Epos 1.7 : Estimating Population Sizes and Allele Ages from Site Frequency Spectra

Bernhard Haubold
Max-Planck-Institute for Evolutionary Biology, Plön, Germany

August 27, 2019

Contents

1 Introduction

The software package epos contains two programs, epos itself and the auxiliary program epos2ages. Epos estimates historical population sizes and epos2ages transforms them into allele ages. In this document I first explain epos and how to build the programs of the package. Then I give a tutorial-style introduction to their usage.

Epos takes site frequency spectra as input. A site frequency spectrum is computed from a haplotype sample. For example, Table 1 shows a sample of $n = 4$ haplotypes, $h_1, h_2, ..., h_4$, with $S = 8$ segregating (polymorphic) sites, $s_1, s_2, ..., s_8$. Each segregating site consists of a column of four zeros and ones, where zero indicates the ancestral state and one a mutation. We can count the number of sites where one, two, or three haplotypes are mutated. This is called the site frequency spectrum (SFS) of the sample, and Table 2A shows the spectrum for our example data. There are seven mutations affecting a single haplotype (singletons), zero mutations affecting two haplotypes (doubletons), and one mutation affecting three haplotypes (tripleton). In many empirical data sets it is not possible to distinguish between segregating sites with r mutant alleles and those with $n - r$ mutant alleles. In this case the spectrum is called *folded* and consists of the number of sites affecting r haplotypes plus the number of sites affecting $n - r$ haplotypes. Table 1B shows the folded version of the spectrum in Table 1A: The singleton category now consists of the sum of unfolded singletons and tripletons, while the number of doubletons remains unchanged.

Table 1: Four example haplotypes

haplotype	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
h_1	1	0	0	0	0	0	0	0
h_2	0	1	0	0	1	0	0	1
h_3	0	1	0	1	0	1	0	0
h_4	0	1	1	0	0	0	1	0

Table 2: Folded (**A**) and unfolded (**B**) site frequency spectrum corresponding to the haplotype sample shown in Table 1

A

r	$f(r)$
1	7
2	0
3	1

B

r	$f(r)$
1	8
2	0

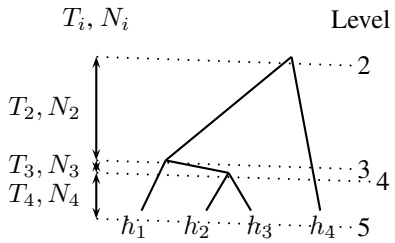

Figure 1: A coalescent for $n = 4$ haplotypes to illustrate the time intervals, T_i, the corresponding population sizes, N_i, and the *levels* in the tree.

`Epos` implements theory by Lynch et al. (2019), which I briefly summarize here. Consider the coalescent for n haplotypes, Figure 1 shows an example for $n = 4$ haplotypes. Any such coalescent can be divided into levels $2, 3, ..., n$, where a level, ℓ, is marked by the coalescence event that reduces the number of lineages from $\ell + 1$ to ℓ. The time intervals $T_i, 2 \leq i \leq n - 1$, denote the segments of the coalescent with i lineages. For each T_i `epos` can calculate a corresponding number of diploid individuals making up the population, N_i. Unfortunately, for larger samples the computation of all N_i invariably returns negative population sizes. So instead of allowing the population size to change at every coalescence event, we pick a subset of levels encapsulating the most important size changes.

Each combination of levels must contain level 2, the root. If no further level is added, N_2 is computed, the population size for the whole coalescent. Similarly, breakpoints at levels 2 and 4 would mean that there is one population size for T_2 and T_3, and another for T_4.

So, given a combination of levels, the population sizes are computed with the proviso that negative values are set to the smallest possible population size, $N = 1$. For each combination of levels and population sizes, the log-likelihood of observing the input site frequency spectrum is calculated (Lynch et al., 2019). We now need an efficient a method for maximizing this likelihood.

The starting point is always a single level, $m = 1$, the root at level 2. Then all level pairs $(2, 3), (2, 4), ..., (2, n)$ are examined and the most likely combination picked. If this improves the likelihood by at least 2 units, the number of levels is increased to $m = 3$ and the search repeats. Algorithm 1 summarizes this strategy. The function `nextConfig` in line 7 hides the details of how the next configuration is picked. I have implemented a *greedy* and an *exhaustive* strategy. Under the greedy strategy, the best configuration of breakpoints found in round m is retained in round $m + 1$ and a single new breakpoint is added. Under the exhaustive strategy, all $\binom{n-1}{m-2}$ possible combinations of breakpoints are investigated in each round. Unfortunately, for large samples, this number quickly leads to unacceptable run times and hence greedy is the default. The greedy and exhaustive search strategies are explained further in the Tutorial in Section 3, where I also apply `epos` to real site frequency spectra before also explaining `epos2ages`.

