Received: 25 June 2021 | Revised: 9 September 2021 | Accepted: 15 September 2021

DOI: 10.1111/1755-0998.13513

RESOURCE ARTICLE

MOLECULAR ECOLOGY RESOURCES WILEY

CKMRpop: Forward-in-time simulation and tabulation of pairwise kin relationships in age-structured populations

Eric C. Anderson

Fisheries Ecology Division, Southwest Fisheries Science Center, National Marine Fisheries Service, NOAA, Santa Cruz, CA, USA

Correspondence
Eric C. Anderson, Fisheries Ecology Division, Southwest Fisheries Science Center, 110 McAllister Road, Santa Cruz, CA 95060, USA.
Email: eric.anderson@noaa.gov

Abstract

In the last five years, interest in close-kin mark–recapture (CKMR), a variant of mark–recapture that uses genetically inferred kin as 'recaptures', has grown dramatically. However, understanding the basis of CKMR, and properly implementing it, remains challenging. This paper describes an R package, CKMRpop, for simulating age-structured populations with user-specified demography, overdispersed variance in reproductive success (allowing for different ratios of effective to census size) and random sampling of individuals. Using compiled code for the simulation makes it feasible to simulate populations of millions of individuals. From the simulation output, pairs of sampled individuals related within a user-specified number of generations are found. Such pairs form the foundation for CKMR inference, and simulating them provides insight for understanding the statistical basis for CKMR and for assessing the feasibility of CKMR in different scenarios. We predict that CKMRpop will serve as an important tool for researchers contemplating CKMR estimation of population size. Furthermore, the methods presented here for identifying and categorizing relationships beyond half-siblings allow a more complete picture of the wide variety of kin pairs encountered in populations. This identifies the fraction of kin pairs that may not be the target of a CKMR experiment, but may be inadvertently mistaken for a more closely related 'target' kin pair. Additionally, as more distant kin categories will likely be accurately inferred from increasingly available and inexpensive whole genome resequencing, understanding the distributions of more distant relationships in populations is a first step towards broadening the scope of CKMR to include them.

KEYWORDS
abundance estimation, CKMR, close-kin mark–recapture, pairwise relationships

1 | INTRODUCTION

Skaug (2001) introduced into the statistical literature the idea that the pairwise relationships, inferred from genetic data, between the members of a sample could be used in a mark–recapture framework to estimate the size of a natural population. While genetic markers obtained non-invasively (e.g. from hair and faeces) have been used to sample and 'recapture' the *same* individuals using a traditional mark–recapture approach (Miller et al., 2005), Skaug (2001) proposed using *related* individuals (*i.e.* kin) as a type of recapture and illustrated the use of such an approach with microsatellite data applied to a population of northern minke whales. This approach has subsequently been termed close-kin mark–recapture (CKMR).

Since Skaug (2001), the last two decades have seen a dramatic increase in the availability of genetic data and a reduction in cost, making CKMR more feasible. Additionally, the underpinnings of CKMR have been further developed, with Bravington, Skaug, et al. (2016) providing a comprehensive description of kin-probability calculations in the presence of covariates and a statistical treatment of sampling design considerations for CKMR by quantifying the Fisher information from the pseudo-likelihood. Finally, CKMR has proven tremendously successful in providing fishery-independent estimates of abundance in large, valuable fisheries (Bravington et al., 2016), as well as in smaller populations of conservation concern (Hillary et al., 2018).

Accordingly, there is considerable interest in applying CKMR, especially in fisheries applications; however, implementing CKMR remains difficult. In addition to the massive sampling and genotyping efforts typically required, the statistical modelling and computation necessary for developing the CKMR pseudo-likelihood in long-lived species is non-trivial. One of the first hurdles, in this regard, can be simply understanding and predicting how many kin pairs of different types might be expected given a certain degree of sampling effort in a population of a given size. This prediction is especially challenging in age-structured populations with long-term sampling programmes carried out over multiple years, or in situations where the sampling is spatially organized in species that are not panmictic, despite very high levels of gene flow, as is often found in the marine environment (Waples, 1998).

An additional obstacle to applying CKMR today is that most of the theory for CKMR has been worked out only for rather close relationships such as parent–offspring and/or half-sibling relationships. Published accounts of CKMR—either theoretical or practical—provide considerably less insight into the expected occurrence of other, more distant, kin pairs, such as cousins. Knowing the expected frequency of occurrence of these other types of kin pairs is important for two separate reasons. First, using genetic data, it can be difficult to distinguish these more distant kin pairs (like half-aunt–niece or half-first–cousin pairs) from the target kin pairs (like half-siblings) that one may wish to use for CKMR. Thus, in designing a CKMR experiment, it is important to know how many non-target kin pairs of different types are expected, because this will affect the reliability of CKMR 'recaptures' of target kin categories—if many first cousins or half-aunt–niece pairs are expected in a CKMR study using half-siblings, then sufficient genetic resources are needed to reliably distinguish a large fraction of these more distant kin categories from the half-siblings.

The second reason that more distant kin pairs are pertinent to CKMR studies today is that, at some point in the near future, it may be possible to reliably distinguish these more distant kin pairs. As the cost of DNA sequencing continues to decline (Fuentes-Pardo & Ruzzante, 2017), it becomes reasonable to entertain the possibility that low-coverage whole genome resequencing in non-model species with well-assembled genomes will provide sufficient information to reliably identify pairwise relationships an additional generation or two more distant than has traditionally been pursued in molecular ecology (Hanghøj et al., 2019) (*i.e.* identifying half-first cousins or great uncles, rather than merely parent–offspring and half-sibling pairs). If such relationships can be reliably identified, then they can be used as additional information in a CKMR framework, but only if we can predict their frequency of occurrence given a population's size and other parameters.

There is a need for two developments that could help with the continued application of CKMR approaches. First, potential users of CKMR would benefit greatly from having a simple, easily understood way to simulate the populations that they study and to carry out, *in silico*, a variety of proposed sampling schemes in these populations. Second, from the results of such simulations, it would be beneficial to be able to categorize and count all the related pairs found in the samples out to an arbitrary degree of relationship. Here, we describe the R package CKMRpop that provides users with a way to accomplish both of these tasks.

