The relative frequency (p) of a gene (let’s say gene A in this case) with fitness (w) in a population of size (N) has the approximate dynamics (equation 1, see figure 1 below),

where  is the population mean fitness (w bar) at time (t)

and the random deviates  are drawn from a Normal distribution with mean zero and the standard deviation

It is assumed that the rest of the genes in the population have fitness equal to 1. The focal gene is thus at a fitness advantage if w > 1.

*4a. Ignoring the stochasticity (setting ), show that p\* = 0 and p\* = 1 are the only equilibria of the system (assuming ). (1p)*

I solved question 4a by solving equation 1 mathematically by replacing w bar and by rearranging these combination of equations together.

A white paper with writing on it

Description automatically generated

Figure 1. Answer for question 4a. p\* = 0 is a trivial solution, while p\* = 1 is a nontrivial solution, and the small graph beside the solution indicates p\* = 1 and it is depended on the fitness w, where w is not equals to 1.

*4b. What are the stability criteria for*p\**= 0? (1p)*

I solved question 4b by solving equation 1 mathematically by using the derivative of the quotient rule and by rearranging the equation. To further investigate its stability, second derivative was carried out.

A white paper with black text

Description automatically generated

Figure 2. Answer for question 4b. An equilibrium of a discrete time system depends on the slope of the renewal function at the equilibrium.

It is stable when: 𝑓′′(p∗) <1, where the second derivative has a negative value.

*4c. Now taking the stochasticity into account again, write an R script that runs a simulation of*pt*for 1000 generations and plots the resulting time series. The starting*p*-value should be 1/(2N), corresponding to a single mutant allele of a diploid individual.****Tip:****Break the simulation as soon as*pt*is equal to or below zero. Do the same thing as soon as*pt*is at or above 1. The break command may be useful. (2p)*

I solved question 4c by coding a simulation model that includes a for loop which takes the frequency of p over n generations

R script and answer for question 4c.

**Doc.name: BIOS13\_exam\_Q4c\_HooiMin.R**

The relative frequency (p) of a gene (let’s say gene A in this case) over time = pt

rm(list = ls())

set.seed(123)

n <- 1000 # number of generations

N <- 1000 # population size

pt <- numeric(n) # frequency of gene A over time

w <- 1.1 # fitness of A

pt[1] <- 1 / (2 \* N) # initial p value of gene A

# run simulation of p for n generations

for (i in 1:n) {

# mean fitness of population A over time

w\_hat <- w \* pt[i] + 1 - pt[i]

# standard deviation of fitness of population A over time

sd\_t <- (1 / w\_hat) \* sqrt(w / (2 \* N) \* pt[i] \* (1 - pt[i]))

# calculate p value for next generation

pt[i+1] <- (w / w\_hat) \* pt[i] + rnorm(1, 0, sd = sd\_t)

# Break if pt is <= 0 or >= 1

if (pt[i+1] <= 0 || pt[i+1] >= 1) {

pt[i+1] <- pt[i] # store p value for each generation

pt <- pt[1:i] # remove NA values

break

}

}

# Plot pt over time

plot(pt, type = "l", xlab = "Generation", ylab = "Frequency (pt)", main = "Gene Spread Simulation")

A graph of a number of cells spread simulation

Description automatically generated

Figure 3. Gene spread simulation

*4d. Write a*function*that takes*N*and*w*as input parameters, runs 1000 simulations like the one above and returns the probability of fixation, i.e. the probability that*p*reaches 1 within 1000 generations. (2p)*

Specifically I implemented simulation using a nested for-loop

R script and answer for question 4d.

**Doc.name: BIOS13\_exam\_Q4d\_HooiMin.R**

rm(list = ls())

set.seed(123) # Set seed for reproducibility

# Usage example:

N <- 200 # population size

w <- 1.1 # fitness of allele A

n <- 1000 # number of generations

num\_simulations <- 1000 # number of simulations

# Function to calculate fixation probability

calculate\_fixation\_probability <- function(N, w, n, num\_simulations) {

fixation\_count <- 0 # count number of fixation events

# Run simulation num\_simulations times

for (i in 1:num\_simulations) {

pt <- numeric(n) # frequency of gene A over time

pt[1] <- 1 / (2 \* N) # initial p value of gene A

# Run simulation of p for n generations

for (g in 1:n) {

# mean fitness of population A over time

w\_hat <- w \* pt[g] + 1 - pt[g]