2 Getting Started

The `epos` package was written in C and Go on a computer running Linux. It depends on two libraries, the Gnu Scientific Library (`lgsl`), and the Basic Linear Algebra Subprograms (`lblas`). Please contact `haubold@evolbio.mpg.de` if there are any problems with the program.

Algorithm 1 Searching for break points in the coalescent

Require: n {Sample size}
Require: f {Array of $n-1$ or $n/2$ site frequencies, the unfolded or folded site frequency spectrum, SFS}
Ensure: m {Number of estimated population sizes}
Ensure: k {Array of m population size change points}
Ensure: N {Array of m population sizes at change points $\mathrm{k}[i], i = 1, 2, ..., m$}

$m \leftarrow 1$ {Initialize to one population size...}
$\mathrm{k}[m] \leftarrow 2$ {...which starts at the root}
$\mathrm{N} \leftarrow \texttt{popSizes}(m, \mathrm{k}, \mathrm{f})$ {Size of constant population}
$l \leftarrow \texttt{logLik}(\mathrm{N}, m, \mathrm{k}, \mathrm{f})$ {Log-likelihood of population size given the SFS}
$l_\mathrm{a} \leftarrow l$ {The initial log-likelihood is also the maximum}
for $m \leftarrow 2$ to n **do**
  while $(\mathrm{k}' \leftarrow \texttt{nextConfig}(m, n)) \neq$ null **do**
    $\mathrm{N}' \leftarrow \texttt{popSizes}(m, \mathrm{k}', \mathrm{f})$
    $l' \leftarrow \texttt{logLik}(\mathrm{N}', m, \mathrm{k}', \mathrm{f})$
    if $l' > l_\mathrm{a}$ **then**
      $\mathrm{k_a} \leftarrow \mathrm{k}'$
      $\mathrm{N_a} \leftarrow \mathrm{N}'$
      $l_\mathrm{a} \leftarrow l'$
    end if
  end while
  if $l_\mathrm{a} < l + 2$ **then**
    $\texttt{report}(\mathrm{N}, \mathrm{k}, m-1)$
    break
  end if
  $\mathrm{k} \leftarrow \mathrm{k_a}$
  $\mathrm{N} \leftarrow \mathrm{N_a}$
  $l \leftarrow l_\mathrm{a}$
end for

- Obtain the package

  ```
  git clone https://www.github.com/evolbioinf/epos
  ```

- Change into the directory just downloaded

  ```
  cd epos
  ```

  and make the programs

  ```
  make
  ```

- Test the programs

  ```
  make test
  ```

  which takes approximately one minute.

- The executables are located in the directory `build`. Place them in your `PATH`.

- Make the documentation

  ```
  make doc
  ```

  This calls the typesetting program `latex`, so please make sure it is installed before entering this command. The typeset documentation is located in

  ```
  doc/epos.pdf
  ```

3 Tutorial

3.1 Epos

I first explain how to test `epos` using simulated data, and then analyze real data.

Simulated Data

`Epos` was developed for estimating variable population sizes. Nevertheless, we begin with simulated constant-size scenarios before generating samples under models with varying population sizes.

Constant Population Size

- First, take a look at one of the simulated data sets supplied with `epos`:

  ```
  cat data/testF.dat
  #r f(r)
  1 5166
  2 2030
  3 1383
  4 1116
  5 981
  6 874
  7 702
  8 654
  9 593
  10 573
  ```

```
11 532
12 494
13 463
14 593
15 236
```

It is a folded data set, so the sample size $n = 30$.

- Let's analyze it using the command

```
epos -l 1e7 -u 1.2e-8 data/testF.dat
```

where `-l 1e7` indicates that 10 Mb of sequence were surveyed and $\mu = 1.2 \times 10^{-8}$ is the mutation rate per site per generation. The result of this command is

```
#InputFile: data/testF.dat
#Polymorphic sites surveyed:      24384
#Monomorphic sites surveyed:    9975616
#m = 1; maximum Log(Likelihood):   -1719.151896 {2}
#m = 2; maximum Log(Likelihood):    -106.137405 {2,11}
#m = 3; maximum Log(Likelihood):    -104.753557 {2,4,11}
#Final Log(Likelihood):             -106.137405
#d^2: 0.00262777
#Level  T[Level]        N[Level]
11      6.69e+03        2.51e+04
2       3.45e+04        7.73e+03
```

