Simulating the decrease of surname number in a constant, increasing and decreasing population

Costanza Polastri

June 2018

Contents

Abstract

The goal of this project is to reproduce the extinction of surnames in a population with a fixed fertility rate.

The code will be analized in detail in the *Model and code* chapter.

The emerging property of the system is an exponential decrease in all cases, as we expect from the mathematical models discussed in [1]. We found however that changing some parameters of the simulation leads to a change in how deep the exponential becomes, as it will be discussed in the *Results* chapter.

Chapter 1

Introduction

[1] studied the surname distributions of a population with fixed and exponentially growing number, and proposed a mathematical model for the decrease of surnames.

Inspired by this work, the present study aims to reproduce some of the results through a Netlogo simulation.

First, the simulation is set up with a number of turtles that varies from 2000 to 10000. The turtles are assigned a random position, a random gender, *male* or *female*, and a random surname among 25 possible ones.

Then, each female turtle is asked to find a male partner to mate with, within a radius r controlled by a slider. The switch *Inbreeding* control the possibility of two turtles with the same surname mating. To each surname is then assigned a color, so that extinction and migration of a family becomes much more easy to visualize.

The two turtles now can generate an offspring. The number of new turtles created follows a Poisson distribution which peaks at *fertility-rate*. This is in contrast with [1], which obtains an exponential law.

Since immediately after reproducing the two parents are asked to die, there is no need for a mortality rate in this model. This constitutes a considerable difference from the approach taken in [1].

The slider *migration-rate* controls the distance every turtle covers when moving after every tick. If we choose *migration-rate* greater, equal or smaller than r we obtain different emerging migration patterns, highlighted by the different colors.

The mutation of surnames is implemented by changing one random letter in the surname of one random turtle. At each tick, the probability of this happening is of $\frac{1}{mutation-rate}$ and therefore quite rare. The procedure *count-surnames* is not capable to count surnames different from the ones the simulation starts with, and therefore the effect of surnames mutation is ignored during data collection.

A possible extension of this project could be to better implement the *count-surnames* procedure and account for mutations, but for the present moment this effect will be ignored.

Finally, the number of surnames does not always display the exponential decrease that would be consistent with data in [1]. This can probably be attribuited to a number of factors, mainly:

- Small population sample, due to limited computing capacity
- In particular, there was not a possibility to simulate a high birth rate due to the simulation becoming extremely slow after just a small number of cycles
- Mortality rate and birth rate were implemented in the same parameter
- No external conditions were implemented, while in the real world events like a mass migration, a war or even just a quick change in the birth rate are quite common[1]

Still, this project provides a quick visualization of small migration processes within a fixed population. It also allows the extinction of surnames to manifest itself completely as an emerging property.

[1] We reference[1]

Chapter 2

Model and code

We will now look in detail at the code.

2.1 Setup

```
to setup
  clear-all
  reset-ticks
  create-turtles number-of-turtles [
    set shape "person"
    setxy random-xcor random-ycor
    set-gender
    set-surname
    set birth-tick ticks ]

end
```

The initial number of turtles is decided by the slider *number-of-turtles* and can go from 2000 up to 10000[1].

When created, the turtles are assigned:

- a random *xy* position
- a random string between *female* and *male* to represent gender
- a random surname among 25 possible ones, together with its corresponding color code[2].

[1]Note that since the device I could run the simulation on has quite limited computing capacity, any simulation with a total number of turtles above 8000 would have made the process extremely slow and was therefore avoided.