In brief, the package CKMRpop gives a simple R interface for running the compiled C program SPIP (Anderson & Dunham, 2005) that provides for fast, forward-in-time simulation of age-structured populations, allowing the parameterization of a number of demographic features such as variability in reproductive success, degree of monogamy/polygamy and migration between demes, as well as the specification of a wide range of sampling schemes in the population(s). Although SPIP has the capacity to simulate unlinked genetic markers in the course of the simulation, the objects of interest within CKMRpop are the simulated population census sizes, the samples collected from the population during the simulation, and, most importantly, the pedigree of the population that is output by SPIP. Users more familiar with other forward-in-time simulation packages, for example the widely used and full-featured software SLiM (Haller & Messer, 2019), can configure the output of those programs (rather than of SPIP) to be used downstream by CKMRpop.

CKMRpop implements functions for extracting textual output from SPIP (or other simulation programs) and organizing it into native R data frames, providing easy access to information about the time of birth and death of all individuals in multiple inter-connected populations, the age- and sex-specific annual census size of the populations, the identities and ages of individuals sampled from the populations, and the complete pedigree of all individuals in the populations over the time of the simulation. Functions are available that produce a variety of summaries and plots to verify that the distributions of rates of birth, death and reproduction are what the user expects. And finally, CKMRpop provides a fast method, implemented in C, using RCpp (Eddelbuettel & François, 2011), to recursively traverse the simulated pedigree and identify all the pairwise relationships, out to a user-specified number of generations, between the sampled individuals from the simulation. The following section provides details about these aspects of the package.

2 | METHODS

2.1 | Interfacing with the program SPIP

The program SPIP was written in C and is intended for use on the Unix/Linux or Windows command line, writing all of its results in text format to the standard output stream. Once CKMRpop has

been installed using R on a user's computer, a function in the package allows for the straightforward download and installation of the OS-specific compiled command-line version of SPIP from GitHub. CKMRpop provides an R wrapper to SPIP, passing input to SPIP, catching error messages, then processing SPIP's text output and passing data frames back to R.

On the command line, all options available for parameterizing the SPIP simulation can be given as long option names preceded by two hyphens, for example, '--max-age 20' on the SPIP command line sets the maximum age of the species being simulated to 20 years. In CKMRpop, these options are specified as named members of a list of options, where the names are the SPIP option names with the preceding dashes removed. For example, if the options to SPIP were to be stored in an R list pars, then setting ' PARS$MAX-AGE <- 20' would store that maximum age option to be passed to SPIP.

Details of SPIP's simulation approach are provided in Anderson and Dunham (2005). Since the original publication on SPIP, the program has been expanded to allow for simulation of multiple demes, each with their own vital rates and with arbitrary, time-dependent rates of migration between them. CKMRpop provides the functions `spip_help()` and `spip_help_full()` to print SPIP's manual files in abbreviated and long format to the R console window. Population size regulation in SPIP is managed by the user specifying the number of incoming individuals to be born in each new cohort. CKMRpop provides functions for determining these numbers and the resulting age structure in scenarios of constant population size, given the age-specific survival rates in the population. The user specifies the number of individuals at the outset of the simulation; these are regarded as unrelated founders of the simulated pedigree, and the simulation must be run forward a sufficient number of generations to simulate the desired, possible pairwise relationships between individuals. Fine-grain control of sampling of the populations within SPIP is available by specifying the fraction of males and females of each age that should be randomly sampled each year. The sampling can be either lethal or non-lethal.

CKMRpop runs SPIP and returns data frames that hold the pedigree of the population, the census sizes before and after the episode of death each year, the times of death of all individuals that died during the simulation, an account of every event of individual migration between the populations, and the IDs, sampling times and populations of origin and sampling of all individuals that were taken as samples from the populations. With these data frames available in R, the package CKMRpop includes a variety of functions for summarizing and visualizing the output. The first package vignette, viewable at https://eriqande.github.io/CKMRpop/articles/species_1_simulation.html, provides a comprehensive description of these functions, and they are also fully documented in the package manual. The most important functions are those that use the pedigree information and the IDs of the samples to identify pairs of different relationship categories. The approach to doing so is described in the next section.

2.2 | Defining relationships, ancestry vectors, etc.

In any natural or randomly simulated population of dioecious members, the number of possible pairwise relationships between two individuals can be quite large, especially when relationships involving more than just one generation of separation are involved. If inbreeding is a possibility, this further increases the number of ways that two individuals might be related. Sometimes, we might be interested specifically in relationships that involve a particular sex of an ancestor, while in some applications that might not matter. In fact, in many cases we can expect that we will want to enumerate relationships into summarized categories. To handle this diversity of goals, we first develop a general approach that is capable of classifying every type of distinct pairwise relationship that could be encountered. We subsequently introduce some approaches for aggregating the output of such a fine-grained classification of relationships into coarser relationship categories.

Two individuals—let us call them *x* and *y*—in a population are related because they share the same individuals amongst their ancestors, or if one is an ancestor of the other. In order to treat these two cases equivalently, we define the 'ancestors' of an individual to include the individual itself, in addition to its formal ancestors. These shared individuals amongst the ancestors of *x* and *y*, need not, of course, be the same ancestors for both *x* and *y*. For example, individual *z* might be *x*'s mother, while *z* could be *y*'s maternal grandmother—a situation that would make *x* at least a half-aunt of *y*. We will consider pairwise relationships that involve shared ancestors going back no more than *n* generations from either of the two members of the pair. This is made simple by recording the identities of the ancestors of each individual, starting from the individual itself, in an *ancestry vector*. If $n = 0$, then we consider only the individual itself, and the ancestry vector is of length 1. If $n = 1$, then we consider the identities of the individual and its two parents: first the father and then the mother. For example, the ancestry vector for individual *x* when $n = 1$ is of length 3 and has the ordered elements: 1) individual *x*, 2) the ID of *x*'s father and 3) the ID of *x*'s mother. If $n = 2$, ancestry out to the grandparents is recorded, giving a vector with ordered elements: 1) individual *x*, 2) the ID of *x*'s father, 3) the ID of *x*'s mother, 4) the ID of *x*'s father's father, 5) the ID of *x*'s father's mother, 6) the ID of *x*'s mother's father and 7) the ID of *x*'s mother's mother. Such a scheme can be carried out to any number of generations, following a simple numbering/ordering convention as shown in Figure 1. In general, the length of the ancestry vector is $L = 2^{n+1} - 1$.

