Pathogen RPS

New Mexico

Supercomputing Challenge

Final Report

April 2, 2014

Team 144

School of Dreams Academy

Team Members

Denton Shaver

James Edington

Teacher

Creighton Edington

Mentors

Creighton Edington

Talysa Ogas

Table of Contents

Executive Summary

Rock-Paper-Scissors is a game where two players simultaneously make one of three shapes with their hands. They make a fist to represent rock, flatten their hand to represent paper, and hold out their index and middle finger while spreading them apart to make scissors. In the game, Rock beats Scissors, Scissors beats Paper, and Paper beats Rock. The goal of the game is to pick the shape you think will beat the shape your opponent's pick.

Other organisms, such as E. Coli, have also been observed 'playing' a game of Rock-Paper-Scissors. Three strains of the bacteria would develop, each one weak to another. The continuous competition would create a stable, always-strengthening system.

The goal of this project is to observe how well pathogens that exist in such a relationship with each other will preserve their biodiversity in the bloodstream, along with regular blood cells. Another goal of this project is to see what the most efficient way of treating this condition would be, and what would happen to the different populations when antibodies are injected, or more of one of the strains is injected into the bloodstream.

Problem Statement

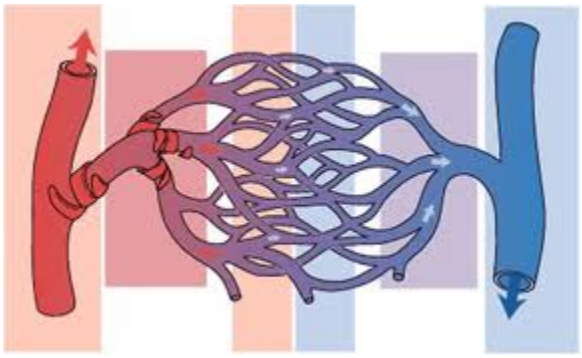
This project looks at what would happen if three pathogens with a symbiotic relationship like the one described above entered the body. The relationship that would take place should help preserve biodiversity in the pathogens, and none of the strains would die off completely because of this. This project is presented as an agent based model which was programmed and run in NetLogo. The model was programmed to have different types of agents and to run on a bitmap. There were four types of agents used, three of these types of agents represented different cells which were infected with different strains of pathogens, and one type that represented uninfected cells which had a possible immunity to the different pathogens. The bitmap was used to create different pathways for the agents to move on without programming each different patch in the interface to be a different color. The different types of agents would move throughout the system; whenever two agents encountered each other, a conflict would occur, to be resolved by taking the agents' types and playing a "Rock-Paper-Scissors" game. The winning agent would consume the losing agent.

Methods

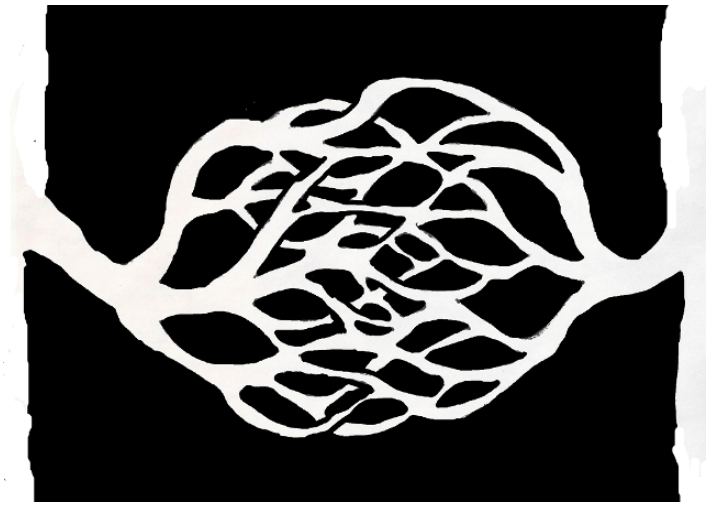
The first step in this project was to research whether or not there were already pathogens that interacted with each other in a competitive symbiotic relationship as described above. We found that E. Coli was known to create such a system with three different strains which each outcompete each other. The three strains are Colicin-producing (C strain), Colicin-resistant (R strain), and Colicin-sensitive (S strain). The C strain produces an antibiotic called colicin; the colicin that this strain produces kills off the S strain. The S strain, however, could outcompete the R strain, which could, in turn, outcompete the C strain. This system has been seen in nature, however, when a study was done on this they found that biodiversity would break down when the different strains of E. Coli were mixed together in a flask, showing that some spatial separation may be necessary in order for these different populations to coexist (Stanford News Service, 30).

