1 Introduction

The Fortran program 'Imputator' consists of several imputation algorithms, all treating each SNP independently. The most sophisticated of the set is an implementation of the iterative peeeling algorithm described in detail in Kerr and Kinghorn [1996].

This algorithm splits the probability that individual i has actual genotype u_i at the SNP into four sources of information / equation building blocks:

- $g(y_i|u_i)$: the likelihood that the individuals own observed genotype y_i is correct, which is determined by the genotyping error rate (g is referred to as 'penetrance value' in Kerr and Kinghorn [1996]);
- $a(u_i)$: the anterior probability, based on the genotype probabilities of its ancestors and siblings (Equation 2, Figure 1);
- $p_{ij}(u_i)$: the posterior probability, based on the genotype probabilities of its descendants and mates (Equation 3, Figure 2).
- $tr(u_i|u_d, u_s)$: the inheritance probability; the probability that i has genotype u_i, given that parents d and s have genotypes u_d and u_s following Mendel's laws.

Note that these are not anterior and posterior probabilities in the Bayesian sense, but refer to being based on individuals anterior versus posterior in the pedigree relative to the focal individual.

2 Kerr & Kinghorn equations

The main equations in Kerr and Kinghorn [1996] are reproduced here, correcting a few type-setting errors in the original, and applying colour-coding to aid comprehension. Note that here subscripts d and s are used for the dam and sire, whereas Kerr and Kinghorn [1996] use m and f for mother and father (or male and female). Here M is used to refer to the set of mates of an individual, and in the next section subscript m is used for matings.

The colour coding scheme is the same as in Figures 1 and 2, highlighting the four main building blocks of the equations.

The probability that individual i has actual genotype u_i given the vector with observed genotypes $\mathbf{y}$ at this SNP is given by

$$Pr(u_i|\mathbf{y}) = h_i = \frac{a_i(u_i)g(y_i|u_i)\prod_{j\in S_i} p_{ij}(u_i)}{L} \tag{1}$$

where L is the likelihood for the pedigree, which in this context is just a scaling factor to ensure $sum_{u_i=0}^{2} Pr(u_i|\mathbf{y}) = 1$.

The anterior probability is calculated as:

$$\begin{aligned} a(u_i) = &\sum_{u_s}\Big\{a_s(u_s)g(y_s|u_s)\prod_{k\in M_s,k\neq d} p_{sk}(u_s) \\ &\times \sum_{u_d}\Big\{a_d(u_d)g(y_d|u_d)\prod_{k\in M_d,k\neq s} p_{dk}(u_d) \\ &\times tr(u_i|u_s,u_d) \\ &\times \prod_{f\in C_{sd},f\neq i}\Big[\sum_{u_f} tr(u_f|u_s,u_d)g(y_f|u_f)\prod_{k\in M_f} p_{fk}(u_f)\Big]\Big\}\Big\} \end{aligned} \tag{2}$$

Posterior probability:

$$\begin{aligned} p_{ij}(u_i) = &\sum_{u_j}\Big\{a_j(u_j)g(y_j|u_j)\prod_{k\in M_j,k\neq i} p_{jk}(u_j) \\ &\times \prod_{o\in C_{ij}}\Big[\sum_{u_o} tr(u_o|u_i,u_j)g(y_o|u_o)\prod_{k\in M_o} p_{ok}(u_o)\Big]\Big\} \end{aligned} \tag{3}$$

Where:

- i: focal individual
- s, d: parents (dam, sire) of i
- f: full sibling of i
- j: mate to i
- o: offspring of i
- $M_.$: Set of mates of 1 individual
- $C_{..}$: Set of offspring (Children) of 2 individuals

At initiation, for all individuals $a(u_i) = g(y_i|u_i)w(u_i)$, where $w(u_i)$ is the probability of genotype u_i under HWE, and $p_{ij}(u_i) = 1/3$. For founders, the anterior probability $a(u_i)$ remains unchanged.

