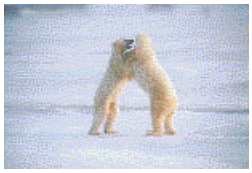
GENECONV Molecular Biology Computer Program



GENECONV: Statistical Tests for Detecting Gene Conversion -- Version 1.81

(Maintained and distributed by Stanley Sawyer --- see the last page.)

Table of Contents

1. INTRODUCTION AND OVERVIEW

Gene conversion is any process that causes a segment of DNA to be copied onto another segment of DNA, or else appears to act in this way. The target segment can be on the same chromosome, on a different chromosome, or in a different organism. Short-segment gene conversion is an important force in evolution, and often takes place at a higher frequency than does point mutation. (Lehrman et al. 1987; Gyllensten et al. 1991; Hilliker et al. 1991, 1994; Guttman and Dykhuizen 1994; see the References.)

Sometimes an ancestral gene conversion event is obvious as a long shared segment between two sequences. In other cases, statistical or graphical methods can be used to detect events that are not as obvious, or to show a pattern of apparent gene conversion in the absence of evidence for specific instances. Procedures for detecting possible events have been based on runs tests (Stephens 1985; Sawyer 1989; Takahata 1994; Sneath 1998), on the detection of changes in local estimated phylogenies (Hudson and Kaplan 1985; DuBose et al. 1988; Hein 1990, 1993; Gyllensten et al. 1991; Maynard Smith 1992; Guttman and Dykhuizen 1994; Grassly and Holmes 1997; McGuire et al. 1997; Smith and Maynard Smith 1998), and on other techniques (Jakobsen et al. 1997, 1998; Weiller 1998). See Drouin et al. (1999) for a more detailed history and for a comparison of a number of these methods.

Statistical procedures can be used to rank the possible gene conversion events in an alignment and also provide multiple-comparison corrected P-values for each instance (Sawyer 1989). The program GENECONV extends the methods in Sawyer (1989) in several ways, such as

- allowing mismatches within possible gene conversion events,
- better global comparison procedures,
- new methods for detecting gene conversion events from outside of the alignment,

- as a program option, listing sequences immediately adjacent to the endpoints of putative gene conversion events, and
- restricting to within-group comparisons for more powerful multiple-comparison procedures.

The theory behind GENECONV as well as the details of using GENECONV are explained below, along with many examples. Most of the examples are included on the Web site as example input and output files. See A quick start: program input and output below if you want to starting using GENECONV immediately. However, you should also look at Assessing significance: pairwise and global P-values at least briefly to see the difference between global and pairwise fragments. (Global fragments have P-values that are multiple-comparison corrected for all possible sequence pairs, while pairwise fragments do not. Global fragments are more important than pairwise fragments. Low P-values for pairwise fragments might be due to a large number of pairwise comparisons. Both P-values are naturally corrected for sequence length.)

For some recent applications of GENECONV and its immediate precedessors, see Chenault and Melcher 1994; Kapur et al. 1995; Revers et al 1996; Witherspoon et al. 1997; Semple and Wolfe 1999; and Padidam et al. 1999.

FINDING GENE CONVERSION EVENTS: FRAGMENTS AND HSAPs

Given an alignment of DNA or protein sequences, GENECONV looks for aligned segments for which a pair of sequences are sufficiently similar to be suggestive of past gene conversion. As a program option, GENECONV will consider only silent sites and codon polymorphisms. There are also ways for finding gene conversion from outside of the alignment; see below.

The basic procedure is as follows. First, monomorphic sites are excluded from the alignment as a control for constant or highly selected sites. One then looks for maximal aligned pairs of segments between sequences that are either (1) pairwise identical and unusually long for that pair of sequences or else (2) have an unusually high score for that pair of sequences, where matches count as +1 and there is a penalty for mismatches. The mismatch penalty depends on the density of polymorphic sites between the two sequences and on a user-specified mismatch intensity parameter. (If monomorphic sites are kept in the alignment but are scored as 0, the effect is the same.) Mismatches could be due to later mutation within a gene conversion segment, or, possibly, due to incomplete mismatch repair during the gene conversion event. High-scoring or unusally long segments are candidates for being possible gene conversion events. P-values are assigned to these events in a manner described in the next section.

In more detail, for a given pair of sequences, we call a polymorphic site *discordant* if the sequences differ at that site and *concordant* if they have the same base (Sawyer 1989). We define a *fragment* as an aligned pair of segments between two sequences that is bounded by either two discordant sites, or else by one discordant site and an end of the alignment. (The entire alignment is excluded, as well as empty fragments.) Unless mismatches are allowed, the fragment must consist entirely of concordant sites. Fragments CAN contain discordant sites if mismatches are allowed. Sometimes we will use the term HSAP (for high-scoring aligned pair) as a synonym for fragment.

When mismatches are allowed, there is a mismatch penalty that is inversely proportional to the total number of site differences between the two sequences. This mismatch penalty is also proportional to a ``gscale" constant that is set by the user. The ``gscale" constant is related to the age of the gene

conversion events that one wishes to detect. See Mismatch penalties below for more detail.

ASSESSING SIGNIFICANCE: PAIRWISE AND GLOBAL P-VALUES

GENECONV finds and ranks the highest-scoring fragments globally for the entire alignment. If pairwise lists are specified (see options for determining fragment lists), this is also done for each sequence pair. Pairwise P-values are assigned that compare each fragment with the maximum that might have been expected for that sequence pair in the absence of gene conversion. Global P-values compare each fragment with all possible fragments for the entire alignment.

Note that an alignment of 7 sequences has 6*7/2=21 possible pairs of sequences. If these sequence comparisons are independent, then one would expect at least one sequence pair out of the 21 to have a statistically significant fragment by chance even if there is no history of gene conversion. Thus primary evidence for gene conversion should come from the global fragment list and global P-values. Once one is sure that there is evidence for gene conversion between a particular pair of sequences, the pairwise lists can give additional qualitative information.

Pairwise P-values have a built-in multiple-comparison correction for the length of the alignment. That is, pairwise P-values automatically allow for the possibility that a fragment is long simply because some fragment may be long by chance in a long alignment. Global P-values are more conservative, and allow for all possible pairs of sequences.

GENECONV assigns P-values by two methods for both global and pairwise P-values. The first method is based on permutations. The second method finds approximate P-values by a method of Karlin and Altshul (1990, 1993). The first method is more accurate, but the second method is computationally much faster. Karlin-Altschul P-values are the basis of the popular BLAST method for finding sequence matches in DNA or protein databases (Altschul et al. 1990). If the number of polymorphisms is sufficiently large, these P-values will be close to the ``pairwise'' permutation-test P-values discussed below.

If the number of polymorphisms is larger yet, Karlin-Altschul P-values that are Bonferroni-corrected for the number of possible sequence pair comparisons (see below) seem to be close to the ``global'' permutation P-values. However, as far as we know, whether this is generally correct is an open mathematical question at present. Karlin-Altschul (KA) P-values are generally more conservative than permutation pairwise P-values. Global KA P-values are more conservative yet, and can be 100 or 1000 times as large as global permutation P-values. (See examples below.)

The permutation procedure is as follows. View the alignment as a two-dimensional array whose rows are sequences and whose columns are polymorphic sites. The highest-scoring fragments are determined for this alignment, both globally and for each pair of sequences. Then, for each of N trials, the columns (polymorphic sites) of this array are randomly permuted among themselves. The columns are permuted as a whole, so that each row sees the same permutation of its bases at polymorphic sites. The maximum fragment score is found for the permuted array, both for each pair of sequences and for all pairs of sequences.

For each of the original fragments, the pairwise P-value is the proportion of permuted alignments for which the maximal fragment score for that pair of sequences is greater than or equal to the original

fragment score. (That is, the pairwise P-value is the number of such permutations divided by N.) The global permutation P-value is the proportion of permuted alignments for which the maximum fragment score for all pairs of sequences in the permuted alignment is greater than or equal to the observed score. In other words, the global permutation P-value is the proportion of permuted alignments for which some fragment for some pair of sequences has a higher score than the observed fragment. By default, GENECONV uses N=10,000 permutations in the permutation procedure.

Global permutation P-values have a built-in multiple-comparison correction for all sequence pairs in the alignment, while pairwise P-values do not. As mentioned above, with completely random data with 7 sequences and 6*7/2=21 pairs of sequences, one would expect approximately one sequence pair to have one or more fragments that are pairwise significant at the 5% level. Using global P-values protects against statistical artifacts of this kind. The analogs of global permutation P-values for Karlin-Altschul P-values are Bonferroni-corrected KA P-values (Bonferroni 1936, Miller 1981). These are KA P-values multiplied by the number of sequence comparisons. For example, if there are 10 sequences in the alignment, then there are 45 sequence comparisons, and a fragment with a (pairwise) KA P-value of 0.001 has a global or Bonferroni-corrected KA P-value of 0.045. There is no simple relation between pairwise and global permutation P-values.

Global P-values require global scores, since each fragment is being compared with the best that could be expected for the entire alignment. Global permutation P-values are based on a globalized version of the BLAST score (Altschul 1993) rather than on the numerical value of the score or fragment length for that pair of sequences. Using the uncorrected fragment length or fragment score can introduce a strong bias against fragments between sequence pairs that are not among the most closely related pairs in the alignment. Specifically, more distantly-related pairs of sequences will most likely have a higher density of site differences. A fragment that is highly significant against this background may have a length that is not significant for a more closely-related pair of sequences with fewer site differences. This means that its global significance (based on uncorrected fragment lengths or scores) may be even less.

GENECONV global measures are based on BLAST-like global scores, which correct for this. The section An immune-system example: corrected scores do make a difference below has an example of an alignment of three sequences in which a particular fragment between two relatively diverse sequences has (i) a pairwise permutation P=0.0011, (ii) a global permutation P=0.263 based on uncorrected fragment lengths, but (iii) a global permutation P=0.0011 based on BLAST-corrected fragment scores. As a program option, GENECONV can use Bonferroni-corrected pairwise permutation P-values for global permutation P-values, exactly as with KA P-values. The BLAST-like global scores are generally more powerful, but sometimes it is advantageous to use Bonferroni-corrected pairwise permutation scores.

The BLAST score is a linear function of the fragment score that depends on the sequence pair. (See An immune-system example below for the formulas in that case.) The pairwise KA P-value is a function of the BLAST score. Specifically,

Pairwise KA P-value = 1 - exp(-exp(-BLAST score))

(Karlin and Altschul 1990, Altschul 1993). If the BLAST score is large, this is approximately equal to exp(-BLAST score).

Both permutation and KA global P-values are multiple-comparison corrected for all possible sequence pairs. If mismatches with mismatch penalties are allowed, the P-values depend on the choice of

mismatch penalties, which is controlled by the value of a parameter ``gscale". (See Mismatch penalties below.)

The polymorphisms and permutation scheme are slightly different when GENECONV is told that the aligned sequences are from a coding region of a gene. Sites within amino-acid polymorphic codon positions are ignored, and codon polymorphisms are used instead of site polymorphisms. See Silent sites and amino acids below for the details.

GENECONV is based on a dynamic programming algorithm that finds and sort HSAPs in O(n) steps for each pair of sequences, where n is the length of the alignment. See Waterman and Eggert (1987) for a somewhat similar algorithm that finds high-scoring nonaligned fragments in $O(n^2)$ operations.

INNER AND OUTER FRAGMENTS

The fragments discussed above are ``inner" fragments, which means that they are evidence of a possible gene conversion event between ancestors of two sequences in the alignment.

GENECONV can also look for *outer* fragments, which are evidence of past gene conversion events that may have originated from outside of the alignment, or else from within the alignment but such that evidence of the source has been destroyed by later mutation or gene conversion. GENECONV has three different ways for detecting ``outer" fragments. (See Outer fragments in the More theory section below.) Sequence alignments generally have more significant ``inner" than ``outer" fragments, but highly significant ``outer" events can be found by any of the thee methods. By default, GENECONV finds significant ``inner" and ``outer sequence" fragments, but can be told to look for all three types of outer fragments, or for none of them.

A QUICK START: PROGRAM INPUT AND OUTPUT

Aligned input sequence files for GENECONV can be in any of a number of different formats, specifically NEXUS, Pearson/FASTA, NBRF/PIR, CLUSTAL, ASF, or PHYLIP interleaved. Example input and output files for a number of these file types are included on the Web site. See Format of sequence files below for more detail.

For definiteness, assume that you have an aligned sequence file called `myseqfile.seqs` in one of these formats. To analyze this alignment for gene conversions using GENECONV, you can enter

```
geneconv myseqfile.seqs
```

in a command-line window (sometimes called a ``Dos prompt" window). The program GENECONV must be in either the default directory (so that you can see it if you enter ```dir`") or else in a directory on the system path. The sequence file `myseqfile.seqs` must be in the default directory. This command will analyze `myseqfile.seqs` using GENECONV's default options, which are generally reasonable. Options can be added on the command line with prefixes ``/" or ``-" . These can either precede or follow the sequence file name and can be in any order. (See examples below.)

Alternatively, you can use GENECONV_HELPER . First, run GENECONV_HELPER, for example by clicking on its name in an appropriate window. Enter ``myseqfile.seqs" under ``Enter Sequence file name..." in the GENECONV_HELPER main menu and any options under ``Enter options" . Or, just enter the input file name and any options together in either location. Make sure that a copy of myseqfile.seqs is available in the file directory that is listed above the GENECONV_HELPER menu, or else enter the name of a directory containing myseqfile.seqs under ``Optional input file path..." . Then enter ``R." for Run in the Main Menu and follow the directions.

By default, GENECONV assumes that the input sequence file is composed of DNA, RNA, or protein sequences and that all polymorphic sites should be used to define fragments. GENECONV will normally guess from the sequences themselves whether they are composed of nucleotides or amino acids. However, you can indicate which explicitly. Alternatively, the sequences can be from a coding region of a gene and GENECONV should consider only polymorphic sites in amino-acid monomorphic codon positions. See Silent sites and amino acids below for more details.

Most GENECONV program options have reasonable default settings and can be ignored. You can also put program settings in a configuration file. (See Configuration files below.) This can be convenient if you do related analyses on more than one alignment with the same option settings, or a variety of similar analyses on the same alignment. GENECONV_HELPER remembers your most recent program options, which makes it easier to run the program more than once with similar settings. See A first example below for some simple examples using GENECONV.

For output, GENECONV writes an ordered list of the most significant fragments to a spreadsheet or spreadsheet-like file. Only significant global fragments are listed by default, but pairwise lists can be generated by command-line options. Output file also contain the program settings and other information. Depending on program options, the output file may begin with dozens of comment lines about the input file along with a list of the program options that were used. The output file can have space-separated columns, tab-separated columns, or be a spreadsheet file in DIF format. The default is space-separated columns so that the output can be read more easily in a text editor or word processor. See Spreadsheet output below for more details.

In sequence data, GENECONV ignores digits (0-9), spaces, parentheses (<>()[]{}), and quote characters ("`'). This allows you to enter line numbers and offsets in sequence data. Otherwise, GENECONV does not distinguish among unrecognized characters, missing data, and indels. By default, GENECONV treats a maximal run of contiguous sites, each with unrecognized data, as a single polymorphism. This is a protection against alignment artifacts. However, GENECONV can be told either to ignore sites with indels or missing data altogether, or else to generate a polymorphism for each site. (See indels and missing data below.)

By default, fragments are included in global or pairwise lists if the P-value is 0.05 or smaller. Fragment lists are further restricted so that, given a high-scoring fragment, there are at most M-1 further maximal fragments that overlap that fragment. The default is M=1, which means that fragments that are listed in the output do not overlap. This is an issue only if mismatches are allowed along with a mismatch penalty. If mismatches are not allowed (which is GENECONV's default), then maximal fragments, by definition, do not overlap. If mismatches are allowed, it may be useful to list the highest-scoring two or three fragments in each cluster of overlapping fragments. The value of M can be set as a program option. See Overlapping fragments below.

Input files can be Microsoft, UNIX, or MacIntosh text files, regardless of the host system. Text files are read as binary files with end-of-line characters treated appropriately. Line buffers are expanded dynamically, so that there is no limit on line size.

Fragment lists can be further limited by restrictions on fragment length, fragment score, or the number of polymorphic sites. (See Fragment lists: other restrictions below.) The lists can also be limited to within-group fragments for an imposed group structure; see the next section.

GROUP STRUCTURE: WITHIN-GROUP FRAGMENTS ONLY

Some sequence alignments have a natural group structure in which gene conversion is expected mostly within the same group. For example, an alignment may contain sequences from several genera, and one may expect gene conversion to occur mostly between sequences of the same genus. (See groups of unequal size below.)

Alternatively, one may have aligned sequences from a gene family on several different chromosomes. For biological reasons, gene conversion between members of the same gene family on the same chromosome may be more likely than between-chromosome gene conversion. For other biological reasons, within-chromosome gene conversion may be less likely than homologous between-chromosome gene conversion.

In any case with group structure, GENECONV can be told to look for within-group gene conversion only. The smaller number of between-sequence comparisons can then allow suspicious fragments to be highly significant after multiple-comparison corrections even when no fragments are significant after multiple-comparison corrections for all possible sequence pairs. (See An example with groups of equal size below.)

POLYMORPHISMS AND POWER

When GENECONV analyzes an alignment, it first removes all monomorphic sites and looks for fragments among the remaining polymorphic sites. The principal reason for this is that monomorphic sites may be fixed for biological reasons. An apparent gene conversion event that contains many monomorphic sites may be given too much statistical weight, and apparent gene conversion events elsewhere in the alignment may be given too little weight. In practice, including monomorphic sites in an alignment appears to give similar results but with decreased power (Sawyer 1989).

Results for many within-species alignments tend to be improved if additional sequences from related species are added to the alignment. The reason for this appears to due to adding polymorphisms to the alignment in a density distribution that matches the distribution of polymorphisms in the within-species alignment. The disadvantage is that adding sequences to the alignment increases the Bonferroni correction for the sequences pairs that you may be interested in, namely those within the first species.

For example, suppose that you are interested in possible gene conversion at a genetic locus within Species I and gather 10 sequences from Species I. As a way of increasing the number of polymorphisms, you align these sequences with 5 additional sequences from Species II. If you analyze

all 15 sequences together, the Bonferroni correction for pairwise comparisons within Species I will rise from 45 (10*9/2) to 105 (15*14/2). GENECONV's method for handling multiple comparisons for global P-values is more efficient than a straightforwards Bonferroni correction, but the relative effects on global P-values are likely to be similar.

One way of reducing this penalty is to say that gene conversion between species is not likely or at least that you are not interested in gene conversion between the two species (assuming that either is actually the case) and tell GENECONV that the two species are groups. Then GENECONV will only look for within-species gene conversion events and adjust Bonferroni corrections accordingly. This reduces the Bonferroni correction from 105 to 55 (10*9/2 + 5*4/2) due to the decreased number of paired comparisons.

As a third alternative, you can use Species II purely to increase the number of polymorphisms and tell GENECONV not to look at any paired comparisons involving sequences from Species II. This restores the original Bonferroni correction of 45 (10*9/2) while benefitting from the increased number of polymorphisms. (See groups of unequal size below.)

