Fish-fisher-consumer:
how to model their interplay?

Valentina Macchiati

June 15, 2017

1 Introduction

The purpose of the work is to reproduce the interaction of the fishes (preys) and the fishers (predators) and the fish market between the same fishermen (sellers) and the consumers (buyers). The final target is to replicate their negotiations and then create the supply and demand schedule. In this kind of market it is important to understand how the consumers' interest in buying at their bid price interplays with the fishers' need to sell all the goods at the end of the day (a cycle of the program) at their ask price.

The model uses NetLogo, an agent-based programming language and integrated modeling environment. NetLogo is open-source so it could be easily used for example by the companies or the public administration. In this language all it is an agent; turtles, patches, links and the observer. This language is very intuitive and perfect to approach the simulation of natural or social phenomena. Besides, thanks to the very wide model library [1], it allows to get in touch with the language effectively in a short time and at the same time it allows the user to create an effective model. So, an agent-based model (ABM) is a computational model that allow us to reproduce the actions and interactions between agents in order to understand how their interplay could alter the environment and what kind of complex phenomena could appear. Individual agents are typically characterized as boundedly rational so they act to obtain what they perceive as their own interests using simple rules to decide.

In the study of the interaction fishes-fishers, it is important to find the situation in which both "prey" and 'predator" are preserved. In fact, if the fishers fish too many the natural resource goes to extinction and then the fishers' incomes end. So it need to find the conditions that preserve the equilibrium of the environment. This kind of model, that could be viewed as a prey-predator one, could be very needful for the ecology. In fact, introducing the real variables and quantities, it is possible to find the possible evolution of the system and if it is necessary introduce some rules to prevent an unfavourable condition for the ecosystem.

In the fish market there are buyers and sellers that have to balance their target of buying or selling. Their will of get so is measured by their ask and bid price. Furthermore, the fishers want to sell all the caught fishes in the same day because these ones are a perishable commodity. The record of the previous interaction helps all the agents to change their price in order to achieve their target. The interest it is focused on the dynamical matching between supply and demand that takes into account the competition between the same fishers and the same consumers.

2 Model

The model could be divided into two part. The first part is related to the interaction between the fishes and the fishers the second one is focused on the sale and the purchase of the fishes between the same fishers and different kind of consumers.

2.1 Agents

2.1.1 Interaction fish-fisher

The idea is to reproduce the dynamics of two kind of fishes in a sea where are also fishing some fishermen. There are only two different breeds; precious one and common one. The choice is made in order to recreate a market with two different kind of goods: expensive and cheap ones. In fact, it is important to see how the supply and demand schedule changes respect not only the total quantity but also the relative one between the different breeds.

The agents of this interactions are the fishes and the fishers; the first ones are caught by the second ones.

```
breed[precious lobster]
precious-own [preciousgroup]
breed[common nemo]
common-own [commongroup]
```

A kind of fish is considered precious when its initial amount is lower than the common one and its growth rate is smaller or equal than the others. Besides, it is observed that the fishes move into schools. So, in order to preserve this feature, each "turtle", depicted as a fish, represents a group of fishes and the size of each group is a breed-own called *preciousgroup* or *commongroup*. The size of each group is a random variable and its maximum value is set by a slider both for common and for precious fishes. Could be recommended to set the maximum number of the common fishes in a school greater than the other in order to guarantee at the beginning of the simulation that the real number of precious fishes is smaller than the common ones.

In this part of the model, all the variables as the initial quantity of the fishes, their growth rate and the maximum side of each group are a slider.

In this way, the model could be used also by the same fishers introducing the correct values of their environment in order to understand its evolution and to improve their income observing the natural resources.

2.1.2 Interaction fisher-consumer

The second part of the model is about the interaction of the fishers (sellers) and the consumers (buyers). Both fishers and sellers have different breed-own necessary to reproduce their negotiations.

```
breed [fishers fisher]
fishers-own [preciousnow commonnow preciousfished commonfished
             mincommonPrice minpreciousPrice
             settledcommonPrice settledpreciousPrice
             listofsettledcommonPrice listofsettledpreciousPrice
             listofpreciousremainders listofcommonremainders
             numpreciousloyal numcommonloyal
             listnumpreciousloyal listnumcommonloyal
             recordpreciousinteraction recordcommoninteraction]

breed [consumers consumer]
consumers-own [fishespreferences fisherspreferences
               preciousQ commonQ loyal? timefidelity recordofshop
               bought? recordbought? maychange? boughtcommon?
               maxpreciousPrice maxcommonPrice
               settledcommonPrice settledpreciousPrice
               listofsettledcommonPrice listofsettledpreciousPrice]
```

First of all, it is important to know the quantity of fishes caught by each fisher everyday. The breed-own called *preciousnow* or *commonnow* and *preciousfished* or *commonfished* consider respectively the precious or common fishes that are caught in a day (one tick) and the record of the caught ones at each time.