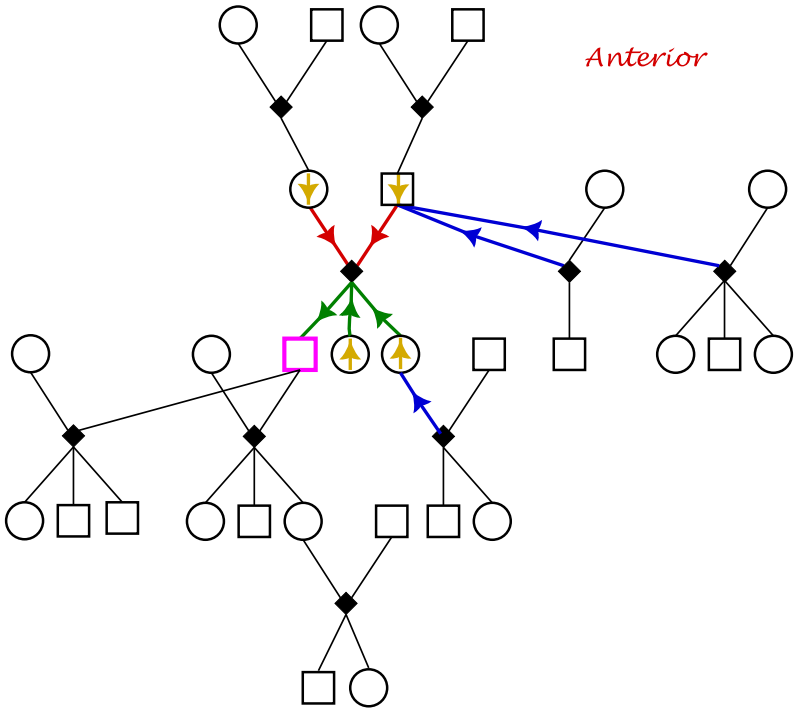

Figure 1: Example pedigree diagram showing contributions to the anterior probability of the focal individual (pink) from anterior (red), posterior (blue), inheritance (green) and penetrance (gold) probabilities.

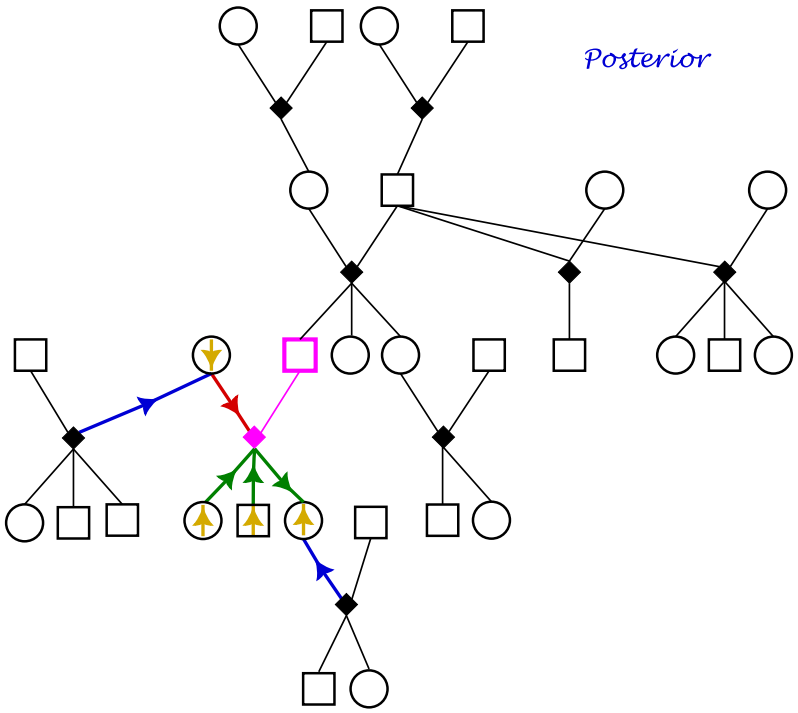

Figure 2: Example pedigree diagram showing contributions to the posterior probability of the focal mating/individual (pink) from anterior (red), posterior (blue), inheritance (green) and penetrance (gold) probabilities.

3 Implementation

3.1 Log scale

All calculations done on log scale for increased precision, but are for brevity and clarity written out here on the normal scale.

Summation requires transforming probabilities from log to normal scale, which may inadvertently lead to rounding to zero when the computer's maximum precision is exceeded. To avoid this, the LogSumExp (LSE) is calculated following https://en.wikipedia.org/wiki/LogSumExp .

Scaling probabilities to sum to unity can cause similar issues. To avoid this, if the LSE is -INF, the non-scaled values are used. If after scaling any of the values is -INF, its value is restored to the non-scaled value. These steps do not affect the imputed values, but without them the convergence check of the iterative peeling algorithm (all probabilities differ by less than `tol`) does not work.

3.2 Non-genotyped individuals

Often the pedigree consists of both genotyped and non-genotyped individuals. With the option `--impute-all` all non-genotyped individuals in the pedigree are added to the bottom of the genotype data as completely blank lines to be imputed.

Under the default settings, non-gentoyped parents of genotyped individuals are added as 'ghosts' to the genotype data in a similar fashion. Their genotype probabilities are calculated alongside the genotyped individuals, but those are only used internally; their imputed genotypes are not included in the output.

3.3 Mating nodes

The pedigree data is stored in two vectors, one with derived type 'individual' and one with type 'matingnode'. This allows fast lookup of all parents, mates, and offspring of an individual.