Which of these three alternatives is preferable for the alignment of 15 sequences depends on the aims of your study. If you are only interested in gene conversion events within Species I, then you can tell GENECONV to look only at those paired comparisons. This will result in the most significant multiple-comparison corrected P-values. If you are interested in within-species comparisons, or in pairwise comparisons among all 15 sequences, then those are what you should look at, even if the individual multiple-comparison corrected P-values will be less significant.

COMPARISON WITH PREVIOUS PROGRAMS: VTDIST AND VTDIST3

The program GENECONV generalizes two earlier computer programs (VTDIST and VTDIST3) based on Sawyer (1989) in the following ways: (i) mismatches within fragments are allowed as a program option; (ii) Karlin-Altschul P-values are found as well as the computationally slower permutation P-values; significant fragments are found and ranked by either P-value; and (iii) BLAST-like scores are used to compare fragments across sequence pairs in multiple-comparison-corrected procedures instead of uncorrected fragment lengths. GENECONV also has two new types of outer fragments. (See Outer fragments below.)

VTDIST3 extends VTDIST by (iv) finding ``pairwise" as well as ``global" (multiple-comparison corrected) permutation P-values and (v) allowing the sequences to have a group structure with searches for within-group fragments only. The latter allows more powerful global statistical procedures for finding within-group apparent gene conversion events. (See An example with groups below.)

The program VTDIST was based on the statistical procedures described in Sawyer (1989). These procedures are based on the statistics (i) MCF, which uses a multiple-comparison corrected (global) procedure to find and list the most significant fragments in an alignment, based on comparing the numbers of polymorphic sites in fragments across all sequence pairs; (ii) SSCF, which uses the sum of the squares of fragment lengths across all sequence pairs. SSCF measures a pattern of apparent gene conversion in the alignment but does not identify individual fragments; and (iii) MUF and (iv) SSUF, which are the same as MCF and SSCF except that fragment lengths are lengths in the alignment instead of the numbers of polymorphic sites. P-values are found by the same permutation procedure described

earlier. VTDIST3 finds pairwise as well as global SSCF scores.

As mentioned earlier (Assessing significance), using uncorrected fragment lengths across all sequence pairs in an alignment can introduce a bias against fragments between more distantly-related sequence pairs. The SSCF statistic is less sensitive to this bias than MCF, but this bias can be a problem in either case. (See An immune-system example: corrected scores do make a difference below.)

2. USING GENECONV

To use GENECONV, you will need a working version of the program, an aligned DNA or protein sequence file, and some idea of what options that you want to use. GENECONV's default options are enough in most cases. GENECONV has approximately 80 options to customize the analyses and output and to handle different kinds of input files. The most important options are described in the examples below. Most options are listed in the Options List section below.

The program GENECONV itself is designed to be used in console or command-line mode. Alternatively, GENECONV_HELPER has a text-window interface that allows you to enter GENECONV program options in an interactive manner. (GENECONV_HELPER calls GENECONV with an appropriate command line.) Future versions of GENECONV will have a GUI interface with drop-down menus, buttons, and graphics edit windows. See A quick start: program input and output above for more information about the mechanics of running GENECONV and GENECONV_HELPER.

C source for GENECONV and GENECONV_HELPER can be downloaded from the GENECONV Web site. Executable versions of these programs for Microsoft Windows 95/98/NT systems are also provided. There are several sample input and output files as well as this documentation file. The C source has been compiled and tested on several different UNIX systems, and should work on any system with an ANSI standard C compiler.

A FIRST EXAMPLE

The example input file `alumw.asf` on the Web site has 14 aligned Alu sequences from 6 different primate species (Maeda et al. 1988). Alu sequences are a family of repeated sequences that have no known biological functions in themselves but are widespread in the DNA of higher primates. The Alu family has approximately 500,000 copies in humans and takes up about 5% of total human DNA (Deininger 1989). The alignment `alumw.asf` has 7 inverted pairs of Alu repeat sequences from 6 species ranging from spider monkey to human. The authors found one apparent gene conversion event between the two spider monkey Alu sequences, but provided no statistical analysis.

To analyze this alignment for gene conversions using GENECONV, enter

```
geneconv alumw.asf  /w123  /lp
```

(See A quick start: program input and output above for the mechanics of using GENECONV and GENECONV_HELPER.)

The option `/w123` initializes GENECONV's internal random-number generator at that value. While not strictly necessary, it guarantees that program output will always be the same, even if run on different computers at different times. If the random-number generator is not initialized explicitly, then it is set from the system clock. The option `/lp` tells GENECONV to produce lists of pairwise significant fragments as well as global lists. Unless you indicate otherwise, GENECONV only creates global lists.

By default, GENECONV finds permutation and KA P-values for the most significant fragments using 10,000 random permutations. Mismatches within fragments are not allowed. (Equivalently, the default mismatch penalty is infinity.) Information about the most significant fragments is written to the output file `alumw.frags` , which is also included on the Web site. Error messages and other information are written to the log file `alumw.sum` .

In this case, GENECONV finds one globally significant inner fragment (that is, significant after multiple-comparison corrections for all possible sequence pairs) and 5 pairwise significant inner fragments. The globally significant inner fragment is the fragment found by Maeda et al (1988). It has a global permutation P-value of P=0.0213 and a pairwise P-value of 0.0010.

The output file `alumw.frags` begins with descriptive comments about the sequence file (the numbers and types of polymorphisms, nucleotide frequencies, etc.) and the options used by GENECONV. Lines with descriptive comments start with ``#" to make it easier to locate lines with information about specific fragments. After these comments is a ``global list" of the most significant fragments in the alignment, which begins with GI for ``global inner" (fragment) in the first two columns. The P-values in this list are multiple-comparison corrected for all sequence pairs, as well as for the length of the alignment. The ``global list" in this case is

```
#   Seq     Sim    BC KA    Aligned Offsets   Num  Num  Tot  MisM
#   Names  Pvalue  Pvalue   Begin  End   Len  Poly Dif  Difs Pen.
GI  S1;S2  0.0213  0.28156   266    296   31    21   0   34  None
#
# One inner fragment listed.
```

This says that the only significant global fragment is between the two sequences `S1` and `S2` (the two spider monkey sequences) at offsets 266-296 in the alignment. The column heading `BC KA` stands for ``Bonferroni-corrected Karlin-Altschul" Pvalue. Note that it is more conservative that the permutation-test P-value in this case. Both P-values are multiple-comparison corrected. The rest of the `GI` line says that the fragment contains 21 sites that are polymorphic in the alignment and that the two sequences differ at 34 sites overall. The final three columns say that the fragment contains no internal mismatches and that there is no mismatch penalty, which is the GENECONV default.

Since GENECONV was entered with the command-line option `/lp`, ``pairwise lists" will follow later in the file. These have the significant fragments for each pair of sequences. Pairwise fragments are not multiple-comparison corrected for sequence pair, so that there are generally many more of them. The list for the 5 pairwise fragments is

```
#   Seq     Sim      KA     Aligned Offsets   Num  Num  Tot  MisM
#   Names  Pvalue  Pvalue   Begin  End   Len  Poly Dif  Difs Pen.
PI  H1;G1  0.0491  0.13001   218    280   63    33   0   12  None
PI  I1;G2  0.0476  0.06919   282    296   15     9   0   49  None
PI  G1;G2  0.0299  0.04969   281    296   16    10   0   47  None
PI  O1;S1  0.0213  0.04325   160    212   53    15   0   34  None
PI  S1;S2  0.0010  0.00309   266    296   31    21   0   34  None
#
# 5 inner fragments listed.
```

The last entry has the pairwise permutation P-value for the spider-monkey fragment. There were no significant global or pairwise outer fragments. (See Outer fragments below.)

Unless told otherwise, GENECONV includes fragments in global lists if the multiple-comparison-corrected permutation P-value is 0.05 or smaller and in pairwise lists if the permutation pairwise P-value is 0.05 or smaller. With 14*13/2=91 pairwise comparisons, one might expect around 5 pairs of sequences to have pairwise significant fragments with a P-value of 0.05 or smaller by chance, as is observed here.

Screen output says that 5 of the 95 permuted polymorphisms are actually runs of polymorphic sites with indels, and that one of these runs is 41 sites long. Most of the information on the screen is also written to the log file `alumw.sum` . If you add the option `/sp` to the command line (or to the `Option` list in GENECONV_HELPER), then the polymorphisms and their offsets will also be written to `alumw.frags` . This output shows that the 41-site run of polymorphic sites occurs at offsets 3-43. The program (with or without `/sp`) takes 10" to run on a Pentium 400Mhz machine running Windows NT. Most of this time is taken up by the 10,000 random permutations and the bookkeeping involved.

Enter `/r` on the command line (in any order after `geneconv`) if the input file has DNA sequences from a coding region and if GENECONV should use silent polymorphisms only. By default, GENECONV uses all polymorphic sites, and determines for itself whether the sequences are DNA/RNA or amino acids (proteins) from the distribution of bases. You can force GENECONV to assume DNA or RNA sequences by entering `/a` and amino-acid (protein) sequences by entering `/p` . See Silent sites and amino acids below for more detail.

Most GENECONV options have both a short form (like `/w123` or `/sp`) and a long or verbose form. The short form is easier to enter on a command line. The long form is more explicit and more readable. The long forms for `/w123` and `/sp` are `-Startseed=123` and `-Showpolys` , respectively. Short-form arguments always begin with `/` and can be grouped together, as in `/rw123sp` . Long-form arguments always begin with `-` and cannot be grouped together in the same way. The long forms for `/r` (silent sites), `/p` (amino-acid sequences), and `/a` (DNA, all polymorphic sites) are `-Seqtype=Silent`, `-Seqtype=Protein`, and either `-Seqtype=Nucleotides` or `-Seqtype=All`. Options are case insensitive, so that you could also enter `-Seqtype=ALL`, `-SEQTYPE=Protein`, or `/R`. Long-form options use initial string matching, so that you can enter `-Showpolysites` instead of `-Showpolys` .

In the last example, GENECONV looked only for fragments without internal mismatches. See Cauliflower mosaic viruses below for an example that is analyzed both with and without mismatch penalties.

WARNING: Some operating systems sometimes break command-line arguments of the form `word1=word2` into two words, so that, for example, `-Seqtype=ALL` on the command-line is converted to `-Seqtype ALL`. This will cause GENECONV to halt with an error message. If this happens, try entering the arguments within quotation marks, as in `"-Seqtype=ALL"` .

PAIRWISE AND GLOBAL P-VALUES

The fragment found by Maeda et al. (1988) has a pairwise permutation P-value of 0.0010 and a global permutation P-value of 0.0213. That is, only 10 permutations out of 10,000 have a longer

fragment for that pair of sequences, but 213 out of 10,000 had some fragment that was this long or longer (with length measured as BLAST score) for SOME pair of sequences.

The difference between the pairwise P-value 0.0010 and the global P-value 0.0213 for the same fragment is related to the fact that, with n=14 sequences, there are 91 possible pairs of sequences. The corresponding KA (Karlin-Altschul) P-values for the same fragment are 0.00309 pairwise and 0.282 globally. Thus the pairwise KA P-value is about 3 times more conservative than the permutation P-values and the Bonferroni-corrected KA P-value is 14 times more conservative. The global KA P-value in this case is the KA P-value multiplied by 91.

To have GENECONV compute only the faster approximate KA P-values, enter (for example)

```
geneconv alumw.asf alm1 /n0  /lp
```

where `/n0` means no permutations. With no permutations, the program runs in around 0.10" (a tenth of a second) on a P400 Pentium machine, as opposed to 10" with 10,000 permutations. The output in this case is written to `alm1.frags` with log output to `alm1.sum` . We used a different output file name here rather than overwrite the previous output for `alumw.asf` in `alumw.frags` . The option `/w123` was not entered here since GENECONV does no randomization in this case, so that initializing the random number generator has no effect.

With the above arguments, GENECONV finds no globally significant KA fragments and 3 pairwise significant KA inner fragments. When no permutations are done, GENECONV's default behavior is to include fragments in global lists if the Bonferroni-corrected KA P-value is 0.05 or less and in pairwise lists if the KA Pvalue is 0.05 or less.

Enter `/n1000` instead of `/n0` to have GENECONV calculate permutation P-values using 1000 permutations instead of 10,000. The same options in long form are `-numsim=1000` and either `-numsim=0` or `-numsim=none` . GENECONV doesn't care what you enter in a long-form option beyond an initial string match and normally ignores case, so that `numsim=none`, `Numsims=None`, and `Numsims_today=NoWay!` have the same effect. (See Basic GENECONV syntax in the glossary.)

To change GENECONV's behavior so that fragments are included in global lists if the global (Bonferroni-corrected) KA Pvalue is 0.15 or less instead of the default 0.05 or less, enter

```
geneconv alumw.asf alm2 /n0 /mkg0.15
```

The long form for the last option is `-MaxKAGlobalPval=0.15`. Conditions on KA P-values for global fragments can be removed entirely by `/mkg2`, where 2 can be replaced by any number that is 1 or greater. The option `/mkg2` in long form is `-MaxKAGlobalPval=None`. You can substitute ``Mc" or ``Glob" for ``Global", where ``Mc" stands for ``Multiple-comparison (corrected)". Thus GENECONV will also understand `-MaxKAMcPval=0.01`. See options for P-value bounds in the Options List below for the options for specifying bounds for pairwise KA P-values and permutation P-values.

As an alternative to specifying bounds based on P-value, you can enter

```
geneconv alumw.asf -listbest /w123
```

The option `-listbest` (short form: `/lb`) tells GENECONV to ignore P-values and list the 8 most significant global fragments overall. If pairwise lists are specified, the 3 most significant fragments are

listed for each pair of sequences. This can be useful if you are not sure how many significant fragments there are and if you want to make sure that you obtain at least some output. It will be obvious from the output whether any of the listed fragments are significant.

Enter `-Nomaxpvals` to remove all restrictions on the P-values of listed fragments. This can result in a huge output file if the alignment is large (unless, of course, you enter `-listbest`).

If one of the two KA P-value bounds is present but the other is removed, then the other KA P-value bound is applied to both pairwise and global lists. This automatic ``crossover'' is not applied to permutation P-value bounds. See Fragment lists: Restricting by P-value below for more details.

CHECKING GENECONV

As a check on GENECONV, you can have GENECONV first permute the polymorphic sites once and then apply the same analyses. The permuted data should have few or no significant fragments. To do this with the input file `alumw.asf` of the last section, enter

```
geneconv alumw.asf rantest /w123 -Randomize_sites  /lp
```

The option `-Randomize_sites` tells GENECONV to permute the polymorphic sites once before proceeding, which should destroy most significance. (For safety, this option has no short form.) As before, `/w123` initializes the random-number seed at `123` for reproducibility, and `rantest` names the output files. The main output is written to `rantest.frags` and the log file is `rantest.sum` .

Running the program with these options finds no significant global inner fragments. The smallest global permutation P-value for inner fragments is 0.5852. There is one significant pairwise inner fragment with a permutation P=0.0158 and KA P=0.0601, but no other significant pairwise fragments. If the n=91 sequence pairs were independent, then we would expect to find one pairwise significant fragment at the 0.01 level and 5 at the 0.05 level purely by chance. Finding one significant pairwise inner fragment is well within these bounds.

As another test, you can run

```
geneconv rand950.fasta /w123  /lp
```

The sample input file `rand950.fasta` has randomized data for a nucleotide alignment with 500 bases, 9 sequences, and 50 polymorphic sites. The output shows that the most significant global fragment has permuation P-value P=0.380, which is not significant. There is only one significant pairwise fragment with a pairwise permutation P-value of P=0.0183. Since there are 9*8/2=28 sequence pairs with 9 sequences, this is consistent with randomness.

By design, no sites in `rand950.fasta` have more than two nucleotides. All polymorphic sites are phylogenetically informative --- that is, there are no bases at a polymorphic site that occur in only one sequence. This means that `rand950.fasta` can also be used as a test input datafile for other programs than either use only informative polymorphic sites or else use noninformative polymorphic sites inefficiently. (GENECONV normally uses all polymorphic sites. To restrict GENECONV to use phylogenetically informative sites only, enter the command-line option `-Mult_poly_only` ; short form: `/dm` . See options for permutations in the glossary.)

The sequence file `rand950.fasta` was generated by a program `make2rand` using its default parameters and initial seed `456`. C source for `make2rand` is on the GENECONV Web site as both Win32 and UNIX text files.

AN EXAMPLE WITH GROUPS OF EQUAL SIZE

The 14 sequences in `alumw.asf` represent a nearby inverted pair of Alu sequences from 7 primate chromosomes. Close similar inverted DNA segment pairs are thought to undergo gene conversion more easily than otherwise, and gene conversion between different species is rare. For both reasons, we would expect most gene conversion events to be between Alu pairs on the same chromosome. Thus we would be justified in looking only at within-pair fragments and applying multiple comparison corrections ONLY for those sequence pairs.

In this case, there are 91 total sequence pairs, but only 7 within-inverted-Alu-pair comparisons. Thus the Bonferroni correction for global within-pair KA P-values is to multiply the pairwise KA Pvalue by 7 instead of 91. The corresponding global permutation P-values based on BLAST-like scores should also be more significant.

The 14 sequences in `alumw.asf` are ordered as 7 consecutive pairs of Alu sequences from the 7 primate chromosomes. GENECONV can be told to view consecutive pairs of sequences in the alignment as groups, and to consider only within-group or within-pair fragments, by entering

```
geneconv alumw.asf /b2 alumwbc /w123  /lp
```

Enter `/b3` instead of `/b2` to tell GENECONV to consider consecutive triples of sequences in `alumw.asf` as groups and look only for within-group fragments. (Since 3 does not divide 14, this will cause GENECONV to exit with an error message.) The long form of `/b2` is `-Blocsize=2`. Output in this case is written to `alumwbc.frags` and `alumwbc.sum`.

The within-block analysis finds one fragment with a global permutation P-value of 0.05 or less, the same as before. The fragment found by Maeda et al. (1988) now has a global permutation P-value of 0.0043 as opposed to 0.0213. The Bonferroni-corrected KA P-value is 0.0217 instead of 0.282. There are two significant pairwise fragments. In addition to the spider monkey fragment, a second significant fragment is found between two gorilla Alu sequences with a pairwise P-value of 0.0299. While this is not statistically significant after correction for multiple comparisons, it is suggestive of a possible second gene conversion event.

In general, homologous Alu sequences on different chromosomes tend to be more similar than nearby Alu sequences on the same chromosome (Deininger 1989). To test for homologous *between-chromosome* gene conversion instead of *within-pair* gene conversion, enter

```
geneconv alumw.asf /h /b2 alumwhbc /w123  /lp
```

(The long-form of `/h` is `-Homologous`.) With the same arrangement of sequences within `alumw.asf`, this tells GENECONV to consider two groups of 7 sequences each (one for each locus) and to look for within-locus gene conversion only. No globally significant inner fragments are found.

