3SEQ Manual

Author: Maciej F. Boni (mboni@oucru.org, maciej.boni@ndm.ox.ac.uk)

Date of this copy: December 12, 2008

When using this software, please cite:

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

When using the `-hs` option or `fullrun` mode, please also cite

> 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 software 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 once $N > 500$. At the end of the run, 3SEQ will report a p-value corrected for the multiple comparisons made in that run; the correction is a Dunn-Šidák correction which is very conservative.

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://darwin.uvigo.es/rdp/rdp.html
```

2 The most important thing to know about 3SEQ

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. In many of the runmodes, you will be asked to specify how much RAM you would like 3SEQ to use, or you will be asked to pass in some other parameters and 3SEQ will compute for you how much RAM would be needed for such a run. For example, in something called `quickrun` mode 3SEQ may ask you

```
Total RAM+SWAP needed is 1526 megabytes.

Is this OK (y/N)?
```

and you would need to confirm with a lowercase `y` for 3SEQ to proceed. In this case, 3SEQ would use a maximum of 1526 megabytes of memory, but it may use much less if the sequences are not very diverse. The more nucleotide diversity in the sequences, the more memory is required.

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 on a Linux system 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 it is not recommended to ask for more than 2.8 or 2.9 gigabytes.

2.1 Before you compile

There are two things you need to do before you compile.

First, you should set a hard memory limit in the 3SEQ code for safety.

Open the file `ProbTables.h` and you will see two lines near the top that look like

```
// this lets program use a max of 2.5GB of RAM + SWAP
#define HARDLIMIT 624999000
```

If you would like your hard memory limit set to 2.5 gigabytes, just leave this part as is. To change this limit, choose a number of bytes that you would like to be your hard memory limit, divide it by four, and replace the number after HARDLIMIT with your new limit.

For 32-bit systems, the maximum number that can appear on this line is INT_MAX which is 2147483647. This means that in its current form, 3SEQ can use a maximum of 8 gigabytes of memory. This can be overridden with the class member ytable::override_hardlimit, but as you will see when you read about xrun mode, you should not need more than 300 megabytes of memory.

Second, go down a few lines in the ProbTables.h file until you see:

```
//
// USER NEEDS TO SET THIS DIRECTORY BEFORE COMPILING
//
#define HARD_CODED_PATH_FOR_PVALUETABLES "/home/maciek/PVALUETABLES/"
```

Change the part inside quotation marks to the directory on your local machine where you will store the pre-computed tables of p-values that 3SEQ will reference if you use xrun mode. This is the directory where the file(s) beginning with PVT_COMPACT_ will be located. You don't need to use pre-computed p-values or xrun mode, but it is recommended that you do so since xrun mode is the most efficient way to use 3SEQ .

2.2 Performance and virtual memory

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.

3 Compiling 3SEQ

3SEQ is written in C++ for Linux/Unix operating systems. If you have the compiler g++ installed on your system you should be able to compile 3SEQ without any problem. In downloading and unzipping the source code, you should have the following 10 files:

```
3seq.cpp
gene.cpp
gene.h
makefile
ProbTables.cpp
ProbTables.h
readfile.cpp
readfile.h
seq.cpp
seq.h
```

Simply type

```
make
```

at the Linux prompt, and the source code should compile into the executable 3seq depending on your filesystem. To test if the compilation was successful, simply type

```
./3seq
```

at the prompt and see if 3SEQ outputs a list of the available runmodes.

4 Input files

Sequences must be aligned and sequence files which are passed in as input to 3SEQ must be in PHYLIP format. The newlines in the sequence files must be unix-style newlines. As in all phylip files, the first line must contain two integers: the number of sequences and the length of each sequence. There are three 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 phylip format, type 3, interleaved

Similar to type 1, except that the full sequence can be broken up onto multiple lines. Suppose there are N sequences. Then, after the first N lines which give the sequence names and the beginnings of the sequences, the file should have clusters of N lines at a time indicating the coninutations of the sequences. In the example below lines 2–4 give the beginnings of the three sequences (e.g. nucleotide positions 1–100), while lines 6–8 give the next parts of these three sequences (e.g. nucleotide positions 101-200). Line 5 is empty. There can be as many empty lines as you like between clusters, but there can be no empty lines within a cluster of sequences.

This file format can be output by Clustal. Example below:

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

           AACGCTCGTAAG....
           AACGCTCTTAAG....
           AACGCTCGTCAG....

           TACGCTCGTTAG....
           TACGCTCCCAAG....
           GACGGGCGTAAG....
```