We let $v_x = (v_{x,1}, \ldots, v_{x,L})$ denote the ancestry vector of individual *x*, and we define v_y similarly for individual *y*. Recall that the v_x's and v_y's are the IDs of the individuals in the ancestries of *x* and *y*, respectively. Now, we define **M** as the *ancestry match matrix* between *x* and *y*. **M** is a matrix of 0's (or FALSE's) and 1's (or TRUE's) with *L* rows and *L* columns, such that $m_{r,c}$, the element in the r^{th} row and c^{th} column of **M**, is equal to 1 (TRUE) if $v_{x,r} = v_{y,c}$, and 0 (FALSE) otherwise. Every possible pairwise relationship between two individuals is captured by the value of **M**, and so, it provides a way to enumerate,

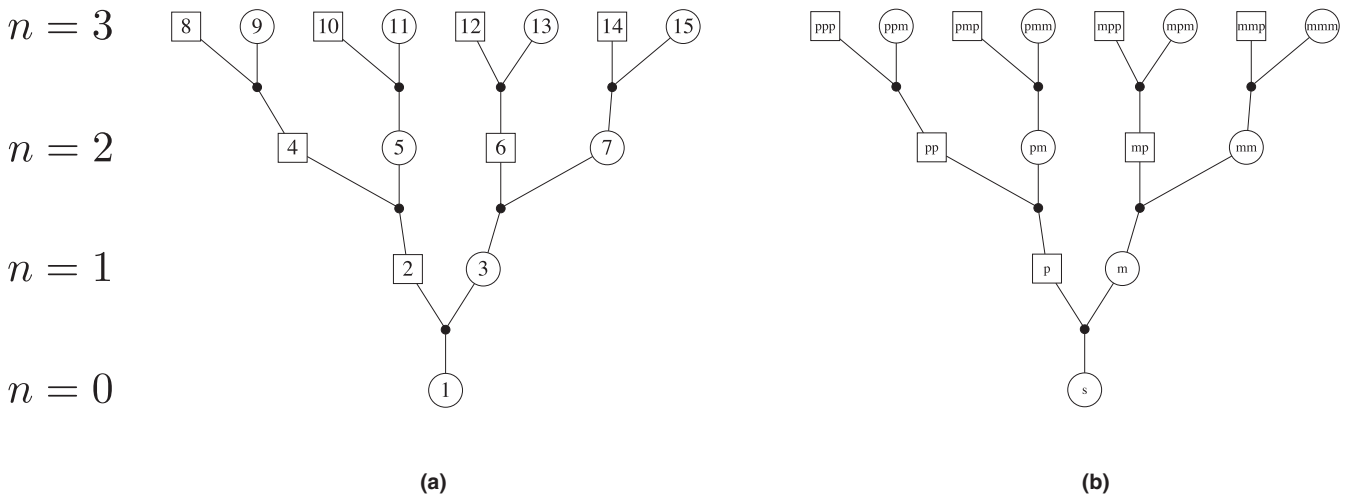


FIGURE 1 Ordering/numbering/naming scheme for the identity of ancestors in an ancestry vector, shown going back for up to $n = 3$ generations. (*a*) The order of an individual's ancestors in an ancestry vector. 1 is the individual itself, 2 is its father, 3 its mother and so forth. (*b*) The abbreviated names given to the members of the ancestry vector. These appear in the *ancestry match matrices* shown in Figure 1. The abbreviations are as follows: s = 'self'; *p*= 'pa' (father); *m*= 'ma' (mother); *pp*= 'pa's pa' (father's father), and so forth, such that, for example, *mmp* = 'ma's ma's pa' (the father of the individual's mother's mother), *etc*. Note that this is not a pedigree as such—it is diagrammatic device used to denote the numbering of identities in an ancestry vector. While it appears that the individual 1 (in a) or *s* (in b) is not inbred, actual inbreeding of the focal individual can be represented by the repeated occurrence of any ancestor's ID within the ancestry vector

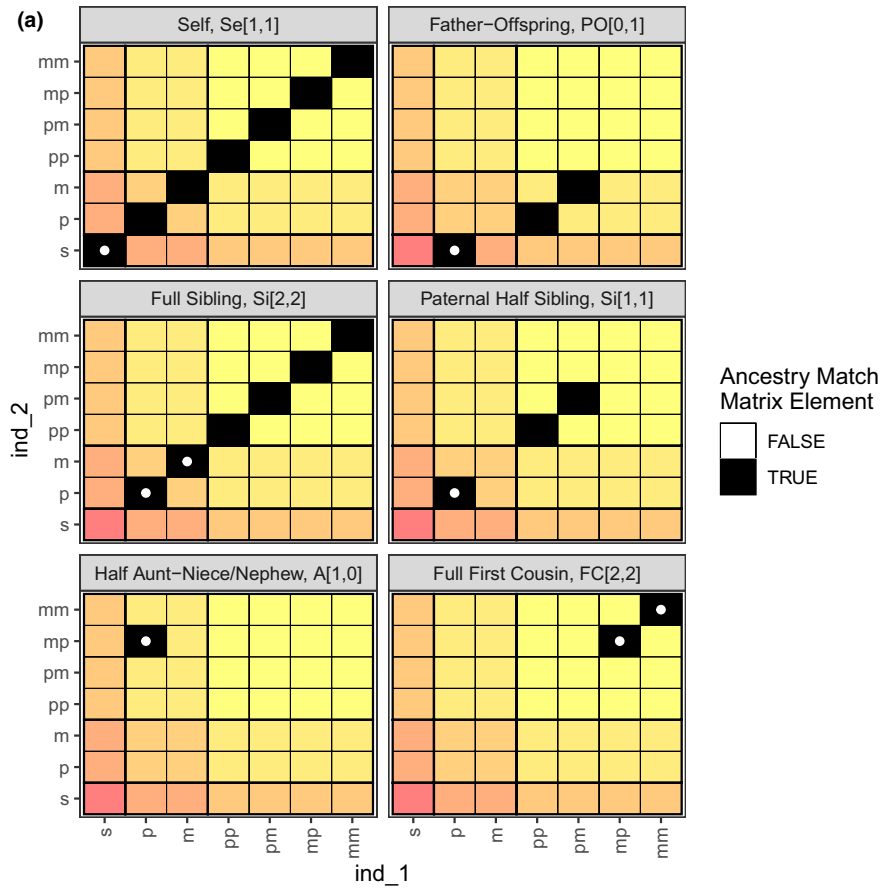
on a fine-grained scale, all the distinct relationships between any pair of individuals. It is worth noting that, in species that remain of the same sex throughout their lives, many values of **M** are impossible (*i.e*. a male ancestor of *x* could not be a female ancestor of *y*); however, we do not impose such a constraint on the form of **M**, as doing so would preclude its use in modelling hermaphroditic, protandrous or protogynous species.

