`discoal`– a coalescent simulator with selection

Andrew D. Kern

kern@biology.rutgers.edu

May 1, 2016

This file serves as the documentation for `discoal`, a program aimed at generating samples under a coalescent model with recombination, step-wise changes in population size, and selection. For users familiar with Richard Hudson's `ms`, the usage of `discoal` will be quite familiar, and indeed was meant to "play nice" with programs the user, or others, may have written for analyzing `ms` style output. `discoal` is not meant to take the place of `ms`, indeed in comparison it is quite limited in what it can do, but instead is meant to add a few models not covered by other simulation programs. In particular, `discoal` can quickly generate samples from models with selective sweeps in the history of the sample along with stepwise population size changes.

`discoal` gets is name from the contraction of "discrete" and "coalescent", because it handles recombination along the chromosome, and its associated programatic bookkeeping, by modeling a discrete number of ancestral sites. This leads to a trade off favoring speed over memory usage. For larger samples and sites modeled, do not be surprised if `discoal` eats up a lot of available RAM, but it will do so to increase the speed at which samples can be generated. This is particularly noticeable for larger recombination rates.

Download and Compile

The source code for `discoal` is available for download from `https://github.com/kern-lab`. Unpack the archive as you would any zip file and to compile you can use the following commands at a terminal:

```
$ cd <PATH TO DISCOAL>
$ make discoal
```

`discoal` should build on most systems without a problem. We have built it on numerous linux systems as well as OS X machines.

Basic usage

As we said above, using `discoal` is very familiar for people used to `ms` style commands. At its most basic a `discoal` command line looks like the following

```
$ ./discoal sampleSize sampleNumber nSites -t theta
```

Here there are four arguments we are passing to `discoal`: sampleSize – the size of each sample, sampleNumber – the number of independent samples to generate, nSites– the number of sites in the sequence to be modeled, and then following the `-t` flag $\theta = 4N_0u$, the mutation rate where N_0 is the population size currently and u is the mutation rate. This is identical to the command lines handed to `ms` with the addition of the nSites parameter. A representative run with its associated output might look like the following:

```
$ ./discoal 3 2 100 -t 2
./discoal 3 2 100 -t 2
1665047201 686400060


//
```

```
segsites: 4
positions: 0.01835 0.09557 0.46556 0.72880
1000
0110
0111

//
segsites: 1
positions: 0.07594
0
1
0
```

Again, we are following Hudson's lead here with output formatted just as `ms` does. This should mean that any secondary analysis software used for `ms` output should work for `discoal` output as well. One note about the seeding of the random number generator (seeds are given on the second line of the output)– seeds are taken from `/dev/urandom`, a special file on most ⋆nix systems that creates pseudorandom numbers from collecting thermal noise from the devices on the machine. This makes `discoal` suitable out of the box for large scale cluster computing where multiple jobs will be launch simultaneously without having to worry about random number generator seeds.

Recombination and Gene Conversion

Recombination (crossing over) is handled by `discoal` by adding the `-r` flag. `-r` takes one parameter, $\rho = 4Nr$ the population recombination rate where r is the probability of a cross over per basepair of sequence being modeled and N is the current population size. Recombination can occur between any of the discrete nSites being modeled. A representative

call with recombination would be:

```
$ ./discoal 3 2 100 -t 2 -r 2.4
```

Gene conversion (recombination without exchange of flanking markers; sometimes called a non-crossover) within the modeled chromosome segment is also simulated by `discoal` by using the `-g` flag. The `-g` option takes two parameters, $\gamma = 4Ng$ the population gene conversion rate where g is the probability of initiating an NCO event per basepair. The second parameter needed is the mean gene conversion tract length, as once a gene conversion (NCO) is initiated it is extended for a geometrically distributed length. For instance:

```
$ ./discoal 3 2 100 -t 2 -r 2.4 -g 2.4 10
```

would specify $\gamma = 2.4$ with a mean gene conversion tract length of 10bp.

