Week 3 Exercise: Kinship Coefficient Calculation Algorithm

Author: Nikola Rasevic

Student Number: 7748976

Department of Mathematics and Statistics

University of Ottawa

Nov 2nd, 2020

Introduction/Methods

The kinship coefficient between two individuals is the probability that two randomly selected alleles, one from each individual, are identical by descent. This is a measure of shared descent between two individuals. Furthermore, the inbreeding coefficient of an individual is equal to the kinship coefficient of their parents. An R algorithm was implemented where the Monte Carlo based approach and more specifically, a gene-dropping based approach, was used to calculate the kinship coefficients.

Before discussing the algorithm, the format of the pedigree files must be specified. The pedigree dataframe must contain 4 columns with headers: ind, father, mother, sex. The individual's number is represented as ind, and father/mother are the individual's parents. The individual's sex is represented in the sex column. 1 is male and 2 is female. If the individual has no parents, the mother and father columns are labeled as 0. An example dataframe and pedigree is shown in Table 1 and Figure 1.

Table 1. Pedigree Dataframe. The simplest pedigree was used. The individual is represented by the ind column. The father and mother columns represent the id of the parents of the individual. The sex of the individual is represented by the sex column. 1 is male and 2 is female.

ind	father	mother	sex
1	0	0	1
2	0	0	1
3	1	2	2

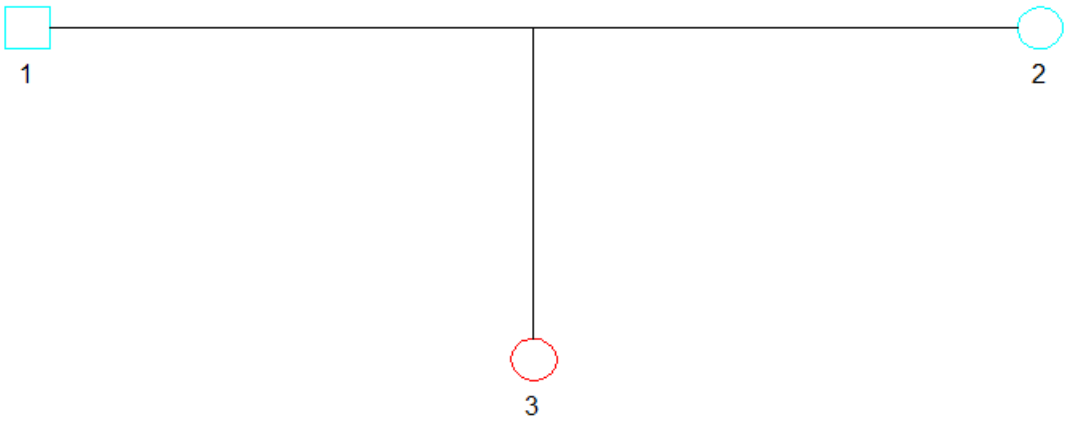

Figure 1. The Pedigree Diagram of the Dataframe from Table 1.

The function used to calculate the kinship is defined as sim(m,n,x,y,mat). Where m represents the pedigree dataframe, n is the number of simulations desired, x and y are the 2 individuals of interest, and mat is a Boolean argument for the access to the matching dataframe.

As mentioned before, the method for estimation involves a gene-dropping approach. This method was accomplished with the function p(m), where m is the pedigree dataframe. First, 3 new columns are added to the dataframe: allele1, allele2 and fill. The paternal and maternal alleles of the individual are represented by allele1 and allele2 respectively. When the allelic states are defined, fill is equal to 1. Otherwise, it is equal to 0. Next, the mother founders and father founders are located on the dataframe, and a proxy numeric allele is given. The paternal alleles for the father founders range from (1, …, 1+4f), and the maternal alleles for the father founders range from (3, …, 3+4f), where f represents the number of father founders. The paternal alleles for the mother founders range from (2, …, 2+4m), and the maternal alleles for the mother founders range from (4, …, 4+4m), where m represents the number of

mother founders. Once all the alleles have been defined for the mother and father founders, the fill column is equal to 1 for them. Table 2 is an example of how this may look.

