`Moments LD` user manual

Corresponding to version 0.0.1

Aaron Ragsdale
Contact: aaron.ragsdale@mail.mcgill.ca

April 4, 2019

Contents

Example code

1 Introduction to `mold`

Welcome to `moments.LD`, a program for simulating linkage disequilibrium statistics. `moments.LD`, or `mold`, can compute a large set of informative LD statistics for many populations, and performs likelihood-based demographic inference using those statistics.

There are three primary features of `mold` to enable LD-based demographic inference: reading and parsing data, building demographic models, and inferring the parameters of those models by comparing model predictions to data. Typically, we use biallelic SNP data, along with a recombination map, to compute two-locus statistics over a range of genetic distances. We then use `mold` to compute expectations for those statistics under the demographic models we want to test, which can include multiple populations with variable migration, splits and mergers, and population size changes. Using a likelihood-based inference approach, we optimize those models to find the set of parameters that best fit the data.

I've tried to make parsing data and defining demographic models as painless as possible, though the complexity of the program does require some amount of script-writing and interaction. Luckily, `mold` is written in Python, a friendly and powerful programming language. If you are already familiar with ∂a∂i or *moments*, or Python in general, you are in a good position to dive right in to `mold`. If you have limited Python experience, this manual should provide the background and examples to get you up to speed and productive with `mold`.

1.1 Getting help and helping us

Undoubtedly, there will be bugs. If you find a bug in `mold`, or more generally if you find certain aspects of the program to be unintuitive or difficult to use, I would appreciate the feedback. Please submit a bug report at `https://bitbucket.org/simongravel/moments/issues`, and I will try to address the issue in a timely manner. Similarly, if you have suggestions for improved functionality or feature requests, those can be submitted in the issues as well or you can contact me directly.

As we do our own research, *moments* and `mold` are constantly improving. Our philosophy is to include any code we develop for our own projects that may useful to others. If you develop *Moments*-related code that you think might be useful to others, please let me know so I can include it with the main distribution.

2 LD statistics

Patterns of linkage disequilibrium (LD) are informative about evolutionary history, for example for inferring recent admixture events and population size changes or localizing regions of the genome that have experienced recent selective events. LD is commonly measured as the covariance (or correlation) of alleles co-occurring on a haplotype. The covariance (D) is

$$\begin{aligned} D = \mathrm{Cov}(A, B) &= f_{AB} - pq \\ &= f_{AB} f_{ab} - f_{Ab} f_{aB}, \end{aligned}$$

and the correlation (r) is

$$r = \frac{D}{\sqrt{p(1-p)q(1-q)}}.$$

We think of expectations of these quantities as though we average over many realizations of the same evolutionary process, but in reality we have only a single observation for any given pair of SNPs. Therefore in practice we take the averages of LD statistics over many independent pairs of SNPs.

$\mathbb{E}[D]$ is zero genome wide, so LD is often measured by the variance of D ($\mathbb{E}[D^2]$) or the square correlation (r^2), where

$$r^2 = \frac{D^2}{p(1-p)q(1-q)}.$$

Because it is difficult to compute expectations for $\mathbb{E}[r^2]$ under even simple evolutionary scenarios, and because it is difficult to accurately estimate $\widehat{r^2}$ from data, we use $\mathbb{E}[D^2$ and related statistics to compare model predictions for LD to data.

2.1 Hill-Robertson statistics

Hill and Robertson (1968) introduced a recursion for $\mathbb{E}[D^2]$ that allows for variable recombination rate between loci and population size changes over time. To solve for $\mathbb{E}[D^2]$, this system requires additional LD statistics, which we call $Dz = D(1-2p)(1-2q)$ and $\pi_2 = p(1-p)q(1-q)$, where p and q are the allele frequencies at the left and right loci, respectively. This system also relies on heterozygosity (H), so from this system we can compute the vector of statistics

$$y = \begin{pmatrix} \mathbb{E}[D^2] \\ \mathbb{E}[Dz] \\ \mathbb{E}[\pi_2] \\ \mathbb{E}[H] \end{pmatrix}.$$

Instead of computing $\mathbb{E}[r^2]$, which is an expectation of ratios, Hill and Robertson (1968) and Ohta and Kimura (1971) studied the related statistic $\sigma_D^2 = \frac{\mathbb{E}[D^2]}{\mathbb{E}[\pi_2]}$. This statistic has the advantage that its expectation can be computed from the Hill-Robertson recursion, and we can accurately compute it from either phased or unphased data.

