1D-Continuous Dynamical Systems (Assignment Sheet 1)

Introduction To Chaos Applied To Systems, Processes And Products (ETSIDI, UPM)

Alfonso Allen-Perkins, Juan Carlos Bueno and Eduardo Faleiro

2026-02-19

Contents

1. The logistic growth model

The logistic growth model describes how a population grows when there are limited resources. A common form of the logistic growth model is

$\frac{dN}{dt} = \dot{N} = rN \cdot \left(1 - \frac{N}{K}\right)$.

Here, $N(t)$ represents the population size at time t. The parameter $r > 0$ is the growth rate, which controls how fast the population increases when it is small. The parameter $K > 0$ is the carrying capacity, the maximum population size that the environment can support. When N is much smaller than K, the population grows almost exponentially. As N approaches K, growth slows down and eventually stops.

Numeric solutions of the logistic growth model

We want to solve the differential equation of the logistic growth model, but instead of solving it by hand, we approximate $N(t)$ numerically on a grid of time points. In R, the `deSolve` package does this for us: we define a function that returns $\dot{N}$, choose parameter values (r, K), pick an initial population $N(0) = N0$, and then call `ode()` to compute the solution over time.

Step 1.1: Load packages

```
library(deSolve)
library(ggplot2)
```

Step 1.2: Define the logistic ODE

We code the equation

$\frac{dN}{dt} = rN \cdot \left(1 - \frac{N}{K}\right)$

as a function that returns dN/dt.

```
logistic_ode <- function(t, state, parameters) {

  # Population size
  N <- state["N"]

  # Parameters
  r <- parameters["r"]
  K <- parameters["K"]

  # Logistic growth equation
  dN <- r * N * (1 - N / K)

  # Return the derivative
  list(c(dN))
}
```

Step 1.3: Choose parameters and initial condition

```
# Growth rate
r <- 0.4

# Carrying capacity
K <- 100

# Initial population size
N0 <- 10
```

Step 1.4: Define the time interval

```
# Time points where the solution is computed
times <- seq(from = 0, to = 30, by = 0.1)
```

Step 1.5: Run the numerical integration

`ode()` returns a matrix-like object; we convert it to a data frame.

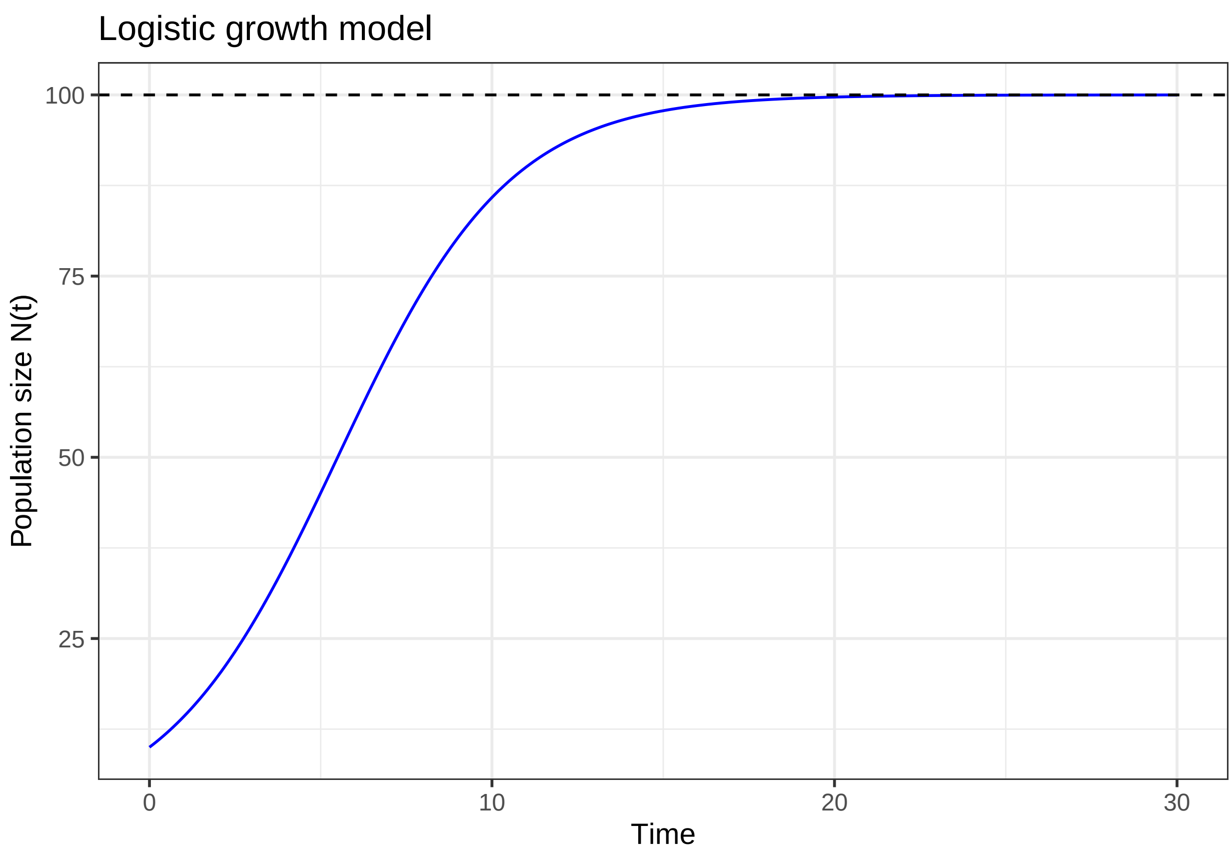
```
# Numerical integration of the logistic equation
solution <- ode(
  y = c(N = N0),
  times = times,
  func = logistic_ode,
  parms = c(r = r, K = K)
```

```
)

# Convert output to a data frame
solution <- as.data.frame(solution)

# Show the first few rows
head(solution)
```

```
##   time        N
## 1  0.0 10.00000
## 2  0.1 10.36581
## 3  0.2 10.74340
## 4  0.3 11.13303
## 5  0.4 11.53497
## 6  0.5 11.94947
```