In any market, the consumer states what price he will pay for a fish (bid price) and the seller also has a selling price for a fish (ask price).

In the model, the consumers look for precious or common fishes depending on the binary breed-own *fishespreferences*; its value is 0 when the consumer wants the precious fishes conversely the value 1 is for the common ones. There are also consumers that first try to buy precious fishes and if they can't do that, they try to buy the common ones; this kind of consumer is characterized by the value 1 of the binary breed-own *maychange?* and the probability to have this kind of consumer is controlled by a slider. Each consumer wants a different quantity of precious or common fishes that is respectively given by the breed-own *preciousQ* and *commonQ*.

Another important aspect is the loyalty [2]. In fact, in a real market a consumer may decide to buy everyday in the same fish shop because of the closeness or the liking or the price or the habits. In order to reproduce this human behaviour, the consumers can be loyal to a fisher, until he has an

availability of fishes, and this characteristic is controlled by the binary breed-own *loyal?* that in the loyal-case takes the value 1. The probability that a consumer is loyal is controlled by a slider. It is important to underline that if the loyal consumer cannot find the wanted quantity of fish in the usual shop, he resets random his preference in the following cycle remaining however a loyal consumer . The breed-own *timefidelity* counts the number of cycles in which the loyal-consumer does not change the fish shop. The loyal-consumers could also change their favourite shop with a probability decided by a slider. This possible change wants to reproduce events like an argument or a displeasure.

There are also the global variables that are accessible by all agents and can be used anywhere in the model. In this case, the globals are used to count the number of fishes or to make lists of the prices that are used for statistic analysis.

```
globals [meancommonprice meanpreciousprice
         preciouspricestandardeviation commonpricestandardeviation
         preciousprice commonprice
         listpreciousbought listcommonbought
         listminpreprice listmincomprice
         whopre? whocom? initpre initcom]
```

2.2 Actions

2.2.1 Setup

The button *setup* creates the "turtles" and it initializes the variables and the breed-own. Below the more important part of the code of this button.

```
to setup
 clear-all
 create-consumers  number-of-consumers[
   ifelse random-float 1 < precious-rate
     [set fishespreferences 0
      ifelse random-float 1 < loyal-rate
            [set fisherspreferences random initial-number-of-boats
             set loyal? 1]
            [set fisherspreferences initial-number-of-boats
             ifelse random-float 1 < rate-of-precious-move-to-common
                    [set maychange? 1]
                    [set maychange? 0]
             set loyal? 0]]
     [set fishespreferences 1
       ifelse random-float 1 < loyal-rate
             [set fisherspreferences random initial-number-of-boats
              set loyal? 1]
             [set fisherspreferences initial-number-of-boats
              set loyal? 0]]]
ask precious
 [set preciousgroup(random max-num-of-precious-fishes-in-a-group)+1]
```

```
ask common
 [set commongroup(random max-num-of-common-fishes-in-a-group)+1]
ask fishers [set minpreciousPrice random-float 6 + 9
             set mincommonPrice random-float 6 + 6]
ask consumers[set preciousQ random 2 + 1
              set commonQ random 3 + 1
              set maxpreciousPrice random-float 2 + 9
              set maxcommonPrice random-float 2 + 7]
reset-ticks
end
```

First, there is the creation of the sea and of the agents (precious and common fishes, fishers and consumers) and then it is set the size of each school of fish. In the display only fishes and fishers are represented and each fish correspond to a group of random size. Then the bid prices, the ask prices and the fishes' quantities wanted by each consumer are initialized. Besides, the consumers are divided into loyal ones and random ones and both of them have a fixed preference of fish. There are only a kind of consumers that prefers the precious fishes but in lack of them decides to buy the common ones if it is possible.