2.2 Multi-population LD statistics

In Ragsdale and Gravel (2018), we extended the Hill-Robertson system to consider LD statistics for multiple populations. Here, we can model population size changes, splits, mergers, and migration events (pulse or continuous). The statistics we consider take the form

$$\mathbf{z} = \begin{pmatrix} \mathbb{E}[D_i D_j] \\ \mathbb{E}[D_i z_{j,k}] \\ \mathbb{E}[\pi_2(i,j;k,l)] \\ \mathbb{E}[H_{i,j}] \end{pmatrix},$$

where i, j, k, l index populations, and

$$D_i z_{j,k} = D_i(1-2p_j)(1-2q_k),$$

$$\pi_2(i,j;k,l) = \begin{cases} p_i(1-p_i)q_k(1-q_k), & i=j, k=l \\ \frac{1}{2}p_i(1-p_j)q_k(1-q_k) + \frac{1}{2}p_j(1-p_i)q_k(1-q_k), & i\neq j, k=l \\ \frac{1}{2}p_i(1-p_i)q_k(1-q_l) + \frac{1}{2}p_i(1-p_i)q_l(1-q_k), & i=j, k\neq l \\ \frac{1}{4}p_i(1-p_j)q_k(1-q_l) + \frac{1}{4}p_i(1-p_j)q_l(1-q_k) & \\ \quad +\frac{1}{4}p_j(1-p_i)q_k(1-q_l) + \frac{1}{4}p_j(1-p_i)q_l(1-q_k), & i\neq j, k\neq l \end{cases},$$

$$H_{i,j} = \begin{cases} p_i(1-p_i), & i=j \\ \frac{1}{2}p_i(1-p_j) + \frac{1}{2}p_j(1-p_i), & i\neq j \end{cases}.$$

From these, we can compare a large number of two-locus statistics. In practice, we work with σ_D^2-like statistics, which are normalized by the π_2 statistic from one of the populations.

3 Getting started

3.1 Getting and installing `mold`

`mold` is packaged and released with `moments`, a python program for running analyses based on the allele frequency spectrum. `moments` is available at `https://bitbucket.org/simongravel/moments`. Because `mold` continues to be developed and improved, it currently lives on the LD branch of the `moments` repository. For this reason, I recommend cloning the `moments` repository and switching to the LD branch. To do this, in the Terminal navigate to the parent directory where you want to copy moments and run `git clone https://bitbucket.org/simongravel/moments.git`. To switch to the LD branch, run `git checkout LD`. And then finally, to install run `sudo python setup.py install`.

3.1.1 Dependencies

`moments` and `mold` depend on a number of Python libraries (We recommend using Python 3).

1. Absolute dependencies: `Python` (version 2.7 or $\geq$ 3.5), `numpy` (version $\geq$ 1.2.0), `scipy` (version $\geq$ 0.6.0), `cython`, `pickle`
2. For Plotting, we use `matplotlib` (version $\geq$ 0.98.1)
3. For Parsing, we take advantage of `hdf5` and `scikit-allel` (version $\geq$ 1.2.0) ()
4. For Demography building, we use `networkx`

I recommend that you install IPython as well. The easiest way to obtain all these dependencies is to use a package management system, such as Conda (`https://conda.io/en/latest/`) or Enthought (`https://www.enthought.com/product/enthought-python-distribution/`).

3.2 Suggested workflow

One of Python's strengths is its interactive nature. When I am first exploring data or writing scripts to build and test models, I often have two windows open: one editing a python script (`script.py`) and the other running an IPython session. That way, I can record my work in the python script and test it as I go. Using IPython, I can call the magic command `%run script.py`, which applies changes I've made in my python script to the IPython session. Note that if I've also changed modules that I've loaded, I'll need to reload those as well. Once I'm happy that I have a usable script, I can call it from the terminal for longer runs of optimization or parsing, using `python script.py`.

Note that we will need to import `moments.LD` to be able to use it: `import moments.LD as mold`.

4 LDstats objects

`mold` represents two-locus statistics using `mold.LDstats` objects, which stores the Hill-Robertson statistics and heterozygosity for one or more populations and for any set of recombination rates. The simplest way to create an `LDstats` object is by defining a set of statistics by hand. For a single population, the order of the LD statistics is $[\mathbb{E}[D^2], \mathbb{E}[Dz], \mathbb{E}[\pi_2]]$ along with heterozygosity $\mathbb{E}[H]$. If the statistics are defined using variables `D2`, `Dz`, `pi2`, and `H`, we would simply call `y = mold.LDstats([[D2, Dz, pi2], [H]], num_pops = 1)`. For example, if we set `D2, Dz, pi2, H = 1e-7, 1e-8, 2e-7, 1e-3` in this example, and then if we print `y`, we would see as the output:

```
Out[1]: LDstats([[1.e-07 1.e-08 2.e-07]], [0.001], num_pops=1, pop_ids=None)
```

To see which statistics each value corresponds to, we can call `y.names()`, which would output:

```
Out[2]: (['DD_1_1', 'Dz_1_1_1', 'pi2_1_1_1_1'], ['H_1_1'])
```

Typically, we either compute the `LDstats` from a demographic model, or we build the object from data. We'll walk through both in later sections. First, we'll use build in model functions to explore what information is stored in `LDstats` objects, and how to manipulate the objects.

To obtain the expected statistics at equilibrium (steady-state demography), we can call `mold.Demographics1D.snm` (snm stands for standard neutral model). We can specify the per-base population size-scaled mutation rate $\theta = 4N_e\mu$ (default set to $\theta = 0.001$) and population size-scaled recombination rates separating loci $\rho = 4N_e r$ (default set to `None`). `mold` can handle any number of recombination rates (zero, one, or multiple), and the returned `LDstats` object will contain LD statistics for as many recombination rates as you gave it.