For illustration, Figure 2a plots the ancestry match matrices for $n = 2$ of six different relationships between two individuals (ind_1 and ind_2). Shown are the relationships of 'self', 'father–offspring', 'full-sibling', 'paternal–half-sibling', 'half-aunt–niece' and 'full-cousin'.

It should be clear that the pattern of black squares in the ancestry match matrix uniquely defines any possible relationship. It is also evident from the different coloured shading in the plots that there are different regions or 'zones' in the ancestry match matrix in which a shared ancestor gives rise to a different type of relationship. Finally, note that some of the blackened cells correspond to the shared ancestors of shared ancestors. This is particularly evident in the 'self' relationship, where, since the ID of the individual itself is shared, so too are the IDs of all of its ancestors. In such cases, the directly relevant shared ancestor is the most recent of the string of shared ancestors, with the remainder simply being the ancestors of that first shared ancestor. We refer to such an individual as a *primary* shared ancestor, with the ancestors of the primary ancestor designated as *secondary* shared ancestors. In Figure 2a, the cells corresponding to primary shared ancestors are highlighted with a white dot in the middle of them. Note that a pair of individuals can have more than one primary shared ancestor. It turns out to be straightforward to identify the secondary (and hence, also, the primary) shared ancestors in an ancestry match matrix: any cell with $m_{r,c} = 1$ in **M** denotes a secondary shared ancestor if $m_{\lfloor \frac{r}{2} \rfloor, \lfloor \frac{c}{2} \rfloor} = 1$, where $\lfloor w \rfloor$ denotes the largest integer less than *w*.

Figure 2b shows the different relationship zones out to $n = 4$ generations. A primary shared ancestor in each of these different zones gives rise to a certain class of relationship, as listed in Table 1. For example, primary shared ancestors in the Si zone give rise to siblings: if a pair shares one primary shared ancestor and it is in the Si zone, then the related pair has a half-sibling relationship; the occurrence of two primary shared ancestors in the Si zone indicates that the members of the pair are full siblings.

The relationship zones shown in Table 1 are listed in a natural order that corresponds to the contribution that a single primary shared ancestor in each zone makes to the coefficient of kinship, ψ, of the related pair (*i.e*. the probability that a gene segregated from *x* and a gene segregated from *y* are identical by descent). This probability is a function of the number of meioses that occur when a gene copy is transmitted from the primary shared ancestor to either member of the pair, and that number can be easily read from the position of the relationship zone in the ancestry match matrix. Let m_1 and m_2 be the number of meioses between the primary shared ancestor and the first (*x*) and second (*y*) individuals of the pair, respectively. In Figure 2b, the first individual is denoted ind_1 and the second individual is denoted ind_2. Visually, in Figure 2b, the values of m_1 and m_2 for any primary shared ancestor can be found by counting the number of different relationship zones that must be entered in order to reach the zone on the left or bottom edge of the figure when travelling left from the primary shared ancestor (for m_1) or down from the primary shared ancestor (for m_2). For example, a primary shared ancestor in the GGP zone above the diagonal gives $m_1 = 0$, $m_2 = 3$, whereas a primary shared ancestor in the FCr1 zone below