2.2.2 Go

The button *go* permits to start the simulation of the fishing and of the fish market.

```
to go
  ask precious [move]
  ask common [move]
  ask fishers[boat-move]
  if sum [preciousgroup] of precious > precious-threshold
    [fish-precious
     interact-precious
     prepare-mean-sd-precious
     reproduce-precious
     change-min-precious-price
     change-max-precious-price]
  if sum [commongroup] of common > common-threshold
    [fish-common
     interact-common
     prepare-mean-sd-common
     reproduce-common
     change-min-common-price
     change-max-common-price]
     plot-all
  count-loyal
  count-remainders
  change-loyalty
  reset-all
  if sum [preciousgroup] of precious < precious-threshold and
     sum [commongroup] of common < common-threshold [stop]
```

```
  tick
end

to interact-precious
   take-account-precious-quantity
   negotiate-loyal-precious
   negotiate-precious
   plot-do-precious-without-loyal
   plot-do-precious
end

to interact-common
   take-account-common-quantity
   negotiate-loyal-common
   negotiate-common
   plot-do-common-without-loyal
   plot-do-common
end
```

There is an *if* condition that stops the execution of the model separately for precious and common fishes when the number of the fishes is less than a threshold chosen by a slider.

Interaction fish-fisher

The first action is the movement of the fishes and the fishers' boats. They can move each day maximum four or eight steps but the possible directions are different. Fishes are more agile so can move in all the directions conversely the boat can only turn left or right with a maximum angle of 45°.

```
to move
  rt random random-float 90 - 45
  fd random 8
end
to boat-move
  rt random random-float 360
  fd random 4
end
```

The fishers catch the fishes which are within a radius decided by a slider.

```
to fish-precious
   ask fishers[set preciousnow sum [preciousgroup] of precious
               in-radius radius-of-fishing
               set preciousfished fput preciousnow preciousfished
               ask precious in-radius radius-of-fishing [die]]
end
to fish-common
   ask fishers[set commonnow sum [commongroup] of common
               in-radius radius-of-fishing
               set commonfished fput commonnow commonfished
               ask common in-radius radius-of-fishing [die]]
end
```

If a fish is near to some boats it is caught by the first fisher that is called by the action *to fish*. It is important to underline that in NetLogo the order the fishers are called is shuffled each time so who catches the contended fishes is not the same every time. The number of fishers, or in other words the number of the boats, could be chosen also by a slider and it does not changed during the execution of the model. Instead, the number of fishes is variable; in fact in each step someone is caught by the fishers and then it dies but also someone bears with a growth rate controlled by a slider. The growth rate is the probability that a fish creates a new fish of its breed, so it inherits of all the variables from its parent.

```
to reproduce-precious
ask precious
 [if random-float 1 < growth-rate-of-precious-fishes
  [hatch-precious 1
   [set preciousgroup(random max-num-of-precious-fishes-in-a-group)+1
    setxy random-xcor random-ycor]]]
end

to reproduce-common
ask common
 [if random-float 1 < growth-rate-of-common-fishes
  [hatch-common 1
   [set commongroup(random max-num-of-common-fishes-in-a-group)+1
    setxy random-xcor random-ycor]]]
end
```

Interaction fisher-consumer

The interaction of the agents is split into a precious and a common part in order to end separately the fishing and the negotiation if the quantity of fishes goes under a threshold, decided by a slider.

At the beginning of the day, the fishers, that have just fished, know the daily amount of caught fishes and then they compare the current quantities with the past ones. If the current quantities of precious or common fishes is lower than the past ones they increase their ask price or conversely they decrease it. In this program, the comparison is only between the current quantities and the previous ones and the changing percent of the fishers' prices depend on the slider *rate-of-change-sunrise*. Below only the precious part of the code that is the same as the common one.

```
to take-account-precious-quantity
 if ticks > 2[
 ask fishers with [preciousnow > 0]
   [if item 0 preciousfished > item 1 preciousfished
     [set minpreciousprice minpreciousprice*(1-rate-of-change-sunrise)]
    if item 0 preciousfished < item 1 preciousfished
     [set minpreciousprice minpreciousprice*(1+rate-of-change-sunrise)]]]
end
```

The core of the fish market is the negotiation. First the consumers inspect the fishers' ask price, then they decide if accept it. Both the consumers and the fishers take note of the results of the previous interaction in order to learn somewhat from the past experiences. In that model the initial random ask and bid prices are initialized very different in order to observe during the execution their convergence and their matching thanks to the record of the past errors or successes. It is important to notice that a consumer has only a purchase attempt per day, except the consumers that, after a failed attempt of buying precious fishes, try to buy common ones.