To give some examples,

```
In [3]: mold.Demographics1D.snm()
Out[3]: LDstats([], [0.001], num_pops=1, pop_ids=None)

In [4]: mold.Demographics1D.snm(rho=0)
Out[4]: LDstats([[1.38888889e-07 1.11111111e-07 3.05555556e-07]], [0.001], num_pops
=1, pop_ids=None)

In [5]: mold.Demographics1D.snm(rho=[0,1,2,10], theta=0.01)
```

```
Out[5]:
LDstats([[1.38888889e-05 1.11111111e-05 3.05555556e-05]
[8.59375000e-06 6.25000000e-06 2.81250000e-05]
[6.25000000e-06 4.16666667e-06 2.70833333e-05]
[2.01612903e-06 8.06451613e-07 2.54032258e-05]], [0.01], num_pops=1, pop_ids=None)
```

In this last example, the four sets of LD statistics correspond to $\rho = 0, 1, 2$, and 10, respectively, while expected heterozygosity is only shown a single time ($\mathbb{E}[H] = 0.01$). In each case, `num_pops` is automatically set to `1`, and because we didn't specify population IDs, `pop_ids = None`. `y.LD()` will return just the LD statistics, while `y.H()` returns just the heterozygosity statistics.

5 Parsing and importing data

1. `mold` can create an LDstats object given a vcf file (and optionally a recombination map, mask files, and population files), for either phased or unphased data
2. Or given a genotype array

5.1 Estimating two-locus statistics from data

(Ragsdale and Gravel, 2019)

5.2 Parsing data from a genotype matrix

1. All loci, not caring about recombination distance (say, all unlinked loci)
2. with the recombination distance between snps known

5.3 Parsing a VCF file

5.3.1 Using a recombination map

5.3.2 Restricting based on features

5.4 Creating bootstrap datasets

```
this is a script dot py
```

Listing 1: **Parsing:** Example of parsing data generated by msprime

6 Specifying a model

General overview of demographic models

1. Attributes of LDstats objects used for building models
2. Splitting, marginalization, swapping
3. Admixture, merging

6.1 Implementation

Manual input of demographic model (gets difficult with more than two populations)

6.2 The `Demography` builder

6.3 Units

1. mutation rate
2. recombination rate
3. unit of time
4. N_e and N_{ref}

```
def bottleneck(params, ns):
    nuB, nuF, T = params
    nu_func = lambda t: [nuB * numpy.exp(numpy.log(nuF/nuB)
                                         * t / T)]

    sts = moments.LinearSystem_1D.steady_state_1D(ns[0])
    fs = moments.Spectrum(sts)
    fs.integrate(nu_func, T)

    return fs
```

Listing 2: **Bottleneck:** At time `TF + TB` in the past, an equilibrium population goes through a bottleneck of depth `nuB`, recovering to relative size `nuF`.

1. With old approach: bottleneck with growth, IM model
2. With demography: Gutenkunst model, Archaic model (w/ denisovans, neand, papuan, EA, EU, Afr)

7 Simulation and fitting the model

7.1 Running the model

7.2 Comparing model to data

7.2.1 Likelihoods

7.2.2 Fitting

1. Parameter bounds
2. Fixed parameters
3. Other options (there are a lot more now)
4. optimizer choice

7.3 Uncertainty analysis

8 Plotting

8.1 Visualizing LD curves

8.2 Residuals

Based on expectations, observations, and covariances

9 The full two-locus frequency spectrum

9.1 `moments.TwoLocus`

9.1.1 Specifying models

9.1.2 Parameters

9.1.3 Selection

10 Frequently asked questions

APR: No one has really asked me questions yet, but here are my own quandaries and answers:

1.

2. How do I cite `mold`?

3. What if I'm having issues running this program?

   Bug: issues

   Bigger issues or difficulties: email

11 Acknowledgements

`moments`, `mold`, and their manuals (and indeed much of my scientific course and accomplishments) are greatly indebted to Ryan Gutenkunst. Not only are these programs modeled off of ∂a∂i's interface and functionality, but also my general approach to writing, testing, and using scientific software has largely been guided and influenced by Ryan. I cannot overstate my gratitude to Ryan for making ∂a∂i open source and accessible, and more personally for his mentorship over the years (if you're reading this Ryan, a gigantic "Thank you!").

References

Hill, W. G. and Robertson, A. (1968). Linkage disequilibrium in finite populations. *Theoretical and Applied Genetics*, 38(6):226–231.

Ohta, T. and Kimura, M. (1971). Linkage disequilibrium between two segregating nucleotide sites under the steady flux of mutations in a finite population. *Genetics*, 68(4):571–580.

Ragsdale, A. P. and Gravel, S. (2018). Models of archaic admixture and recent history from two-locus statistics. *BioRxiv*, doi:10.1101/489401.

Ragsdale, A. P. and Gravel, S. (2019). Unbiased estimation of linkage disequilibrium from unphased data. *BioRxiv*, doi:10.1101/557488.