However, there is one possibly suggestive *pairwise outer sequence* fragment (P=0.0436) at offsets 1-57 in one of the two spider monkey Alu sequences. An outer sequence fragment is a maximal DNA segment in a single sequence for which the bases in that sequence are unique within its group at all polymorphic sites in the alignment (see Outer fragments below). This suggests that the first part of this Alu sequence might have been the result of a gene conversion event from outside of that chromosome. The significant spider monkey inner fragment is at offsets 266-296, so that there is no overlap. However, a gene conversion at either fragment would make the other fragment appear more significant by its effect on the distribution of polymorphic sites.

CONFIGURATION FILES

The command line in the last example has only four options, but might be inconvenient to enter if you were doing many related analyses. The situation would be worse with more options, as in (for example)

```
geneconv alumw.asf  alumwh2  /b2 /h /lp /n1000  /w123
  /mip2 /mig0.25 /mkp0.08
```

This command line is also quite cryptic due to the abbreviated short-form options. Using the equivalent long-form expressions for the options would be clearer, but might overflow the system command-line buffer. A solution to both problems is to write a configuration file with these options in their long forms and then refer to the configuration file.

For example in this case, write a text file called `myopts.cfg` with contents

```
#GCONV_CONFIG
   alumw.asf  alumwh2
   -Options  Blocsize=2   Homologous  ListPairs
     Numsims=1000  Startseed=123
     % P-value conditions on fragments
     MaxSimPairwisePval=none  MaxSimGlobalPval=0.25
     MaxKAPairWisePval=0.08  Endoptions
```

The name of a configuration file must end with `.cfg` . The first word in a configuration file must be `#GCONV_CONFIG` . That is, the file must begin with zero or more spaces, then these 13 characters, then one or more additional spaces or end-of-line characters. The matching is case insensitive, so that `#gconv_config` or `#Gconv_Config` will also work. The character `%` in a configuration file tells GENECONV to treat the rest of that line as a comment and to ignore it. The character `%` need not be the first character on that line. If the first character after the `%` is `!` (as in `%!` . . .), then the comment is also written to the output file.

In general, a configuration file is read exactly as if its contents after `#GCONV_CONFIG` were on the command line. However, there are a few expressions that can be used only in configuration files. One example is the group `-Options ... Endoptions`, whose sole purpose is to allow you to leave off the initial `-` in long-form options for greater legibility. An `Options` block ends automatically at the end of a configuration file, so that `Endoptions` is optional here. Words in a configuration file can include embedded spaces if they are enclosed in single or double quotes and are contained in one line.

The command

```
geneconv myopts.cfg
```

now produces exactly the same output as the last displayed GENECONV command line. (The output is not quite the same, since the command line and the contents of all configuration files are written to the heading of `alumwh2.frags`. However, the contents of `alumwh2.frags` are the same after the heading.)

Configuration files can call other configuration files, either by listing a file name ending in `.cfg` or by using the option `-Config=<configname>`. GENECONV will only object if it detects an infinite recursion. That is, if `A.cfg` calls `B.cfg` which calls `C.cfg` which calls `A.cfg` again, then GENECONV will exit with an error message complaining about a closed-loop recursion.

Commands can be repeated, whether they are on the command line, in configuration files, or both. If they are repeated, normally the earlier version of the command is ignored and only the last version is used. For example, suppose that you want to use the options in `myopts.cfg` except that you want to change the value of `-MaxSimGlobalPvalues=0.25` to `0.01`. This can be accomplished by entering

```
geneconv alumw.asf myopts.cfg /mig0.01
```

(`/Mig0.01` is the short form for `-MaxSimGlobalPvalues=0.01`.) If you entered `/mig0.01` before `myofpts.cfg` on the command line, then `/mig0.01` would be overwritten by `-MaxSimGlobalPvalues=0.25` in `myopts.cfg` and would have no effect.

GROUPS OF UNEQUAL SIZE

In the last example, all groups were exactly the same size. If you had an alignment from several genera and wanted to look at within-genus gene conversion only, then the numbers of sequences from the various genera may not be the same. GENECONV can handle groups of different sizes, but must be told explicitly what sequences are in which group.

Suppose, for example, that the sequences in `alumw.asf` came from different groups and that we were interested only in within-group gene conversion. The first step is to write the group structure within a configuration file. For example, write a configuration file `alugroup.cfg` with contents

```
#GCONV_CONFIG
  -Startseed=123   -MaxSimGlobalPval=0.05
  -group GRPI   S1  S2
  -group GRPII  H1 C1 I2
  -group GRPII  O1  O2
  -group GRPIII G1 G2 R1 R2
```

The two-letter names are the names of the individual Alu sequences in the alignment `alumw.asf`. For example, `S1` and `S2` are the two spider monkey Alu sequences, and `G1` and `G2` are from a gorilla chromosome.

The four `-group` commands in `alugroup.cfg` define three groups consisting of 11 of the 14 sequences in `alumw.asf`. The other 3 sequences in `alumw.asf` are ignored. `-Group` commands can be used only within a configuration file. See Defining groups and skipping and listing sequences and Syntax of group commands below for more information about the syntax of `-Group` commands.

The first word after each `-Group` command is the group name. Subsequent words are assumed to be sequence names for that group, up to the next word that begins with `-` or `/`, the name of a configuration

file, or the end of the configuration file. Sequence names are added to the same group if a group with that name already exists, so that several `-Group` commands can be used for the same group. The four `-Group` commands define a group `GRPI` with the two spider-monkey sequences `S1` and `S2`, a group `GRPII` with 5 sequences, and a third group `GRPIII` with 4 sequences.

If you enter

```
geneconv alumw.asf alugroup.cfg alugroupout  /lp
```

then GENECONV will analyze `alumw.asf` looking for within-group fragments only. In this case, GENECONV finds one globally significant within-group inner fragment and two pairwise significant within-group fragments. The only globally significant fragment is the same spider-monkey fragment, which now has global permutation P=0.0067 and Bonferroni-corrected KA P=0.053. The multiple-comparison corrected P-values are based on the 17 possible within-group sequence comparisons for three groups of sizes 2, 5, and 4.

Group names that begin with ``Gen_Poly'' are treated in a special way. These are used to add more polymorphic sites to the alignment, but GENECONV does not look for significant fragments within these groups. For example, if `GRPIII` in `alugroup.cfg` is replaced by `GEN_POLY_GRPIII`, then no fragments among the sequences in the former `GRPIII` will appear in the output and the multiple-comparison correction for Bonferroni-corrected KA P-values will be reduced from 17 to 11. Pairwise fragments and pairwise P-values for sequence pairs within GRPI and GRPII will be exactly the same as before.

CAULIFLOWER MOSAIC VIRUSES

The sample input file `camv9.asf` has nine sequences of cauliflower mosaic virus (CaMV; Chenault and Melcher 1994). This particular plant virus has a single circular chromosome consisting of double-stranded DNA. The sequences in `camv9.asf` are complete circular viral chromosomes, so that gene conversion events in the physical sequences may overlap the endpoints of the aligned sequences. The sequences are 8110bp in aligned length and have 856 polymorphisms. Eleven (11) of these polymorphisms are contiguous blocks of sites with indels. (By default, GENECONV groups contiguous blocks of sites with indels into single polymorphisms; see indels and missing data below.)

These sequences were analyzed by entering

```
geneconv camv.cfg camvg0
```

where the configuration file `camv.cfg` is

```
#GCONV_CONFIG
  camv9.asf  -Seq_skip C4 CX
  -Options   Circular  Startseed=123  DumpJseqs
  Endoptions
```

The option `-Circular` (short form `/c`) tells GENECONV that the endpoints of the alignment are physically adjacent, and that GENECONV should look for ``round-the-corner'' fragments as well as fragments that are contiguous within the alignment. Two of the nine sequences, named `C4` and `CX` in `camv9.asf` , are discussed by Chenault and Melcher (1994) but are not considered here. The sequence

C4 is very similar to another sequence in the alignment, and CX is very divergent from the others. The command -Seq_skip C4 CX tells GENECONV to ignore these two sequences, so that only seven of the nine sequences are used in the analysis. See skipping and listing sequences below for the syntax of Seq_skip and the related command Seq_list .

With these options, GENECONV finds 4 globally significant inner fragments and 3 globally significant outer sequence fragments among the 7 sequences (P<0.05). The part of the output file camv9.frags that describes the 4 global inner fragments is

```
#   Seq     Sim    BC KA    Aligned Offsets   Num  Num  Tot  MisM
#   Names  Pvalue  Pvalue   Begin  End   Len  Poly Dif  Difs Pen.
GI  CN;CD  0.0000  0.00006  8018 CA  18  111    28   0  420  None
GI  CD;CS  0.0000  0.00011  7564   7963  400    33   0  361  None
GI  CD;CS  0.0157  0.02561  6734   6943  210    23   0  361  None
GI  CM;CD  0.0165  0.02933  7564   7756  193    19   0  415  None
```

Here ``GI'' stands for ``global inner (fragment)'' and ``BC'' means ``Bonferroni-corrected''. ``CA'' is ``circle around'' for ``round-the-corner'' fragments that overlap the endpoints. Note that the most significant fragment encircles the alignment endpoints. For any two sequences, Tot Difs is the total number of sites at which the two sequences differ. (These are the ``discordant sites'' in finding gene conversion events above.) ``Num Poly'' is the number of polymorphic sites within the fragment.

The option ``DumpJseqs'' in the the configuration file tells GENECONV to write a second output file camvg0.jseqs that lists the bases that are immediately adjacent to the endpoints of the significant fragments. (These are called ``junction sequences''.) This may be useful in identifying preferred gene conversion insertion points in the alignment. In this case, this output shows that 5 of the 7 globally significant fragments begin with the sequences AAT or AAG. This may indicate a preference for gene conversion events in CaMV to insert preferentially at AA sequences. See Alignments at gene conversion endpoints below for more detail. Chenault and Melcher (1994) remarked that many observed junction sequences in this alignment resemble junctions observed in experimentally generated recombinants or initiation sites for DNA or RNA reverse transcription.

MISMATCH PENALTIES

One or both segments involved in a gene conversion event in camv9.asf may have been affected by later mutation. GENECONV can also look for fragments with internal mismatches and use fragment scores with a mismatch penalty.

As mentioned earlier, mismatch penalties in GENECONV are coordinated across different sequence pairs by a scaling or ``gscale'' value, which must be a nonnegative integer. Specifically, suppose that an alignment has a total of ``Npoly'' polymorphic sites among all sequences and that a particular pair of sequences differ at ``Ndiff'' polymorphic sites. (These are the ``discordant'' sites.) If ``gscale'' is positive, matches at polymorphic sites are scored as +1 and mismatches as -MM where

```
MM = Mismatch penalty = Npoly * Gscale / Ndiff
```

Here MM is rounded up to the nearest integer, and both ``Ndiff'' and ``Npoly'' are assumed positive. If gscale=0, then mismatches within fragments are not allowed, which is equivalent to assuming an infinite mismatch penalty. The value of gscale must be either zero or a positive integer. Smaller positive gscale values mean smaller mismatch penalties, and larger positive gscale values mean more

severe penalties. If `gscale>0`, the entire alignment (viewed as a fragment) has a negative or zero score.

Unfortunately, there is no best setting for `gscale`. Different mismatch penalty values make the resulting P-values more powerful for detecting gene conversion events of different ages, or, equivalently, for different relative likelihoods of gene conversion and mutation (Karlin, Dembo, and Kawabata 1990; Altschul 1993). Sometimes `gscale=2` or `gscale=3` will find more significant fragments than does `gscale=0` or `gscale=1`. Usually, however, higher `gscale` values have results that are intermediate between those for `gscale=0` (no mismatches) and `gscale=1` (smallest mismatch penalties).

The default value of `gscale` is `gscale=0`, which prohibits fragments with internal mismatches. To have GENECONV analyze sequences in the previous section with, for example, `gscale=1`, enter

```
geneconv camv.cfg camvg1 /g1
```

The long form of `/g1` is `-gscale=1`. (See options for determining fragment lists below.) We use a different name for the output and log files here so as not to overwrite previous output.

With `-gscale=1`, GENECONV finds 6 globally significant inner fragments as opposed to 4 before with the default `gscale=0`. The part of the output file `camvg1.frags` with the 6 global fragments is

```
#   Seq     Sim    BC KA    Aligned Offsets   Num  Num  Tot  MisM
#   Names  Pvalue  Pvalue   Begin  End   Len  Poly Dif  Difs Pen.
GI  CD;CS  0.0000  0.00000  7564 CA  18  565    67   4  361    3
GI  CB;C1  0.0002  0.00035  5058   7039 1982   207  25  238    4
GI  CN;CD  0.0003  0.00039  8018 CA  18  111    28   0  420    3
GI  CN;CS  0.0056  0.01302  7290 CA  26  847    99  16  334    3
GI  C1;CS  0.0244  0.05402  7290   7731  442    46   5  362    3
GI  CM;CD  0.0486  0.09692  7564   7756  193    19   0  415    3
```

Note that the number of globally significant fragments has increased from 4 to 6, and the number of highly significant global fragments (P<0.01) from 2 to 4. The most significant global fragment has 4 internal mismatches and a mismatch penalty of 3 per mismatch. The next most significant global inner fragment has 25 internal mismatches. There is a new, highly significant, fragment involving two new sequences that overlaps the endpoints. Some of the significant fragments are also longer. Other significant fragments have disappeared because they are no longer significant once mismatch penalties are allowed.

FANCIER OUTPUT

Fragments in some alignments can have Bonferroni-corrected KA P-values of 10^{-36} or smaller (Padidam et al 1999). In this case, the most significant several dozen fragments may have the same value in fragment listings for both permutation and KA values, namely the relatively uninformative `0.00000`. If you use the command-line option `-ExpFormat` in the last example, GENECONV will display KA P-values in exponential format:

```
#   Seq     Sim     BC KA    Aligned Offsets   Num  Num  Tot  MisM
#   Names  Pvalue   Pvalue   Begin  End   Len  Poly Dif  Difs Pen.
GI  CD;CS  0.0000  1.56e-06  7564 CA  18  565    67   4  361    3
GI  CB;C1  0.0002  3.48e-04  5058   7039 1982   207  25  238    4
GI  CN;CD  0.0003  3.91e-04  8018 CA  18  111    28   0  420    3
GI  CN;CS  0.0056  1.30e-02  7290 CA  26  847    99  16  334    3
GI  C1;CS  0.0244  5.40e-02  7290   7731  442    46   5  362    3
```

```
GI  CM;CD  0.0486  9.69e-02  7564   7756  193    19   0  415    3
```

This gives more information about highly significant KA P-values. If an additional two digits of accuracy in the non-exponential format would be enough, you can enter `-WideCols`:

```
#   Seq     Sim     BC KA     Aligned Offsets   Num  Num  Tot  MisM
#   Names  Pvalue   Pvalue    Begin  End   Len  Poly Dif  Difs Pen.
GI  CD;CS  0.0000  0.0000016  7564 CA  18  565    67   4  361    3
GI  CB;C1  0.0002  0.0003481  5058   7039 1982   207  25  238    4
GI  CN;CD  0.0003  0.0003912  8018 CA  18  111    28   0  420    3
GI  CN;CS  0.0056  0.0130236  7290 CA  26  847    99  16  334    3
GI  C1;CS  0.0244  0.0540173  7290   7731  442    46   5  362    3
GI  CM;CD  0.0486  0.0969171  7564   7756  193    19   0  415    3
```

The options `-ExpFormat` and `-WideCols` have no effect on the `Sim Pvalue` column. If both commands are entered, then `-ExpFormat` is used.

While the GENECONV output above gives the location of the fragment, it does so in terms of offsets in the *alignment*, which include the indels or gap characters that were introduced to align the sequences. If you enter the option `-Annotate` , GENECONV will also list the *unaligned* offsets of the fragment in both sequences (that is, NOT counting indels). GENECONV also gives the unaligned fragment lengths in both sequences, which may be different from one another and also different from the aligned fragment length. When computing unaligned offsets and lengths in non-protein alignments, GENECONV recognizes the IUB codes for ambiguous nucleotides (for example, Y for C or T, or B for one of C/G/T) but treats all other characters as gap characters.

GENECONV normally lists global fragments in order of increasing global P-values, as above. If you enter the option `-SortGfragsBySeqs` (short form: `/sg`), then the significant global fragments are listing alphabetically by sequence name. If groups were entered, global fragments are first sorted by groups. Enter `-SortNames` (short form `/sn`) to sort all group and sequence name lists alphabetically. If you enter `-SortGfragsBoth` (short form: `/sa`), then each global list appears in the output sorted by increasing P-value followed by the same list sorted alphabetically.

If the sequences in `<Seqfile>` werer truncated from a longer alignment and you want to use the original offsets, you can enter `-AddOffset=<num>`. This command will add `<num>` to all offsets in GENECONV displays. If you enter

```
geneconv camv.cfg camvg1alt /g1 /sg -AddOffset=942 -Annotate
```

then the global inner fragment list becomes (excluding the last four columns)

```
#   Seq     Sim    BC KA    Aligned Offsets         In Seq1           In Seq2
#   Names  Pvalue  Pvalue   Begin  End   Len    Begin  End   Len   Begin  End   Len
GI  C1;CB  0.0002  0.00035  6000   7981 1982    5943   7924 1982   5940   7920 1981
GI  C1;CS  0.0244  0.05402  8232   8673  442    8175   8596  422   8177   8589  413
GI  CD;CM  0.0486  0.09692  8506   8698  193    8415   8605  191   8430   8620  191
GI  CD;CN  0.0003  0.00039  8960 CA 960  111    8867 CA 960  109   8882 CA 960  109
GI  CD;CS  0.0000  0.00000  8506 CA 960  565    8415 CA 960  561   8424 CA 960  561
GI  CN;CS  0.0056  0.01302  8232 CA 968  847    8174 CA 968  825   8177 CA 968  816
```

Increasing offsets uniformly in an alignment of circular sequences may not be the most appropriate thing to do, since it is unlikely that the current and longer alignments were both circular. However, this feature can be useful for a noncircular alignment. A change in displayed offsets that amounts to a

rotation of circular sequences may be implemented in a future version of GENECONV.

The option `-Fancy` or `-Fancy_output` (short form: `/f`) sets both `-ExpFormat` and `-Annotate`. Another option `-Showbcpwkapvals` tells GENECONV to display both Bonferroni-corrected and pairwise KA P-values in all global and pairwise lists. GENECONV normally displays global KA P-values in global lists and pairwise KA P-values in pairwise lists. There is also an option `-Showlog10pvals` that displays `-log10` of the appropriate KA P-value in both lists.

SPREADSHEET OUTPUT

GENECONV normally writes output with space-separated columns, which is convenient for reading in a text editor. This output is not convenient for spreadsheet programs, however, since many columns have internal spaces. If you enter `-Dumptab`, GENECONV will generate tab-separated columns. This can be read correctly by most spreadsheet programs but will display awkwardly in a word processor. If you enter `-Dumpdif`, then the output will be a spreadsheet file in DIF format. If you have sequence names with embedded spaces or semicolons, then you may have some difficulties reading the DIF file into a spreadsheet.