[2]The color makes the visualization of small migration events much easier but has no effect on the data collection.

```
to set-gender

  let choose-gender random 2
  if choose-gender = 0 [set gender "female"]
  if choose-gender = 1 [set gender "male"]

end
```

```
to set-surname

  let numstring random (25) + 1
  if numstring = 1 [set surname "aaaa" set color 5]
  if numstring = 2 [set surname "bbbb" set color 15]
  if numstring = 3 [set surname "cccc" set color 25]
  if numstring = 4 [set surname "dddd" set color 35]
  if numstring = 5 [set surname "eeee" set color 45]
  if numstring = 6 [set surname "ffff" set color 55]
  if numstring = 7 [set surname "gggg" set color 65]
  if numstring = 8 [set surname "hhhh" set color 75]
  if numstring = 9 [set surname "iiii" set color 85]
  if numstring = 10 [set surname "jjjj" set color 95]
  if numstring = 11 [set surname "kkkk" set color 105]
  if numstring = 12 [set surname "llll" set color 115]
  if numstring = 13 [set surname "mmmm" set color 125]
  if numstring = 14 [set surname "nnnn" set color 135]
  if numstring = 15 [set surname "oooo" set color 109]
  if numstring = 16 [set surname "pppp" set color 119]
  if numstring = 17 [set surname "qqqq" set color 29]
  if numstring = 18 [set surname "rrrr" set color 39]
  if numstring = 19 [set surname "ssss" set color 49]
  if numstring = 20 [set surname "tttt" set color 59]
  if numstring = 21 [set surname "uuuu" set color 69]
  if numstring = 22 [set surname "wwww" set color 79]
  if numstring = 23 [set surname "xxxx" set color 89]
  if numstring = 24 [set surname "yyyy" set color 99]
  if numstring = 25 [set surname "zzzz" set color 139]

end
```

2.2 The *make-babies* procedure

```
to go

  move-turtles
  ask turtles with [gender = "female"] [make-babies]
  count-surnames
  if mutation = true  [ ask turtles [decide-mutation]]
```

```
  bargraph-surname
  tick

end
```

In the *go* procedure, all female turtles are asked to execute the *make-babies* procedure.

First, they find a male partner within a radius r, set by a slider. If the switch *Inbreeding* is *off*, the potential male partner is required to have a different surname than the female partner, otherwise any male will do.

New turtles are created, and they have the following characteristics:

- A random gender
- The surname of the male turtle their mother chose and its corresponding color
- A birth date *ticks* which is not used in this project, but can be very useful if someone wants to implement some mortality rate

The number of offsprings is decided through a random Poisson distribution with mean on *fertility-rate*, controlled by a slider [3].

The two parents are then asked to die. This simplifies the model, since birth rate and mortality rate are de facto incorporated in the same parameter.

```
to make-babies

  let dad one-of other turtles
  ifelse inbreeding = true
  [
    set dad one-of other turtles with [gender = "male" and  distance
    myself < radius]
  ]
  [
    set dad one-of other turtles with [gender = "male" and  distance
    myself < radius and surname != [surname] of myself]
  ]

if dad != nobody
  [  set generation "old"
     ask dad [ set generation "old"]

      hatch random-poisson fertility-rate

      [ set-gender
        set surname [surname] of dad
        set color [color] of dad
```

[3]Note that if the slider is significantly higher than 2.00, say above 2.50, the simulation becomes quite heavy very fast and it was therefore not possible to run it. The reader is encouraged to try the simulation on a higher performing computer.

```
        set shape "person"
        set generation "new"
        set birth-tick ticks
      ]
  ]

  ask turtles with [generation = "old"] [die]

end
```

2.3 Mutation of surnames

The mutation of surnames was not taken into account during the collection of data, since its effects were ignorable most of the time. However, since it constitutes a great occasion for expanding the project, it's worth looking at the *decide-mutation* procedure.

```
to decide-mutation

let probability-of-mutation random (mutation-rate)

if probability-of-mutation = 1 [mutate]

end
```

First, the procedure *decide-mutation* is called. This procedure first generates a random number between 1 and *mutation-rate*, which is controlled by a slider.

Then if the number generated is equal to 1, the procedure *mutate* is called and the mutation happens.

```
to mutate

let random-letter "0"
let random-num random (25) + 1
if random-num = 1 [set random-letter "a"]
if random-num = 2 [set random-letter "b"]
if random-num = 3 [set random-letter "c"]
if random-num = 4 [set random-letter "d"]
if random-num = 5 [set random-letter "e"]
if random-num = 6 [set random-letter "f"]
if random-num = 7 [set random-letter "g"]
if random-num = 8 [set random-letter "h"]
if random-num = 9 [set random-letter "i"]
if random-num = 10 [set random-letter "j"]
if random-num = 11 [set random-letter "k"]
if random-num = 12 [set random-letter "l"]
if random-num = 13 [set random-letter "m"]
if random-num = 14 [set random-letter "n"]
```

```
19  if random-num = 15 [set random-letter "o"]
20  if random-num = 16 [set random-letter "p"]
21  if random-num = 17 [set random-letter "q"]
22  if random-num = 18 [set random-letter "r"]
23  if random-num = 19 [set random-letter "s"]
24  if random-num = 20 [set random-letter "t"]
25  if random-num = 21 [set random-letter "u"]
26  if random-num = 22 [set random-letter "w"]
27  if random-num = 23 [set random-letter "x"]
28  if random-num = 24 [set random-letter "y"]
29  if random-num = 25 [set random-letter "z"]
30
31
32  ask one-of turtles
33
34  [
35  let position-to-change random (3)
36  if position-to-change = 0 [set surname replace-item 0 surname random-
      letter]
37  if position-to-change = 1 [set surname replace-item 1 surname random-
      letter]
38  if position-to-change = 2 [set surname replace-item 2 surname random-
      letter]
39  if position-to-change = 3 [set surname replace-item 3 surname random-
      letter]
40  ]
41
42  end
```

