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

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

September 24, 2018

1 Introduction

`Epos` estimates historical population sizes based on site frequency spectra. A site frequency spectrum is computed from a haplotype sample. Table 1 shows a sample of $n = 4$ haplotypes, h_1–h_4, with $S = 8$ segregating (polymorphic) sites, s_1–s_8. Each segregating site consists of a column of four zeros and ones, where the zeros indicate the ancestral state and the ones 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 mutations and those with $n - r$ mutations. In this case the spectrum is *folded* by adding the number of sites affecting r haplotypes to the number of sites affecting $n-$ haplotypes. Table 1B shows the folded version of the spectrum in Table 1A: The single tripleton is added to the seven singletons, while the number of doubletons remains unchanged. In addition to population sizes, `epos` can also compute the average ages of mutations affecting $1, 2, ..., n - 1$ haplotypes.

`Epos` implements theory developed by Michael Lynch (Arizona State University) and Peter Pfaffelhuber (Freiburg University). The theory is similar to the method developed by Liu and Fu (2015) and will be described in a forthcoming paper.

2 Getting Started

`Epos` was written in C 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.

- Change into the `epos` directory

  ```
  cd epos
  ```

  and list its contents

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

```
ls
```

- Generate `epos`

```
make
```

- List its options

```
./epos -h
```

- Run tests

```
bash scripts/test.sh
```

3 Tutorial

I first explain how to test `epos` using simulated data, and then how it can be applied to real data.

3.1 Simulated Data

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

3.1.1 Constant Population Size

- We simulate samples with $n = 30$ haplotypes using the well-known coalescent simulator `ms` (Hudson, 2002):

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

These can automatically be converted to site frequency spectra using my program `sfs`, which is available form the same `github` page as (`evolbioinf`):

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

This spectrum is used by `epos` to estimate population sizes:

```
ms 30 1 -t 10 | sfs | epos -U -l 1000
#InputFile: stdin
#Polymorphic sites surveyed: 48
#Monomorphic sites surveyed: 952
#LogLik: 5581.73
#Level  T[Level]  N[T]
2       2.34e+06  6.06e+05
```

where `-U` an unfolded site frequency spectrum, and `-l` is the sequence length, $l = 1000$. `Epos` interprets the population mutation parameter, $\theta = 4N_\mathrm{e}\mu$, as per-site, which means that under constant population size the expected effective size is

$$E[N_\mathrm{e}] = \frac{\theta}{4\mu l}.$$

Since `epos` uses by default $\mu = 5 \times 10^{-9}$, the expected population size for our simulation data is therefore 500,000. We can check 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
sfs -i           |                # compute spectra for individual samples
epos -U -l 1000  |                # estimate population sizes
grep -v '^#'     |                # remove hashed lines
awk '{s+=$3;c++}END{print s/c}' # compute average population size
502000
```

which is close to the expected 500,000. Let's see if folding the spectrum changes this result by adjusting the options for `sfs` and `epos`:

```
ms 30 1000 -t 10 |
sfs -f -i        |  # fold the spectra of individual samples
epos -l 1000     |
grep -v '^#'     |
awk '{s+=$3;c++}END{print s/c}'
481000
```

This is again very similar to the expected value of 500,000.

3.1.2 Variable Population Size

- For estimating variable population sizes we need the program `epos2plot` to plot size as a function of time. `Epos2plot` is also available form the `evolbioinf` page on `github`. We begin again by simulating samples under constant population size, but this time add plotting

  ```
  ms 30 1000 -t 10 |
  sfs -i           |
  epos -U -l 1000  |
  epos2plot > epos1.dat # generate data ready for plotting
  ```

  Draw the graph by applying the program `gnuplot`[1] to the file `fig1.gp`, which is part of the `epos` package:

  ```
  gnuplot -a fig1.gp
  ```

  to get Figure 1. The fit between the estimated median size and the true size is excellent for most of the plot, though it drifts upward in the very distant past. It is important to realize that the number of samples from which the estimates are derived declines into the past. Near the present we have