To create a model accurately representing the behavior of the three strains of E. Coli, we needed to create a system sufficiently large to reach an equilibrium, with the appropriate system dynamics.

To model this, we needed to first create a simulated environment, then create the different strains of bacteria, and finally program the strains to behave like the actual bacteria would. First off, it was necessary to find a drawing or picture of capillaries to use as the background and pathways for the agents in the program. After finding a picture, it needed to be changed to a smaller image size and into a black and white image.



The image was changed to black and white, and had several features removed because only the main system was required. When the picture was finished it was saved as a PNG image.

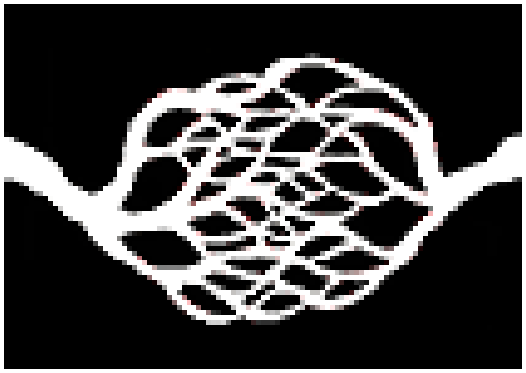


When the picture was ready it needed to be set as a bitmap for NetLogo, and the movement of the agents throughout the system had to be figured out.

Once the image was set as the bitmap, the next step was to figure out the movement for the turtles. The goal for the movement was to get them to disperse through the system, and to get them to act like they were in an enclosed system. The easier step was to get them to act like they were in an enclosed system by programming them according to the quadrant they were in and what their current x coordinate was. It was decided that if the agents were on the far right side, they would move up until they hit the top of the map, and appear on the far left side. While on the far left side they would move down until they reached the middle, where they either turned to the right and entered the system, or continue moving down and appear on the far right side.

When they were going through the capillary system they would disperse randomly through it, and eventually come out and join with the agents moving up on the far right side. We programmed the movement according to what quadrant they were in, and tested it. During testing we noticed that when agents moved from one area to another one, they would leave big spaces empty with no agents, also the agents would begin to streamline as the program ran for longer.

After we figured this out, we decided to approach the movement in a different way. The first thing we decided for the movement was to make the agents only move from the left side to the right side. In order to help with this we decided to modify the image more.



The far right and left areas were filled in with black, so now the agents only move from left to right, and disperse throughout the capillary system. After deciding on the movement we began programming. For programming the model we needed four different types of agents: one that represented C strain (‘rocks’), one that represented R strain (‘paper’), and one that represented S strain (‘scissors’). We also needed one that represented regular blood cells. The different types of agents were declared as different breeds and given different colors. Then they were programmed to initially spawn only on the white parts of the image. They were also programmed to only move on white spaces and only create new agents on white spaces. This was done by telling the program to check if a patch was white: if it was white, an agent would spawn. The movement was done by telling the agents to do a wiggle, then check if the patch ahead was white, it it was the agent would move, if the patch ahead was black the agent would redo the wiggle and check again.

Programming

After the movement was programmed, they had to be programmed to play Rock-Paper-Scissors. To do this they were told to look at the patches around them for a patch with an agent.

```
to move
  ask turtles
  [

    if can-move? 1
    [
      ifelse ([pcolor] of patch-ahead 1 = white)
      [
        forward 1
      ]
      [
        random-placement
      ]
      if count turtles-here > 0
      [
        random-placement
      ]
    ]
  ]
end
```

If one of the patches had an agent it would look at what breed the other agent was. If it was the same breed or the breed of the other agent was a blood, they would ignore eachother and keep moving. However, if the other agent was a rock, paper, or scissor, and they were not the same breed they would play Rock-Paper-Scissors.

```
to wiggle
  ask turtles
  [
    set heading random 180
    set ycor ycor + one-of [ -1 1]
  ]


end
```

