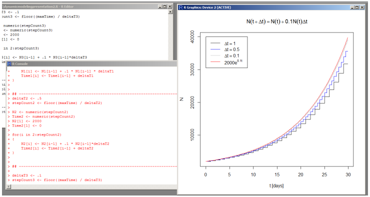
Dynamic Models - Showcase

Gunther Krauss

Crop Science Research Group

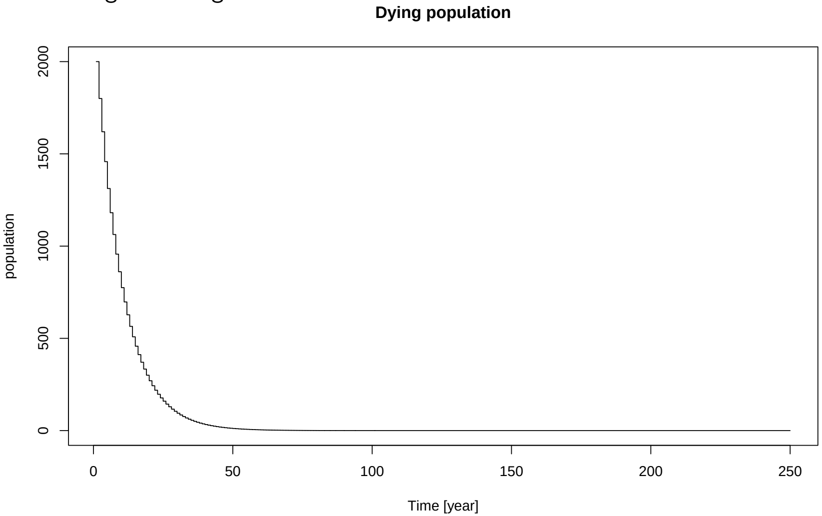
Institute of Crop Science and Resource Conservation (INRES)

January 18, 2022



Foxes in paradise

Foxes have to eat rabbits to survive. But in paradise this is a *no go*.
Therefore their population decreases each year by a constant rate: $\frac{\Delta N}{N} = -.1$.
Calculating in R we get



Real world rabbits

... meet and ~~eat~~ are eaten by foxes. Let `R` denote the number of rabbits and `F` the number of foxes, then we get the formulas:

$$\begin{aligned} \Delta R_n &= \quad .1R_n - .2R_nF_n \\ \Delta F_n &= -.1F_n + .15R_nF_n \end{aligned}$$

If we enter these formulas in our R loop (and do all the initialisation), we can calculate the number of rabbits and foxes over time.

Foxes and rabbits

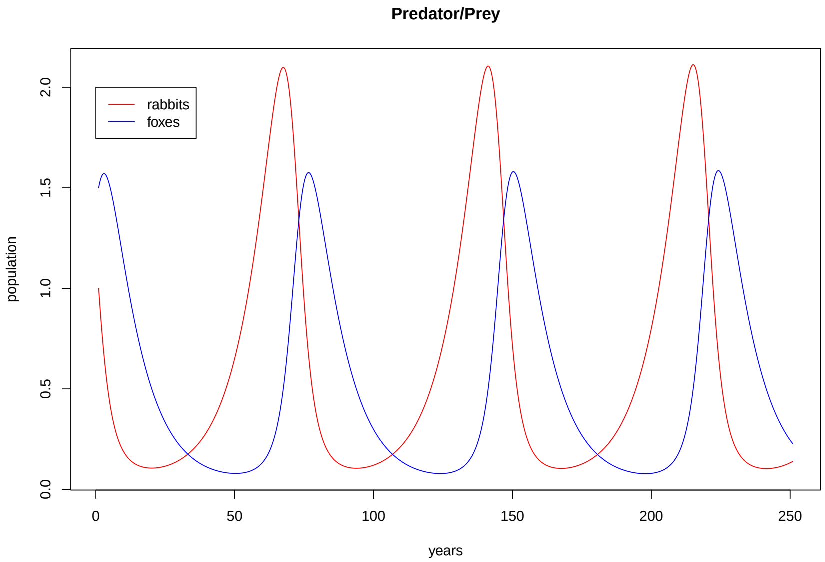
```r
EndTime <- 250
steps <- 100
A <- numeric(EndTime*steps)
B <- numeric(EndTime*steps)
deltaA <- numeric(EndTime)
deltaB <- numeric(EndTime)

A[1] <- 1.0
B[1] <- 1.5

for(i in 2:(steps*EndTime))
{
  deltaA[i-1] <-   .1 * A[i-1] - .20 *  A[i-1]*B[i-1]
  deltaB[i-1] <-  -.1 * B[i-1] + .15 *  A[i-1]*B[i-1]

  A[i] <- A[i-1] + deltaA[i-1]*1/steps
  B[i] <- B[i-1] + deltaB[i-1]*1/steps
}
```