If your sequence file is named `seqfile.txt`, then space-separated output will normally be called `seqfile.frags`, tab-separated output will be `seqfile.tab`, and DIF-file output `seqfile.dif`. The log file will be named `seqfile.sum`. Since an input sequence file CANNOT end with `.frags`, `.tab`, `.dif`, `.jseqs`, or `.sum` (GENECONV will exit with an error message), the output files cannot overwrite the input sequence file.

If you enter `-Dumptab` or `-Dumpdif`, then `frags` output will be suppressed unless you also enter `-Dumpfrags`. If you enter all three options, then GENECONV will write output files in all three formats. Enter `-Nodump` to suppress all three of these options (and all output files except for the log file). To suppress all output INCLUDING the log file, enter `-No_output`. If you enter `-Dumpall`, then GENECONV will write files in five different output formats.

There is a fourth output file option `-Dumpjseqs` that writes sequences adjacent to the apparent gene conversion endpoints to the file `seqfile.jseqs`. See Alignments at gene conversion endpoints below for more details.

SILENT SITES AND AMINO ACIDS

The sample input and output files on the Web site include two examples of DNA coding region alignments (`gnd7.asf`; Sawyer et al. 1987; `maz8act.aln`; Drouin et al, 1999). There is also a protein alignment (`coxprot.asf`; Sneath et al. 1975).

The option `-Seqtype=SILENT` (short form: `/r`) tells GENECONV that the alignment is from a DNA coding region and to use silent-site polymorphisms only. With this option, GENECONV uses codon polymorphisms instead of site polymorphisms, since silent sites within the same codon position are likely to be correlated. Codon positions that are amino-acid polymorphic or that have unrecognized data or indels are ignored. Amino-acid polymorphisms may have been the result of strong selection and recent mutation and may be clustered in the alignment. If this happens, the significance of fragments

elsewhere would be artificially enhanced. In any event, amino-acid polymorphisms are rare in most coding-region alignments.

If `Seqtype=SILENT`, GENECONV normally preserves classes of silent codon polymorphisms with the same degeneracy when it permutes codon polymorphisms (Sawyer 1989; the classes are 4-fold, 3-fold, 2-fold, and irregular degeneracy). See options for permutations below for options for customizing how permutations are done. The option `-Mitcodes` (short form: `/dv`) tells GENECONV to use the (mammalian) mitochondrial DNA codes for amino acids to determine silent polymorphisms instead of the standard nuclear coding set. This option has no effect unless `Seqtype=SILENT`.

The alignment `gnd7.asf` has 7 sequences from the *gnd* locus in the bacterium *Escherichia coli*. The *gnd* locus transcribes the enzyme 6-phosphogluconate dehydrogenase. Sawyer (1989) found a significant pattern of gene conversion at this locus, although few events were individually significant. With the options

```
geneconv gnd7.asf /r /w123  /lp
```

GENECONV finds two significant global inner fragments (P<0.05), both with P<0.03. Only one of these fragments is significant (P<0.05) using the global MCF P-values of Sawyer (1989). (See Comparison with previous programs: VTDIST and VTDIST3). In addition, GENECONV finds 6 inner fragments with pairwise P-values of 0.05 or less. See the sample output file `gnd7.frags` .

The options

```
geneconv gnd7.asf gnd7g1 /g1 /r /w123  /lp
```

tell GENECONV to look for significant fragments with mismatches and mismatch penalties given by `Gscale=1`. GENECONV now finds three (3) significant global inner fragments, all with P<0.03. However, GENECONV finds 5 pairwise significant fragments as opposed to 6 with `Gscale=0` (P<0.05), so that allowing mismatches does not always guarantee finding more significant fragments.

By default, GENECONV reads all sequences into memory. It then assumes that the sequences are DNA or RNA if 85% or more of the bases are in TCAGUN or tcagun, exclusive of non-alphabetic characters, which will generally be treated as indels. Otherwise, GENECONV assumes that the input file has protein (amino-acid) sequences with the standard single-letter codes for amino acids. If you enter `-Seqtype=PROTEIN` (short form: `/p`) or `-Seqtype=NUCL` or `-Seqtype=ALL` (short form: `/a`), then GENECONV will assume that the sequences are proteins (respectively DNA/RNA) without reading in the sequences first. This will save computer memory, since GENECONV normally stores sequence data only for polymorphic sites.

The only difference between GENECONV's default behavior for DNA/RNA and for amino-acid sequences is that GENECONV assumes a larger alphabet in the latter case. With the larger alphabet, fewer bases are considered unrecognized and treated as indels. See indels and missing data below for a detailed description of how GENECONV treats unrecognized bases.

If `Seqtype=PROTEIN`, GENECONV recognizes an additional (nonstandard) amino acid code letter X. This might be used, for example, to distinguish the two codons AGC and AGT for serine from the other four codons TCA, TCC, TCG, and TCT for serine, on the grounds that a single point mutation cannot carry a serine codon from one of these groups to the other. If the letter X does not appear in a

protein alignment, then this addition to the standard amino acid codes has no effect. If `Seqtype=SILENT`, the two classes of serine codons are combined.

The sample input file `coxprot.asf` has nine protein sequences from bacteria that are thought to be too distantly related to easily undergo gene conversion. When GENECONV is run with the options

```
geneconv coxprot.asf /w123  /lp
```

the most significant global inner fragment has P=0.170. When the analysis is repeated with `Gscale=1` --- that is, with the additional command-line option `/g1` --- there is one marginally significant global inner fragment (P=0.076). See the sample output files `gnd7.frags`, `gnd7g1.frags`, `coxprot.frags`, and `coxprotg1.frags` for the details.

AN IMMUNE-SYSTEM EXAMPLE: CORRECTED SCORES CAN MAKE A DIFFERENCE

The sample input file `mhc3.nex` has an alignment of three genes from the *Mhc* immune-system complex (Klein and Shonbach 1993, Takahata 1994). Two of the sequences (S2 and S3) are relatively similar. The third sequence (S1) is more divergent. There is an apparent gene conversion event between S1 and S2.

The apparent gene conversion event is not significant for the global MCF test of Sawyer (1989; P=0.263), although the SSCF test is significant for the alignment (P=0.031; Takahata 1994; see Comparison with previous programs: VTDIST and VTDIST3). Both MCF and SSCF compare fragments across sequence pairs by means of uncorrected fragment lengths (measured as numbers of polymorphic sites). Takahata (1994) applies a runs test for polymorphic sites in this alignment and finds that the set of fragments between sequences S1 and S2 are highly significant (P=0.003) and are also highly significant between S2 and S3 (P=0.004).

The MCF test compares uncorrected polymorphism lengths of fragments across all pairs of sequences. As mentioned earlier (Assessing Significance), this can cause a bias against fragments between relatively distant sequences, since these have a higher number of distinguishing sites. This alignment is an excellent example of this phenomenon.

Instead, GENECONV uses weighted global scores to correct for different numbers of distinguishing sites between pairs of sequences. When GENECONV is applied to this alignment by entering

```
geneconv mhc3.nex /w123  /lp
```

then GENECONV finds one globally significant inner fragment (P=0.0011). This is the fragment identified by Takahata (1994). This fragment is also the only pairwise significant inner fragment, also with P=0.0011. If the MCF score is used, the most significant global fragment has P=0.263.

If the option `/sb` (long form: `-ShowBlast`) is entered on the command line, then GENECONV writes the formulas for the BLAST-like weighted global scores to the output file. This results in the following table being written to `mhc3.frags` for global inner fragments:

```
# BLAST-like scores (GI) used in global comparisons (gscale=0):
#   S1;S2   BLAST = 1.2321*NPOLYS + 2.8332    MaxNpolys= 6
#   S1;S3   BLAST = 3.1781*NPOLYS + 3.1355    MaxNpolys= 1
#   S2;S3   BLAST = 0.4055*NPOLYS + 2.0794    MaxNpolys= 6
#
# where NPOLYS is the number of polymorphisms in the fragment.
```

Both the distantly related pair `S1;S2` and the close pair `S2;S3` have fragments with 6 consecutive polymorphic sites. The sequence pair `S1;S2` differs overall at 17 sites and `S2;S3` at 8 sites. The uncorrected fragment-length score is 6 for both fragments, but a run of 6 concordant sites out of 8 sites is not significant for `S2;S3`. Thus neither fragment can be significant for the uncorrected MCF score.

In contrast, the global score (calculated from the table above for NPOLYS=6) is 10.226 for the `S1;S2` fragment and 4.512 for `S2;S3`. This is enough to make the `S1;S2` fragment highly significant (P=0.0011).

When the same alignment is analyzed allowing mismatches by entering

```
geneconv mhc3.nex mhc3g1 /g1 /sb /w123  /lp
```

then the global score table written to `mhc3g1.frags` is

```
# BLAST-like scores (GI) used in global comparisons (gscale=1):
#   S1;S2   BLAST = 1.1600*SCORE + 2.5677    MaxScore= 6
#   S1;S3   BLAST = 3.1764*SCORE + 3.1304    MaxScore= 1
#   S2;S3   BLAST = 0.2101*SCORE + 0.2766    MaxScore= 6
#
# where SCORE = (npolys-ndifs)*1 + ndifs*(-mismatpen)
```

The long fragment between `S1` and `S2` has the same global and pairwise permutation P-values as with `gscale=0` (P=0.0011). With `gscale=1`, the global scores of a fragment with 6 matches and no mismatches are now 9.528 for `S1;S2` and 1.537 for `S2;S3`. This places an even greater premium on fragments between the more diverse sequence pair `S1;S2`.

The runs test of Takahata (1994) and the fragment tests are not directly comparable, since the runs tests looks at the distribution of all fragment lengths and GENECONV's fragment scores look at particular fragments only. Takahata's (1994) highly significant runs-test result between sequences S2 and S3 (P=0.004) is caused by four fragments, of which two are of length 6. These four fragments can be displayed by entering

```
geneconv mhc3.nex mhc3all -nomaxpval -minscore=1 -minnpoly=1 /w123  /lp
```

but none of the four fragments individually have a pairwise permutation P-value below 0.500. By default, GENECONV ignores fragments whose pairwise score or polymorphism length is less than 2. The program options above relax these constraints. There are a total of seven fragments with these options. See Fragment lists: Other restrictions or the Options List below for more detail about options for controlling fragment lists.

In particular, using GENECONV's BLAST-like global scores instead of the uncorrected number of polymorphic sites can make a difference, in this case global P=0.0011 versus global P=0.263.

The KA (Karlin-Altschul) P-value for the most significant fragment is 0.0104 and the Bonferroni-corrected KA P-value is 0.0312. Here the global permutation P-value is more significant than

the global KA P-value by a factor of 30.

WHEN GLOBAL (BLAST-LIKE) SCORES ARE NOT ENOUGH

In most cases, for permutation P-values, BLAST-like global scores have more power for detecting fragments than Bonferroni-corrected pairwise P-values. However, in some cases, Bonferroni-corrected pairwise permutation P-values are more significant. When this happens and the difference is significant, GENECONV lists the fragments in a special block after the global list in output files.

The sample input file `maz8act.aln` is an alignment from the coding regions of 8 actin genes from `Zea mays` (Drouin et al. 1999). The alignment has 336 codon positions (1008bp), of which 214 are amino-acid monomorphic. Moniz de Sa and Drouin (1996) found evidence from both the alignments and adjacent introns of two gene conversion events. The first event is at the first 875bp of the alignment between sequences Maz56 and Maz63. The second event is at the last 130bp between another pair of sequences (Maz56 and Maz81). The input

```
geneconv maz8act.aln -SeqType=SILENT /g1 -listbest /w123
```

tells GENECONV to analyze this alignment by permuting silent codon positions with mismatch penalties set by `Gscale=1`. (See Silent sites and amino acids and Mismatch penalties.) The command `-listbest` tells GENECONV to list the 8 most significant global fragments regardless of P-value. The resulting global output in `maz8act.frags` is

```
#   Seq           Sim    BC KA    Aligned Offsets   Num  Num  Tot  MisM
#   Names        Pvalue  Pvalue   Begin  End   Len  Poly Dif  Difs Pen.
GI  Maz81;Maz56  0.0000  0.00041   874   1008  135    27   0  103    3
GI  Maz56;Maz63  0.1649  0.56483     1    879  879   187   3   20   11
GI  Maz95;Maz89  0.3121  > 1.0     271    444  174    40  10  126    2
GI  Maz83;Maz56  0.6474  > 1.0     859   1008  150    31   5  106    3
GI  Maz56;Maz89  0.9555  > 1.0     655    867  213    40   4   63    4
GI  Maz87;Maz89  0.9621  > 1.0     289    321   33     7   0  133    2
GI  Maz81;Mac1   0.9948  > 1.0     163    264  102    15   2  114    2
GI  Maz87;Maz63  0.9998  > 1.0     283    309   27     6   0  139    2
#
# 8 inner fragments listed.
# 895 overlapping fragments discarded.
#
# ADDITIONAL PAIRWISE FRAGMENTS with BC Pairwise SimPval < 0.05
#   OR LISTED GLOBAL FRAGMENTS with significantly better BC SimPval
# Here BC SimPval is the Pairwise Sim Pvalue multiplied by 28.
# (Three pairwise fragments considered per pair.)
#
#   Seq          BC Sim  BC KA    Aligned Offsets   Num  Num  Tot  MisM
#   Names        Pvalue  Pvalue   Begin  End   Len  Poly Dif  Difs Pen.
AI  Maz56;Maz63  0.0028  0.56483     1    879  879   187   3   20   11
#
# One inner fragment listed.
```

The second fragment listed is between sequences `Maz56` and `Max63` at offsets 1-879 and has 187 silent polymorphic codon positions with three mismatched silent codons. The remaining bases in the range 1-879 are in amino-acid polymorphic codon positions. The final codon position is amino-acid polymorphic, so that the fragment base range is actually 1-876. If `Seqtype=SILENT`, GENECONV treats monomorphic and amino-acid polymorphic codon positions in the same way.

This fragment is highly significant as measured by Bonferroni-corrected pairwise permutation P-values, but is not signficicant using either permuted global BLAST-like scores nor Bonferonni-corrected Karlin-Altschul P-values.

If the alignment is analyzed using all polymorphic sites, then the second fragment shrinks to offsets 117-876 and has a BLAST-like global permutation P-value of P=0.0328. The Bonferroni-corrected pairwise permutation P-value is still P=0.0028.

If you enter `-bcsim` (no short form), then GENECONV will ignore its BLAST-like global scores and use Bonferroni-corrected pairwise P-values for permutation as well as for KA P-values. If `-bcsim` is added to the command line in the last example (using all polymorphic sites), the first few lines of the inner fragment global list become

```
#   Seq          BC Sim  BC KA    Aligned Offsets   Num  Num  Tot  MisM
#   Names        Pvalue  Pvalue   Begin  End   Len  Poly Dif  Difs Pen.
GI  Maz81;Maz56  0.0028  0.00001   878   1008  131    43   1  164    3
GI  Maz56;Maz63  0.0028  0.10375   117    876  760   280   5   33   12
GI  Maz83;Maz89  0.2912  0.41592   181    218   38    16   0  173    3
GI  Maz56;Maz89  0.3724  > 1.0     655    869  215    74   7   95    4
```

ALIGNMENTS AT GENE CONVERSION ENDPOINTS

GENECONV gives you the option of writing an output file that displays the bases immediately adjacent to the endpoints of the significant fragments. This may be useful for finding motifs for gene conversion entry points.

Specifically, the command line or configuration file option `-Dumpjseqs` tells GENECONV to write a separate output file with extension `.jseqs` . This file displays 10 bases before and after each fragment endpoint in all fragment lists in the output. (This differs from Version 1.01 of GENECONV, which listed only significant pairwise fragments. Here ``jseqs" stands for ``junction sequences.")

For example, recall that the configuration file `camv.cfg` that we used for the Cauliflower mosaic virus alignment `camv9.asf` included the option `DumpJseqs`. The command

```
geneconv camv.cfg camvg0
```

writes an output file `camvg0.frags` with 4 significant global inner fragments (P<0.05; see Cauliflower mosaic virus above). The bases immediately adjacent to the endpoints of both sets of fragments are written to the file `camvg0.jseqs` . The file `camvg0.jseqs` contains the following output for the 4 global fragments:

```
# Global inner fragments (8110 aligned bases):
# (Sites with indels in both sequences are ignored.)
# Display:  Before <Frag...ment> After
#
GI  CN v CD (8018,CA18, len=111, Sim Pvalue=0.0000):
  CN:  TATACTATAA <GCTAAGGGAA....AGCCATGAAT> CGGTTTAAAG
  CD:  TATACTATAT <GCTAAGGGAA....AGCCATGAAT> AGGTCTATGA
GI  CD v CS (7564,  7963, len=400, Sim Pvalue=0.0000):
  CD:  ACATTTCCAT <AATAATGTGT....AGGCCCTGTG> TAAGGTAAGA
  CS:  AT******** <AATAATGTGT....AGGCCCTGTG> CAAGGTAAGA
```

```
GI  CD v CS (6734,  6943, len=210, Sim Pvalue=0.0157):
  CD:  AGGACTCATT <AAGACGATCT....TAAAAAGGTA> ATTCCTACAG
  CS:  AGGTCTCATC <AAGACGATCT....TAAAAAGGTA> GTTCCCACTG
GI  CM v CD (7564,  7756, len=193, Sim Pvalue=0.0165):
  CM:  TTTTCTCCGT <AATAATGTGT....AGATCTTTGT> CGTGAATATA
  CD:  CATT*TCCAT <AATAATGTGT....AGATCTTTGT> GGTGAATATA
```

For each significant fragment, the first line begins with `GI` (for ``Global Inner'' fragment), then lists the sequence names, then the aligned offsets (beginning and end), the aligned length, and the global P-value for that fragment. The next two lines display 10 bases before and 10 bases after the endpoints of the fragments, excluding indels. Here < denotes the left endpoint of the fragment and > the right endpoint.

The file also has information about 3 significant global outer sequence fragments (P<0.05), each with P<0.01:

```
# Global outer-sequence fragments (8110 aligned bases):
# (Sites with indels are ignored.)
# Display:  Before <Frag...ment> After
#
GO  CJ (7564,  7602, len=39, Sim Pvalue=0.0000):
  CJ:  TTTTCTCCAT <AAATAATGTG....GGAAATTAGG> GTTCTTATAG
GO  CN (6755,  6819, len=65, Sim Pvalue=0.0002):
  CN:  CCCGAGTAAT <AATCTCCAGG....TAGGACCTAA> CTGCATCAAG
GO  CS (6139,  6191, len=53, Sim Pvalue=0.0052):
  CS:  TAAAGCCATC <GGACTTCTTA....CCTGAACCTA> GCAGTTCAGT
```

Five of the 7 globally significant fragments begin with either `AAT` or `AAG` . This may indicate a tendency for gene conversion fragments to insert preferentially at `AA` sequences in the CaMV chromosome. Chenault and Melcher (1994) remark that many observed junction sequences in this alignment resemble junctions observed in experimentally generated recombinants or initiation sites for DNA or RNA reverse transcription.

