3SEQ Manual

Authors:

Ha Minh Lam – lamhm@oucru.org, minhlam.ha@gmail.com

Maciej F Boni – mfb9@psu.edu, maciekboni@gmail.com

Date of this copy: February 8, 2018

When using this software, please cite:

> Lam HM, Ratmann O, Boni MF. Improved algorithmic complexity for the 3SEQ recombination detection algorithm. *Mol Biol Evol*, 35(1):247–251, 2018.

If you are making extensive use of the Hogan-Siegmund approximations or some other basic part of the algorithm as developed in the original 2007 paper, you may also find it appropriate to cite one of these two papers:

> Boni MF, Posada D, Feldman MW. An exact nonparametric method for inferring mosaic structure in sequence triplets. *Genetics*, 176:1035–1047, 2007.

> Hogan ML, Siegmund D. Large deviations for the maxima of some random fields. *Advances in Applied Mathematics*, 7:2-22, 1986.

Contents

1 What is 3SEQ?

3SEQ is a command-line program for identifying mosaic structure or recombination in nucleotide sequence data. 3SEQ takes as input a data set with a minimum of three aligned sequences, and it tests whether any sequence in the data set is a recombinant or mosaic of any other two sequences in the data set. Recombinant sequences are allowed to have one or two breakpoints. Multi-breakpoint recombinants may be inferred by running 3SEQ multiple times; see section 7.4.

Given an input data set, 3SEQ will consider all combinations of three sequences to see if some sequence is a mosaic of two other sequences. If your input contains N sequences, 3SEQ will make $N \cdot (N-1) \cdot (N-2)$ comparisons which can be quite time-consuming when $N > 500$, and in these cases parallelization may be useful. At the end of the run, 3SEQ will report a p-value corrected for the multiple triplet comparisons made in that run; the statistical correction used is a Dunn-Šidák correction, which is very conservative as it assumes independence of all the tests being performed.

3SEQ currently runs on Linux/Unix systems and on Macintosh systems with OS X or later. Darren Martin (University of Cape Town) has coded a version of 3SEQ for Windows and it appears in the RDP package downloadable from the RDP homepage at

```
http://web.cbio.uct.ac.za/~darren/rdp.html
```

2 The most important thing to know: memory usage

3SEQ is a memory-intensive software program. This means that 3SEQ will use as much RAM and as much virtual memory (SWAP) as you allow it to. Usually, 3SEQ will tell you the amount of memory it will need and it will ask you to confirm that this is ok (this feature can be turned off if 3SEQ will be called repeatedly from a script using the -p option). This means that you need to know how much memory (RAM) and virtual memory (SWAP) your system has. The easiest way to find this out is to type 'top' at the command prompt. Some information and a table of processes will appear, and the fourth and fifth lines that appear in the command window should look something like:

```
Mem:   2066396k total,   727952k used,  1338444k free,    90996k buffers
Swap:  1076312k total,        0k used,  1076312k free,   307280k cached
```

These two lines tell you that you have 2066396 kilobytes (about 2 gigabytes) of random-access memory and 1076312 kilobytes (about 1 gigabyte) of virtual memory. This means that you can ask 3SEQ to use up to 3 gigabytes of memory, though in this case it would not be recommended to use more than 2.8 or 2.9 gigabytes.

Note that using virtual memory can significantly decrease the performace of your system. If your system has 2GB of RAM and 1GB of SWAP, optimal performace will be achieved when allowing 3SEQ to use around 1.8GB to 2.0GB of memory. You may notice a significant drop-off in speed if you allow 3SEQ to use 2.2GB or 2.4GB of memory. Sometimes you may need this much memory to perform a particular calculation. If you don't think you need to use the SWAP space, it is better not to use it.

2.1 The p-value table

3SEQ uses a pre-computed table of p-values for its calculations as this is more efficient than computing p-values on the fly. The first time you run 3SEQ , you will need to build a table like this, or you will need to download one before use.

