Simulation and inference on exact solutions of coalescent distributions under diverse demographies.

Janek Sendrowski*, Asger Hobolth†
Contact: sendrowski.janek@birc.au.dk, asger@math.au.dk

March 6, 2024

Abstract

Coalescent theory has proven to be a powerful framework, but exact solutions have so far only been obtainable under relatively simple scenarios, and current simulators such as msprime only provide stochastic solutions, necessitating the use of approximate Bayesian computation (ABC) for parameter inference. Recent advancements in phase-type theory now permit exact solutions for all (joint-)moments of coalescent distributions under complex demographic scenarios. These include moments of tree height, total branch length and the site-frequency spectrum (SFS) under demographies with multiple demes, varying population sizes and migration rates, multiple merger coalescents, and, to some extent, multiple loci. Here we present the theory underlying these developments and propose PhaseGen – a Python-based software package simulating these moments, and incorporating a maximum likelihood estimation (MLE) framework for parameter inference. The package is comprehensively tested, and documented at phasegen.readthedocs.io.

1 Introduction

Population genetic simulators are essential tools in evolutionary biology, allowing researchers to explore complex demographic scenarios, develop hypotheses regarding evolutionary processes, and infer parameters under a given demographic model. There exists a variety of tools either employing

*Bioinformatics Research Center, Aarhus University, Denmark.
†Department of Mathematics, Aarhus University, Denmark.

backward or forward simulations, and designed with or without direct parameter inference capabilities. Current coalescent-based simulators such as msprime provide stochastic solutions, which gives them great flexibility while mirroring the data generation process (Baumdicker et al., 2021). However, the stochasticity of the results, necessitates the simulation of many replicates for obtaining good estimates of summary statistics, especially when the underlying demographic model is complex and the statistic is based on higher moments. Furthermore, parameter inference necessitates the use of approximate Bayesian computation (ABC).

Another set of tools are forward simulators where the allele frequency trajectories forward in time are described by (often diffusion-based) differential equations (Gutenkunst et al., 2009; Jouganous et al., 2017). The great advantage of forward simulations is that they readily allow for the integration of selection. They have, however, different caveats such as model initialization, choice of run times, and they tend to be overall much less efficient that backward simulations. Furthermore, analytical solutions to the underlying differential equations are rarely available necessitating the use numerical approximations which are not always numerically stable, and higher-order moments are not readily obtainable (Jouganous et al., 2017).

Phase-type theory constitutes a unified mathematical framework that offers solutions for mixtures and convolutions of exponential distributions (Bladt and Nielsen, 2017). It is applicable to coalescent theory as the coalescent is phase-type distributed (Hobolth et al., 2019). One previous limitation of the application of phase-type theory was the restriction to time-homogeneous coalescent processes. We describe here how this limit is overcome. PhaseGen is equipped with a demography interface akin to Msprime, designed to handle temporal changes. This is achieved by discretizing the demography into piece-wise constant epochs, enabling the simulation of more intricate demographic scenarios, such as population splits, mergers, and bottlenecks.

2 Phase-type distribution

2.1 Introduction

Consider a Markov jump process $\{X_t\}_{t\geq 0}$ with finite state-space $E = \{1, 2, \ldots, p\}$ and let $S(t) = \{s(t)_{ij}\}_{i,j=1,\ldots,k}$ be the intensity matrix holding the exponential rates of jumping from state i to j at time t. Define the time until absorption as

$$\tau = \inf\{t > 0 : X_t \in B\},$$

where B is the set of absorbing states. τ is said to be (time-inhomogeneous) phase-type distributed with state space E, initial state vector $\alpha = (\alpha_i)_i$ and intensity matrix $S(t)$, and we write

$$\tau \sim IPH(\alpha, S).$$

2.2 The coalescent

In the following, we construct a state space for describing the tree height for the standard coalescent. For this we require a state space $E = \{1, \ldots, n\}$ with n states, where n denotes the number of initial lineages. The only absorbing state is $B = \{1\}$ and our transition rates are given by

$$s_{ij} = \begin{cases} \binom{i}{2}/N_e & \text{if } i = j - 1, \\ 0 & \text{otherwise,} \end{cases}$$

where states are indexed with respect to the ordering $\phi(i) = i, i \in E$, and N_e is the effective population size. Henceforth, we refer to the state space keeping track of the number of lineages at a time as the default state space.

