Estimation of Inbreeding Coefficient Using Monte Carlo Methods[*]

Yilin Lan

04/26/2022

Abstract

The Hardy-Weinberg Equilibrium plays a key role in methods of human genetics. It describes that the allele frequencies and the geno-type frequencies are constant across generations, and there should be a simple relation between these two types of frequencies in a large random-mating population given that there exist no selection, mutation, and migration. When the selection assumption is violated, the inbreeding model is used to test the departure from the HW equilibrium. There have been multiple methods of estimating the inbreeding coefficient, but none of them demonstrated a state-of-the-art performance due to the difficulties of solving complex integrals. Therefore, in this paper we propose an improvement to the current inbreeding coefficient estimation methods by introducing the Markov Chain Monte Carlo as an integration approximator. The experiment result shows that the Metropolis-Hastings-within-Gibbs was the best performer among all the MCMC approaches for both high-dimensional and low-dimensional data.

Contents

Keywords— Inbreeding Coefficient Estimation

[*]Code and data are available at: https://github.com/YilinLan/Inbreeding_Coefficient

1 Introduction

The Hardy-Weinberg(HW) Equilibrium, states that, in a large random-mating population, assuming no selection, mutation, migration, the allele frequencies and the geno-type frequencies are constant from generation to another, and there is a simple relation between geno-type and allele frequencies (HG 1908). This law is significant as many approaches in human genetics rely on the presence of Hardy- Weinberg Equilibrium. In particular, at a bi-allelic marker, the frequencies of the two alleles (A or B) are p and q, where p = 1.

However, in practice the assumptions are often violated, and to test the depar- ture from Hardy Weinberg Equilibrium we can simply calculate the expected geno-type frequencies and compare it with the observed ones using a chi-squared test. Alternatively, a number of methods have emerged for obtaining point esti- mates of f, a parameter measuring departure from HW caused by inbreeding. And the reasons why such methods are unsatisfactory are discussed by Ayres and Balding (Ayres 1998). If inbreeding (i.e. selection) is the main violation of HW assumptions, caus- ing deviation from HW, the inbreeding model may be appropriate (Mal ´ecot 1969), where p_{ij}, the relative frequency of the geno-type $AiAj$ is:

$$p_{ii} = p_i(f + (1 - f)p_i) \tag{1}$$

$$p_{ij} = 2p_i p_j (1 - f) \tag{1}$$

where pi denotes the frequency of allele A_i, and f is the inbreeding coefficient. When f = 0, equation one gives the HW proportions. When f = 1, the maxi- mum value, hetero-zygotes never arise. The value of f can be negative and it is bounded by below by the requirement that the population frequencies of each homo-zygote be non-negative, which lead to:

$$f \geq \left(\frac{-p_{min}}{1 - p_{min}}\right) \tag{2}$$

where p_{min} is the smallest frequency. (Ayres 1998)

In some models for population subdivision, f can be interpreted as the prob- ability that an individual's two genes are identical by descent (Crow and Kimura 1970), in which case it is constrained to be non-negative. Nei Chesser (Nei and Chesser 1983) and Robertson Hill (Robertson and Hill 1984) proposed their point estimators for the inbreeding coefficient, but these estimators do not explicitly take account of the inbreeding model and may, in the multi-allelic case, give estimates that conflict with the bound (Ayres 1998).

Ayres Balding (Ayres 1998) proposed the maximum likelihood estimator, which respect the bound under the inbreeding model. Assuming random sampling of genotypes, the likelihood is:

$$P(n_{ij}|f, p_1, ..., p_k) = C_1 \prod_{i=1}^{k} (p_i(f + (1 - f)p_i)^{n_{ii}} \prod_{j=i+1}^{k} (2p_i p_j (1 - f))^{n_{ij}} \tag{3}$$

where C1 is a constant. For k = 2, equation three is readily maximized (Weir 1996) to obtain

$$\hat{f}_{mle} = 1 - \left(\frac{2n_{12}n}{(2n_{11} + n_{12})(n_{12} + 2n_{22})}\right) \tag{4}$$

However, for k > 2, the likelihood cannot be maximized analytically, but numerical methods (Robertson and Hill 1984) and EM algorithm (Hill and Andwalliker 1995) can be employed. The likelihood function obtained by setting each parameter other than f (i.e. $p_i' s$) equal to its MLE $\hat{pi}$ provides a measure of the support given by the data to different possible values for f, but it ignores uncertainty in the p_i (Ayres 1998). While this problem can be solved by integration over the joint distribution of p_i, leading to marginal likelihood of f, the exact integration can be unfeasible, and we can approximate the integration by Markov Chain Monte Carlo(MCMC) algorithms.