After the agents were programmed to play Rock-Paper-Scissors, we needed to add agents that would be “injected” to experiment with treating the condition. We decided to try two different ways of treating the condition. One of these ways included three different kinds of “antibodies”, these “antibodies” were programmed to kill off certain kinds of agents then die off. The rock agents were killed off by anti-rock agents, paper agents were killed off by anti-paper agents, and scissors agents were killed off by anti-scissors agents. We held different tests for different kinds of “anti-agents”, so they were not released together in the same tests. The other way we looked at treating the condition was injecting more of the rock, paper, or scissors agents.

The reason for doing this, was to see if we could kill off one kind of agent by injecting more of another kind of agent. Like with the anti-agents, we injected more of each kind of agent in different tests.

Some of the main methods have been shown here.

```
to random-placement
  setxy (pxcor + one-of [ -1 0 1 ]) (pycor + one-of [ -1 0 1 ])
  if [pcolor] of patch-here = black
  [
    random-placement
  ]
end
```

```
to play-anti ;lets antivenom play the game
  ask turtles
  [
    if one-of turtles-on neighbors4 != nobody
    [
      let opponent ( one-of turtles-on neighbors4 )
      let opponent-breed ( [ breed ] of opponent )
      let my-breed breed
      if ( ( my-breed = opponent-breed )
        or ( opponent-breed = bloods ) or ( my-breed = bloods))

;         or (my-breed = rocks) or (opponent-breed = rocks)
;         or (my-breed = papers) or (opponent-breed = papers )
;         or (my-breed = scissors) or (opponent-breed = scissors ) )
      [
        stop
      ]
      if ( ( my-breed = anti-rocks and opponent-breed = rocks )
        or ( my-breed = anti-papers and opponent-breed = papers )
        or ( my-breed = anti-scissors and opponent-breed = scissors )
        or (my-breed = rocks and opponent-breed = anti-rocks )
        or ( my-breed = papers and opponent-breed = anti-papers )
        or ( my-breed = scissors and opponent-breed = anti-scissors ) )
      [
        ask opponent
        [
          die
        ]
        die
      ]
    ]
  ]
end
```

To program the anti-agents and injections we used sliders, and buttons. Sliders were used determine how many of each kind of agent was injected, and buttons were used to actually

“inject” the agents into the system. For the anti-agents we had to program to kill off the original types of agents. We did this in a similar way to the way we programmed the original agents to play their game of Rock-Paper-Scissors. They would look at the patches around them for agents, if they found one they would look at what breed the agent was, if the agent it found was the type of agent it was told to kill off, it would kill it, then die out, otherwise it would move forward.