`Epos` prints intermediate configurations of m breakpoints: As already described in Algorithm 1, it starts with a single breakpoint, $m = 1$, the root at at level 2. The root level must always be present and the population size starting there extends to the next breakpoint printed, or, if there is none as in this case, the leaves of the coalescent. The log-likelihood of the input site frequency spectrum given the best-fitting constant population size is -1719.15. In the next round, $m = 2$, one new breakpoint is added at level 11. The log-likelihood increases to -106.14, which is 1613 log-units greater than the previous log-likelihood. As the minimum improvement is 2 log-units, the search is continued in round $m = 3$. The breakpoint at level 11 from the previous round is retained and the next best breakpoint, level 4, is added. The log-likelihood grows by roughly one units, so the search is abandoned and the population sizes for the levels 2 and 11 are printed, `N[Level]`. This search strategy is *greedy*, because it cannot revise the level configurations found in previous rounds.

- Instead of searching for the optimal set of breakpoints, these can be supplied in the same notation as used for the intermediate results using the `-L` option.

```
epos -l 1e7 -u 1.2e-8 -L 2,4,11 data/testF.dat
#InputFile: data/testF.dat
#Polymorphic sites surveyed:      24384
#Monomorphic sites surveyed:    9975616
#Final Log(Likelihood):             -104.753557
#d^2: 0.00249224
#Level  T[Level]        N[Level]
11      6.80e+03        2.55e+04
4       1.32e+04        6.84e+03
2       3.51e+04        8.23e+03
```

- In addition to the population size at each level added to the model, `epos` prints the time of the levels in generations, `T[Level]`. If we call T_k the time the coalescent spends with k lines, then

$$\texttt{T[Level]} = \sum_{k=\texttt{Level}}^{n} E[T_k],$$

where N_k is the population size during T_k, allows the computation of `T[Level]` as a simple function of the population size.

- In addition to the log-likelihood, a goodness-of-fit measure is printed, d^2, as suggested by Lapierre et al. (2017). This is the sum of squared differences between the observed site frequency spectrum and the spectrum implied by the estimated population sizes. The observed and expected site frequency spectra can be printed using the `-o` option

```
epos -l 1e7 -u 1.2e-8 -o data/testF.dat
...
#mark  r   o     e            nor(o)     nor(e)
#sfs   1   9526  9589.046040  0.390666  0.393057
#sfs   2   3998  3939.189168  0.163960  0.161468
...
#sfs   15  472   495.190548   0.019357  0.020298
```

  where `o` and `e` are the observed and expected values, and `nor(.)` their normalized versions.

- d^2 is computed from the normalized observed and expected spectra

```
epos -l 1e7 -u 1.2e-8 -o data/testF.dat |
grep '^#sfs'                            |
awk '{d=$5-$6;d2+=d*d/$6}END{print d2}'
0.002628
```

- Under the exhaustive search all possible combinations of levels are tested in each round. This mode is switched on using `-E` n, meaning up to n rounds of exhaustive search are conducted before the program reverts to the greedy strategy, or quits. Apply the exhaustive strategy to the data set just analyzed:

```
epos -l 1e7 -u 1.2e-8 -E 30 data/testF.dat
#InputFile: data/testF.dat
#Polymorphic sites surveyed:      24384
#Monomorphic sites surveyed:    9975616
#m = 1; maximum Log(Likelihood):   -1719.151896 {2}
#m = 2; maximum Log(Likelihood):    -106.137405 {2,11}
#m = 3; maximum Log(Likelihood):     -99.505349 {2,4,9}
#m = 4; maximum Log(Likelihood):     -97.138964 {2,5,12,13}
#m = 5; maximum Log(Likelihood):     -96.360180 {2,6,14,15,24}
#Final Log(Likelihood):              -97.138964
#d^2: 0.00185004
#Level  T[Level]        N[Level]
13      4.51e+03        2.26e+04
12      8.35e+03        1.27e+05
5       8.80e+03        7.01e+02
2       3.62e+04        9.15e+03
```

  The option `-E 30` indicates that up to 30 levels, that is all possible levels in the coalescent, can be added by exhaustive search. The first two rounds are always the same as under the greedy regime. However, at $m = 3$ level 11 from the previous round is dropped and levels 4 and 9 are added. The increase is roughly 6.7 units, which is more than the 2 units required for the acceptance of a new level, the search continues to $m = 4$. Here levels 4 and 9 from the previous round are replaced by levels 5, 12, and 13 leading to a likelihood increase by 2.3 a fifth and final round of, $m = 5$. Here levels 5, 12, and 13 are replaced by 6, 14, 15, and 24, but the likelihood increase is less than one and, the search is stopped, and we get the result for $m = 4$.

