Lecture 9

Intraspecific Competition and Logistic Growth

Dr. Ido Filin
ifilin@univ.haifa.ac.il

17 December 2012

Outline

1. Intraspecific Competition

2. Logistic Growth

3. A discrete-time model of population regulation

Population growth vs. Relative/Per-capita growth

- **Population growth rate (PGR)** is the rate or increment of change in population size/density.

$$\frac{dN}{dt}, \quad \frac{dn}{dt}, \quad \Delta N, \quad \Delta n$$

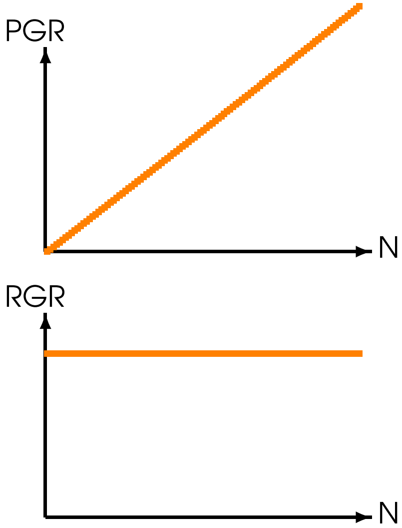
- **Relative/per-capita growth rate (RGR)** is the mean per-capita contribution of an individual to population growth.

$$\frac{1}{N}\frac{dN}{dt}, \quad \frac{1}{n}\frac{dn}{dt}, \quad \frac{\Delta N}{N}, \quad \frac{\Delta n}{n}$$

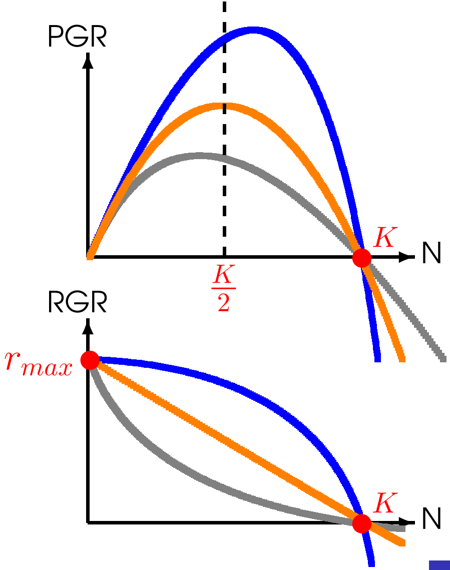
- Unregulated (exponential or geometric) population growth and regulated (density-dependent) growth show different patterns of change in PGR and RGR as density increases.

Population growth vs. Relative/Per-capita growth

Unregulated



Regulated



Density-independent vs. Density-dependent regulating factors

- Some regulating factors are density-independent: seasonal frosts or droughts, fires, storms or other catastrophes.
- Other mortality factors are density-dependent: increased starvation risk, increased risk of injury or death through competitive interactions, risk of disease or predation, etc.
- We can write total death rate as sum of density-independent terms and density-dependent terms.
- For example, $d = d_0 + d_1 N$ (d_0 and d_1 are constants.)
- Of course, density-dependence does not have to be linear (other functional forms are possible).

Density-independent vs. Density-dependent regulating factors

- Similarly, fecundity / birth rate can be written as $b = b_0 + b_1 N$. (Typically, b_1 is negative, as we expect per-capita fecundity to decrease as density rises).
- Density-dependent population regulation is the result of biotic interactions:
  - Intraspecific competition – more conspecifics, less resources per-capita.
  - Interspecific competition – more competitors (from any species), less resources per-capita.
  - Predation – more predators, higher mortality.
  - Outbreaks of disease – a kind of predation.

Intraspecific competition

- Individuals of the same species have similar needs and behavior in terms of resources, habitat, timing of lifecycle events etc.

- Therefore, individuals should suffer strong competition from conspecifics, under conditions of crowding.

- These competition effects eventually manifest themselves as reduced fecundity and survival rates.

Intraspecific competition

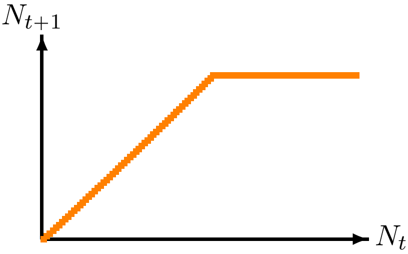
Types of intraspecific competition

1. **Scramble vs. Contest**
   - In scramble competition all individuals suffer more or less the same reduction in fecundity or same increase in mortality.
   - In contest competition there are "winners" and "losers" – all or nothing.
     "Winners" do not suffer reduction in survival or fecundity.
     "Losers" suffer maximum reduction.