All the data analysis in this paper uses R studio (R Core Team 2021) with tidyverse (Wickham et al. 2019), ggplot2 (Wickham 2016), kableExtra (Zhu 2021).

2 Data

2.1 Data Simulations

2.1.1 Dataset 1: Biallelic Site

When a specific locus in a genome that contains two observed alleles, then this site is called biallelic site, and in our study, k = 2. Suppose the inbreeding coefficient, f, among our observed sample is 0.05, and there are 200 people in our sample, with an allele frequency of $p_1 = 0.25$, $p_2 = 0.75$, the genotype frequencies can be simulated by equation (1) using the inbreeding model. Then, the phenotype frequencies are computed as $n_{ij} = n \times p_{ij}$, which is our observed data. However, this simulation usually gives us phenotype frequencies that are not integers, so we approximate them to give us a more practical observation.

2.1.2 Dataset 2: Multiallelic Site

When a specific locus in a genome that contains three or more observed alleles, then this site is called multiallelic site, and in our study, we particularly consider k = 6. Still suppose the inbreeding coefficient, f, among our observed sample is 0.05, and there are 1000 people in our sample, with an allele frequency of $p_i =$ (0.02, 0.06, 0.075, 0.085, 0.21, 0.55) for i = 1, 2, ..., 6, the genotype frequencies can be simulated by equation (1) using the inbreeding model. Similarly to k = 2, the phenotype frequencies are computed as $n_{ij} = n \times p_{ij}$, which is our observed data. Also, this simulation usually gives us phenotype frequencies that are not integers, so we approximate them again to give us a more practical observation.

2.2 Methodology

2.2.1 Metropolis-Hastings-within-Gibbs

Let's take a look at the full joint density:

$$\begin{aligned}
\pi(n_{ij}) = P(n_{ij}|f, p_1, \dots, p_k) &= C_1 \prod_{i=1}^{k} (p_i(f + (1-f)p_i))^{n_{ii}} \prod_{j=i+1}^{k} (2p_ip_j(1-f))^{n_{ij}} \\
&= C1 \prod_{i=1}^{k} (p_i(f + (1-f)p_i)^{n_{ii}} [(2p_ip_{i+1}(1-f))^{n_{i,i+1}} \dots (2p_ip_k(1-f))^{n_{i,k}}] \\
&= C_1[(p_1(f + (1-f)p_a))^{n_{11}} (2p_1p_2(a-f))^{n_{12}} \dots (2p_1p_k(1-f))^{n_{1,k}}] \\
&\quad [(p_2(f + (1-f)p_2))^{n_{22}} (2p_2p_3(1-f))^{n_{23}} \dots (2p_2p_k(1-f))^{n_{2k}}] \\
&\quad \dots \\
&\quad [(p_{k-1}(f + (1-f)p_{k-1}))^{n_{k-1,k-1}} (2p_{k-1}p_k(1-f))^{n_{k-1,k}}] \\
&\quad [(p_k(f + (1-f)pk))^{nkk}]
\end{aligned} \tag{5}$$

This is a product of a lot of numbers between 0 and 1, thus, taking log of this joint density can make computation easier and avoid the problem of extremely small value:

$$\begin{aligned}
log(\pi(n_{ij})) &= log(P(n_{ij}|f, p_a, \dots, p_k)) \\
&= n_{11}log(p_1(f + (1-f)p_1)) + n_12log(2p_1p_2(1-f)) + \dots + n_{1k}log(2p_1p_k(1-f)) \\
&\quad + n_{22}log(p2(f + (1-f)p_2)) + n_{23}log(2p_2p_3(1-f)) + \dots + n_{2k}log(2p_2p_k(1-f) \\
&\quad \dots \\
&\quad + n_{k-1,k-1}log(p_{k-1}(f + (1-f)p_{k-1}) + n_{k-1,k}log(2p_{k-1}p_k(1-f)) \\
&\quad + n_{kk}log(p_k(f + (1-f)p_k)) \\
&= \sum_{i=1}^{k} n_{ii}log(p_i(f + (1-f)p_i) + \sum_{i=1}^{k} \sum_{j=1+1}^{k} n_{ij}log(2p_ip_j)(1-f))
\end{aligned} \tag{6}$$

The proposal function for p_i:

$$p_u^t \sim Unif[max(0, p_u^{t-1} - \epsilon_p), min(p_u^{t-1} + \epsilon_p, p_u^{t-1} + p_v^{t-1}]$$

$$p_v^t = p_u^{t-1} + p_v^{t-1} - p_u^t \tag{7}$$

The proposal function for f:

$$f \sim Unif[max(0, p_u^{t-1} - \epsilon_p), min(p_u^{t-1} + \epsilon_p, p_u^{t-1} + p_v^{t-1})] \tag{8}$$

where p_{min} is the minimum pi at step t, ϵ_f.