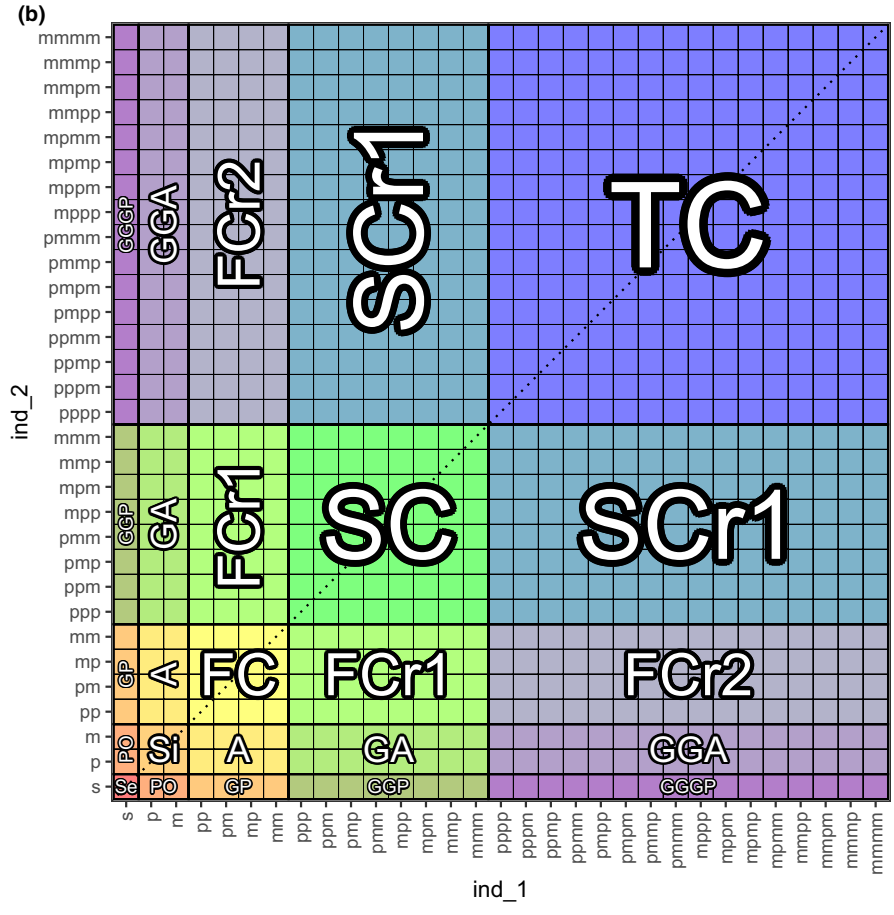


FIGURE 2 The Ancestry Match Matrix. (a) Ancestry match matrices to $n = 2$ generations of six example relationships indicated above each panel. Notation after the first comma in each name is the *dominant* relationship notation (see text). The blackened cells represent shared individuals in the ancestry vectors of the two members (ind_1 and ind_2) of the pair. The blackened cells with a white dot in the centre show the primary shared ancestors. The *x*- and *y*-axis values denote specific elements of the ancestry vectors of ind_1 and ind_2, as described in Figure 1. The different colours in the background of each cell show different relationship 'zones'. (b) Colour-coded relationship zones and their labels for the ancestry match matrix to $n = 4$ generations. The labels and meanings of each zone are defined in Table 1. The zones that straddle the diagonal: Se, Si, FC, SC and TC, are zones in which shared ancestors give rise to symmetrical relationships—those in which the relationship that a has to b is the same as the relationship that b has to a. All other zones are associated with non-symmetrical relationships

the diagonal gives $m_1 = 3$, $m_2 = 2$. The probability that a non-inbred x and y each segregates a gene that is identical by descent with the same gene inherited from a single primary shared ancestor in a given relationship zone is $\psi^{(1)} = 2^{-(m_1+m_2+1)}$ (because x segregates a copy of either of the two genes from the ancestor with probability 2^{-m_1}, and y segregates a copy of the same gene as x from the ancestor with probability $\frac{1}{2} \cdot 2^{-m_2}$). Thus, Table 1 is sorted in decreasing order of $\psi^{(1)}$. When two zones have the same $\psi^{(1)}$, they are placed in ascending order of max $\{m_1, m_2\}$, thus listing zones with the capacity for more primary shared ancestors first.

The properties of the ordering in Table 1 suggest a simple way of summarizing the relationship of a pair. We define the *dominant* relationship of a pair as the intersection of two pieces of information:

1. the first relationship zone (ordered as in Table 1) in which the pair has at least one shared primary ancestor,
2. the number of primary shared ancestors in that relationship zone.

In practice, for the non-symmetrical relationships (which are those that have relationship zones that do not straddle the dotted, diagonal line in Figure 2b), we record, in an ordered list of length two, the number of primary shared ancestors in the zone occurring above the diagonal and in the zone occurring below the diagonal, as this demarcates the orientation of non-symmetrical relationships (*e.g.* parent–offspring versus offspring–parent). This provides for a compact notation, examples of which appear in the facet headers of Figure 2a. For example, the father–offspring pair in the figure is notated PO[0,1], indicating that the PO zone above the dotted diagonal line includes 0 primary shared parents, while 1 primary shared ancestor occurs in the PO zone below the dotted diagonal line; therefore, the second individual of the pair (ind_2) is the parent while ind_1 is the child (because the shared ancestor is 'self' to ind_2 and is 'father' to ind_1). Similarly, the half-aunt-niece/nephew relationship is denoted by A[1,0] which tells us that the aunt/uncle in the relationship is ind_1. When notating the symmetrical relationships, the number of primary shared ancestors in the entire zone is merely written twice. For example, a half-sibling is written as Si[1,1] and a full first cousin is written as FC[2,2]. In every symmetrical relationship, the second number (after the comma) is redundant, since it is always the same as the first; however, this convention is convenient for the programming involved in categorizing relationships.

It must be stressed that the dominant relationship in a pair is not necessarily the only relationship between the members of the pair. Primary shared ancestors in later zones could provide additional relationships. For example, you could have paternal half-siblings that are also first cousins through the maternal lineage. In general, however, in large populations, and with moderate values of n, such *compound* relationships are encountered far less frequently than non-compound relationships (by the same reasoning that kin triads are encountered much less frequently than kin pairs in such circumstances, see, for example, section 4.3 in Bravington, Skaug, et al., 2016). Accordingly, it is worthwhile to summarize pairwise relationships according to what we call the *dominant* relationship, as it provides a lower bound on the kinship of the pair. CKMRpop provides this summary by dominant relationship type. It also returns detailed plots of all the distinct ancestry match matrices, found in a simulation, within each dominant relationship category, so that the occurrence of compound (and inbred) relationships within the simulation can be assessed.

TABLE 1 Relationship 'zones' within the ancestry match matrix to $n = 4$ generations, listed in decreasing order of $\psi^{(1)}$, the contribution that a single primary shared ancestor in the zone makes to the coefficient of kinship of the pair. The `Symm' column contains a Y for zones associated with symmetrical relationships and an N otherwise

Code	Symm	$\psi^{(1)}$	Description
Se	Y	$\frac{1}{2}$	Self or monozygous twin
PO	N	$\frac{1}{4}$	Parent/offspring
Si	Y	$\frac{1}{8}$	Sibling
GP	N	$\frac{1}{8}$	Grandparent/grandchild
A	N	$\frac{1}{16}$	Avuncular (aunt/niece, *etc.*)
GGP	N	$\frac{1}{16}$	Great-grandparent/gg-offspring
FC	Y	$\frac{1}{32}$	First cousin
GA	N	$\frac{1}{32}$	Great avuncular
GGGP	N	$\frac{1}{32}$	Great-great-grandparental
FCr1	N	$\frac{1}{64}$	First cousin, once removed
GGA	N	$\frac{1}{64}$	Great-great avuncular
SC	Y	$\frac{1}{128}$	Second cousin
FCr2	N	$\frac{1}{128}$	First cousin, twice removed
SCr1	N	$\frac{1}{256}$	Second cousin, once removed
TC	Y	$\frac{1}{512}$	Third cousin