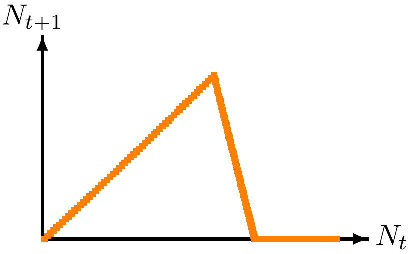
Intraspecific competition

Schematic representation of scramble and contest competition.

Contest



Scramble



Example of contest: A fixed number of territories that individuals compete for.
Example of scramble: Food divided equally, but there is a minimum requirement to survive and reproduce successfully. If not enough food per individual, all starve to death.

Intraspecific competition

Types of intraspecific competition

1. **Scramble vs. Contest**
   - In scramble competition all individuals suffer more or less the same reduction in fecundity or same increase in mortality.
   - In contest competition there are "winners" and "losers" – all or nothing.
     "Winners" do not suffer reduction in survival or fecundity.
     "Losers" suffer maximum reduction.
2. **Interference vs. Exploitation**
   - In Interference competition there is direct interaction (aggression) among individuals, where one individual prevents or reduces access to resources from the other.
   - In exploitation competition there are no direct interactions – individuals affect each other by depleting a common resource.

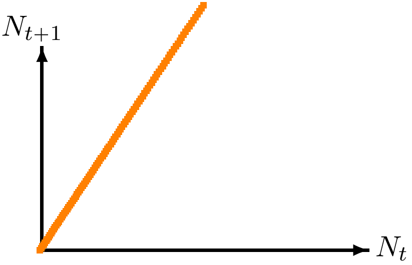
Of course these are just extremes of a spectrum of types of intraspecific competition.

Level of compensation in density-dependence

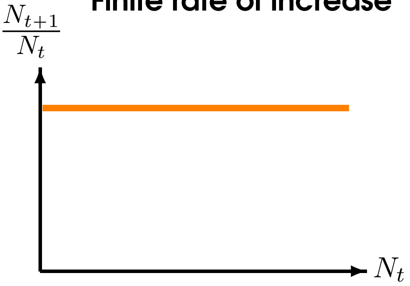
Density-independent growth: linear recursion relation

$N_{t+1} = \lambda N_t$ or $\frac{N_{t+1}}{N_t} = \lambda = const.$

Recursion relation

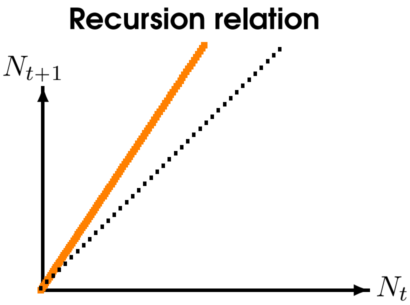


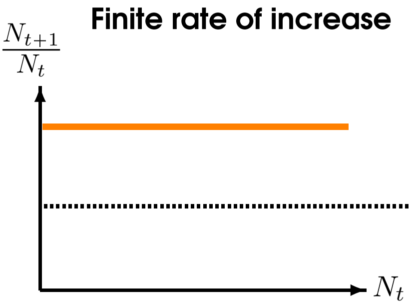
Finite rate of increase



Level of compensation in density-dependence

The long-term equilibrium population size can be obtained by intersection of the N_{t+1} curve with the unity line: $N_{t+1} = N_t$ (i.e., when finite rate of increase, $\frac{N_{t+1}}{N_t}$, is equal to 1).

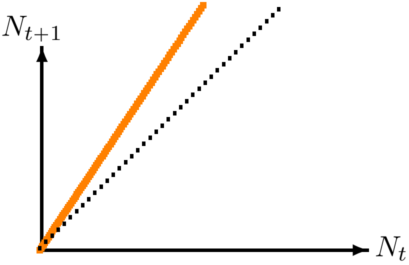




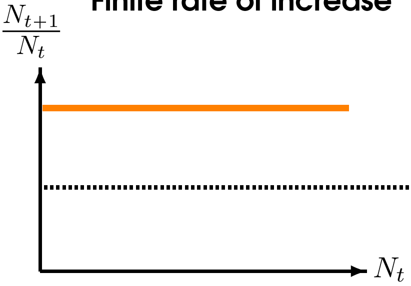
Level of compensation in density-dependence

For density-independent growth there is no such intersection, and therefore, no equilibrium population size.

Recursion relation



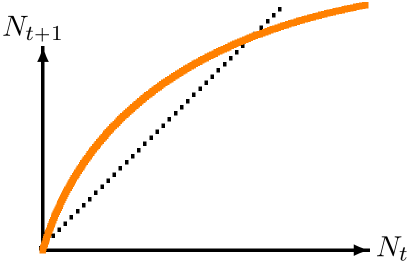
Finite rate of increase



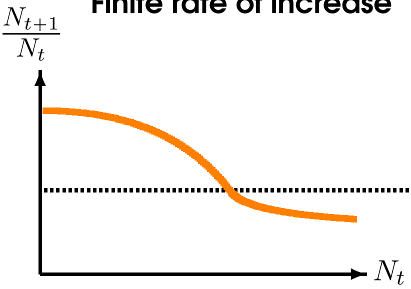
Level of compensation in density-dependence

Undercompensating density-dependence:
slope of recursion relation decreases over time, but it never reaches an asymptote.