This p-value table can occupy as much as several gigabytes of RAM, depending on the exact dimensions of table you will be using, and this is the reason to be aware of the amount of RAM on your system. As an example, if your alignment has fewer than 800 polymorphic sites, 3SEQ will be able to compute all p-values for candidate recombinants using a table that fits into 650mb of RAM. For an alignment with 1400 polymorphic sites, a table occupying 3.5gb of RAM would be necessary to guarantee that all p-values will be computed exactly. However, in

this case, it is more efficient to use a smaller p-value table ($\sim$1gb RAM), which will still allow the vast majority of recombinant candidates to be reported with exact p-values; for the minority where this is not possible, the p-values will be approximated with Hogan-Siegmund approximations.

There are three ways for you to set up your p-value table

1. generate your own local p-value table with `3SEQ` (recommended, see step 3a below)
2. download a p-value table from `http://mol.ax/3seq`
3. ask `3SEQ` to fetch p-values from a table on the `mol.ax` server every time it runs (not implemented yet)

3 Getting Started

`3SEQ` is written in `C++` for Linux/Unix and for Macintosh computers using OS X or later. If you are using a Mac, you will need to have a set of tools called `XCode` installed on your computer, which is freely downloadable from the Mac App Store. Once `XCode` is installed properly, your Mac will have an application called Terminal, and `3SEQ` will run from the Terminal program's command line. On either Unix/Linux or Mac systems, inside the Terminal application, simply type

```
g++ --version
```

at the command prompt to make sure you have the Gnu C compiler installed; any version number higher than version 4 should be fine.

Step 1. Download the zip file with all the source files into a local folder on your computer. At the command prompt type

```
unzip 3seq-170612.zip
```

then,

```
cd 3seq-170612
```

and then simply type

```
make
```

at the prompt to compile the source code. For advanced users, type `make help` to see the other build options.

Step 2. To test if the compilation was successful, simply type

```
./3seq
```

at the prompt and see if `3SEQ` outputs a list of the available runmodes as well as a short message saying that the program is not yet associated with a p-value table.

Step 3a. Generate a p-value table with `3SEQ` by typing

```
./3seq -g my3seqTable700 700
```

which will generate a p-value table file called `my3seqTable700` of size $700 \times 700 \times 700$. On a modern laptop ($\geq$2GHz processor, $\geq$4gb RAM) this should take between 5 and 20 minutes. Generating a p-value table of size $1000 \times 1000 \times 1000$ should take between 20 and 45 minutes, and this file will take about 1.3gb of disk space. A laptop with 16gb of RAM and an up-to-date linux/unix system will be able to build a table of size 1800.

Step 3b. Alternatively, download a p-value table from `http://mol.ax/3seq` and save it in a local directory.

Step 4. To check that the p-value table is read in correctly, use the 'check' mode by typing

```
./3seq -c my3seqTable700
```

and this will both check that the downloaded file is valid and it will automatically associate this file with the 3SEQ executable. You can now check individual p-values in the table by typing any three integers on the command line like so

```
Enter M N K to test : 300 200 60
```

which should give you a p-value of 4.4542×10^{-16}.

```
Enter M N K to test : 300 200 20
```

should give you a p-value of 0.00304938.

```
Enter M N K to test : 300 200 10
```

should give you a p-value of 0.44096.

```
Enter M N K to test : 700 700 250
```

should give you a p-value of 3.21187×10^{-38}. You can use the up arrow for history. Press Esc to exit this mode.

You can check which p-value table is associated with your 3SEQ executable by typing `./3seq` at the prompt without any arguments. On a linux system, this information is stored in

```
~/.config/3seq/3seq.conf
```

and on Mac it is stored in

```
~/Library/3seq/3seq.conf
```

4 Input files

Sequences must be aligned and sequence files which are passed in as input to 3SEQ must be in either PHYLIP or FASTA format. Remember to check that the newlines in the sequence files are unix-style newlines or mac-style newlines, depending on your platform. As in all phylip files, the first line must contain two integers: the number of sequences and the length of each sequence. In phylip files, the sequence names cannot contain spaces. There are two types of phylip formats that are acceptable.