  ```
  head -n 5 epos1.dat
  #Time  LowerQ  Median  UpperQ    SampleSize
  0      25300   484000  1.81e+07  1000
  128    25300   484000  1.81e+07  1000
  324    26700   489000  1.81e+07  1000
  345    29800   492000  1.81e+07  1000
  ```

[1]`http://www.gnuplot.info/`

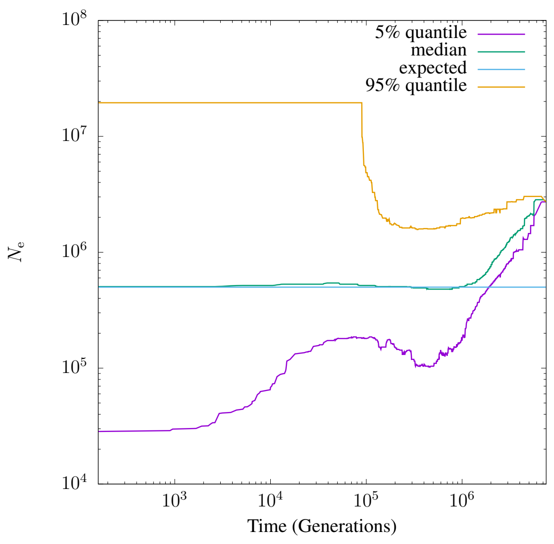

Figure 1: Estimating constant population size. For details see text.

while the furthest point in the past is usually computed from a single sample:

```
tail -n 5 epos1.dat
6.58e+06  2.34e+06  2.36e+06  2.71e+06  5
6.6e+06   2.34e+06  2.4e+06   2.71e+06  4
7.06e+06  2.34e+06  2.4e+06   2.71e+06  3
7.21e+06  2.4e+06   2.71e+06  2.71e+06  2
7.3e+06   2.71e+06  2.71e+06  2.71e+06  1
```

Also, the variation in Figure 1 is large, especially close to the present.

- Next, we simulate the scenario in Figure 2b of Liu and Fu (2015) with a single, approximately threefold change in population size from 7778 to 25636, which takes place 6809 generations in the past. Since this simulation includes substantial recombination, we replace `ms` by its fast re-implementation, `mspms` (Kelleher et al., 2016):

```
mspms 30 1000 -t 12310 -r 9750 10000000 -eN 0.066 0.3 |
sfs -i                                                |
epos -u 1.2e-8 -l 10000000 -U                         |
epos2plot > epos2.dat
```

  This takes about 11 minutes. Notice the `-u` option for `epos`, which specifies the mutation rate used by Liu and Fu (2015), $\mu = 1.2 \times 10^{-8}$. Plot the result

```
gnuplot -p fig2.gp
```

  to get Figure 2. 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, we simulate haplotypes under the exponential growth scenario Liu and Fu (2015) used in their Figure 2e:

```
mspms 30 1000 -t 432000 -r 340000 10000000 -G 46368 -eN 0.0001027 0.008889 |
sfs -i                                                                      |
epos -u 1.2e-8 -l 10000000 -U                                               |
epos2plot > epos3.dat
```

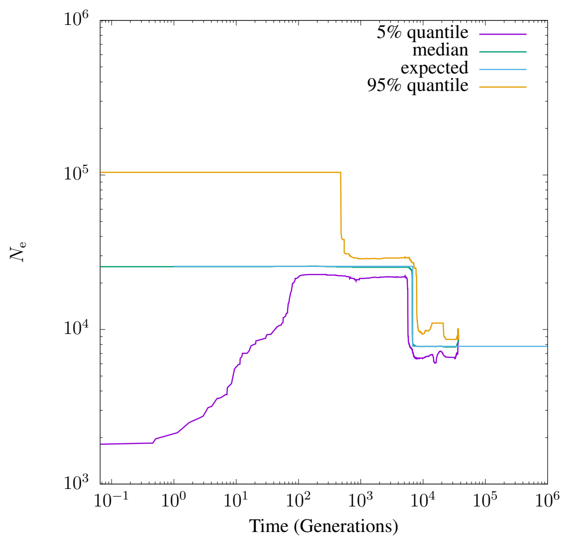

Figure 2: Estimating population sizes under a model with one instantaneous change. See text for details.

Plot this

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

to get Figure 3, where the estimation is less accurate than in the one-step scenario in Figure 2. This illustrates the difficulty of accurately capturing population size changes in the recent past.

3.2 Real Data

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