Step 1.6: Plot $N(t)$ and show the carrying capacity K

```
# Create a plot using the data frame for the logistic model
ggplot(data = solution, aes(x = time, y = N)) +

  # Add a line representing N(t)
  geom_line(color = "blue") +

  # Add a horizontal line for the carrying capacity K
  geom_hline(yintercept = K, linetype = "dashed") +

  # Label the x-axis
  xlab("Time") +

  # Label the y-axis
  ylab("Population size N(t)") +

  # Add a title to the plot
  ggtitle("Logistic growth model") +

  # Use a clean black-and-white theme
  theme_bw()
```



Step 1.7: Compare different initial conditions (same r, K)

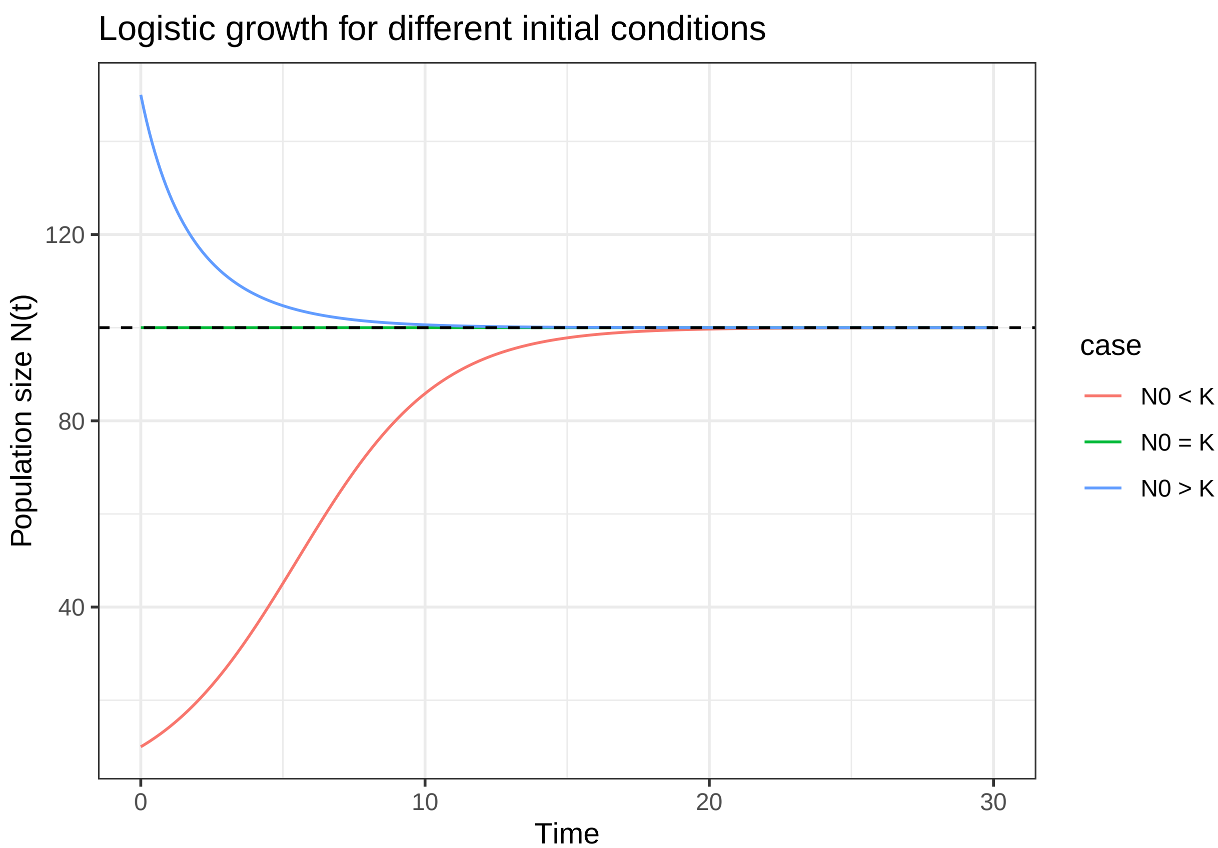
Here we keep r and K fixed, but try $N_0 < K$, $N_0 = K$, and $N_0 > K$.

```
# Three initial conditions
N0_1 <- 10     # N0 < K
N0_2 <- K      # N0 = K
N0_3 <- 150    # N0 > K

# Solve the model for N0 = 10
solution1 <- ode(
  y = c(N = N0_1),
  times = times,
  func = logistic_ode,
  parms = c(r = r, K = K)
)
solution1 <- as.data.frame(solution1)

# Solve the model for N0 = K
solution2 <- ode(
  y = c(N = N0_2),
  times = times,
  func = logistic_ode,
  parms = c(r = r, K = K)
)
```

```r
solution2 <- as.data.frame(solution2)

# Solve the model for N0 = 150
solution3 <- ode(
  y = c(N = N0_3),
  times = times,
  func = logistic_ode,
  parms = c(r = r, K = K)
)
solution3 <- as.data.frame(solution3)
```