In order to obtain more complex summary statistics such as the site-frequency spectrum (SFS), we need to augment the state space to keep track of the number of lineages that subtend i lineages at a time. Let each state be an n-tuple $e = (a_1, a_2, \ldots, a_n)$ with e_i denoting the number of lineages that subtend i lineages. Our state space is given by $E = \{e : \sum_i i a_i = n\}$, and the only absorbing state is $B = \{(0, 0, ..., 1)\}$. The transition rates are given by (Hobolth et al., 2019)

$$s_{\phi(e_1),\phi(e_2)} = \begin{cases} \binom{a_i}{2}/N_e & \text{if } e_1 = (a_1, \ldots, a_n), \\ & \quad e_2 = (a_1, \ldots, a_i - 2, \ldots, a_{2i} + 1, \ldots, a_n), \\ & \quad a_i \geq 2, \\ a_i a_j / N_e & \text{if } e_1 = 2(a_1, \ldots, a_n), \\ & \quad e_2 = (a_1, \ldots, a_i - 1, \ldots, a_j - 1, \ldots, a_{i+j} + 1, \ldots, a_n), \\ & \quad a_i a_j \geq 1, \\ 0 & \text{otherwise,} \end{cases}$$

where ϕ is an arbitrary ordering. We refer to this state space as the block-counting state space, and to a_i as lineage block. Details on how to support multiple demes and two loci are shown in the Appendix (A).

2.3 Moments

The first moment of τ can be written as

$$\mathbb{E}(\tau) = \int_0^\infty \mathbb{1}_{E\setminus B}(X_t)dt,$$

where $\mathbb{1}_{E\setminus B}$ is the indicator function outputting 1 as long as X_t is in a non-absorbing state and 0 otherwise.

In order to compute more complex summary statistics such as the total branch length and SFS, we want to assign different weights to each state. We do this by defining rewards. Let $r : E \to \mathbb{R}_{\geq 0}$ map each state to its chosen reward.

Then

$$\int_s^t r(X_t)dt,$$

is the reward accumulated over time $[s, t]$ with respect to r.

In order to compute this quantity, we need to evaluate

$$\int_s^t P(s,u)R(u)P(s,t)du, \tag{1}$$

where $P(s,t)$ is a transition matrix describing the probability of transitioning from state i at time s to state j at time t, and $R = \{r_{ii} = r(i)\}$ is a (diagonal) reward matrix.

In practice, determining $P(s,t)$ for time-inhomogeneous processes requires evaluating a product integral, i.e.,

$$P(s,t) = \prod_s^t (I + S(u)du), \tag{2}$$

which is analytically intractable (I is the identity matrix). However, rather than evaluating (2) numerically and accept numerically errors, we propose a

scheme in which the demography is discretized into n epochs during which S is constant. We can then obtain exact solutions under this discretized demography.

We can use Van Loan's method to evaluate (1) and higher-order moments, including cross moments (Hobolth and Jensen, 2011; Van Loan, 1978).

Assume for now that S is constant over time. Let the Van Loan matrix of order k be the $(k+1) \times (k+1)$ block matrix

$$V_k(S, R_1, \ldots, R_k) = \begin{bmatrix} S & R_1 & 0 & \cdots & 0 & 0 \\ 0 & S & R_2 & \cdots & 0 & 0 \\ 0 & 0 & S & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & S & R_k \\ 0 & 0 & 0 & \cdots & 0 & S \end{bmatrix},$$

where R_i are reward matrices. Note that we can choose the R_i to be different if we are interested in cross moments. Also note that, in practice, we always apply zero rewards to absorbing states.

Exponentiating V_k, we obtain the accumulated reward per state over time $\delta = t - s$:

$$M_k(S, R_1, \ldots, R_k, \delta) = \exp(V_k \cdot \delta).$$

The reward can be obtained by extracting the upper right block of M_k, which we denote M_k^{sub}, whence the final moment can be computed as

$$m_k = k! \cdot \alpha \cdot M_k^{sub} \cdot \mathbf{e},$$

where α is the initial state vector and $\mathbf{e}$ a vector of ones.