Recursion relation



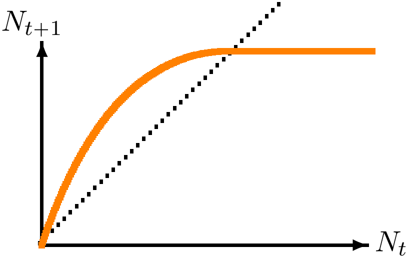
Finite rate of increase



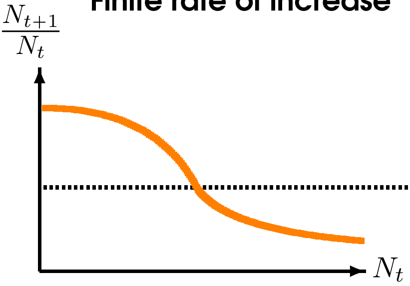
Level of compensation in density-dependence

Exactly compensating density-dependence:
For high enough density (N_t), $N_{t+1} = const$, independent of initial density N_t.

Recursion relation



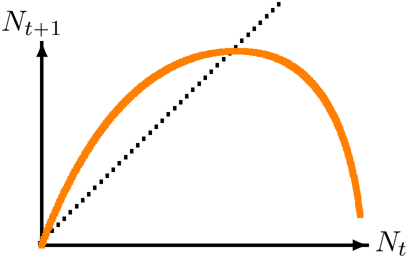
Finite rate of increase



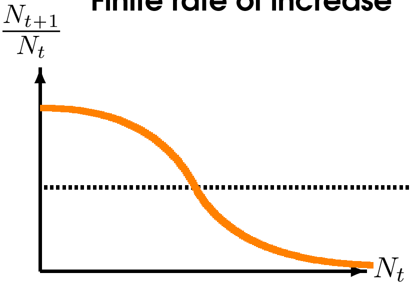
Level of compensation in density-dependence

Overcompensating density-dependence:
N_{t+1} decreases with increasing density (N_t), if N_t is high.
This type of density-dependence can potentially cause population collapse (a drop from very high N_t to very low N_{t+1}) and fluctuations in population size.

Recursion relation



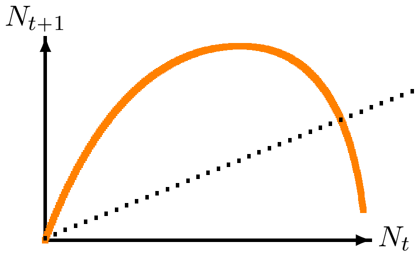
Finite rate of increase



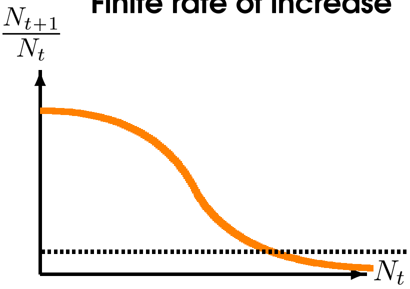
Level of compensation in density-dependence

Overcompensating density-dependence:
N_{t+1} decreases with increasing density (N_t), if N_t is high.
This type of density-dependence can potentially cause population collapse (a drop from very high N_t to very low N_{t+1}) and fluctuations in population size.

Recursion relation

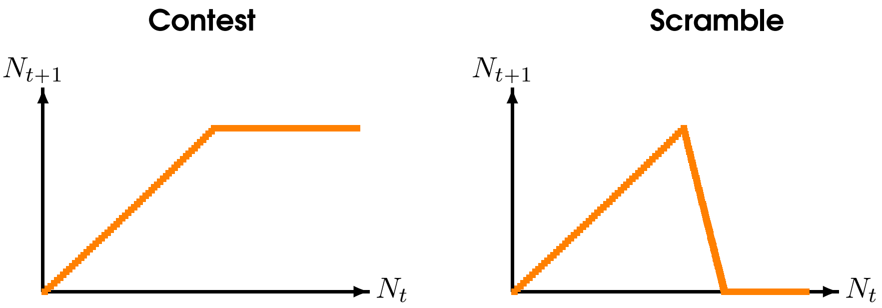


Finite rate of increase



Intraspecific competition

Schematic representation of scramble and contest competition.

