# Evolutionary Dynamics Homework 9

Minghang Li

November 28, 2024 09:11 (Europe/Berlin)

## Problem 1: Eden model dynamics

Consider a two-dimensional Eden growth model with von Neumann neighborhood

### (b) Simulate different update rules

Consider the R code at https://git.io/vF0KH. What is the update implemented in the code? Modify the code to simulate the other two update rules from question (a) and then explain the difference between the spatial boundareis of the pouplation obtained for the different evolutionary dynamics.

#### Update rule implemented in the code

The update rule code is implemented in the following block:

```r
candidate <- sample(1:num_has_space,
                    1,
                    prob = unlist(how_many_spaces) * sapply(has_space, function(e)
                      sites[e[1], e[2]]))
```

It basically selects a cell that will divide in this iteration, with a probability that depends on both the number of empty spaces in the neighbourhood and the fitness of the cell itself. This implementation is most similar to the *bond focused* update rule in question (a), where we randomly choose an $S_1$ (occupied) site with probability proportional to the number of adjoining $S_0$ (empty) sites.

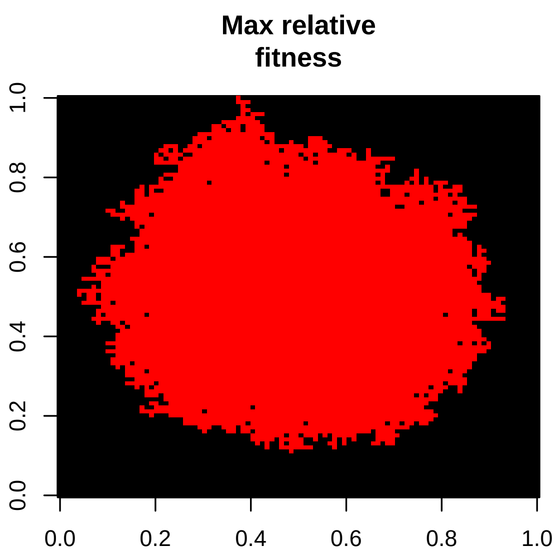
#### Simulate the other two update rules

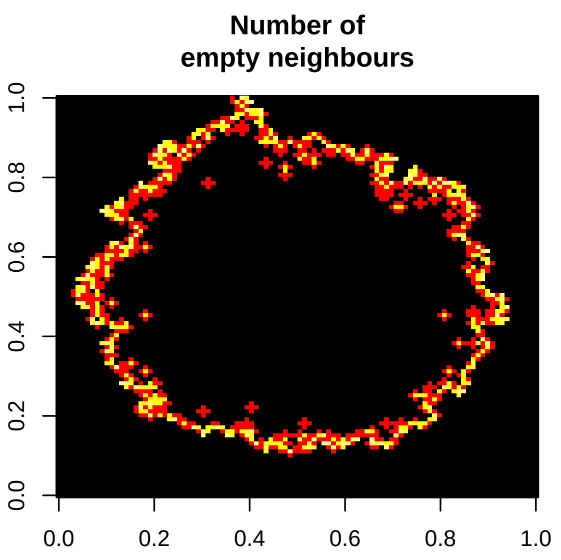
#### Available site-focused

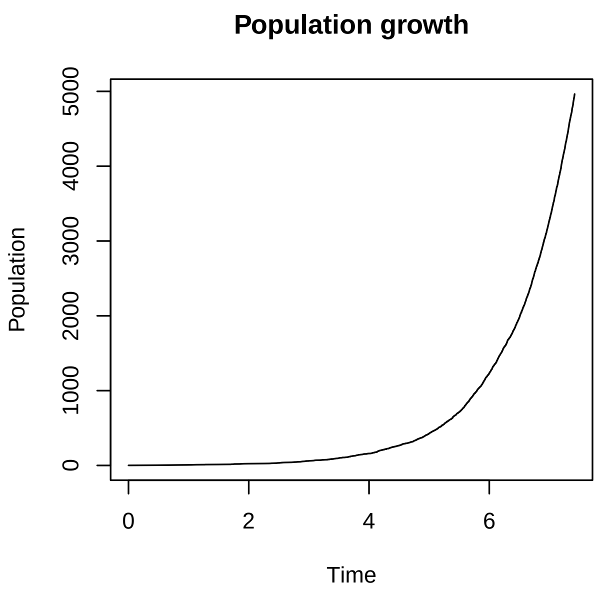
- *Available site-focused* update rule: randomly choose an $S_0$ that adjoins at least one $S_1$ site, and switch it from $S_0$ to $S_1$.