Now, to obtain the moments for our discretized demography, let δ_i be the time spent and S_i the intensity matrix in epoch i, respectively. We can calculate $M_k^i = M_k(S_i, R_1, \ldots, R_k, \delta_i)$ for all i. The matrix containing the total accumulated reward over all epochs is then obtained by

$$\hat{M}_k = \prod_i M_k^i. \tag{3}$$

In practice, sufficiently many epochs are generated so as to ensure we have reached almost sure absorption. Note that (3) constitutes a discretization of the product integral (2).

The accumulated reward in each state can then be obtained by extracting the upper right block of $\hat{M}_k$, which we denote by $\hat{M}_k^{sub}$. The final moment can then be computed as

$$m_k = k! \cdot \alpha \cdot \hat{M}_k^{sub} \cdot e,$$

where α is the initial state vector and e a vector of ones.

To determine the probability of almost sure absorption, we discretize the transition matrix $P_i = \exp(S_i)$, using the same discretization scheme, i.e.,

$$t_{abs} = \min_i \{t : \alpha(\prod_i P_i)e \approx 0\},$$

where α and e are the same as above.

2.4 2-epoch example

To illustrate the method, assume we have $n = 3$ lineages, a single deme, and a 2-epoch demography with $N_e = 1, t \in [0, 2)$ and $N_e = 0.5, t \in [2, \infty)$. Assume furthermore we are interested in the variance of the total branch length so that it will suffice to work with the default state space. let $(N_1, N_2) = (1, 0.5)$ be the effective population sizes in epochs 1 and 2. Also assume we start with 3 lineages in epoch 1 with a probability of 1, so that $\alpha = (1, 0, 0)$.

We have

$$S_1 = \{s_{ij}\}/N_1 = \begin{bmatrix} -3 & 3 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad S_2 = \{s_{ij}\}/N_2 = \begin{bmatrix} -6 & 6 & 0 \\ 0 & -2 & 2 \\ 0 & 0 & 0 \end{bmatrix},$$

using an ordering of $\phi(i) = 4 - i$ where i is the number of lineages, i.e., state 1 is the state with 3 lineages, etc. The reward matrix for the total branch length is given by

$$R = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

i.e., we reward each non-absorbing state with its number of lineages. The resulting Van Loan matrices are

$$V_1 = \begin{bmatrix} S_1 & R & 0 \\ 0 & S_1 & R \\ 0 & 0 & S_1 \end{bmatrix} = \left[\begin{array}{ccc|ccc|ccc} -3 & 3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & -3 & 3 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

$$V_2 = \begin{bmatrix} S_2 & R & 0 \\ 0 & S_2 & R \\ 0 & 0 & S_2 \end{bmatrix} = \left[\begin{array}{ccc|ccc|ccc} -6 & 6 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 2 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & -6 & 6 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & -6 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

We then compute the accumulated reward per epoch as

$$M_1 = \exp(V_1 \cdot \delta_1) \approx \left[\begin{array}{ccc|ccc|ccc} 0.002 & 0.199 & 0.798 & 0.015 & 0.889 & 1.69 & 0.045 & 2.002 & 2.316 \\ 0 & 0.135 & 0.865 & 0 & 0.541 & 1.188 & 0 & 1.083 & 1.293 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0.002 & 0.199 & 0.798 & 0.015 & 0.889 & 1.69 \\ 0 & 0 & 0 & 0 & 0.135 & 0.865 & 0 & 0.541 & 1.188 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0.002 & 0.199 & 0.798 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.135 & 0.865 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right],$$

$$M_2 = \exp(V_2 \cdot \delta_2) \approx \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 1 & 0 & 0 & 1.5 & 0 & 0 & 1.75 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1.5 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

where $\delta_1 = 2$ and $\delta_2 = t_{abs} \coloneqq 31$, the time until almost sure absorption.