4.1 phylip format, type 1

After the first line of the file, each line corresponds to one sequence. The first bit of text on the line corresponds to the accession number or name of the sequence; make sure this part has no spaces or else 3SEQ will not be able to read in the sequences. This type of file should look something like:

```
 3 1000
sample1    GTCCGCTGAAAG....
sample2    GCTCGTGGAATG....
sample3    GCTCGACCCAAG....
```

4.2 phylip format, type 2

Very similar to type 1, only this format has a newline after the sequence names. Again, no spaces are allowed in the sequence name.

```
 3 1000
sample1
GTCCGCTGAAAG....
sample2
GCTCGTGGAATG....
sample3
GCTCGACCCAAG....
```

4.3 fasta format

Fasta file will simply have a ">" character before the sequence name. In fasta format, spaces are allowed in the sequence name.

```
> sequence01
GTCCGCTGAAAG...
AACGCTCGTAAG...
> sequence02
GTCCGCTGAAAG...
AACGCTCGTAAG...
> sequence03
GTCCGCTGAAAG...
AACGCTCGTAAG...
```

4.4 How to check if your file is in the right format

Simply type

```
./3seq -info mysequencefile.phy
```

or

```
./3seq -i mysequencefile.phy
```

at the prompt and 3SEQ should give you some information about the sequences in that file. If this does not work, your input file may be in the wrong format.

5 Quickest way of getting started

If your p-value table is built and your sequences are in the right format, type

```
./3seq -f mysequencefile.phy -d
```

to execute a 'full run' of 3SEQ . This will calculate the best set of breakpoints for each candidate recombinant sequence. The flag `-d` tells 3SEQ to include only genetically distinct sequences.

6 Run Modes

The "run mode" is the first command-line argument you need to specify. You do not need to type out the entire name of the run mode; the first letter will suffice. For example `3seq -v` will give you version information while `3seq -f` will perform a full run of the algorithm.

6.1 version mode

Typing `./3seq -v` should give you basic information about the version number and what to cite when using 3SEQ .

6.2 help mode

Typing `./3seq -h` at the prompt will give you a list of runmodes and options.

6.3 info mode

Notes: -b and -e command-line arguments will not work here.

Typing `./3seq -i myfile.aln` gives you information on the sequences in your sequence file.

6.4 write mode

This mode simply writes an alignment to a new file.

Note: remember to use -a option here if you want all sites and not just polymorphic sites.

Typing

```
./3seq -w myFile.aln myNewFile.aln -a
```

simply outputs the datafile to `myNewFile.aln`. If you omit the -a option, only polymorphic sites will be output. Using this mode, you can create new subsegments and subsets of the data. A common use would be to remove duplicate sequences. To do this, type

```
./3seq -w mySeqs.aln myUniqueSeqs.aln -a -d
```

To create a new phylip file with just nucleotides 400–600 from the first 12 sequences in the data file, simply type

```
./3seq -w mySeqs.aln mySeqsSubset.aln -a -f400 -l600 -b0 -e11
```

Or, if you have a text file called `my_acc_nums` containing sequence names, separated by whitespace, you can use the `-subset` option to create a new data file:

```
./3seq -w mySeqs.aln mySeqsSubset.aln -a -subset my_acc_nums
```

6.5 single mode

This mode will loop through all sequence triplets, and it will look for single-breakpoint recombinants only. No p-value table is required for this mode as the p-value for this scenario can be computed analytically.

Currently this only reports a p-value and generates a log file.

6.6 triplet

This gives you information on the `P`-informative sites, `Q`-informative sites, and maximum descent for a particular triplet. Using the dengue data set included with the program, try

```
./3seq -triplet den2.aln D2p8-377/? D2Mexic/? D2Tonga/74
```

and `3SEQ` should display some information on these three sequences, their distance to one another, and the "maximum descent" statistic which tells you how likely it is that the child sequence is a mosaic of the parent sequences.