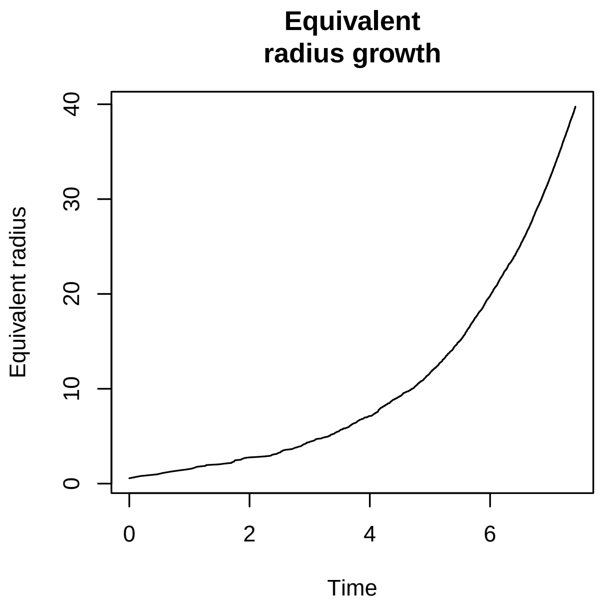
```r
####################################
####### initialization:
####################################

# Initialize site states:
sites <- matrix(0, nrow = grid_width, ncol = grid_width)
num_occupied <- length(occupied)
for (i in 1:num_occupied)
  sites[occupied[[i]][1], occupied[[i]][2]] <- 1

# neighbourhood size:
nhood_size <- length(nhood(1, 1, nhood_type))
```

```r
# Initialize list of empty sites adjacent to occupied ones:
num_empty_adj <- 0
empty_adj <- list()
index_map <- matrix(NA, nrow = grid_width, ncol = grid_width)

# Find initial empty sites adjacent to occupied ones:
for (x in 2:(grid_width - 1))
  for (y in 2:(grid_width - 1)) {
    if (sites[x, y] < 1) {
      # if site is empty
      neighbours <- nhood(x, y, nhood_type)
      if (any(sapply(neighbours, function(n)
        sites[n[1], n[2]] > 0))) {
        # has occupied neighbor
        num_empty_adj <- num_empty_adj + 1
        empty_adj[[num_empty_adj]] <- c(x, y)
        index_map[x, y] <- num_empty_adj
      }
    }
  }

# Initialize timer and output:
timer <- 0
output_df <- data.frame(Time = timer, Population = num_occupied)

####################################
####### run the model:
####################################

for (iter in 1:max_iter) {
  if (num_empty_adj == 0)
    break

  # Pick random empty site adjacent to occupied:
  candidate_idx <- sample(num_empty_adj, 1)
  new_site <- empty_adj[[candidate_idx]]

  # Switch from empty to occupied:
  sites[new_site[1], new_site[2]] <- 1
  num_occupied <- num_occupied + 1

  # Remove chosen site from empty_adj list:
  empty_adj[[candidate_idx]] <- empty_adj[[num_empty_adj]]
  index_map[empty_adj[[num_empty_adj]][1], empty_adj[[num_empty_adj]][2]] <- candidate_idx
  empty_adj[[num_empty_adj]] <- NULL
  index_map[new_site[1], new_site[2]] <- NA
  num_empty_adj <- num_empty_adj - 1

  # Add new empty neighbors to empty_adj list:
  neighbours <- nhood(new_site[1], new_site[2], nhood_type)
  for (n in neighbours) {
    if (n[1] >= 1 &&
        n[1] <= grid_width && n[2] >= 1 && n[2] <= grid_width &&
        sites[n[1], n[2]] < 1 && is.na(index_map[n[1], n[2]])) {
      num_empty_adj <- num_empty_adj + 1
      empty_adj[[num_empty_adj]] <- n
      index_map[n[1], n[2]] <- num_empty_adj
    }
  }

  # Update time (using simpler rate based on population):
  timer <- timer + rexp(1, num_occupied)

  # Record time and population:
  output_df <- rbind(output_df, c(timer, num_occupied))

  # Check boundary:
  if (new_site[1] >= grid_width || new_site[1] <= 1 ||
      new_site[2] >= grid_width || new_site[2] <= 1)
    break
}
```