```
head 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 simply the sum of all counts, and there is no need to pass the sequence length to `epos` using `-l`. Moreover, the site frequency spectrum of Kap is folded, i. e. there is no need for `-U`. Hence we analyze our example data without any options:

```
epos kap144i.dat
#InputFile: ../data/kap144i.dat
#Polymorphic sites surveyed: 18037
```

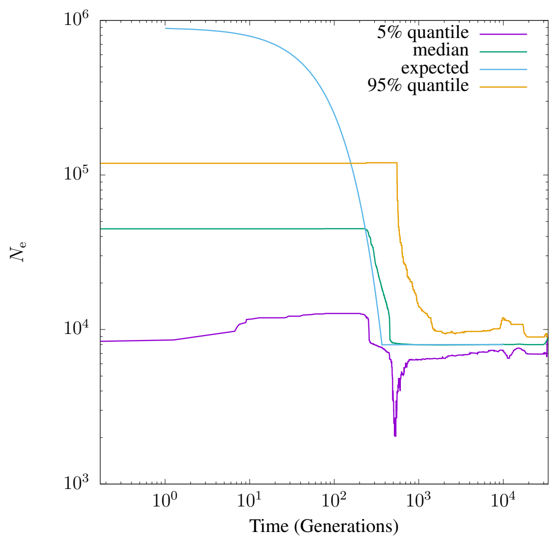

Figure 3: Estimating population sizes under an exponential growth model. See text for details.

```
#Monomorphic sites surveyed: 185297
#LogLik: 2.1501e+06
#Level  T[Level]  N[T]
25      5.18e+04  3.73e+05
2       3.90e+06  1.00e+06
```

- This only gives a single point estimate, and in the absence of further samples it is hard to judge its reliability. However, `epos` implements the bootstrap to investigate the robustness of results based on single samples. To run 1000 bootstrap replicates, enter

  ```
  epos -b 10000 kap144i.dat | epos2plot > epos4.dat
  ```

  This takes about five minutes. Plot the result

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

  to get Figure 4.

4 The Averge Age of an Allele

- The expected time to the most recent common ancestor is

$$E[T_{\mathrm{MRCA}}] = 2\left(1 - \frac{1}{n}\right),$$

  where time is measured in $2N$ generations (Wakeley, 2009, p.76). Now, if $n = 2$, $E[T_{\mathrm{MRCA}}] = 1$. We simulate a site frequency spectrum with $n = 2$,

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

  estimate the population size,

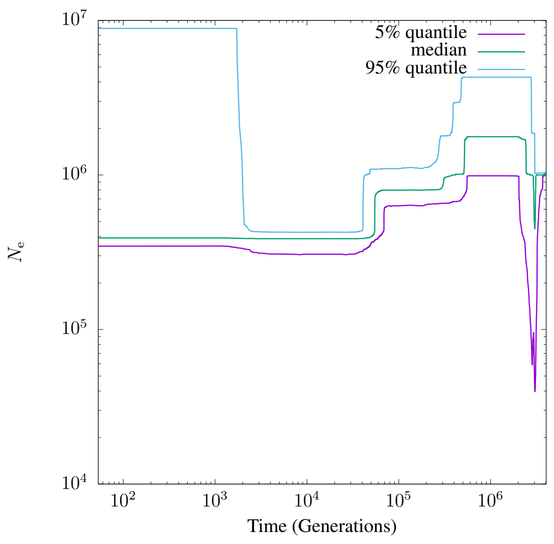

Figure 4: Estimating the population size of the *D. pulex* Kap population.

```
epos -U -l 1000 test.sfs
#InputFile: test.sfs
#Polymorphic sites surveyed: 15
#Monomorphic sites surveyed: 985
#LogLik: 5804.25
#Level   T[Level]   N[T]
2        1.50e+06   7.50e+05
```