- You will have noticed the markedly longer run time of the exhaustive search compared to its greedy version. The likelihood increase purchased by this increased effort is roughly 9 units, which we deem significant. So exhaustive searching can lead to better results, but in practice this strategy is so time consuming that we only use the greedy search for the remainder of this Tutorial.

- So far, we do not know what the true result of our analysis should be, so we cannot assess its quality. To do this, we need to simulate haplotypes under a given scenario. First, we simulate one sample with $n = 30$ haplotypes and constant population size using the coalescent simulator `ms` (Hudson, 2002):

  ```
  ms 30 1 -t 10
  ```

  This can automatically be converted to a site frequency spectrum using my program `ms2sfs`[1]:

  ```
  ms 30 1 -t 10 | ms2sfs
  ```

  Finally, `epos` reads the spectrum and estimates population sizes from it:

  ```
  ms 30 1 -t 10 | ms2sfs | epos -U -l 1000
  #InputFile: stdin
  #Polymorphic sites surveyed:         55
  #Monomorphic sites surveyed:        945
  #m = 1; maximum Log(Likelihood):     -52.219407 {2}
  #m = 2; maximum Log(Likelihood):     -43.733126 {2,9}
  #m = 3; maximum Log(Likelihood):     -42.565671 {2,3,9}
  #Final Log(Likelihood):              -43.733126
  #d^2: 2.15424
  #Level  T[Level]        N[Level]
  9       5.66e+05        1.54e+06
  2       1.43e+06        2.47e+05
  ```

  where `-U` indicates an unfolded site frequency spectrum. Since this is a simulation, your results from now on are bound to differ from mine in numerical detail, though not in the general trend.

- `Epos` interprets the population mutation parameter, $\theta = 4N_e\mu$, as per-site, which means that under constant population size the expected effective size is

  $$E[N_e] = \frac{\theta}{4\mu l}.$$

  Since `epos` uses by default $\mu = 5 \times 10^{-9}$, the expected population size for our simulation data is 500,000. We can test this by simulating 1000 samples, calculating a site frequency spectrum for each, and averaging over the estimated population sizes:

  ```
  ms 30 1000 -t 10 |  # generate 1000 samples of 30 haplotypes
  ms2sfs           |  # compute one spectrum per sample
  epos -U -l 1000  |  # estimate population sizes for each spectrum
  grep -v '^#'     |  # remove hashed lines
  awk '{s+=$3;c++}END{print s/c}' # compute average population size
  1.2362e+06
  ```

  This is more than twice as large as the expected 500,000 and illustrates that the average is not a good statistic for monitoring the "majority" behavior of `epos` estimates. What about the median? We repeat the simulation, but this time save the population sizes to the file `tmp.txt`:

  ```
  ms 30 1000 -t 10 |  # generate 1000 samples of 30 haplotypes
  ms2sfs           |  # compute one spectrum per sample
  epos -U -l 1000  |  # estimate population sizes for each spectrum
  grep -v '^#'     |  # remove hashed lines
  cut -f 3  > tmp.txt # extract the third column and save it
  ```

[1] `https://www.github.com/evolbioinf/sfs`

Now we count the results:

```
wc -l tmp.txt
1598 tmp.txt
```

and look up the middle of the sorted estimates, the median:

```
sort -g tmp.txt | head -n 799 | tail -n 1
4.67e+05
```

which is reasonably close to the expected value of 5×10^5. Let's see if folding the spectrum changes this result:

```
ms 30 1000 -t 10 |
ms2sfs -f        |  # fold the spectra
epos -l 1000     |  # eliminate the -U option
grep -v '^#'     |
cut -f 3         |
sort -g > tmp.txt
```

Again, count the results

```
wc -l tmp.txt
1563 tmp.txt
```

and look up the median

```
head -n 781 tmp.txt | tail -n 1
4.42e+05
```

As before, this is similar to the expected 500,000.

Variable Population Size

- For estimating variable population sizes, we need to plot size as a function of time. This is implemented in the program `epos2plot`[2]. Begin again by simulating samples under constant size, but this time add more polymorphisms and include, for increased realism, recombination. Since recombination slows down `ms`, we replace it by its faster re-implementation, `mspms` (Kelleher et al., 2016). The parameters for this `mspms` run are taken from (Liu and Fu, 2015, Figure 2a).

```
mspms 30 1000 -t 4800 -r 3800 10000000 |
ms2sfs                                 |
epos -l 1e7 -U -u 1.2e-8               |
epos2plot > epos1.dat # generate data ready for plotting
```

This takes a few minutes. Draw the graph by applying the program `gnuplot`[3] to the file `fig1.gp`, which is part of the `epos` package:

```
gnuplot -p scripts/fig1.gp
```

to get Figure 2. The fit between the estimated median size and the true size is good for most of the plot, except for the very recent past.