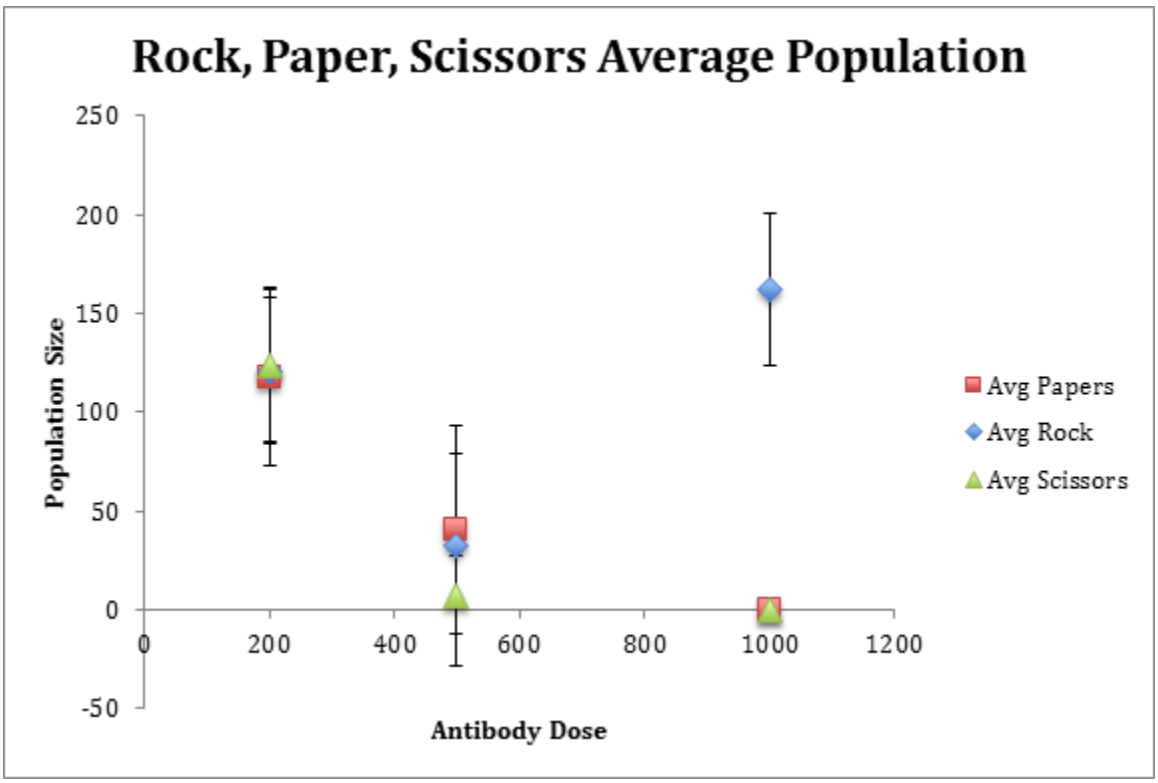
When the programming was done, we conducted 7 different kinds of tests. One where we injected anti-rock agents, one where we injected anti-paper agents, and one where we injected anti-scissors agents. Other tests were where we injected more of each type of the original agents, we injected more rocks, more papers, and more scissors. We also conducted a test where nothing was injected and the model just ran. Each test was conducted at least thirty times to collect data. While the model was being run the data was being recorded. After the data was collected it was exported from NetLogo and into an excel file.

Verification and Validation

Though there are no cases so far where a human being has been infected with this kind of infection, there are cases where bacteria such as E. *coli* use competitive symbiotic relationships to maintain biodiversity and keep from dying off. Since this is the case, this project could be seen as a preventative step towards the future, in case a human being, or even an animal becomes infected with this kind of disease.

Results

Since there were several different kinds of tests there were different results. The first tests conducted were the tests where nothing was injected. The results of these were that the populations of the different types of agents were stable, with occasional spikes every now and then. For the tests where additional amounts of one of the agents were injected, no species died out, the tests where In the tests where the antibodies were injected, two species died out, the species that was specifically hunted by the antibodies, and the species that hunted that one.



The most efficient method of eliminating one pathogen (thereby breaking the chain of dependency and eliminating the entire colony) is to inject antibodies into the system to kill off one strain, which would in turn kill off a second strain leaving only one strain left. The final strain would have to be killed off with a second injection.

Conclusions

In conclusion, the most efficient way of curing this type of infection would be with two vaccines, one to specifically kill off one species of pathogen, which would in turn kill off another species, and another one to kill off the remaining species of pathogen. In the future, more tests can be done where the injections are given when certain populations are at different levels. For example, an injection could be given when one population is at a very high point, or when its at a low point.

Significant Achievement

The significant achievements of this project included finding out that an injection of antibodies into the system kills not only the species that it is supposed to kill, but another species also ends up dying off too. Another achievement was taking a step of preventing this condition from happening in the future

Acknowledgements

We would like to thank our mentors Creighton Edington, for helping us with our code and understanding our project, and Talysa Ogas for helping us with the Final Report, and the presentation for our evaluation.

Resources

"Capillaries." kidport. kidport, n.d. Web. 2 Jan. 2014. <http://www.kidport.com/reflib/science/humanbody/cardiovascular/Capillaries.htm>.

Levy, Dawn. "'Rock Paper Scissors' preserves biodiversity in bacterial neighborhoods." Stanford News Service. Ed. Brendan Bohannan. Stanford, 8 July 2002. Web. 2 Jan. 2014. <http://news.stanford.edu/pr/02/bohannan724.html>.