Step 1.8: Plot both solutions together

```r
# Add a label so we can tell the curves apart
solution1$case <- "N0 < K"
solution2$case <- "N0 = K"
solution3$case <- "N0 > K"

# Combine all solutions into one data frame
all_solutions <- rbind(solution1, solution2, solution3)

# Create the plot
ggplot(data = all_solutions, aes(x = time, y = N, color = case)) +

  # Draw one line for each initial condition
  geom_line() +

  # Show the carrying capacity K
  geom_hline(yintercept = K, linetype = "dashed") +

  # Label the axes
  xlab("Time") +
  ylab("Population size N(t)") +

  # Title
  ggtitle("Logistic growth for different initial conditions") +

  # Clean theme
  theme_bw()
```



What to notice:

- If $N_0 < K$, the solution increases toward K.
- If $N_0 > K$, it decreases toward K.
- If $N_0 = K$, it stays at K.

Step 1.9: Compare different growth rates r (same N_0, K)

Now we keep N_0 and K fixed, and change r.

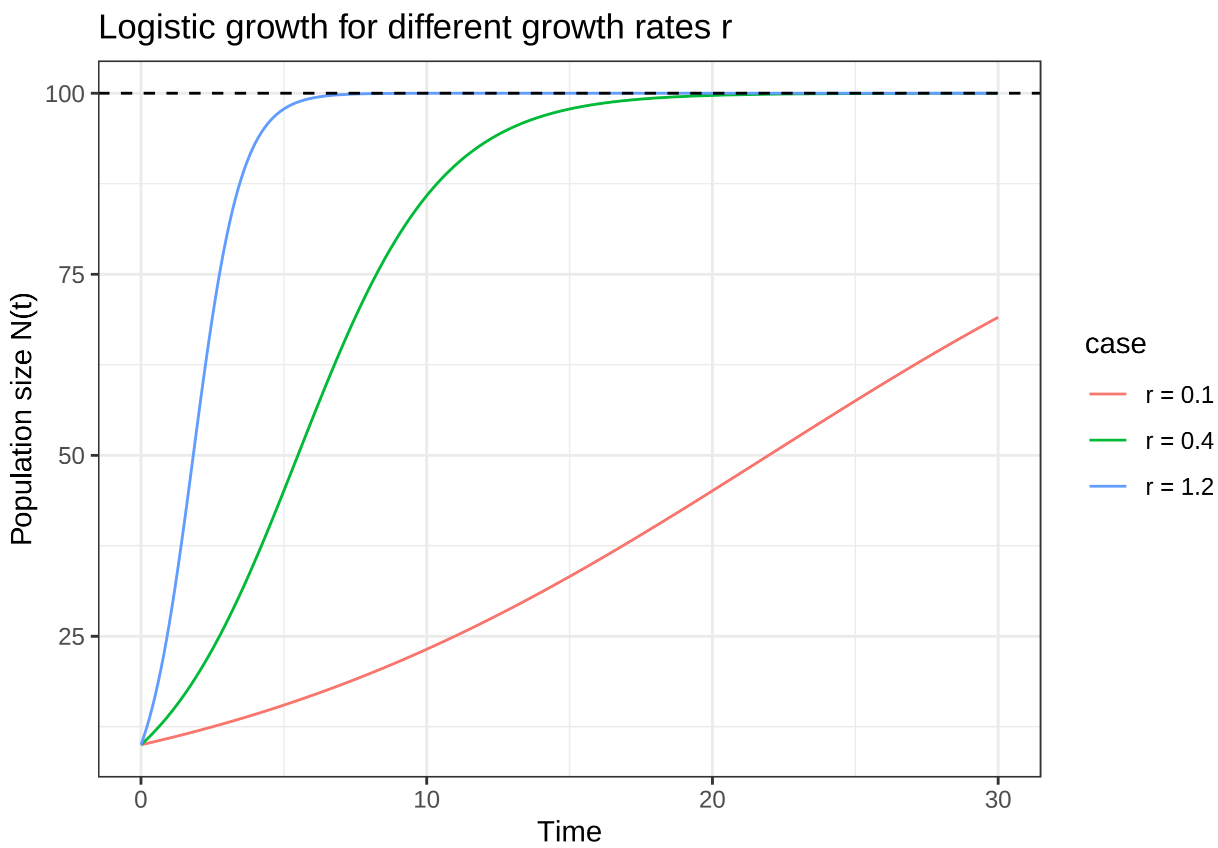
```
# Fix the initial condition and carrying capacity
N0_fixed <- 10
K_fixed  <- K

# Three different growth rates
r1 <- 0.1
r2 <- 0.4
r3 <- 1.2

# Solve the model for r = 0.1
solution_r1 <- ode(
```

```r
  y = c(N = N0_fixed),
  times = times,
  func = logistic_ode,
  parms = c(r = r1, K = K_fixed)
)
solution_r1 <- as.data.frame(solution_r1)

# Solve the model for r = 0.4
solution_r2 <- ode(
