SpectralTDF User Manual

1 Introduction

SpectralTDF is a program for computing the transition density function (TDF) of the diffusion approximation of the Wright-Fisher process with general diploid selection and recurrent mutation. The program allows for populations with selection, mutation, and demographic parameters that vary over time in a piecewise constant manner. This document describes basic usage of the program. A detailed description of the method can be found in *M Steinrücken, EM Jewett, and YS Song (2016). SpectralTDF: transition densities of diffusion processes with time-varying selection parameters, mutation rates and effective population sizes. Bioinformatics. **32**:795–797.*

1.1 Overview of the model

We consider a biallelic locus with two alleles, A_0 and A_1, evolving in a population whose size, mutation rates, and selection coefficients are constant within each of K distinct epochs. The kth epoch has effective size N_k (diploid individuals) and duration τ_k generations (for example, see Figure 1).

Within the kth epoch, the probability that a copy of allele A_0 mutates to allele A_1 in a single meiosis is a_k, and the corresponding probability that a copy of allele A_1 mutates to allele A_0 is b_k. In addition, selection acts in such a way that the relative fitness of an individual carrying i copies of allele A_1 is $1 + s_{ki}$ $(i = 1, 2)$.

Denote the frequency of allele A_1 by x. We wish to compute the probability distribution, $p(x;t)$, of x at some time t in generations in the interval $[0, T]$, where $T = \sum_{k=0}^{K} \tau_k$ is the cumulative length of all epochs (Figure 1). The `spectralTDF` program can be used to compute the value of the TDF at an arbitrarily chosen set of time points between the start of the first epoch and the end of the final epoch.

2 Installation

Simply download the file `spectralTDF.jar` from `https://sourceforge.net/projects/spectraltdf/` and place it in a convenient location on your computer. From a terminal window, switch to the directory containing `spectralTDF.jar` (e.g., `$cd <directory name>` on Linux or Mac) and run it using the command described in Sections 4 and 5.

3 Basic usage

The inputs to the program are the population sizes $\vec{N} = (N_1, ..., N_K)$, epoch durations $\vec{\tau} = (\tau_1, ..., \tau_K)$, mutation rates $\vec{a} = (a_1, ..., a_K)$ and $\vec{b} = (b_1, ..., b_K)$, selection coefficients $\vec{s_1} = (s_{1,1}, ..., s_{K,1})$ and $\vec{s_2} = (s_{1,2}, ..., s_{K,2})$, initial condition (initial frequency, mutation-selection balance, or mutation-drift balance), and a set of times $\vec{t} = (t_1, t_2, ...)$ at which the TDF will be evaluated. A full list of options is given below.

4 Options

- `--initialCondition`: Specifies the initial distribution of the frequency of allele A_1 at the beginning of the first epoch. The `initialCondition` option takes one of three integer arguments $(1, 2$, or $3)$ corresponding to the following initial conditions: 1 (initial frequency) if this option is used, the user must also specify the value of the initial frequency using the `--initFrequency` option. 2 (mutation-selection balance), 2 (mutation-drift balance).

- `--initFrequency`: (Optional) Specifies the frequency, x_0, of allele A_1 at the beginning of the first epoch. This option is not required if the `--initialCondition` option is used to specify a different initial distribution (mutation-selection balance or mutation-drift balance).

- `--effPopSizes`: Specifies the effective size of the population (number of diploid individuals) in each epoch. If there is more than one epoch, `effPopSizes` must be a vector of the form `[`N_1`,` N_2`,...,`N_K`]`. If there is a single epoch, `effPopSizes` can be a vector with one element (i.e, `[`N_1`]`) or a single number (i.e., N_1).

- `--epochDurations`: Specifies the duration of each epoch in units of generations. If there is more than one epoch, `epochDurations` must be a vector of the form `[`τ_1`,` τ_2`,...,`τ_K`]`. If there is a single epoch, `epochDurations` can be a vector with one element (i.e, `[`τ_1`]`) or a single number (i.e., τ_1).