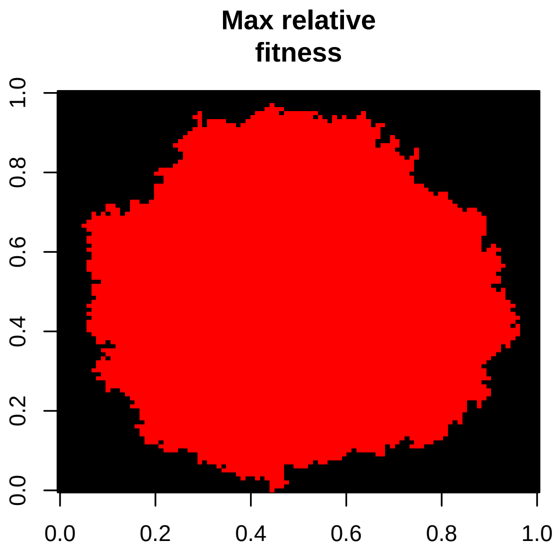


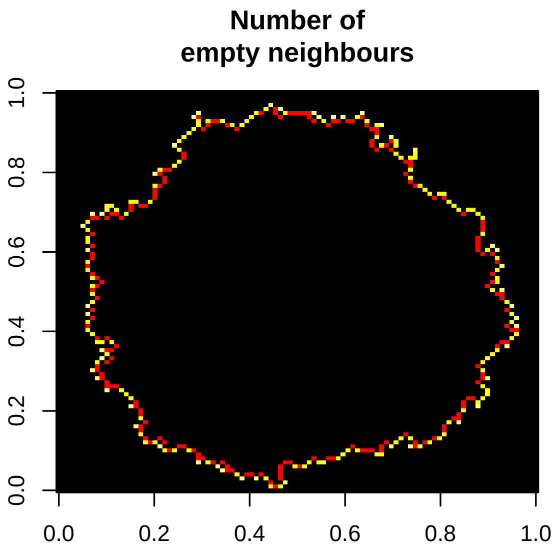
**Cell-focused**

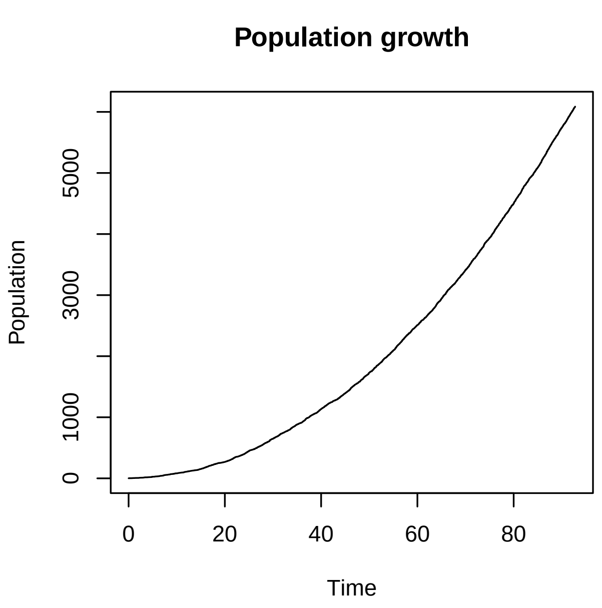
- *Cell-focused* update rule: randomly choose an $S_1$ site that adjoins at least one $S_0$ site, then sample uniformly an $S_0$ neighbour and switch it to $S_1$.

