Modelling Biological Systems Exam 2016

Sergii Gladchuk

December 30, 2016

3. The dynamics of an age structured population.

Consider a population consisting of juveniles (J) and adults (A). The juveniles mature to become adults at a rate g but also die at a rate μ_J. The adults reproduce at a rate b and die at a rate $\mu_0(1+cA)$, where c is a positive constant representing density dependence of the mortality. The dynamics can be written

$$\begin{cases} \frac{dJ}{dt} = bA - gJ - \mu_J J \\ \frac{dA}{dt} = gJ - \mu_0(1+cA)A \end{cases}$$

a) Where is the non-trivial equilibrium?

Equilibrium means there is no change of A and J with time in other words:

$$\begin{cases} \frac{dJ}{dt} = 0 \\ \frac{dA}{dt} = 0 \end{cases}$$

So this system should be solved:

$$\begin{cases} bA - gJ - \mu_J J = 0 \\ gJ - \mu_0(1+cA)A = 0 \end{cases}$$

$$\begin{cases} J(g+\mu_J) = bA \\ gJ = \mu_0 + \mu_0 cA^2 \end{cases}$$

$$\begin{cases} J = \frac{bA}{g+\mu_J} \\ J = \frac{\mu_0+\mu_0 cA^2}{g} \end{cases}$$

The above system will be used to build isoclines in later task. Now variable A can be found from both equations:

$$\frac{bA}{g+\mu_J} = \frac{\mu_0 + \mu_0 cA^2}{g}$$

$$gbA = (\mu_0 A + \mu_0 cA^2)(g+\mu_J)$$

$$\mu_0 gA + \mu_0\mu_J A - gbA + \mu_0\mu_J cA^2 + \mu_0 cgA^2 = 0$$

$$A(A + \frac{\mu_0 g + \mu_0\mu_J - gb}{\mu_0\mu_J c + \mu_0 cg}) = 0$$

So there is one trivial solution $A = 0$, non-trivial:

$$A^* = \frac{gb - \mu_0 g - \mu_0 \mu_J}{\mu_0 \mu_J c + \mu_0 cg}$$

And J will be:

$$J^* = \frac{b(gb - \mu_0 g - \mu_0 \mu_J)}{(g + \mu_J)(\mu_0 \mu_J c + \mu_0 cg)}$$

Last two equations describe co-existence equilibrium (both A and J are constant)

b) Calculate the Jacobian matrix of that equilibrium!

Jacobian matrix in this case would look like:

$$\begin{pmatrix} \frac{\partial f_J}{\partial J} & \frac{\partial f_J}{\partial A} \\ \frac{\partial f_A}{\partial J} & \frac{\partial f_A}{\partial A} \end{pmatrix}$$

So 4 partial derivative with A^* and J^* should be solved:

$$\left.\frac{\partial f_J}{\partial J}\right|_* = \left.\frac{\partial}{\partial J}\Big(bA - gJ - \mu_J J\Big)\right|_* = -g - \mu_J$$

$$\left.\frac{\partial f_J}{\partial A}\right|_* = \left.\frac{\partial}{\partial A}\Big(bA - gJ - \mu_J J\Big)\right|_* = b$$

$$\left.\frac{\partial f_A}{\partial J}\right|_* = \left.\frac{\partial}{\partial A}\Big(gJ - \mu_0(1 + cA)A\Big)\right|_* = g$$

$$\left.\frac{\partial f_A}{\partial A}\right|_* = \left.\frac{\partial}{\partial A}\Big(gJ - \mu_0(1 + cA)A\Big)\right|_* = \left.\frac{\partial}{\partial A}\Big(gJ - \mu_0 A - \mu_0 cA^2\Big)\right|_* = \left.\Big(-\mu_0 - 2\mu_0 cA\Big)\right|_* =$$

$$= -\mu_0 - 2\mu_0 c\left(\tfrac{gb - \mu_0 g - \mu_0 \mu_J}{\mu_0 \mu_J c + \mu_0 cg}\right) = -\mu_0 - \tfrac{2gb - 2\mu_0 g - 2\mu_0 \mu_J}{\mu_J - g} = \tfrac{3\mu_0 g - 2gb + \mu_0 \mu_J}{\mu_J - g}$$

So Jacobian matrix:

$$\begin{pmatrix} -g - \mu_J & b \\ g & \frac{3\mu_0 g - 2gb + \mu_0 \mu_J}{\mu_J - g} \end{pmatrix}$$

c) Usage of the Jacobian

How would you use the Jacobian to evaluate the stability properties of the equilibrium (don't do it!)?