2.3 | Identifying related, sampled pairs from the simulated pedigree

After the SPIP simulation has completed and its output has been imported into R, the necessary elements for identifying related pairs are 1) the pedigree: a data frame with three columns, kid, pa and ma, that give the IDs of the two parents of every child in the simulated population(s); and 2) a vector of the IDs of the sampled individuals (the individuals that were randomly sampled during the course of the simulation). In large simulated populations, the pedigree may consist of millions of rows, so an efficient algorithm for identifying related pairs amongst the much smaller number of sampled individuals is necessary. CKMRpop implements such a function in RCpp for speed. Briefly, first, the pedigree is stored in memory as a directed graph (a collection of vertices and directed edges), with array access to the sampled individuals. Then, starting from each sampled individual, a recursive function, `search_up()`, implements a depth-first search for other sampled individuals that extends up

the pedigree for *n* generations above the sampled individual. At the same time, with each invocation of `search_up()` (including recursive invocations by the `search_up()` function itself), another algorithm, `search_down()`—a depth-first search for sampled individuals *down* the pedigree for *n* generations from the current vertex (individual)—is initiated *and* completed. All possible pairs of samples identified during this process are then identified as related. After running this algorithm, starting from each sample, each distinct pair that is found is stored in a list. The ancestry vectors of the members of those pairs are obtained from the pedigree, and then the ancestry match matrix for each related pair is computed using an outer-product-like operation on the ancestry vectors.

CKMRpop returns a data frame of all related pairs amongst the samples, along with the covariates of each pair member that are pertinent to a CKMR study. As noted in Bravington, Skaug, et al. (2016), these covariates include the time (year) and place (deme) of birth and of capture of each individual, and hence, the age of capture of each individual is known. These covariates determine the different categories the pairs belong to—each with their own kinship probabilities, *sensu* Bravington, Skaug, et al. (2016)—for calculating the CKMR pseudo-likelihood. Users can censor such information, if desired, to simulate a case where such covariates are latent variables and are not directly observed.

If the kin pairs found in a CKMR sampling experiment are largely independent of one another, then the CKMR pseudo-likelihood should behave much like a true likelihood (Bravington, Skaug, et al., 2016). This property is not enjoyed if identified kin pairs are not largely independent. To assist the user in identifying situations where kin pairs should not be treated as independent, CKMRpop constructs a graph with individuals as vertices and kin-pair relationships indicated by edges between the pair members, and it identifies the connected components in this graph. Such an exercise provides the user with an immediate and comprehensive list of all pairs that participate in triads or in groups with a higher number of pairs—an obvious indication of non-independence of the pairs. This graph is available in a tidy format providing easy viewing and manipulation by the user, and CKMRpop includes built-in plots for viewing and interpreting these relationships between pairs. Of course, for advanced users, CKMRpop provides a fully customizable individual-based simulation, the outputs of which can be used to directly evaluate the suitability of the pseudo-likelihood by calculating Bravington, Skaug, et al. (2016)'s Equation (4.5) which may suggest exploring an alternative to the pseudo-likelihood (Bravington, Skaug, et al., 2016, p. 268).

3 | RESULTS AND DISCUSSION

The development version of the R package that implements the functionality described above is available for free download from https://github.com/eriqande/CKMRpop, and the stable version of the package is available on CRAN at https://cran.r-project.org/web/packages/CKMRpop/index.html. The package vignettes give full examples of its use in two different scenarios: 1) a single-population version of an estuary-nursing elasmobranch and 2) a simple multiple population/sampling-site version that includes migration between demes. Additional examples for a variety of life histories will continue to be added to the package to help new users quickly understand how to set the SPIP parameter values to model their own research species and scenarios. We foresee that the availability of CKMRpop will simultaneously serve the multiple purposes of: providing an extensible platform for performing power-analysis simulations for CKMR, helping biological researchers understand the statistical underpinnings of CKMR and helping statistical researchers appreciate the consequences of biological complexities that impact CKMR; delivering much-needed information about the occurrence of non-target kin pairs, beyond parent–offspring and half-sibling pairs, in CKMR studies; and offering a simulation mechanism for understanding the potential to use more distant relationships in CKMR.

Another R package called fishSim, available on GitHub (https://github.com/SMBaylis/fishSim), allows the identification of kin pairs in simulated populations. It has been used to perform simulations assessing the effect of dispersal limitation and spatially variable sampling probabilities on CKMR estimates of population size, reproductive schedules and survival (Conn et al., 2020). Like CKMRpop, the fishSim package allows the user to configure a forward-in-time simulation of a population (or populations). Unlike CKMRpop, however, the code implementing the simulation in fishSim is entirely written in R and has been designed so that it is easily customizable, by writing a few custom functions in R, to account for nearly any imaginable life history or sampling oddity. By contrast, the population simulations possible in CKMRpop are restricted to those that can be obtained by adjusting the options in SPIP. On the other hand, the compiled code of SPIP runs considerably faster than fishSim, especially for large populations (Table 2).

FISHSIM's kin-finding algorithm employs ancestry vectors, such as CKMRpop, although does not take the same ancestry match matrix approach. There are several separate functions in fishSim for identifying related pairs. The first, `findRelatives()` (as well as a parallelized version, `findRelativesPar()`), appears to require that all possible pairs of samples be compared with one another—a computational task that scales quadratically with the number of sampled individuals. An additional fishSim function for finding relatives, called `quickin()`, is now recommended for routine use, as it is much faster; however, it may not capture the full array of possible relationships between pairs of inbred individuals (Shane Baylis, pers. comm.). CKMRpop's kin-finding algorithm, based on a recursive search of the pedigree, finds relatives with run times that scale linearly in the number of sampled individuals. Since that recursive search is implemented in compiled code, it runs quite quickly; however, there is some overhead in setting up the data structures, which is done in interpreted R code.

