1 The model

Consider a diploid population of effective size N_e. Polymorphism data from K types of sites are available ($K \geq 2$). Insertions and deletions are referred to as variants of type 1 and 2, respectively, whereas neutral SNPs are of type K; other types of variants, which must be SNPs (e.g., 0-fold SNPs), are ordered arbitrarily in between. We summarise the data using the site-frequency spectrum (SFS). Specifically, let $\boldsymbol{d}_k = (d_{k,1}, d_{k,2}, ..., d_{k,n-1})$, where $k \in \{1, .., K\}$ and $d_{k,i}$ is the number of sites of type k where the derived allele is presented i times in a sample of size n. We define the scaled mutation rate to insertions as $\theta = 4N_e u_1$, where u_1 the mutation rate per site per generation. To model mutational bias, we define $\kappa = u_2/u_1$, where u_2 is the deletion mutation rate per site per generation, so that the scale mutation rate to deletions is $\theta\kappa$. For the other types of variants, we assume that there is a separate mutation rate parameter for each type of variants, defined as $\theta_k = 4N_e u_k$ (for $k \geq 3$). For ease of presentation, θ and θ_1 will be used interchangeably below; the same applies to $\theta\kappa$ and θ_2.

Selection is assumed to be additive, so that, at a polymorphic site of type k where the ancestral allele, A_0, and the derived allele, A_1, co-segregate, the fitnesses of the three possible genotypes, A_0A_0, A_0A_1, and A_1A_1, are 1, $1 + s_k$, and $1 + 2s_k$, respectively. The scaled selection coefficients can be defined accordingly as $\gamma_k = 4N_e s_k$. For the neutral SNPs, $\gamma_K \equiv 0$.

According to standard population genetic theory, given θ_k, γ_k, and the number of sites of type k sequenced, m_k, the expected number of segregating sites where the derived allele is represented i times ($i \in \{1, ..., n-1\}$), denoted by $\psi_{k,i}$, is given by

$$\psi_{k,i} = m_k \theta_k r_i \int_0^1 B(i|n, x) f(x, \gamma_k) dx, \tag{1}$$

where

$$B(i|n, x) = \binom{n}{i} x^{i-1}(1-x)^{n-i-1},$$

$$f(x, \gamma_k) = \frac{1 - e^{-\gamma_k(1-x)}}{1 - e^{-\gamma_k}},$$

and r_i are for controlling demographic effects, with $r_1 \equiv 1$. For numerical stability, we use the following when γ_k is small:

$$f(x,\gamma_k) =$$
$$\frac{1}{24}(1-x)\Big(24+\gamma_k x\big[12+\gamma_k\big(-2+x(4-\gamma_k(1-x))\big)\big]\Big)+O(\gamma_k^4) \tag{2}$$

When $\gamma = 0$, (1) reduces to

$$\psi_{k,i} = \frac{m_k\theta_k r_i}{i}. \tag{3}$$

For convenience, we denote the integral in (1) as $\tau_i(\gamma_k)$, so that $\psi_{k,i}$ can be rewritten as:

$$\psi_{k,i} = m_k\theta_k r_i \tau_i(\gamma_k). \tag{4}$$

To model polarisation errors for insertions and deletions, we assume that, for a polymorphic site of type k ($k \in \{1,2\}$), the probability that its ancestral state is misidentified is e_k. Because when the ancestral state of a derived insertion mutation with i copies in the sample is misidentified, it will be wrongly identified as a deletion with $n-i$ copies. Thus, the expected SFS for insertions including polarisation errors, denoted by $\Psi_{1,i}$, can be written as

$$\begin{aligned}\Psi_{1,i} &= (1-e_1)\psi_{1,i} + e_2\psi_{2,n-i} \\ &= \begin{cases}(1-e_1)m_1\theta r_i\tau_i(\gamma_1) + e_2 m_2\theta\kappa r_{n-i}\tau_{n-i}(\gamma_2), & \text{if } i \neq n-i; \\ (1-e_1)m_1\theta r_i\tau_i(\gamma_1) + e_2 m_2\theta\kappa r_i\tau_i(\gamma_2), & \text{if } i = n-i.\end{cases}\end{aligned} \tag{5}$$

The SFS for deletions is

$$\begin{aligned}\Psi_{2,i} &= (1-e_2)\psi_{2,i} + e_1\psi_{1,n-i} \\ &= \begin{cases}(1-e_2)m_2\theta\kappa r_i\tau_i(\gamma_2) + e_1 m_1\theta r_{n-i}\tau_{n-i}(\gamma_1), & \text{if } i \neq n-i; \\ (1-e_2)m_2\theta\kappa r_i\tau_i(\gamma_2) + e_1 m_1\theta r_i\tau_i(\gamma_1), & \text{if } i = n-i.\end{cases}\end{aligned} \tag{6}$$