[2] `https://www.github.com/evolbioinf/epos2plot`
[3] `http://www.gnuplot.info/`

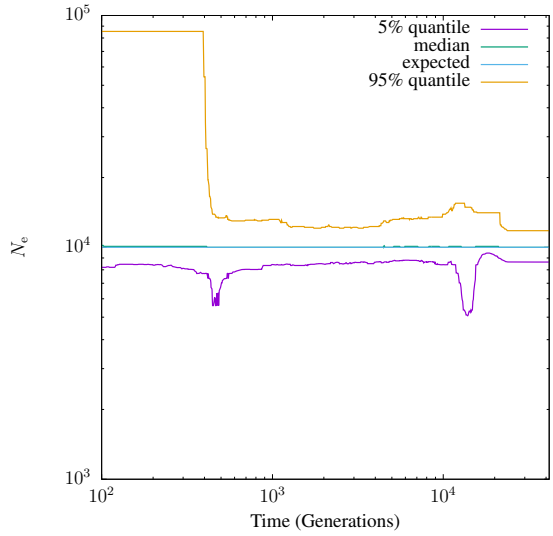

Figure 2: Estimating constant population size. For details see text.

- Next, we simulate under the scenario used in Figure 2b of Liu and Fu (2015) with a single, approximately threefold change in population size from 7778 to 25636, which occurred 6809 generations ago:

```
mspms 30 1000 -t 12310 -r 9750 1e7 -eN 0.066 0.3 |
ms2sfs                                           |
epos -u 1.2e-8 -l 1e7 -U                         |
epos2plot > epos2.dat
```

  This again takes a few minutes.

- Plot the result

```
gnuplot -p scripts/fig2.gp
```

  to get Figure 3. The fit between the median estimated population size and its true value remains excellent. However, the variation in estimates is again large, particularly toward the present.

- As a last example, simulate haplotypes under the exponential growth scenario Liu and Fu (2015) used in their Figure 2e:

```
mspms 30 1000 -t 432000 -r 340000 1e7 -G 46368 -eN 0.0001027 0.008889 |
ms2sfs                                                                 |
epos -u 1.2e-8 -l 1e7 -U                                               |
epos2plot > epos3.dat
```

  Plot this

```
gnuplot -p fig3.gp
```

  to get Figure 4, where the estimation is precise until very close to the present, when it starts to diverge. This illustrates the difficulty of accurately calculating population size changes in the recent past.

Real Data

I analyze the two site frequency spectra distributed with `epos`. One is from a population of water flea, *Daphnia pulex*, the other from a human population, the Yoruba, who live in south-western Nigeria. The *Daphnia* data is provided by Mike Lynch (Arizona State University), the Yoruba spectrum was computed by Lapierre et al. (2017) from the 1000 human genomes project data.

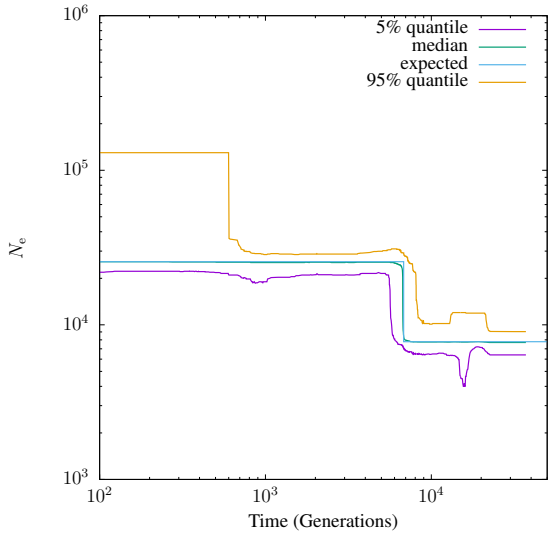

Figure 3: Estimating population sizes under a model with one instantaneous size change. See text for details.

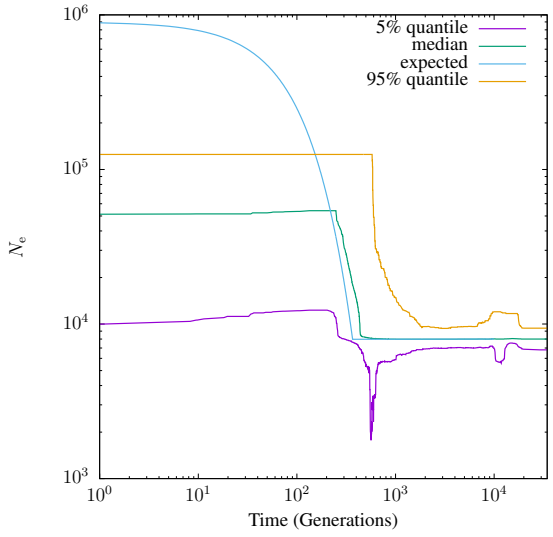

Figure 4: Estimating population sizes under an exponential growth model. See text for details.

