4-24-2018

Analysis of Organ Donations Using Agent Based Modeling

Jared Tyler Grogan

Analysis of Organ Donations Using Agent Based Modeling

by

Jared Tyler Grogan

An Honors Capstone

submitted in partial fulfillment of the requirements

for the *Honors Diploma*

to

The Honors College

of

The University of Alabama in Huntsville

24 April 2018

Honors Capstone Director: Dr. Brinda Mahalingam, Lecturer of Economics

Student	Date 24 April 2018
Director	Date 4/25/2018
Department Chair	Date 4/24/18
Honors College Dean	Date 4/26/18

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Jared Grogan

Student Name (printed)

Student Signature

24 April 2018

Date

Table of Contents

Dedication:

The author wishes to dedicate his project summary to Dr. Mahalingam, Dr. Orman, Professors Gramm, Schnell, Wilhite, and Card, my friends Juan and Joseph, and to any other individuals who at any time endured my incessant rambling while completing the project in addition to the insight and support they have given me in the past four years.

Abstract

In ECN 499, “agent-based models” (hereinafter referred to as ABMs) are the primary focus of the course. An ABM is a model wherein agents, or individuals, are given various attributes and abilities whereupon they are placed into a defined world to interact based upon these attributes and abilities. An ABM is useful to study phenomena in a simple, abstract manner, and they allow for greater flexibility with respect to the framework within which the model is defined.

This project write-up summarizes the conceptualization, construction, and implementation of an ABM using NetLogo, an open-source GUI-based program which is primarily suited for building models using simple syntax and assets included with the program. Specifically, the author creates three separate models to simulate the process of organ donation. Since laboratory experiments or random controlled trials are difficult or infeasible with respect to organ donations, ABMs are an ideal alternative to investigate the effects of changing the donation process. While the models do not exhibit significant differences in organ allocation, the author nonetheless gains invaluable experience with building models and proposing a research question.

Introduction

In the United States of America, over 130 million adults are registered organ donors (U.S. Department of Health & Human Services). Meanwhile, twenty people die each day waiting for a transplant, and over 116 thousand populate the national transplant waiting list as of August 2017 (U.S. Department of Health & Human Services). While this disparity in interested donors and waiting recipients may be explained due to factors such as the transport and preservation of organs, blood type compatibility, and others, the number above nevertheless piqued the interest of the author. As an aspiring economist, I am interested in analyzing institutions and systems to see whether they produce efficient or optimal outcomes, and organ donations appear to be an area where improvements may be made.

However, I cannot gather groups of donors and individuals needing organs, hand out money to the recipients, and have the donors auction off their organs to see what happens. Fortunately, agent-based models provide an alternative to the logistical and ethical challenges of a real-world experiment with organ donations. Over the past several months, I have conceptualized, constructed, and implemented three distinct models of organ donations that seek to illustrate the results of introducing different rules and procedures to the process of receiving a transplant.

This paper summarizes the process of creating the models, the issues that appeared during the process, and an assessment of the overall quality of the models.

Conceptualizing the Models

As discussed in the introduction, the organ donation system has gathered my interest and is the focus of my models. Before I began coding, however, I dedicated time and effort to learning more about the procedures and definitions that comprise the donation system. Otherwise, I would be attempting to create agents and interactions that may or may not reflect reality in any way whatsoever. Once the preliminary research was complete, I began the "bottom-up" approach to creating an ABM. In other words, I focused on what defines an agent, what defines an interaction, and what defines the world within which the agents interact. From there, the global attributes of the models may be more clearly defined and recorded.