The *mutate* asks one of the turtles to replace one random letter of its surname with a random letter previously generated.

The reasons why the effects of this procedure were ignored during the data collection were two:

- Since the probability of just one single mutation happening is itself $\frac{1}{mutation-rate}$, and even if it does happen it will affect only a single turtle among several thousands of agents. The effects were judged negligible.
- It would have been impractical to count surnames in a number that was constantly increasing, and it would have made the graphs more difficult to interpret.
- Since the probability is $\frac{1}{mutation-rate}$, a higher number in the switch actually leads to a lower probability of mutation happening. The model, as it is right now, is counterintuitive and needs further implementation.

The switch *mutation* is, for the previous reasons, kept *off* for all the simulations discussed in this project. The reader is encouraged to switch it back *on* and see how it affects the results, especially in combination with the *inbreeding* switch.

2.4 Filling the graphs

We will now discuss the procedures that fill the graphs.

The project displays 3 graphs with the following informations:

- A count of the total number of turtles, which can vary quite a lot even with a 2.00 fertility rate, so is worth keeping track of
- A count of the total amount of surnames, which displays most of the time an exponential decrease as an emerging property
- A bar graph with a count of all the individual surnames, worth discussing more in details

```
1
2 to count-surnames
3
4  set surname-list [
5  "aaaa" "bbbb" "cccc" "dddd" "eeee"
6  "ffff" "gggg" "hhhh" "iiii" "jjjj"
7  "kkkk" "llll" "mmmm" "nnnn" "oooo"
8  "pppp" "qqqq" "rrrr" "ssss" "tttt"
9  "wwww" "xxxx" "yyyy" "zzzz"
10 ]
11
12 set number-of-surnames 0
13 foreach surname-list [x -> if any? turtles with [surname = x] [set number
     -of-surnames number-of-surnames + 1]  ]
14
15 end
```

First, the total number of surnames is calculated: the procedure runs through all the initial surnames, listed in *surname-list*, and simply increments the global variable *number-of-surnames* whenever it finds at least one turtle for each surname.

```
1  to bargraph-surname
2
3  set surname-list [
4  "aaaa" "bbbb" "cccc" "dddd" "eeee"
5  "ffff" "gggg" "hhhh" "iiii" "jjjj"
6  "kkkk" "llll" "mmmm" "nnnn" "oooo"
7  "pppp" "qqqq" "rrrr" "ssss" "tttt"
8  "wwww" "xxxx" "yyyy" "zzzz"
9  ]
10
11 let  counts 0
12 let countslist []
13 foreach surname-list [
```

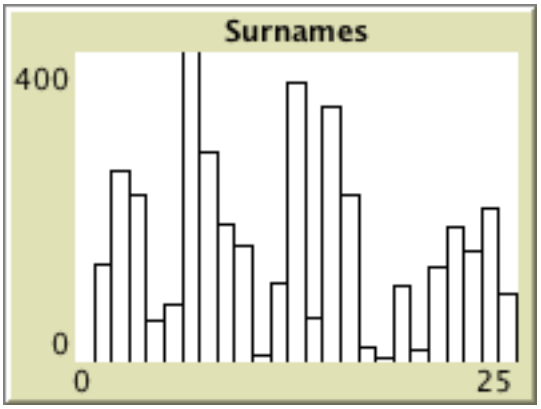

Figure 2.1: The bar graph of surnames.

```
14 x -> set counts count turtles with [surname = x]
15 set countslist lput counts countslist
16 ]
17
18 set-current-plot "Surnames"
19 clear-plot
20 set-current-plot-pen "bar"
21
22 let i 1
23 foreach countslist [y -> plotxy i y      set i i + 1]
24
25 end
```

Then, for each surname in the global list *surname-list* the procedure counts how many turtles have that surname, and puts the value in another list.