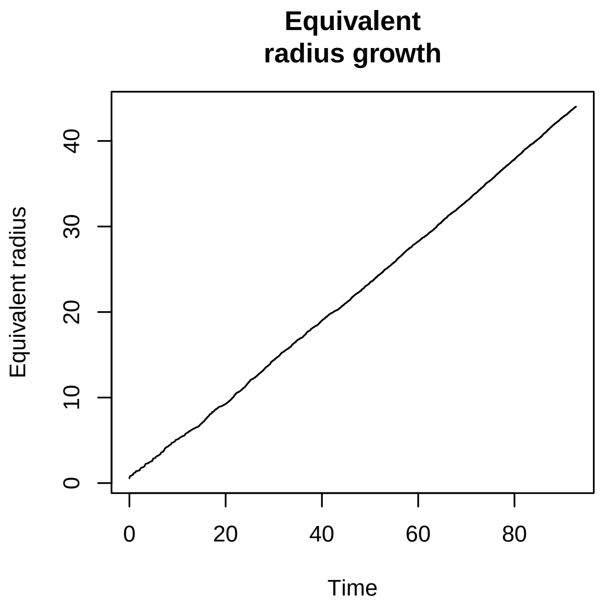
Basically just remove the scaling using `how_many_spaces` in the `sample` function.

```
candidate <- sample(1:num_has_space, 1, prob = sapply(has_space, function(e)
  sites[e[1], e[2]]))
```









**Analysis**

Available site-focused update rule grows the fastest to grid edge, but creates spiky, irregular boundaries and shows holes within occupied areas. It suggests that random selection of empty sites leads to uneven growth. Cell-focused update rule has rather slow growth rate, creates smooth, compact patterns.

## Problem 2: Diffusion approximation of a spatial Moran model

Consider the spatial Moran model for a mutation spreading through an infinite row of demes. Let $\mu$ denote the death rate, $s$ the fitness advantage of the mutant, $m$ the dispersion probability, and $N$ the number of individuals per deme. Assume initially that $n_i = N, \forall i \leq 0$ and $n_i = 0$ otherwise, where $n_i$ is the number of mutants in deme i. The rate at which mutants in a deme die is $\mu n_i$. Once an individual has died, it is replaced by the offspring of a wild type individual of its own deme with probability

$$P_1 = \frac{N-n_i}{(1+s)n_i + (N-n_i)} \approx \frac{N-n_i}{N},$$

and it is replaced by the offspring of a wild type individual from any of the neighbouring demes with probability

$$P_2 = \frac{m(N-n_{i-1})}{2N} + \frac{m(N-n_{i+1})}{2N} \approx \frac{m(2N-(n_{i-1}+n_{i+1}))}{2N}.$$

**(a) Using these results, show that the rate for the number of mutants in deme $i$ decreasing by one individual is as follows:**

$$W_i^-(\mathbf{n}) = \frac{\mu}{N} n_i \left( (N-n_i) - \frac{m}{2} n_i'' \right)$$

where $n_i'' = n_{i-1} + n_{i+1} - 2n_i$.

---

*Proof.* $W_i^-(\mathbf{n})$ is basically $\mu n_i((1-m)P_1 + P_2)$.

$$\begin{aligned}
W_i^-(\mathbf{n}) &= \mu n_i \left( (1-m) \cdot \frac{N-n_i}{N} + \frac{m(2N-(n_{i-1}+n_{i+1}))}{2N} \right) \\
&= \frac{\mu n_i}{N} \left( N - n_i - m(N-n_i) + \frac{m(2N-(n_{i-1}+n_{i+1}))}{2} \right) \\
&= \frac{\mu n_i}{N} \left( N - n_i - m\frac{\cancel{2N} - 2n_i - \cancel{2N} + n_{i-1} + n_{i+1}}{2} \right) \\
&= \frac{\mu n_i}{N} \left( N - n_i - \frac{m}{2}(n_{i-1} + n_{i+1} - 2n_i) \right) \\
&= \frac{\mu}{N} n_i \left( (N-n_i) - \frac{m}{2} n_i'' \right)
\end{aligned}$$