Jacobian matrix can be used to define if equilibrium at state A^*, J^* is stable. This can be done by calculating eigenvalues from the matrix. If all real parts of eigenvalues are negative then equilibrium is stable. Also complex eigenvalues will lead to oscillation - if real parts are all negative then trajectory will spiralize to equilibrium point calculated in a) and if any of eigenvalues is positive it will spiralize away from equilibrium.

d) Write an R script that plots the isoclines of the system!

Use, if you wish, the parameter values $b = 1, g = 0.5, \mu_J = 0.3, \mu_0 = 0.2, c = 0.01$

This system of equations will be used to plot isoclines:

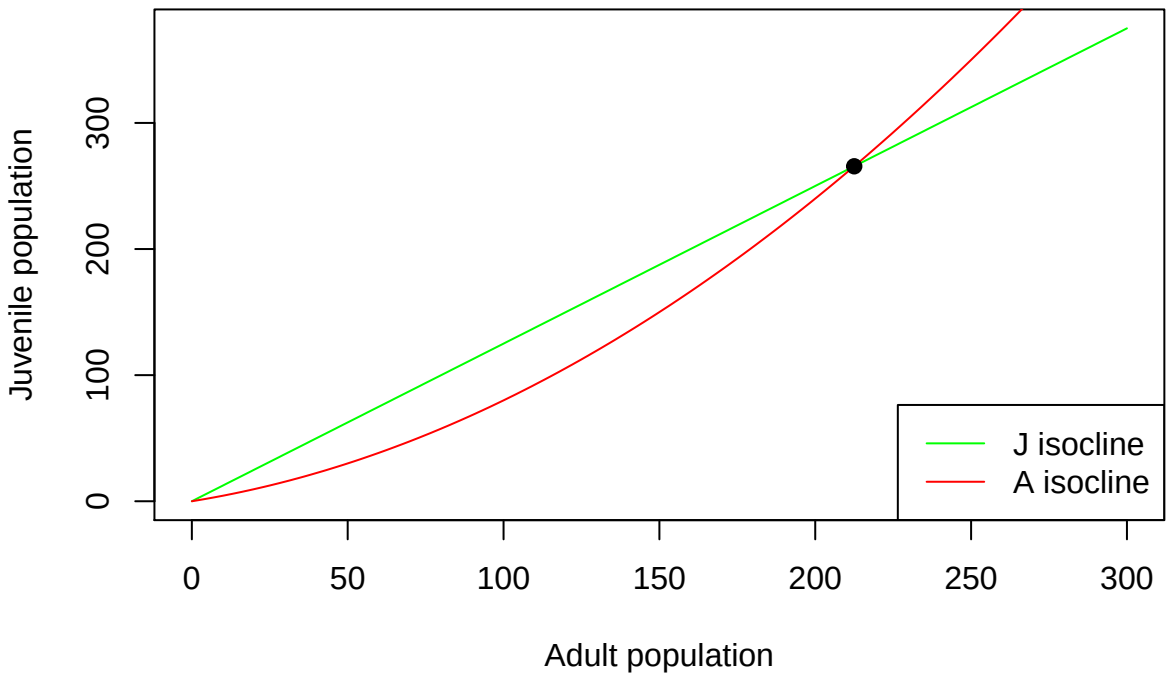
$$\begin{cases} J = \frac{bA}{g+\mu_J} \\ J = \frac{\mu_0+\mu_0 cA^2}{g} \end{cases}$$