The Waiting List

According to the Organ Procurement and Transplantation Network (OPTN), over 80 percent of the waiting list candidates are waiting for kidneys; furthermore, a large majority of donors are deceased at the time of transplantation (Health Resources and Services Administration, "Organ Donation Statistics," 2018). Furthermore, the OPTN lists "waiting time, immune system incompatibility, pediatric status, distance to donor's hospital, and survival benefit" in addition to blood type compatibility as determinants of the allocation of a kidney (ibid, "How Organ Allocation Works," 2016). Specifically, an Estimated Post-Transplant Survival (EPTS) score is assigned to all adult patients on the kidney waitlist to determine the likelihood of survival; a lower EPTS score means a patient is more likely to survive, and therefore, low scoring patients receive "increased priority" over individuals with higher scores who have been waiting a similar amount of time (ibid, 2014).

This information provided several insights with respect to formulating my models. First, I should focus on one organ (e.g. kidneys) in my model, but include different "blood types" to

reflect compatibility issues. Secondly, my donors should only be able to donate once; since most actual donors are deceased, there are limitations to how many times they may donate. Third, the "viability" of a donor and/or recipient is clearly a consideration of the organ donation system, and therefore, my agents need to be assigned a measure of their viability for donation. Finally, due to the differentiating factors, I need to be able to deliver a list of agents compatible with a donor so that the model doesn't allocate organs to individuals that would not accept the organ in reality.

A Market Approach

With the above information, I may create a simplified simulation of the organ donation system as-is. However, I am also interested in simulating a "market" to allocate organs. While this is currently illegal in the United States,[1] I sought to investigate the effects of allowing the selling of organs to see if outcomes for recipients changed in in any significant manner. Therefore, my agents must additionally possess money for exchange, a method of evaluating the value of an organ relative to the money, and a method to determine whether to make an exchange.

Donors' Valuation of Organs

Most of the challenges I faced during the project were concerned with the market-based simulations of the organ network. Since there is no such (legal) market in existence, I had to apply my own reasoning to determine how valuations and exchanges would occur. This leads to a few pressing questions about humanity. For instance, would donors consider the viability of

[1] The National Organ Transplant Act, introduced by Orrin Hatch in 1983 and passed in 1984, "prohibits the purchase or sale of human organs if such transfer affects interstate commerce."

their recipient when selling an organ, or would they exclusively consider the money that could be made by selling the organ? This question compelled me to form two separate market models, each exploring one side or the other. In one model, only money matters. In the other, the amount of money resulting from selling an organ is weighted by the probability that the recipient survives the transplantation. This will be detailed later in the manuscript.

Recipients' Valuation of Organs

Just as donors' behavior perplexed me, so too, did the motivations of the recipients in a market environment. I ultimately propose that recipients would exhibit a positive relationship between their expected time remaining to live without a transplant and the amount of money that they would offer for a transplant. In other words, as time passes, a recipient would offer a larger share of their money to receive an organ and eventually offer their entire amount of wealth to survive. However, there may be individuals who would, after some time, realize that their chances are waning, and thereby give up the search to allocate their wealth to heirs instead. Furthermore, there may be those who reverse the relationship between time waiting and the amount of money offered: their rationale would lead them to the conclusion that the sooner they receive an organ, the better the outcome, and therefore, they would offer a premium for minimizing the amount of time waiting for a donation.

These conflicting motivations interested my advisor, but we ultimately decided that most recipients would likely exhibit the positive relationship, and therefore, the other cases aren't considered in the models. What remained, then, was the mechanism for selling the organs. I opted for a simple auction wherein the donor obtains a list of compatible recipients and sells to that recipient who offers the best bid.

Creating the Models

With all the properties of the organ donation system in mind, including my proposed properties of a market system, I began the process of choosing a programming language in which to implement the models. For the ECN 499 course, we used NetLogo,[2] an open-source language with visual assets and a focus upon ABMs, to create sample models early in the course. The syntax of NetLogo was quite simple compared to C and C++, and it handles many of the more complex data structure components of a model on its own, meaning it was ideal for my intentions. Despite never working with NetLogo previously, I quickly acclimated myself with the rules and procedures that govern how the code is compiled and set out to create my models.