Population size changes

`discoal` can simulate an arbitrary number of step-wise, instantaneous population size changes. Each population size change in the sample history is set with a `-en` flag which specifies the time of the population size change, the ID of the population to change (zero indexed), and the ratio of the new population size to the current population size, N_0. At this point its worth noting that `discoal` measures time in units of $2N_0$ generations, rather that $4N_0$ as in `ms`. If you forget this point, don't worry it says as much in the usage line of the program if one runs `discoal` with no arguments. Multiple changes in population size can be specified by adding additional `-en` statements to the command line. By way of example, the following command line specifies a bottleneck population history where at time 0.5 the population crashes to 10% of its initial size and then at time 1.2 it rebounds to 80% of its initial size.

```
$ ./discoal 3 2 100 -t 2 -r 2.4 -en 0.5 0 0.1 -en 1.2 0 0.8
```

Multiple populations

`discoal` can simulate the coalescent for multiple populations under models with both continuous gene flow and pulse admixture events. The `-p` flag establishes the number of populations and the respective sample sizes from each population. Note that the sample size of a population can be zero, indicating no sampled individuals came from that population. The `-M` flag sets all migration rates between populations to a specific value (scaled in units of $4N_0$). The `-m` flag allows individual migration rates between populations to be set separately. Here is an example of a 3 population island model where 2 alleles have been sampled from each population and where there is symmetric migration at rate $4N_0m = 0.05$

```
$ ./discoal 6 2 100 -t 2 -r 2.4 -p 3 2 2 2 -M 0.05
```

Population splitting events (forward in time) can use modeled using the `-ed` flag. As above, its important to note that `discoal` uses zero indexing for populations. Let's imagine we have three populations that diverge from one another such that two are sister taxa and one is an outgroup. Call p_0 the $0th$ population, then our tree might be ((p_0,p_1),p_2). We can specify this topology and its associated divergence times (1.0 and 5.0) as

```
$ ./discoal 6 2 100 -t 2 -r 2.4 -p 3 2 2 2 -ed 1.0 0 1 -ed 5.0 1 2
```

Note the command above doesn't have migration and is thus pure isolation. You can add that back in using the `-m` or `-M` flag. We might want to imagine that at there was an admixture between p_0 and p_2 recently, say 0.02 time units in the past, and that 15% of individuals p_0 came from p_2 at that time. We could add this to our model using the `-ea` flag

```
$ ./discoal 6 2 100 -t 2 -r 2.4 -p 3 2 2 2 -ed 1.0 0 1 -ed 5.0 1 2 -ea 0.02 0 1 0 0.15
```

The `-ea` flag takes as its arguments a time for the event, the ID of the admixed population, the IDs of the two founding populations (one can be the same as the admixed population), and the proportion of ancestry that comes from the first founding population ID listed.

One important note about using multiple population together with selection: `discoal` only models selective sweeps in population zero (p_0). When the simulation enters into a sweep phase we do not allow for migration. If we were to, we would need to model the allele trajectory in two or more populations concurrently, and we have not implemented this possibility. Also, we have not yet implemented an event that would allow migration rates to change over time, this is planned and should be available soon.

Selection

`discoal` can simulate samples with single selective sweeps in the history of a sample, requiring the site under selection to be within the bounds of the modeled locus. This is done using the now conventional technique of altering the genealogy of a sample to be conditional upon the trajectory of an allele moving through the population to eventual fixation (forward in time) [1, 4]. `discoal` can simulate both deterministic sweep trajectories [1, 4] and stochastic trajectories [2, 6]. In addition coalescent simulations can be generated conditional upon the fixation of a neutral mutation in the population [7]. All of these cases are easily handled by `discoal` using the group of `-w` flags. `-wd` generates deterministic selective sweeps, `-ws` stochastic selective sweeps, and `-wn` performs neutral fixations. Each `-w` flag takes one parameter, τ the time of fixation looking backward in time. Thus if $\tau = 0$ the fixation of the focal site has occurred immediately prior to sampling. Two other parameters are necessary when specifying sweeps. 1) the location of the site that has fixed using the `-x` flag. `-x` for convenience takes a floating point number between 0 and 1, thus translating the discrete sites modeled to the real line (defaults to 0.5). 2) The strength of selection given as $\alpha = 2Ns$ specified via the `-a` flag.