Table 2. The Initial Dataframe created from p(m). The first four columns are identical to Table 1. The paternal and maternal allelic states are allele1 and allele2 respectively. Fill is a conditional variable that is equal to 1 if the maternal and paternal alleles of the individual is defined and 0 otherwise.

ind	father	mother	sex	allele1	Allele2	fill
1	0	0	1	1	3	1
2	0	0	1	2	4	1
3	1	2	2	0	0	0

Next, in order to fill the child's allelic state, a while-loop is to be introduced. While there is a 0 located in the fill column, an algorithm to fill the child's alleles must continue. The algorithm is as follows, for each row where the child has a father and mother, and does not have their alleles filled, allele1 of the child is equal to either of the father's alleles and allele2 of the child is equal to either of the mother's alleles. The choice of which allele is transmitted from parent to child is randomly selected with equal probabilities. Once all rows in the fill column are set to 1, the while-loop ends. After this selection and the completion of this algorithm, the table will look something like that of Table 3. This dataframe is the final output of the p(m) function.

Table 3. The Final Dataframe created from p(m). Allelic states are filled using a gene-dropping approach.

ind	father	mother	sex	allele1	allele1	fill
1	0	0	1	1	3	1
2	0	0	1	2	4	1
3	1	2	2	3	2	1

Next, the sim(m,n,x,y,mat) function is introduced. This function simulates the p(m) function n times and records the allelic states of the individuals of interest. Afterwards, the kinship coefficient is computed as the proportion of matches between 2 randomly sampled alleles, one from each individual. In detail, the sim(m,n,x,y,mat) function creates a new dataframe with 5 columns and n rows. Each row represents a

simulation. The columns are defined as x's maternal alleles, x's paternal alleles, y's maternal alleles, y's paternal alleles and match. Each time a simulation of p(x) occurs, the allelic states of the individuals of interest (x and y) are written into these first four columns. Next, an allele is randomly selected from each individual. If the alleles match, the value for the match column is 1, and 0 otherwise. The kinship coefficient is calculated as the proportion of 1's in the match column. This dataframe is called the matching dataframe. To illustrate what the matching dataframe may look like, Table 4 is an example dataframe for 10 simulations of the above pedigree. The kinship coefficient of individuals 1 and 3 are of interest. mat = TRUE is to be selected to view this matching dataframe. This can be accomplished with the following line of code: sim(ped_mat, 10, 1, 3, mat=TRUE).

Table 4. The Matching Dataframe created from sim(m,n,x,y,mat=TRUE). In this version, n (number of simulations) = 10. X = 1 and Y = 3 are the individuals of interest. The first 2 columns represent the allelic state of individual 1. The 3rd and 4th columns represent the allelic state of individual 3. A random sample of 1 from each individual is obtained, and if they match, the condition variable: match is equal to 1, and 0 otherwise.

1's paternal alleles	1's maternal alleles	3's paternal alleles	3's maternal alleles	match
1	3	1	2	1
1	3	1	4	1
1	3	1	4	0
1	3	3	2	0
1	3	1	4	0
1	3	3	4	0
1	3	1	2	0
1	3	1	4	0
1	3	1	4	0
1	3	1	2	1

This gave a kinship coefficient of 0.3. However, it is known that the true kinship coefficient is 0.25. This margin of error can be attributed to a low number of simulations. The higher the number of simulations, the more accurate the estimate. This will be further explained in the algorithm evaluation section. Figure 2 is a brief summary of the algorithm.