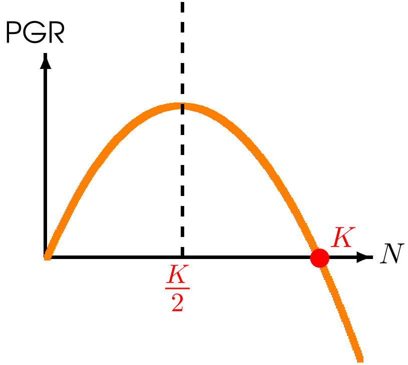


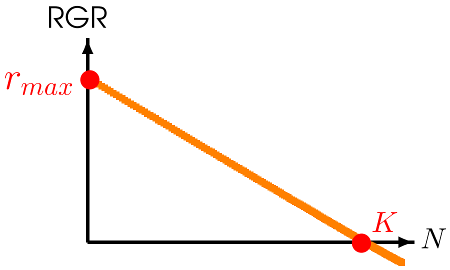
Example of contest: A fixed number of territories that individuals compete for.
Example of scramble: Food divided equally, but there is a minimum requirement to survive and reproduce successfully. If not enough food per individual, all starve to death.

Outline

Logistic growth

- Simplest form of density-dependence is linear.
- Always start with a simple model – otherwise it is difficult to draw conclusions.
- Recruitment is a quadratic function – i.e., a parabola.
- Maximum recruitment (maximum PGR) occurs at $K/2$.





Logistic growth

- Continuous time model is given by

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

- I.e., the expression for exponential growth, multiplied by a competition factor that is increasingly smaller than 1, as population size/density increases.
- RGR or per-capita growth rate is not constant, but given by the linear density-dependence relation

$$r(N) = r_{max}\left(1 - \frac{N}{K}\right)$$

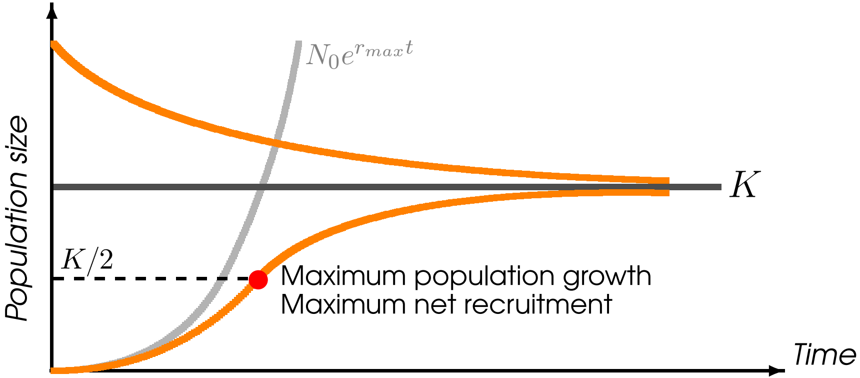
- An analogous discrete time model is given by

$$\lambda(N_t) = 1 + r_{max}\left(1 - \frac{N_t}{K}\right)$$

Logistic growth

The logistic growth curve (continuous time):

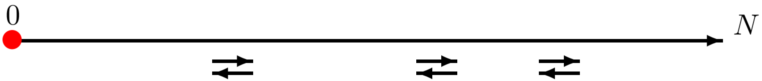
$$N(t) = \frac{N_0 K}{N_0 + (K - N_0)e^{-r_{max}t}}$$



Stationarity and stability

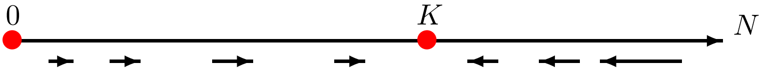
- **Stationary points** represent special values of the variable that do not change over time.
- I.e., if we start at a stationary point, we remain on it on subsequent times.
- Therefore, stationary points are defined zero rate of change: $\Delta N = 0$ or $dN/dt = 0$.
- E.g., for the logistic growth model we have two stationary points, $N = 0$ and $N = K$:

- For exponential growth, only $N = 0$:



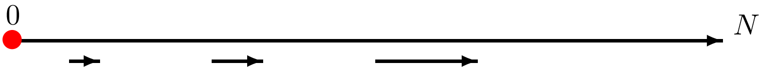
Stationarity and stability

- A stationary point can be either **stable** or **unstable**.
- Any deviation from a stable stationary point would tend to decrease over time – i.e., a restoring "force" operating towards the point.
- Any deviation from an unstable stationary point would tend to increase over time – i.e., a repelling "force" away from the point.
- We can determine stability graphically.
- E.g., for the logistic model $N = 0$ is unstable, and $N = K$ is stable:

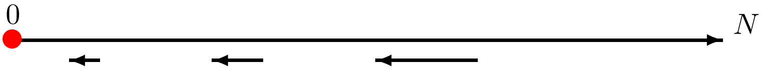


Stationarity and stability

- For the exponential model, $N = 0$ is unstable, if $r > 0$:



- And stable, if $r < 0$:

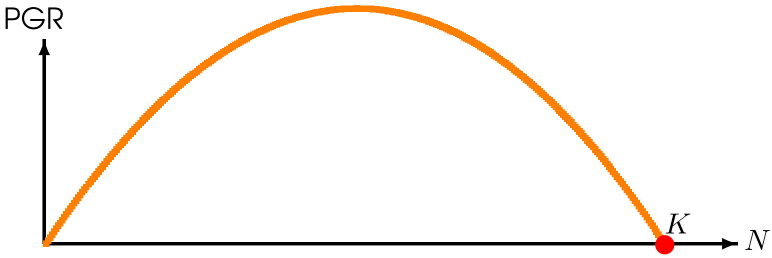


- Ultimately, checking for stability requires mathematical analysis using methods from calculus, linear algebra and nonlinear dynamics.
- But the graphical method is sufficient for our purposes.
- We will return to this subject when we talk about interspecific competition.