The whole idea is that we will first update a pair of p_u and p_v, then setting the new proposed $p_u^t + p_v^t = previous$ $p_u + p_v$ guarantees that $\sum_{i=1}^k p_i = 1$.

Then we accept/reject our proposed p_u and p_v with Metropolis-Hastings rule, where the full joint density function and proposed function are from equation (5) and (7). Next we propose f with distribution in (8).(Note that only when we accept the p, we can propose new f). During the whole process, we adjust the ϵ_p in order to change the $\epsilon_f$to control our acceptance rate, the (positive) value of ϵ_p is chosen to obtain reasonable acceptance rates, if $\epsilon_p$is too large, the chain will 'sticks' too much in one place and, hence, converge very slowly; if too small, the chain will make frequent but very small moves and again will converge slowly. Since our joint density is very complicated, so we deal with it by taking logarithms, e.g. accept iff $log(U_n) < log(A_n)$, where $U_n \sim Unif[0, 1]$ and $A_n = \left(\frac{g(f^{new}, P^{new})q(f_{old}, P_{old})}{g(f^{old}, P^{old})q(f^{new}, P^{new})}\right)$

2.2.2 Gibbs Sampler

Instead of using regular Component-wise Metropolis-Hastings algorithm, we also tried to propose each coordinate according to its conditional distribution, conditioned on all other coordinates. From the full joint distribution (5), the conditional distributions of f, p_1, ..., p_k are computed as:

$$g(f|p_1, ..., p_k, n_{ij}) = \prod_{i=1}^k [f + (1-f)p_i]^{n_{ii}} \prod_{j=i+1}^k (1-f)^{n_{ij}} \tag{9}$$

for $\left(\frac{-p_{min}}{(1-p_{min})}\right) \leq f \leq 1$

$$g(p1|f, p_2, ..., p_k, n_{ij}) = p_1(f + (1-f)p_1)^{n_{11}}(p_1(1-f))^{n_{12}} \tag{10}$$

for $0 \leq p_1 \leq 1$

$$g(p_2|f, p2, nij) = p_2(f + (1-f)p_2)^{n_{22}}(p_2(1-f))^{n_{12}+n_{23}} \tag{11}$$

for $0 \leq p_2 \leq 1$

$$g(p_3|f, p2, nij) = p_3(f + (1-f)p_3)^{n_{33}}(p_3(1-f))^{n_{34}+n_{23}} \tag{12}$$

for $0 \leq p_3 \leq 1$

$$g(p_4|f, p2, nij) = p_4(f + (1-f)p_4)^{n_{44}}(p_4(1-f))^{n_{34}+n_{45}} \tag{13}$$

for $0 \leq p_4 \leq 1$

$$g(p_5|f, p2, nij) = p_5(f + (1-f)p_5)^{n_{55}}(p_5(1-f))^{n_{56}+n_{45}} \tag{14}$$

for $0 \leq p_5 \leq 1$

$$g(p_6|f, p2, nij) = p_6(f + (1-f)p_6)^{n_{66}}(p_6(1-f))^{n_{56}} \tag{15}$$

for $0 \leq p_6 \leq 1$

Using a systematically scan, we propose each p_i according to its conditional density and normalize them like we did for the initial values to ensure the sum is 1. In this case, we always accept our proposal, and then we update f using its conditional distribution.

2.2.3 Independence Sampler

As the performance of M-H algorithm being discussed in the results section, we notice it works quite well. Hence we thought it might be a good idea to use a special case of M-H algorithm, the independence sampler, to see if it can provide us with a more efficient approach. The full joint distribution is stated before as (5), the proposal distribution for the moving function f is:

$$f \sim Unif(max(\frac{-p_{min}^{t}}{1-p_{min}^{t}}, f-\epsilon_f), min(f+\epsilon_f, 1)) \tag{16}$$

where p_{min} is the minimum p_i at step t, ϵ_f. Since our joint density is very complicated and can be really small for the value, we deal with it by taking logarithms, e.g. accept if $log(U_n) < log(A_n)$, where $U_n \sim unif[0,1]$ and $A_n = \left(\frac{g(Y_n)q(X_{n-1})}{(g(X_{n-1})q(Y_n)}\right)$. Thus, the proposed states Y_n are independent of their previous states X_{n-1}. In implementation, we just ignore the simple case where k = 2 and only consider when k = 6, where MCMC is more likely to be required as numeric methods are implausible.