The number of bases displayed before and after the fragment endpoints can be customized by the option `-jseqlen=<num>`. The default value is `jseqlen=10`.

3. MORE ABOUT USING GENECONV

INDELS AND MISSING DATA

When reading sequence data, GENECONV ignores digits (0-9), white space characters, parentheses (<>()[]{}), and quote characters ("`'). Otherwise, GENECONV does not distinguish among indels, unrecognized characters, and missing data.

GENECONV has three options for handling indels or missing data. The default behavior gathers together runs of contiguous sites in which each site has at least one indel or unrecognized character. Maximum runs of such sites are treated as a single polymorphism. The states of the polymorphism are runs of sites, so that the polymorphism may be very fine for a long run.

If you enter `-Skip_indels` or `-No_indels` (short form `/es`), then GENECONV ignores all sites with missing data.

If you enter `-Use_individual_indels`, then GENCONV will use all polymorphic sites with unrecognized bases or indels as polymorphisms. The last option should NEVER be used for alignments with indels, since it can introduce spurious evidence for gene conversion.

The options `-Indel_blocs` (short form `/eb`) and `-Indel_runs` (`/er`) restore GENECONV's default behavior. This option and `-Skip_indels` provides broadly similar results, with `-Indel_runs` having slightly greater power in some cases due to the slightly larger number of polymorphisms. The option `-Use_individual_indels` can lead to clusters of polymorphic sites due to alignment artifacts. These clusters make fragments elsewhere in the alignment appear to be more significant, and can definitely cause fragments to be highly significant that would not be significant under the other two options.

Codon positions with missing data are ignored if `Seqtype=SILENT` . This is effectively the same as if you had entered `-Skip_indels` .

When analyzing nucleotide data, GENECONV recognizes the characters `TCAGU` and `tcagu` as nucleotides and generally treats all other letters as indels. Bases are converted to upper case internally and bases `U` are converted to `T` internally. If you enter `-Use_iubcodes` or `-Use_iub`, then GENECONV recognizes the standard IUB codes for ambiguous nucleotides (for example, Y for C or T, or B for one of C/G/T) but treats them as if they were different nucleotides. That is, GENECONV does not consider the possibility that a Y in one sequence might match a C or T in another sequence. However, if `-Use_iubcodes` is set, it does not consider Y as an indel that might form part of a run of sites with indels.

Distinct unrecognized characters contribute to polymorphisms. That is, distinct unrecognized characters make polymorphisms finer. In particular, coding all unrecognized characters as the same symbol can produce different fragments.

OUTER FRAGMENTS

The *inner fragments* defined earlier (see finding gene conversion events and Inner and outer fragments in the Introduction) are possible indications of gene conversion events between two sequences in the alignment, or else between ancestors of two sequences in the alignment. GENECONV can also detect events in which the source cannot be identified, either because its ancestor is not represented in the sample, or perhaps because the source segment has later become so degraded that it is no longer detectable. *Outer fragments* (as opposed to inner fragments) are suggestive of these events.

GENECONV can be asked to look for three different types of outer fragments. These are called *outer sequence* fragments, *outer pair* fragments, and *outer group* fragments, respectively. Outer group fragments are only defined when the alignment has a group structure (see balanced and unbalanced groups above). We now describe the three types of outer fragments.

OUTER SEQUENCE FRAGMENTS: If mismatches within fragments are prohibited, an outer sequence fragment is a maximal segment in which a single sequence has a unique base at all polymorphic sites in the alignment. The sequence is allowed to agree with the other sequences at sites that are monomorphic in the alignment (which may in fact be fixed in the species or genus), but must have a unique base at each polymorphic site. In particular, an outer sequence fragment depends on a single sequence rather than on a pair of sequences. These fragments are called ``outer fragments'' in

Sawyer (1989) and in the programs VTDIST and VTDIST3.

Fragment scores, mismatch penalties, and KA P-values are defined for outer sequence fragments analogously to inner fragments. For outer sequence fragments, ``matches" (score=+1) are polymorphic sites at which the given sequence has a unique base. ``Mismatches" (with a negative score) are polymorphic sites at which the sequence agrees with at least one other sequence in the alignment. ``Pairwise" and ``pairwise lists" for outer sequence fragments mean for a single sequence rather than for a pair of sequences. (We use the same terms, even though there are no ``pairs".) We use the same permutations to define permutation P-values as for inner fragments.

If mismatches are allowed, mismatch penalties are defined as

```
MM = Mismatch penalty = Npoly * Gscale / Ndiff
```

rounded up to the nearest integer, where `gscale` is the gscale value and ``Npoly" is the number of polymorphic sites. However, ``Ndiff" is now the total number of polymorphic sites at which the given sequence agrees with at least one other sequence in the alignment.

As one might expect, there tend to be few long outer sequence fragments in alignments with large numbers of sequences. However, highly significant outer sequence fragments are not uncommon in the sample output files, which have up to 14 sequences.

If the alignment has a group structure (either balanced or unbalanced), the definition is modified so that, for a ``match", the given sequence has a unique base at that site only within the group that contains that sequence. With this definition, outer sequence fragments should be sensitive to gene conversion from outside that group as well as from outside the alignment, A large alignment with group structure is more likely to have significant outer sequence fragments than without group structure.

OUTER PAIR FRAGMENTS: Without mismatches, an outer pair fragment is a maximal segment in which a pair of sequences differ at all polymorphic sites in the alignment. The fragment is bounded by either two sites at which the two sequences agree, or else by one such site and an end of the alignment. In a sense, an outer pair fragment is a mirror image of an inner fragment. A long outer pair fragment between an otherwise similar pair of sequences suggests a gene conversion event from a third sequence, which may or may not be represented in the alignment.

Fragment scores, mismatch penalties, and KA P-values are defined for outer pair fragments analogously to inner fragments except that ``matches" (score=+1) are polymorphic sites at which the given sequences differ and ``mismatches" (with a negative score) are polymorphic sites at which the two sequence agree. ``Pairwise" and ``pairwise lists" refer to pairs of sequences, rather than single sequences as for outer sequence fragments. In the definition of mismatch penalty above, ``Ndiff" for outer pair fragments is the total number of sites at which the two sequences agree.

OUTER GROUP FRAGMENTS: These are defined only if an alignment has a group structure as defined by either `-Blocsize` (`/b`) or `-Group` statements. (See balanced and unbalanced groups above.) Without mismatches, an outer group fragment is a maximal segment in which a particular group is polymorphic at all polymorphic sites in the alignment. A run of polymorphic sites within an otherwise similar group may indicate a gene conversion event from outside the group. Outer group fragments depend on a group rather than on a sequence or on a pair of sequences, so that ``pairwise" and ``pairwise lists" refer to groups rather than to single sequences or to pairs of sequences. Fragment scores, mismatch penalties, and KA P-values are defined analogously as before.

GENECONV OPTIONS FOR OUTER FRAGMENTS: By default, GENECONV looks for significant inner and outer sequence fragments, and uses permutation P-values to measure their significance (see Assessing Significance above). KA P-values are also given in the output. GENECONV uses the same set of permutations for inner and outer fragments.

To have GENECONV look for significant outer pair or outer group fragments, enter `-Outerpair` or `-Outergroup` (short forms: `/OP` or `/OG`). Enter `-Allouter` or `/OA` to have GENECONV look for all three types of outer fragments, and `-Outerseq=off` or `/ONS` to ignore outer sequence fragments. Enter `-Noouter` or `/ONA` to tell GENECONV to ignore all outer fragments. See Options for fragment types below for more options, and see Basic GENECONV syntax for the rules for turning GENECONV options on and off.

If you enter `-Outersims=off`, then GENECONV does permutations for inner fragments but relies only on KA P-values for outer fragments. This allows you to use the more accurate permutation P-values to find significant inner fragments and the computationally faster KA P-values to list outer fragments. Of course, if your sequence alignment is huge, then even KA P-values may be computationally demanding. The short form of `-Outersims=off` is `/ONI`. If you prohibit permutations by entering `-numsims=0` or `/n0`, then `-Outersims` has no effect. The options `-Outersims`, `-Outersims=on`, or `/OI` restore GENECONV's default behavior.

REMARK: Since outer fragments depend on correlated differences between sequences in an alignment, they may possibly be due to parallel evolution rather than gene conversion. Because of this, you may wish to disregard outer fragments that are shorter than some preset alignment length.

SEQUENCE FILE FORMATS

Input aligned sequence data files can be in NEXUS, Pearson/FASTA, NBRF/PIR, CLUSTAL, ASF, or PHYLIP interleaved format. See Maddison et al. (1997) in the Literature Cited section for a description of the NEXUS format. ASF formats were used in the predecessors of GENECONV. NEXUS character, data, and unaligned formats are supported. Among the sample input files, `mhc3.nex` is in NEXUS format, `rand950.fasta` is in Pearson/FASTA format, `maz8act.aln` is in CLUSTAL format, and `hfetal.asf` is in ASF listed-polymorphism format. The remaining sample input files are in ASF interleaved format.

Input files can be Microsoft, UNIX, or MacIntosh text files, regardless of the host system. Text files are read as binary files with end-of-line characters treated appropriately. Line buffers are expanded dynamically, so that there is no limit on line size.

GENECONV determines the input sequence file format as follows. If the option `-Phylip` is entered, then PHYLIP interleaved format is assumed. If `-Phylip` is not entered, GENECONV reads the first word in the file. Specifically, initial space characters or blank lines in the file are ignored, then nonspace characters are read until the next space or end-of-line character. If the first word is identical with #NEXUS (independent of case), then a NEXUS file is assumed. If the word is CLUSTAL (in upper case), then that format is assumed. If the first character of the first word is > , then Pearson/FASTA format is assumed unless the fourth character in the word is ; (semi-colon), in which case NBRF/PIR format is assumed.

If none of these conditions are met, then GENECONV tries again at offsets 128 and 256 in the file to allow for a possible MacIntosh resource fork. (This step is skipped if you enter `-Nomac` .) If still none of these tests are met, then GENECONV assumes that the sequence file is in ASF interleaved or listed-polymorphism format. See the next section for more details about the two ASF formats.

In interleaved formats, each line of sequence bases normally must begin with a sequence tag name or label. If the sequence names in different interleaved blocks do not match in order, then GENECONV halts with an error message. Sequence names are stored and displayed as they are first read but are matched in a case-insensitive manner. GENECONV does not check if different sequences use the same or different sequence tag names. However, if you use the same sequence label for every sequence, then the output may be hard to read.

If enter `Nolabels` in a NEXUS format statement, or if you enter the command line option `-Nolabels` in ASF format, then lines of sequence bases must NOT begin with sequence labels. In that case, GENECONV uses the labels `S1`, `S2`, `S3`, ... internally and in output. Similarly, GENECONV adds capital `S` to the beginning of sequence names that are entirely digits to avoid confusion with sequence offsets in output. Control characters are excised from sequence names if that is necessary.

Sequence names and other strings can include embedded spaces as long as the name, or else the part of the name that contains the spaces, is enclosed either by a pair of double quotes or else by a pair of single quotes. Doubled quote characters are escaped within quotes, so that "x = ""abc""" unquotes to x = "abc". By default, embedded spaces in sequence names are converted to underscores as the sequence names are read. In general, underscores and embedded spaces are equivalent in sequence name matches. Enter `-Use_underscores=off` (short form `/qb`) to have GENECONV keep the original sequence names with embedded spaces. Since GENECONV uses semicolons to separate sequence pairs, sequence names that have embedded semicolons (`;`) are enclosed in quotes in output.

Input sequence file names must not end with `.frags`, `.dif`, `.tab`, `.jseqs`, `.plot`, `.sum`, or `.cfg`, but are otherwise arbitrary. The first five extensions are reserved for GENECONV output files and `.sum` is for log files. If you run GENECONV with an input sequence file name that ends with one of these extensions, then GENECONV will exit with an error message. If the sequence file name ends with `.cfg`, then GENECONV will assume that it is a configuration file that contains program options. (See Configuration files.) In general, GENECONV follows the same conventions and has the same command-line and configuration-file options for all input formats, except for input file format options that appear to be inconsistent with that format.

MATCH CHARACTERS (NEXUS and ASF formats only): If the first line of data in an interleaved block or group of data begins with `!` followed by a space character, then that line is treated as a ``match'' or ``consensus'' line for that group and not as sequence data. Thereafter in that interleaved group, any occurrence of a ``matchchar'' character (defined below) in sequence data is replaced by the corresponding character in the match line. If there is no match line, then ``matchchar'' matches the first line in the group. (See the next section for an example.) Match characters are only supported in NEXUS and ASF format, since the other formats appear not to use them.

By default, there is no ``matchchar'' in NEXUS and ASF formats unless it is explicitly defined. (This is a change from previous versions of GENECONV, in which the default value of `Matchchar` was - in ASF format.) In a NEXUS file, `Matchchar` must be defined in a NEXUS `Format` statement. In ASF format, it should be defined in a ``distinguished comment'' in the `<Seqfile>`, but can be entered as a command-line or configuration-file option. (See the next section ASF sequence file formats , options for

input file format , or the sample input files `alumw.asf` and `mhc3.nex`.) If a ``gap" or ``missing" character is defined and has the same value as ``matchchar", then ``matchchar" is removed.

GENECONV ignores digits (0-9), spaces, parentheses (<>()[]{}), and quote characters ("`') in sequence data in all formats. Otherwise, GENECONV does not distinguish among unrecognized characters, missing data, and indels. GENECONV treats all characters that are not recognized --- for example, not one of the characters `TCAGU` or `tcagu` for nucleotide alignments --- as if they were indel or gap characters. If you enter `-Use_iub`, then GENECONV recognizes the IUB codes for ambiguous nucleotides (see indels and missing data above), but treats them as distinct nucleotides. Distinct unrecognized characters contribute to polymorphisms. Case is not significant in reading sequence bases, so that nucleotides can be entered as either `TCAGU` or `tcagu`. Similary, the difference between `T` and `U` is not significant in DNA alignments.

Since the input file is read after command-line options and configuration files are parsed, NEXUS and ASF commands in the input file (when they are implemented) take precedence over the corresponding options on the command line or in configuration files. Options can be set in ASF format files in ``distinguished comments". (See ASF sequence file formats below.) In NEXUS format statements, GENECONV interprets `datatype=PROTEIN` as `Seqtype=PROTEIN` and `datatype=DNA` as `Seqtype=ALL` unless you have entered `Seqtype=SILENT` earlier.

In interleaved formats, GENECONV tries to allow for input files that show the results of minor mailing program or word processor mishaps. Specifically, in interleaved data, if a sequence tag name is immediately followed by the end of a line, then the line end is treated as a space. (An exception is made if this happens within open quotes in the sequence name, which generates a fatal error.) In particular, sequence names and data can occur on alternating lines. After the first line of an interleaved group, GENECONV also allows for data lines with premature line breaks, or that appear to have the label of the next line attached to the end of the current line.

``Distinguished comments" in ASF format (see the next section), and NEXUS comments of the form `[!<Comment>]`, and written to GENECONV output files. The command-line or configuration-file option `-nocomm` stops GENECONV from these NEXUS comments to output files. Previous verions of GENECONV supported a NEXUS command `Comment` for NEXUS comments to be passed to output files. This nonstandard NEXUS command is still supported, but may not be supported in future versions of GENECONV. ASF distinguished comments are also parsed for certain command-line options.

Some final technical comments about how GENECONV reads NEXUS format files: NEXUS key words are matched in a case insensitive manner, so that ``begin", ``Begin", and ``BEGIN" are interpreted in the same way. Following the NEXUS specification, ``BeginData" does NOT match ``Begin" as a NEXUS key word. (In contrast, GENECONV key words are matched by initial strings.) Words or tokens outside of data are truncated at 2000 bytes and also at the end of a line. Thus an unbalanced quotation mark will not cause the rest of the file to be read into one token. However, an unbalanced `[` in a NEXUS file will cause the read of the file to be read as a comment. NEXUS tokens can include embedded spaces and punctuation as long as the token (or an appropriate part of the token) is enclosed by either single quotes or by double quotes.

In NEXUS format, GENECONV looks for a single set of sequences in a `Data`, `Characters`, or `Unaligned` block. After finding these sequences, it stops reading the input file at the closing NEXUS semicolon or at the end of the file. In particular, no further NEXUS commands are read after GENECONV begins reading aligned or unaligned sequence data.

ASF SEQUENCE FILE FORMAT

The sample input files `alumw.asf`, `camv9.asf`, `coxprot.asf`, `gnd7.asf`, and `hfetal.asf` are examples of ASF format sequence files. The first six lines of `alumw.asf` are

```
Primate Alu sequences
#! 7 inverted pairs from 7 primate chromosomes
#! (Maeda, Wu, and Reneke, 1988)
#! Matchchar=.   (Matches first or consensus line.)
# Number of sequences and number of bases:

14  296
```

In general, any line whose first non-blank character is one of `#;%$[` in an ASF format file is assumed to be a comment and is ignored. Blank lines are also ignored. The first noncomment (and nonblank) line in an ASF file is a description line. The second line of the file (ignoring blank lines and comments) must begin with the number of sequences, perhaps after initial spaces. If the second word on this line is a nonnegative number, it is treated as the number of bases. GENECONV needs the number of sequences, but counts the number of bases for itself. If the number of bases given is nonnegative but incorrect, then GENECONV writes a warning message to the screen and to the log file but uses its own count.

If the first character after the comment character is `!`, then the comment is written to the output file. These ``distinguished comment'' lines are also parsed for option settings. For example, the third distinguished comment line above sets the value of `Matchchar` equal to `.` (See sequence file formats or options for input file format .) The size line --- with entries `14 296` above --- is also parsed for options settings, but not the description line.

If there is a third number on the second or size line (after the numbers of sequences and bases), and if this third number is nonzero, then GENECONV assumes that the file is in ASF ``listed polymorphism'' format. This format is described in comment lines at the top of the sample input file `hfetal.asf` (which is in this format) and will not be discussed further. Otherwise, GENECONV assumes that the file is an ``interleaved'' ASF sequence file. The sample input files `alumw.asf`, `camv9.asf`, `gnd7.asf`, and `coxprot.asf` are all interleaved ASF format files.

After some further comment and blank lines, `alumw.asf` continues with

```
#          10         20         30         40         50         60
!  GGCTGGGCGT GGTGGCTCAC GCCTGTAATC CCAGCACTTT GGGAGGCCGA GGTGGGTGGA
H1 ..G..****. .......... ......G... ....T..A.. CA........ ..A.......
H2 ......AT.C .........G ..T.....A. .......... .......... ..CA..CA..
I1 ..G..****. .......... ......G... .......... CA........ ..A.......
I2 ......AT.C .........G ..T.....A. .......... ........A. ..CA..CA..
```

These are the first 6 lines of a group of 16 lines that constitutes the first interleaved group in the alignment. This group has the first 60 bases for each of the 14 aligned sequences. Here `H1`, `H2`, `I1`, and `I2` are the labels or sequence tag names for the first four sequences. These sequence tags must be present unless you enter `-Nolabels` . (See sequence file formats above or options for input file format below.) This group of 16 lines is followed by four other similar groups that contain the rest of the 296 bases in the alignment.

