

Evolutionary Modelling

Dr Tin-Yu Hui

<tin-yu.hui11@imperial.ac.uk>

Feb 2021

I suppose you have already known...

- some statistical properties on the **Wright-Fisher** (WF) model and genetic drift [Matteo]
- the basic principles of gene-drive, and how we apply the technology to help suppress mosquito populations and eradicate human malaria [Austin]

Objectives

- Build stochastic forward simulators in R
 - for the WF model
 - for a gene drive model
- Use them to solve problems
 - i.e. Monte-Carlo methods

1. Drift simulator

- The WF model describes the change in allele frequency in a population due to genetic drift.
- During reproduction, an offspring is formed by sampling (with replacement) from the parent generation.
 - famously modelled by the urn model
 - binomial sampling of alleles
- Other assumptions include
 - constant population size
 - discrete generation
 - random mating
 - closed population
 - no mutation, no selection

- One locus, two alleles (labelled as 0 and 1)
- Diploid individuals
- Parameters:
 - N : the (constant) population size
 - p_0 : the initial frequency of allele “0”
 - t : number of generations

```
# GENETIC DRIFT SIMULATOR  
sim_genetic_drift<-function()  
{  
  
}
```

```
# GENETIC DRIFT SIMULATOR
sim_genetic_drift<-function(p0=0.5, t=10, N=10)
{
# INITIALISATION

# PROPAGATION

# OUTPUTS
}
```

- Initialisation
 - what are the object and data types (lists/matrices/vectors, numeric/character/logical)?
 - N individuals, each carrying two alleles, matrices maybe?
 - how do we utilise the input arguments?
 - the initial condition(s) of the simulation?


```

# INITIALISATION
# population IS A LIST STORING ALL THE ALLELIC CONFIGURATIONS
# FROM GEN 0 TO t, HENCE (t+1) ELEMENTS
population<-list()
length(population)<-t+1
# (OPTIONAL) GIVE NAMES TO THE ELEMENTS OF population
names(population)<-rep(NA, t+1)
for (i in 1:length(population))
    {names(population)[i]<-paste(c('generation', i-1), collapse='')}
# ALSO KEEP TRACK ON THE ALLELE FREQ OVER TIME, AS A VECTOR
allele.freq<-rep(NA, t+1)

# AT GEN 0 THERE ARE (2*N*p0) COPIES OF ALLELE 0
# SHUFFLE THESE ALLELES, AND ASSIGN THEM INTO A 2-BY-N MATRIX
k<-ceiling(2*N*p0)
population[[1]]<-matrix(sample(c(rep(0, k), rep(1, 2*N-k))), nr=2)
# THE INITIAL ALLELE FREQ
allele.freq[1]<-sum(population[[1]]==0)/(2*N)

```

- Propagation
 - use information from gen i to sample alleles at gen $(i+1)$
 - or from gen $(i-1)$ to i
 - use a loop for recursive calculation?
 - constant N throughout
 - `for()`? `sample()`?

```
# PROPAGATION
for (i in 1:t)
{
  # SAMPLE ALLELES FOR THE NEXT GEN
  # BASED ON THE allele.freq AT THE CURRENT GEN
  population[[i+1]]<-matrix(sample(0:1, size=2*N,
prob=c(allele.freq[i], 1-allele.freq[i]), replace=TRUE),
    nr=2)
  # THE ALLELE FREQ AT THE NEXT GEN
  allele.freq[i+1]<-sum(population[[i+1]]==0)/(2*N)
}
```