□

\end{proof}

**(b) Show that the rate of expectation of the number of mutants in deme $i$ is as follows:**

In general,

$$\frac{d\mathbb{E}[n_i]}{dt} = \mathbb{E}\left[W_i^+(\mathbf{n}) - W_i^-(\mathbf{n})\right]$$

where $\mathbb{E}[X]$ denotes the expected value of a random variable $X$. Using the expression of $W_i^+(\mathbf{n})$ given in the lecture and applying the approximation $\mathbb{E}[n_i n_k] \approx \mathbb{E}[n_i]\mathbb{E}[n_k]$ (called the *mean-field approximation*), show that

$$\frac{d\mathbb{E}[n_i]}{dt} = \frac{\mu m}{2}\mathbb{E}[n_i''] + \frac{s\mu}{N}(N - \mathbb{E}[n_i]) \left( \mathbb{E}[n_i] + \frac{m}{2}\mathbb{E}[n_i''] \right)$$

---

*Proof.* From lecture, we know that

$$W_i^+(\mathbf{n}) = \frac{\mu(1+s)}{N}(N-n_i)\left[n_i + \frac{m}{2}n_i''\right].$$

Then we have

$$\begin{aligned}
\frac{d\mathbb{E}[n_i]}{dt} &= \mathbb{E}\left[W_i^+(\mathbf{n}) - W_i^-(\mathbf{n})\right] \\
&= \mathbb{E}\left[\frac{\mu(1+s)}{N}(N-n_i)\left[n_i + \frac{m}{2}n_i''\right] - \frac{\mu}{N}n_i\left((N-n_i) - \frac{m}{2}n_i''\right)\right] \\
&\text{(Denote } \frac{m}{2}n_i'' \text{ as } M \text{ for succinctness)} \\
&= \mathbb{E}\left[\frac{\mu(1+s)}{N}(N-n_i)\left[n_i + M\right] - \frac{\mu}{N}n_i(N-n_i-M)\right] \\
&= \mathbb{E}\left[\frac{s\mu}{N}(N-n_i)(n_i+M)\right] + \mathbb{E}\left[\frac{\mu}{N}\left[(N-n_i)(n_i+M) - n_i(N-n_i-M)\right]\right] \\
&= \mathbb{E}\left[\frac{s\mu}{N}(N-n_i)(n_i+M)\right] + \mathbb{E}\left[\frac{\mu}{N}\left[(\cancel{Nn_i} + NM - \cancel{n_i^2} - \cancel{n_iM} - \cancel{n_iN} + \cancel{n_i^2} + \cancel{n_iM})\right]\right] \\
&\text{(Change } M \text{ back to } \frac{m}{2}n_i''\text{)} \\
&= \frac{s\mu}{N}\mathbb{E}\left[(N-n_i)\left(n_i + \frac{m}{2}n_i''\right)\right] + \mu\mathbb{E}\left[\frac{m}{2}n_i''\right] \\
&= \frac{s\mu}{N}(N-\mathbb{E}[n_i])\left(\mathbb{E}[n_i] + \frac{m}{2}\mathbb{E}[n_i'']\right) + \frac{\mu m}{2}\mathbb{E}[n_i'']
\end{aligned}$$

□

## (c) Show that the process can be approximate by diffusion equation.

We can approximate distance along the row of demes using the continuous variable $x = li$, where $l$ is the deme width. Setting $u(x) = \mathbb{E}[n_i]/N$, use the previous approximation to show that process proved in (b) can be approximated by the diffusion equation

$$\frac{\partial u}{\partial t} = D[1+s(1-u)]\frac{\partial^2 u}{\partial x^2} + \mu s u(1-u)$$

where diffusion coefficient $D = uml^2/2$.