To obtain the total accumulated reward over all epochs, we compute

$$\hat{M} = M_2 \cdot M_1 \approx \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 1 & 0 & 0 & 2.797 & 0 & 0 & 5.477 \\ 0 & 0 & 1 & 0 & 0 & 1.865 & 0 & 0 & 3.053 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2.797 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1.865 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right], \tag{4}$$

from which we can extract the total accumulated reward in the upper right block. We have

$$\hat{M}^{sub} \approx \begin{bmatrix} 0 & 0 & 5.477 \\ 0 & 0 & 3.053 \\ 0 & 0 & 0 \end{bmatrix}.$$

We can then compute the (uncentered) second moment by

$$m_2 \approx 2! \cdot \alpha \cdot \hat{M}^{sub} \cdot e = 2 \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 5.477 \\ 0 & 0 & 3.053 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \approx 10.955.$$

After obtain the mean m_1, using the same method, we can obtain the variance by

$$\sigma^2 = m_2 - m_1^2 \approx 10.955 - 2.797^2 \approx 3.132.$$

3 Runtime & precision

This framework has the advantages of being able to handle zero rewards out-of-the-box and using matrix exponentiation instead of inversion which can avoid precision problems. Is has the disadvantage of requiring to work with matrices of size $n * (k + 1)$, where n is the total number of states and k the moment's order. But there are efficient implementations for matrix exponentiation, also in Python, where SciPy's `expm()` uses a Padé approximation.

4 Code example

In the code snippet below, we define a coalescent distribution with a 2-deme, 4-epoch demography and a total number of lineages of `n=8`. As our ancestry model we choose the beta coalescent with `alpha=1.7`. Upon construction the different distributions are available.

```
import phasegen as pg

coal = pg.Coalescent(
    n=pg.LineageConfig({'pop_0': 3, 'pop_1': 5}),
    model=pg.BetaCoalescent(alpha=1.7),
    demography=pg.Demography(
        pop_sizes={
            'pop_1': {0: 1.2, 5: 0.1, 5.5: 0.8},
            'pop_0': {0: 1.0}
        },
        migration_rates={
            ('pop_0', 'pop_1'): {0: 0.2, 3: 0.3},
            ('pop_1', 'pop_0'): {0: 0.5}
        }
    )
)
```

We now proceed to plotting some quantities, namely the demography, tree height distribution, expected SFS and correlation matrix of different SFS bins. Note that we adjust the maximum time shown to be the 99th percentile of the tree height distribution. The output of the snippet below is shown in Figure 1

```
import numpy as np
import matplotlib.pyplot as plt

_, axs = plt.subplots(2, 2, figsize=(9, 7))
t = np.linspace(0, coal.tree_height.quantile(0.99), 100)

coal.demography.plot(ax=axs[0, 0], show=False, t=t)
coal.tree_height.plot_pdf(ax=axs[0, 1], show=False)
coal.sfs.mean.plot(ax=axs[1, 0], show=False, title='SFS')
coal.sfs.corr.plot(ax=axs[1, 1], show=False, title='2-SFS')

plt.tight_layout()
plt.show()
```