Daphnia pulex

- As an example for real data, we use a site frequency spectrum obtained from the Kap population of *Daphnia pulex*:

```
head data/kap144i.dat
0 185297
1 1987
2 1138
3 851
4 729
5 672
6 542
7 509
8 459
9 430
```

Notice the "zero-class", which does not appear in the example spectra in Tables 2A and B. The zero-class gives the number of monomorphic sites. If a spectrum contains a zero-class, the sequence length is the sum of all allele counts, and `epos` does not require the sequence length to be passed via the `-l` option. Moreover, the site frequency spectrum of Kap is folded, hence no `-U`:

```
epos data/kap144i.dat
#InputFile: data/kap144i.dat
#Polymorphic sites surveyed:      18098
#Monomorphic sites surveyed:     185297
#m = 1; maximum Log(Likelihood):    -973.298394 {2}
#m = 2; maximum Log(Likelihood):    -306.995057 {2,24}
#m = 3; maximum Log(Likelihood):    -304.105038 {2,3,24}
#m = 4; maximum Log(Likelihood):    -297.264486 {2,3,8,24}
#m = 5; maximum Log(Likelihood):    -297.158028 {2,3,8,24,55}
#Final Log(Likelihood):             -297.264486
#d^2: 0.0039003
#Level  T[Level]        N[Level]
24      5.75e+04        3.93e+05
8       3.73e+05        7.94e+05
3       2.98e+06        1.82e+06
2       3.13e+06        7.89e+04
```

- A classical problem when estimating model parameters is "over-fitting". This refers to the fact that random quirks of a data set can strongly influence the estimation and hence generate a result specific to the particular data set but misleading with respect to the underlying population. We avoid over-fitting by requiring that a new level improves the log-likelihood of the model by at least 2 units (Algorithm 1). A popular alternative is k-fold cross-validation (Goodfellow et al., 2016, p. 118f). We can invoke this procedure with $k = 5$, a typical value, using

```
epos -k 5 data/kap144i.dat
...
#Final Log(Likelihood):             -297.30
#d^2: 0.003906
#Level  T[Level]        N[Level]
26      4.97e+04        3.76e+05
24      8.78e+04        2.74e+06
6       4.40e+05        5.63e+05
4       2.64e+06        4.12e+06
2       3.05e+06        1.55e+05
```

which is very similar to the result with the log-likelihood threshold.

- In many empirical data sets singletons are regarded as unreliable. To ignore singletons,

```
epos -x 1 data/kap144i.dat
...
#Final Log(Likelihood):             -309.175057
#d^2: 0.00546917
#Level  T[Level]        N[Level]
29      3.37e+04        2.93e+05
2       3.94e+06        1.01e+06
```

To ignore singletons and doubletons,

```
epos -x 1,2 data/kap144i.dat
#InputFile: data/kap144i.dat
#Polymorphic sites surveyed:      14973
#Monomorphic sites surveyed:     185297
#m = 1; maximum Log(Likelihood):    -543.807666 {2}
#m = 2; maximum Log(Likelihood):    -310.507644 {2,39}
#m = 3; no improvement
#Final Log(Likelihood):             -310.507644
#d^2: 0.00634133
#Level  T[Level]        N[Level]
39      7.75e-02        1.00e+00
2       3.92e+06        1.01e+06
```

which does not look like a very convincing result.

- A run of `epos` only gives a single point estimate, and in the absence of further samples it is hard to judge its reliability. However, the program `bootSfs`, which is also part of the `sfs` package bootstraps site frequency spectra to assess the robustness of results based on single samples. To run 1000 bootstrap replicates, enter

```
bootSfs -i 1000 kap144i.dat | epos | epos2plot > epos4.dat
```

which takes about five minutes. Plot the result

```
gnuplot -p fig4.gp
```

to get Figure 5.