General Properties of the Models

While the methods of organ allocation would differ with each model, most components would be identical or similar between the models. For instance, all models run in a torus-shaped world comprised of a grid. Each square on the grid represents an agent, and adjacent squares are considered "neighbors." Time is represented discretely by "ticks." I will now detail the variables and procedures that define the agents and initiation of the simulations.

Agents

In my organ donation model, there are ultimately six types of agents that may populate the world at any time. In the beginning, there are three primary types: "donors," "recipients," and

[2] Northwestern University hosts a website dedicated to NetLogo which includes downloads of the software and other resources. The author highly suggests viewing some of the sample models at <https://ccl.northwestern.edu/netlogo/>.

"potential donors." As the simulation progresses, one may observe "post-donors," "post-recipients," and "deceased recipients."

The primary agents are assigned a uniformly randomly distributed value, denoted *organtype* in the code, that reflects the different compatibility types of an organ. Each agent is also assigned a value denoted *viability* which seeks to mimic the purpose of an EPTS score (except in this case, a higher *viability* value reflects a higher probability of survival after a transplant or a higher quality organ for recipients and donors, respectively). The *viability* value is normally distributed with a mean and standard deviation each determined by a slider comprised of incremented values.

Beyond these variables, there are several other variables that function as status indicators and therefore determine the type of agent a given grid square can be. Specifically, one may refer to Table 1 for a summary of the variables.

VARIABLE	DESCRIPTION
ACTIVE	1 if seeking donation/recipient; 0 if not donating; -1 if donated, received, or deceased
RECRUIT	1 if donating; 0 otherwise
MATCH	1 if matched, 0 otherwise
PARTNER	Reports the identity of an agent's match
COOLDOWN	1 if received an organ during the last tick; 0 otherwise
VISION	Determines how far a donor looks for recipients

Table 1: Summary of Status Variables

Depending on the values of *active, match, and recruit*, agents assume one of the six aforementioned agent types. For example, an agent with the combination (-1, 1, 0) would be a "post-recipient." Finally, for the market models, variables defining money (*cash, trycash*) and utility or satisfaction (*util, tryutil*) are used to assist with the auction process.

Initiation

For each model, the world is initiated in similar manners, the exceptions being the variables unique to the market models. Each agent is given *organtype* and *viability* values, and then the following occurs. First, 80 percent of agents are selected at random. If their *viability* value is above 50, these selected agents begin the simulation as donors. Next, all agents not selected as donors are collected, and a uniformly random value is generated between 0 and 100 for each of these agents; if this value is less than a predetermined value that reflects the percentage of recipients desired, a uniformly random value called *survive* is assigned to the agent. The agent then begins the simulation as a recipient. All other agents are given a *survive* value of 0, and the remaining agents that have not been assigned a role are designated as potential donors. After this occurs, a random value is assigned as the *vision* of all agents.

At this point, a world of agents seeking to donate or receive organs is created. While this world is quite simplified compared to reality, it nevertheless exhibits qualities and quantities that are relevant to organ allocation and avoid computationally complex operations, which are desired properties in an ABM. Now the mechanisms of organ allocation itself within the models may be explained.

Allocation of Organs

Now that agents exist in my generated world, they need procedures to interact with each other. The following paragraphs summarize the algorithms used to run the models.

General Procedures

Each model has three main stages. First, they are initialized as described above. Then, a tick occurs, wherein three procedures occur. Finally, a check is made to see if the model should continue running.

The three procedures that occur during a tick of time each complete a distinct set of tasks. First, there is a check for survival of recipients and post-recipients which is determined by *survive* and *viability*. For recipients, if the number of ticks that has occurred is equal to his or her *survive* value, a random number between 0 and 100 is chosen; if their *viability* value is less than this value, they become deceased. In a sense, *survive* acts as a "life expectancy," with death being impossible prior to reaching that tick value. Similarly, for post-recipients, a random number between 0 and 100 is chosen; if their *viability* value is less than this value, they also become deceased. The check for post-recipients occurs only once after the tick in which they received an organ and reflects the chance for the transplant to fail.