The . characters in the display are ``match characters", which tell GENECONV to replace this character by the corresponding character in either the ``consensus" line for that group or else the first line in the group. Note the definition in `#! Matchchar=.` in the head of `alumw.asf`. The * characters are indels, which are gaps introduced to form an alignment of the sequences.

The comment characters `#;%$[` can also be changed by program options (see options for input file format below). For example, suppose that you have an ASF sequence file that has * instead of . as the match character, that has comment lines whose first non-blank character is > or %, and that has some sequence tag names that begin with #. The last feature would normally cause the sequence data for these sequences to be read as comments and ignored. If you run GENECONV with the program options

```
-Commchars=>%   -Matchchar=*
```

then GENECONV will use >% as comment-line characters (and no other comment-line characters) and will use * as the match character. To remove `Matchchar`, replace * by a blank character or enter `-Matchchar=0`. The sequence names beginning with # will now be read correctly. All . characters in sequence data in the input file will now be unrecognized and hence treated as indel or gap characters.

Since you are unlikely to want to run GENECONV on the same file with several different `Commchar` and `Matchchar` settings, it is safer to put these settings in a ``distinguished comment" near the top of the file. Specifically, enter

```
#! Commchars=>%   and   Matchchar=*
```

near the top of the file. This line will not only be passed onto the output file, but will also change the settings of these two options. Some other examples of distinguished comments that also change parameter settings are

```
#! This file should be analyzed with Seqtype=PROTEIN
#! There are also no sequence labels: Nolabels
```

Parsed commands in a distinguished comment line must be preceded by a space unless they are at the beginning of the line, so that (for example) `Wasseqtype=ALL` would not be recognized. Distinguished comment lines are parsed for changes to `Commchar`, `Seqtype`, `Matchchar`, `Gapchar`, `Misschar`, `Numbersep`, and `Nolabel` options, but for no other variables. The size line is also parsed for these options. (See also Silent sites and amino acids and options for input file format below.)

After GENECONV has begun reading sequence or consensus sequence data, distinguished comments are neither saved for output files nor parsed for options settings.

On the command line or in configuration files, the short forms for `-Commchar=>%` and `Matchchar=*` are `/$>%` and `/!*`. The string of new comment characters can be enclosed in matching " or ' characters, but this is not required. The string in the short-form expression `/$...` should not be immediately followed by other short-form options. That is, enter `/$>% /n0` and not `/$>%n0`.

Since the input sequence file is read after all command-line variables are parsed, settings in distinguished comments in the sequence file take precedence over command line and configuration file settings.

SEARCH PATHS: FINDING FILES IN OTHER DIRECTORIES

If you enter input or configuration file names without an explicit directory path, as in

```
geneconv alumw.asf doalu.cfg
```

then GENECONV looks for `alumw.asf` and `doalu.cfg` in the computer default directory (or folder). You can give GENECONV a list of other directories in which to look for input sequence files and configuration files that are not in the default directory. This could be useful if you want to keep input sequence files in one directory while writing output files elsewhere, or in general if you want GENECONV to fetch input sequence files or configuration files from some directory other than the default directory.

For example, if you enter (assuming Microsoft or DOS path name conventions)

```
geneconv alumw.asf -inputpath=c:\data\aluseqs;e:\myseqs;a:
```

then GENECONV will look for `alumw.asf` first in the default directory, then in `c:\data\aluseqs`, then in `e:\myseqs`, and then on drive `a:` . The output file will say where GENECONV actually found the version of `alumw.asf` that it used. Similarly,

```
geneconv -inputpath=c:\data\mostconfigs;a: alusamp.cfg
```

tells GENECONV to look for the configuration file `alusamp.cfg` first in the default directory, then in `c:\data\mostconfigs`, and so forth.

Note that `-inputpath=...` was entered BEFORE `alumsamp.cfg` in the last example. This is because GENECONV reads and parses command-line options and configuration files as it encounters them. If `alusamp.cfg` came before `-inputpath=...` on the command line above, then GENECONV would not know about `-Inputpath=...` before it read `alusamp.cfg`. If `alusamp.cfg` did not exist in the default directory, then GENECONV would exit with an error message. The same problem does not arise with input sequence files since configuration files and command-line options are parsed *before* GENECONV reads the input file.

To add to a current `Inputpath`, enter `-Add_inputpath=...` . To remove `Inputpath`, enter `-Inputpath=` with a space after the = sign. You can enter `-Filepath=...` or `-input=...` instead of `-inputpath=...` with the same effect.

If you enter an input sequence file path explicitly, as in

```
geneconv c:\myseqs\alumw.asf -inputpath=...
```

then GENECONV will not use `Inputpath` to look for that file. A single drive name, as in `geneconv c:alumw.asf -inputpath=...` , counts as a path in this sense and GENECONV will not look in `Inputpath`.

It can be useful to define a configuration file (called, for example, `fpath.cfg`) with a list of all the directories in which you might keep aligned sequence files. Then

```
geneconv fpath.cfg myseqs.bin
```

will produce output in the current directory no matter where you are in the computer system. For example, `fpath.cfg` might consist of the lines

```
#GCONV_CONFIG
    -add_filepath=c:\viruses\bin;d:\bacteria\mine
    -add_filepath=c:\sequences\molds\slime
```

Note that we used ``-add_filepath'' instead of ``-filepath'' so that several configuration files of the form ``fpath.cfg'' might be combined.

These examples assume Microsoft or DOS syntax for path names and directory lists. Disk names (such as `c:`) do not exist in UNIX, and the path name separators `\` and `;` in DOS are normally replaced by `/` and `:`. An example command line on a UNIX system is

```
geneconv alumw.asf -inputpath=~/data/aluseqs:~/myseqs:/etc/bin
```

By UNIX convention, ~ means the user's home directory.

FRAGMENT LISTS: RESTRICTING BY P-VALUE

If GENECONV were to list all fragments in its output, then the output might be huge for a large alignment. By default, GENECONV outputs global lists only and restricts global lists by P-value, which should normally keep output lists to manageable size. In addition, you can add restrictions on the length of a fragment, on its pairwise score, and have explicit bounds on the number of fragments in the lists. Enter `-ListPair` or `/lp` to have GENECONV produce pairwise lists as well (see options for determining fragment lists). We discuss conditions on P-values first.

When GENECONV does permutations, its default behavior is to include fragments in global lists if the multiple-comparison corrected (or global) permutation P-value is 0.05 or less. If pairwise lists are specified, fragments are included if the pairwise permutation P-value is 0.05 or less. When permutations are done, no restrictions on KA P-values are applied unless you enter the options explicitly. (See Assessing significance: pairwise and global P-values above for a discussion of permutation and KA P-values and for a discussion of global and pairwise fragment lists.)

When GENECONV does no permutations, fragments are included in global lists if the global (or Bonferroni-corrected) KA P-value is 0.05 or less and in pairwise lists if the KA P-value is 0.05 or less.

If permutations are done but (for example) the pairwise permutation P-value condition is removed by a program option like `/mip5` or `-MaxSimPairPval=off`, then GENECONV applies the pairwise KA P-value condition instead unless you remove pairwise KA P-value bounds by entering (for example) `-MaxKAPairPval=off` or `/mkp2`. (See options for P-value bounds in fragment lists below.)

If no permutations are done and if one of the KA P-value conditions is removed, then GENECONV applies the remaining KA P-value condition to both pairwise and global lists. That is, if you enter

```
geneconv seqfile /n0 /mkp5 /mkg0.05
```

then GENECONV includes fragments in both pairwise and global lists only if the Bonferroni-corrected P-value is 0.05 or less. Note that, without a convention of this sort, GENECONV may list millions of fragments for each pair of sequences. This ``crossover'' does not apply to permutation P-value bounds.

FRAGMENT LISTS: OTHER RESTRICTIONS

By default, GENECONV does not output lists of pairwise significant fragments. To have GENECONV include pairwise lists as well, enter `-ListPairs` (short form: `/lp`) or `-PairWise` . The options `-NoPairs` (short form: `/lp0`) or `-NoPairList` turn pairwise lists off.

GENECONV can be told to ignore fragments that do not meet conditions such as a minimum length, number of polymorphic sites, or pairwise score. By default, ignores fragments unless they have at least 2 polymorphisms and have a minimum pairwise score of 2, and lists a maximum of 2000 fragments in each fragment list. See options for determining fragment lists below for additional detail.

Enter `-minlength=<num>` or `/ml<num>` to limit the length of fragments in terms of aligned bases (including indels). If you enter `-minlength=50` or `/ml50`, then GENECONV will ignore all fragments with an aligned length of less than 50 bases. The default is `minlength=1`. (That is, there is no default restriction on aligned fragment length. Recall that, by definition, a fragment must contain at least one polymorphic site.)

Enter `-minnpoly=<num>` or `/mn<num>` to limit the length of fragments in terms of the number of polymorphic sites. If you enter `-minnpoly=10` or `/mn10`, then GENECONV will ignore fragments unless they contain at least 10 polymorphic sites. The default `minnpoly=2`, so that GENECONV does not normally consider fragments with less than 2 polymorphic sites.

Enter `-minscore=<num>` or `/ms<num>` to limit fragments in terms of their pairwise score --- that is, the number of matching polymorphic sites minus the mismatch penalty times the number of mismatches, if fragments with mismatches are allowed. If fragments with mismatches are not allowed, the score is the same as the number of polymorphic sites. If you enter `-minscore=30` or `/ms30`, then GENECONV will ignore fragments with a pairwise score less than 30. The default `minscore=2`, so that GENECONV does not normally consider fragments with score than 2.

Enter `-listglobal=<num>` or `/lg<num>` to tell GENECONV to list at most `<num>` fragments in any global list. If you enter `-listglobal=50` or `/lg50`, then GENECONV will limit the global list for each fragment type to 50 fragments or less. The default value is `-listglobal=2000`.

Enter `-listperpair=<num>` or `/lp<num>` to tell GENECONV to list at most `<num>` fragments for any particular pair of sequences and fragment type. As a side effect, pairwise lists are turned on if `num>0` . Thus, if you enter `-listperpair=20` or `/lp20`, then GENECONV will display pairwise lists but limit all pairwise lists to the 20 highest-scoring fragments or less. The limit is applied independently for all sequence pairs. The particular commands `-listperpair=0` (`lp0`) and `-listperpair=off` are equivalent to `-NoPairLists` and do not affect GENECONV's internal `listperpair` setting. The default value is `-listperpair=2000` .

The command `-listbest` or `/lb` has the same effect as entering `-listglobal=8` , `-listperpair=3` , and `-Nomaxpval` , except that pairwise lists are not specified. That is, you have to enter `-PairLists` or `/lp` or the equivalent to get pairwise lists.

The option `-Maxoverlap=<num>` controls the maximum number of overlapping fragments for each sequence pair. The default is one fragment, so that overlapping fragments for the same sequence pair are normally excluded. See Overlapping fragments for the details.

FRAGMENT LISTS: IF COMPUTER MEMORY IS A PROBLEM

Large alignments have the potential of producing a huge number of fragments. Some fragment list restrictions control limit the size of output files, but GENECONV may still have to manage large fragment lists internally. If the only P-value restrictions are on simulated P-values, then GENECONV may have to keep track of large fragment lists since it will only know the permutation P-values after it does the permutations. Restrictions on KA Pvalues, fragment lengths, and score are applied as GENECONV builds fragment lists, and so may help to reduce memory requirements.

Suppose that one is interested in the most significant nonoverlapping global fragments with a mismatch penalty. (GENECONV's default behavior is to list at most one fragment in an overlapping group of fragments for the same sequence pair; see Overlapping fragments below.) In a large alignment, it is possible that the 3000 most significant fragments for a particular pair of sequences overlap the fragment of highest significance, but that there are other highly significant fragments elsewhere in the alignment. The global KA P-values of these fragments might vary from 10^{-47} to 10^{-10}. In this case, GENECONV would have to keep track of more than 3000 fragments to find the second most-significant nonoverlapping fragment for that sequence pair.

GENECONV expands its fragment buffers automatically when it needs to, so that you will normally not have to worry about these problems. However, if your computer shows signs of running short of memory, then it may help to impose conditions on fragment length or to set lower KA P-value bounds.

INCLUDING MONOMORPHIC SITES IN POLYMORPHISM LISTS: FRAGMENTS WITH TWO SEQUENCES

GENECONV normally excludes sites that are monomorphic in the alignment before performing permutations and computing fragment scores. (See finding gene conversion events: fragments and HSAPs above.) This is an important control for sites in the alignment that might be held fixed by selection or a locally reduced mutation rate. If fixed sites have a nonuniform distribution along the alignment and are permuted along with the polymorphic sites, then they can cause fragments containing them to be artificially significant. If the definition of fragment score is modified so that sites that are monomorphic in the alignment are scored as 0 instead of 1, then it would not matter whether or not monomorphic sites were permuted along with the polymorphic sites.

Excluding monomorphic sites means that GENECONV cannot be used with an alignment of two sequences. If there are only two sequences, then all polymorphic sites distinguish the two sequences, so that all polymorphic sites are mismatches in all fragments. If mismatches are not allowed, this means that they would be no inner fragments (which have to have positive length). Similarly, the entire alignment is composed of bases that are unique at each polymorphic site. This means that there are no outer sequence fragments either, since these have to be less than the entire alignment. (See Outer

fragments.)

If you enter the option `-Include_monosites`, then GENECONV includes monomorphic sites in its internal lists of polymorphisms and, effectively, scores monomorphic sites as +1 in fragments instead of 0. If monomorphic sites are uniformly distributed in the alignment in the absence of gene conversion, then GENECONV would be able to find statistically significant apparent gene conversion events from an alignment of two sequences. Note, however, that gene conversion might have eliminated statistical evidence for clustered monomorphic sites. Thus the option `-Include_monosites` should be used only with care. (See options for determining fragment lists below.)

OVERLAPPING FRAGMENTS

If mismatches in fragments are not allowed, then, by definition, fragments are maximal runs of polymorphic sites (in the alignment) that agree between two sequences. Always, fragments are bounded by either two mismatches between the sequences or else a mismatch and an end of the alignment. (See finding gene conversion events above or Sawyer 1989.) Thus fragments for the same two sequences cannot overlap if mismatches are not permitted. This no longer holds if mismatches are allowed. A high-scoring fragment may be extended to a longer fragment with a lower score by adding a mismatch and a run of matches, and then to a fragment with a higher score yet by adding another mismatch and a longer run of matches. Without a convention for limiting these overlaps, an alignment for a positive gscale may have millions of significant overlapping fragments.

How GENECONV handles this problem is as follows. All fragments (overlapping or not) are first sorted by score in each pairwise list. If a variable `Maxoverlap` is positive, then GENECONV remembers the first fragment, and keeps the `Maxoverlap` highest-scoring fragments in the pairwise list that overlap that fragment. By default `Maxoverlap=1`, which means that only the first fragment in the overlapping group is kept. If a fragment is found that does not overlap the first fragment, then it is also kept and acts as a screen in the same way. Only the first fragment in each overlapping group acts as a screen. (In this sense, ``overlapping groups'' are defined by their first fragment.) If `Maxoverlap=1`, this results in a nonoverlapping list of fragments that may contain only a small portion of the original list. Global lists are built up from pairwise lists that have already been screened for overlaps, so that there is no censoring of overlapping fragments for different sequence pairs.

When the pairwise lists are first sorted, fragments with the longest aligned length are chosen first from groups with the same score. This means that, if `Maxoverlap=1` and if there are ties for highest score in an overlapping group of fragments, then GENECONV chooses the longest fragment. (In computer terminology, the primary sort key is score and the secondary sort key is aligned length. There is also a tertiary sort on the number of mismatches and a quaternary sort on the starting endpoint.) This is done because of the author's suspicion that, in many cases, the true `gscale` may be less than one and mismatch penalties with `gscale=1` may be too large. This tie-breaking scheme represents a change from Version 1.02 of GENECONV, in which ties by score in pairwise lists were broken by choosing the fragment with the leftmost starting position and then the leftmost endpoint.

To have GENECONV list up to 3 overlapping fragments for each high-scoring fragment --- any of which might have been the actual gene conversion event --- enter `-Maxoverlap=3` (short form: `/v3`). To remove screening for overlaps altogether, enter `-Maxoverlap=0`, `-Maxoverlap=none`, or `/v0`. If `-Gscale=1` and the alignment is complex, this may result in a very large output file.

DEFINING GROUPS, SKIPPING SEQUENCES, AND LISTING SEQUENCES

The option -Seq_skip=<Namelist> tells GENECONV to ignore the sequences in <Namelist> when it reads an input sequence file. The similar command -Seq_list=<Namelist> tells GENECONV to consider ONLY those sequences. The syntax for <Namelist> is slightly different on the command line than in a configuration file (see below). -Skip_list can be used as a synonym for -Seq_skip , and -Subset can be used in place of -Seq_list .

The similar command -List_only=<SeqPairList> tells GENECONV to use the same input file sequences as it would otherwise, but only include fragments in output files for the sequence pairs in <SeqPairList>. The principal difference between -List_only and -Seq_list is that the nonlisted sequences are used to define polymorphic sites for -List_only but not for -Seq_list . The -List_only command can be useful for obtaining a quick idea of what is happening between two sequences in a sequence environment in which there would otherwise be a large amount of output. -Listonly=... can be used as a synonym for -List_only=....

Group (-Group) commands are used to define groups when GENECONV is told to look for within-group gene conversion. Group commands can only be used in configuration file. (See Groups of unequal size above and Syntax of group commands below.)

NOTE: If your input file has sequence names that are pure numbers (such as ``33''), use these names (``33'') to refer to them in GENECONV options even though they will be displayed with a prepended S (``S33'') in most GENECONV output.

The syntax for using these commands is as follows:

SEQ_SKIP and SEQ_LIST:
SYNTAX ON THE COMMAND LINE: In this case <Namelist> is a comma-separated list of sequence names, for example -Seq_skip=Random,Dull or -Seq_skip=Seq1,NotThis,33 . Sequence name matching is case insensitive, so that -Seq_skip=RANDOM will also prohibit a sequence named Random. If GENECONV reads a <Namelist> with embedded commas within quoted sequence names, then it will interpret it correctly, but the operating system may strip off the quotes on command-line arguments before sending command-line strings to GENECONV.

Enter -Add_seq_skip=<MoreNames> to add sequences to a current Seq_skip list. Enter -Seq_skip= (with a blank space after the = sign) to remove the current Seq_skip list.

Similarly, -Seq_list=<Namelist> tells GENECONV to consider ONLY those sequences, -Add_seq_list=<MoreNames> adds more names to the list, and -Seq_list= removes the list, so that GENECONV will again consider all sequences in the input file.

If a Seq_skip'd sequence is not found in your input file, then GENECONV will just assume that you wanted to be safe. However, GENECONV will normally exit with an error message if a sequence in a Seq_list command is not found in your input file. Enter -Notfound_nonfatal to tell GENECONV not to exit in that case. GENECONV will then proceed with the Seq_list'd sequences that it could find.

