene 73:237-244.

Higgins, D. G. and P. M. Sharp. 1989. Fast and sensitive multiple sequence alignments on a microcomputer. CABIOS 5:151-153.

Higgins, D.G., Bleasby, A.J. and Fuchs, R. 1992. CLUSTAL V: improved software for multiple sequence alignment. Computer Applications in the Biosciences (CABIOS), 8(2):189-191.

Konings, D. A. M., P. Hogeweg, and B. Hesper. 1987. Evolution of the primary and secondary structures of the E1a mRNAs of the adenovirus. Mol. Biol. Evol. 4:300-314.

Lake, J. 1991. The order of sequence alignment can bias the selection of tree topology. Mol. Biol. Evol. 8:378-385.

Maddison, D. R. 1991. The discovery and importance of multiple islands of most-parsimonious trees. Sys. Zool. 40:315-328.

Mindell, D. 1991. Aligning DNA sequences: homology and phylogenetic weighting. in M. J. Miyamoto and J. Cracraft, eds. Phylogenetic Analysis of DNA Sequences. Oxford University Press, New York. pp. 73-89.

Needleman, S. B., and C. D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48:443-453.

Patterson, C. 1982. Morphological characters and homology. pp 21-74. In K. A. Joysey and A. E. Friday (eds.) Problems of Phylogenetic Reconstruction. London: Academic Press.

Ramsköld, L., and L. Werdelin. 1991. The phylogeny and evolution of some phacopid trilobites. Cladistics 7:29-74.

Sankoff, D. D., and P. Rousseau. 1975. Locating the vertices of a Steiner tree in arbitrary space. Math. Prog. 9:240-246.

Sankoff, D. D., and R. J. Cedergren. 1983. Simultaneous comparison of three or more sequences related by a tree. in D. Sankoff and J. B. Kruskal eds. Time Warps, String Edits, and Macromolecules: the Theory and Practise of Sequence Comparison. Addison-Wesley, Reading, MA. pp. 253-264.

Swofford, D. L. 1981. On the utility of the distance Wagner procedure. in V. A. Funk and D. R. Brooks, eds. Advances in Cladistics: Proceedings of the First Meeting of the Willi Hennig Society. New York: New York Botanical Garden. pp. 25-43.

Swofford, D. L. 1993. PAUP version 3.1, Program and documentation. Champaign, IL.

Thompson, J.D., Higgins, D.G. and Gibson, T.J. 1994. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. Nucleic Acids Research, 22(22):4673-4680.

Wheeler, W. C. 1990. Combinatorial weights in phylogenetic analysis: a statistical parsimony procedure. Cladistics 6:269-278.

Wheeler, W. C. and D. S. Gladstein. 1994. MALIGN: a multiple sequence alignment program. J. Hered.

Wheeler, W. C., J. Gatesy, and R. DeSalle. 1995. Elision: a method for accommodating multiple molecular sequence alignments with alignment-ambiguous sites. Mol. Phyl. Evol. 4:1-9.