Modeling and simulating malaria

Daan Meijers, Steven Raaijmakers

March 2017

1 Model definition and implementation

Malaria is one of the most deadly diseases in the world. The model created for this assignment is used to simulate the spread of this disease using a 2 dimensional cellular automata. In this automata every cell can contain a single human and multiple mosquito's. During each time step a mosquito will either move along the grid or bite a human when one is present. This biting can result in the transmission of the disease from mosquito to human or vice verse. When a human is infected it will experience an incubation period of 14 days after which it will either die or become immune for the disease. In order to maintain the population in this simulation every death will trigger a birth.

Classes and Functions

In order to create a functioning model two classes and a couple functions have been added to the provided framework. These classes are used to store data for mosquito's or humans and each contain some functions for managing these classes.
The `Human` class contains three n by m arrays for a model with size n by m. These arrays contain the state of the humans, the age of the humans and the time to live. The possible states are: 0-dead/empty, 1-susceptible, 2-infected and 3-immune. The age of a human is in the building phase a random generated integer between 0 and 200 in order to keep a dynamic simulation. Each new human generated during the simulation will have an age-timer of 200. The ttl (or incubation timer) is a timer which counts down from 14 to 0, after which the human either dies or becomes immune. Besides a `build()` function, which initializes these arrays, the human class also contains three other functions, of which the most important one is the `update()` function. The `update()` function loops over all the humans, decrementing the age-timer and if the human is infected it will also decrement the ttl-timer. After this it checks whether the ttl-timer reached 0 and it will decide whether this human dies or becomes immune. This function concludes with checking whether the age-timer reached 0, which would also result in the death of this human. When a human dies the `death()` function is called which will reset the cell and call the `birth()` function. This function will then create a new human with state 1 (susceptible) at a random

empty location.
For each mosquito a separate instance of the `Mosquito` class is made. This class holds the coordinates of a mosquito, a Boolean whether it is infected or not and a timer which indicates the hungriness. This class contains one other function besides the `initialization()` function which is the `walk()` function. This function moves the mosquito to another cell by incrementing or decrementing the x and y coordinate-values at random.

Program Flow

Each simulation starts with a build function, which starts with calculating the amount of mosquito's to generate according to the `mosquito` parameter. This amount is equal to the `mosquito` parameter multiplied by the amount of cells. Then a list of `Mosquito` instances is created and a single instance of the `Human` class.
After the building phase the simulation begins by performing the `step()` function for each time step. The `step()` function start with looping over all the `Mosquito` instances. A mosquito will either walk or bite depending on whether it share a cell with a human an various other variables. The following table shows the outcomes of all possible situations.

Table 1: Human vs Mosquito outcome

Human State	Mosquito Hungry	Mosquito Infected	Result
na	No	na	Mosquito Walks
0	na	na	Mosquito Walks
1	Yes	Yes	Human Becomes Infected
1/3	Yes	No	Mosquito Eats
2	Yes	No	Mosquito becomes infected
2/3	Yes	Yes	Mosquito Eats

Note: Other than resetting the hunger-timer the Mosquito eating has no effect.

As can be seen from the graph above humans can only become infected when they are in state 1 (susceptible) and the mosquito is both infected and hungry. An empty cell or not hungry mosquito will result in a walk and an infected human and a non-infected, hungry mosquito will result in that mosquito becoming a new carrier.

After all mosquito's have acted the `Human` class will update itself.
The implemented prevention method is a mosquito net, which reduces the chance of being successfully bitten to the given parameter value.

2 Fitting the model parameters

The used parameters can be viewed in the following table:

Table 2: Parameters

Name	Value
height	100px
width	100px
humans	70%
mosquitos	60%
m_infected	50%
p_mosquito_human	30%
p_human_mosquito	90%
has_mosquito_net	0%
humans_of_age_under_15	40%

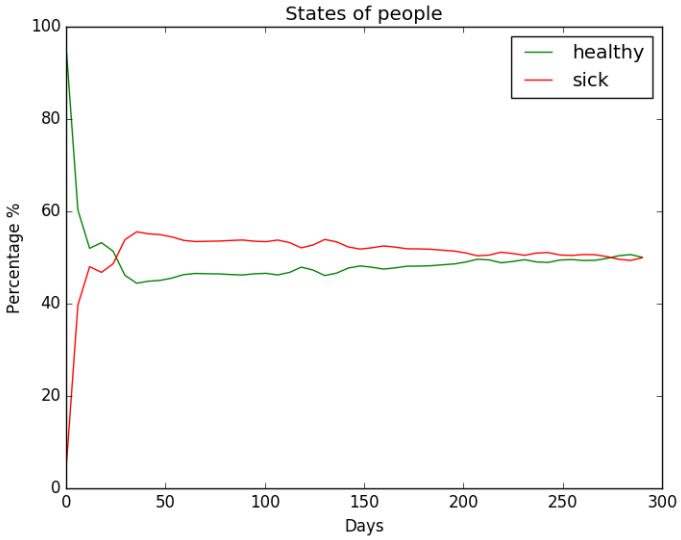
humans refers to the percentage of humans on the field created by the height * width, same goes for *mosquitos*. *m_infected* is the percentage of those mosquito's carrying malaria. *p_mosquito_human* describes the probability of a mosquito becoming a carrier of malaria by biting a human who already had malaria. *p_human_mosquito* is the probability of a human getting infected with malaria after he's been bitten by an infected mosquito[1].

In the information found at internewskenya (`www.internewskenya.org/dataportal/assets/downloads/xls/KMIS2010_final.xlsx`) we can see various scenario's described. We can see the percentage of infected children is the highest at the lake situation, where more than 50% of all interviewed children are infected. This is the model we're trying to recreate, and since this model only describes children we need to find the ratio between children and adults. Which is approximately 40% [2]. After running some simulations we noticed that the impact of the age that humans get assigned on initialisation has little to no effect on the prevalence, this may be due to the fact that age and the prevalence are probably not correlated.