2.2.4 Other Monte Carlo methods

Importance sampling seems to be impossible if we don't have detailed information about the sample group, as we would not be able to find the kernel of distribution of allele frequencies and inbreeding coefficient (f,p_1,...,p_k), to sample from, so it's too inefficient, and in some way make no sense as in figure 2. Rejection Sampler also seem to be unreasonable, as it's to hard to find the suitable K and f(x) to bound our joint density function, even for simplest case where k=2.

3 Result

3.1 MLE on Dataset 1

When k = 2, we can use equation (4) for our MLE estimation. When simulating the data, we used the nearest integer of n_{ij}, e.g. 1 for 1.36, to get the first estimate, and since this value is below the exact value of n_{ij}, the estimate of f will be smaller/larger; then we use the next nearest integer of $n_{ij} + 1$, e.g 2 for 1.36 + 1, to get the second estimate festimate2. Averaging the first and second estimator would give us a final MLE estimate with a more narrow error than only using one of them. The final estimate we get is around 0.05281472 when n = 200. Notice: if we do not round n_{ij} as an integer, we can get 0.05 exactly.

3.2 M-H Algorithm on Dataset 1

Here's a table of acceptance rate by di erent values of ϵ_p.

Table 1: Acceptance rate by different epsilon p when k = 2

Epsilon	Acceptance Rate	Standard Error
0.01	0.7961	0.0046836
0.02	0.6367	0.0025845
0.03	0.4869	0.0025490
0.04	0.3691	0.0020880
0.05	0.2788	0.0020126
0.06	0.2202	0.0021710
0.07	0.1574	0.0022045
0.08	0.1285	0.0024934
0.09	0.1016	0.0028035
0.10	0.0922	0.0034737

From the summary table 1, we notice when $\epsilon_p = 0.02$ to 0.05, the acceptance rates are away from 0 and 1, as well as relative low standard error from 0.03 to 0.08. By setting $\epsilon_p = 0.03$, the algorithm was performed for 10000 iterations with a length of 1000 "burn-in," and we give our estimate for f:

$$\hat{f} = \frac{1}{(M-B)} \sum_{i=B+1}^{M} f_i = 0.05230045$$

and a 95% Confidence interval for this estimator is: (0.04777199, 0.05682892). This Confidence interval covers our true theoretical value 0.05, which is good.

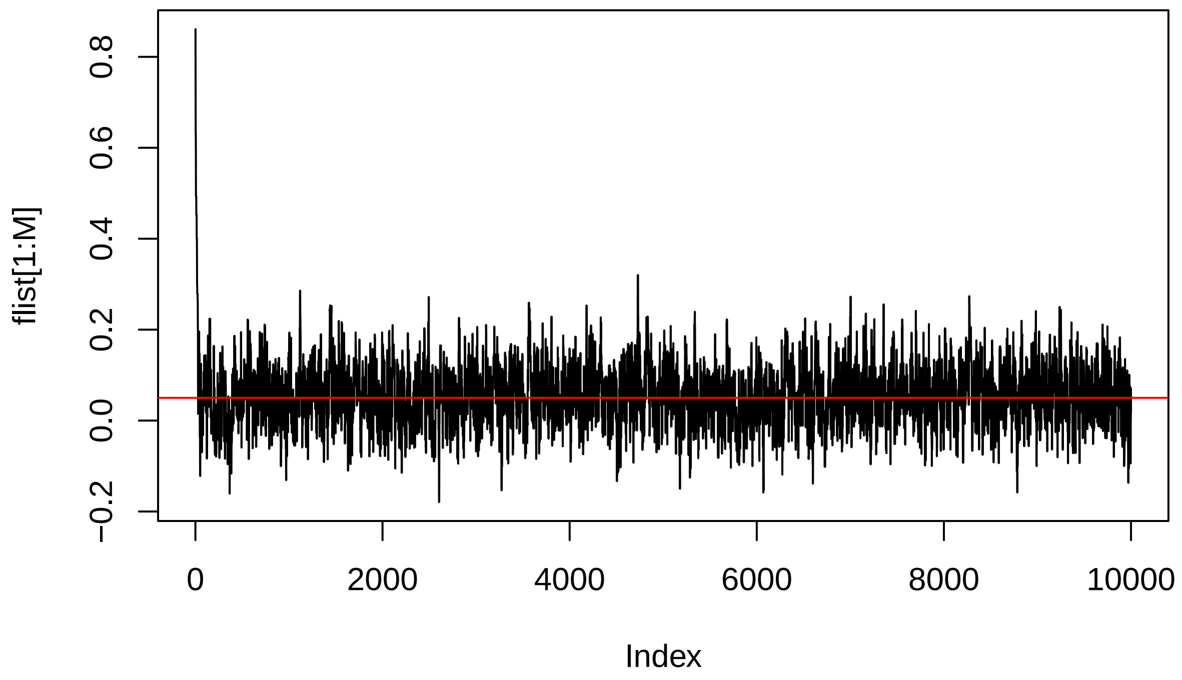

Figure 1: The chain converges compares to the true value in dataset 1

From the graph 1 we notice that its good "mixing," low uncertainty and the chain converges very quickly, and stays pretty close to the true value of f (around 0.05).