This mode will also generate a file called `3s.61_64_4.triplet.eps` that shows you where all the informative sites are graphically. The '61', '64', and '4' are the orders of these threee sequences in the data set (starting counting at 1). The eps file portrays, along the length of the sequence, the informative sites in that sequence triplet. Red lines are informative sites of type `P`, i.e. those where the child sequence is similar to parent `P`, the first parent, but different from parent `Q`, the second parent. Blue lines are informative sites of type `Q`, i.e. those where the child sequence is similar to parent `Q` but different from parent `P`. The gray lines (they should be about 1/3 the height of the red and blue lines, starting at the bottom edge of the diagram) indicate polymorphic sites that were not informative for that sequence triplet.

If you open the eps file and it doesn't look very nice, or if the lines are not visible due to the sequence length, simply open the eps file in a text editor and the first ten lines will look like

```
%%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 2972 240
%%%%EndComments

%%%%%%%%%%%%
/Times-Roman findfont
7 scalefont
setfont

0.001 setlinewidth

/w {2} def    % This is the width of a small box on the plot

/h {240} def    % This is the height of a small box on the plot
```

You can adjust the heights and widths of the red and blue lines here. The red and blue lines are actually drawn as filled boxes in postscript format.

6.7 fullrun mode

This is the main run mode to be used for detecting recombinant triplets.

This mode will read in a p-value table from a file on disk. Thus, you should have a file of pre-computed p-values already associated with 3SEQ . You can check if this is the case with 'check' mode.

To begin your run, type

```
./3seq -f mysequencefile.phy
```

or

```
./3seq -fullrun mysequencefile.phy
```

to begin looping through your sequence set. You will be prompted as to whether you are sure that you want to use a given amount of RAM and whether you want to overwrite existing files generated with previous runs of 3SEQ , if these exist.

It is good practice to run this analysis as

```
./3seq -f mysequencefile.phy -d -id myTuesdayRun0001
```

as the `-d` flag will remove identical sequences from the analysis, and the `-id` option will give this run's output files the chosen prefix so they can be located and filtered easily.

An auto id tag can be generated by running

```
./3seq -f mysequencefile.phy -d -id-auto
```

When the run is finished, 3SEQ will produce 3 or 4 output files. See the section at the end of the manual for how to interpret the output files.

7 Options

7.1 -a

When this option is selected, all nucleotides are used in the analysis rather than just polymorhpic sites. The only time you would really want to use this option is in `write` mode when you are writing your sequences to a new file.

7.2 -b -e

These options stand for 'beginning' and 'end', and they allow you to test a subset of the sequences to see if they are recombinant. For example, if your data set has 10 sequences, and you simply want to test if the first three are recombinant, run the program as

```
./3seq -f mysequenefile.aln -b0 -e2
```

The multiple comparisons correction will now be done as if you had tested all the sequences for recombination. To turn this behavior off and do a multiple comparisons corrections for only the comparisons done during the actual run, use the -# option, like so

```
./3seq -f mysequenefile.aln -b0 -e2 -#
```

The default behavior is set to correct for all potential comparisons so that you can run 3SEQ in parallel. For example, if your computer has two processors, you can run

```
./3seq -f mysequenefile.aln -b0 -e4 -id 001
./3seq -f mysequenefile.aln -b5 -e9 -id 002
```

on a set of 10 sequences. Just make sure that you have enough memory as these two processes will use separate p-value tables. The -id option above will append the strings 001 and 002 to the output files so that you know which output file came from which run.

7.3 -d

Distinct sequences only. This option removes duplicate sequences, i.e. those sequences that are identical, nucleotide for nucleotide, to other sequences in the data set.

Note that gaps pose a transitivity problem for defining 'identical'. The sequences AA-- and AAAA are considered identical – because the first sequence contains a subset of the nucleotides of the second sequence – and the first sequence is removed. However, the sequences AA--A and AAAA- are not identical, as neither one is a subset of the other and both are kept in the dataset.