The other parameters were found using an algorithm which runs the simulation for $t = 300$ days. We have chosen 300 because after running multiple simulations we have found that the percentages are stabilizing after 200 days. For each day we know the sick and healthy people, thus we know the prevalence. If the prevalence of the stable part is near 50%: we've found the correct parameters, otherwise the simulation will be run again with different (random) parameters.

The rounded parameters we have found produced the graph in figure 1.

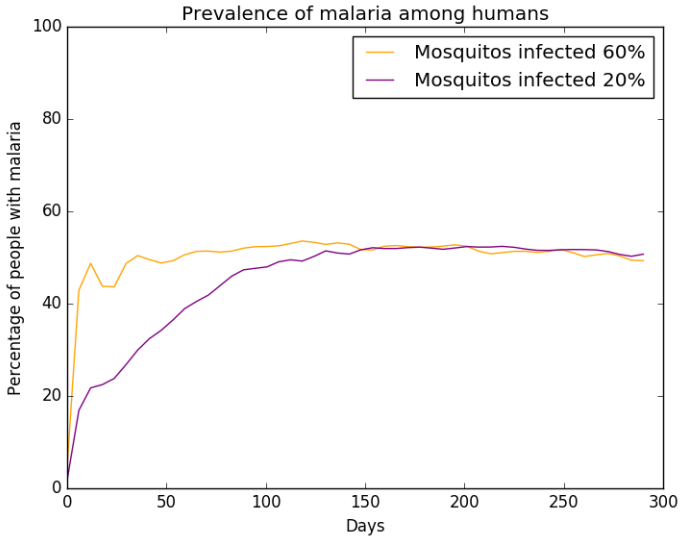
Figure 1: Graph of states of people, using our found parameters



3 Experiments and analysis

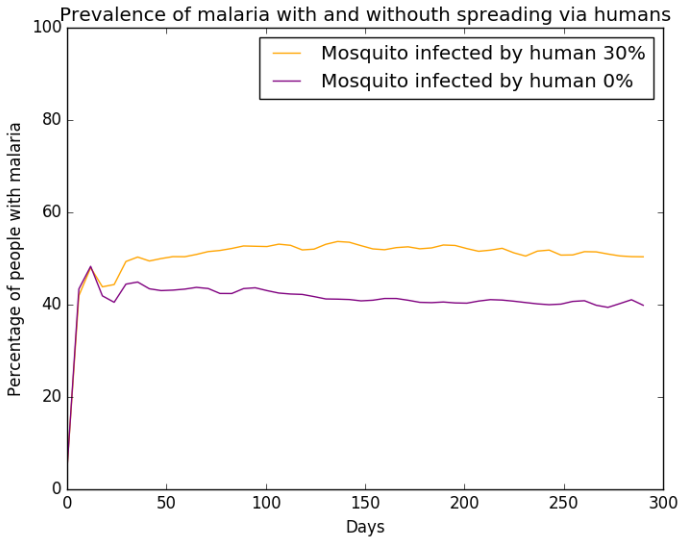
The following experiments where made by adjusting just one parameter, and using the previously discussed values.

Figure 2: Prevalence of malaria by changing the amount of initial infected mosquitos



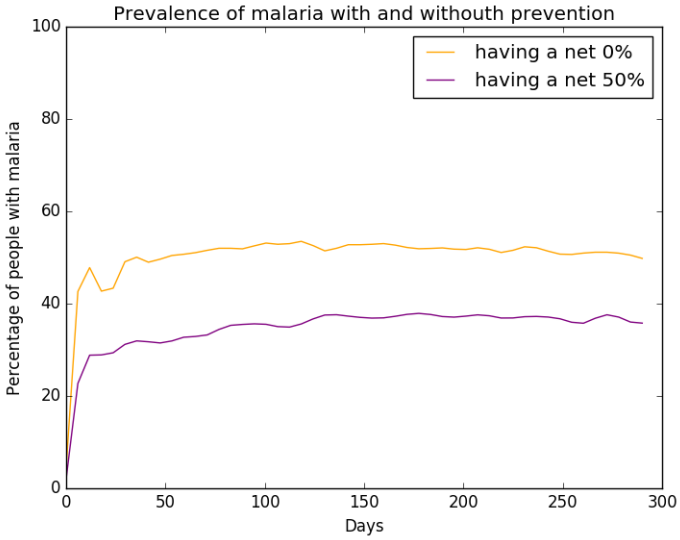
In the first experiment (figure 2) we reduced the the amount of infected mosquito's from 60% to 20%. As seen in the figure the virus takes much longer to spread in the experiment using the 20% but after a while it reaches the same stable state.

Figure 3: Prevalence with and without probability of mosquito getting infected by human



In the second experiment (figure 3) we changed the **probability** of a mosquito becoming a carrier of malaria after it bites an infected human. We see the same rise at the start but after a few days the prevalence of malaria is much lower. This probability (p_mosquito_human) can be changed in real life by handing preventive pills to the people.

Figure 4: Prevalence of malaria with mosquito nets



In the last figure (figure 4) it's clearly shown what the impact of mosquito's net is to the prevalence (of malaria among humans). It takes a lot more days for the virus to spread, and reaches a (somewhat) stable state which is much lower in comparison to the prevalence without prevention.

References

[1] JK Baird DL Doolan C Dobaño. *Acquired Immunity to Malaria.* 2009.

[2] indexmundi. *Kenya Demographics Profile 2016*. `http://www.indexmundi.com/kenya/demographics_profile.html`. 2016.