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

Simply type

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

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 you've compiled 3SEQ successfully and if you've verified that your input files are in the correct format the easiest way to get started is to use to download the pre-computed p-value tables from

```
http://www.cggh.ox.ac.uk/?title=MRC_Centre_Maciej_Boni
```

and use `xrun` mode. The pre-computed tables are binary files with names like `PVT_COMPACT_400` and `PVT_COMPACT_600` that should be downloaded and stored in the directory that you set in section 2.1 of this document. The '400' and '600' at the end of these file names mean that p-values can be computed for up to 400 or 600 informative sites, respectively.

On the website, you will find zipped (compressed) versions of the files. Simply download `_PVT_COMPACT_600.tgz` into the directory set in section 2.1. To extract one of these files, simply type

```
tar xvzf _PVT_COMPACT_600.tgz
```

and this should create a file called `PVT_COMPACT_600` in the same directory. You only need one of these files and the larger the file the more exact p-values you will be able to compute. The simplest thing to do now is to type

```
3seq -xrun mysequencefile.phy -d -bp-none
```

and this will look for recombination in your sequence set. The `-d` option restricts the analysis to distinct sequences, and the `-bp-none` option means that no breakpoints will be computed during this run (this is faster).

If this worked, you may notice that some p-values were not computed. You can redo the run by typing

```
3seq -x mysequencefile.phy -d -bp-none -hs
```

and the `-hs` option tells the algorithm to approximate uncomputable p-values with the Hogan-Siegmund method. Note that you can simply type `-x` instead of the full `-xrun`.

If you cannot download the pre-computed p-value tables, the easiest thing to do is to use 'fullrun' mode. Simply type

```
./3seq -fullrun mysequencefile.aln 800
```

or

```
./3seq -f mysequencefile.aln 800
```

and everything should be self-explanatory. The '800' at the end of these commands means that you are allowing 3SEQ to use up to 800 megabytes of memory. Choose this number in the appropriate way for your system.

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

Typing `./3seq -v` should give you basic information about copyright, version number, licensing information, and what to cite when using 3SEQ . The output should look something like:

```
3SEQ -- software for identifying recombination in sequence data.

Copyright  2006-08 Maciej F. Boni.  All Rights Reserved.
Licensed for non-commercial use only.

When using this software software, please cite

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

If you use the '-hs' option or 'fullrun' mode, please also cite

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

version 0.80708
```

6.2 help

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

6.3 info

Notes on options: -b and -e options do not work.

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

6.4 write

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

Typing `./3seq -w myfile.aln -a` simply outputs the datafile to the standard output. 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. For example, to create a new phylip file with just nucleotides 400–600 from the first 12 sequences in the data file, simply type

```
./3seq -w myfile.aln -a -f400 -l600 -b0 -e11 > newfile.aln
```

Or, if you have a text file called `my_acc_nums` containing certain sequences' accession numbers, separated by whitespace, you can use the -subset option to create a new data file:

```
./3seq -w myfile.aln -a -f400 -l600 -subset my_acc_nums > newfile.aln
```

6.5 single

6.6 triplet

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

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

and 3SEQ should display

```
Informative sites of type P             45
Informative sites of type Q             187
Maximum Descent                         156

Breakpoint Ranges:

        0-5 and 1460-1464

# Breakpoint Pairs: 30
# Breakpoint Ranges: 1

Minimum length of recombinant segment is 21nt.
(excluding gaps and ambiguous nucleotides)
```

This mode will also generate a file `3s.triplet.60.63.3.eps` that shows you where all the informative sites are graphically. The '60', '63', and '3' are the indices of these threee sequences in the data set (the same indices that appear in the `3s.rec` file; see section 8.1). 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.