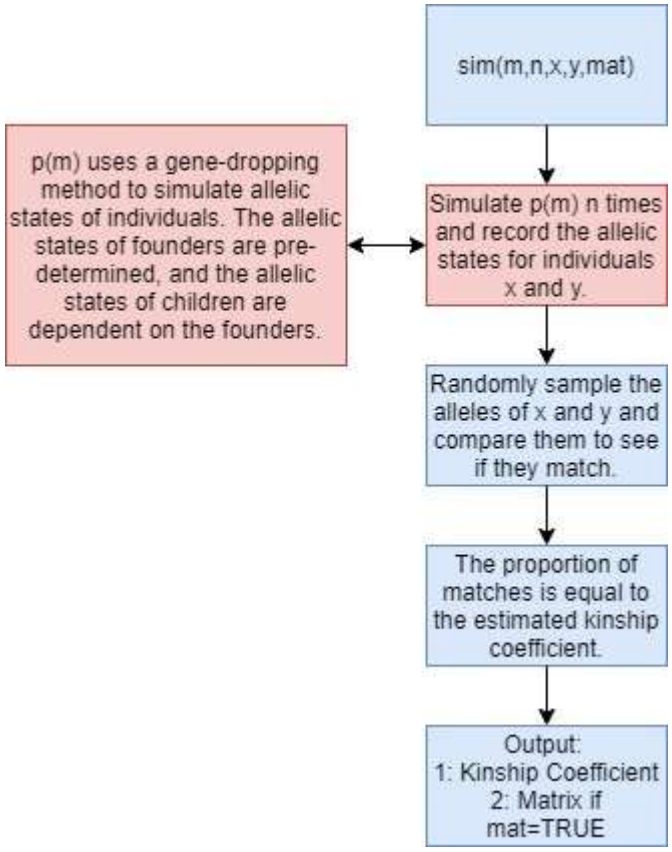

Figure 2. A Simplified Flowchart of the Kinship Coefficient Calculation Algorithm.

Algorithm Evaluation

Due to the gene-dropping approach for estimating allelic states, this algorithm can handle any pedigree size that R can handle. However, there are two disadvantages to gene-dropping. The first is the accuracy of the estimate, since it is simulation based. The second is the time needed to run the algorithm. The p(m) function has 2 for-loops nested in a while-loop as well as 2 more for-loops for a total of 5 loops. The sim (m,n,x,y,mat) function also has 2 for-loops, with p(m) nested in one of the loops. This influences the time needed to run the algorithm.

5 different pedigree datasets with differing complexities were used to evaluate the algorithm. They are described in the table below. Organized refers to the datasets listing the individuals in proper numeric order. However, in order to be efficient, only the evaluation of the most complicated dataset (5) will be

shown. 100 values of the kinship coefficient were estimated for the last pedigree. The kinship coefficient for two individuals that are close to one another (parent and child), semi-close to one another (grandparent and grandchild) and far from one another (great-grand parent and great-grand child) were calculated. The results of the evaluation are seen in Figures 3-5.

Table 5. Five Different Datasets Available.

Dataset	Number of Individuals	Number of Inbreeding	Organized
1	3	0	Yes
2	6	0	Yes
3	12	0	No
4	5	1	Yes
5	13	1	No