The graph *Surnames* is the filled with both lists. An example of how it looks at around tick 50 is shown in figure 2.1.

Chapter 3

Results

We will now look at the results of the simulation with different sets of parameters.

Before discussing the exponential decreases of the surname total number, we will show some examples of graphs of the number of turtles, to point out that even with a fixed fertility rate of 2.00 the number of turtles can increase or decrease up to 3000 more or less than the initial number.

Note that all simulations are run up to tick 1100.

3.1 The number of turtles

The *fetility-rate* switch controls a variable that determines how many babies will the female turtles *of an entire generation* have, not just a single couple. This makes the model much more quick to run while still being accurate, but if a few consecutive cycles happen to have a quite low fertility rate, the population can quickly be cut in half, because the entire population is having just 1 kid per female, not just one couple.

This explains the great variation in the population number even when the fertility rate is set to 2.00 and should therfore keep the population constant. As shown in figure 3.1, the population can decrease significantly, have high and low fluctuations or remain comparatively constant throughout the simulation.

3.2 The migration distance

The *migration-rate* slider controls how many steps the turtles will advance after reproducing. When the *radius* is kept constant at 1.0 it is interesting to see how different migration distances influence the emerging property of the population.

Case 1: *migration-rate* slightly greater than *radius*.

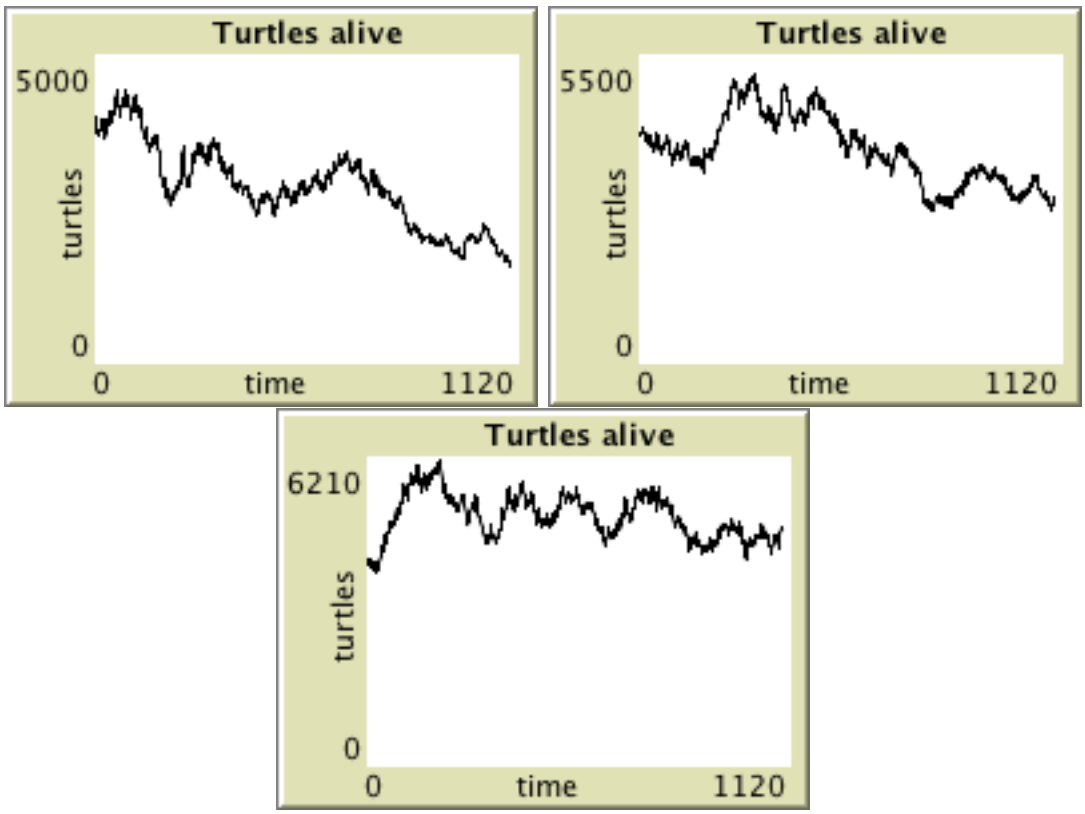

Figure 3.1: Different population counts with a fertility rate of 2.00.