Application to sustainable harvesting

- If we harvest a logistically growing population at a constant rate (i.e., individuals or kilos per unit time) – the stationary population size will decrease.
- For example, commercial fishing depletes natural fish populations.
- Denoting harvest rate by H, the dynamics is given by

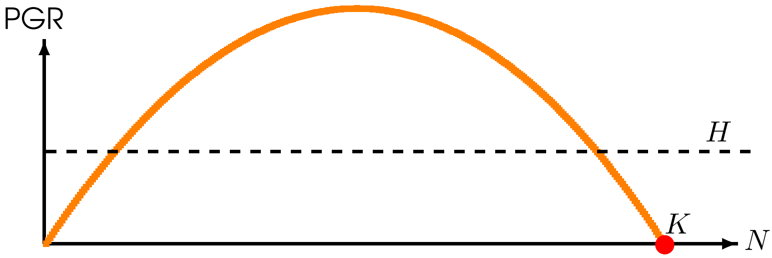
$$\frac{dN}{dt} = r_{max}N\left(1 - \frac{N}{K}\right) - H$$



Application to sustainable harvesting

- If we harvest a logistically growing population at a constant rate (i.e., individuals or kilos per unit time) – the stationary population size will decrease.
- For example, commercial fishing depletes natural fish populations.
- Denoting harvest rate by H, the dynamics is given by

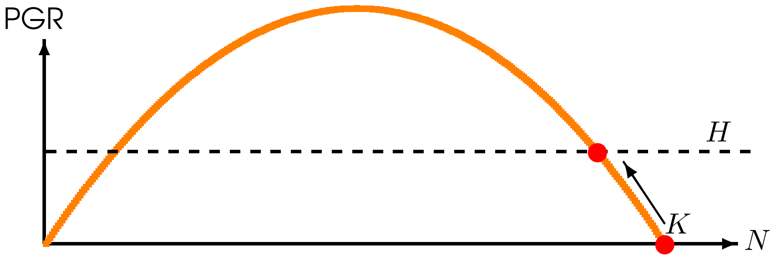
$$\frac{dN}{dt} = r_{max} N \left(1 - \frac{N}{K}\right) - H$$



Application to sustainable harvesting

- If we harvest a logistically growing population at a constant rate (i.e., individuals or kilos per unit time) – the stationary population size will decrease.
- For example, commercial fishing depletes natural fish populations.
- Denoting harvest rate by H, the dynamics is given by

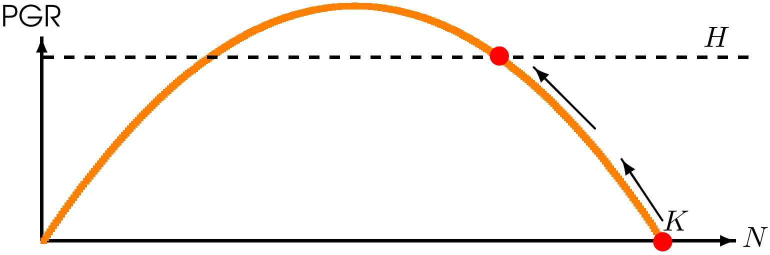
$$\frac{dN}{dt} = r_{max} N \left(1 - \frac{N}{K}\right) - H$$



Application to sustainable harvesting

- If we harvest a logistically growing population at a constant rate (i.e., individuals or kilos per unit time) – the stationary population size will decrease.
- For example, commercial fishing depletes natural fish populations.
- Denoting harvest rate by H, the dynamics is given by

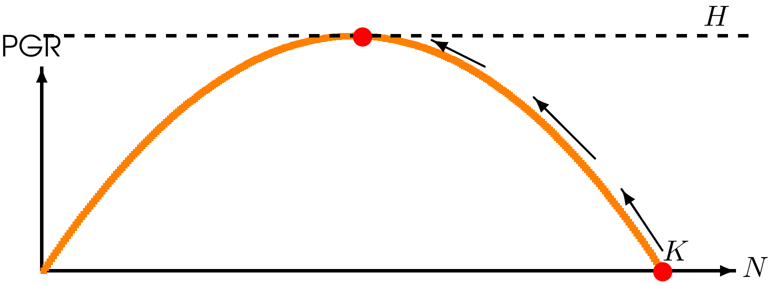
$$\frac{dN}{dt} = r_{max}N\left(1 - \frac{N}{K}\right) - H$$