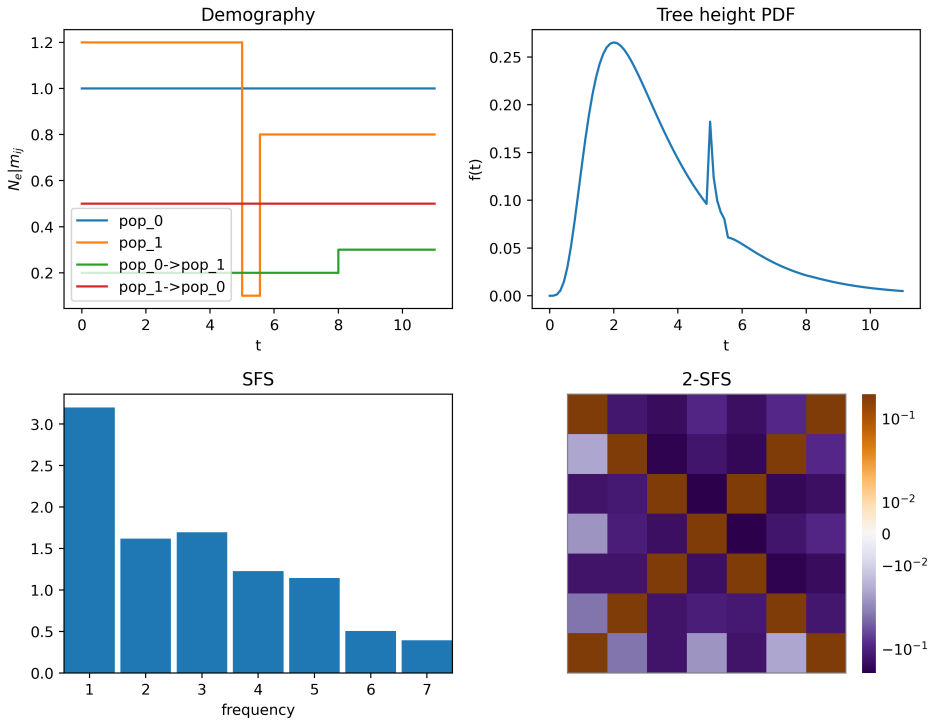

Figure 1: Output from code snippet in Section 4. **Top left**: population sizes and migration rates over time. **Top right**: tree height distribution. **Bottom left**: expected SFS. **Bottom right**: 2-SFS correlation matrix.

4.1 Other quantities

5 Discussion

Acknowledgements

References

Franz Baumdicker, Gertjan Bisschop, Daniel Goldstein, Graham Gower, Aaron P Ragsdale, Georgia Tsambos, Sha Zhu, Bjarki Eldon, E Castedo Ellerman, Jared G Galloway, Ariella L Gladstein, Gregor Gorjanc, Bing Guo, Ben Jeffery, Warren W Kretzschumar, Konrad Lohse, Michael Matschiner, Dominic Nelson, Nathaniel S Pope, Consuelo D Quinto-Cortés, Murillo F Rodrigues, Kumar Saunack, Thibaut Sellinger, Kevin Thornton, Hugo van Kemenade, Anthony W Wohns, Yan Wong, Simon Gravel, Andrew D Kern, Jere Koskela, Peter L Ralph, and Jerome Kelleher. Efficient ancestry and mutation simulation with msprime 1.0. *Genetics*, 220(3):iyab229, 12 2021. ISSN 1943-2631. doi: 10.1093/genetics/iyab229. URL `https://doi.org/10.1093/genetics/iyab229`.

Mogens Bladt and Bo Friis Nielsen. *Matrix-Exponential Distributions in Applied Probability*. Probability Theory and Stochastic Modelling. Springer New York, NY, 2017. doi: 10.1007/978-1-4939-7049-0.

Ryan Gutenkunst, Ryan Hernandez, Scott Williamson, and Carlos Bustamante. Inferring the joint demographic history of multiple populations from multidimensional snp frequency data. *PLoS genetics*, 5:e1000695, 10 2009. doi: 10.1371/journal.pgen.1000695.

Asger Hobolth and Jens Ledet Jensen. Summary statistics for endpoint-conditioned continuous-time markov chains. *Journal of Applied Probability*, 48(4):911–924, 2011. ISSN 0021-9002. doi: 10.1239/jap/1324046009.

Asger Hobolth, Arno Siri-Jégousse, and Mogens Bladt. Phase-type distributions in population genetics. *Theoretical Population Biology*, 127:16–32, 2019. ISSN 0040-5809. doi: https://doi.org/10.1016/j.tpb.2019.02.001. URL `https://www.sciencedirect.com/science/article/pii/S0040580919300140`.

Julien Jouganous, Will Long, Aaron P Ragsdale, and Simon Gravel. Inferring the Joint Demographic History of Multiple Populations: Beyond the Diffusion Approximation. *Genetics*, 206(3):1549–1567, 07 2017. ISSN 1943-2631. doi: 10.1534/genetics.117.200493. URL `https://doi.org/10.1534/genetics.117.200493`.

C. F. Van Loan. Computing integrals involving the matrix exponential. *IEEE Trans. Automatic Control*, 23(3):395–404, 1978.

A Transition Rates