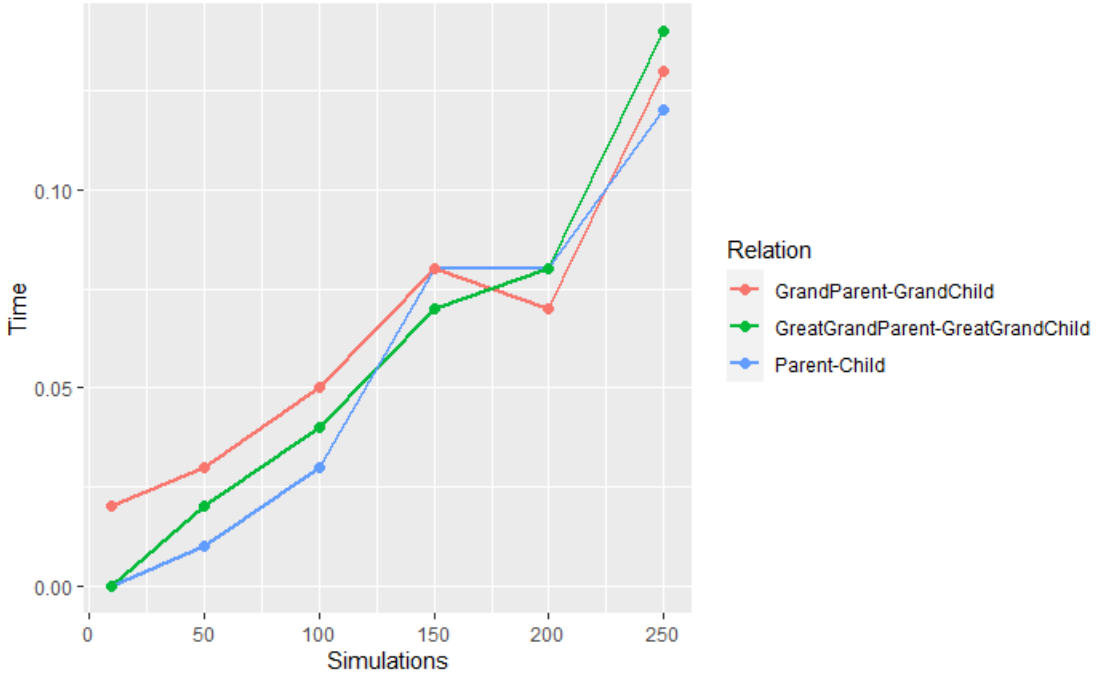

Figure 3. Algorithm Time (seconds) vs. Simulations for Pedigree #5. Time elapsed increases in a somewhat linear trend with increasing number of simulations. There does not seem to be an obvious effect between individual relations and the trend.

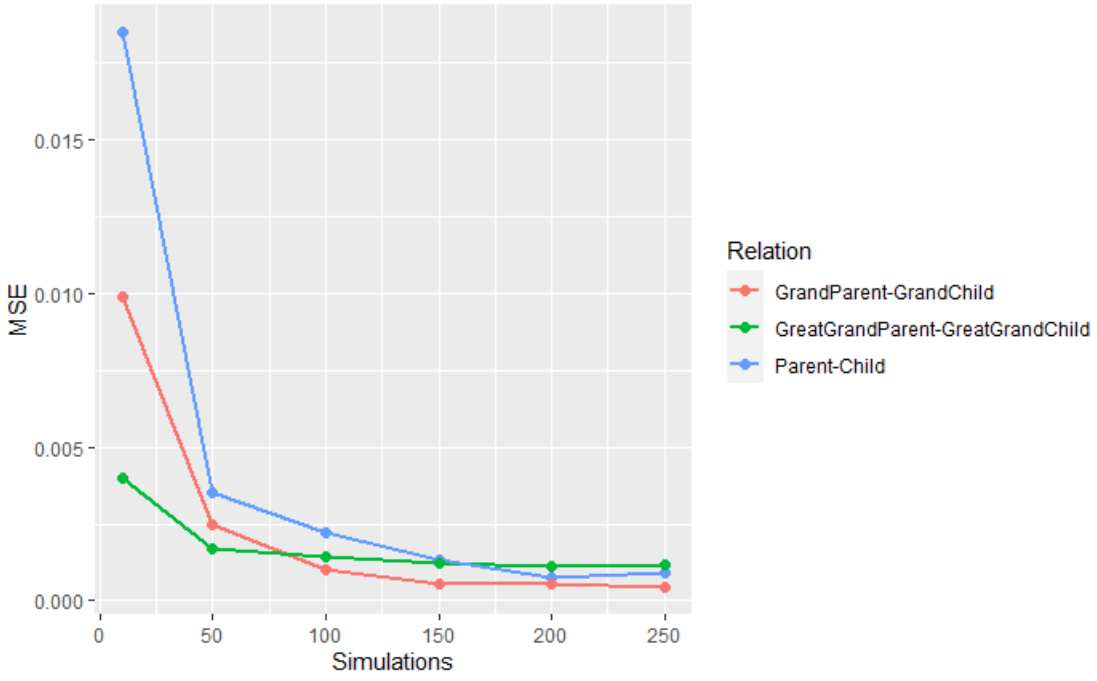

Figure 4. MSE vs. Simulations for Pedigree #5. The MSE decreases with increasing number of simulations. This means that with increased number of simulations, the algorithmic estimate becomes more accurate. However, the rate of decrease begins to plateau at roughly 150 simulations. As the relation between individuals becomes furthered, the effect that the number of simulations has on the MSE becomes more dampened.