3.3 M-H Algorithm on Dataset 2

Here's a table of acceptance rate by different values of ϵ_p. We notice when $\epsilon_p = 0.006$ or 0.014, the acceptance rates are away from 0 and 1, as well as relative low standard error from 0.01 to 0.02. By setting $\epsilon_p = 0.01$, the algorithm was performed for 10000 iterations with a length of 1000 "burn-in," and we give our estimate for f:

Table 2: Acceptance rate by different epsilon p when k = 6

Epsilon	Acceptance Rate	Standard Error
0.001	0.7757	0.0152250
0.002	0.8177	0.0055514
0.003	0.7918	0.0018236
0.004	0.7407	0.0014351
0.005	0.6935	0.0013073
0.006	0.6444	0.0012522
0.007	0.5955	0.0011613
0.008	0.5612	0.0012465
0.009	0.5181	0.0010130
0.010	0.4823	0.0010515
0.011	0.4532	0.0010518
0.012	0.4229	0.0009680
0.013	0.3811	0.0009350
0.014	0.3631	0.0010044
0.015	0.3399	0.0010051
0.016	0.3231	0.0009907
0.017	0.2950	0.0008371
0.018	0.0000	0.0000000
0.019	0.2572	0.0008286
0.020	0.2460	0.0008862

$$\hat{f} = \frac{1}{(M-B)} \sum_{i=B+1}^{M} f_i = 0.05072752$$

and a 95% Confidence interval for this estimator is: (0.04837185, 0.05308320). From the table 2 ,this Confidence interval covers our true theoretical value 0.05, which is good.

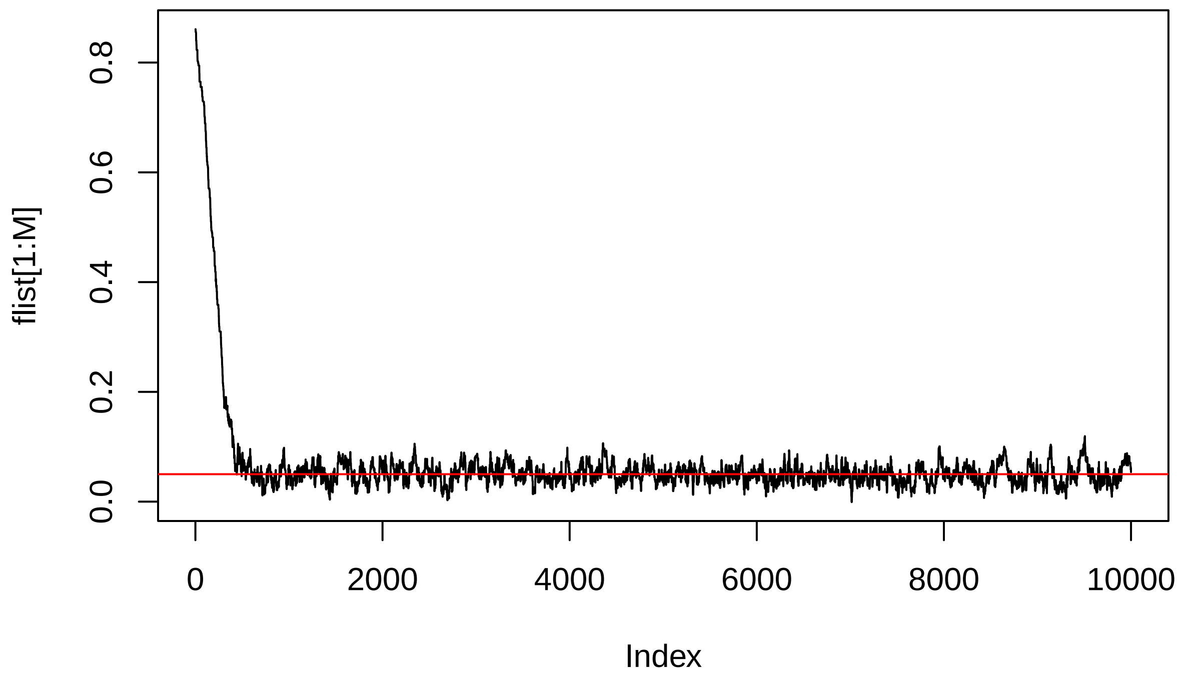

Figure 2: The chain converges compares to the true value in dataset 2

Seeing from the graph 2 we found that the chain converges within 1000 iterations, and it stays close to the true f value as well, And good 'mixing' and pretty low uncertainty.