Application to sustainable harvesting

- If we harvest a logistically growing population at a constant rate (i.e., individuals or kilos per unit time) – the stationary population size will decrease.
- For example, commercial fishing depletes natural fish populations.
- Denoting harvest rate by H, the dynamics is given by

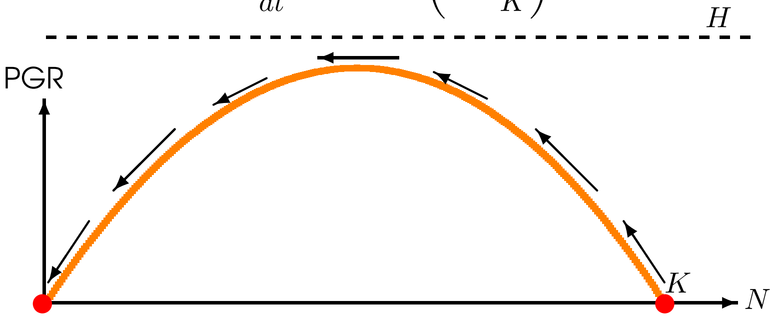
$$\frac{dN}{dt} = r_{max}N\left(1 - \frac{N}{K}\right) - H$$



Application to sustainable harvesting

- If we harvest a logistically growing population at a constant rate (i.e., individuals or kilos per unit time) – the stationary population size will decrease.
- For example, commercial fishing depletes natural fish populations.
- Denoting harvest rate by H, the dynamics is given by

$$\frac{dN}{dt} = r_{max} N \left(1 - \frac{N}{K}\right) - H$$



Application to sustainable harvesting

- Overharvesting (e.g., overfishing) occurs when H exceeds the maximal possible net recruitment rate (PGR).

- The natural population collapses, resulting in loss of the natural resource.

- For example,

  - Fisheries collapse – resulting not only in damage to nature, but also economic collapse of industries and human communities.

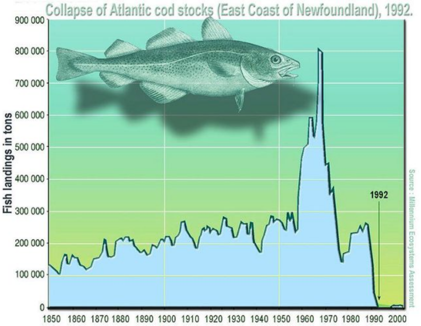
  - Overgrazed grasslands/pastures (grazed by livestock) turn into deserts – again resulting in subsequent collapse of human societies.

  - Overhunted animals go extinct.

Application to sustainable harvesting

- Following collapse of the resource, harvesting must be stopped for a long period, to allow the natural population to recover and exceed the maximum PGR point (in logistic growth, to exceed $K/2$).

- **Sustainable harvesting** can then be achieved if H is lower than the maximal PGR.



Outline

An R program for regulated populations

This program is similar to the one we wrote for unregulated growth:

1. `GeometricGrowth <- function( popSize, lambda ) { return( lambda * popSize) }`
2. `genNum <- 32; Ninitial <- 1`
3. `increaseRate <- 1.6;`
4. `PopGrowthFunc <- GeometricGrowth`
5. `N <- numeric(genNum); Time <- seq(from = 0, by = 24, length = genNum)`
6. `N[1] <- Ninitial`
7. `for ( index in 2:genNum ) { N[index] <- PopGrowthFunc( popSize = N[index-1], lambda = increaseRate ) }`
8. `plot( Time, N, xlab = "Time[hours]" )`

An R program for regulated populations

Note that we use a general parameter, `PopGrowthFunc`, to represent the population growth function. We assign to it the function for geometric growth.

1. `GeometricGrowth <- function( popSize, lambda ) { return( lambda * popSize) }`
2. `genNum <- 32; Ninitial <- 1`
3. `increaseRate <- 1.6;`
4. `PopGrowthFunc <- GeometricGrowth`
5. `N <- numeric(genNum); Time <- seq(from = 0, by = 24, length = genNum)`
6. `N[1] <- Ninitial`
7. `for ( index in 2:genNum ) { N[index] <- PopGrowthFunc( popSize = N[index-1], lambda = increaseRate ) }`
8. `plot( Time, N, xlab = "Time[hours]" )`

An R program for regulated populations

You can assign a function to an R-variable, just like assigning a value to a variable. The case here is similar to: `x <- 3; y <- x`. Eventually, both `x` and `y` hold the value 3.

```
1 GeometricGrowth <- function( popSize, lambda )
  { return( lambda * popSize) }
2 genNum <- 32; Ninitial <- 1
3 increaseRate <- 1.6;
4 PopGrowthFunc <- GeometricGrowth
5 N <- numeric(genNum); Time <- seq(from = 0, by = 24,
  length = genNum)
6 N[1] <- Ninitial
7 for ( index in 2:genNum )
  { N[index] <- PopGrowthFunc( popSize = N[index-1],
  lambda = increaseRate ) }
8 plot( Time, N, xlab = "Time[hours]" )
```

An R program for regulated populations

Similarly, you assign a function to `GeometricGrowth` and the assign the contents of `GeometricGrowth` (i.e., the function) to `PopGrowthFunc`.

