`demes`: a pain-free specification for complex demography

Graham Gower[*], Aaron P. Ragsdale[*], ...Others here... , Kevin Thornton[**], and Jerome Kelleher[**]

[*,**]Denote equal contribution

April 30, 2021

Abstract

Introduction

The ever-increasing amount of genetic sequencing data from genetically and geographically diverse species and populations has allowed us to infer complex demography and study life history at fine scales. An integral component to such population genetics studies is simulation. Software to either simulate whole genome sequences (Thornton, 2014, Kelleher et al., 2016, Haller and Messer, 2019, Adrion et al., 2020) or informative summary statistics of diversity (Gutenkunst et al., 2009, Kamm et al., 2017, Jouganous et al., 2017) have enabled the increasing complexity of genomic studies, with many software packages able to handle large sample sizes, many interacting populations, and deviations from random-mating models of panmixia.

The ability to simulate complex demographic scenarios, however, does not come without its own set of obstacles and frustrations. First, specifying models with many populations and parameters can be cumbersome and error-prone (e.g., (Ragsdale et al., 2020)). It is time-consuming and tedious to implement and then verify the correctness of such demographic models. Making matters worse, each software package typically has its own application programming interface (API), which makes it difficult to translate models between simulation software.

Second, many software API are not particularly easy to read, especially when there are a large number of demographic events and parameters. This not only makes debugging difficult, but it can be difficult to even notice if a mistake has been made. A common interface for unambiguously specifying demographic models would reduce implementation errors and promote ease-of-use for many simulation software packages.

Finally, simulated or inferred demographic models are regularly shared in the literature, and for someone else to be able to reproduce or reimplement some demographic scenario, it needs to be described fully and without ambiguity. However, it is common for reported demographic models to lack clear or even complete descriptions, hindering reproducibility. A human-readable description for such models would facilitate their sharing and reimplementation.

Here, we present `demes`, a specification of complex demographic models that is both human readable and can be directly passed to a growing number of simulation software. `Demes` supports model specification in a high-level `YAML` format (Ben-Kiki et al., 2009) that is designed to be as easy as possible to implement, read, and share. Models are then represented in an unambiguous low-level format within Python that checks model validity. The initial Python release of `demes` (version 0.1, described here) supports *static* demographic descriptions, that is, models with fixed parameters. Future releases will support [[drawing parameters from distributions, for example

for ABC-based inference, or describing "dynamic" models where parameters may be fit to data, e.g. `moments` or `∂a∂i`]].

The `demes` specification

- Demographic models are comprised of one or more interacting populations, along with description, doi, time units and generation times.
- Populations (or demes) are treated as discrete collection of exchangeable individuals following a well-defined set of rules
- These rules include population sizes, times of existence, and other information about the mating system (e.g. selfing or cloning rates)
- Relationships between demes are defined by ancestors/descendents and possible continuous or instantaneous migration between coexisting demes

High-level model specification in `YAML`

- YAML allows us to write this information in a human-readable and structured manner (lists, keys/items, etc)
- Implicit values, so that models can be written compactly
- Example of an IM Model 1

```yaml
description: A two-population isolation-with-migration model. Demes 1 and 2
  split with size reductions followed by exponential growth.
time_units: generations
demes:
- name: ancestral
  description: The ancestral deme that splits into two child demes.
  epochs:
  - start_size: 5000
    end_time: 2000
- name: deme1
  description: The first descendent deme.
  ancestors:
  - ancestral
  epochs:
  - start_size: 1000
    end_size: 10000
- name: deme2
  description: The second descendent deme.
  ancestors:
  - ancestral
  epochs:
  - start_size: 2000
    end_size: 8000
migrations:
- demes:
  - deme1
  - deme2
  rate: 1e-4
```

Demographic Model 1: An IM model.

Unambiguous low-level representation

- The Python API, which can read in a YAML file or be built using the Builder
- Demographic model validitation
- Unambiguous, redundant information
- Designed to be programatically read by simulation software backends

Extensions and future directions

Software support for `demes`

APR: at time of publication. TBD...

- `msprime`: coalescent simulation (Kelleher et al., 2016)
- `stdpopsim`: a collection of maintained species and demographic models, which includes a large number of models specified using `demes`(Adrion et al., 2020)
- `fwdpy11`: forward-time Wright-Fisher simulation (Thornton, 2014)
- `moments`: compute expected diversity statistics (SFS, LD) and perform demographic inference using summary statistics (Jouganous et al., 2017, Ragsdale and Gravel, 2019)
- ∂`a`∂`i`: compute the expected SFS and perform demographic inference using summary statistics (Gutenkunst et al., 2009)

Other stuff...

- `demesdraw` (`https://github.com/grahamgower/demesdraw.git`)
- General framework for inference (have been working on that over in `moments`, and perhaps it could be generalized to work with any simulation engine
- Defining distributions of parameter values, perhaps?

References

Jeffrey R Adrion, Christopher B Cole, Noah Dukler, Jared G Galloway, Ariella L Gladstein, Graham Gower, Christopher C Kyriazis, Aaron P Ragsdale, Georgia Tsambos, Franz Baumdicker, et al. A community-maintained standard library of population genetic models. *eLife*, 9:e54967, 2020.

Oren Ben-Kiki, Clark Evans, and Brian Ingerson. Yaml ain't markup language (yaml™) version 1.1. *Working Draft 2008-05*, 11, 2009.

Ryan N Gutenkunst, Ryan D Hernandez, Scott H Williamson, and Carlos D Bustamante. Inferring the joint demographic history of multiple populations from multidimensional snp frequency data. *PLoS genetics*, 5(10):e1000695, 2009.

Benjamin C Haller and Philipp W Messer. Slim 3: Forward genetic simulations beyond the wright–fisher model. *Molecular biology and evolution*, 36(3):632–637, 2019.

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, 2017.

John A Kamm, Jonathan Terhorst, and Yun S Song. Efficient computation of the joint sample frequency spectra for multiple populations. *Journal of Computational and Graphical Statistics*, 26(1):182–194, 2017.

Jerome Kelleher, Alison M Etheridge, and Gilean McVean. Efficient coalescent simulation and genealogical analysis for large sample sizes. *PLoS computational biology*, 12(5):e1004842, 2016.

Aaron P Ragsdale and Simon Gravel. Models of archaic admixture and recent history from two-locus statistics. *PLoS genetics*, 15(6):e1008204, 2019.

Aaron P Ragsdale, Dominic Nelson, Simon Gravel, and Jerome Kelleher. Lessons learned from bugs in models of human history. *American Journal of Human Genetics*, 107(4):583–588, 2020.

Kevin R Thornton. A c++ template library for efficient forward-time population genetic simulation of large populations. *Genetics*, 198(1):157–166, 2014.

Appendix

APR: Perhaps and example of a many-population model, and API to simulate using `msprime`, `fwdpy11`, and `moments`?