Selective Breeding (Foxes) Evolutionary Simulation

By Peter Smiley

Table Of Contents

Relevant Links

- Website → https://evanenss.github.io/PetersCBSFinalProject/
- Code → https://github.com/psmiley2/SelectiveBreedingCBSProject

How To Use

- “Breed for Kindness” button → Selectively Breeds for the white circles
- “Breed for Aggression” button → Selectively Breeds for the red circles
- Browser Refresh → Reset Simulation
- Generation # → How many generations of breeding have occurred
- Average Aggression # → The average aggression of the agents in the simulation. 0 representing all perfectly kind agents. 1 representing all perfectly aggressive agents.

Description

This project is based on the Russian Fox Farm Experiment. When the simulation starts, there are 100 agents (circles) on the screen. Each agent starts with a randomly generated aggression somewhere between 0 and 1.

The closer the agent's aggression is to 0 the more kind it is. The closer the agent's aggression is to 1, the more aggressive it is. The long green number at the top left of the screen shows the total average aggression for the population of agents.

If you click breed for kindness, the kindest 50 agents will be selected to be bred for the next generation. Each of the 50 agents will be paired with another one. These pairs will move towards each other representing that they are mating. These two relatively kind agents will then produce children (with a bit of mutation mixed in). These children will then be added to the next generation. Every generation that passes represents a round of mating that occurs.

If you click breed for aggression, the same algorithm will be run; however, the agents that breed will be the 50 most aggressive ones.

This hopefully provides an interactive way to see selective breeding in action.

Analysis

The outcome of this project was a little bit disappointing for me at first, but it revealed something incredibly interesting about selective breeding that I didn't fully grasp before.

The reason it was disappointing at first was because as you can see, the first round of selective breeding makes the most impact. Then, in order to make a real impact, you have to keep clicking in the opposite direction to wait for mutation to inch up the aggression one way or another. The more I thought about it, the more this makes sense to me. This really highlighted that to make a lot of progress quickly with selective breeding, having a distributed gene pool is super important. If all of the foxes are super aggressive, then selective breeding would still take a really long time to reach kind foxes. You would essentially have to wait for mutation to run its course.

Therefore, if I were to run the Russian Fox Farm experiment myself I would not start with one bunch of foxes from the same place. I would try to get as many different foxes from all over to get as diverse of a population as I can. Then, the impact I can make with selective breeding will be super significant super fast.

Code Description

- This code chunk right here determines how the foxes are selected and paired up in the mate for kindness algorithm.
  - First we sort the foxes from kindest to most aggressive
  - Then we splice the list so we only take the first half of the sorted list (representing the kinder half of the foxes)
  - Then we shuffle that list of kind foxes and pair them randomly

```
let sorted = [...this.foxes].sort((a, b) => (a.aggression < b.aggression)
? 1 : -1)
let survivors = sorted.splice(sorted.length / 2)
let shuffleKindAgents = this.shuffle(survivors)

let pairs = []
for (let i = 0; i < shuffleKindAgents.length; i+=2) {
      shuffleKindAgents[i].hasTarget = true
      shuffleKindAgents[i].target = shuffleKindAgents[i+1]
      shuffleKindAgents[i+1].hasTarget = true
      shuffleKindAgents[i+1].target = shuffleKindAgents[i]
      pairs.push([shuffleKindAgents[i], shuffleKindAgents[i+1]])
}
```

- This next code chunk shows how the mating works in the mate for kindness algorithm.
  - For all of the mates that got paired up in the code block above, we take the average of their aggressions, apply a random mutation, and then add the results to the next gene pool.

```
for (let i = 0; i < pairs.length; i++) {
      let avg = (pairs[i][0].aggression + pairs[i][1].aggression) / 2
      let daughterVal1 = avg + random(-0.01, 0.01)
      let daughterVal2 = avg + random(-0.01, 0.01)
           daughters.push(daughterVal1)
           daughters.push(daughterVal2)
}
```