7.4 -f, -l, -x

These options let you look at subsequences of the sequence alignment input file; -f and -l stand for 'first' and 'last'. For example, if the alignment is 1000nt long, and you have determined that region 500-750 is most likely a recombinant segment, you can test this sub-segment for further recombination with the command

```
./3seq -f mysequenefile.aln -f500 -l750
```

In addition, you can test the concatenation of nucleotides 1–499 and 751–1000, with the command

```
./3seq -f mysequenefile.aln -f500 -l750 -x
```

which 'cuts out' the section from 500–750, and looks for recombination in the concatenation of the left over segments.

You can use the -f and -l options by themselves. To test the last 500 nucleotides of a 1000nt segment, use

```
./3seq -f mysequenefile.aln -f501
```

To test the first 500 nucleotides, use

```
./3seq -f mysequenefile.aln -l500
```

7.5 -L

One potential problem in identifying recombination is that identified recombinant segments can be very short. Ideally, after finding a mosaic signal in a sequence triplet, we would want to obtain a confirmation that this is a true signal by inferring phylogenies for the different segments and demonstrating the the phylogenetic trees are incongruent. This is difficult to do if one of the segments is very short.

The -L option allows you to count recombination events that result in recombinant segments of a minimum length. The default is set to 100nt. If you would like 3SEQ to report the number of recombinant sequences where both inferred segments will have a minimum length of 500nt or more, simply run

```
./3seq -f mysequenefile.aln -L500
```

7.6 -n

Number of times to repeat a given run. This is meant to be used with the -rand option. The command

```
./3seq -f myseqs.phy -rand 30 -n 100
```

will perform 100 random subsamples of 30 sequences (without replacement) from your alignment and report a p-value for each subsample.

7.7 -p

Not yet implemented in 2017 version.

Only available in fullrun mode. Use this when 3SEQ is being called by a python program (or some other script), and 3SEQ will format the output as a tab-delimited string. Currently, there are seven numbers in this string. From left to right, they are:

1. Dunn-Šidák corrected p-value for the entire data set
2. number of p-values computed exactly
3. number of p-values bounded
4. number of p-values computed with a Hogan-Siegmund approximation
5. number of triplets that were skipped (no p-value assigned)
6. number of sequences in the data set
7. number of recombinant triplets

Use this option when performing power analyses.

7.8 -r

Will not be implemented in versions 1.7 and later.

7.9 -rand

Randomly select a subset of sequences (without replacement) and test for recombination in this subset. For example,

```
./3seq -f myseqs.phy -rand 30
```

will randomly select 30 of your sequences and test for recombination among these 30. The intended use here is with the -n option, like so

```
./3seq -f myseqs.phy -rand 30 -n 100
```

which will repeat 100 subsamples and give you 100 p-values. This is a useful tool when comparing the extent of recombination in two datasets of very different sizes.

7.10 -t

Set the type 1 error threshold. Since the Dunn-Šidák correction done in the analysis is very conservative, you may want to relax the signficance threshold to take a look at some of the triplets that did not survive a multiple-comparisons correction at the .05-level. For example, use

```
./3seq -f mysequenefile.aln -t0.25
```

to look at all triplets whose p-values after a Dunn-Šidák correction are < 0.25.

If your dataset is highly recombinant and you want to take a look at only the most significant recombinant triplets, you can lower the threshold

```
./3seq -f mysequenefile.aln -t1e-25
```

and only triplets with an uncorrected $p < 10^{-25}$ will be recorded.

The default is set to `-t0.05`.

7.11 -y

The algorithm will be run in `Yes/No`-mode. This means that as soon as a recombinant is found, `3SEQ` will stop running. The found recombinant will be written to the file `3s.rec`. Power analyses will proceed much more quickly if this option is used. The `-L` option has no effect on the algorithm's stopping condition.

7.12 -#

See section on `-f` and `-l` options.