- `--mutToA1`: Specifies the per-base, per-meiosis mutation rate to allele A_1 from allele A_0 in each epoch. If there is more than one epoch, `mutToA1` must be a vector of the form `[`a_1`,` a_2`,...,`a_K`]`. If there is a single epoch, `mutToA1` can be a vector with one element (i.e, `[`a_1`]`) or a single number (i.e., a_1).

- `--mutFromA1`: Specifies the per-base, per-meiosis mutation rate from allele A_1 to allele A_0 in each epoch. If there is more than one epoch, `mutFromA1` must be a vector of the form `[`b_1`,` b_2`,...,`b_K`]`. If there is a single epoch, `mutFromA1` can be a vector with one element (i.e, `[`b_1`]`) or a single number (i.e., b_1).

- `--s1`: Specifies the selection coefficient for heterozygotes in each epoch (see the Basic Usage section). If there is more than one epoch, `s1` must be a vector of the form `[`s_{11}`, `s_{21}`,...,`s_{K1}`]`. If there is a single epoch, `s1` can be a vector with one element (i.e, `[`s_{11}`]`) or a single number (i.e., s_{11}).

- `--s2`: Specifies the selection coefficient for homozygotes in each epoch (see the Basic Usage section). If there is more than one epoch, `s2` must be a vector of the form `[`s_{12}`, `s_{22}`,...,`s_{K2}`]`. If there is a single epoch, `s2` can be a vector with one element (i.e, `[`s_{12}`]`) or a single number (i.e., s_{12}).

- `--evaluationTimes`: Specifies the time points at which to evaluate the TDF. All times are given in units of generations and must lie in the interval $[0, T]$ where $T = \sum_{k=0}^{K} \tau_k$. If there is more than one epoch, `evaluationTimes` must be a vector of the form `[`t_1`, `t_2`,...,`t_K`]`. If there is a single epoch, `evaluationTimes` can be a vector with one element (i.e, `[`t_1`]`) or a single number (i.e., t_1).

- `--numGridPts`: Specifies the number, n_g, of points in frequency space at which the TDF will be evaluated. Grid points are evenly spaced within the interval $[0, 1]$: i.e., $\{0, 1/n_g, 2/n_g, ..., (n_g - 1)/n_g, 1\}$.

- `--precision`: (Optional) Specifies the number of digits to use in calculations requiring high precision. The default value is 60. The default value of `precision` should be sufficient when the values of the population-scaled selection coefficients are small to moderate ($\sigma_1, \sigma_2 \leq 10$). However, when one or more of the population-scaled selection coefficients is large ($10 < \sigma_1, \sigma_2$), it may be necessary to increase the value of `precision` when evaluating the TDF at short times ($t \approx 0.01$ coalescent units of N_k generations). In practice, a value of 150 should be sufficient even for strong selection coefficients ($\sigma_1, \sigma_2 \sim 100$). To avoid long running times, try increasing the value of the `matrixCutoff` first.

- `--matrixCutoff`: (Optional) Specifies the dimension at which the square matrix representation of the diffusion operator (Equation 5, supplemental methods) is truncated. The default value is 150. For larger selection coefficients ($10 < \sigma_1, \sigma_2$), it may be necessary to increase the value of `matrixCutoff` when evaluating the TDF at short times ($t \approx 0.01$ coalescent units of N_k generations). In practice, a matrix cutoff of `1,000` should be sufficient even for large selection coefficients ($\sigma_1, \sigma_2 \sim 100$).

- `-Xmx`: (Optional) Sets the maximum heap size for the program. Use this option if the program throws an "out of memory" error. For example, use "`java -Xmx1G -cp spectralTDF.jar ...`" to set the maximum heap size to 1 GB.

- `| grep -v \#`: (Optional: must appear after all other flagged options) Removes lines of the output beginning with the `#` character. These lines contain information about the progress of the program, such as timing and warnings.