*Note*: we assume that $\partial u(x)/\partial i$ is the expected difference in $u$ between the $i$th and $(i+i)$th deme and that $\partial^2 u(x)/\partial i^2$ is the expected difference in $\partial u(x)/\partial i$ between the $i$th and $(i+i)$th deme. Compute $\partial u(x)/\partial i^2$ and perform a change of variables.

---

*Proof.*

$$\begin{aligned}
\frac{d\mathbb{E}[n_i]}{dt} &= \frac{s\mu}{N}(N-\mathbb{E}[n_i])\left(\mathbb{E}[n_i] + \frac{m}{2}\mathbb{E}[n_i'']\right) + \frac{\mu m}{2}\mathbb{E}[n_i''] \\
\frac{1}{N}\frac{d\mathbb{E}[n_i]}{dt} &= s\mu\left(1 - \frac{\mathbb{E}[n_i]}{N}\right)\left(\frac{\mathbb{E}[n_i]}{N} + \frac{m}{2}\frac{\mathbb{E}[n_i'']}{N}\right) + \frac{\mu m}{2}\frac{\mathbb{E}[n_i'']}{N} \\
\frac{\partial u}{\partial t} &= s\mu(1-u)\left(u + \frac{m}{2}\frac{\mathbb{E}[n_i'']}{N}\right) + \frac{\mu m}{2}\frac{\mathbb{E}[n_i'']}{N} \\
\frac{\partial u}{\partial t} &= \mu s u(1-u) + \frac{\mu m}{2}\frac{\mathbb{E}[n_i'']}{N}[1+s(1-u)]
\end{aligned}$$

Now we just need to tidy $\mathbb{E}[n_i'']/N$ up!

$$\begin{aligned}\frac{\mathbb{E}[\mathbb{n}''_{\mathbb{i}}]}{N} &= \frac{\mathbb{E}[n_{i+1}-n_i]}{N} - \frac{\mathbb{E}[n_i - n_{i-1}]}{N} \\ &= \mathbb{E}[u_{i+1}-u_i] - \mathbb{E}[u_i - u_{i-1}] \\ &= \frac{\partial u}{\partial i} - \frac{\partial u}{\partial i - 1} \\ &= \frac{\partial^2 u}{\partial i^2}\end{aligned}$$

Change of variables:

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial i^2}\frac{\partial^2 i}{\partial x^2} = \frac{\partial^2 u}{\partial i^2}\frac{1}{l^2}$$

Combine everthing we have:

$$\begin{aligned}\frac{\partial u}{\partial t} &= \mu s u(1-u) + \frac{\mu m}{2}\frac{\mathbb{E}[n''_i]}{N}[1+s(1-u)] \\ &= \mu s u(1-u) + \frac{\mu m l^2}{2}\frac{\partial^2 u}{\partial x^2}[1+s(1-u)]\end{aligned}$$