SYNTAX IN A CONFIGURATION FILE (SEQ_SKIP/SEQ_LIST):
If -Seq_skip=<Namelist> or -Seq_list=<Namelist> is entered in a configuration file, then

<Namelist> is assumed to be a space-separated list, for example

```
-Seq_skip=Random Dull "No Way" A7,B1
```

The initial = is optional, so that -Seq_skip Random Dull ... has the same effect. Sequence names can be enclosed in quotes. In this example, the sequence name No Way has an embedded space and A7,B1 has an embedded comma.

In a configuration file, <Namelist> is terminated at (i) any word that begins with / or - , which GENECONV treats as either a short-form or verbose-form option, (ii) at any word that ends in .cfg, which GENECONV treats as the name of a configuration file, or (iii) at the end of the configuration file. GENECONV reads // as a short-form command that does absolutely nothing, so that // can be used to end <Namelist> .

The options -Seq_skip and -Seq_list CANNOT be used within an -Options environment in a configuration file. Otherwise, the plausible-looking command in an options environment

```
Seq_list=Random Dull  Nolog  Endoptions
```

would cause GENECONV to exit with an error message that it could not find sequences named ``Nolog" and ``Endoptions" , unless you really did have sequences with names Nolog and Endoptions . (Normally, ``-Nolog" tells GENECONV to suppress writing a log file and ``Endoptions" ends an -Options environment; see Configuration files above.) In contrast, the - in -Nolog in

```
-Seq_list=Random Dull  -Nolog  -SortGfragsBySeqs
```

tells GENECONV to terminate -Seq_list at Dull .

LIST_ONLY:

SYNTAX ON THE COMMAND LINE: In this case <SeqPairList> is a SEMICOLON-separated list of pairs of sequences, where the sequence pairs are COMMA-separated. Recall that, by GENECONV convention, ``pair" means two sequences for inner and outer-pair fragments, a single sequence for outer-sequence fragments, and a group name for outer-group fragments. (See Outer fragments above.) As an example,

```
-List_only=N1,N2;BB;Random
```

tells GENECONV to list inner fragments for the sequence pair N1 and N2 (and also outer-pair fragments if you entered -Outerpair or /op), outer-sequence fragments for sequence BB and for sequence Random, and perhaps also outer-group fragments for groups with those names. Sequence names with embedded commas or semicolons cannot be used on the command line with this option.

SYNTAX IN A CONFIGURATION FILE (LIST_ONLY):

This is similar to the command-line syntax except that spaces are used instead of semicolons and the = sign after List_only is optional. As with Seq_List, <SeqPairList> is ended by any word that begins with - or / (which is then treated as a command), the name of another configuration file, or the end of the present configuration file. List_only CANNOT be used within an options environment, for the same reason as for Seq_skip and Seq_list. As an example, the following list ends just before -Nolog .

```
-Listonly N1,N2  Seq3,Seq17  BB  Random  "No Way"  -Nolog
```

```
% This is a configuration-file comment.
```

The syntax of `Listonly` has been changed from version 1.70 of GENECONV, in which the roles of commas and semicolons were switched.

SYNTAX OF GROUP COMMANDS:

`-Group` commands are used to define groups when GENECONV is told to look only for within-group gene conversion (see Groups of unequal size above). These commands CANNOT be used on the command line and MUST BE USED within a configuration file. Their syntax in a configuration file is

```
-Group <GROUPNAME> <Namelist>
```

as in (for example)

```
-Group Amphibians  Frog  Toad  "Spotted Toad"
```

There can be an optional = after `-Group` and an optional colon after `<GROUPNAME>`. Here `<Namelist>` has the same syntax as for `Seq_skip` in a configuration file (see above). In particular, `<Namelist>` is terminated with (i) any word that begins with `/` or `-` , which GENECONV then treats as another command, (ii) any word that ends in `.cfg`, which GENECONV then treats a configuration file, or (iii) the end of the configuration file. As with `Seq_list` and `List_only`, `-Group` CANNOT be used within an `-Options` environment.

Successive `-Group` commands with the same `<GROUPNAME>` add to the same group. (See groups of unequal size above for an example.) If a sequence name specified in a `-Group` command is not in your input file, then GENECONV will normally exit with an error message. In you enter `-Notfound_nonfatal`, then GENECONV will ignore these sequences.

Group names that begin with ``Gen_Poly'' are treated in a special way. These are used to add more polymorphic sites to the alignment, but GENECONV does not look for significant fragments within these groups. See groups of unequal size above.

ACKNOWLEDGMENTS

We would like to thank Claude Fauquet and Malla Padidam for many helpful contributions such as suggesting that GENECONV Version 1.00 list fragments in a spreadsheet-like format, for help in formulating the definitions of outer group and outer pair fragments, for suggesting the name GENECONV, and for providing moral support during the upgrading process that led to the current program.

This work was partially supported by NSF grant DMS-9707045.

4. LIST OF GENECONV OPTIONS

BASIC GENECONV SYNTAX

The basic command-line syntax for the program `geneconv` is

```
geneconv word1 word2 word3 ....
```

Here each ``word..." can be one of four types: (i) the name of an input sequence file, output file, or log file, (ii) a short-form option or list of short-form options, which must begin with `/`, (iii) a long-form or verbose option, which must begin with `-`, or (iv) the name of a configuration file, which must end with `.cfg`. GENECONV assumes that any argument that is not of one of the types (ii,iii,iv) is an input or output file name (i). An example with all four types in order is

```
geneconv myseqs.bin /pw123z -Circular myseqs.cfg
```

When GENECONV encounters a file name (i), it assumes that it is the input sequence file name unless the input sequence file name has already been assigned. In that case, the file name is the output file name. If an output file name already exists, it is the log file name. If a name for output files is not entered explicitly, then it is taken from the input sequence file name. If a log file name is not given explicitly, it is taken from the output file name. GENECONV will exit with an error message if no input file is entered or if more than three file names are assigned in this way.

Endings are imposed on output and log file names that indicate the file type. In particular, if the output or log file name contains a period, only the name up to the last period is used. For example, if the input file is named `myseqs.seqs` (or `myseqs.fromAfrica`), then the output file is `myseqs.frags` and the log file `myseqs.sum` unless you tell GENECONV otherwise. (See Spreadsheet output above for more details about output file types and endings.)

Note that these conventions might mean that earlier output from `myseqs.fromAsia` is overwritten if you are not careful. However, if the input file name is `myseqs_fromAfrica` (with an underscore instead of a period) or `myseqs.fromAfrica.txt`, then the output will be `myseqs_fromAfrica.frags` or `myseqs.fromAfrica.frags`. You can also set the outfile name explicitly, for example by entering `-Outfile=myseqs.fromAfrica.xxx`. The ending `xxx` will be replaced by the appropriate output or log file ending.

If GENECONV encounters either `-Config=<filename>` or a file name ending with `.cfg` , then the configuration file is expanded in place. If `<filename>` does not end with `.cfg`, then `.cfg` is appended. If a configuration file command is encountered within a configuration file, then the second configuration file is expanded in place recursively. GENECONV exits with an error message if a closed loop of configuration files would otherwise result. (There are no other restrictions on configuration file nesting.) See Configuration files above for more details.

Short-form and long-form options are case insensitive, except for the string part of options with string arguments. That is, `/F` is the same as `/f` and `numsims=100` is the same as `NumSims=100`, but, if your operating system has case-sensitive file names, then `-Outfile=MyOutPut` will produce different output than `-Outfile=myoutput` .

Long-form options are matched by minimal substrings as well as being case insensitive. That is, `-numsim=1000` is the same as `-numsims=1000`, and `-Showpoly`, `-ShowPolys`, and `-ShowPolymorphisms` have the same effect.

If `Myopt` is a long-form option that takes Yes/No values, then `-Myopt`, `-Myopt=yes`, `-Myopt=on`, and `-Myopt=true` all set `Myopt` equal to TRUE. The options `-Myopt=no`, `-Myopt=off`, and `-Myopt=false` all set `Myopt` equal to FALSE. If `Myopt` takes an integer argument, then `-Myopt=no`, `-Myopt=off`, and `-Myopt=false` all set `-Myopt=0`. Thus `-Numsims=false` or `-Numsims=No` both set `-Numsims=0`, which suppresses permutations. Minimal string matching is done here also, so that `-Myopt=nope` and `-MYOPT=NoWay!` have the same effect as `-myopt=false`.

Similar conventions hold for options that set P-value bounds. `MaxKAGlobPval=0.05` lists fragments only if they have a global Karlin-Altschul P-value of 0.05 or less, `MaxKAGlobPval=off` (or `false` or `no`) removes K.A. global bounds, and `MaxKAGlobPval=on` (or `true` or `yes`) restores the previous numerical bounds.

GENECONV OPTIONS BY CATEGORY

The options below are grouped by type rather than alphabetically to make them easier to browse. Short forms of options are included in parentheses (when a short form exists). Links are given to earlier sections for examples and a more detailed discussion.

OPTIONS FOR FILE NAMES AND OUTPUT FILES:

`-Seqfile=<filename>`	Set input sequence file name. Alternatively, the name of the input file can be listed by itself. (See the discussion in A first example and in Basic syntax.)
`-Outfile=<filename>`	Set the root for output file names. (GENECONV supplies the file extension depending on the file type.) This can also be listed by itself on the command line. (See Basic syntax.)
`-Logfile=<filename>`	Set the root for the log file name. (Basic syntax)
`-Nolog`	Do not write a log file.
`-Logappend (/z)`	If the log file already exists, append to it rather than overwrite it.
`-Dumptab`	Write tab-separated output. (Spreadsheet output)
`-Dumpdif`	Write DIF-format spreadsheet output. (Spreadsheet output)
`-Dumpfrag`	Write space-separated output. This is the default. (Spreadsheet output)
`-Dumpjseq`	Write an output file with Alignments at gene conversion endpoints.
`-Dumpall`	Write output in 5 different output formats. (Spreadsheet output)
`-Jseqlen=<jlen>`	If `-Dumpjseq` is entered, write this many bases at fragment endpoints. The default is 10 . (Alignments at gene conversion endpoints)
`-Nodump`	Do not write any output files. This can be used to clear previous `-Dumpxxx` settings, as in `-Nodump -Dumpfrags` .
`-No_output`	Do not write any output files OR a log file.

OPTION FOR BATCH MODE:

By default, GENECONV ends with a ``Press Enter..." message if there were serious errors but not otherwise. This message requires user input to continue. This allows GENECONV to be used in a

batch or script file with less chance that serious errors or missing input files early in the batch file will go unnoticed.

-Hold_Window (/x) Tell GENECONV to end with a ``Press Enter..." message even it there were no errors. This is useful if you run GENECONV with a `Run...` or `Start...` command in Microsoft Windows. Without this command, a console window will appear with screen output that abruptly disappears at a successful completion.

OPTIONS FOR READING INPUT FILES:

-Seqtype=<TYPE> (/a, /r, /p) Type of input file. Here `<TYPE>` is one of AUTO (no short form), NUCL or ALL (`/a`), SIL (`/r`), or PROT (`/p`). Initial case-insensitive matches are done, so that AUTOMATIC, NUCLEOTIDES, SILENT, and PROTEIN also work. The default is AUTO, which tells GENECONV to ``autosense" between NUCL and PROT . (See A first example and Silent sites and amino acids)

-Skip_indels (/es) Ignore all sites with any indels or missing data. Automatically done if `Seqtype=SILENT` . Synonym: `-No_indels` (Indels and missing data)

-Indel_blocs (/eb /er) Treat runs of sites, each with indels or missing data, as a single polymorphism. This is the default if `Seqtype=NUCL` or `PROT` . Ignored if `Seqtype=SILENT` . Synonyms: `-Indel_run` and `-Use_indels` (Indels and missing data)

-Use_individual_indels Treat sites with indels or missing data as individual polymorphisms. This option should be used only with extreme care, since it can cause spurious evidence for gene conversion. Ignored if `Seqtype=SILENT` (Indels and missing data)

-Matchchar=<ch> (/!<ch>) Use `<ch>` as the character that matches the first or the consensus sequence in ASF format input files. Specification in ASF `<Seqfile>` overrides this setting. (Other formats either do not support `Matchchars` or else specify `Matchchar` in other ways.) `Matchchar=0` or `Matchchar=` (followed by a blank) removes `Matchchar` . Default is no `Matchchar`. (sequence file formats)

-Gapchar=<ch>
-Misschar=<ch> Set either parameter equal to `<ch>` . If `Matchchar` is the same as `Gapchar` or `Misschar` , then `Matchchar` is removed. In Pearson/FASTA and NBRF/PIR formats, all non-alphabetical characters in sequence data other than `-`, `*`, and `Gapchar` are ignored. Since GENECONV normally treats all unrecognized characters as indels, there is no other effect of these options (sequence file formats, ASF sequence file format)

-Commchar=<str> (/$<str>) ASF format only: If any character in `<str>` is the FIRST NONBLANK CHARACTER in a line in a sequence input file, then that line is treated as a comment. The default is `#;%$[` . Some comments in ASF files are parsed for options settings. See ASF sequence file format

-Nolabels ASF format only: Assume sequence data have no sequence tag names in the `<Seqfile>`. GENECONV will assign sequence names `S1`, `S2`, `S3`, ... for output, as it does if `NoLabels` is specified in a NEXUS file. (Sequence file formats)

-Nomac — By default, GENECONV looks for key words that indicate input sequence file format not only at the beginning of the file, but also at file offsets 128 and 256. This allows for a possible MacIntosh resource fork at the beginning of the file. This could cause unexpected behavior if these offsets are in the middle of a comment. This option suppresses looking at file offsets 128 and 256 for these key words. (Sequence file formats)

-Nocomm — NEXUS format only: GENECONV normally passes comments of the form `[!...]` in NEXUS input files to GENECONV output files. This option suppresses these pass-through comments. (Sequence file formats)

-Use_iub — Treat IUB codes for ambiguous nucleotides as distinct nucleotides. (Indels and missing data)

-Mitcodes (/dv) — If `Seqtype=SILENT`, use mammalian mitochondrial nucleotide codes for amino acids instead of the standard (nuclear) codes. (Silent sites and amino acids)

-Use_underscores (/qu; off=/qb) — Convert embedded spaces in sequence and group names to underscores. Default: ON. (Sequence file formats)

OPTIONS FOR WRITING OUTPUT FILES:

-Annotate — Write unaligned fragment offsets and unaligned fragment lengths to output files as well as aligned offsets and aligned lengths. (Fancier output)

-ExpFormat — Write Karlin-Altschul (or BLAST-like) P-values to output files in exponential format. (Fancier output)

-WideCol — Write Karlin-Altschul P-values to output files in wider columns. Suppressed if `-ExpFormat` (Fancier output)

-Fancy (/f) — Set the two flags `-Annotate` and `-ExpFormat` . (Fancier output)

-ShowBlast (/sb) — Write tables with formulas for BLAST-like global scores to output files. (Corrected scores do make a difference)

-Showpoly (/sp) — Write a list of the polymorphisms and their offsets to output files. (A first example)

-ShowBcPwKaPvals — Write both pairwise and Bonferroni-corrected K.A. P-values to output files in both global and pairwise fragment lists. (Fancier output)

-Showlog10pvals — Write `-log10` of the appropriate K.A. P-values in a separate column in output files. (Fancier output)

-Truncate_pvals — Convert ``> 1.00" to 1.00 in columns of BC KA Pvalues and ``< 0.00" to 0.00 in columns of -log10(KAPvalues). This may be useful if these columns must be numerical.

-SortName (/sn) — Sort sequences and groups alphabetically in output. Otherwise, sequences and groups are sorted in the order in which they are first encountered.

-SortGfragsBySeq (/sg) — Sort global fragment lists alphabetically by sequence names. Otherwise, global lists are sorted by increasing P-value. This option first calls `-SortName` internally.

-SortGfragsBoth (/sa)	Show global fragment lists sorted by increasing P-value followed by the same list sorted alphabetically by sequence name. This option first calls `-SortName` internally.
-AddOffset=<num>	Add `<num>` to alignment offsets in all GENECONV output. This is useful if the sequences in `<Seqfile>` are parts of longer sequences and you want to use the offsets from the longer sequences.
-Numbersep=<ch>	Replace comma by `<ch>` in numerical output of the form 123,456 .
-Show_maxmeansims	Write a table to output files that shows the maximum, mean, and standard deviation of simulated values of the maximum global score.
-ShowUnal	Write a table with the unaligned lengths of input sequences to output files.
-Logfragments	Write the significant fragments to the log (sum) file as well as to output files.
-Use_underscores (/qu; off=/qb)	Convert embedded spaces in sequence and group names to underscores. Default: ON. (Sequence file formats)
-Tok_linesize=<num>	Change the line size (line width) in word-wrapped lists in outfile files to `<num>` . This may be useful for spreadsheets or word processors that have wider natural line widths. Does not affect screen messages or log file. Default: 72 characters. Minimum value: 50 .
-Notimer	Suppress elapsed-time messages on screen and in the log file. This options has no effect when compiled for UNIX.
-Showfragdata (/sd)	List pairwise maximum scores, mismatch penalties, and the number of fragments stored per pair. Only for those interested in GENECONV inner workings.

OPTIONS FOR DETERMINING FRAGMENT LISTS:

-Gscale=<num> (/g<num>)	Set the gscale parameter for determining mismatch penalties. Default: gscale=0, which means that mismatches are not allowed. <Num> must be a nonnegative integer. (Mismatch penalties)
-Circular (/c)	Assume that alignments are physically circular and allow fragments to overlap the alignment endpoints. (Cauliflower mosaic viruses)
-Linear	Restore the default behavior of assuming linear, noncircular alignments. (This has the same effect as -Circular=off ; see Basic GENECONV syntax)
-ListPair (/lp) **-PairList** **-PairWise**	Include pairwise lists in output files. (Fragment lists: Other restrictions)
-NoPairs (/lp0) **-NoPairLists** **-NoPairWise**	Suppress pairwise lists in output files. (Fragment lists: Other restrictions)
-ListPerPair=<num> (/lp<num>)	If num>0, display pairwise lists but list at most <num> fragments in any individual pairwise list. Default limit: 2000. The particular commands -ListPerPair=0 (or /lp0) and -ListPerPair=off are equivalent to -NoPairLists or -ListPair=off and do not affect the ListPerPair setting. (Fragment lists: Other restrictions)
-ListGlobal=<num> (/lg<num>)	List at most <num> fragments in any individual global list. Default: 2000 (Fragment lists: Other restrictions)
-ListBest (/lb)	List the 8 most significant fragments in all global lists without P-value restrictions. Along with a command that specifies pairwise lists, list the best 3 fragments per pair. Equivalent to -ListGlobal=8 , -ListPerPair=3 , and -NoMaxPval , except that pairwise lists are not automatically specified. (Pairwise and global P-values, Fragment lists: Other restrictions)
-Minlength=<num> (/ml<num>)	Ignore fragments with aligned length less than <num> bases. Default: 1. Must be at least 1. (Fragment lists: Other restrictions)
-Minnpoly=<num> (/mp<num>)	Ignore fragments with fewer than <num> polymorphisms. Default: 2. Must be at least 1. (Fragment lists: Other restrictions)
-Minscore=<num> (/ms<num>)	Ignore fragments with pairwise scores less than <num>. Default: 2. Must be at least 1. (Fragment lists: Other restrictions)
-Maxoverlap=<num> (/v<num>)	Allow at most <num> fragments to overlap a higher-scoring fragment if mismatches are allowed. Default: 1 , which prohibits overlapping fragments. (Overlapping fragments)
-Include_monosites	Include monomorphic sites in polymorphism lists. This removes controls for constant sites, but permits analysis with two sequences. (Including monomorphic sites in polymorphism lists)

OPTIONS FOR P-VALUE BOUNDS IN FRAGMENT LISTS:

In the options below, Pairwise can be replaced by Pair or Pw , and Global can be replaced by Glob or Mc . Examples are -MaxSimPwPval=0.02 or -MaxKaMcPval=0.05. Short forms of options are included in parentheses. See Pairwise and global P-values and Fragment lists: Restricting by P-value for discussion and examples. By convention, setting a P-value bound to 1.00 or greater (for example, /mip2) tells GENECONV to remove that particular bound.