5 Example

Consider the population shown in Figure 1, in which the population sizes and epoch durations are $\vec{N} = (1000, 600, 900)$ diploid individuals and $\vec{\tau} = (200, 100, 200)$ generations, respectively. Suppose that the mutation and selection parameters are given by $\vec{a} = (10^{-8}, 10^{-8}, 10^{-8})$, $\vec{b} = (10^{-8}, 10^{-8}, 10^{-8})$, $\vec{s_1} = (0.002, -0.004, 0.001)$, and $\vec{s_2} = (0.003, -0.006, 0.0015)$. In other words, A_1 is an allele with incomplete dominance that is beneficial in the first and third epochs, and deleterious in the second epoch. If the initial frequency is $x_0 = 0.5$ and we wish to evaluate the TDF at the time points $t = 250$ and $t = 500$ generations (i.e., half way through the first epoch and at the end of the final epoch), the input to the JAVA program is

```
java -cp spectralTDF.jar TDF.spectralTDF
--initFrequency 0.5
--initialCondition 1
--effPopSizes [1000,600,900]
--epochDurations [200,100,200]
--mutToA1 [10E-8,10E-8,10E-8]
--mutFromA1 [10E-8,10E-8,10E-8]
--s1 [0.002,-0.004,0.001]
--s2 [0.003,-0.006,0.0015]
--numGridPts 200
--evaluationTimes [250,500]
| grep -v \#
```

In the above code, the flagged option `--numGridPts 200` specifies that the value of the TDF will be computed at the points $x \in \{0, 1/200, 2/200, ..., 199/200, 1\}$. The optional suffix | `grep -v \#` suppresses lines of the output prefixed by the character #, which contain information about the progress of the program.

6 Output

The output of the `spectralTDF` program is a set of lists, or a single list if the TDF is evaluated at a single time point. Unless the | `grep -v \#` option is used, these lists will be interspersed with lines beginning with the hash character (#) that provide additional information about the progress of the script.

The lists appear in the same order as the times specified in the `--evaluationTimes` option. In other words, if the argument of `--evaluationTimes` is $[t_1, t_2, ..., t_K]$, the ith list corresponds to time t_i. The jth element of list i is the value of the TDF, evaluated at time t_i at frequency $x = (j-1)/n_g$, where n_g is the number of grid points specified using the `numGridPts` command. For example, if we set $n_g = 5$ in the example in section 5, then

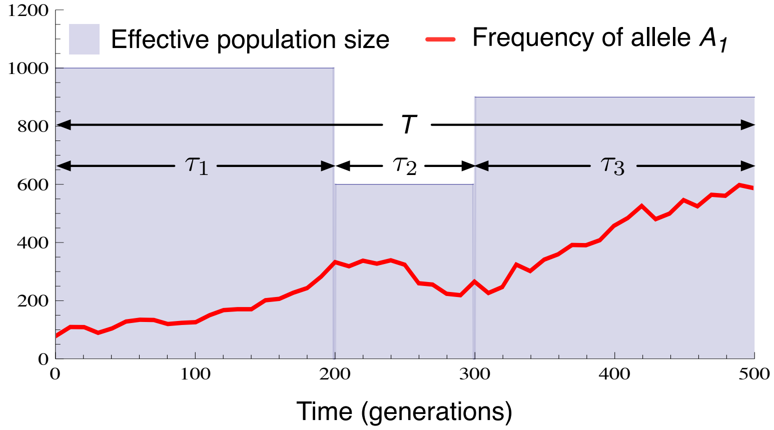

Figure 1: Diagram of the model. A population has constant size in each of K epochs ($N_1 = 1000$, $N_2 = 600$, $N_3 = 900$). An allele, A_1, at a locus of interest evolves over time, subject to pressures of mutation and selection that are constant within each epoch.

the output of the program would be:

```
0E-188
0.459154079063002383484886361705502585498005601267670717998271
1.61743503227723959309472406152287868317065751727293254431199
1.96789885585701800190526097855704384215014440887976765718803
0.904179600804491822080824955924524422256778410101795846508473
0E-187

0E-185
0.768569174306731933893301460171053213004135355961825177523729
1.22592350162545137327861312644970049234266133674300067924303
1.36561785869401486236970644827606827440667625100727778433317
1.06451917884729924522973590938117124737345410341294165838489
0E-185
```