Population genetics: inference using the coalescent

Peter Beerli

November 6, 2005

1 Simple estimators

The coalescent gives an excellent framework for population genetics, but does not really talk about inference, in many cases we would need to know the genealogy (topology and times) or some relationship between number of variable sites and population size. Watterson constructed an estimator using the number of segregating sites assuming an infinite sites mutation model and his estimator is

$$\theta = S / \sum_{i=1}^{n-1} \frac{1}{i}$$

where S is the number of segregating sites in the sample. Recognize that the population size here is $\theta = 4N\mu_{\text{locus}}$. The mutation rate is typically used on a per locus basis. Watterson's estimator is very simple and delivers good estimates, for more complicated scenarios there are no such simple estimators available. Simple estimators for more difficult scenarios often assumed that the true genealogy is known. Most often we have no clue about the true genealogy but we approximate this true genealogy using the data and generate the best genealogy using phylogenetic inference. This approach has a difficulty because it only works well when this one tree is much better than all the others. Several methods to estimate population growth are based on this method, for example the skyline plot approach by Pybus and friends.

2 Maximum likelihood

Estimating parameters of a population genetic model is rather simple under the likelihood principle because we can calculate probabilities for the Kingman coalescent, we also know how to calculate trees from genetic data assuming a mutation model. But how to do this in praxis. We simple start with a rather basic observation that we want in principle to to find the values of the parameters Ψ, for example $\Psi = (\Theta_1, \Theta_2, ..., \Theta_n, M_{21}, ...M_{n-1,n})$ that have the highest probability and perhaps we also want to find the values that are in a credible set around that highest probability value. for that we would like to calculate Prob(Ψ|Data). Using Bayes' theorem

$$\text{Prob}(\Psi|\text{Data}) = \frac{\text{Prob}(\Psi)\text{Prob}(\text{Data}|\Psi)}{\text{Prob}(\text{Data})}.$$

For a given data set we can assume that Prob(Data is constant and we also can assume that if the prior distribution Prob(Ψ) does not convey any particular information, we simply could look at the likelihood Prob(Data|Ψ) = $L(\Psi)$. Frequentist prefer likelihood because it does not force a prior opinion onto the reader of the results. Wether this is a good or bad thing might depend on the study. Expressing our problem this way does not help much because we should also specify the models used

$$\begin{aligned}\text{Prob}(\text{Data}|\Psi) =& L(\Psi) = f(\Psi, \text{coalescence model}, \text{mutation model}) \\ =& \int_G \text{p}(G|\Psi)\text{Prob}(D|G)dG \qquad (1)\end{aligned}$$

this looks easy because we already learned how to calculate all the quantities shown, but because we also have learned that there are a very large number of potential genealogies we need to do MCMC and things get somewhat more complicated. How to translate our formula 1 into an algorithm?

2.1 Derivation of the importance sampling function

We want to calculate formula (1) but would need to sum over all possible genealogies. This functions can be transformed into an importance sampling function by assuming that $L(\Psi)$ is an expectation and we sample from a distribution whose density is g instead of the correct density, f

$$\begin{aligned}L(\Psi) = \text{Prob}(D|\Psi) &= \sum_G \text{Prob}(G|\Psi)\text{Prob}(D|G) &(2)\\ &= \mathbb{E}_f(h) = \sum_G hf &(3)\\ &= \sum_G h\frac{f}{g}g = \mathbb{E}_g(h\frac{f}{g}) &(4)\end{aligned}$$

Suppose that

$$\begin{aligned}g &= \frac{\text{Prob}(G|\Psi_0)\text{Prob}(D|G)}{\sum_G \text{Prob}(G|\Psi_0)\text{Prob}(D|G)}, &(5)\\ h &= \text{Prob}(D|G), &(6)\end{aligned}$$

and

$$f = \frac{\text{Prob}(G|\Psi)}{\sum_G \text{Prob}(G|\Psi)} = \text{Prob}(G|\Psi), \tag{7}$$

since

$$\sum_G \text{Prob}(G|\Psi) = 1, \tag{8}$$

then we have

$$\mathbb{E}_f(h) = \sum_G \frac{\text{Prob}(G|\Psi)}{\sum_G \text{Prob}(G|\Psi)} \text{Prob}(D|G) = \sum_G \text{Prob}(G|\Psi)\text{Prob}(D|G) \tag{9}$$