To assess run times of CKMRpop and fishSim, I simulated a population for 100 years with life history (survival and reproduction) like that in the FISHSIM vignette (see https://github.com/eriqande/CKMRpop_vs_fishSim_run_times for details). I started each simulation in CKMRpop or fishSim with the same number, N_0, of founders.

TABLE 2 Run times for the forward-in-time simulation phase and the kin-pair finding phase of comparable simulations using CKMRpop and fishSim. Times are elapsed times in seconds, t_{CP} for CKMRpop and t_{FS} for fishSim. For pair finding in fishSim, t_{FS}^{fRP} is using `findRelativesPar()` with 20 cores, and t_{FS}^{qk} is using `quickin()`. N_0: number of founders; $N_{P,CP}$ and $N_{P,FS}$:total number of simulated individuals in pedigree from CKMRpop and FISHSIM, respectively; S_{CP} and S_{FS}: number of sampled individuals, respectively

	Forward-in-time simulation				Finding sampled, related pairs				
N_0	$N_{P,CP}$	$N_{P,FS}$	t_{CP}	t_{FS}	S_{CP}	S_{FS}	t_{CP}	t_{FS}^{qk}	t_{FS}^{frP}
500	35,467	20,615	0.2	3.0	237	140	2.0	0.1	1.7
1000	70,736	60,583	0.3	7.0	448	280	3.3	0.2	7.9
2,500	177,270	149,274	0.7	19.4	1,168	700	9.1	0.5	78.0
5,000	352,876	290,158	1.4	40.7	2,326	1,400	18.2	1.4	488.9
10,000	707,442	546,793	2.8	88.1	4,620	2,800	36.5	4.3	3,457.5
25,000	1,766,157	1,413,680	7.2	349.1	11,319	7,000	95.1	32.2	53,263.5
50,000	3,536,055	2,846,662	15.5	1,096.4	22,935	14,000	213.0	123.2	–
100000	7,068,476	5,752,280	30.6	3,485.2	46,282	28,000	458.1	501.0	–

Because of differences in how the two packages implement population size regulation and sampling, it is not possible to simulate the same, exact, population sizes, but they are close in terms of the total number of individuals in the simulated pedigree, N_P, and the total number of individuals sampled, S. For different values of N_0, I ran a single simulation with each software package and recorded the number of seconds required to complete the population simulation and the kin-finding parts of each analysis. For kin-finding with `findRelativesPar()`, FISHSIM was allowed to use 20 cores, while kin-finding with `quickin()` and within CKMRpop was done with a single core. All analyses were run on a node of a modern, Linux computer cluster. The results are presented in Table 2.

As expected, the `findRelativesPar()` function in fishSim becomes impractical as the number of samples increases. On the other hand, fishSim's quickin() function is impressively fast—faster than the kin-finding routine in CKMRpop until the sample sizes get up into the low 10 thousands. As the population sizes increase, the run-time disparities for the forward population simulation phase of fishSim and CKMRpop become more pronounced. When millions of individuals are simulated, fishSim requires greater than 100-fold more time (as might be expected, as SPIP's simulation is done with compiled code). However, for simulating small populations, this does not represent a great cost, and fishSim's customizability in the R language could be a great benefit.

While the use of relationships beyond half-siblings (like half-avuncular, or half-first cousin relationships) in CKMR is an exciting prospect, doing so poses several complications that must be addressed. Most importantly, the ages and years of birth of the ancestors of individuals sampled for CKMR are not observable. This means there is not an obvious and direct way to control for variance in reproductive success and cohort strength while using such pairs for CKMR. By contrast, half-siblings born in different years are, by definition, not part of the same cohort, and therefore, restricting focus in CKMR to only those half-sibling pairs that include members from different cohorts provides one defensible and straightforward way to account for variance in reproductive success due to variation in recruitment strength. Since the reproductive events that might inflate the kinship probabilities for half-first cousins or half-avuncular pairs are those that give rise to the ancestors of the sample, rather than to the members of the sample, themselves, and because there is no way to know the ages of those ancestors, there is no easy way to restrict sampled pairs to only those whose ancestors were definitely not part of a shared reproductive event. Accordingly, using more distant kin pairs in CKMR may require accounting for variance in reproductive success by explicitly modelling and estimating it within the CKMR likelihood. It will be important to assess how the estimation of different parameters (*e.g.* the total population size versus the per-generation dispersal rate) is influenced by variance in reproductive success when using distant kin. Having a simulation package—especially one like CKMRpop, that allows the user to specify extra-Poisson variation in offspring number—will be critical for helping move forward in this realm.

A typical run of a moderately sized population in CKMRpop will be quite fast (less than a second, to several seconds). However, as is typical of forward-in-time, individual-based simulations, when the simulated population sizes get up into the millions, each run by SPIP can require considerable time and memory and produce sizable output. This, in turn, can substantially lengthen the time and memory required for functions in CKMRpop to process the output. For example, in simulating a teleost with a 20-year life span for 60 years with a stable population size of 1.5 million fish of age 2 and older, SPIP produces 9.3 Gb of output. The entire process of one simulation of this size and the associated identification and categorization of kin pairs from 10,000 sampled individuals requires roughly half an hour using four CPUs on a modern server. As seen in our comparison, fishSim likely would have required many hours for the forward-in-time simulation phase. An alternative to printing the entire pedigree (as done by SPIP) would be to store the ancestry vectors for all individuals for only a few generations and then only print them for the sampled individuals. The CKMRpop vignettes show an example of doing this with the SLiM software. Such an approach carries the advantage that the text output is considerably smaller, and there is no

longer a necessity to recursively traverse the pedigree to find the ancestry vectors for the sampled individuals, eliminating the overhead involved in setting up the data structures for that recursive traversal. Using SLiM would also provide the user with a far more customizable framework for simulations, allowing, for example, simulation of individuals interacting in continuous space and the simulation of genome-scale data for the sampled individuals. Nonetheless, for very large problems, an alternative to forward-in-time simulations may, in some cases, be desirable.