- Output
 - what are the outputs?
 - `return()` and exit the function

```
# OUTPUTS. YOU CAN ONLY return() ONE ELEMENT FROM A FUNCTION  
# TO RETURN MULTIPLE ELEMENTS, PUT ALL YOUR OUTPUTS INTO ONE BIG LIST  
return(list(population=population, allele.freq=allele.freq))
```

```

sim_genetic_drift<-function(p0=0.5, t=10, N=50)
{
# population IS A LIST STORING ALL THE ALLELIC CONFIGURATIONS
# FROM GEN 0 TO t, HENCE (t+1) ELEMENTS
population<-list()
length(population)<-t+1
# (OPTIONAL) GIVE NAMES TO THE ELEMENTS OF population
names(population)<-rep(NA, t+1)
for (i in 1:length(population))
    {names(population)[i]<-paste(c('generation', i-1), collapse='')}
# ALSO KEEP TRACK ON THE ALLELE FREQ OVER TIME, AS A VECTOR
allele.freq<-rep(NA, t+1)

# AT GEN 0 THERE ARE (2*N*p0) COPIES OF ALLELE 0
# SHUFFLE THESE ALLELES, AND ASSIGN THEM INTO A 2-BY-N MATRIX
k<-ceiling(2*N*p0)
population[[1]]<-matrix(sample(c(rep(0, k), rep(1, 2*N-k))), nr=2)
# THE INITIAL ALLELE FREQ
allele.freq[1]<-sum(population[[1]]==0)/(2*N)

# PROPAGATION
for (i in 1:t)
    {
    # SAMPLE ALLELES FOR THE NEXT GEN
    # BASED ON THE allele.freq AT THE CURRENT GEN
    # FIRST WAY OF SAMPLING
    population[[i+1]]<-matrix(sample(0:1, size=2*N, prob=c(allele.freq[i], 1-allele.freq[i]),
        replace=TRUE), nr=2)
    # THE ALLELE FREQ AT THE NEXT GEN
    allele.freq[i+1]<-sum(population[[i+1]]==0)/(2*N)
    }

# PUT ALL YOUR OUTPUTS INTO ANOTHER LIST AND THEN RETURN THEM
return(list(population=population, allele.freq=allele.freq))
}

```

```
R Console
> set.seed(12345)
> sim_genetic_drift(p=0.5, t=4, N=10)
$population
$population$generation0
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    1    0    0    0    0    0    1    1    1
[2,]    1    1    0    0    1    0    0    1    0    1

$population$generation1
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    0    0    0    1    0    1    1    0    0
[2,]    1    0    1    0    1    1    0    1    0    1

$population$generation2
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    0    0    1    1    1    0    1    0    0    1
[2,]    1    0    1    1    0    0    1    1    1    1

$population$generation3
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    0    0    0    0    0    0    1    1    0    1
[2,]    1    0    1    1    0    1    1    0    0    1

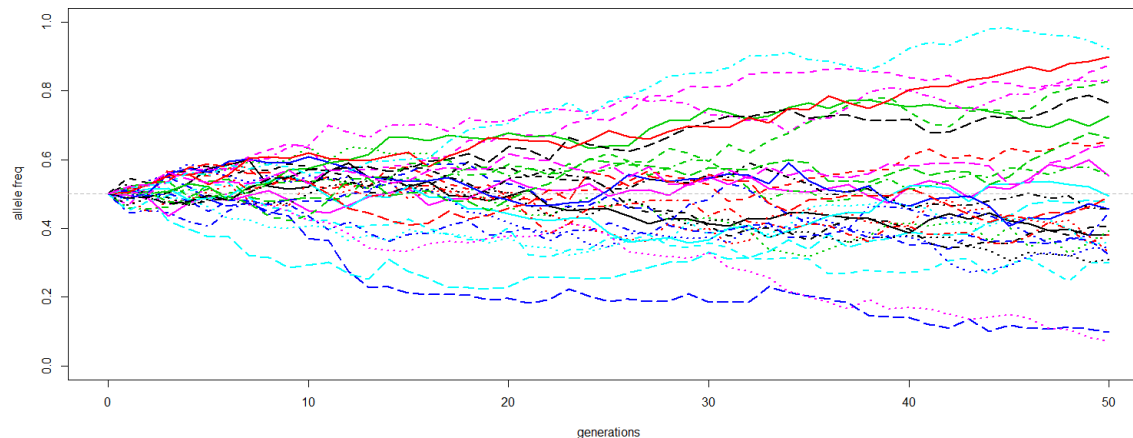
$population$generation4
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    0    0    1    1    1    0    1    1    0
[2,]    0    0    0    0    0    1    1    0    1    0

$allele.freq
[1] 0.50 0.50 0.40 0.55 0.55

> |
```