# standard deviation of fitness of population A over time

sd\_t <- (1 / w\_hat) \* sqrt(w / (2 \* N) \* pt[g] \* (1 - pt[g]))

# calculate p value for next generation

pt[g+1] <- (w / w\_hat) \* pt[g] + rnorm(1, 0, sd = sd\_t)

if (pt[g+1] <= 0) { # if p value is less than 0

pt[g+1] <- 0 # set p value to 0

} else if (pt[g+1] >= 1) { # if p value is greater than 1

pt[g+1] <- 1 # set p value to 1

}

}

if (pt[n + 1] == 1) { # if fixation occurs

fixation\_count <- fixation\_count + 1 # add 1 to fixation count

}

}

# calculate fixation probability

fixation\_probability <- fixation\_count / num\_simulations

return(fixation\_probability) # return fixation probability

}

# run simulation

fixation\_prob <- calculate\_fixation\_probability(N, w, n, num\_simulations)

# print fixation probability

print(paste("Probability of fixation:", fixation\_prob))

**"Probability of fixation: 0.26"**

*4e. Write a script that uses the function in d) and plots the probability of fixation for a range of*w*-values from 0.9 to 1.1. Do the plot for*N*= 20 and*N*= 200. (You should be able to see that small populations are more likely to accumulate deleterious mutations than large ones.) (1p)*

To speed up the stimulations, here I run 100 of stimulations, but one can increase it to 1000, which takes 1-2 minutes.

R script and answer for question 4e.

**Doc.name: BIOS13\_exam\_Q4e\_HooiMin.R**

rm(list = ls())

# Define parameters

n <- 1000 # number of generations

num\_simulations <- 100 # number of simulations

# define function to calculate fixation probability

calculate\_fixation\_probability <- function(N, w, n, num\_simulations) {

set.seed(123) # Set seed for reproducibility

fixation\_count <- 0 # count number of fixation events

# run simulation for num\_simulations times

for (i in 1:num\_simulations) {

# frequency of gene A over time

pt <- numeric(n)

# initial p value of gene A

pt[1] <- 1 / (2 \* N)

# run simulation of p for n generations

for (g in 1:n) {

# mean fitness of population A over time

w\_hat <- w \* pt[g] + 1 - pt[g]

# standard deviation of fitness of population A over time

sd\_t <- (1 / w\_hat) \* sqrt(w / (2 \* N) \* pt[g] \* (1 - pt[g]))

# calculate p value for next generation

pt[g+1] <- (w / w\_hat) \* pt[g] + rnorm(1, 0, sd = sd\_t)

if (pt[g+1] <= 0) { # if p value is less than 0

pt[g+1] <- 0 # set p value to 0

} else if (pt[g+1] >= 1) { # if p value is greater than 1

pt[g+1] <- 1 # set p value to 1

}

}

if (pt[n + 1] == 1) { # if fixation occurs

fixation\_count <- fixation\_count + 1 # add 1 to fixation count

}

}

# calculate fixation probability

fixation\_probability <- fixation\_count / num\_simulations

return(fixation\_probability) # return fixation probability

}

# Range of w-values

w\_values <- seq(0.9, 1.1, by = 0.01)

# Initialize vectors to store probabilities for N = 20 and N = 200

prob\_N\_20 <- numeric(length(w\_values))

prob\_N\_200 <- numeric(length(w\_values))

# Loop through each w-value

for (i in 1:length(w\_values)) {

w <- w\_values[i] # set w-value

# Calculate fixation probability for N = 20 and N = 200

prob\_N\_20[i] <- calculate\_fixation\_probability(20, w, n, num\_simulations)

prob\_N\_200[i] <- calculate\_fixation\_probability(200, w, n, num\_simulations)

}

# Plotting the results

plot(w\_values, prob\_N\_20, type = "l", col = "blue", xlab = "w-values",

ylab = "Fixation Probability", ylim = c(0, 1),

main = "Fixation Probability vs. w-values for Different Population Sizes")

lines(w\_values, prob\_N\_200, type = "l", col = "red")

legend("topright", legend = c("N = 20", "N = 200"),

col = c("blue", "red"), lty = 1)

# Optional: Add a horizontal line at y = 0.5 for reference

abline(h = 0.5, lty = 2, col = "black")

A graph with lines and numbers

Description automatically generated

Figure 4. Fixation probability vs fitness values (w)