6.7 quickrun

Notes on options: `-p` option not implemented yet

If you don't have the pre-computed p-value tables to use `xrun` mode, `quickrun` mode is the quickest way to analyze sequence data; however, it may leave some p-values uncomputed and it requires some input on your part about the diversity of the data set.

To compute p-values, the `quickrun` mode of 3SEQ builds a table in memory of the $\mathbf{y}_{m,n,k,j}$ values defined above equation (8) of the Boni et al. *Genetics* manuscript. In `quickrun` mode, you tell 3SEQ to build a table of size $m \times m \times k \times k$ by specifying m and k on the command-line like so:

```
./3seq -quickrun mysequencefile.aln 130 100
```

This will build a table in memory of size $130 \times 130 \times 100 \times 100$ which occupies 645 megabytes. All p-values will be calculated from this one table. For some sequence triplets, 3SEQ may not be able to calculate a p-value using this table, and for others a p-value will be approximated. The number of exact computations, approximations, and non-computations will be reported when the algorithm finishes running.

How should you choose m and k? The first place to start is by getting an idea of the sequence diversity in your dataset. Consider the dengue virus data provided with the source files. If you type

```
./3seq -info den2.aln
```

you should discover that the maximum pairwise distance between any two sequences in this dataset is 236nt. The maximum number of informative sites (of type `P` or of type `Q`; see manuscript) is probably less than 236, but it could in principle be as high as 236. To be sure that you would compute all p-values, you would build a table of size $236 \times 236 \times 236 \times 236$; 3SEQ will then warn you that this would require almost 12 gigabytes of RAM and you would then decide to build a smaller table.

One way to build a smaller table is to lower k. This will allow the approximation of p-values for many triplets, but some triplets' p-values will probably not be computable if you lower k too far. For example, try

```
./3seq -quickrun den2.aln 236 50
```

and you will notice that this only requires 532 megabytes of RAM and computes about 80% of the p-values exactly (49,572 are approximated and 12,270 are not computed). As 3SEQ is looping through all the sequence triples, it displays a progress counter

```
QUICK RUN Progress --  74.398%          (1.156626e-09)
```

which shows you in parentheses on the right hand side the current uncorrected p-value for the dataset.

A second way to build a smaller table is to use the maximum number of informative sites (rather than the maximum pairwise distance) for the m-dimension of the table. To determine the maximum number of informative sites, use dryrun mode. In the dengue data, the maximum number of informative sites is 210. Try

```
./3seq -quickrun den2.aln 210 56
```

This uses about the same amount of memory as the previous run (528MB) but it computes 86% of p-values exactly.

6.8 dryrun

The dryrun mode is just like quickrun mode except that no table is built. You can use dryrun mode to find out the maximum number of informative sites (of one type) in all sequence triples in your data set. You can simply type

```
./3seq -dryrun den2.aln 1 1
```

and wait for the results. This mode does in fact loop through all the triples in the data set which means it runs in $\mathcal{O}(N^3)$-time where N is the number of sequences. As this mode is running, you will see a progress indicator

```
DRY RUN Progress --  79.978%            (208)
```

which tells you, in the parentheses in the far right, the maximum number of informative sites of either kind (P or Q) that have so far been found.

6.9 fullrun

Notes on options: -hs option automatically enabled

This mode will build multiple tables and attempt to compute as many p-values exactly as possible. You simply need to specify how many megabytes of RAM you want 3SEQ to use, and it will build and free tables inside the allotted memory until it has made as many exact p-value computations as possible. For example,

```
./3seq -fullrun den2.aln 532
```

can compute or approximate all but four p-values for the dengue data set, but this run took 105 seconds on a 3GHz processor, whereas the quickruns only took about 9 seconds each.