Some probabilistic questions

- The WF model suggests genetic drift will not change the mean allele frequency but will increase the uncertainty (variance) surrounding it

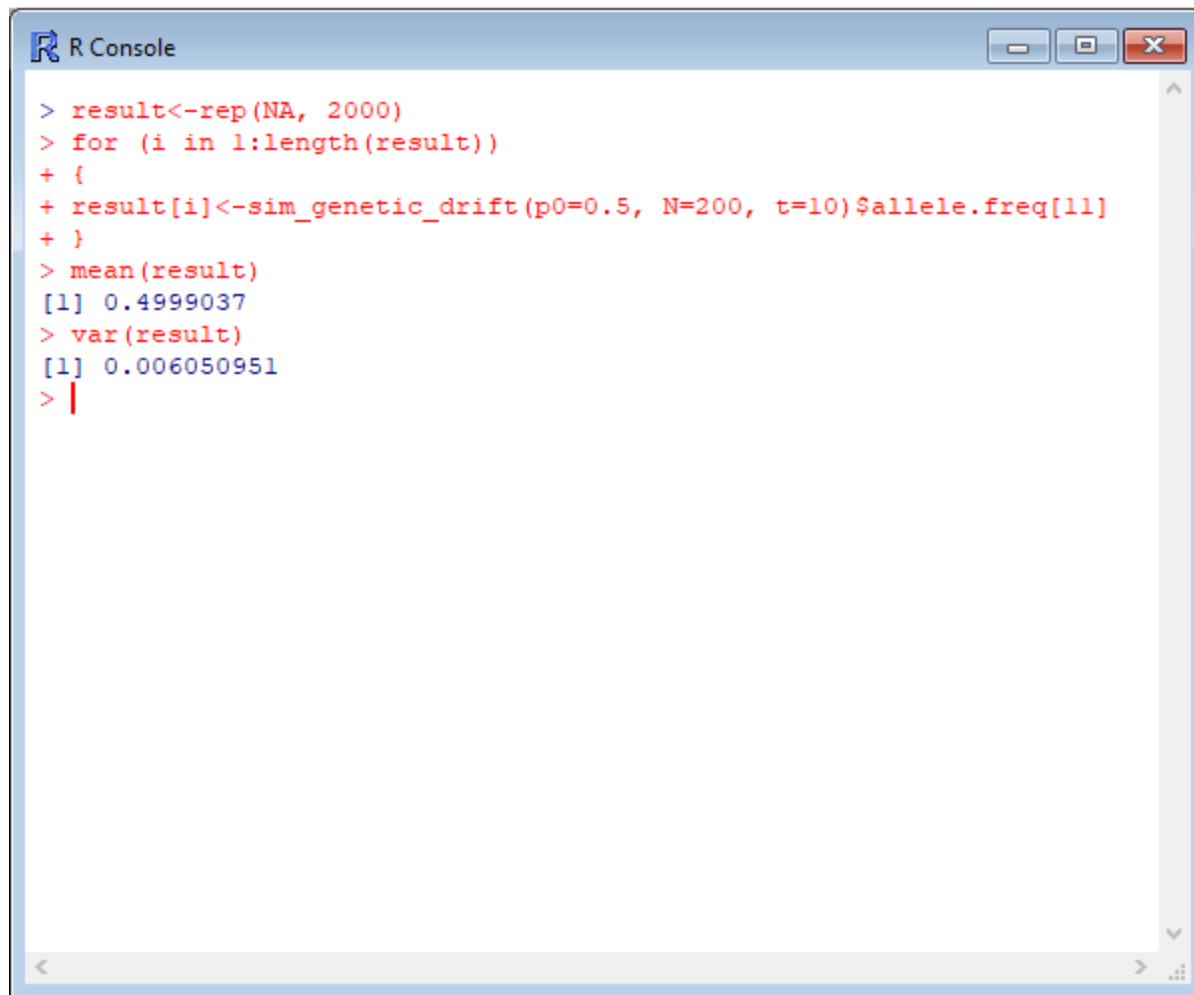


- What is the mean and variance of allele frequency after $t = 10$ generations, given $p_0 = 0.5$, and $N = 200$?