For the SNPs (i.e., for $k \in \{3,...,K\}$), we can similarly define

$$\begin{aligned}\Psi_{k,i} &= (1-e_k)\psi_{k,i} + e_k\psi_{k,n-i} \\ &= \begin{cases}(1-e_k)m_k\theta_k r_i\tau_i(\gamma_k) + e_k m_k\theta_k r_{n-i}\tau_{n-i}(\gamma_k), & \text{if } i \neq n-i; \\ m_k\theta_k r_i\tau_i(\gamma_k), & \text{if } i = n-i.\end{cases}\end{aligned} \tag{7}$$

The data, $\boldsymbol{d} = (\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_K)$, are dependent on the following parameters: θ_1, ..., θ_K, γ_1, ..., γ_{K-1}, e_1, ..., e_K, and r_2, ..., r_{n-1}, which we denote collectively as $\boldsymbol{\Theta}$. We assume that the observed number of segregating sites where the derived allele is represented i times follows a Poisson distribution with mean Ψ_i (Poisson random field). The likelihood function is:

$$l(\boldsymbol{d}|\boldsymbol{\Theta}) = \prod_{k=1}^{K}\prod_{i=1}^{n-1} e^{-\Psi_{k,i}}\frac{(\Psi_{k,i})^{d_{k,i}}}{d_{k,i}!} \tag{8}$$

Taking the natural logarithm, we have

$$L(\boldsymbol{d}|\boldsymbol{\Theta}) = \sum_{k=1}^{K}\sum_{i=1}^{n-1}\Big(-\Psi_{k,i} + d_{k,i}\ln(\Psi_{k,i})\Big) + C, \tag{9}$$

where C is a constant and can be ignored when trying to maximise the ln-likelihood.

To make use of more efficient search algorithms, it is necessary to derive the partial derivatives of (9). First, for $\Psi_{1,i}$ we have

$$\begin{cases} \frac{\partial\Psi_{1,i}}{\partial\theta} = (1-e_1)m_1 r_i\tau_i(\gamma_1) + e_2 m_2\kappa r_{n-i}\tau_{n-i}(\gamma_2); \\ \frac{\partial\Psi_{1,i}}{\partial\kappa} = e_2 m_2\theta r_{n-i}\tau_{n-i}(\gamma_2); \\ \frac{\partial\Psi_{1,i}}{\partial\gamma_1} = (1-e_1)m_1\theta_1 r_i\frac{d\tau_i(\gamma_1)}{d\gamma_1}; \\ \frac{\partial\Psi_{1,i}}{\partial\gamma_2} = e_2 m_2\theta_2 r_{n-i}\frac{d\tau_{n-i}(\gamma_2)}{d\gamma_2}; \\ \frac{\partial\Psi_{1,i}}{\partial e_1} = -m_1\theta_1 r_i\tau_i(\gamma_1) = -\psi_{1,i}; \\ \frac{\partial\Psi_{1,i}}{\partial e_2} = m_2\theta_2 r_{n-i}\tau_{n-i}(\gamma_2) = \psi_{2,n-i}. \end{cases} \tag{10}$$

The equations for $\Psi_{2,i}$ are similar:

$$\begin{cases} \frac{\partial\Psi_{2,i}}{\partial\theta} = (1-e_2)m_2\kappa r_i\tau_i(\gamma_2) + e_1 m_1 r_{n-i}\tau_{n-i}(\gamma_1); \\ \frac{\partial\Psi_{2,i}}{\partial\kappa} = (1-e_2)m_2\theta r_i\tau_i(\gamma_2); \\ \frac{\partial\Psi_{2,i}}{\partial\gamma_1} = e_1 m_1\theta_1 r_{n-i}\frac{d\tau_{n-i}(\gamma_1)}{d\gamma_1}; \\ \frac{\partial\Psi_{2,i}}{\partial\gamma_2} = (1-e_2)m_2\theta_2 r_i\frac{d\tau_i(\gamma_2)}{d\gamma_2}; \\ \frac{\partial\Psi_{2,i}}{\partial e_1} = m_1\theta_1 r_{n-i}\tau_{n-i}(\gamma_1) = \psi_{1,n-i}; \\ \frac{\partial\Psi_{2,i}}{\partial e_2} = -m_2\theta_2 r_i\tau_i(\gamma_2) = -\psi_{2,i}. \end{cases} \tag{11}$$

For the non-neutral SNPs, we have

$$\begin{cases} \frac{\partial\Psi_{k,i}}{\partial\theta_k} = m_k\big[(1-e_k)r_i\tau_i(\gamma_k) + e_k r_{n-i}\tau_{n-i}(\gamma_k)\big], \\ \frac{\partial\Psi_{k,i}}{\partial\gamma_k} = m_k\theta_k\big[(1-e_k)r_i\frac{d\tau_i(\gamma_k)}{d\gamma_k} + e_k r_{n-i}\frac{d\tau_{n-i}(\gamma_k)}{d\gamma_k}\big], \\ \frac{\partial\Psi_{k,i}}{\partial e_k} = m_k\theta_k\big[-r_i\tau_i(\gamma_k) + r_{n-i}\tau_{n-i}(\gamma_k)\big], \text{ if } i \neq n-i, \\ \frac{\partial\Psi_{k,i}}{\partial e_k} = 0, \text{ if } i = n-i. \end{cases} \tag{12}$$