3.4 Gibbs Sampler

$$\hat{f} = \frac{1}{(M-B)} \sum_{i=B+1}^{M} f_i = 0.003558$$

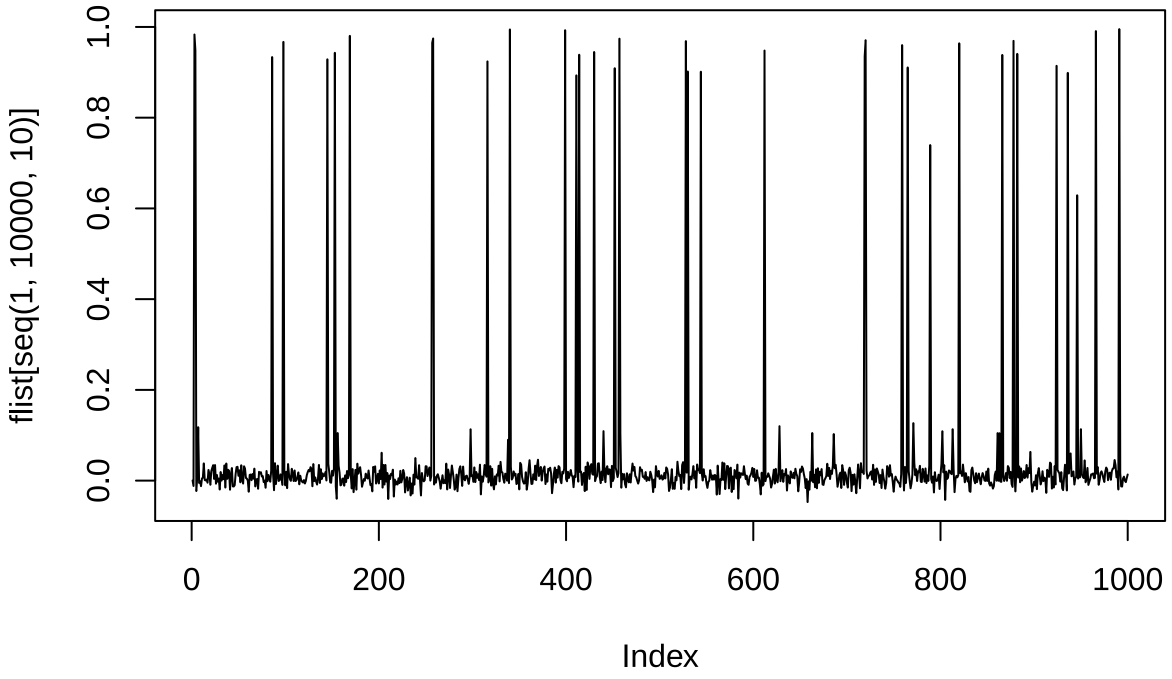

Figure 3: The chain converges compares to the true value under Gibbs sampler

Seeing from the graph 3 we notice that, the chain converges, to around 0.03, performing not well as a little bit away from our true value (0.05). Also, we can see that it has very high uncertainty, which is mainly due to the fact that the coefficient, f is correlated with our p_i. When we were using the conditional distribution to generate p_i, problems arises, which will be discussed in details in the discussion section.

3.5 Independence Sampler

The result was not good, whereas the chain does not actually converge although it converges to our true value at the beginning for a moment. Even when we tuned the eps.p, it does not make any better.