Secondly, there is a check for recruitment of potential donors. A subset of potential donors is selected, and if they meet the minimum *viability* threshold (50 or greater), they become active donors. Otherwise, a potential donor remains as a potential donor. Third, there is the process of finding a match. This varies with each model and is therefore defined for each below.

Matching Outside of a Market

The first model is distinct from the market models due to its lack of a market. In particular, the first model seeks to resemble the current system to an extreme degree. One may describe this model as the "benevolent dictator" model due to its matchmaking procedure, as described below.

In the “dictator” model, a master list of living recipients is generated. Each donor then refines the list based upon his or her *organtype* value and sorts by the difference between the *survive* value of a recipient and the current amount of ticks that have occurred. Agents outside the range of the donor’s *vision* are then excluded. This results in a list of compatible recipients near the donor with those who have the least amount of time remaining to find a match at the top. The donor in question then is forced to match with this recipient, and the organ is “exchanged” (i.e. the donor becomes a post-donor, and the recipient, a post-recipient). In reality, donors are not compelled to follow through if a match is made, so the “dictator” reflects the best-case scenario of the current system. Specifically, if all donors did follow through with their donation pledge, the “benevolent dictator” model would be quite close to the actual organ allocation system!

Matching in a Market

My goal in pursuing this project was to compare the current donation system to a market-based system. As discussed before, I decided that two donor types could arise: “selfish” donors who care only about money, and “selfless” donors who seek to maximize their amount of wealth but qualify this desire with a check for the expected survival of a recipient. Both types of donors perform auctions in a similar manner, but the conditions for accepting a bid are different.

For “selfish” donors, a list is created as in the “dictator” model, except it is not ordered by *survive*. Instead, each recipient reports to the donor the value of their wealth multiplied by the ratio between the current tick count and their *survive* value (or their entire wealth otherwise); this is the “bid” of the recipient. The donor checks all bids and accepts the highest bid, whereupon the organ is exchanged for the bid amount.

Meanwhile, for “selfless” donors, the same process occurs, except the donor compares his cash amount before a potential exchange to the product of his cash amount after a potential exchange and the value of the recipient’s *viability* divided by 100. In mathematical terms, the exchange is made if

$$(C_d + B_r)\frac{v_r}{100} \geq C_d$$

where C_d represents the cash amount of the donor *d*, B_r represents the bid of a recipient *r*, and v_r denotes the *viability* value of recipient *r*. This means that a “selfless” donor may decide not to donate unless the bid and probability of survival are sufficiently large enough to exceed the value of the cash the donor currently has.

Evaluation of Models' Effectiveness

An evaluation of the models finds that, while the procedures differ between them, they appear to behave similarly. Similar counts of donations and deaths occur in all three models, which seems to suggest that a "benevolent dictator" is just as effective as a market of "selfish" donors or a market of "selfless" donors. These results are consistent when varying the parameters that initialize each of the models.

The interpretation of these results should be made carefully, however. While one could argue that allowing individuals to sell their organs would not cause additional deaths based upon the models, one should recall that the models do not consider several relevant factors when deciding on organ allocation such as the ethical issues of selling organs, combinations of "selfish" and "selfless" donors, or the switching costs associated with converting from one system to the other.

Meanwhile, I am overall satisfied with the models as-is, but I feel that more can be done to investigate the market types. For instance, my models don't allow new recipients to spawn; the recipients that are present at the beginning of the simulation are the only recipients to ever exist in the simulation. Furthermore, the bidding process is somewhat simplified and doesn't consider other factors that could affect the amount a recipient would set aside for bidding. My two market models separate "selfish" and "selfless" donors; if time permitted, I would have created a "hybrid" market simulation with varying amounts of each type of donor. Finally, other attributes of organ allocation, such as preservation times, operation costs, and "cold feet" from donors aren't considered in my models. Nevertheless, I learned a considerable amount about agent-based modeling by undertaking this project, and additionally, I became more aware of the attributes of the organ donation system that exists in the United States today.