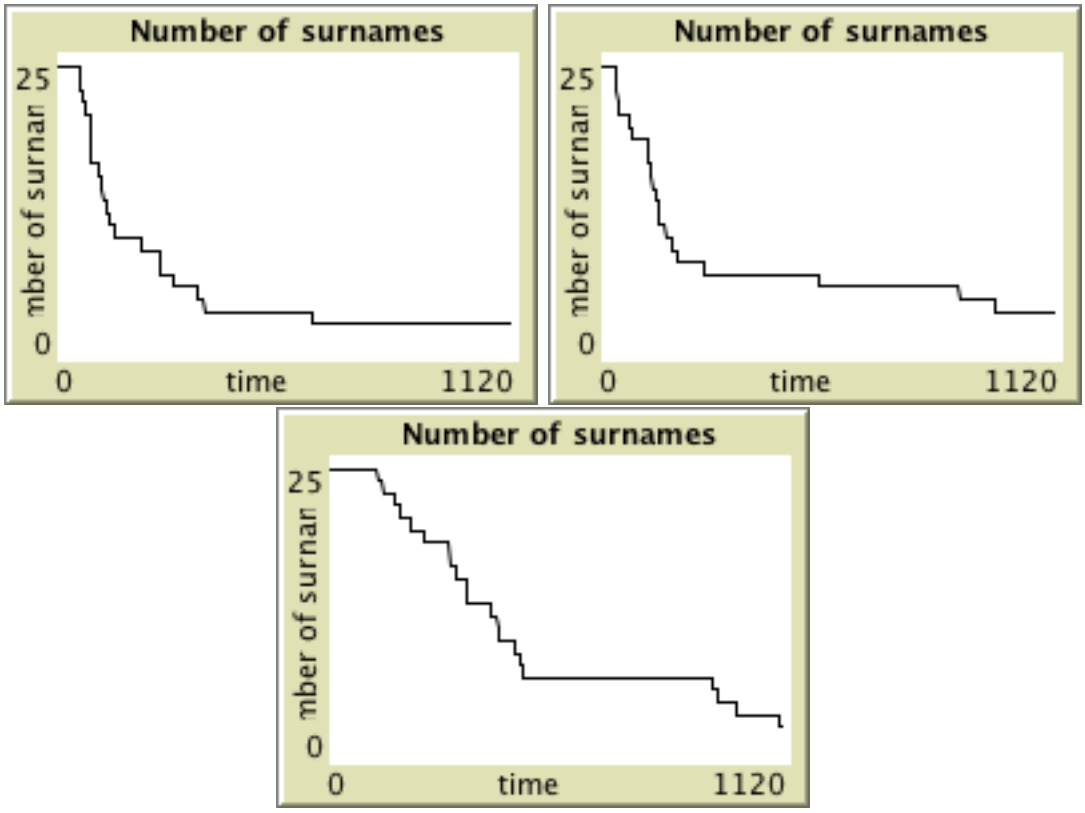

Figure 3.2: *migration-rate = 2.5*

In this the exponential decrease was comparatively visible in most cases, as shown in figures 3.2.

Case 2: *migration-rate* and *radius* are identical.

In this case the exponential decrease is less evident, to the point of being mistaken for a linear or polinomial decrease if the simulation is not run for enough ticks, see figures 3.3.

Case 3: *migration-rate* is much greater than *radius*.

In this case the exponential decrease emerges easily in all cases, see figures 3.4.

3.3 The *Inbreeding* switch

Activating the possibility for turtles to pick a partner with the same surname makes the chances of a surname going extinct lower. With the inbreeding switch turned *on* then, the exponential decrease becomes less evident and in some case the graph resembles a linear or polinomial decrease. See figure 3.5.