6.10 xrun

Notes on options: -p option not implemented yet

This mode will read in a p-value table from a file on disk. In order for this mode to work, you should have a file of pre-computed p-values in the directory where the 3SEQ executable lives. This file should begin with the twelve characters PVT_COMPACT_ and the following characters should give the dimension of the table. For example, a file named PVT_COMPACT_400 will let you calculate a p-value for any triple where the maximum number of P-informative sites of and the maximum number of Q-informative sites are both ≤ 400. When using the p-value table PVT_COMPACT_400, running

```
./3seq -xrun mysequenefile.aln
```

will compute and skip the same exact triples as the `quickrun` command

```
./3seq -quickrun mysequenefile.aln 400 400
```

but the `xrun` mode will use considerably less memory.

If you have multiple `PVT_COMPACT_` files in the `3SEQ` 'p-value-table' directory, one will be chosen at random. Simply rename the ones you do not want to use to something like `pVT_COMPACT_` if you want `3SEQ` to ignore them during `xrun` mode.

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 the sequence file to the standard output.

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 -fullrun mysequenefile.aln 400 -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.

These options have been included in the program so you can run `3SEQ` in parallel. For example, if your computer has two processors, you can run

```
./3seq -fullrun mysequenefile.aln 400 -b0 -e4 -id 001
./3seq -fullrun mysequenefile.aln 400 -b5 -e9 -id 002
```

on a set of 10 sequences. Just make sure that you have 800 megabytes of memory free because the above two processes will each request 400 megabytes of memory from the operating system. 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.

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 -x mysequenefile.aln -f500 -l750
```

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

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

To test the first 500 nucleotides, use

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

7.5 -L

7.6 -p

Currently only works in `fullrun` mode. Use this when 3SEQ is being called by a python program, and 3SEQ will format the output as a tab-delimited string. Currently, there are only two numbers in this string. If recombination was found, the string

```
-99    -99
```

will be returned. If no recombination was found, the string

```
p-value    0
```

will be returned. Use this option when performing power analyses.

7.7 -r

7.8 -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 -x 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 -x mysequenefile.aln -t1e-25
```

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

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

7.9 -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.

7.10 -#

7.11 -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.

7.12 -bp-none

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

7.13 -hs

Uses the Hogan-Siegmund approximation to obtain p-values for those triplets whose p-values could not be computed any other way. This option is automatically enabled in `fullrun` mode.

7.14 -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 -x mysequenefile.aln -id FLU2007
```

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

7.15 -nexus

Use in `write` mode. The command

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

will convert your file to nexus format.

7.16 -subset

7.17 -subset-remove

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-[39]     D2Mexic/?-[63]     D2PRico-va/69-[18]
```

where the number in brackets at the end of every sequence name is the index of that sequence in the dataset. This is included so that you can get breakpoint details on the above triplet by running

```
./3seq -t den2.aln  D2-D87-1040/87  D2Mexic/?  D2PRico-va/69
```

which will create a file called `3s.triplet.39.63.18.eps` which you can view to look at the positions of informative sites in that sequence triplet.

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 triple
7. p : p-value for this triple if the p-value was computed exactly or with Hogan-Siegmund approximation ; if bounds were obtained for this p-value, then this column will hold the lower bound of the p-value range.
8. $p_{\max}$: upper bound of the p-value range if the p-value was approximated; if the p-value was calculated exactly or with Hogan-Siegmund approximation, this column will hold the number -99.
9. 1 if Hogan-Siegmund approximation was used for this triple; 0 otherwise
10. for exact or Hogan-Siegmund computations: $\log_{10}(p)$; for approximate computations: $\log_{10}(p_{\max})$.
11. for exact or Hogan-Siegmund computations: Dunn-Sidak correction of p; for approximate computations: Dunn-Sidak correction of $p_{\max}$; in decimal format.
12. for exact or Hogan-Siegmund computations: Dunn-Sidak correction of p; for approximate computations: Dunn-Sidak correction of $p_{\max}$; in mantissa-exponent format.

13+. columns 13 and higher will hold the possible 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.

8.3 3s.skippedpvals

This is only created if the `-r` option is used. Shows the triplets for which p-value computations were skipped. This file will have 6 columns which will be the same as the first 6 columns of the `3s.rec` file.