  y = c(N = N0_fixed),
  times = times,
  func = logistic_ode,
  parms = c(r = r2, K = K_fixed)
)
solution_r2 <- as.data.frame(solution_r2)

# Solve the model for r = 1.2
solution_r3 <- ode(
  y = c(N = N0_fixed),
  times = times,
  func = logistic_ode,
  parms = c(r = r3, K = K_fixed)
)
solution_r3 <- as.data.frame(solution_r3)
```

Step 1.10: Plot the solutions for different growth rates

```r
# Add labels so we can tell the curves apart
solution_r1$case <- "r = 0.1"
solution_r2$case <- "r = 0.4"
solution_r3$case <- "r = 1.2"

# Combine all solutions into one data frame
all_r_solutions <- rbind(solution_r1, solution_r2, solution_r3)

# Create the plot
ggplot(data = all_r_solutions, aes(x = time, y = N, color = case)) +

  # Draw one line for each growth rate
  geom_line() +

  # Show the carrying capacity K
  geom_hline(yintercept = K_fixed, linetype = "dashed") +

  # Label the axes
  xlab("Time") +
  ylab("Population size N(t)") +

  # Title
  ggtitle("Logistic growth for different growth rates r") +
```

```r
# Clean black-and-white theme
theme_bw()
```



2. Fixed points of the logistic growth model

For the continuous-time logistic model:

$\frac{dN}{dt} = \dot{N} = rN \cdot (1 - K \cdot N)$,

a fixed point (also called an equilibrium) is a value N^* where $\frac{dN}{dt} = \dot{N} = 0$. We can find these graphically and numerically.