$$\hat{f} = \frac{1}{(M-B)} \sum_{i=B+1}^{M} f_i = 0.02897391$$

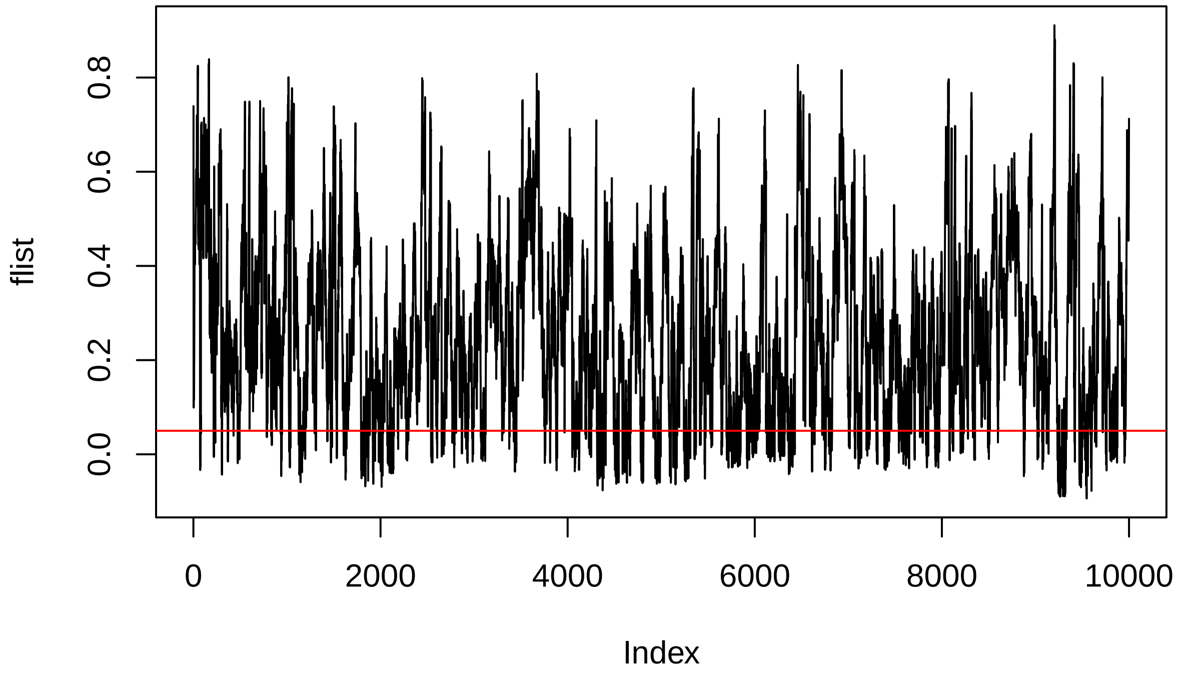

Figure 4: The chain converges compares to the true value under independence sampler

Seeing the graph 4 and the results, we can see that Independence Sampler performs poorly, converges far away from our goal (red line – 0.05), and it has very high uncertainty. The reason of this failure might be that Y_n are i.i.d and independent of X_{n-1}, actually they are correlated by looking the formula (1). We know that in Genetic field, people's Genes are all correlated, just isolated some genotypes will definitely a ect the inbreeding coefficient, which could be the main part for this algorithm's failure.