□

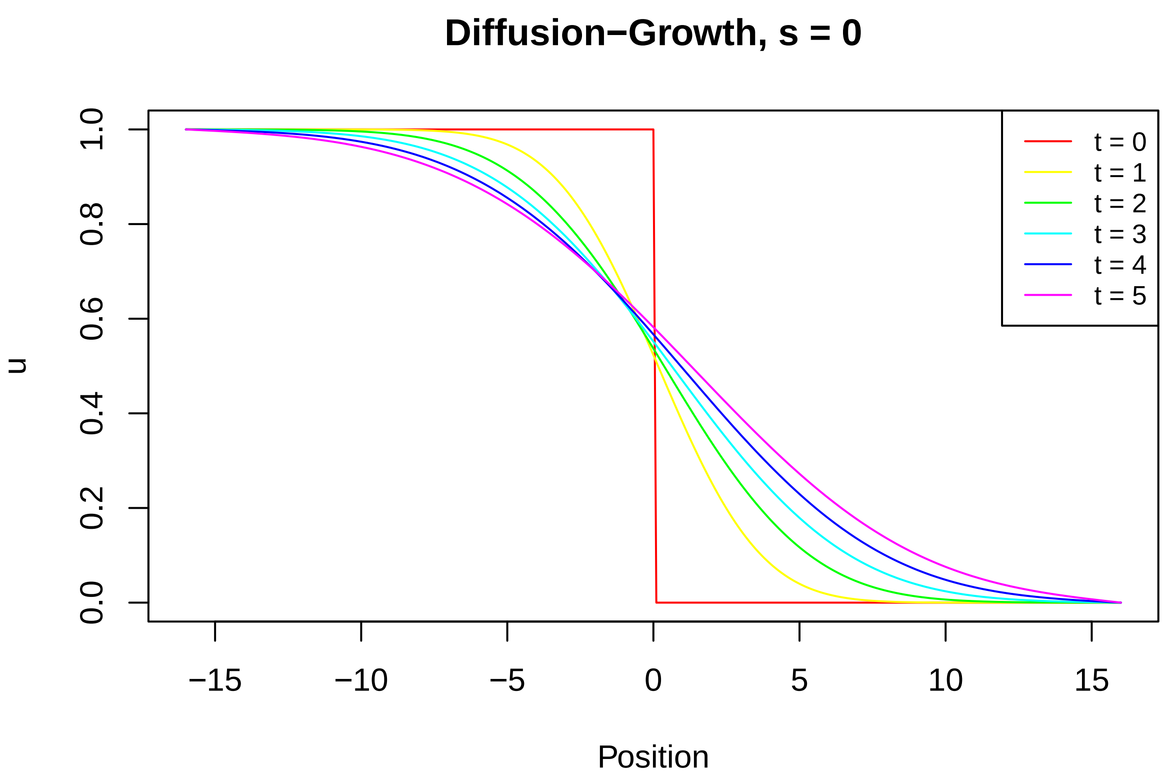
## (d) See below.

```r
# Parameters
dx <- 0.1  # Space step
dt <- 0.001  # Time step
L <- 16  # Domain length [-L, L]
T <- 5  # Total time
mu <- 0.1  # death rate
m <- 0.3 # Dispersion probability
s <- 0  # Fitness advantage
```