Step 2.1: Define the growth function $f(N) = rN \cdot (1 - K \cdot N)$

```r
# f(N) = dN/dt
f <- function(N) {
  r * N * (1 - N / K)
}
```

Step 2.2: Find fixed points analytically (one line)

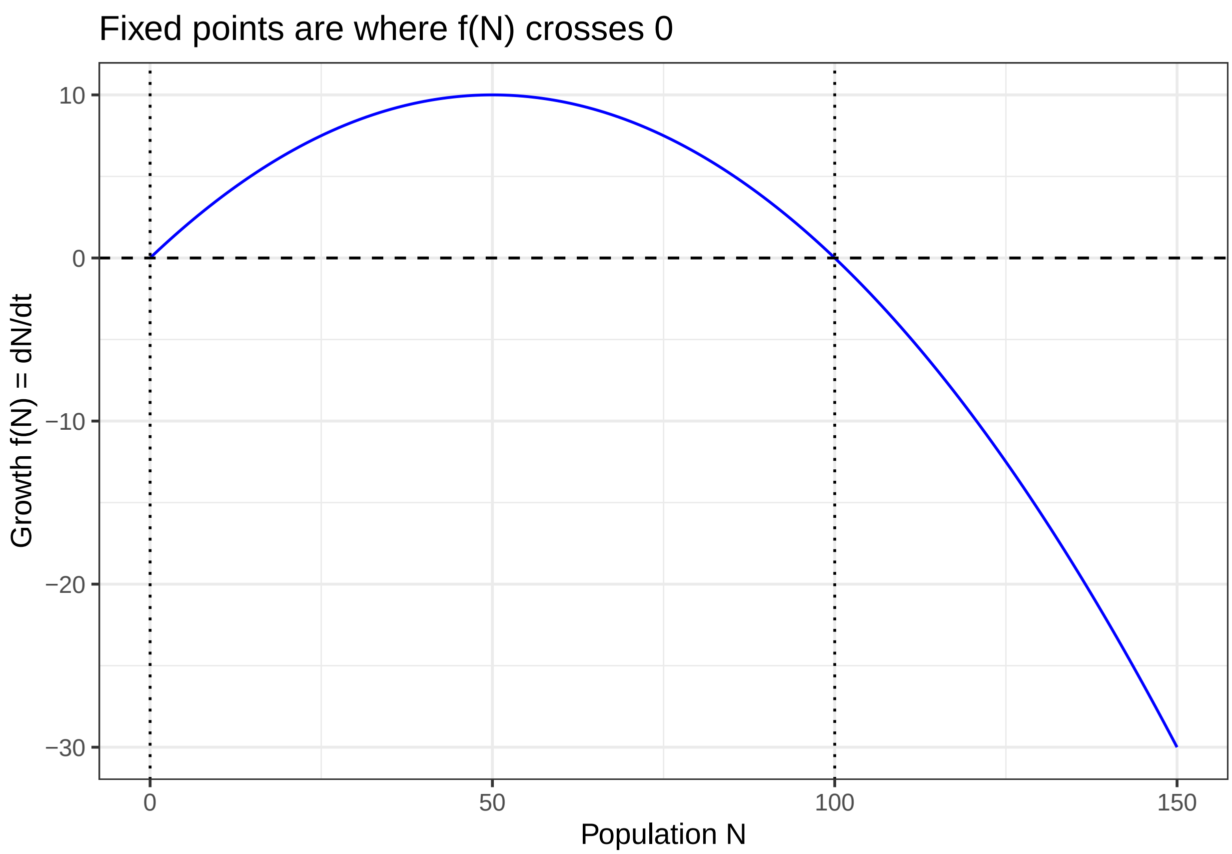
Because $rN(1 - N/K) = 0$, the fixed points are:

- $N^* = 0$
- $N^* = K$

(We'll now "see" them in plots and confirm numerically.)

Step 2.3: Graphical method: plot $f(N)$ and look where it crosses 0

```r
# Create a grid of N values to plot f(N)
N_grid <- seq(0, 1.5*K, by = 0.1)

data_f <- data.frame(
  N = N_grid,
  dNdt = f(N_grid)
)

# Plot f(N) = dN/dt
ggplot(data = data_f, aes(x = N, y = dNdt)) +

  # The curve f(N)
  geom_line(color = "blue") +

  # The line y = 0 (where fixed points happen)
  geom_hline(yintercept = 0, linetype = "dashed") +

  # Mark the fixed points we expect: 0 and K
  geom_vline(xintercept = c(0, K), linetype = "dotted") +

  # Labels
  xlab("Population N") +
  ylab("Growth f(N) = dN/dt") +
  ggtitle("Fixed points are where f(N) crosses 0") +

  theme_bw()
```



Step 2.4: Numerical method: use `uniroot()` to solve $f(N) = 0$

`uniroot()` needs an interval where the function changes sign.

```
# Root near 0 (choose an interval close to 0)
root0 <- uniroot(f, interval = c(0, 1))$root
root0
```