3.6 Importance Sampling

$$\hat{f} = 0.5205245$$

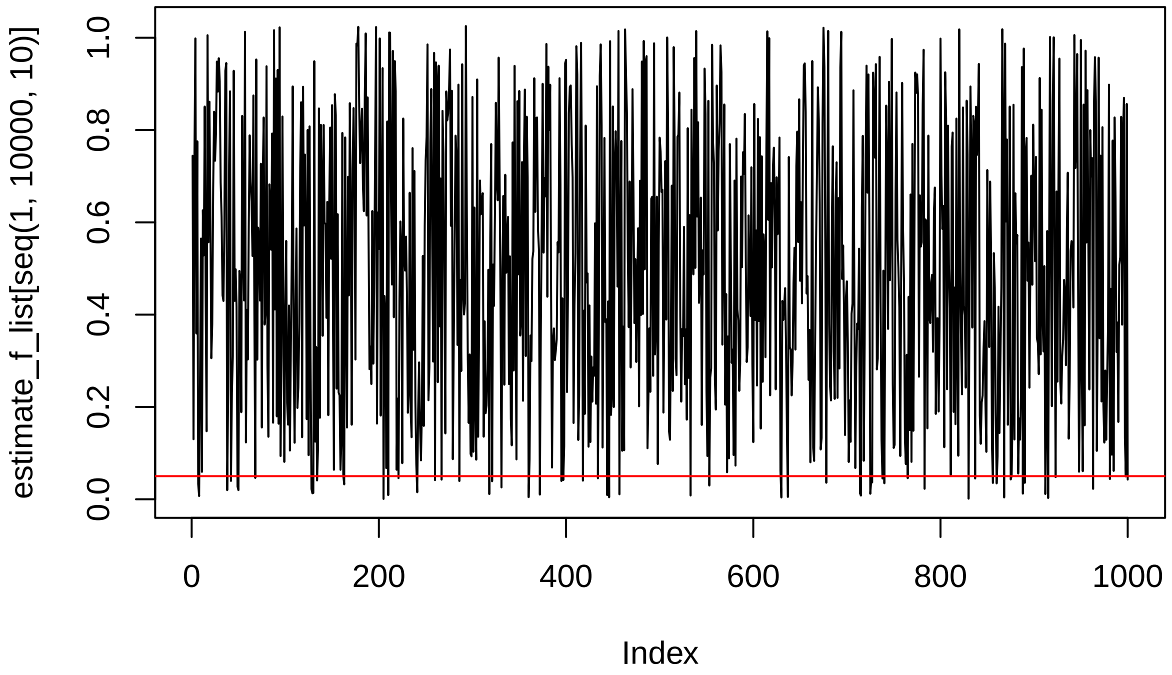

Figure 5: The chain converges compares to the true value under importance sampling

From the graph 5,The Importance Sampling performs quite bad even for the simplest case when k = 2, as stated in section 3.4 the distribution of f is really hard to determine, that's the main reason causing this results. Moreover, we found that Importance sampler could be used for some simple, low-dimention function, but impossible to implement for high-dimention, complicate functions.

4 Discussion

For the Gibbs sampler, since the conditional distribution for each parameter is very unique, so we have to generate them from their density function. Firstly we considered to use a MCMC method to generate the sample, but later we man- aged to generate a sample each time by looking for the root of $U_n = F(x)$ where $Un \sim Unif[0,1]$ and $F(x)$ is the cumulative distribution for the parameters conditional distribution, which is an approach of inverse CDF. However, as the conditional distribution is in form of product and power of number of observation, it can get significantly small. To avoid the problem of zero in the denominator when updating f, we set the parameters to 10^20 if the computer recognize it as a zero, especially for $p_4, ..., p_6$. Even though, the value of above parameters gets close to zero frequently, causing f close to 1 frequently as shown in the graph. Unfortunately, we didn't manage to solve this problem by taking log of the conditional distribution, as the scale of CDF changes if the density changes, making it really difficult and inaccurate to find roots for $U_n = log(F(x))$. Moreover, the R package, distr, was employed to generate random samples from the conditional distribution. But it spends super long time make the density as a distribution when the power of the function is big (e.g. > 4).

Overall, the MCMC that we use, Metropolis-Hastings-within-Gibbs performed the best, in both low-dimension and high-dimension case, providing us with a pretty accurate estimate to the true parameter. However, the proposal distribution requires prior knowledge of the parameters. However, all other algorithms were unsatisfactory. For Independent Sampler, the estimate was far away from the true value, and gives a large standard error. On the other hand, both of the Importance Sampling and Rejection Sampler performed pretty bad, as K and $f(x)$ are hard to find. Gibbs Sampler provided an estimate that is more close true value comparing with other unsatisfactory MCMC algorithms, but still with a high uncertainty. Nevertheless, the Gibbs sampler itself is a very powerful and efficient MCMC algorithm as long as we have a reasonable conditional distribution.

Reference

Ayres, D. J., K. L. Balding. 1998. “Measuring Departures from Hardy–Weinberg: A Markov Chain Monte Carlo Method for Estimating the Inbreeding Coefficient.” *Heredity*, no. 80: 769–77.

Crow, J. F., and M. Kimura. 1970. “An Introduction to Population Genetics Theory.”

HG, Hardy. 1908. “Mendelian Proportions in a Mixed Population.” *Science*, no. 28: 49–50.

Hill, Babiker, W. G., and D. Andwalliker. 1995. “Estimation of Inbreeding Coefficients from Genotypic Data on Multiple Alleles, and Application to Estimation of Clonality in Malaria Parasites.” *Genet Res*, no. 65: 53–61.

Mal´ecot, G. 1969. “Measuring Departures from Hardy–Weinberg: A Markov Chain Monte Carlo Method for Estimating the Inbreeding Coefficient.” *The Mathematics of Heredity.*

Nei, M., and R. K. Chesser. 1983. “Estimation of Fixation Indices and Gene Diversities.” *Ann Hum Genet*, no. 47: 253–59.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Robertson, A., and W. G. Hill. 1984. “Deviations from Hardy–Weinberg Pro- Portions: Sampling Variances and Use in Estimation of Inbreeding Coefficients.” *Genetics*, no. 107: 703–18.

Weir, B. S. 1996. “Genetic Data Analysis II Sinauer Associates.”

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D’Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. “Welcome to the tidyverse.” *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Zhu, Hao. 2021. *kableExtra: Construct Complex Table with ’Kable’ and Pipe Syntax.* https://CRAN.R-project.org/package=kableExtra.