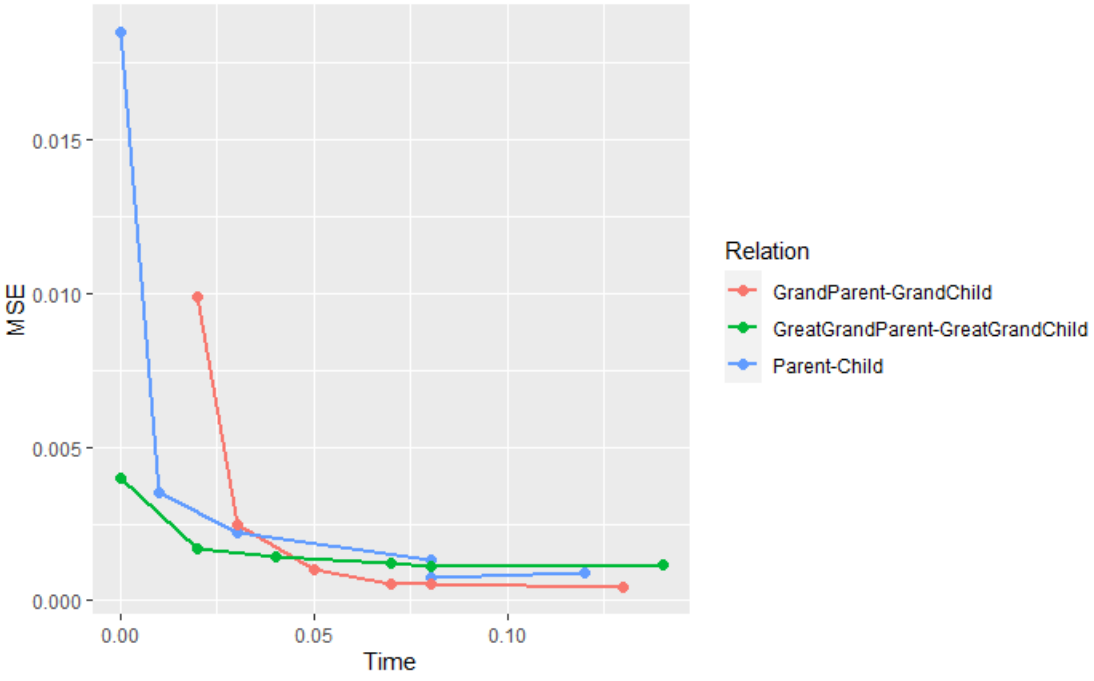

Figure 5. MSE vs. Algorithm Time (seconds) vs. for Pedigree #5. There is a trade off between time and accuracy. The optimal values for both time and MSE are between 50 to 200 simulations.

As seen above, as the number of simulations increase, the time elapsed for the algorithm increases which is to be expected. The mean-square error also tends to decrease with increasing simulations, however, this MSE begins to plateau < 0.0025 when simulations ≈ 150 whereas the time elapsed does not. When plotting MSE with time, it is seen that the algorithm is most optimal in terms of time efficiency and accuracy at around the 2^{nd} to 4^{th} points (50 to 200 simulations). An interesting observation is that, as the predicted kinship coefficient between two individuals decreases, the effect of number of simulations on MSE also decreases. Thus, the relation between individuals plays a role in the relation between MSE and number of simulations as well. Overall, the mean-square error is low with a large enough simulation size which means the algorithm is accurate, however, the gain of accuracy comes with the cost of increased time. After roughly 200 simulations, the gain of accuracy becomes negligible while time continues to increase. The other pedigrees were evaluated as well, and showed similar accuracy, but faster computation times. This makes sense as the algorithm is what determines

the accuracy of the estimate, and the algorithm is the same for all pedigrees. However, the other pedigrees are smaller and thus, the kinship coefficients are calculated more quickly.

Conclusion

An algorithm for the computation of the kinship coefficient for two individuals was created. The algorithm is Monte Carlo based, and more specifically, it involves gene-dropping. Due to the allelic states being simulated, the algorithm produces a biased estimate. When evaluating accuracy, the algorithm's MSE levels off at roughly 150 simulations (MSE < 0.0025). Time for the algorithm to run is also a concern due to the multitude of loops. Time continues to grow with increasing number of simulations. The size of the pedigree should also be considered when time is of concern. Thus, it is recommended that the user of this algorithm performs anywhere between 200-300 simulations. However, if time is not an issue, the accuracy will continue to grow with increased number of simulations, although minimally. If a low number of simulations is to be used, the relation of the two individuals can have an impact on the accuracy of the kinship coefficient estimate.