and estimate the age of singleton alleles

```
epos -U -l 1000 -a test.sfs
#InputFile: test.sfs
#Polymorphic sites surveyed: 15
#Monomorphic sites surveyed: 985
#r  A[r]
1   1.5e+06
```

which is twice the population size, as expected, given that `epos` estimates time in units of generations.

- For larger samples the population sizes might look like this

```
ms 30 1 -t 1000 | sfs | tee test.sfs | epos -U -l 10000000
#InputFile: stdin
#Polymorphic sites surveyed: 4169
#Monomorphic sites surveyed: 9.99583e+06
#LogLik: 1.51132e+08
#Level   T[Level]   N[T]
16       1.30e+01   9.73e+01
8        3.62e+03   1.18e+04
2        1.94e+04   4.59e+03
```

and the corresponding allele sizes like that

```
epos -U -l 10000000 -a test.sfs
```

```
#InputFile: test.sfs
#Polymorphic sites surveyed: 4169
#Monomorphic sites surveyed: 9.99583e+06
#r  A[r]
1   3117.22
2   3741.8
3   4455.55
...
29  19359.7
```

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

```
epos -a .kap144i.dat | head
#InputFile: ../data/kap144i.dat
#Polymorphic sites surveyed: 18037
#Monomorphic sites surveyed: 185297
#r  A[r]
1   190499
2   309342
3   396787
...
72  2.68069e+06
```

5 Change Log

- Version 0.1 (Oct. 25, 2017)
  - First running version based on the GSL.
- Version 0.2 (Oct 26, 2017)
  - Used `LAPACKE_dgesv` in `getPopSizes`; makes no difference compared to the previous version.
- Version 0.3 (Oct 26, 2017)
  - Used `LAPACKE_dgesvx` in `getPopSizes`; makes no difference compared to previous version.
- Version 0.4 (Oct 27, 2017)
  - Allow construction of SFS based on `-t` switch.
  - Allow switching between the GSL algorithm (`-g`) and the default LAPACK algorithm.
- Version 0.5 (November 2, 2017)
  - Allow the straight usage of the trapezoid matrix for solving the system (`-T`).
  - Print out coefficient matrix (`-p`).
- Version 0.6 (November 10, 2017)
  - Use `long double` in `getPopSizesTri2`; did not help.
- Version 0.7 (November 10, 2017)
  - Implement Peter Pfaffelhuber's formula; working only partially.
- Version 0.8 (November 10, 2017)
  - Peter's equation (6) is working.