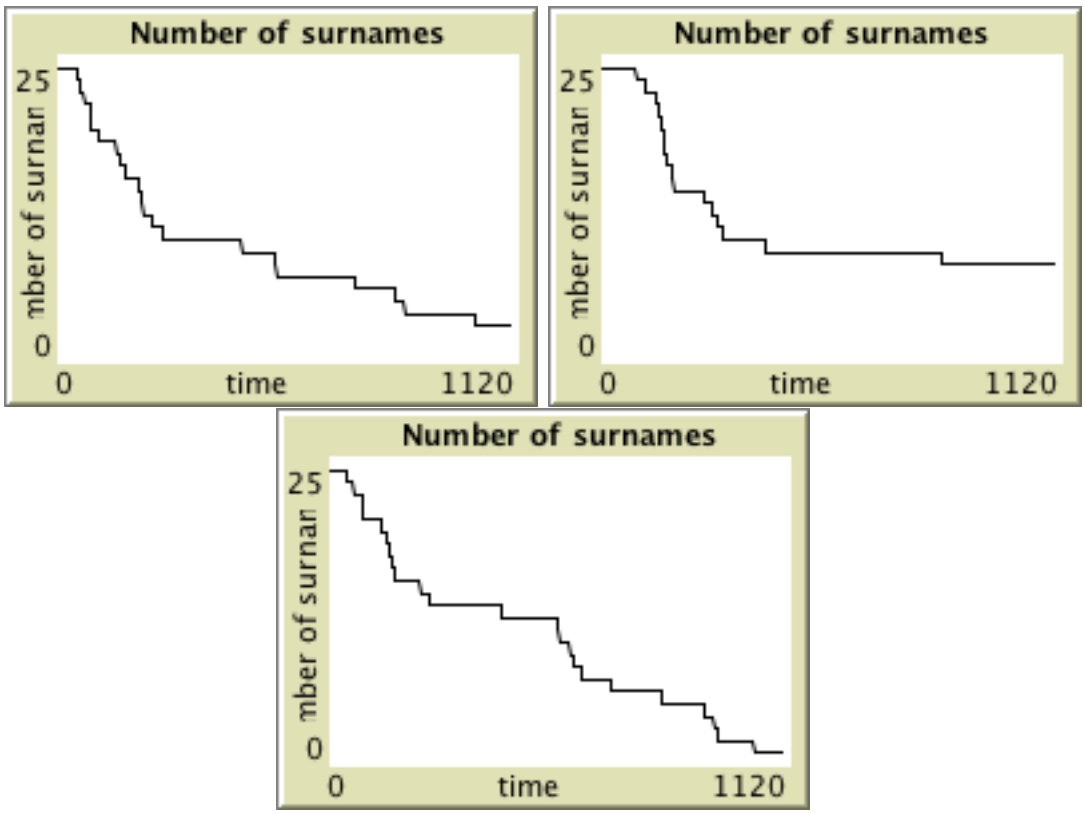

Figure 3.3: *migration-rate = 1.0*

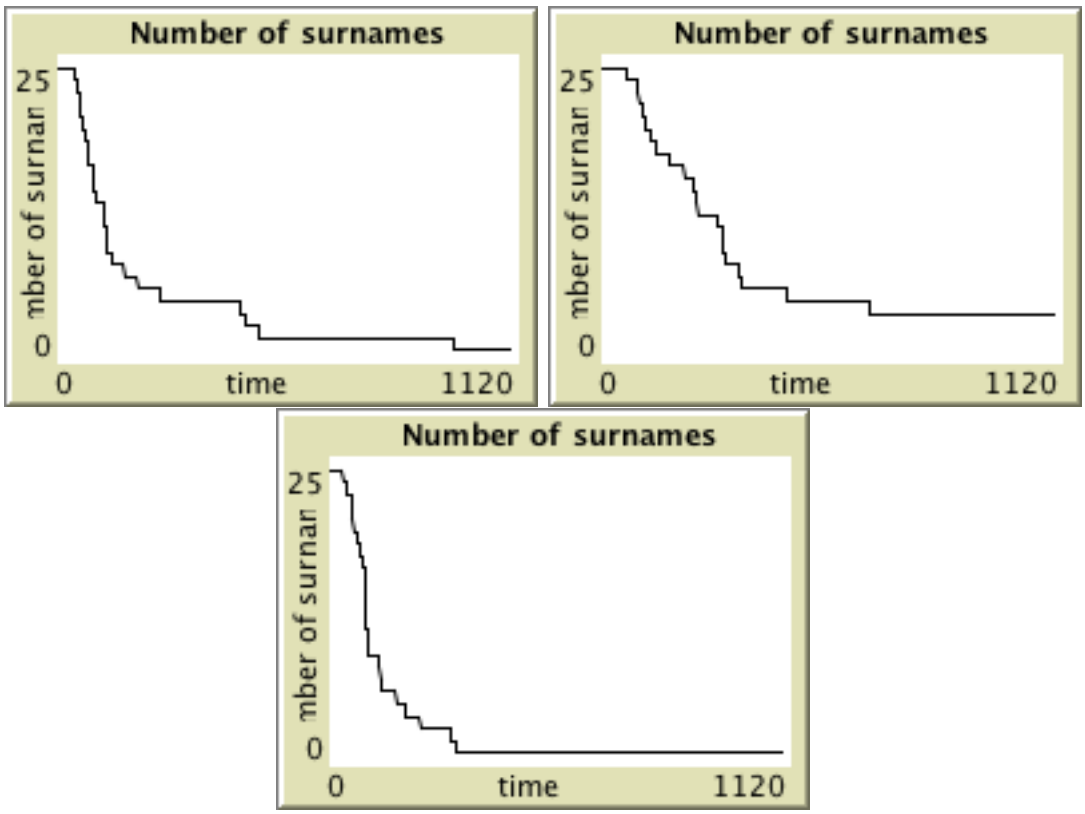

