An R Package for Multigene Descent Probabilities

Charles J. Geyer

July 15, 2023

1 Introduction

Here we design an R package that incorporates some of the functionality of a very old (and defunct) S package for statistical genetics described by Geyer (1988). In particular, our new package will do multigene descent probabilities as described originally by Thompson (1983) and as implemented by Geyer (1988).

2 Pedigrees

We work relative to a defined pedigree in which every individual has either two parents or none specified. Those with none specified are called *founders*. A pedigree may be specified by a *triplets matrix* having three columns and each row gives the names of a non-founder individual, its father, and its mother, in that order. We check that no individual is its own ancestor. Optionally, we check that sexes are consistent (no individual is both father and mother). This check is optional so that we can deal with hermaphroditic organisms.

Any ancestors of individuals not in the pedigree — including parents of founders — are assumed to not be individuals in the pedigree. That is, we are assuming that all unknown individuals are not any known individuals.

3 Descent Probabilities

Thompson (1983) defines *multigene descent probabilities* $g_S(B_1, \ldots, B_n)$ to be the probability that genes at one autosomal locus randomly chosen from each of the individuals B_1, …, B_n are all descended from genes (not necessarily the same gene) in some set S of genes in individuals in the pedigree. The individuals B_1, …, B_n need not be distinct. The set S can be specified by giving for each individual in the pedigree an integer 0, 1, or 2 that says how many of its genes (at the autosomal locus in question) are in the set S. (We are assuming a diploid organism.)

Since the order of B_1, …, B_n does not matter to the definition, we may assume these arguments are in sorted order in some order that always has offspring before parents. There always is such an order because no individual can be its own ancestor, and such an order can be found by the topological sort algorithm (Aho, et al., 1983, Section 6.6) which is implemented by R function `tsort` in R package `pooh` (Geyer, 2017). Thus in what follows we have $B_1 = \cdots = B_r$ for some $r \geq 1$, and we have B_1 not equal to nor an ancestor of B_i for $i > r$. We also adopt the convention

$$g_S() = 1, \tag{1a}$$

which makes sense because the empty set of genes chosen from the empty set of individuals is always contained in genes descended from S (because the empty set is contained in any set).

Theorem 1. *Assume arguments are in an order that has offspring before parents. If B_1 is not a founder, contains no genes of S, and occurs only once in the arguments ($n = 1$ or $B_1 \neq B_2$), then*

$$g_S(B_1, \ldots, B_n) = \tfrac{1}{2} g_S(M_1, B_2, \ldots, B_n) + \tfrac{1}{2} g_S(F_1, B_2, \ldots, B_n) \tag{1b}$$

where F_1 is the father of B_1 and M_1 is the mother of B_1.

If B_1 is not a founder, contains no genes of S, and occurs r times in the arguments ($B_1 = \cdots = B_r$ and $n = r$ or $B_r \neq B_{r+1}$), then

$$\begin{aligned} g_S(B_1, \ldots, B_n) = {} & (\tfrac{1}{2})^{r-1} g_S(B_1, B_{r+1}, \ldots, B_n) \\ & + \left[1 - (\tfrac{1}{2})^{r-1}\right] g_S(M_1, F_1, B_{r+1}, \ldots, B_n) \end{aligned} \tag{1c}$$

where F_1 and M_1 are as before.

If B_1 is a founder and contains no genes of S, then

$$g_S(B_1, \ldots, B_n) = 0. \tag{1d}$$

If B_1 contains two genes of S and occurs r times in the arguments ($B_1 = \cdots = B_r$ and $n = r$ or $B_r \neq B_{r+1}$), then

$$g_S(B_1, \ldots, B_n) = g_S(B_{r+1}, \ldots, B_n). \tag{1e}$$

If B_1 is not a founder, contains one gene of S, and occurs r times in the arguments ($B_1 = \cdots = B_r$ and $n = r$ or $B_r \neq B_{r+1}$), then

$$\begin{aligned} g_S(B_1, \ldots, B_n) &= (\tfrac{1}{2})^r g_S(B_{r+1}, \ldots, B_n) \\ &\quad + \tfrac{1}{2}\left[1 - (\tfrac{1}{2})^r\right]\left[g_S(F_1, B_{r+1}, \ldots, B_n) + g_S(M_1, B_{r+1}, \ldots, B_n)\right]. \end{aligned} \tag{1f}$$

If B_1 is a founder, contains one gene of S, and occurs r times in the arguments ($B_1 = \cdots = B_r$ and $n = r$ or $B_r \neq B_{r+1}$), then

$$g_S(B_1, \ldots, B_n) = (\tfrac{1}{2})^r g_S(B_{r+1}, \ldots, B_n). \tag{1g}$$

Proof. If B_1 is not a founder and S contains no genes of B_1 and $B_1 \neq B_2$, then a gene chosen at random from B_1 is equally to have come from F_1 or M_1 and is equally likely to be either of the genes in F_1 or M_1 by Mendel's laws. Since B_1 has no genes of S, the only way it can have genes descended from S is if they come through F_1 or M_1.