Conclusion

Would market-based allocation of organs improve outcomes for recipients? Based upon my models, no meaningful answer can be confidently given for this question. That being said, my primary goal, to conceptualize, create, and observe agent-based models that simulate organ allocation was largely successful.

In the future, I intend to further investigate the effects of organ allocation, but I will also consider ABMs of topics such as global trade, immigration, minimum wages, and other economic phenomena. My experience with NetLogo acquired through this project will assist me in these endeavors, and I gained meaningful insight in to the general process of creating a model without any previous experience in doing so. Furthermore, I've seen examples of other models at work while participating in ECN 499 that have inspired me to ask new questions in the context of ABMs.

Works Cited

"Learn How Organ Allocation Works." *Organ Procurement and Transplantation Network*, Health Resources and Services Administration, U.S. Department of Health & Human Services, Mar. 2018, optn.transplant.hrsa.gov/learn/about-transplantation/how-organ-allocation-works/.

"Organ Donation Statistics." *Organ Donation Statistics: Why Be an Organ Donor?*, Health Resources and Services Administration, U.S. Department of Health & Human Services, www.organdonor.gov/statistics-stories/statistics.html.

"Organ Procurement and Transplantation Network." *Data - OPTN*, Health Resources and Services Administration, U.S. Department of Health and Human Services, Mar. 2018, optn.transplant.hrsa.gov/data/.

S.2048 - National Organ Transplant Act, 1984. www.congress.gov/bill/98th-congress/senate-bill/2048. Accessed 1 Apr. 2018.

United States, Congress, "A Guide to Calculating and Interpreting the Estimated Post-Transplant Survival (EPTS) Score Used in the Kidney Allocation System (KAS)." *A Guide to Calculating and Interpreting the Estimated Post-Transplant Survival (EPTS) Score Used in the Kidney Allocation System (KAS)*, OPTN Kidney Transplantation Committee, 20 Mar. 2014. optn.transplant.hrsa.gov/media/1511/guide_to_calculating_interpreting_epts.pdf.

Appendix A: Code for "Dictator" Model

```
globals[needorgan donations deaths recruitments reciplist seed]
patches-own[organtype recruit match partner active survive viability cooldown vision]

to setup
  clear-all
  set needorgan 1
  set donations 0
  set deaths 0
  set recruitments 0
  set reciplist []
  init-patches
  set-reciplist
  reset-ticks
end


to go
  if(needorgan = 0)[STOP]
  check-survive
  check-recruit
  find-match
  tick
end

to init-patches
  ask patches[

    set organtype random ORGCOUNT
    set match 0
    set viability init-viability
    set recruit ifelse-value(random 100 < 80 and viability >= 50)[1][0]
    set survive init-survive
    set active ifelse-value(survive > 0)[1][0]
    set pcolor init-pcolor
    set cooldown 0
    set vision random VIZ
  ]
end

to set-reciplist
  set reciplist sort-on [survive] patches with [pcolor = red]
end

to-report get-mylist [o]
  let ml []
  let neighborhood patches in-radius vision
  foreach reciplist[
    x -> ask x[
      if (organtype = o and member? self neighborhood)[
```

```
      set ml fput x ml
    ]
   ]
  ]
  report ml
end

to-report init-survive
  ifelse(recruit = 1)[report 0]
  [
   ifelse(random 100 < RECIPSPAWN)[report random RECIPEXP + 1][report 0]
  ]
end

to-report init-viability
  let v 0
  set v random-normal VMEAN VSTD
  if (v < 0) [set v 0]
  if (v > 100) [set v 100]
  report precision v 2
end

to-report init-pcolor
  ifelse(recruit = 1)[

   ifelse(random 100 < DONORSPAWN)[set active 1
    report green]
   [report yellow]
  ]
  [
   ifelse(active = 1)[report red][report yellow]
  ]
end

to check-survive
  let recips patches with [pcolor = red]
  let postrecips patches with [pcolor = orange]
  ask recips [
   if(ticks > survive)[
    set pcolor black
    set active -1
    set deaths deaths + 1
   ]
  ]
  ask postrecips [
    if (cooldown = 1)[
      ifelse (random 100 > viability)[
        set pcolor black
        set deaths deaths + 1
      ][set cooldown 0]
    ]
```

```
  ]
  set needorgan check-need
end

to-report check-need
  ifelse(count patches with[pcolor = red] = 0)[report 0][report 1]
end

to check-recruit
  let recruits patches with [pcolor = yellow]
  ask recruits [
    if(viability > 50)[
      set recruit 1
      set pcolor init-pcolor
    ]
  ]
end

to find-match
  let donors patches with [pcolor = green]
  ask donors[
    if (not empty? reciplist)[
      let mylist get-mylist organtype
      if (not empty? mylist)[
        set match 1
        set partner first mylist
        set pcolor blue
        set active -1
        set donations donations + 1
        ask partner[
          set match 1
          set partner myself
          set pcolor orange
          set active -1
          set cooldown 1
        ]
      ]
      set-reciplist
    ]
  ]
end
```