Figure 3.4: *migration-rate = 5.0*

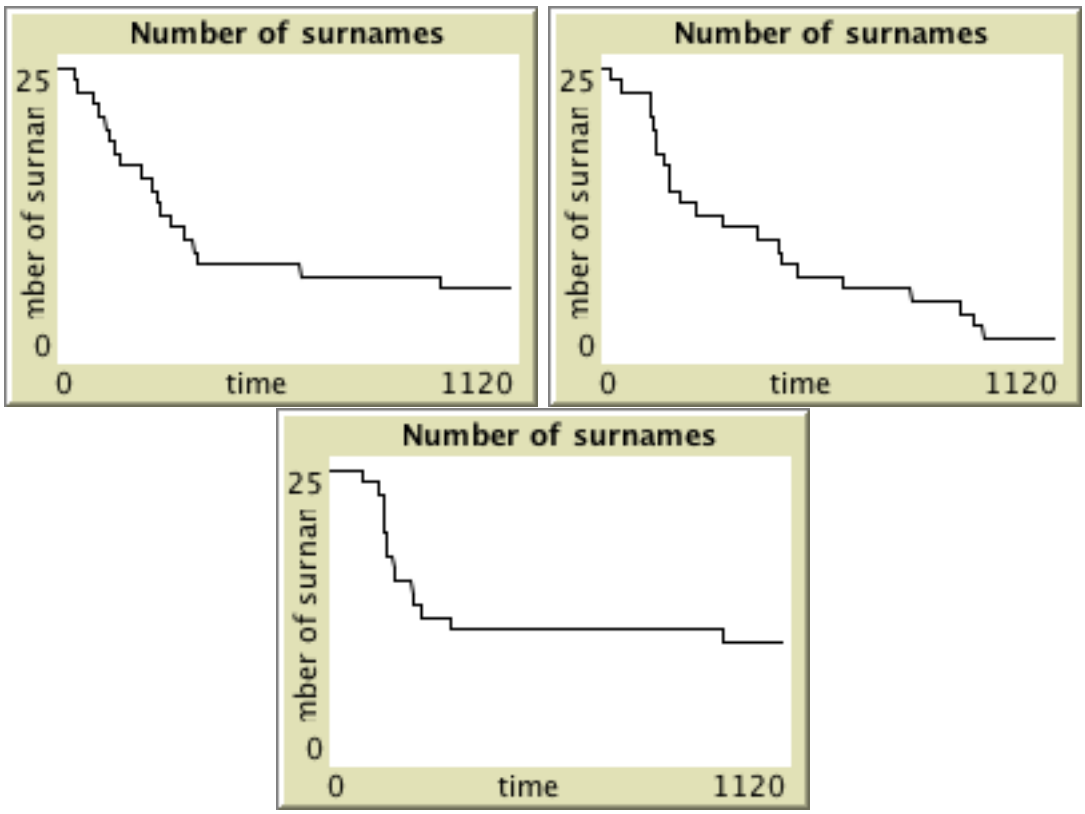

Figure 3.5: *Inbreeding* switch turned *on*.