If B_1 is not a founder and S contains no genes of B_1 and $B_1 = \cdots = B_r$ and $B_1 \neq B_{r+1}$ or $r = n$, then with probability $(\frac{1}{2})^{r-1}$ the same gene is chosen from B_1, …, B_r, in which case the probability that this gene and randomly chosen genes from B_{r+1}, …, B_n are descended from S is $g_S(B_1, B_{r+1}, \ldots, B_n)$. Otherwise, two different genes are chosen from B_1, …, B_r, in which case one must be a randomly chosen gene from F_1 and the other must be a randomly chosen gene from M_1 and the probability that these genes and randomly chosen genes from B_{r+1}, …, B_n are descended from S is $g_S(M_1, F_1, B_{r+1}, \ldots, B_n)$.

If B_1 is a founder and S contains no genes of B_1, then from the assumption that none of the ancestors of B_1 — all of whom are unknown — are any of the known individuals in the pedigree who collectively contain the genes in S it follows that B_1 cannot contain any genes descended from S.

If S contains two genes of B_1 and $B_1 = \cdots = B_r$ and $B_1 \neq B_{r+1}$ or $r = n$, then all genes chosen from B_1, …, B_r must come from S because they are chosen from these two genes of B_1 contained in S. Hence (1e) holds.

If B_1 is not a founder and S contains one gene of B_1 and $B_1 = \cdots = B_r$ and $B_1 \neq B_{r+1}$ or $r = n$, then with probability $(\frac{1}{2})^r$ the genes randomly chosen from B_1, …, B_r are all the one gene of B_1 contained in S, in which case the probability that the genes of B_{r+1}, …, B_n are descended from S is $g_S(B_{r+1}, \ldots, B_n)$. Otherwise, some of the genes chosen from B_1, …, B_r are the gene of B_1 not contained in S, in which case this gene is equally likely to have come from F_1 or M_1, in which case the probability that this gene and the genes of B_{r+1}, …, B_n are descended from S is $g_S(F_1, B_{r+1}, \ldots, B_n)$ or $g_S(M_1, B_{r+1}, \ldots, B_n)$.

If B_1 is a founder and S contains one gene of B_1 and $B_1 = \cdots = B_r$ and $B_1 \neq B_{r+1}$ or $r = n$, then with probability $(\frac{1}{2})^r$ the genes randomly chosen from $B_1, \ldots, B_r$ are all the one gene of B_1 contained in S, in which case the probability that the genes of $B_{r+1}, \ldots, B_n$ are descended from S is $g_S(B_{r+1}, \ldots, B_n)$. Otherwise, some of the genes chosen from $B_1, \ldots, B_r$ are the gene of B_1 not contained in S, in which case this gene came from some ancestor of B_1 — and all these ancestors are unknown, hence none are any of the known individuals in the pedigree who collectively contain the genes in S — it follows that the gene of B_1 not contained in S is not descended from S. □

Our equations (1b) through (1g) are unnumbered displayed equations on pp. 33–34 in Geyer (1988), except that two typographical errors have been corrected, one minor (a missing parenthesis) and one major: what is $1 - (\frac{1}{2})^{r-1}$ in our (1c) is $2^{r-1} - 1$ in Geyer (1988) and in Thompson (1983, equation (7)). This error, if reproduced in the code for the old S package `sped`, could have produced probabilities greater than one in case $B_1 = B_2 = B_3$. It looks like this error was *not* reproduced in the code (see `http://users.stat.umn.edu/~geyer/sped/src/gnx.c`) so that code did not have this particular error. Our equation (1a) is also in Geyer (1988) run into the text on p. 34.

Our (1d) through (1g) do not appear in Thompson (1983) who says only that our (1b) and (1c) are are to be used with boundary conditions that specify when individuals B_i have genes in S. Presumably, our (1a) and (1d) through (1g) are the boundary conditions Thompson referred to, because Geyer was Thompson's research assistant when Geyer (1988) was written.