- Method #1
 - Mathematical derivations
 - Work out the distribution of the allele frequency, calculate the mean and variance of the associated random variables
 - Diffusion approximation, Markov matrices and processes etc.
 - ☹️
- Method #2
 - Monte-Carlo methods
 - The average results from repeated simulations
 - 😊

Monte Carlo simulations

- Suppose our aim is to approximate the mean and variance of allele frequency due to genetic drift
 - simulate the WF model with known parameters $p_0 = 0.5$, $N = 200$, and $t = 10$
 - record the allele frequency at the final generation
 - repeat the simulation, independently, for 10,000 times (say) to obtain 10,000 such final allele frequencies
 - calculate the sample mean and variance of these 10,000 final frequencies, and these are your empirical answers



```
> result<-rep(NA, 2000)
> for (i in 1:length(result))
+ {
+ result[i]<-sim_genetic_drift(p0=0.5, N=200, t=10)$allele.freq[11]
+ }
> mean(result)
[1] 0.4999037
> var(result)
[1] 0.006050951
> |
```

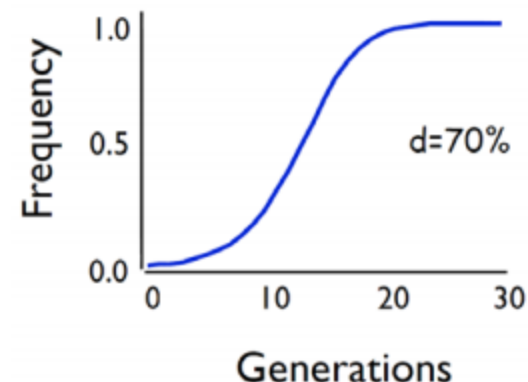
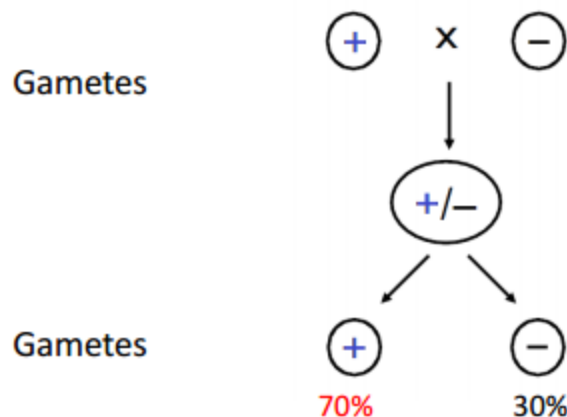
- $Var(p_t) = p_0(1 - p_0)[1 - \left(1 - \frac{1}{2N}\right)^t]$
 - derived by Waples (1989)
 - in our case it is about 0.0061
- The result from MC simulation will converge to the “true answer” if the number of independent simulations $\rightarrow \infty$
 - the stochastic nature means a different answer is expected if you rerun the MC simulation
 - intrinsic variance $\propto 1/(\text{number of simulations})$

- What is the distribution of persistence time of an allele, if $p_0 = 0.05$ and $N = 100$?
 - hint: you may need to modify your WF simulator
 - if an allele goes fixed/extinct, exit the loop
 - `while()`?
 - also calculate the mean persistence time
 - as an exercise!

2. Gene drive simulator

- Similar to the WF model but with “add-ons”
 - drive
 - selection
 - fluctuating population size
- Two types of alleles
 - 0 is the wildtype allele (WT)
 - 1 is the artificially introduced transgene (TG)

- Drive: super-Mendelian inheritance of a gene (the transgene in our case), resulting in unbalanced number of gametes being produced by the heterozygotes
 - an individual carrying 01 heterozygote produces gamete 0 with proportion $(1 - d)$, and gamete 1 with proportion d , $d > 50\%$ is required for TG to drive
 - this allows TG to spread even with a low initial frequency



- Selection:
 - 11 homozygote does not survive till adulthood
 - produces no offspring
- Combining both effects
 - 00 homozygote adults produce WT gamete with probability 1
 - 01 heterozygote adults produce WT gamete with probability $(1 - d)$, and TG gamete with probability d
 - 11 homozygotes produce no gametes