Each matingnode element consists of a numeric index m, the indices of the two parents d and s (either or both of which may be 0), and a pointer array with the individual indices of the offspring. The latter has a length of at least 1. Each individual element contains a pointer array with the matingnodes with its offspring, which may be of length 0 (no offspring).

3.4 Initiation

To avoid use of if/else statements when looking up the anterior or posterior probability of a genotyped vs non-genotyped vs unknown parent, the arrays storing these values have lower and upper bounds of 0 and N_T respectively. In the pedigree, unknown parents are indicated with index 0, all genotyped individuals with their row number in the genotype file ($1 : N_G$), and any non-genotyped individuals with indices $(N_G + 1) : N_T$.

For each SNP, the data arrays are initiated by

- copying the genotype data at SNP l for all N_G individuals into vector $\boldsymbol{Y}$; the values at index 0 and indices $> N_G$ remain 'missing' (-1)
- creating array $\boldsymbol{A}$ with anterior probabilities with bounds (0:2, 0:N_T), and initialise at the genotype probabilities at HWE.

- creating array $\boldsymbol{P}$ with posterior probabilities with bounds (0:2,2,0:N_M), where N_M is the total number of mating nodes. $\boldsymbol{P}_{.,1,m}$ stores $p_{ds}(u_d)$, and $\boldsymbol{P}_{.,2,m}$ stores $p_{ds}(u_s)$. It is initialised at 1/3, i.e. uniform probabilities.

- creating array $\boldsymbol{H}$ with the genotype probabilites $Pr(u_i|\boldsymbol{y})$, with bounds (0:2, 0:N_T). Its initial values are calculated during the first loop of the iterative peeling algorithm.

3.5 Iterative peeling

This consists of three steps, which are repeated until all genotype probabilities in $\boldsymbol{H}$ differ by less than `tol` from one iteration to the next. These steps are:

- calculate a_i for each individual, after sorting individuals by generation number starting from the top of the pedigree

- calculate p_m for each mating, after sorting matings by generation number starting from the bottom

- calculate h_i for each individual using equation 1.

Calculating anterior probabilities from top to bottom, and posterior probabilities from bottom to top, ensures faster convergence.

3.6 Calculation of anterior & posterior probabilities

There is large overlap between how the anterior and posterior probabilities are calculated (Figures 1 and 2, Equations 2 and 3). The difference is the focal mating node: for the anterior probability it is the node with the parents and full siblings of focal individual i, while for the posterior probability it is the node with (one of) its mate(s) and their offspring.

Accordingly, two functions where created that are called during both the calculation of $a(u_i)$ and $p_{ij}(u_i)$:

- *f_mates*: a function to sum the posterior probabilities across all mates, optionally excluding one (the mate in the current focal node) ;

- *f_offspring*: a function to sum the penetrance probabilities ($g(y|u)$) and output of F_mates across all offspring of a node, optionally excluding one (the current focal individual)

The remaining terms in the equations are the anterior and penetrance probabilities of 1 or 2 other individuals (j for $p_{ij}(u_i)$; d and s for $a(u_i)$). These are stored in arrays, as described in the earlier subsection 'Initiation'.

3.7 Data cleaning

The threshold separating 'probably wrong' from 'probably correct' is calculated as

$$T(\text{wrong}|y) = 1 - \frac{p(y|u=y)}{\sum_{z=0}^{2} p(y|u=z)} \tag{4}$$

If the genotype probability $Pr(u_i|\mathbf{y}) < T(\text{wrong}|y)$, then the genotype is set to missing. Since $Pr(u_i|\mathbf{y})$ includes the term $g(y_i|u_i)$, this implies that the evidence from relatives regarding u_i is stronger than the evidence provided by y_i (e.g. observed as heterozyogte, but both parents and all offspring are identical homozygotes).

After the genotype data has been cleaned, the iterative peeling algorithm is run again. Probabilities are not reset, and therefore this second peeling step is considerably faster than the first.

During the subsequent imputation step, this genotype is treated as any other missing genotype.

3.8 Imputation

Once the genotype probabilities are calculated, imputation is largely straightforward: set any missing genotype equal to the most-likely u_i. When a threshold `T_impute` is specified, any missing genotypes for which the maximum h is below `T_impute` will be left missing.

When two genotypes are (nearly) equally likely, e.g. when the individual is the result of a homozygote X heterozygote cross and has no offspring, by default the genotype is set to heterozygote. This can be changed with option `--when-in-doubt`.

References

RJ Kerr and BP Kinghorn. An efficient algorithm for segregation analysis in large populations. *Journal of Animal breeding and Genetics*, 113(1-6):457–469, 1996.