7.13 -bp-all

Breakpoint locations will be determined for all recombinant triplets. This slows down the running of the algorithm when there are many recombinant sequences. Default is to record the "best" set of breakpoints for each recombinant, the best set being the one coresponding to the lowest p-value.

7.14 -bp-none

Breakpoint locations will not be determined. This speeds up the running of the algorithm when there are many recombinant sequences. Default is to record the "best" set of breakpoints for each recombinant, the best set being the one coresponding to the lowest p-value.

This option should be enabled, together with the `-nr` option, for power analyses.

7.15 -fasta

Use in `write` mode. The command

```
./3seq -w myfile.aln myfile.afa -a -fasta
```

will convert your file to fasta format.

7.16 -nohs

Suppresses use of the Hogan-Siegmund approximations. When this option is enabled, triplets whose p-values could not be computed using the p-value table will simply be omitted from the analysis.

7.17 -id

Labels the output files 3s.rec, 3s.long_recombinants, 3s.pvalhist, and 3s.skippedpvals so that you can distinguish them from previous runs. For example, the run

```
./3seq -f mysequenefile.aln -id FLU2007
```

will create the files FLU2007.3s.rec, FLU2007.3s.long_recombinants, and FLU2007.3s.pvalhist.

7.18 -id-auto

As above, but an an automatically generated time stamp is used for the id.

7.19 -nexus

Use in write mode. The command

```
./3seq -w myfile.aln myfile.nex -a -nexus
```

will convert your file to nexus format.

7.20 -subset

Uses a subset of sequences. Create a text file with sequence names, each one on a separate line. Then run

```
./3seq -f mySeqFile.phy -subset mySeqNames.txt
```

7.21 -subset-remove

Removes a subset of sequences. Create a text file with sequence names, each one on a separate line. Then run

```
./3seq -f mySeqFile.phy -subset-remove mySeqNames.txt
```

8 Output files

8.1 3s.rec

The output file 3s.rec has a minimum of 13 columns. Each row represents a triplet of sequences where a two-breakpoint recombination signal is significant at the level p (Dunn-Sidak corrected) which is specified by the user with the -t option; default is $p = .05$. The first three columns of this file are the sequences P, Q, and C, respectively. The text in these columns will look like:

```
D2-D87-1040/87      D2Mexic/?      D2PRico-va/69
```

The remaining columns of the output file 3s.rec are as follows:

4. number of P-informative sites
5. number of Q-informative sites
6. maximum descent observed for the triplet
7. p : uncorrected p-value for this triplet
8. 1 if Hogan-Siegmund approximation was used for this triple; 0 otherwise

9. $\log_{10}(p)$

10. Dunn-Sidak correction of p

11. Dunn-Sidak correction of p, in mantissa-exponent format.

12. the minimum length of the recombinant segments; i.e.the minimum length of the shorter of the two recombinant segments.

13+. columns 13 and higher will hold the potential breakpoint locations for this sequence triplet; all breakpoints that minimize expression (4) in the Boni et al. (2007) *Genetics* paper are included here.

8.2 3s.pvalhist

This file is a histogram of p-values computed from all triples in the analysis. The file will look like

```
0       5135    0.7507310       -0.125
1       1160    0.1695906       -0.771
2       322     0.0470760       -1.327
3       105     0.0153509       -1.814
...
```

which means that 5135 p-values (uncorrected) fell in the range $0.1 < p \leq 1.0$, 1160 p-values fell in the range $10^{-2} < p \leq 10^{-1}$, 322 p-values fell in the range $10^{-3} < p \leq 10^{-2}$, 105 p-values fell in the range $10^{-4} < p \leq 10^{-3}$, etc.

The third column shows what fraction of p-valus fell in this range, and the fourth column is $\log_{10}$ of the third column. For a perfectly uniform distribution, the fourth column should be: $-0.046, -1.046, -2.046, -3.046, \ldots$

8.3 3s.log

This holds the same output that you see displayed on the screen during a `3SEQ` run.

8.4 3s.longRec

This holds the names of the recombinant sequences for which both recombinant segments were longer than the length specified by the `-L` option.