Building on our previous example then, lets generate a sample with a single selective sweep at position 50 (the middle) in our locus with $\alpha = 1000$ and $\tau = 0.05$ using stochastic sweep trajectories

```
$ ./discoal 3 2 100 -t 2 -r 2.4 -ws 0.05 -a 1000 -x 0.5
```

This can also be combined with populations size changes to generate samples with stepwise constant population size and selective sweeps

```
$ ./discoal 3 2 100 -t 2 -r 2.4 -ws 0.05 -a 1000 -x 0.5 -en 0.5 0 0.1 -en 1.2 0 0.8
```

Another feature of `discoal` is its ability to generate what are now being called soft selective sweeps [3, 6]. To generate sweeps from standing variation the user can specify f_0, the frequency at which a previously neutral allele became beneficial. This is done using `-f` flag. For such situations, conditional allele frequency trajectories for the sweep site are generated as per [6]. If one is instead interested in a soft sweep model with recurrent mutation towards a beneficial allele the `-uA` flag generates sweeps where one of the possible moves is a change in selective class through mutation for alleles not originally linked to the beneficial mutation [5].

As an example, we can take our previous hard sweep command line and adjust it so that the mutation drifts until frequency 0.1 where it then becomes beneficial

```
$ ./discoal 3 2 100 -t 2 -r 2.4 -ws 0.05 -a 1000 -x 0.5 -f 0.1
```

Simulating selective sweeps, beyond the sampled locus

Often we are interested in examining the effects of linked selection on a neutral locus, rather than examining the site of selection itself. To facilitate this `discoal` will perform simulations with sweeps where the sweep occurs at a specified genetic distance to the left of the locus sampled as well as the time of the sweep. This occurs if the user specifies the `-ls`, `-ld`, or `-ln` options corresponding to a stochastic selective sweep, a deterministic selective sweep, or a neutral fixation respectively. In this case the coalescent simulation is carried out according to the Braverman *et al.* (1995) model, in which distance between the selected and neutral locus is scaled as the expected number of recombination events (i.e. $2Nr$). This is

important to keep in mind for the user as recombination distances within the sampled locus will be in units of $4Nr$.

Priors on parameters

Sometimes when generating simulations one is interested in simulating over a range of parameter values. `discoal` allows uniform priors to be set on all of its parameters with the `-P` family of flags. It can generate priors on population size changes as well, but at present can only be used for up to two size changes.

Further support

If you are having trouble compiling or running the software don't hesitate to contact me via email or phone. Also please please report all bugs that you might find!

References

[1] J M Braverman, R R Hudson, N L Kaplan, C H Langley, and W Stephan. The hitchhiking effect on the site frequency spectrum of dna polymorphisms. *Genetics*, 140(2):783–796, Jun 1995.

[2] Graham Coop and Robert C Griffiths. Ancestral inference on gene trees under selection. *Theor Popul Biol*, 66(3):219–232, 2004.

[3] Joachim Hermisson and Pleuni S Pennings. Soft sweeps: molecular population genetics of adaptation from standing genetic variation. *Genetics*, 169(4):2335–52, 2005.

[4] Yuseob Kim and Wolfgang Stephan. Detecting a local signature of genetic hitchhiking along a recombining chromosome. *Genetics*, 160(2):765–77, Feb 2002.

[5] Pleuni S Pennings and Joachim Hermisson. Soft sweeps ii–molecular population genetics of adaptation from recurrent mutation or migration. *Mol Biol Evol*, 23(5):1076–84, 2006.

[6] Molly Przeworski, Graham Coop, and Jeffrey D Wall. The signature of positive selection on standing genetic variation. *Evolution*, 59(11):2312–23, Nov 2005.

[7] F Tajima. Relationship between dna polymorphism and fixation time. *Genetics*, 125(2):447–454, 1990.