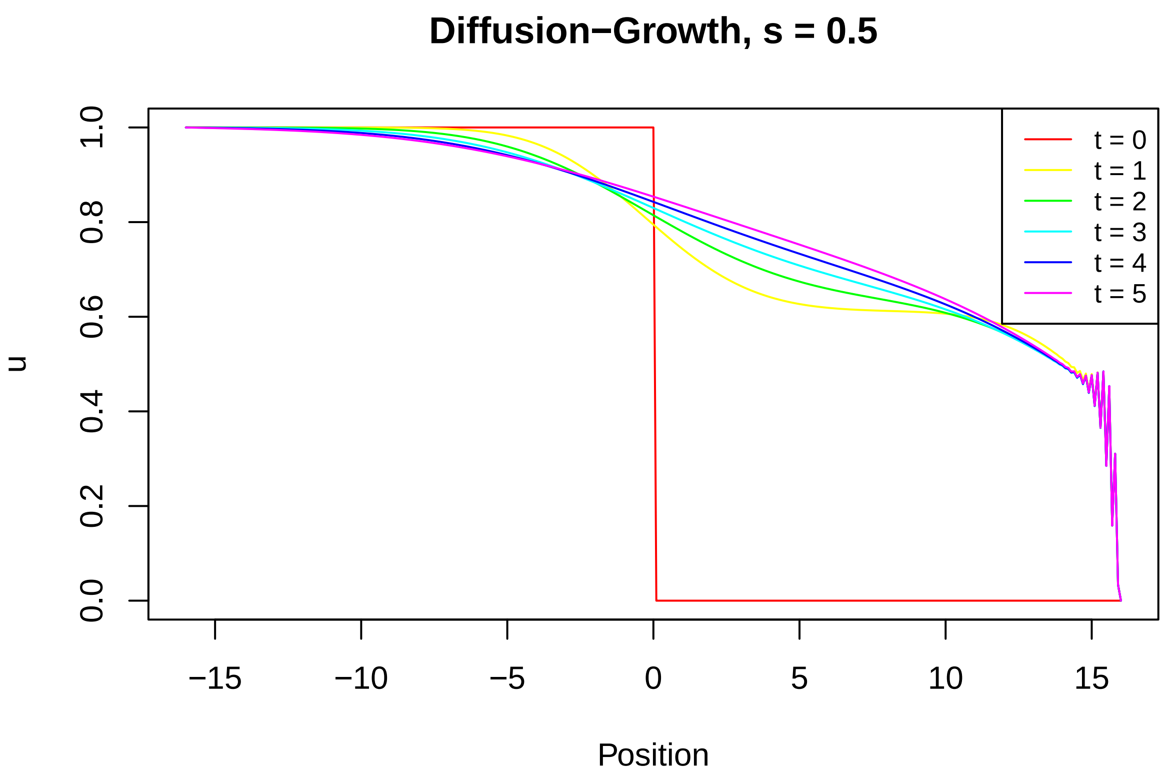
```r
simulate <- function(
  dx=0.1, dt=0.001, L=2, T=1, mu=1, m=0.2, s=0
) {
  x <- seq(-L, L, by=dx)
  t <- seq(0, T, by=dt)
  nx <- length(x)
  nt <- length(t)
  D <- mu*m*L^2/2  # Diffusion coefficient

  u <- matrix(0, nrow=nx, ncol=nt)
  u[x <= 0, 1] <- 1  # Set u=1 (meaning n=N) for x 0
  u[x > 0, 1] <- 0   # Set u=0 (meaning n=0) for x>0

  for(j in 1:(nt-1)) {
    for(i in 2:(nx-1)) {
      d2u_dx2 <- (u[i+1,j] - 2*u[i,j] + u[i-1,j])/dx^2

      u[i,j+1] <- u[i,j] + dt * (
        D * (1 + s*(1-u[i,j])) * d2u_dx2 +
        mu * u[i,j] * (1-u[i,j])
      )

      u[i,j+1] <- max(0, min(1, u[i,j+1]))
    }

    # Boundary conditions (Dirichlet)
    u[1,j+1] <- u[1,j]
    u[nx,j+1] <- u[nx,j]
  }
  return(
    list(
      x=x, t=t, u=u
    )
  )
}
```

```r
plot_diffusion <- function(x, t, u, times_to_plot = c(0, 0.2, 0.5, 1), s=0) {
  # Plot results at different time points

  colors <- rainbow(length(times_to_plot))
  plot(
    x,
    u[, 1],
    type = 'l',
    ylim = c(0, 1),
    col = colors[1],
    xlab = 'Position',
    ylab = 'u',
    main = paste('Diffusion-Growth, s =', s)
  )

  for (i in 2:length(times_to_plot)) {
    j <- which.min(abs(t - times_to_plot[i]))
    lines(x, u[, j], col = colors[i])
  }

  legend(
    'topright',
    legend = paste('t =', times_to_plot),
    col = colors,
    lty = 1,
    cex = 0.8
  )
}
```

```r
res <- simulate(dx, dt, L, T, mu, m, s)
plot_diffusion(res$x, res$t, res$u, seq(0, T), s=0)
```



```r
res <- simulate(dx, dt, L, T, mu, m, s=0.5)
plot_diffusion(res$x, res$t, res$u, seq(0, T), s=0.5)
```



In the case $s = 0$ the shape of the function will remain unchanged, while for $s > 0$, there will be a shift towards the right, like a wave. The biology intuition is that the mutant with more fitness will expand over neighbouring demes with a speed proportional to its selective advantage and the diffusion constant.

## (e) See below.

$D = \mu m l^2/2$ is the diffusion coefficient, where:

$m$ is the dispersal probability that:

- Controls how easily individuals move between demes
- Higher $m$ means faster spatial spread

$\mu$ is the death rate:

- Affects both diffusion speed and population growth
- Appears in the growth term $\mu s u(1-u)$

$s$ as the selective advantage:

- Measures fitness advantage of mutants
- When $s = 0$, only random diffusion occurs
- When $s > 0$, adds directional movement to the wave