The Yoruba Population

- Analyze the full spectrum:

```
epos -u 1.2e-8 -l 2.9e9 data/FoldSFS_YRI.txt
#InputFile: data/FoldSFS_YRI.txt
#Polymorphic sites surveyed:   20440078
#Monomorphic sites surveyed: 2879559922
...
#Final Log(Likelihood):            -1177.60
#d^2: 4.38702e-05
#Level  T[Level]        N[Level]
216     8.76e+01        1.02e+06
142     4.21e+02        3.41e+04
121     4.24e+02        6.75e+02
```

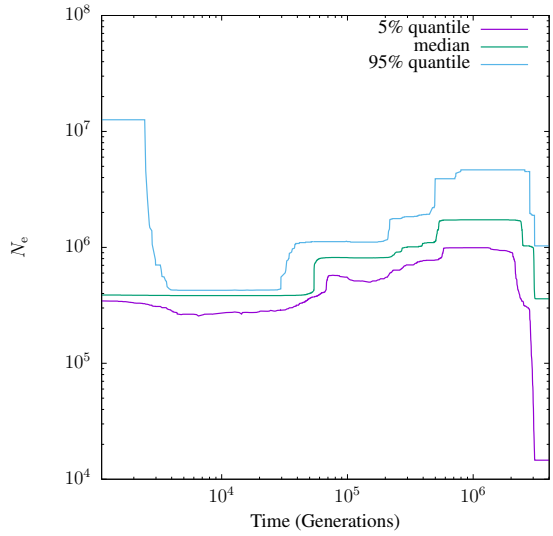

Figure 5: Estimating the population size of the *D. pulex* Kap population.

```
13        9.13e+03        2.90e+04
10        1.18e+04        2.43e+04
3         4.94e+04        2.42e+04
2         7.07e+04        1.06e+04
```

where $\mu = 1.2 \times 10^{-8}$ is the mutation rate used by Lapierre et al. (2017) and 2.9×10^9 the number of sequenced nucleotides in the human genome.

- Lapierre et al. (2017) excluded the singletons in their analysis. To do this in `epos`, we rerun the analysis with `-x 1`

```
epos -x 1 -u 1.2e-8 -l 2.9e9 data/FoldSFS_YRI.txt
#InputFile: data/FoldSFS_YRI.txt
#Polymorphic sites surveyed:   15937781
#Monomorphic sites surveyed: 2879559922
...
#Final Log(Likelihood):            -3132.160271
#d^2: 8.04994e-05
#Level  T[Level]        N[Level]
181     3.70e-03        1.00e+00
28      3.51e+03        2.79e+04
6       2.23e+04        2.88e+04
5       2.30e+04        3.59e+03
2       7.61e+04        1.77e+04
```

- As with the *Daphnia* data, we can bootstrap the Youruba mutation spectrum. Figure 6A shows the demography based on 10^4 bootstrap samples including all allele classes and assuming 24 years per generation. The apparent jump from 3×10^4 to 10^6 approximately 2000 years ago is abolished by excluding singletons (Figure 6B). In contrast, the baseline size of approximately 3×10^4 remains unchanged. The bootstrap analysis including all mutation classes takes approximately 15 CPU hours, a bit less if singletons are excluded. Such massive run times are best managed by starting, say, 50 jobs with 200 resamplings on a multi-core machine. `Epos` lends itself to this kind of pseudo-parallelization, as its memory consumption is only 4.3 MB per run on the Yoruba sample, which is negligible on current computers.