For the neutral SNPs, we just need to exclude the second equation from (12) and setting γ_K to zero in the other two equations. For the derivatives with respect to r_i, we have, for $i \neq n-i$,

$$\begin{cases} \frac{\partial \Psi_{k,i}}{\partial r_i} = (1-e_k) m_k \theta_k \tau_i(\gamma_k), \text{ for } k \in \{1, ..., K\} \\ \frac{\partial \Psi_{1,i}}{\partial r_{n-i}} = e_2 m_2 \theta_2 \tau_{n-i}(\gamma_2) \\ \frac{\partial \Psi_{2,i}}{\partial r_{n-i}} = e_1 m_1 \theta_1 \tau_{n-i}(\gamma_1) \\ \frac{\partial \Psi_{k,i}}{\partial r_{n-i}} = e_k m_k \theta_k \tau_{n-i}(\gamma_k), \text{ for } k \in \{3, ..., K\} \end{cases} \tag{13}$$

When $i = n-i$, we have

$$\begin{cases} \frac{\partial \Psi_{1,i}}{\partial r_i} = (1-e_1) m_1 \theta_1 \tau_i(\gamma_1) + e_2 m_2 \theta_2 \tau_i(\gamma_2) \\ \frac{\partial \Psi_{2,i}}{\partial r_i} = (1-e_2) m_2 \theta_2 \tau_i(\gamma_2) + e_1 m_1 \theta_1 \tau_i(\gamma_1) \\ \frac{\partial \Psi_{k,i}}{\partial r_i} = m_k \theta_k \tau_i(\gamma_k), \text{ for } k \in \{3, ..., K\} \end{cases} \tag{14}$$

With the above, we can calculate the partial derivative of the ln-likelihood function as:

$$\frac{\partial L(\boldsymbol{d}|\boldsymbol{\Theta})}{\partial x} = \sum_{k=1}^{K} \sum_{i=1}^{n-1} \big(-1 + \frac{d_{k,i}}{\Psi_{k,i}} \big) \frac{\partial \Psi_{k,i}}{\partial x}, \tag{15}$$

where $x \in \boldsymbol{\Theta}$. If a parameter x is transformed, such that $y = h(x)$ and $x = h^{-1}(y)$, using the chain rule, we have

$$\begin{aligned} \frac{\partial L(\boldsymbol{d}|\boldsymbol{\Theta})}{\partial y} &= \sum_{k=1}^{K} \sum_{i=1}^{n-1} \left(-\frac{\partial \Psi_{k,i}(h^{-1}(y))}{\partial y} + d_{k,i} \frac{\partial \ln[\Psi_{k,i}(h^{-1}(y))]}{\partial y} \right) \\ &= \sum_{k=1}^{K} \sum_{i=1}^{n-1} \left(-1 + d_{k,i} \frac{d\ln(z)}{dz}\bigg|_{z=\Psi_{k,i}(h^{-1}(y))} \right) \frac{\partial \Psi_{k,i}(h^{-1}(y))}{\partial y} \\ &= \sum_{k=1}^{K} \sum_{i=1}^{n-1} \big(-1 + \frac{d_{k,i}}{\Psi_{k,i}(x)} \big) \frac{\partial \Psi_{k,i}(z)}{\partial z}\bigg|_{z=h^{-1}(y)} \frac{dh^{-1}(z)}{dz}\bigg|_{z=y}, \end{aligned} \tag{16}$$

where $\boldsymbol{\Theta}_{tr}$ represents the case whereby some of the parameters are transformed.

2 Comparing derivative-free and derivative-based search algorithms

Using `test_model_sim` and the following command line options:

```
test_model_sim /Users/Kai/Data/netbeans-c/Indel/tests/result /Users
   /Kai/Data/netbeans-c/Indel/tests/log/ 30 10 "500, 600, -5, 0,
   0.01, 0.05" "1, 10000, -50, 50, 0, 1" 10 1e-10 1 50
```

The number of function evaluations and the return MLE estimates were compared. The MLEs are generally very similar. But the simplex algorithm sometimes returned sub-optimal estimates (out of the 10 independent runs). But the derivative-based algorithms seem to always give consistent results across runs. The following table compares the average number of function evaluations. Note that, having the evaluate the derivatives, the actual number of function evaluations for the derivative-based methods should be about twice the number shown in the table.

Algorithm	Mean # func. eval.	Derivative?
COBYLA	Failed	N
BOBYQA	696.98	N
PRAXIS	Failed	N
SBPLX	1268.59	N
Nelder-Mead (simplex)	941.67	N
TNEWTON_PRECOND_RESTART	7254.97	Y
MMA	133.92	Y
VAR1	96.81	Y
VAR2	96.81	Y
L-BFGS	84.53	Y
SLSQP	62.14	Y

`MMA:` This frequently failed (hitting the maximum number of function evaluations).

`PRAXIS:` Not good for problems with bound constraints.

`COBYLA:` Constantly hitting the maximum number of function evaluations.