Appendix B: Code for "Selfless Donors" Model

```
globals[needorgan donations deaths recruitments money]
patches-own[organtype recruit match partner active survive viability cooldown vision cash trycash util tryutil]

to setup
  clear-all
  reset-ticks
  set needorgan 1
  set donations 0
  set deaths 0
  set recruitments 0
  init-patches
end

to go
  if(needorgan = 0)[STOP]
  check-survive
  check-recruit
  find-match
  tick
end

to init-patches
  ask patches[

    set organtype random ORGCOUNT
    set match 0
    set viability init-viability
    set recruit ifelse-value(random 100 < 80 and viability >= 50)[1][0]
    set survive init-survive
    set active ifelse-value(survive > 0)[1][0]
    set pcolor init-pcolor
    set cooldown 0
    set vision random VIZ
    set cash random ENDOWMENT
    set trycash 0
    set util init-util
    set tryutil 0
  ]
end

to-report init-survive
  ifelse(recruit = 1)[report 0]
  [
    ifelse(random 100 < RECIPSPAWN)[report random RECIPEXP + 1][report 0]
  ]
end
```

```
to-report init-viability
  let v 0
  set v random-normal VMEAN VSTD
  if (v < 0) [set v 0]
  if (v > 100) [set v 100]
  report precision v 2
end

to-report init-pcolor
  ifelse(recruit = 1)[

    ifelse(random 100 < DONORSPAWN)[set active 1
      report green]
    [report yellow]
  ]
  [
    ifelse(active = 1)[report red][report yellow]
  ]
end

to-report init-util
  ifelse (pcolor = red)[
    report (survive - ticks) * cash
  ][
  report cash
  ]
end

to check-survive
  let recips patches with [pcolor = red]
  let postrecips patches with [pcolor = orange]
  ask recips [
    if(ticks > survive)[
     set pcolor black
     set active -1
     set deaths deaths + 1
    ]
  ]
  ask postrecips [
      if (cooldown = 1)[
        ifelse (random 100 > viability)[
          set pcolor black
          set deaths deaths + 1
        ][set cooldown 0]
      ]
  ]
  set needorgan check-need
end

to-report check-need
  ifelse(count patches with[pcolor = red] = 0)[report 0][report 1]
```

```
end

to check-recruit
  let recruits patches with [pcolor = yellow]
  ask recruits[
    if(viability > 50)[
      set recruit 1
      set pcolor init-pcolor
    ]
  ]
end

to find-match
  let donors patches with [pcolor = green]
  ask donors[
    let recips patches with [pcolor = red and organtype = [organtype] of myself]
    let myrecips recips in-radius vision
    if any? myrecips[
      let prtnr nobody
      foreach [self] of myrecips[
        x ->
        set trycash cash + (ticks / [survive] of x) * [cash] of x
        set tryutil trycash * ([viability] of x) / 100
        if (tryutil > util)[set prtnr x]
      ]
      if (prtnr != nobody)[set-match prtnr]
    ]
  ]
end

to set-match [p]
  set match 1
  set partner p
  set pcolor blue
  set active -1
  set cash trycash
  set util tryutil
  set money money + (trycash - cash)
  set donations donations + 1
  ask partner [
    set match 1
    set partner myself
    set pcolor orange
    set active -1
    set cash (1 - ticks / survive) * cash
    set util survive * cash
    set cooldown 1
  ]
end
```