4 Special Descent Probabilities

These come originally from Thompson (1986). We follow Geyer (1988).

4.1 Gammas

The fraction of genes in individual B that comes from founder A is

$$\gamma(A, B) = g_{S_A}(B) \tag{2}$$

where S_A is the set of genes that contains the two genes of A and no other genes.

4.2 Betas

If, as before, F_i is the father of B_i and M_i is the mother of B_i, then

$$\beta(A, B_i) = g_{S_A}(F_i, M_i) \tag{3}$$

is the bilineal contribution of founder A to individual B_i, the probability that both genes of B_i are descended from genes of founder A.

4.3 Alphas

Now let T_A be the set of genes that contains one gene of founder A and no other genes, and let F_i and M_i be as above, then

$$\alpha(A, B_i) = 2g_{T_A}(F_i, M_i) \tag{4}$$

is the inbreeding of individual B_i relative to founder A, the probability that both genes of individual B come from the same gene in founder A.

4.4 Inbreeding Coefficients

Then

$$\alpha(B) = \sum_{A \in \text{Founders}} \alpha(A, B) \tag{5}$$

4.5 Kinship Coefficients

The kinship coefficient of individuals B_i and B_j is

$$\phi(B_i, B_j) = 2 \sum_{A \in \text{Founders}} g_{T_A}(B_i, B_j) \tag{6}$$

This is not an efficient way to calculate kinship coefficients, also called coancestry coefficients (Lynch and Walsh, 1998, pp.135 and 763). They give the following recursive equations and boundary conditions. If B_i is not a founder, then

$$\phi(B_i, B_i) = \frac{1 + \phi(F_i, M_i)}{2} \tag{7a}$$

and if B_i is not a founder and not an ancestor of B_j

$$\phi(B_i, B_j) = \frac{\phi(F_i, B_j) + \phi(M_i, B_j)}{2} \tag{7b}$$

and if B_i is a founder

$$\phi(B_i, B_i) = 1. \tag{7c}$$

5 A Problem with C Variable-Length Arrays

The C code underlying the R functions in this package uses C variable-length arrays in order to not have to do complicated and tricky cleanup of allocated memory in case of user interrupt and also to have memory allocated in a recursive function call to be freed as soon as the function returns (and the stack is popped).

The problem is that sometimes these C functions use C variable-length arrays of zero length. And the Clang undefined behavior sanitizer used by CRAN says that something about this is undefined behavior.

Hence we need to debug this. We got kicked of CRAN about this issue.

Apparently (stackoverflow post found by googling) zero-length variable-length arrays are illegal. Hence we need to get rid of them.

In Theorem 1 above, cases (1e), (1f), and (1g) all have recursive function calls involving $g_S(B_{r+1}, \ldots, B_n)$, which means the same as $g_S()$ in case $r = n$. And the current code tries to create a zero-length variable-length array to hold the arguments of this function call.

We somehow need to avoid this allocation and hence might as well avoid the function call because $g_S() = 1$ always by (1a).

Actually the fixed turned out to be easy. In cases (1e) and (1g) we just return the correct value (if $r = n$) before the allocation of the zero-length variable-length would happen. In case (1f) the allocated variable-length array has length $n - r + 1 > 0$ to accomodate the other two recursive function calls.

References

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1983). *Data Structures and Algorithms.* Addison-Wesley, Reading, MA.

Geyer, C. J. (1988). Software for Calculating Gene Survival and Multigene Descent Probabilities and for Pedigree Manipulation and Drawing. Technical Report No. 153, Department of Statistics, University of Washington. `https://www.stat.washington.edu/article/tech-report/software-calculating-gene-survival-and-multigene-descent-probabilities-and`

Geyer, C. J. (2017). R package `pooh` (Partial Orders and Relations), version 0.3-2. `http://cran.r-project.org/package=pooh`.

Lynch, M., and Walsh, B. (1998). *Genetics and Analysis of Quantitative Traits.* Sinauer Associates, Sunderland, MA.

Thompson, E. A. (1983). Gene extinction and allelic origins in complex genealogies (with discussion). *Proceedings of the Royal Society of London. Series B, Biological Sciences*, **219**, 241–251. https://doi.org/10.1098/rspb.1983.0072.

Thompson, E. A. (1986). Ancestry of alleles and extinction of genes in populations with defined pedigrees. *Zoo Biology*, **5**, 161–170. https://doi.org/10.1002/zoo.1430050210.