2.4.1 Fixed point near $N = 0$

```
## [1] 0
```

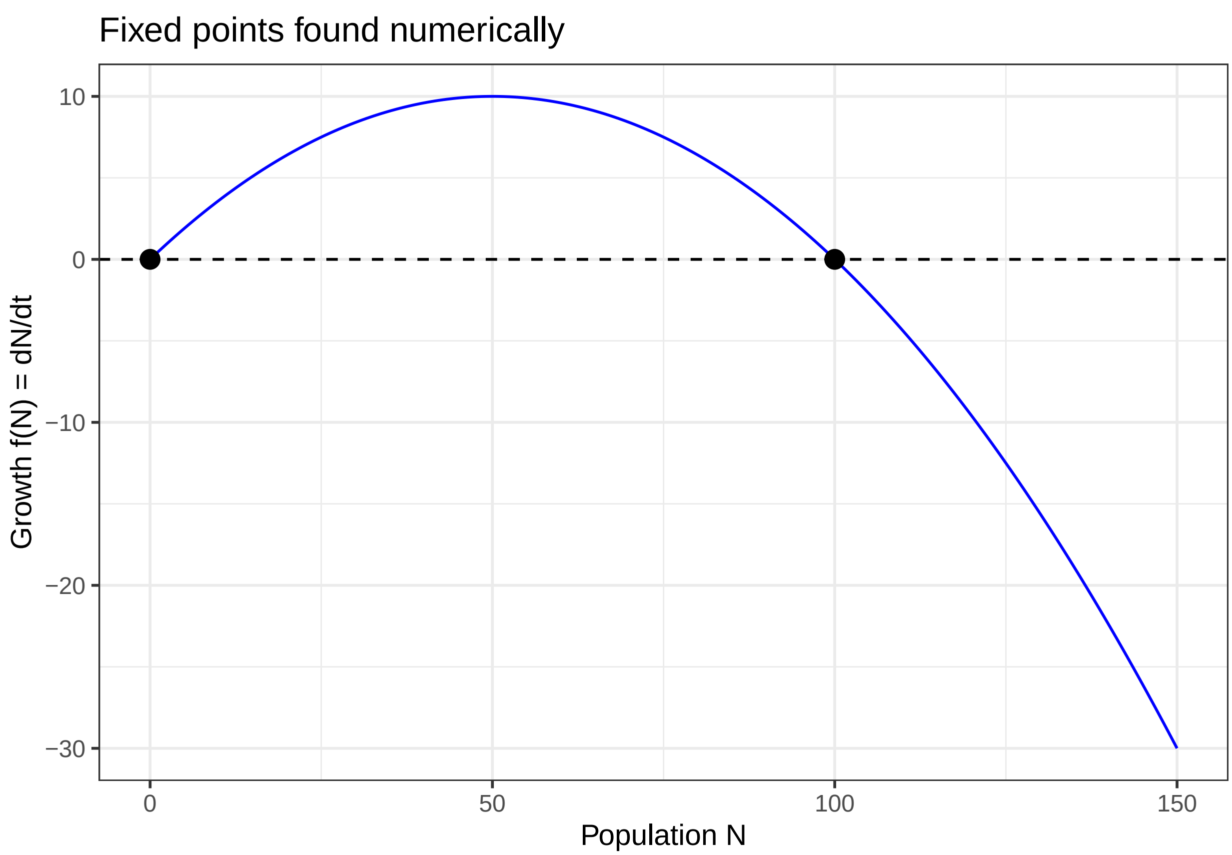
```
# Root near K (choose an interval around K)
rootK <- uniroot(f, interval = c(0.5*K, 1.5*K))$root
rootK
```

2.4.2 Fixed point near $N = K$

```
## [1] 99.99999
```

Step 2.5: Add the numerically found fixed points to the plot

```
fixed_points <- data.frame(
  N = c(root0, rootK),
  dNdt = c(f(root0), f(rootK)),
  label = c("root near 0", "root near K")
)

ggplot(data = data_f, aes(x = N, y = dNdt)) +
  geom_line(color = "blue") +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_point(data = fixed_points, aes(x = N, y = dNdt), size = 3) +
  labs(
    x = "Population N",
    y = "Growth f(N) = dN/dt",
    title = "Fixed points found numerically"
  ) +
  theme_bw()
```



3. Linear stability analysis of the fixed points of the logistic growth model

For a 1D ODE of the form

$\frac{dN}{dt} = \dot{N} = f(N).,$

a fixed point N^* is linearly stable if the slope $f'(N^*) < 0$, and unstable if $f'(N^*) > 0$. (If $f'(N^*) = 0$, linearization is inconclusive.)

For the logistic model, $f(N) = rN \cdot \left(1 - \frac{N}{K}\right)$.