1. `GeometricGrowth <- function( popSize, lambda ) { return( lambda * popSize) }`
2. `genNum <- 32; Ninitial <- 1`
3. `increaseRate <- 1.6;`
4. `PopGrowthFunc <- GeometricGrowth`
5. `N <- numeric(genNum); Time <- seq(from = 0, by = 24, length = genNum)`
6. `N[1] <- Ninitial`
7. `for ( index in 2:genNum ) { N[index] <- PopGrowthFunc( popSize = N[index-1], lambda = increaseRate ) }`
8. `plot( Time, N, xlab = "Time[hours]" )`

An R program for regulated populations

How should we modify this program to include density-dependence?

1. `GeometricGrowth <- function( popSize, lambda ) { return( lambda * popSize) }`
2. `genNum <- 32; Ninitial <- 1`
3. `increaseRate <- 1.6;`
4. `PopGrowthFunc <- GeometricGrowth`
5. `N <- numeric(genNum); Time <- seq(from = 0, by = 24, length = genNum)`
6. `N[1] <- Ninitial`
7. `for ( index in 2:genNum ) { N[index] <- PopGrowthFunc( popSize = N[index-1], lambda = increaseRate ) }`
8. `plot( Time, N, xlab = "Time[hours]" )`

An R program for regulated populations

Firstly, we add the logistic growth function

1. `GeometricGrowth <- function( popSize, lambda ) { return( lambda * popSize) }`
2. `LogisticGrowth <- function( popSize, rmax, K ) { return( popSize + rmax*popSize*(1-popSize/K) ) }`
3. `genNum <- 32; Ninitial <- 1`
4. `increaseRate <- 1.6;`
5. `PopGrowthFunc <- GeometricGrowth`
6. `N <- numeric(genNum); Time <- seq(from = 0, by = 24, length = genNum)`
7. `N[1] <- Ninitial`
8. `for ( index in 2:genNum ) { N[index] <- PopGrowthFunc( popSize = N[index-1], lambda = increaseRate ) }`
9. `plot( Time, N, xlab = "Time[hours]" )`

An R program for regulated populations

Secondly, we add parameters for logistic growth

1. `GeometricGrowth <- function( popSize, lambda ) { return( lambda * popSize) }`
2. `LogisticGrowth <- function( popSize, rmax, K ) { return( popSize + rmax*popSize*(1-popSize/K) ) }`
3. `genNum <- 32; Ninitial <- 1`
4. `increaseRate <- 1.6; maxRGR <- 0.6; capacity <- 100`
5. `PopGrowthFunc <- GeometricGrowth`
6. `N <- numeric(genNum); Time <- seq(from = 0, by = 24, length = genNum)`
7. `N[1] <- Ninitial`
8. `for ( index in 2:genNum ) { N[index] <- PopGrowthFunc( popSize = N[index-1], lambda = increaseRate ) }`
9. `plot( Time, N, xlab = "Time[hours]" )`

An R program for regulated populations

Next, we change `PopGrowthFunc` to use logistic instead of geometric

1. `GeometricGrowth <- function( popSize, lambda )`
   `{ return( lambda * popSize) }`
2. `LogisticGrowth <- function( popSize, rmax, K )`
   `{ return( popSize + rmax*popSize*(1-popSize/K) ) }`
3. `genNum <- 32; Ninitial <- 1`
4. `increaseRate <- 1.6; maxRGR <- 0.6; capacity <- 100`
5. `PopGrowthFunc <- LogisticGrowth`
6. `N <- numeric(genNum); Time <- seq(from = 0, by = 24,`
   `length = genNum)`
7. `N[1] <- Ninitial`
8. `for ( index in 2:genNum )`
   `{ N[index] <- PopGrowthFunc( popSize = N[index-1],`
   `lambda = increaseRate ) }`
9. `plot( Time, N, xlab = "Time[hours]" )`

An R program for regulated populations

Finally, we add arguments to the function call in the for-loop

```
1 GeometricGrowth <- function( popSize, lambda )
  { return( lambda * popSize) }
2 LogisticGrowth <- function( popSize, rmax, K )
  { return( popSize + rmax*popSize*(1-popSize/K) ) }
3 genNum <- 32; Ninitial <- 1
4 increaseRate <- 1.6; maxRGR <- 0.6; capacity <- 100
5 PopGrowthFunc <- LogisticGrowth
6 N <- numeric(genNum); Time <- seq(from = 0, by = 24,
  length = genNum)
7 N[1] <- Ninitial
8 for ( index in 2:genNum )
  { N[index] <- PopGrowthFunc( popSize = N[index-1],
  lambda = increaseRate, rmax = maxRGR, K = capacity )
  }
9 plot( Time, N, xlab = "Time[hours]" )
```