- Let $\{x_{00}, x_{01}, x_{11}\}$ be the number of individuals carrying 00, 01, 11 genotypes at the parental generation
- What are the gametic frequencies for WT and TG?
 - proportion of WT gametes = $\frac{x_{00} * 1 + x_{01} * (1-d)}{x_{00} + x_{01}}$
 - proportion of TG gametes = $\frac{x_{01} * d}{x_{00} + x_{01}}$

- Population dynamics: the population size is no longer constant because of the two forces acting on the it:
 - the lethal 11 homozygote, suppressing the population
 - the lack of intra-species competition when population size is small, reverting it back to the carrying capacity
- For the second point we introduce the **Beverton-Holt** (1957) model to regulate the population size:
 - $N_{t+1} = \frac{R_0 N_t}{1 + N_t/M}$ with two extra parameters
 - discrete version of logistic growth, $(R_0 - 1)M$ is the carrying capacity.
 - “density dependence”

- Initialisation
 - input arguments
 - ...
- Inner functions
 - BH population model
 - calculate genotype counts
- Propagation
 - ...
- Outputs
 - ...



- Tuesday
 - now-15:30 coding time (I am here to help!)
 - 15:30-16:30 Q&A, MC simulation
- Wednesday
 - 10:00-11:00 answers to Problems I to V (Katie)
 - 11:30-12:30 advanced gene drive construct (Katie)

- P.30-36 are some suggestions to help you code the gene drive simulator.
- P.37-41 are the problem sets that need to be solved via MC simulations.
- P.43-46 are some computing notes. In particular, the introduction of `doParallel` package and `foreach()` function.
 - parallelise your MC simulations

(Suggestions)

- Input arguments
 - q_0 : initial TG frequency
 - t
 - d , the transmission rate of the TG allele
 - N_0 : initial population size
 - M and R_0 , two extra parameters for BH model
- Initialisation
 - data types/storages, similar to the drift model
 - also keep track on the population size
 - TG mosquitoes carrying 01 heterozygote are released. Existing wildtype mosquitoes are all 00 homozygotes.

```
sim_gene_drive<-function(q0=0.05, d=0.6, t=10, N0=500, R0=2, M=500)
{
# INITIALISATION
...

# PROPOGATION
...


# OUTPUTS
...
}
```

- Functions can be defined within another function
 - e.g. an inner function for the BH population model
 - e.g. a function to calculate genotype counts
 - be careful of “variable scope”


```

sim_gene_drive<-function(q0=0.05, d=0.6, t=10, N0=500, R0=2, M=500)
{
# INITIALISATION
...

# INNER FUNCTION 1. BH POPULATION MODEL.
bh<-function(N, R0, M)
{return(ceiling(R0*N/(1+N/M)))}
# INNER FUNCTION 2. FUNCTION TO CALCULATE GENOTYPE COUNTS (00, 01, 11).
count_genotype<-function(x)
{
temp<-apply(x, 2, sum)
return(c(sum(temp==0), sum(temp==1), sum(temp==2)))
}

# PROPAGATION
...

# OUTPUTS
...
}

```

- Propagation

- `for(t in ...)`
- calculate new population size
- any early exit clauses? e.g. when population has crashed, or when there is no more TG alleles?
- calculate the WT:TG gametic frequency from the parental generation
- `sample()` new individuals
- ...

```

# PROPAGATION
for (i in ???????)
{
    # CALCULATE THE NEW POPULATION SIZE. ONLY genotype[1]+genotype[2] WILL SURVIVE TILL ADULTHOOD
    ?????????????????????????????????????????????????????????????
    # EARLY EXIT CONDITION 1, IF POPULATION SIZE DROP TO 1
    if (population.size[i+1]<=1)
    {
        ?????????????????????????
    }
    # EARLY EXIT CONDITION 2, IF THERE IS NO MORE TG ALLELE
    if (genotype[2]+genotype[3]==0)
    {
        ?????????????????????????
    }

    ...
    ...
    ...
}
# OUTPUTS
...
}

```

- Outputs
 - return a big list
 - same as before

Problem I (warm up)

- Given $N_0 = 500$, $M = 500$, $R_0 = 2$
- Our first construct has transmission rate $d = 0.6$
- Releasing frequency: $q_0 = 5\%$
- What is the expected TG frequency after $t = 50$ generations?
- Besides, can you detect any population decline?