Step 3.1: Define $f(N)$ and its derivative $f'(N)$

```
# f(N) = dN/dt
f <- function(N) {
  r * N * (1 - N / K)
}

# Derivative f'(N) (computed by hand)
fprime <- function(N) {
  r * (1 - 2 * N / K)
}
```

Step 3.2: Linear stability numerically (evaluate the slope at the fixed points)

Assume you already computed `root0` and `rootK` with `uniroot()` (as in the previous step).

```
# Slopes at the fixed points
slope0 <- fprime(root0)
slopeK <- fprime(rootK)

slope0
```

```
## [1] 0.4
```

```
slopeK
```

```
## [1] -0.3999999
```

For logistic growth (with $r > 0$):

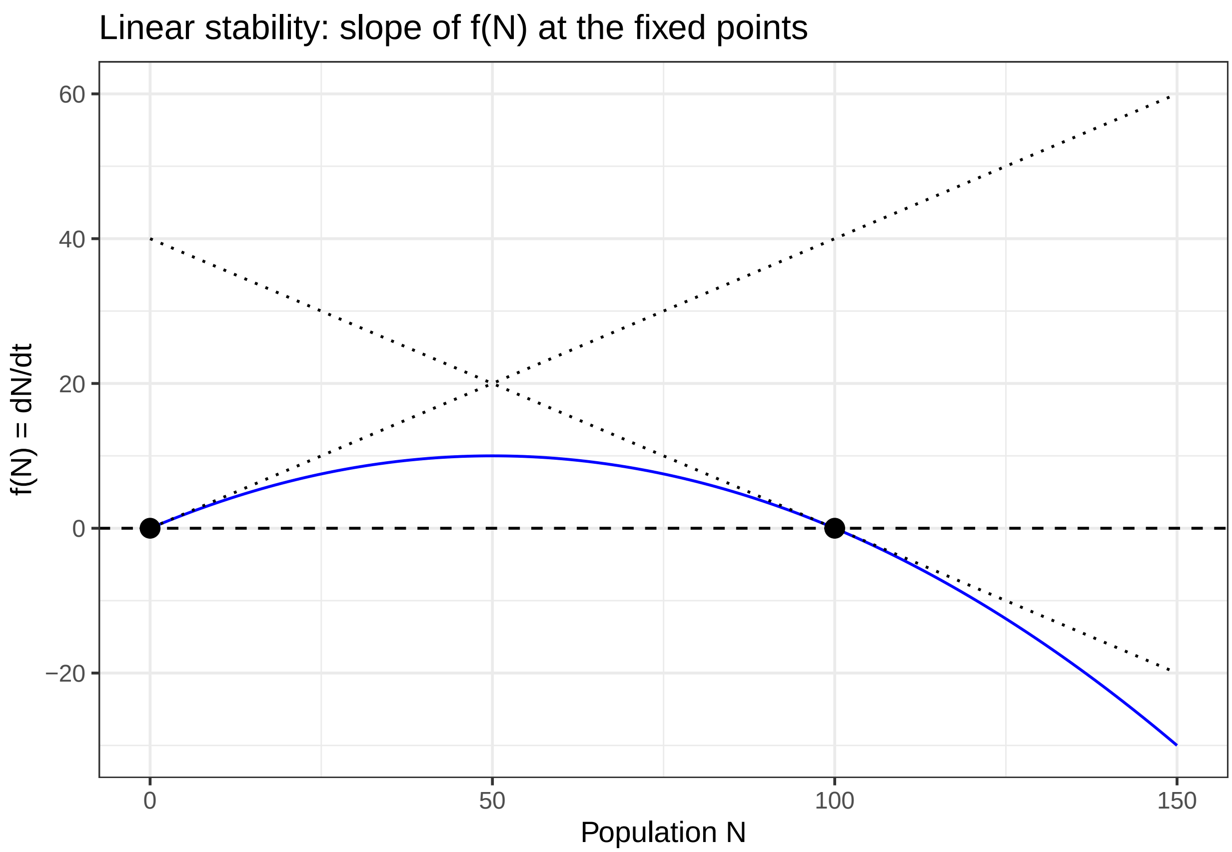
- at $N^* \approx 0$, slope is positive $\to$ **unstable**
- at $N^* \approx K$, slope is negative $\to$ **stable**

Step 3.3: Graphical stability (plot $f(N)$ and show the tangent slope at the fixed points)