Also equilibrium point will be checked based on equations

$$A^* = \frac{gb - \mu_0 g - \mu_0 \mu_J}{\mu_0 \mu_J c + \mu_0 cg}$$

$$J^* = \frac{b(gb - \mu_0 g - \mu_0 \mu_J)}{(g + \mu_J)(\mu_0 \mu_J c + \mu_0 cg)}$$

```
# vecror of population adults
A = c(0:300);
#constrants
b <- 1;
g <- 0.5;
mu_j <- 0.3;
mu_0 <- 0.2;
c <- 0.01;

isocline1 <- function(vec){
  line1 <- b*vec/(g+mu_j);
  line1;
}
isocline2 <- function(vec){
  line2 <- (mu_0*vec + mu_0 * c * vec**2) / g;
  line2;
}

#equilibrium
J_eq <- b*(g*b-mu_0*g-mu_0*mu_j)/((g+mu_j)*(mu_0*mu_j*c+mu_0*c*g));
A_eq <- (g*b-mu_0*g-mu_0*mu_j)/(mu_0*mu_j*c+mu_0*c*g);

#plot
par(mfrow = c(1,1))
plot (A, isocline1(A), type = 'l', col = 'green',
      xlab = 'Adult population', ylab = 'Juvenile population')
lines(A, isocline2(A), col = 'red')
points(A_eq,J_eq, pch=19)
legend('bottomright', legend = c('J isocline','A isocline'),
       lty=c(1,1), col=c('green','red'))
```



e) Another script

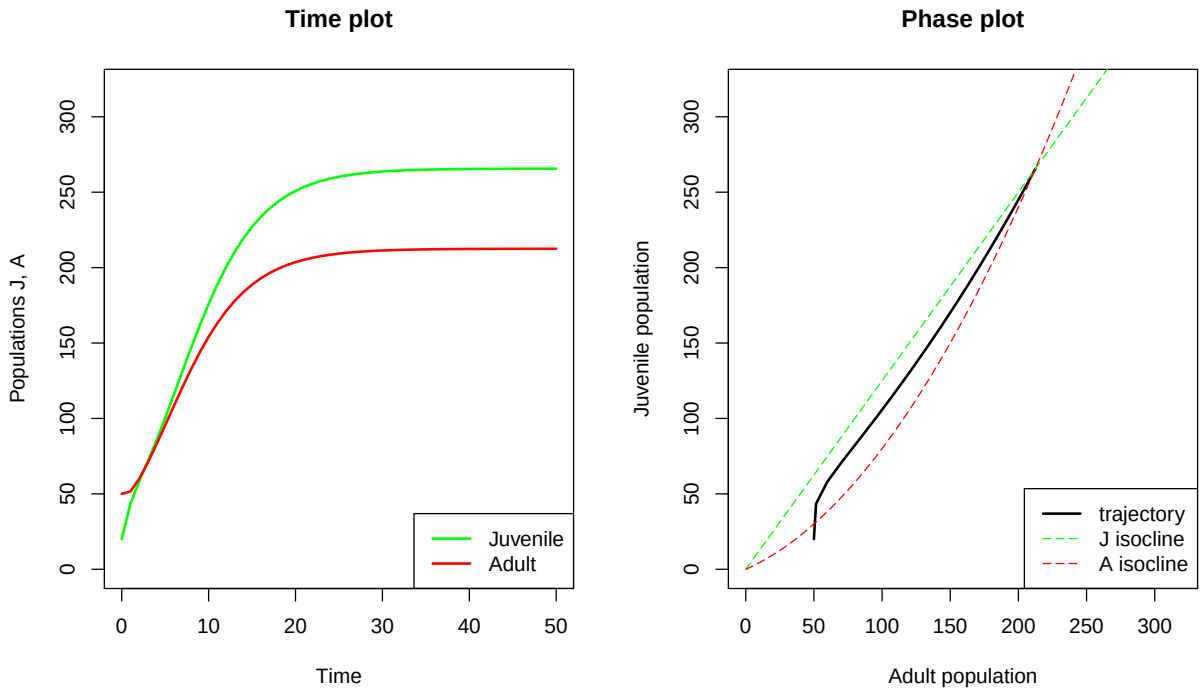
Write another script that simulates the differential equations and plots the result in two ways:

- *as A and J vs. time*
- *in the phase plane together with the isoclines*

(you may use the same parameter values as above, and an initial condition of your own choice)

```
library(deSolve)

age_population <- function(t, JA, P) {
  # extract vector content
  J <- JA[1]
  A <- JA[2]

  # calculate the growth rates:
  dJdt <- P$b * A - P$g * J - P$mu_j * J
  dAdt <- P$g * J - P$mu_0 * (1 + P$c * A) * A

  # output
  list(c(dJdt, dAdt))
}
```

```r
# set the vector of time-points for the output
timevec <- seq(0, 50, by=1)

# list of parameters
P <- list(b = 1,
          g = 0.5,
          mu_j = 0.3,
          mu_0 = 0.2,
          c = 0.01)

# initial parameters
JA0 <- c(J=20, A=50)

# call the ode function
out <- ode(y = JA0, func = age_population, times = timevec, parms = P)

par(mfrow = c(1,2))
maxPop <- max(out[,3], out[,2]) * 1.2;
#ploting populations in time plots
plot(out[,1], out[,2], type='l', col='green', main = 'Time plot',
     xlab='Time', ylab='Populations J, A', lwd = 2, ylim= c(0,maxPop))
lines(out[,1], out[,3], col='red', lwd = 2)
legend('bottomright', legend = c('Juvenile','Adult'),
       lty=c(1,1), col=c('green','red'), lwd = c(2,2))

#ploting phase-plane plot (J vs. A)

plot(out[,3], out[,2], type='l', main = 'Phase plot',
     xlab = 'Adult population', ylab='Juvenile population',
     xlim = c(0,maxPop), ylim=c(0,maxPop), lwd = 2)

#add isoclines
isocline1 <- function(vec, P){
  line1 <- P$b*vec/(P$g+P$mu_j);
  line1;
}
isocline2 <- function(vec, P){
  line2 <- (P$mu_0*vec + P$mu_0 * P$c * vec**2) / P$g;
  line2;
}
vec <- seq(0,maxPop)
lines (vec, isocline1(vec, P), col = 'green', lty = 5)
lines(vec, isocline2(vec, P), col = 'red', lty = 5)

legend('bottomright', legend = c('trajectory', 'J isocline','A isocline'),
       col=c('black','green','red'), lwd = c(2,1,1), lty = c(1,5,5))
```

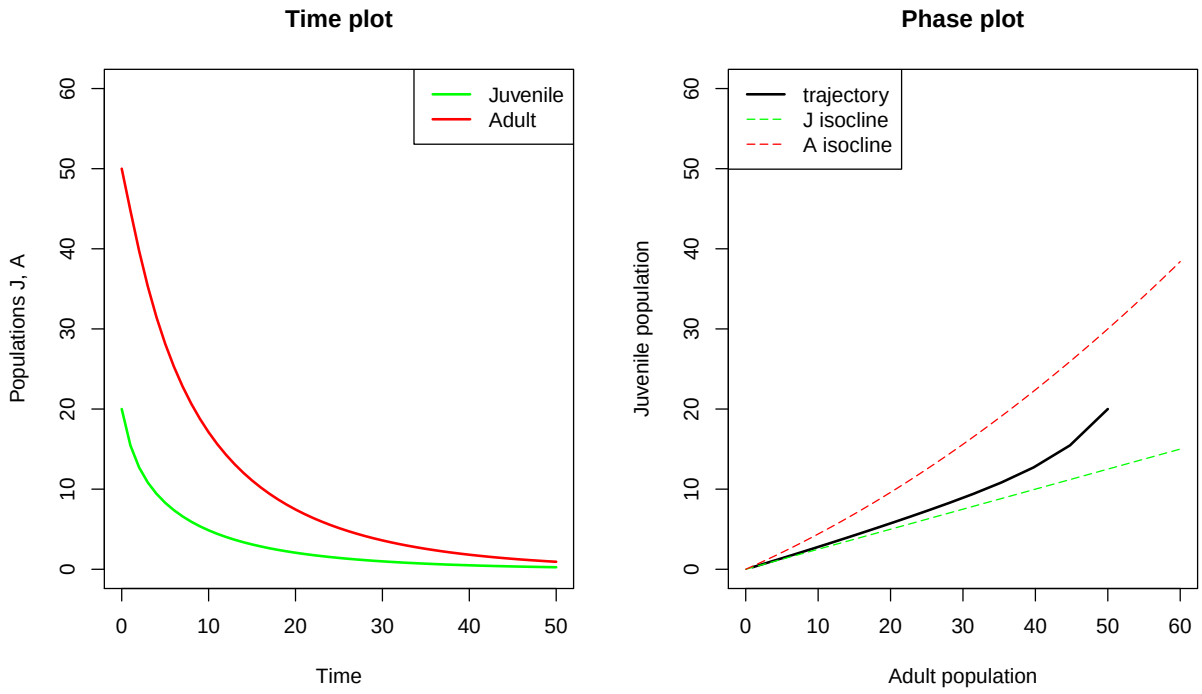


By running script with 20 Juveniles and 50 Adults as starting population - equilibrium is reached at time of about 30. Phase plot also shows that equilibrium found in a) is stable because trajectory stops on the iseclines crossing.