The approach presented here for identifying kin pairs by the ancestry match matrix is ideally suited for simulating kin pairs in a retrospective fashion. Just as the coalescent process allows modelling the lineages of a sample, backward in time, without having to explicitly simulate all the individuals in a population, so too will it be possible to simulate kin pairs by an analogous approach. Such an approach has already been implemented in non-age-structured populations to simulate segments of genome in diploid populations (Gasbarra et al., 2005). That method could be adapted to the simulation of kin pairs with very fast run times. Analogous to the coalescent, the run times of such simulations would depend only on the sample size and the age structure of the population, but not on the total population size. Furthermore, we note that the combination of the ancestry match matrix approach, for defining relationships, and the retrospective view for conceptualizing populations could provide for a fully general, efficient approach to calculating kinship probabilities, both with and without covariate uncertainty. Such a general framework could considerably simplify the implementation of CKMR inference across a wide variety of population and sampling scenarios and is actively being researched.

ACKNOWLEDGEMENTS

I am supremely grateful for my long association with Hans Skaug and Mark Bravington and the considerable efforts both of them have made in helping me to understand the vagaries of the CKMR approach. The inspiration for the R package CKMRpop came from working with David Portnoy and Carlos Garza to identify and evaluate opportunities for CKMR applications in the Atlantic and Pacific Oceans and the Gulf of Mexico. I gratefully acknowledge many helpful discussions with both of them. Shane Baylis pointed me to fishSim's impressive quickin() function for the run-time comparison, Paul Conn and Daniel Ruzzante provided helpful comments on a pre-submission version of the manuscript, and Robin Waples provided valuable feedback as an early user of CKMRpop. Peter Ralph and two additional anonymous referees provided many insightful comments that greatly improved this manuscript as well as the utility of CKMRpop.

OPEN RESEARCH BADGES

This article has earned an Open Data Badge for making publicly available the digitally-shareable data necessary to reproduce the reported results. The data is available at https://cran.r-project.org/web/packages/CKMRpop/index.html, https://github.com/eriqande/CKMRpop, https://eriqande.github.io/CKMRpop/, https://github.com/eriqande/CKMRpop_vs_fishSim_run_times.

DATA AVAILABILITY STATEMENT

CKMRpop is available as an R package with these associated links:

- Stable version available on CRAN: https://cran.r-project.org/web/packages/CKMRpop/index.html
- Development version and entire revision history on GitHub: https://github.com/eriqande/CKMRpop
- Online version of all package documentation: https://eriqande.github.io/CKMRpop/
- Code for run-timing comparisons of CKMRpop and fishSim: https://github.com/eriqande/CKMRpop_vs_fishSim_run_times

ORCID

Eric C. Anderson https://orcid.org/0000-0003-1326-0840

REFERENCES

Anderson, E. C., & Dunham, K. K. (2005). spip 1.0: a program for simulating pedigrees and genetic data in age-structured populations. *Molecular Ecology Notes*, *5*, 459–461.

Bravington, M. V., Grewe, P. M., & Davies, C. R. (2016). Absolute abundance of southern bluefin tuna estimated by close-kin mark-recapture. *Nature Communications*, *7*, 1–8. https://doi.org/10.1038/ncomms13162

Bravington, M. V., Skaug, H. J., & Anderson, E. C. (2016). Close-kin mark-recapture. *Statistical Science*, *31*, 259–274. https://doi.org/10.1214/16-STS552

Conn, P. B., Bravington, M. V., Baylis, S., & Ver Hoef, J. M. (2020). Robustness of close-kin mark–recapture estimators to dispersal limitation and spatially varying sampling probabilities. *Ecology and Evolution*, *10*, 5558–5569. https://doi.org/10.1002/ece3.6296

Eddelbuettel, D., & François, R. (2011). Rcpp: Seamless R and C++ Integration. *Journal of Statistical Software*, *40*, 1–18.

Fuentes-Pardo, A. P., & Ruzzante, D. E. (2017). Whole-genome sequencing approaches for conservation biology: Advantages, limitations and practical recommendations. *Molecular Ecology*, *26*, 5369–5406. https://doi.org/10.1111/mec.14264

Gasbarra, D., Sillanpää, M. J., & Arjas, E. (2005). Backward simulation of ancestors of sampled individuals. *Theoretical Population Biology*, *67*, 75–83. https://doi.org/10.1016/j.tpb.2004.08.003

Haller, B. C., & Messer, P. W. (2019). SLiM 3: Forward genetic simulations beyond the Wright-Fisher model. *Molecular Biology and Evolution*, *36*, 632–637.

Hanghøj, K., Moltke, I., Andersen, P. A., Manica, A., & Korneliussen, T. S. (2019). Fast and accurate relatedness estimation from high-throughput sequencing data in the presence of inbreeding. *Gigascience*, *8*, giz034. https://doi.org/10.1093/gigascience/giz034

Hillary, R. M., Bravington, M. V., Patterson, T. A., Grewe, P., Bradford, R., Feutry, P., Gunasekera, R., Peddemors, V., Werry, J., Francis, M. P., Duffy, C. A. J., & Bruce, B. D. (2018). Genetic relatedness reveals total population size of white sharks in eastern Australia and New Zealand. *Scientific Reports*, *8*, 1–9. https://doi.org/10.1038/s41598-018-20593-w

Miller, C. R., Joyce, P., & Waits, L. P. (2005). A new method for estimating the size of small populations from genetic mark–recapture data. *Molecular Ecology*, *14*, 1991–2005. https://doi.org/10.1111/j.1365-294X.2005.02577.x

Skaug, H. J. (2001). Allele-sharing methods for estimation of population size. *Biometrics*, *57*, 750–756.

Waples, R. S. (1998). Separating the wheat from the chaff: patterns of genetic differentiation in high gene flow species. *Journal of Heredity*, *89*, 438–450. https://doi.org/10.1093/jhered/89.5.438

How to cite this article: Anderson, E. C. (2022). CKMRpop: Forward-in-time simulation and tabulation of pairwise kin relationships in age-structured populations. *Molecular Ecology Resources*, 22, 1190–1199. https://doi.org/10.1111/1755-0998.13513