

Probabilities of multilocus genotypes in SIB recombinant inbred lines

Kamel Jebreen, Marianyela Petrizzelli, Olivier C. Martin

1 Introduction

Set the working directory to the emplacement of the source file “SibFun.R” and load it to the working environment. You can download the “SibFun.R” file from <https://github.com/Kamel20/PMGISRIL>,

```
setwd("the directory")
source("sibFun.R")
```

Then load the required packages:

```
library(eplot)
library(rlist)
library(rmarkdown)
```

2 Input

This code just needs to input the locus number and the recombination rates values. For example for $L = 3$

```
L = 3
recRates = c(0.4, 0.2, 0.3)
```

3 The variables names

To gain time we found the list of inheritance indexes (all Qs) that contribute in the system before.

```
allQs = list.load("allVarTillL=10.rds")
EquivalenceQs = list.load("allContrVarTillL=10.rds")
distinctQs = allQs[[L]]$symQs
nonSymQs = allQs[[L]]$nonsymQs
scEq = EquivalenceQs[[L]]
```

or you can use their function to create it again

```
allQs = systemVar(L)
```

```
##
```

```
## Find the inheritance indexes that are contributing to this system, please wait ...
```

So all Qs variables are

```
nonSymQs = allQs$nonsymQs
nonSymQs
```

```
## [1] "000" "001" "002" "002" "010" "011" "012" "012" "020" "021" "022"
## [12] "023" "020" "021" "023" "022" "011" "010" "012" "012" "001" "000"
## [23] "002" "002" "021" "020" "022" "023" "021" "020" "023" "022" "022"
## [34] "023" "020" "021" "023" "022" "020" "021" "002" "002" "000" "001"
```

```
## [45] "012" "012" "010" "011" "022" "023" "021" "020" "023" "022" "021"
## [56] "020" "012" "012" "011" "010" "002" "002" "001" "000"
```

and the distinct Qs are

```
distinctQs = allQs$symQs
distinctQs
```

```
## [1] "000" "001" "002" "010" "011" "012" "020" "021" "022" "023"
```

Note that the first equation of the linear system is given by the fact that $\sum(Q_s) = 1$

```
SQ = table(nonSymQs)
SQ
```

```
## nonSymQs
## 000 001 002 010 011 012 020 021 022 023
## 4 4 8 4 4 8 8 8 8 8
```

This means that

```
## 4Q(000)+4Q(001)+8Q(002)+4Q(010)+4Q(011)+8Q(012)+8Q(020)+8Q(021)+8Q(022)+8Q(023)=1
```

For each distinct Q, the equivalence Q in F_2 generation are

```
scEq = allCrossOver(distinctQs = distinctQs)
```

```
## 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
```

```
scEq
```

```
## [[1]]
## [1] "020" "022" "002" "000"
##
## [[2]]
## [1] "020" "022" "002" "000"
##
## [[3]]
## [1] "021" "023" "001" "002"
##
## [[4]]
## [1] "020" "022" "002" "000"
##
## [[5]]
## [1] "020" "022" "002" "000"
##
## [[6]]
## [1] "021" "023" "001" "002"
##
## [[7]]
## [1] "010" "012" "020" "023"
##
## [[8]]
## [1] "010" "012" "020" "023"
##
## [[9]]
## [1] "012" "011" "021" "022"
##
## [[10]]
## [1] "012" "011" "021" "022"
```

4 Find the system $AQ = B$

The system required to compute all self-consistent equations except one that replaced by equation $\sum(Qs) = 1$.

```
res = twoWayRILsib(L, distinctQs, nonSymQs, scEq)

##
## # ===== 3 - Loci ===== #
##
## 1. The First equation in the system is:
## 4Q(000)+4Q(001)+8Q(002)+4Q(010)+4Q(011)+8Q(012)+8Q(020)+8Q(021)+8Q(022)+8Q(023)=1
##
## 2. Computing the Self-consistent equations: ...
## 1 of 9
## 2 of 9
## 3 of 9
## 4 of 9
## 5 of 9
## 6 of 9
## 7 of 9
## 8 of 9
## 9 of 9
## done
```

Hence, the matrix A is

```
A = res$A
A[1:3,1:2]

##      000      001
## SQ  "4"      "4"
## 000 "2*(0.5)*(1-r12)*(1-r23)-1" "0"
## 001 "2*(0.5)*(1-r12)*(r23)"      "-1"
```

and, the matrix B is

```
B = res$B
B[1:3]

## [1] 1 0 0

To solve the linear system you should evaluate the symbolic matrix

AA = evalMatrix(A = A, recRates = recRates)

## r12 = 0.4      r23 = 0.2      r34 = 0.3      r13 = 0.44      r24 = 0.38      r14 = 0.476
```

For example,

```
AA[1:3,1:2]

##      [,1] [,2]
## [1,] 4.00  4
## [2,] -0.52  0
## [3,] 0.12 -1
```

Hence,

```
sol = solve(AA, B)
names(sol) = distinctQs
```

```
sol
##      000      001      002      010      011      012
## 0.03413811 0.01322519 0.01308306 0.01125016 0.02666891 0.01045223
##      020      021      022      023
## 0.01164653 0.01027273 0.02641467 0.01048960
```

This means,

```
## Q(0,0,0) = 0.03413811 , Q(0,0,1) = 0.01322519 , Q(0,0,2) = 0.01308306
## Q(0,1,0) = 0.01125016 , Q(0,1,1) = 0.02666891 , Q(0,1,2) = 0.01045223
## Q(0,2,0) = 0.01164653 , Q(0,2,1) = 0.01027273 , Q(0,2,2) = 0.02641467
## Q(0,2,3) = 0.01048960
```

5 Verification by simulation

Note that the sum of all Q's equal to 1

```
QsProbs = rbind(sol, table(nonSymQs))
sum(QsProbs[1,] * QsProbs[2,])

## [1] 1
```

5.1 Convert Qs to Frequencies

We should first convert the Qs to genotypes frequencies:

```
Fexp = QsToFreq(L, sol)

## 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Fexp

## [1] 0.17056473 0.09414115 0.08767703 0.14761709 0.14761709 0.08767703
## [7] 0.09414115 0.17056473
```

5.2 Compute the frequencies by simulation

Choose high number RIL generation, we choose $nRILS = 50000$ RIL.

```
nRILS = 50000

Then, define the binary hetzrgouse  $F_2$  genertaion:

childGenotype = matrix(c(rep(0, L), rep(1, L), rep(0, L), rep(1, L)), ncol = L, byrow = TRUE)
childGenotype
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    1    1    1
## [3,]    0    0    0
## [4,]    1    1    1
```

Now, run the simulation over $nRILS$

```
f = rep(0, 2^L)
for (i in 1:nRILS){
  child = Get_One_RIL(L, recRates, childGenotype, type = "sib")
  f[binTodec(child[1,])+1] = f[binTodec(child[1,])+1]+1
}#EndFor
Fsim = f /nRILS
Fsim
```

```
## [1] 0.17166 0.09136 0.08858 0.14824 0.14674 0.08782 0.09514 0.17046
```

5.3 Simulation Accuracy

You can compare the analytic results with the simulation one and compute the mean square error (MSE) for that

```
mean((Fexp - Fsim)^2)
```

```
## [1] 1.492024e-06
```

For more details see ([Jebreen et al., 2019](#)).

Bibliography

Jebreen, K., Petrizzelli, M., and Martin, O. C. (2019). Probabilities of multilocus genotypes in sib recombinant inbred lines. *Statistical Genetics and Methodology, a section of the journal Frontiers in Genetics*(Submitted).