An R program for regulated populations

But there is a problem. Logistic growth function does not have an argument named `lambda`

1. `GeometricGrowth <- function( popSize, lambda ) { return( lambda * popSize) }`
2. `LogisticGrowth <- function( popSize, rmax, K ) { return( popSize + rmax*popSize*(1-popSize/K) ) }`
3. `genNum <- 32; Ninitial <- 1`
4. `increaseRate <- 1.6; maxRGR <- 0.6; capacity <- 100`
5. `PopGrowthFunc <- LogisticGrowth`
6. `N <- numeric(genNum); Time <- seq(from = 0, by = 24, length = genNum)`
7. `N[1] <- Ninitial`
8. `for ( index in 2:genNum ) { N[index] <- PopGrowthFunc( popSize = N[index-1], lambda = increaseRate, rmax = maxRGR, K = capacity ) }`
9. `plot( Time, N, xlab = "Time[hours]" )`

An R program for regulated populations

We solve that by adding the ... argument to both functions

1. `GeometricGrowth <- function( popSize, lambda, ... ) { return( lambda * popSize) }`
2. `LogisticGrowth <- function( popSize, rmax, K, ... ) { return( popSize + rmax*popSize*(1-popSize/K) ) }`
3. `genNum <- 32; Ninitial <- 1`
4. `increaseRate <- 1.6; maxRGR <- 0.6; capacity <- 100`
5. `PopGrowthFunc <- LogisticGrowth`
6. `N <- numeric(genNum); Time <- seq(from = 0, by = 24, length = genNum)`
7. `N[1] <- Ninitial`
8. `for ( index in 2:genNum ) { N[index] <- PopGrowthFunc( popSize = N[index-1], lambda = increaseRate, rmax = maxRGR, K = capacity ) }`
9. `plot( Time, N, xlab = "Time[hours]" )`

An R program for regulated populations

The ... argument allows additional arguments, unspecified in the function declaration, to be passed to the function

1. ```
   GeometricGrowth <- function( popSize, lambda, ... )
   { return( lambda * popSize) }
   ```
2. ```
   LogisticGrowth <- function( popSize, rmax, K, ... )
   { return( popSize + rmax*popSize*(1-popSize/K) ) }
   ```
3. ```
   genNum <- 32; Ninitial <- 1
   ```
4. ```
   increaseRate <- 1.6; maxRGR <- 0.6; capacity <- 100
   ```
5. ```
   PopGrowthFunc <- LogisticGrowth
   ```
6. ```
   N <- numeric(genNum); Time <- seq(from = 0, by = 24,
   length = genNum)
   ```
7. ```
   N[1] <- Ninitial
   ```
8. ```
   for ( index in 2:genNum )
   { N[index] <- PopGrowthFunc( popSize = N[index-1],
   lambda = increaseRate, rmax = maxRGR, K = capacity )
   }
   ```
9. ```
   plot( Time, N, xlab = "Time[hours]" )
   ```
   ```

An R program for regulated populations

We can now switch between geometric and logistic growth, just by changing the parameter `PopGrowthFunc`

1. `GeometricGrowth <- function( popSize, lambda, ... ) { return( lambda * popSize) }`
2. `LogisticGrowth <- function( popSize, rmax, K, ... ) { return( popSize + rmax*popSize*(1-popSize/K) ) }`
3. `genNum <- 32; Ninitial <- 1`
4. `increaseRate <- 1.6; maxRGR <- 0.6; capacity <- 100`
5. `PopGrowthFunc <- LogisticGrowth`
6. `N <- numeric(genNum); Time <- seq(from = 0, by = 24, length = genNum)`
7. `N[1] <- Ninitial`
8. `for ( index in 2:genNum ) { N[index] <- PopGrowthFunc( popSize = N[index-1], lambda = increaseRate, rmax = maxRGR, K = capacity ) }`
9. `plot( Time, N, xlab = "Time[hours]" )`

An R program for regulated populations

Save and run the program for different cases (geometric vs. logistic, different parameter settings, etc.)

```
1  GeometricGrowth <- function( popSize, lambda, ... )
   { return( lambda * popSize) }
2  LogisticGrowth <- function( popSize, rmax, K, ... )
   { return( popSize + rmax*popSize*(1-popSize/K) ) }
3  genNum <- 32; Ninitial <- 1
4  increaseRate <- 1.6; maxRGR <- 0.6; capacity <- 100
5  PopGrowthFunc <- LogisticGrowth
6  N <- numeric(genNum); Time <- seq(from = 0, by = 24,
   length = genNum)
7  N[1] <- Ninitial
8  for ( index in 2:genNum )
   { N[index] <- PopGrowthFunc( popSize = N[index-1],
   lambda = increaseRate, rmax = maxRGR, K = capacity )
   }
9  plot( Time, N, xlab = "Time[hours]" )
```