The plot


Predator/Prey
rabbits
foxes
population
years
0.0
0.5
1.0
1.5
2.0
0
50
100
150
200
250

Not part of the exercises - just for your information:

Fibonacci numbers as a dynamic model?

Fibonacci numbers as a dynamic model?

From the Fibonacci formula

$$t_{n+1} = t_n + t_{n-1}$$

we get

$$\Delta t_n = t_{n+1} - t_n = t_{n-1}$$

Difference to our dynamic models:

- Daily increment depends on the state of the day before

Fibonacci numbers as a dynamic model?

From the Fibonacci formula

$$t_{n+1} = t_n + t_{n-1}$$

we get

$$\Delta t_n = t_{n+1} - t_n = t_{n-1}$$

Difference to our dynamic models:

- Daily increment depends on the state of the day before
- We need initial values for first two days

In the rabbit model, the ratio between rate and state is constant $\frac{\Delta N}{N} = c$.

Let's look at the ratio for fibonacci numbers:

$$\frac{\Delta t_n}{t_n} = \frac{t_{n-1}}{t_n}$$

We calculate this ratio by R:

```r
t[1:(m-1)]/t[2:m]
```

```
#  [1] 1.0000000 0.5000000 0.6666667 0.6000000 0.6250000 0.6153846 0.6190476
#  [8] 0.6176471 0.6181818 0.6179775 0.6180556 0.6180258 0.6180371 0.6180328
# [15] 0.6180344 0.6180338 0.6180341 0.6180340 0.6180340 0.6180340 0.6180340
# [22] 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340
# [29] 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340
# [36] 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340
# [43] 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340 0.6180340
```

Fibonacci numbers approximated by dynamic model

For the fibonacci numbers above stage 18, the ratio becomes nearly constant 0.618034.

For higher stages, we can calculate the fibonacci numbers as a dynamic model:

$$t_{19} = 4181$$

$$\Delta t_n \approx 0.618034 t_n \quad \text{for} \quad n \geq 19$$

By the way: italian mathematician Fibonacci used his numbers to calculate rabbit population.

Conclusion: for small numbers / stages, Fibonacci numbers are different from our dynamic models. For higher stages / numbers, the series become similar.

Fibonacci numbers as a system of difference equations

To overcome the problem that the rate of Fibonacci numbers depend on the previous day's value, we do a trick by declaring a new state variable, that represents the previous day's value. We define $s_i := t_{i-1}$ and rewrite the formula as $t_{i+1} = t_i + t_{i-1} = t_i + s_i$.

Then we have

$$\Delta t_i = t_{i+1} - t_i = t_i + s_i - t_i = s_i$$

and

$$\Delta s_i = s_{i+1} - s_i = t_i - s_i$$

with $t_1 = 1$ and $s_1 = t_0 = 0$. The differences depend now formally only on the state of the day, not on the day before, so that we can use the Euler method.

Calculating as a dynamic system

```
EndTime <- 50
s <- numeric(EndTime)
t <- numeric(EndTime)
deltas <- numeric(EndTime)
deltat <- numeric(EndTime)

s[1] <- 0
t[1] <- 1

for(n in 2:EndTime)
{
  deltas[n-1] <- t[n-1] - s[n-1]
  deltat[n-1] <- s[n-1]

  s[n] <- s[n-1] + deltas[n-1]
  t[n] <- t[n-1] + deltat[n-1]

}
```

Show the result

```
t
```

```
#  [1]          1           1          2          3          5          8
#  [7]         13          21         34         55         89        144
# [13]        233         377        610        987       1597       2584
# [19]       4181        6765      10946      17711      28657      46368
# [25]      75025      121393     196418     317811     514229     832040
# [31]    1346269     2178309    3524578    5702887    9227465   14930352
# [37]   24157817    39088169   63245986  102334155  165580141  267914296
# [43]  433494437   701408733 1134903170 1836311903 2971215073 4807526976
# [49] 7778742049 12586269025
```