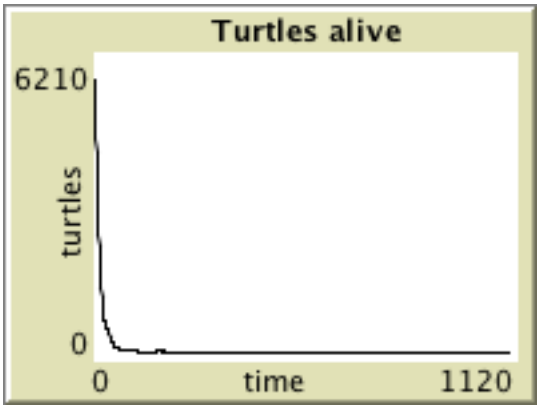

Figure 3.6: Population decrease with a fertility rate of 1.85.

3.4 Descreasing population

When we set the *fertility-rate* switch at 1.85, the population experiences a deep exponential decrease, as shown in figure 3.6.

The exponential decrease of the number of surnames is therefore very deep itself and reaches 1 very quickly in a short amount of ticks when compared to other simulations. However, in some cases a few surnames reach an equilibrium and the number of surnames stops approaching zero. See figure 3.7.

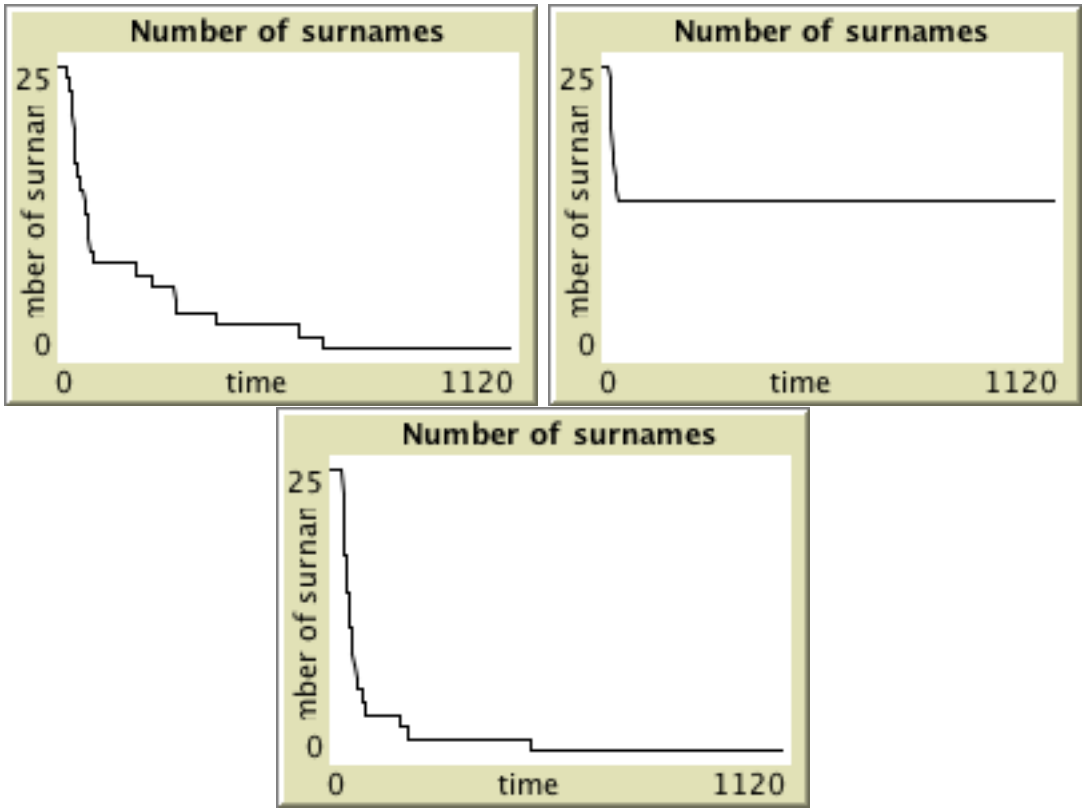

Figure 3.7: Two cases of deep exponential decrease and one of equilibrium between a few surnames with decreasing population.

Bibliography

Susanna C. Manrubia, Damian H. Zanette, *At the boundary between biological and cultural evolution: The origin of surname distributions*, 2002.