$$= \mathbb{E}_g(\frac{f}{g}h) = \mathbb{E}_g\left(\frac{\text{Prob}(G|\Psi)\text{Prob}(D|G)}{\left(\frac{\text{Prob}(G|\Psi_0)\text{Prob}(D|G)}{\sum_G \text{Prob}(G|\Psi_0)\text{Prob}(D|G)}\right)}\right), \tag{10}$$

so that

$$L(\Psi) = L(\Psi_0)\ \mathbb{E}_g\left(\frac{\text{Prob}(G|\Psi)\text{Prob}(D|G)}{\text{Prob}(G|\Psi_0)\text{Prob}(D|G)}\right). \tag{11}$$

where

$$L(\Psi_0) = \sum_G \text{Prob}(G|\Psi_0)\text{Prob}(D|G).$$

The expectation can be estimated by its average over the simulation

$$\frac{L(\Psi)}{L(\Psi_0)} = \frac{1}{m}\sum_i^m \frac{\text{Prob}(G_i|\Psi)}{\text{Prob}(G_i|\Psi_0)}. \tag{12}$$

In Markov chain Monte Carlo approaches the goal is to sample from the posterior and concentrate the sampling on regions with higher probabilities. In our scheme we are approximating the target function $\text{Prob}(G|\Psi)\text{Prob}(D|G)$ with $\text{Prob}(G|\Psi_0)\text{Prob}(D|G)$, this may be a rather crude approximation when Ψ_0 is very different from Ψ, but after several updating chains the target and sampling distribution are verysimilar, this approach is at least for later chains in our approach with ten short and two long chains nearly optimal. Sampling from the prior alone $\text{Prob}(G|\Psi)$, for example, is very inefficient, as was tried by Felsenstein in 1988.

2.2 Finding the MLE

To find the maximum likelihood estimate we simply maximize the above function. This has proven more difficult than we thought at the beginning, one needs to maximize in many dimension and simple bisection does not help a lot. The maximization problem has driven many people towards Bayesian approaches because there one does not need to do it. But....

The likelihood version of `migrate` uses an interactive improvement of the driving parameter Θ_0, following this algorithm:

Algorithm 1 MLE approach as implemented in `migrate`

```
Choose an arbitrary $\Theta_0$ to drive the Markov chain.
(2) Run a "short" Markov chain using this driving parameter $\Theta_0$, typically several thousand genealogies are sampled from $\text{Prob}(\cdot|\Theta_0)\text{Prob}(D|\cdot)$.
Calculate the maximum likelihood estimate of $\Theta$ based on the sampled genealogies.
Replace the driving value: $\Theta_0 \leftarrow \Theta$.
Continue at (2) for 5-10 "short" chains.
Redo 2 to 5 for "long" chains, typically tens or hundreds of thousands are evaluated.
Report last $\Theta$
```

3 Bayesian estimators

We used Bayes theorem earlier for the justification of likelihood but of course we can use it straight to do Bayesian inference

$$\text{Prob}(\Psi|\text{Data}) = \frac{\text{Prob}(\Psi)\text{Prob}(\text{Data}|\Psi)}{\text{Prob}(\text{Data})}.$$

We draw priors from arbitrary distributions (see Bayesian inference chapter) and calculate the likelihood (formula 1). The denominator is often not explicitely calculated. We calculate for an coalescent estimator

$$\text{Prob}(\Psi|D) \sim \text{Prob}(\Psi) \int_G \text{p}(G|\Psi)\text{Prob}(D|G)dG \tag{13}$$

For example we could evaluate this algorithm 2

4 Study questions

1. Difference between likelihood and Bayesian that is typical for coalescent approaches?

Algorithm 2 Bayesian approach as implemented in `migrate`

```
set start parameters $\Psi$
generate start genealogy
loop {until you get tired to wait for results}
    calculate the probability of the current configuration (parameters and genealogy)
    $f_o = prob(\Psi)\mathrm{Prob}(G|\Psi)\mathrm{Prob}(D|G)$
    with frequency x draw a random parameter or the genealogy for updating
    update the parameter using a prior distribution or update the genealogy
    calculate the probability of the new configuration (parameters and genealogy) $f_n = \mathrm{Prob}(\Psi)\mathrm{Prob}(G|\Psi)\mathrm{Prob}(D|G)$
    accept or reject using the Metropolis-Hastings Ratio $\frac{f_n}{f_o}\frac{q(x_o|x_n)}{q(x_n|x_o)}$
    if accepted then
        use new configuration
        store new parameter set
    else
        use old configuration
        store old parameter set
    end if
end loop
```