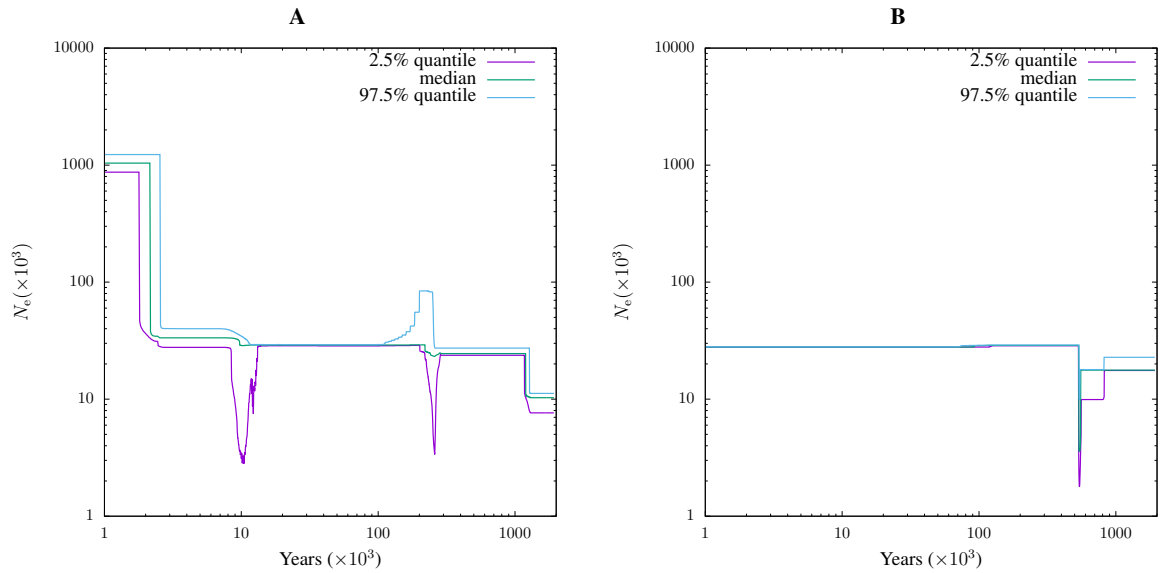

Figure 6: Analysis of the Yoruba site frequency spectrum including the singletons (**A**) and excluding them (**B**).

3.2 `Epos2ages`

The program `epos2ages` converts population sizes computed by `epos` to average ages of alleles (Lynch et al., 2019).

- To start with a simple example, simulate a site frequency spectrum with $n = 2$,

  ```
  ms 2 1 -t 10 | ms2sfs > test.sfs
  ```

  estimate the population size,

  ```
  epos -U -l 1000 test.sfs
  #InputFile: test.sfs
  #Polymorphic sites surveyed:          36
  #Monomorphic sites surveyed:         964
  #m = 1; maximum Log(Likelihood):      -7.07 {2}
  #m = 2; no improvement
  #Final Log(Likelihood):               -7.07
  #d^2: 0
  #Level T[Level] N[Level]
  2 3.60e+06 1.80e+06
  ```

  and the age of singletons

  ```
  epos -U -l 1000 test.sfs | epos2ages -n 2
  #r  A[r]     V(A[r])     P[r]
  1   3.6e+06  1.296e+13  1.8e+06
  ```

  where the second column gives their average age as 3×10^6, twice the population size. Since `epos2ages` follows the convention of measuring time in units of $2N$ generations, this is the correct result. The other two output columns of `epos2ages` are the variance of the age, and the average population size experienced by the allele, which again agrees with the previous `epos` result.

- For larger samples of 30 haplotypes the population sizes might look like this

```
ms 30 1 -t 1000 | ms2sfs | tee test.sfs | epos -U -l 1e7
#InputFile: stdin
#Polymorphic sites surveyed:       3055
#Monomorphic sites surveyed:    9996945
#m = 1; maximum Log(Likelihood):   -1520.427498 {2}
#m = 2; maximum Log(Likelihood):   -1162.883232 {2,4}
#m = 3; maximum Log(Likelihood):   -1156.803456 {2,4,16}
#m = 4; no improvement
#Final Log(Likelihood):            -1156.803456
#d^2: 1.13251
#Level  T[Level]        N[Level]
16      6.08e+02        4.56e+03
4       8.24e+03        7.15e+03
2       8.24e+03        1.00e+00
```

and the corresponding allele sizes

```
epos -U -l 1e7 test.sfs | epos2ages -n 30
#r      A[r]    V(A[r])         P[r]
1       1713.61 5.28278e+06     6467.4
2       2602.4  7.05225e+06     6596.96
3       3236.13 8.04631e+06     6678.69
...
29      8237.33 9.83744e+06 6956.52
```

- Finally, we can estimate the average age of alleles for the Kap population:

```
epos2ages -n 144 data/kap144i.out
#r      A[r]    V(A[r])      P[r]
1       232151  4.18264e+11  1.38013e+06
2       382824  6.51857e+11  1.4013e+06
3       493525  8.05644e+11  1.41684e+06
...
143     3.13086e+06  2.09611e+12  1.60264e+06
```

4 Change Log

The change log can be accessed via the repository web page

`https://github.com/evolbioinf/epos`

or inside a local copy of the repository by executing

```
git log
```

References

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, Cambridge, Massachusetts, 2016.

R. R. Hudson. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18: 337–338, 2002.

J. Kelleher, A. Etheridge, and G. McVean. Efficient coalescent simulation and genealogical analysis for large sample sizes. *PLoS Comput. Biol.*, 12:1–2, 2016.

M. Lapierre, A. Lambert, and G. Achaz. Accuracy of demographic inferences from the site frequency spectrum: the case of the Yoruba population. *Genetics*, 206:439–449, 2017.

X. Liu and Y.-X. Fu. Exploring population size changes using SNP frequency spectra. *Nature Genetics*, 47: 555–562, 2015.

M. Lynch, B. Haubold, P. Pfaffelhuber, and T. Maruki. Inference of historical population-size changes with allele-frequency data. *Submitted*, 2019.