By default, GENECONV only outputs global lists. Enter -ListPair or /lp to have GENECONV

produce pairwise lists as well.

-Maxsimglobalpval=<pval> (/mig<pval>)	Restrict global fragment lists to fragments with permutation P-value equal to `<pval>` or less. Examples: `-MaxSimGlobPval=0.01` or `/mig0.01` . Default: 0.05 .
-Maxsimpairwisepval=<pval> (/mip<pval>)	Restrict pairwise fragment lists to fragments with permutation P-value equal to `<pval>` or less. Default: 0.05 .
-Maxkaglobalpval=<pval> (/mkg<pval>)	Restrict global fragment lists to fragments with Karlin-Altschul P-value equal to `<pval>` or less. Default: 0.05 if no permutations are done, otherwise by default not applied.
-Maxkapairwisepval=<pval> (/mkp<pval>)	Restrict pairwise fragment lists to fragments with Karlin-Altschul P-value equal to `<pval>` or less. Default: 0.05 if no permutations are done, otherwise by default not applied.
-Nomaxsimpval (/miz)	Remove all permutation P-value conditions on pairwise and global fragment lists.
-Nomaxkapval (/mkz)	Remove all Karlin-Altschul P-value conditions on pairwise and global fragment lists.
-Nomaxpval (/mz)	Remove all P-value conditions on pairwise and global fragment lists. This may result in very large output files.
-Bcsims **-Noblast**	For global permutation P-values, use Bonferroni-corrected pairwise permutation P-values. as opposed to BLAST-like global scores. Equivalent to `-Useblast=off` . (Assessing significance: pairwise and global P-values and When global (BLAST-like) scores are not enough)
-Useblast	For global permutation P-values, use BLAST-like global scores as opposed to Bonferroni-corrected pairwise permutation P-values. This is the default. (Assessing significance: pairwise and global P-values and When global (BLAST-like) scores are not enough)

OPTIONS FOR FRAGMENT TYPES TO ANALYZE:

-Innerpair Look for and display significant inner (pair) fragments. This is the default. Enter `-Innerpair=off` to suppress. (Finding gene conversion events)

-Outerseq (/os) Look for and display significant outer sequence fragments. This is the default. Enter `-Outerseq=off` or `/ons` to suppress. (Outer fragments)

-Outerpair (/op) Look for and display significant outer pair fragments. (Outer fragments)

-Outergroup (/og) Look for and display significant outer group fragments. This option has no effect unless groups are defined. (Outer fragments, Groups of equal size, Groups of unequal size)

-Allouter (/oa) Look for and display significant outer sequence, outer pair, and outer group fragments. (Outer fragments)

-Noouter (/ona) Suppress analyses for all outer fragments. (Outer fragments)

-Outersims (/oi) Compute permutation P-values for outer fragments if permutations are done. Default: ON. Enter `-Outersims=off` (short form: `/oni`) to compute only KA P-values for outer fragments (Outer fragments)

OPTIONS FOR PERMUTATIONS:

-Numsim=<num> (/n<num>) Compute permutation P-value using `<num>` permutations. To suppress permutations, enter `-Numsim=0` or `-Numsims=off` . Default: 10,000 permutations. (Assessing significance, Pairwise and global P-values)

-Irrupt=<num> (/i<num>) Display a number on the computer screen after every `<num>` permutations. Enter `-Irrupt=off` or `/i0` to suppress. Defaults: 500 in Microsoft, 0 (suppressed) if compiled for UNIX.

-Startseed=<num> **-Seed=<num> (/w<num>)** Initialize GENECONV's random number seed at `<num>` . This option has no effect unless either permutations are done or `-Randomize_sites` is entered. The seed is initialized from the system clock if no `-Startseed` commands are entered or if `-Startseed=0` . (A first example)

-Randomize_sites Randomize the order of polymorphic sites before analyses. This should destroy most if not all statistical significance and is a test for GENECONV. (Checking GENECONV)

-Ignore_degen (/da) If `-Seqtype=SILENT` and if permutations are done, permute all polymorphisms in one class. The default with `-Seqtype=SILENT` is to permute 4-fold degenerate, 3-fold degenerate, 2-fold degenerate, and irregular silent codon polymorphisms separately. (Silent sites and amino acids, Sawyer 1989 in the References)

-Combine_3degen (/d3) If `-Seqtype=SILENT` and permutations are done, and if `-Ignore_degen` is not entered, do permutations with 3-fold and 2-fold degenerate sites combined in one class. There is only one 3-fold degenerate amino acid (isoleucine) in the usual nuclear encoding scheme, and its third codon (ATA) is rare or absent in some alignments. (Silent sites and amino acids)

-Simp_poly_only (/ds) Use only *simple polymorphisms* for polymorphic sites. By definition, a simple polymorphisms is a polymorphism with the same nucleotide or amino acid in all sequences but one and a second nucleotide or amino acid in the remaining sequence.

-Mult_poly_only (/dm) Use only *multiple polymorphisms* for polymorphic sites. By definition, a multiple polymorphism is a polymorphism other than a simple polymorphism. (Sometimes

`  '  polymorphism is a polymorphism other than a simple polymorphism. (Sometimes these are called *phylogenetically informative* polymorphisms.) GENECONV will exit with a complaint if both `-Simp_poly_only` and `-Mult_poly_only` are entered. (Checking GENECONV)

OPTIONS FOR CONFIGURATION FILES:

`-Config=<str>` Read `<filename>` as a configuration file. If `<filename>` does not end with `.cfg` , then `.cfg` is appended. (Configuration files)

`-Group=<gname> <str1> <str2> ...` Define or extend a group for searching for within-group gene conversions. (Groups of unequal size; Defining groups, skipping sequences, and listing sequences)

`-Option` Open an options block in a configuration file. (Configuration files)

`-Endoption` Close an options block in a configuration file. (Configuration files)

OPTIONS FOR GROUPS:

`-Bloc=<k>` **`(/b<k>, /bh<k>)`** Define groups of `<k>` consecutive sequences each for within-group gene conversion. The syntax `/bh<k>` also sets the flag `Homologous` (An example with groups of equal size)

`-Homologous (/h)` If groups of consecutive sequences of equal size are defined by `-Bloc=<k>` or `/b<k>`, look for *homologous* instead of *within-group* fragments. (An example with groups of equal size)

`-Notfound_nonfatal` If a sequence in a `-Group` or `-Seq_list` command is missing, continue with the sequences that can be found. By default, GENECONV exits if such a sequence is missing. (Defining groups and skipping and listing sequences)

`-Group=<gname> <str1> <str2> ...` Define or extend a group for searching for within-group gene conversion. (Groups of unequal size; Defining groups, skipping sequences, and listing sequences)

`<gname>=Gen_Poly....` Group names that begin with ``Gen_Poly'' are treated in a special way. These add more polymorphic sites to the alignment, but GENECONV does not look for significant fragments within these groups.

OPTIONS FOR PATHS, SEQUENCE LISTS, AND OFFSET RANGES:

-Filepath=<str> **-Input=<str>**	Set a search path for input sequence files and configuration files. The two forms are equivalent. Remove a pre-existing path by `-Filepath=` (followed by a blank). Since GENECONV matches initial strings, you can also enter `-Inputpath=<str>` or `-Inputfiles=<str>` (Search paths)
-Add_filepath=<str> **-Add_input=<str>**	Add to an existing search path for input files and configuration files. The two forms are equivalent. Since GENECONV matches initial strings, you can also enter `-Add_Inputpath=<str>` or `-Add_Inputfiles=<str>` (Search paths)
-Seq_list=<str> **-Subset=<str>**	Use ONLY these sequences from the input file. The two forms are equivalent. (Defining groups and skipping and listing sequences)
-Add_seq_list=<str> **-Add_subset=<str>**	Add sequences to the current `Seq_list` list. The two forms are equivalent. (Defining groups and skipping and listing sequences)
-Seq_skip=<str> **-Skip_list=<str>**	IGNORE sequences with names in `<str>` in the input file. The two forms are equivalent. (Defining groups and skipping and listing sequences)
-Add_seq_skip=<str> **-Add_skip_list=<str>**	Add sequences to the current `Seq_skip` list. The two forms are equivalent. (Defining groups and skipping and listing sequences)
-List_only=<PairList> **-Listonly=<PairList>**	Show only fragments from these sequence pairs in output files, but use other sequences for polymorphisms. The two forms are equivalent. (Defining groups and skipping and listing sequences)
-Add_List_only=<PairList> **-Add_Listonly=<PairList>**	Add sequence pairs to the current `<PairList>` . The two forms are equivalent. (Defining groups and skipping and listing sequences)
-Test=<PairList>	Show special low-level information about the sequence pairs in `<PairList>` . Only for those interested in the inner workings of GENECONV. `<PairList>` has the same syntax as in `-Listonly`. (Defining groups and skipping and listing sequences)
-Base_range=<n1-n2>	Use only bases at positions within these offsets (inclusively) in the input file. Only one base range can be used. Example: `-base_range=20-100.`
-Base_skip=<n1-n2>	IGNORE bases at positions within these offsets (inclusively) in the input file. Only one base range can be used. Example: `-base_skip=50-70.`

LITERATURE CITED

Altschul, S. F. (1993) A protein alignment scoring system sensitive at all evolutionary distances. *Jour. Molec. Evol.* 36, 290-300.

Altschul, S. F., W. Gish, W. Miller, E. W. Myers, and D. J. Lipman (1990) Basic local alignment search tool. *Jour. Molecular Biology* 215, 403-410.

Bonferroni, C. E. (1936) Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze* 8, 3-62.

Chenault, Kelly D., and Ulrich Melcher (1994) Phylogenetic relationships reveal recombination among isolates of Cauliflower Mosaic Virus. *Jour. Molec. Evol.* 39, 496-505.

Deininger, P. (1989) SINEs: short interspersed repeated DNA elements in higher eucaryotes. Pp619-636 *in* D. Berg and M. Howe (eds.), *Mobile DNA.* American Society for Microbiol., Washington DC.

Drouin, G., Frederic Prat, Michael Ell, and G. D. Paul Clarke (1999) Detecting and characterizing gene conversions between multigene family members. *Molec. Biol. Evol.* 16, 1369-1390.

DuBose, R., D. E. Dykhuizen, and D. L. Hartl (1988) Genetic exchange among natural isolates of bacteria: recombination within the *phoA* locus of *Escherichia coli*. *Proceedings National Acad. Sci. U.S.A.* 85, 7036-7040.

Grassly, N. C., and E. C. Holmes (1997) A likelihood method for the detection of selection and recombination using sequence data. *Molec. Biol. Evol.* 14, 239-247.

Guttman, David S., and Daniel E. Dykhuizen (1994) Clonal divergence in *Escherichia coli* as a result of recombination, not mutation. *SCIENCE* 266, 1380-1383.

Gyllensten, U. B., M. Sundvall, and H. A. Erlich (1991) Allelic diversity is generated by intraexon sequence exchange at the *DRB1* locus of primates. *Proceedings National Acad. Sci. U.S.A.* 88, 3686-3690.

Hein, J. (1990) Reconstructing evolution of sequences subject to recombination using parsimony. *Math. Biosci.* 98, 185-200.

Hein, J. (1993) A heuristic method to reconstruct the history of sequences subject to recombination. *Jour. Molec. Evol.* 36, 396-405.

Hilliker, A. J., S. H. Clark, and A. Chovnick (1991) The effect of DNA sequence polymorphisms on intragenic recombination in the *rosy* locus of *Drosophila melanogaster*. *GENETICS* 129, 779-781.

Hilliker, A. J., G. Harauz, A. G. Raume, M. Gray, S. H. Clark, and A. Chovnick (1994) Meiotic gene conversion track length distribution within the *rosy* locus of *Drosophila melanogaster*. *GENETICS* 137, 1019-1026.

Hudson, R. R., and N. L. Kaplan (1985) Statistical properties in the number of recombination events in the history of a sample of DNA sequences. *GENETICS* 111, 147-164.

Jakobsen, I.B., S. R. Wilson, and S. Easteal (1997) The partition matrix: Exploring variable phylogenetic signals along nucleotide sequence alignments. *Molec. Biol. Evol.* 14, 474-484.

Jakobsen, I.B., S. R. Wilson, and S. Easteal (1998) Patterns of reticulate evolution for the classical class I and II HLA loci. *Immunogenetics* 48, 312-23.

Kapur, Vivek, S. Kanjilal, M. R. Hamrick, Ling-Ling Li, T. S. Whittam, S. A. Sawyer, and J. M. Musser (1995) Molecular population genetic analysis of the streptokinase gene of *Streptococcus pyogenes*: mosaic alleles generated by recombination. *Molecular Microbiology* 16, 509-519.

Karlin, S., and S. F. Altschul (1990) Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings National Acad. Sci. U.S.A.* 87, 2264-2268.

Karlin, S., and S. F. Altschul (1993) Applications and statistics for multiple high-scoring segments in molecular sequences. *Proceedings National Acad. Sci. U.S.A.* 90, 5873-5877.

Karlin, S., A. Dembo, and T. Kawabata (1990) Statistical composition of high-scoring segments from molecular sequences. *Annals of Statist.* 18, 571-581.

Klein, J., and C. Schonbach (1993) Origins of Mhc diversity. *In* B. G. Solheim, S. Ferrone, and E. Muller (eds), *The HLA system in clinical transplantation*, 16-37, Springer, Heidelberg New York.

Lehrman, M., D. Russell, J. Goldstein, and M. Brown (1987) Alu-Alu recombination deletes splice acceptor sites and produces secreted low density lipoprotein receptor in a subject with familial hypercholestremia. *Jour. Biol. Chem.* 262, 3354-3351.

Maddison, W. P., Swofford, D. L. and Maddison, D. R. (1997) NEXUS: an extendible file format for systematic information. *System. Biol.* 46, 590-621.

Maeda, N., C.-I. Wu, J. Bliska, and J. Reneke (1988) Molecular evolution of intergenic DNA in higher primates: pattern of DNA changes, molecular clock, and evolution of repetitive sequences. *Molec. Biol. Evol.* 5, 1-20.

Maynard Smith, J. (1992) Analyzing the mosaic structure of genes. *Jour. Molec. Evol.* 34, 121-134.

Maynard Smith, J. and N. H. Smith (1998) Detecting recombination from gene trees. *Molec. Biol. Evol.* 15, 590-599.

McGuire, G., F. Wright, and M. J. Prentice (1997) A graphical method for detecting recombination in phylogenetic data sets. *Molec. Biol. Evol.* 14, 1125-1131.

Miller, Rupert G. (1981) Simultaneous statistical inference. 2nd ed., Springer Verlag, pages 6-8.

Moniz de Sa, M., and G. Drouin (1996) Phylogeny and substitution rates of angiosperm actin genes. *Molec. Biol. Evol.* 13, 1198-1212.

Padidam, M., S. Sawyer, and C. M. Fauquet (1999) Possible emergence of new geminiviruses by frequent recombination. In press, *Virology*.

Revers, F., O. Le Gall, T. Candresse, M. Le Romancer, and J. Dunez (1996) Frequent occurrence of recombinant potyvirus isolates. *Journal of General Virology* 77, 1953-1965.

Sawyer, S. A. (1989) Statistical tests for detecting gene conversion. *Molecular Biology and*

Evolution 6, 526-538.

Sawyer, S. A., D. E. Dykhuizen, and D. L. Hartl (1987) Confidence interval for the number of selectively neutral amino acid polymorphisms. *Proceedings National Acad. Sci. U.S.A.* 84, 6225-6228.

Semple, C., and K. Wolfe (1999) Gene duplication and gene conversion in the *Caenorhabditis elegans* genome. *Jour. Molec. Evol.* 48, 555-564.

Sneath, P., M. Sackin, and R. Ambler (1975) Detecting evolutionary incompatibilities from protein sequences. *Systematic Zoology* 24, 311-332.

Sneath, P. (1998) The effect of evenly spaced constant sites on the distribution of the random division of a molecular sequence. *Bioinformatics* 14, 608-616.

Stephens, J. C. (1985) Statistical methods of DNA sequence analysis: Detection of intragenic recombination or gene conversion. *Molec. Biol. Evol.* 2, 539-556.

Takahata, Naoyuki (1994) Comments on the detection of reciprocal recombination or gene conversion. *Immunogenetics* 39, 146-149.

Waterman, M., and M. Eggert (1987) A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *Jour. Mol. Biol* 197, 723-728.

Weiller, Georg F. (1998) Phylogenetic profiles: A graphical method for detecting recombinations in homologous sequences. *Molec. Biol. Evol.* 15, 326-335.

Witherspoon, David J., T. G. Doak, K. R. Williams, A. Seegmiller, J. Seger, and G. Herrick (1997) Selection of the protein-coding genes of the *TBE1* family of transposable elements in the ciliates *Oxytrichia fallax* and *O. trifallax*. *Molec. Biol. Evol.* 14, 696-706.

HOW TO CITE GENECONV

A paper describing GENECONV has not yet appeared. In the meantime, try

S. A. Sawyer (1999) GENECONV: A computer package for the statistical detection of gene conversion. *Distributed by the author, Department of Mathematics, Washington University in St. Louis, available at* `http://www.math.wustl.edu/~sawyer`.

The reference

Sawyer, S. A. (1989) Statistical tests for detecting gene conversion. *Molecular Biology and Evolution* 6, 526-538.

is out of date for GENECONV, but could be used instead if journal policies only allow references to printed publications.

Send email comments to  *sawyer@math.wustl.edu*

Maintained and supported by:

Stanley Sawyer
Department of Mathematics
Washington University in St. Louis
St. Louis, Missouri 63130, USA

Web address: `http://www.math.wustl.edu/~sawyer`
Email address: `sawyer@math.wustl.edu`

This work was partially supported by NSF grant DMS-9707045.

Version 1.00 posted September 14, 1998
Version 1.70 posted November 21, 1999
Version 1.81 posted August 29, 2000