- Version 0.9 (November 10, 2017)
  - Peter's equation in arbitrary precision in MPFR library. Numerical stability achieved with 329 bits per number.
- Version 0.10 (November 30, 2017)
  - Implemented equation (3) from Peter's memo dated Nov. 13. Not working.
- Version 0.11 (December 1, 2017)
  - Implemented revised equation (3) from Peter's memo dated Nov. 30. Not working.
- Version 0.12 (December 15, 2017)
  - Implemented Estimator 2.1 from Peter's memo dated Dec. 2, 2017. Code is running, but the results look odd.
- Version 0.13 (December 16, 2017)
  - Fixed the implementation of Estimator 2.1; results OK now.
- Version 0.14 (December 18, 2017)
  - Implemented optimization strategy for folded SFS; working.
  - Implemented optimization strategy for unfolded/even SFS; working.
  - Implemented optimization strategy for folded/odd SFS; not working yet.
- Version 0.15 (December 18, 2017)
  - Fixed error in search for optimal number of steps.
- Version 0.16 (December 19, 2017)
  - Fixed error in left-hand side of folded/odd equation; working.
  - Changed search strategy.
  - Changed default value of `-d` from 10^{-6} to 10^{-3}.
  - Simplified user interface.
  - Included set of test cases (`test.sh`).
- Version 0.17 (December 20, 2017)
  - Fixed searching routine.
  - Changed default value of `-d` from 10^{-3} to 10^{-2}.
  - Included addition of the λ-factor; seems to make no difference.
- Version 0.18 (December 20, 2017)
  - Fixed the λ-factor; computation is now much stabilized.
- Version 0.19 (January 11, 2018)
  - Included the λ-factor in the computation of Ψ. Computations now applicable to real data.
  - Changed the default-value of λ from 10^{-7} to 10^{-5}.
- Version 0.20 (January 11, 2018)
  - Consider zero-class mutations in computation, if present.
  - Changed default λ to 2×10^{-5} to get all data sets to run.

- Version 0.21 (January 13, 2018)
  - Reverted output of levels, going from the present into the past.
  - Default output is now as a function of times instead as a function of levels.
  - Included "step-wise" option for plotting times and levels.
  - Fixed time computation.
  - Included error message for negative population sizes.
  - Removed memory leaks and other subtle bugs using `valgrind`.
- Version 0.22 (January 16, 2018)
  - Fixed important bug in function foldedEpsi, where variable `b` was computed as a function of `sfs->f[n/2]` instead of, now `sfs->f[n/2-1]`.
- Version 0.23 (January 16, 2018)
  - Search for optimal λ.
  - Allow arbitrary level as first entry in level list.
  - Output λ, Ψ, and the levels added to make it easier to follow the program.
  - Reduced program to folded/even case.
- Version 0.23 (January 17, 2018)
  - Catch GSL-exceptions.
  - Changed output format
- Version 0.24 (January 18, 2018)
  - Not quite sure what changed.
- Version 0.25 (January 18, 2018)
  - Fixed bug in computation of the λ-term in `foldedEpsi`.
- Version 0.26 (January 19, 2018)
  - Set $\lambda = 0$ and add levels until negative population sizes appear. This is fast and appears to be effective.
- Version 0.27 (January 24, 2018)
  - Output number of polymorphic and monomorphic sites surveyed.
- Version 0.28 (???)
- Version 0.29 (January 28, 2018)
  - Fixed missing resetting of times during iteration over files.
  - Removed superfluous option for step-wise output (`-s`).
  - Output name of input file.
  - Removed inclusion of `mpfr.h` from `epos.c`.
  - Removed search for the initial level to add; by definition this must be 2, i. e. one population size for the entire coalescent.
- Version 0.30 (January 31, 2018)
  - Fixed computation of the mutation rate. Previously I multiplied the per site mutation rate with the number of monomorphic positions. Now it is mutated by the number of all positions.

  - Added bootstrapping.