But this equilibrium is stable because of the parameters used (population is pretty happy - e.g. birth rate b is higher then death rate of juveniles μ_J). By decreasing birth rate below death rate - equilibrium does not exist in phase plot (positive values) - and population dies (trajectory goes to 0) :

```
library(deSolve)

age_population <- function(t, JA, P) {
  # extract vector content
  J <- JA[1]
  A <- JA[2]

  # calculate the growth rates:p
  dJdt <- P$b * A - P$g * J - P$mu_j * J
  dAdt <- P$g * J - P$mu_0 * (1 + P$c * A) * A

  # output
  list(c(dJdt, dAdt))
}

# set the vector of time-points for the output
timevec <- seq(0, 50, by=1)

# list of parameters
P <- list(b = 0.2, # deacrease of birth rate less then 0.3
          g = 0.5,
          mu_j = 0.3,
```

```
          mu_0 = 0.2,
          c = 0.01)

# initial parameters
JA0 <- c(J=20, A=50)

# call the ode function
out <- ode(y = JA0, func = age_population, times = timevec, parms = P)

par(mfrow = c(1,2))
maxPop <- max(out[,3], out[,2])*1.2
#ploting populations in time plots
plot(out[,1], out[,2], type='l', col='green', main = 'Time plot',
     xlab='Time', ylab='Populations J, A', lwd = 2, ylim= c(0,maxPop))
lines(out[,1], out[,3], col='red', lwd = 2)
legend('topright', legend = c('Juvenile','Adult'),
       lty=c(1,1), col=c('green','red'), lwd = c(2,2))

#ploting phase-plane plot (J vs. A)

plot(out[,3], out[,2], type='l', main = 'Phase plot',
     xlab = 'Adult population', ylab='Juvenile population',
     xlim = c(0,maxPop), ylim=c(0,maxPop), lwd = 2)

#add isoclines
isocline1 <- function(vec, P){
  line1 <- P$b*vec/(P$g+P$mu_j);
  line1;
}
isocline2 <- function(vec, P){
  line2 <- (P$mu_0*vec + P$mu_0 * P$c * vec**2) / P$g;
  line2;
}
vec <- seq(0,maxPop)
lines (vec, isocline1(vec, P), col = 'green', lty = 5)
lines(vec, isocline2(vec, P), col = 'red', lty = 5)

legend('topleft', legend = c('trajectory', 'J isocline','A isocline'),
       col=c('black','green','red'), lwd = c(2,1,1), lty = c(1,5,5))
```



f) Cannibalism extention

Extend the model to include cannibalism, i.e. that adults feed on the juveniles (a common behaviour among many fish, reptiles, and other groups). Motivate all model extensions! Feel free to make extra assumptions, if necessary. The Lotka-Volterra predator-prey equations may be a source of inspiration.

The easy way of implementing cannibalism is by simply adding one more factor for decreasing of Juveniles, which are the food for Adults, and influence Adult mortality, since with more food Adults should live happy and long life. Here is modified system of deferential equations for population with cannibalism:

$$\begin{cases} \frac{dJ}{dt} = bA - gJ - \mu_J J - kJA \\ \frac{dA}{dt} = gJ - \mu_0(1 + cA - skJA)A \end{cases}$$

, where:

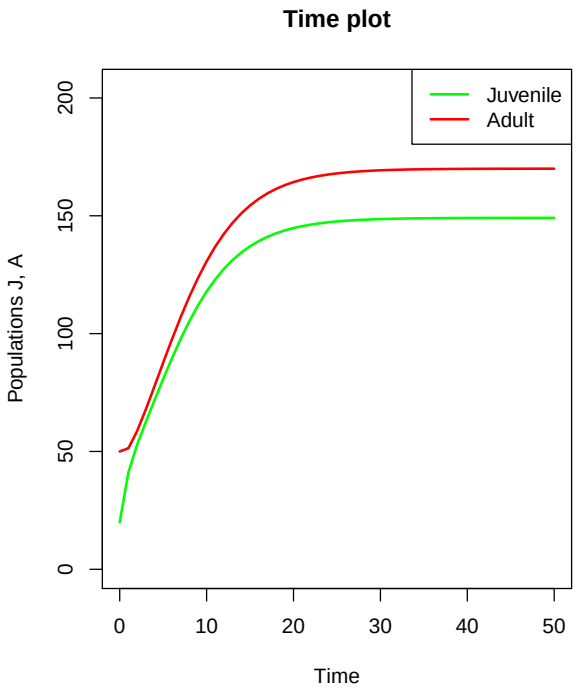
k – cannibalism rate. Means how many juveniles are eaten by adults per time unit. So overall cannibalism is proportional to density of both populations.

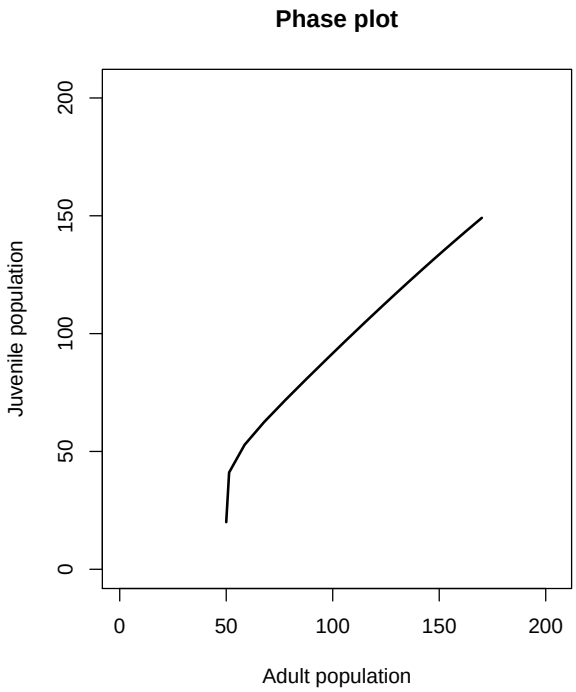
s – survival rate coefficient due to cannibalism. How cannibalism helps to decrease mortality of adults, since some of them die because of food absence.

So now with introduction of cannibalism model will look like:

```
library(deSolve)

age_population_cannibalizm <- function(t, JA, P) {
  # extract vector content
  J <- JA[1]
  A <- JA[2]
```

```
  # calculate the growth rates:
  #dndt <- P$r0 * n - P$a * n * p
  #dJdt <- P$b * A - P$g * J - P$mu_j * (1 + P$k * J * A) * J
  dJdt <- P$b * A - P$g * J - P$mu_j * J - P$k * J * A
  dAdt <- P$g * J - P$mu_0 * (1 + P$c * A - P$s * P$k * J * A) * A

  # output
  list(c(dJdt, dAdt))
}

# set the vector of time-points for the output
timevec <- seq(0, 50, by=1)

# list of parameters
P <- list(b = 1,
          g = 0.5,
          mu_j = 0.3,
          mu_0 = 0.2,
          c = 0.01,
          k = 0.002,
          s = 0.01)

# initial parameters
JA0 <- c(J=20, A=50)

# call the ode function
out <- ode(y = JA0, func = age_population_cannibalizm, times = timevec, parms = P)

par(mfrow = c(1,2))
maxPop <- max(out[,3], out[,2]) * 1.2;
#ploting populations in time plots
plot(out[,1], out[,2], type='l', col='green', main = 'Time plot',
     xlab='Time', ylab='Populations J, A', lwd = 2, ylim= c(0,maxPop))
lines(out[,1], out[,3], col='red', lwd = 2)
legend('topright', legend = c('Juvenile','Adult'),
       lty=c(1,1), col=c('green','red'), lwd = c(2,2))

#ploting phase-plane plot (J vs. A)

plot(out[,3], out[,2], type='l', main = 'Phase plot',
     xlab = 'Adult population', ylab='Juvenile population',
     xlim = c(0,maxPop), ylim=c(0,maxPop), lwd = 2)
```


Time plot
Juvenile
Adult
Populations J, A
0
50
100
150
200
0
10
20
30
40
50
Time


Phase plot
Juvenile population
0
50
100
150
200
0
50
100
150
200
Adult population