Appendix C: Code for "Selfish Donors" Model

```
globals[needorgan donations deaths recruitments money]
patches-own[organtype recruit match partner active survive viability cooldown vision cash trycash util tryutil]

to setup
  clear-all
  reset-ticks
  set needorgan 1
  set donations 0
  set deaths 0
  set recruitments 0
  init-patches
end

to go
  if(needorgan = 0)[STOP]
  check-survive
  check-recruit
  find-match
  tick
end

to init-patches
  ask patches[

    set organtype random ORGCOUNT
    set match 0
    set viability init-viability
    set recruit ifelse-value(random 100 < 80 and viability >= 50)[1][0]
    set survive init-survive
    set active ifelse-value(survive > 0)[1][0]
    set pcolor init-pcolor
    set cooldown 0
    set vision random VIZ
    set cash random ENDOWMENT
    set trycash 0
    set util init-util
    set tryutil 0
  ]
end

to-report init-survive
  ifelse(recruit = 1)[report 0]
  [
    ifelse(random 100 < RECIPSPAWN)[report random RECIPEXP + 1][report 0]
  ]
end
```

```
to-report init-viability
  let v 0
  set v random-normal VMEAN VSTD
  if (v < 0) [set v 0]
  if (v > 100) [set v 100]
  report precision v 2
end

to-report init-pcolor
  ifelse(recruit = 1)[

    ifelse(random 100 < DONORSPAWN)[set active 1
      report green]
    [report yellow]
  ]
  [
    ifelse(active = 1)[report red][report yellow]
  ]
end

to-report init-util
  ifelse (pcolor = red)[
    report (survive - ticks) * cash
  ][
  report cash
  ]
end

to check-survive
  let recips patches with [pcolor = red]
  let postrecips patches with [pcolor = orange]
  ask recips [
    if(ticks > survive)[
      if(random 100 + 1 > viability)[
        set pcolor black
        set active -1
        set deaths deaths + 1
      ]
    ]
  ]
  ask postrecips [
      if (cooldown = 1)[
        ifelse (random 100 > viability)[
          set pcolor black
        set active -1
          set deaths deaths + 1
        ][set cooldown 0]
      ]
  ]
  set needorgan check-need
end
```

```
to-report check-need
  ifelse(count patches with[pcolor = red] = 0)[report 0][report 1]
end
to check-recruit
  let recruits patches with [pcolor = yellow]
  ask recruits[
    if(viability > 50)[
      set recruit 1
      set pcolor init-pcolor
    ]
  ]
end
to find-match
  let donors patches with [pcolor = green]
  ask donors[
    let recips patches with [pcolor = red and organtype = [organtype] of myself]
    set trycash cash
    let myrecips recips in-radius vision
    if any? myrecips[
      let prtnr nobody
      foreach [self] of myrecips[
        x ->
        let bid min list (ticks / [survive] of x * [cash] of x) ([cash] of x)
        if (cash + bid > trycash)[
          set prtnr x
          set trycash cash + bid
        ]
      ]
      if (prtnr != nobody)[set-match prtnr]
    ]
  ]
end
to set-match [p]
  let bid trycash - cash
  set match 1
  set partner p
  set pcolor blue
  set active -1
  set money money + bid
  set cash cash + bid
  set donations donations + 1
  ask partner [
    set match 1
    set partner myself
    set pcolor orange
    set active -1
    set cash cash - bid
    set cooldown 1
  ]
end
```