- Version 0.31 (February 1, 2018)
  - Computation of mutation rate was correct in previous version, after all, so reverted to that.
- Version 0.32 (February 7, 2018)
  - Introduced the `-m` switch.
- Version 0.33 (February 7, 2018)
  - Fixed δ computation in function `delta` in `util.c`. This reduces the computation for $m = 1$ to Watterson's estimator, as expected.
- Version 0.34 (February 9, 2018)
  - Reintroduced unfolded spectrum (`-u`) and compared to the equations in Peter's memo of December 19, 2017.
  - Reintroduced λ and set it by default to 10^{-7}.
  - Reintroduced δ and set it by default to 0.0.
  - Fixed sample size computation at the end of `sfs.c`.
- Version 0.35 (February 9, 2018)
  - Introduced estimation of $\lambda = 1./\mu/$`args->f`.
- Version 0.36 (February 14, 2018)
  - Changed setting of λ to $\lambda = \mu \times$ `args->f`. By default `args->f` $= 1$.
- Version 0.37 (February 16, 2018)
  - Fixed sample size computation for folded/even in function `getSfs` in `sfs.c`.
- Version 0.38 (February 16, 2018)
  - Included working `-m` switch.
- Version 0.39 (February 19, 2018)
  - Fixed passing of λ in iterated runs.
  - Fixed numerical underflow when multiplying with λ in `foldedEpsi` in `foldedE.c`.
  - Fixed numerical underflow when multiplying with λ in `psi` in `unfolded.c`
  - Included check for positive Ψ in both cases.
  - Expanded verbose output.
- Version 0.40 (February 21, 2018)
  - Fixed error in `foldedEpsi` in `foldedE.c`
  - Removed `if(m > 1)` from `getCoeffMat` in `foldedE.c`.
  - Switched `n/2` in `foldedEpsi` to `n/2.`.
  - Ensured that `sfs->u` is always set to a value in `getSfs`.
  - Replace `u = args->u` by `u = sfs->u` in `getCoeffMat` in `unfolded.c`.
  - Re-implemented `foldedEpsi` in `foldedE.c`
- Version 0.41 (February 22, 2018)

  - Changed `4.*u*(n/2.)` to `4.*u/(n/2.)` in `foldedEpsi` in `foldedE.c`. This was a bug in the computation of Ψ for the folded/even case.

- Version 0.42 (February 22, 2018)
  - Removed line `prevMinPsi = DBL_MAX` in `foldedE` in `foldedE.c`.
- Version 0.43 (February 22, 2018)
  - Added diagnostic output in case negative population sizes are found.
- Version 0.44 (February 22, 2018)
  - Ψ now also reported if only one level is included.
- Version 0.45 (February 23, 2018)
  - Included the `-n` option to allow negative population sizes.
- Version 0.46 (February 23, 2018)
  - Fixed `if(change > args->d)`-phrase in `unfolded` and `foldedE`. This lacked re-computation of the population sizes with the best new level added, and assignment of Ψ.
- Version 0.47 (February 26, 2018)
  - Reorganized code to remove duplication. The searching for best population sizes is now done in only one place, `getPopSizes` in `popSizes.c`.
  - Added printing of intermediate population sizes if the `-V` option is used.
- Version 0.48 (March 2, 2018)
  - Multi-threaded version.
- Version 0.49 (March 6, 2018)
  - Reverted to single-threaded behavior by removing `-t` from the options list and setting it to 1 in the background. This avoids the occasional race-conditions observed with the multi-threaded version.
- Version 0.50 (March 14, 2018)
  - Find number of levels through cross-validation (`-c`).
- Version 0.51 (March 14, 2018)
  - Find lambda through cross-validation (`-L`).
- Version 0.52 (March 17, 2018)
  - Include reporting of negative population sizes in verbose output (`-V`).
  - Changed `prevMinPsi = prevMinPsi;` in `getPopSizes` to `prevMinPsi = currMinPsi;`.
- Version 0.53 (April 7, 2018)
  - Always allow negative population sizes.
  - Cross-validation by default.
  - $\lambda = 0$ by default.
  - If negative population sizes are found, the program searches for optimal λ by going through $\lambda = 0..\mu$. This is slow and would need to be optimized in future versions.
  - Added Scripts for extracting quantiles from `epos` output.

- Version 0.54 (April 11, 2018)
  - Fixed array out-of-bounds error in `shuffleArr` in `sfs.c`.
- Version 0.55 (April 12, 2018)
  - "Unfolded" mode not working; so I removed that option for now.
- Version 0.56 (May 31, 2018)
  - Fixed Error in documentation.
- June 13, 2018
  - Posted `epos` on `github`. Please refer to the commit messages for details on subsequent changes.

References

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.

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

J. Wakeley. *Coalescent Theory: An Introduction*. Roberts & Company, Colorado, 2009.