This plot shows the curve $f(N)$, the line $y = 0$, and the **tangent lines** at each fixed point.

```
# Grid for plotting f(N)
N_grid <- seq(0, 1.5*K, by = 0.1)

data_f <- data.frame(
  N = N_grid,
  dNdt = f(N_grid)
```

```
)

# Build tangent lines: y = f'(N*) (N - N*) + f(N*)
# But f(N*) = 0 at fixed points, so: y = f'(N*) (N - N*)
tangent <- function(N, Nstar) {
  fprime(Nstar) * (N - Nstar)
}

data_tan0 <- data.frame(N = N_grid, y = tangent(N_grid, root0), which = "tangent at N* ~ 0")
data_tanK <- data.frame(N = N_grid, y = tangent(N_grid, rootK), which = "tangent at N* ~ K")

ggplot(data_f, aes(x = N, y = dNdt)) +
  geom_line(color = "blue") +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_point(data = data.frame(N = c(root0, rootK), dNdt = c(0, 0)),
             aes(x = N, y = dNdt), size = 3) +
  geom_line(data = data_tan0, aes(x = N, y = y), linetype = "dotted") +
  geom_line(data = data_tanK, aes(x = N, y = y), linetype = "dotted") +
  xlab("Population N") +
  ylab("f(N) = dN/dt") +
  ggtitle("Linear stability: slope of f(N) at the fixed points") +
  theme_bw()
```



At each fixed point, look at the *tilt* of the dotted tangent line:

- tangent slope **upward** (positive) → unstable

- tangent slope **downward** (negative) → stable

Step 3.4: Numerical derivative (finite difference) as a "no-calculus" check

This estimates $f'(N)$ using nearby values of f.

```r
# Numerical derivative of f using a small step h
num_derivative <- function(N, h = 1e-6) {
  (f(N + h) - f(N - h)) / (2 * h)
}

num_slope0 <- num_derivative(root0)
num_slopeK <- num_derivative(rootK)

num_slope0
```

```
## [1] 0.4
```

```r
num_slopeK
```

```
## [1] -0.3999999
```

This should agree with `fprime(root0)` and `fprime(rootK)` (up to small numerical error).

Step 3.5: Put results in a simple table

```r
results <- data.frame(
  fixed_point = c("N* near 0", "N* near K"),
  Nstar = c(root0, rootK),
  slope_exact = c(slope0, slopeK),
  slope_numeric = c(num_slope0, num_slopeK)
)

results
```

```
##   fixed_point    Nstar slope_exact slope_numeric
## 1   N* near 0  0.00000   0.4000000     0.4000000
## 2   N* near K 99.99999  -0.3999999    -0.3999999
```

A note on the linear stability of fixed points

Consider a one-dimensional dynamical system of the form

$\frac{dN}{dt} = f(N)$,

and let N^* be a fixed point, meaning that $f(N^*) = 0$.

To study the stability of N^*, we look at what happens when the system starts close to this value. Suppose the population is slightly perturbed:

$N(t) = N^* + \varepsilon(t)$,

where $\varepsilon(t)$ is small. The key question is whether this perturbation grows or shrinks over time.

Near the fixed point, the function $f(N)$ can be approximated by its linear (first-order) Taylor expansion:

$f(N^* + \varepsilon(t)) \approx f(N^*) + f'(N^*) \cdot \varepsilon.$

Since $f(N^*) = 0$, the equation becomes:

$\frac{d\varepsilon}{dt} \approx f'(N^*) \cdot \varepsilon.$

This linear differential equation has the explicit solution:

$\varepsilon(t) = \varepsilon(0) \cdot e^{f'(N^*) \cdot t}.$

The behavior of the solution depends entirely on the sign of $f'(N^*)$:

- If $f'(N^*) < 0$, the exponential term decays to zero as time increases, and the perturbation $\varepsilon(t)$ becomes smaller. In this case, trajectories return to the fixed point, and N^* is linearly stable.
- If $f'(N^*) > 0$, the exponential term grows with time, and the perturbation $\varepsilon(t)$ becomes larger. In this case, trajectories move away from the fixed point, and N^* is linearly unstable.

Intuitively, when $f'(N^*) < 0$, the graph of $f(N)$ has a negative slope at the fixed point, so small deviations are pushed back toward equilibrium. When $f'(N^*) > 0$, the slope is positive, and small deviations are amplified instead.

For the logistic growth model,

$f(N) = rN \cdot \left(1 - \frac{N}{K}\right),$

$f'(N) = r \cdot \left(1 - \frac{2N}{K}\right).$

The fixed point $N^* = 0$ satisfies $f'(0) = r > 0$ and is therefore unstable, while the fixed point $N^* = K$ satisfies $f'(K) = -r < 0$ and is therefore stable.