Problem II (releasing strategy)

- Given $N_0 = 500$, $M = 500$, $R_0 = 2$ (i.e. the same population profile)
- And the same construct with $d = 0.6$
- [Releasing strategy] If the releasing frequency of the transgene is too low then it may not survive or spread. We would like find the minimum releasing frequency to ensure our TG can survive for at least $t = 20$ generations with 98%+ confidence.
 - hint: the survival probability can be approximated by the proportion of runs with have non-zero TG frequency at the end. Repeat this for a range of q_0 , $0.4\% \leq q_0 \leq 2\%$, say

Problem III (construct design)

- Given $N_0 = 500$, $M = 500$, $R_0 = 2$, the same population profile
- Choose a reasonable releasing frequency (based on the previous analysis)
- Our laboratory team is developing a new (and potentially stronger) TG construct
- What is the minimum transmission rate d required if we aim to reduce the mosquito population by 40% in $t = 30$ generations?

Problem IV (targeting different population profiles)

- A new TG with $d = 0.75$ is going to be released in two distinct populations of the same size $N_0 = 500$
 - Population A has $R_0 = 2$ and $M = 500$
 - Population B has $R_0 = 6$ and $M = 100$
- Note that the two populations have the same carrying capacity $(R_0 - 1)M$, but Population B has a much higher intrinsic growth rate
- Evaluate the performance of this construct on these two populations in terms of population suppression. Please focus on the first 30 generations.

Problem V (equilibrium freq)

- Use the same population profile of $N_0 = 500$, $M = 500$, $R_0 = 2$
- (i) show that, given a value of d ($d = 0.6$, say), the final (equilibrium) TG frequency in $t = 50$ generations does not depend on the initial frequency.
- (ii) comment on the equilibrium TG frequency.

Possible extensions include

- Migration with multiple populations
- Seasonal population dynamics
- Heterozygote (dis)-advantages
- The arise of resistant allele
- Drive systems with genetic linkage (multiple loci), multiple drives

- Computationally demanding
 - repeated simulations
 - memory issue when N goes large
- Efficient coding
 - vectorisation in R
 - C/C++? C with R interface?
- Parallel implementation
 - R packages for multicore cpu (`doParallel` etc)
 - College's HPC
 - GPU?

doParallel package

- A built-in R package to replace the generic `for()` by the parallelised `foreach()` loop
 - with some grammatical changes
 - similar to `mclapply()`
- Iterations are sent to different cpu cores and executed simultaneously (but also independently)
 - ideal for MC simulations
- Parallel random number generation is “handled” by the package
 - unlike HPC, where it is your responsibility to make sure different nodes are using different random seeds/states
- Smaller tasks are less likely to be benefited from parallelisation because of overhead
 - performance gain is also problem-specific

```
# TRADITIONAL FOR LOOP
# CALCULATE THE DISTRIBUTION OF THE SUM OF 30 CHISQ(DF=1)
RANDOM VARIABLES
result<-rep(NA, 1000)
for (i in 1:1000)
{
temp<-rchisq(30, df=1)
result[i]<-sum(temp)
}
```

```
# PARALLELISED VERSION doParallel package
# LOAD THE PACKAGE
require(doParallel)
# CREATE A LOCAL "CLUSTER" OBJECT cl. THE NUMBER INSIDE THE BRACKET
MUST NOT EXCEED YOUR CPU CORE COUNTS. REGISTER THE CLUSTER BEFORE USE.
cl<-makeCluster(6)
registerDoParallel(cl)
# foreach
# .combine IS AN ARGUMENT SPECIFYING HOW RESULTS FROM DIFFERENT RUNS
ARE COMBINED. POSSIBLE WAYS ARE c, cbind, rbind. DEFAULT IS A LIST.
# REMEMBER %dopar%
result<-foreach(i=1:1000, .combine='c') %dopar%
{
temp<-rchisq(30, df=1)
return(sum(temp))
}
# STOP THE CLUSTER AFTER USE
stopCluster(cl)
```