As written before, in order to make the model more proper there are not only random consumers but also loyal ones. The feature of the last ones is that they have a favourite shop and they always buy the fishes until there is an availability. In fact in the loyal case the negotiation goes always well; the buyers accept the fishers' price if it is lower than his own bid price or conversely the sellers and the buyers agree with the mean price between the ask and bid one. However, if the favourite shop does not have the desired quantity of fishes, the loyal consumer cannot buy what he wants so he changes his favourite shop and he starts to be loyal to another seller. Below only the precious part of the code that is the same as the common one.

```
to negotiate-loyal-precious
 ask consumers with [fishespreferences=0 and bought?=0 and loyal?=1]
  [ifelse count fishers with [who=[fisherspreferences] of myself and
    preciousnow > [preciousQ] of myself]> 0
   [let sellingfisher one-of fishers with
       [preciousnow > 0 and who = [fisherspreferences] of myself]
    let minimum list ([minpreciousPrice] of sellingfisher)
      ((maxpreciousPrice+[minpreciousPrice] of sellingfisher)/2)
    let price min minimum
    set recordofshop fput [who] of sellingfisher recordofshop
    set settledpreciousPrice price
    set listofsettledpreciousPrice fput price listofsettledpreciousPrice
    set preciousprice fput price preciousprice
    repeat preciousQ [set listpreciousbought fput price listpreciousbought]
    set bought? 1
    set recordbought? fput 1 recordbought?
    set timefidelity timefidelity + 1
    ask sellingfisher[set preciousnow (preciousnow-[preciousQ] of myself)]]
   [set fisherspreferences random initial-number-of-boats
    set timefidelity 0
    set recordbought? fput 0 recordbought?]]
end
```

In the model, there is the possibility that a loyal consumer changes his favourite shop with a probability fixed by the slider *rate-change-loyalty*.

```
to change-loyalty
  ask consumers with [loyal? = 1]
   [if timefidelity > 4
```

```
    [if random-float 1 < rate-change-loyalty
     [set timefidelity 0]]]
end
```

The loyal-negotiation happens before the random one because it is as if the sellers put aside the commodities for the loyal ones because they guarantee them a sure and maybe greater income. So, for the random consumers is not available all the initial amount of fishes.

The random consumers does not have a favourite shop so they compare the price of a random number of shops that have the availability of desired fishes. This could happens because the number of shop is small so all of them are in the same marketplace or anyway they are close to each other. The number of prices that a consumer compares is random because the confrontation takes time and not all the consumers have the patience to do that. There also a slider *max-diff-accepted* that take into account how much a consumer could spend more than his price to secure to buy some fishes. In other words, this slider is an index of the will of the consumers to obtain the fishes. It is important to remember that the fishes are a perishable commodity so the interest of the fishers is selling all of them in a day. To make this possible, they change their price during the day in order to encourage the consumers with low prices if the past interactions are unsuccessful or conversely they raise the prices when the demand is high. The slider *rate-of-change-inaday* sets the changing percent of the fishers' prices during the day. Besides, the model should be initialized so that the asked quantity of fishes is a bit more than the offered one to encourage the competition between the consumers during the execution. Below only the precious part of the code that is the same as the common one.

```
to negotiate-precious
  ask consumers with [fishespreferences = 0 and loyal? = 0 and bought? = 0][
   if count fishers with [preciousnow > [preciousQ] of myself ] > 0
    [let num count fishers with [preciousnow > [preciousQ] of myself]
  ask n-of(random num + 1) fishers with [preciousnow >[preciousQ] of myself]
   [set listminpreprice fput minpreciousPrice listminpreprice]
  let minprice min listminpreprice
  let sellingfisher one-of fishers with
   [preciousnow > [preciousQ] of myself and minpreciousPrice = minprice]
  set whopre? [who] of sellingfisher
  let diff maxpreciousPrice - minprice
  ifelse diff >=  -1 * max-diff-accepted
   [set preciousprice fput minprice preciousprice
    set listofsettledpreciousPrice fput minprice listofsettledpreciousPrice
    set preciousprice fput minprice preciousprice
    repeat preciousQ[set listpreciousbought fput minprice listpreciousbought]
    set settledpreciousPrice minprice
    set bought? 1
    set recordbought? fput 1 recordbought?
    ask fishers with [who = whopre?]
     [set preciousnow (preciousnow - [preciousQ] of myself)
```

```
      set recordpreciousinteraction fput 1 recordpreciousinteraction]]
   [set recordbought? fput 0 recordbought?
    ask fishers with [who = whopre?]
     [set recordpreciousinteraction fput 0 recordpreciousinteraction]]
  set listminpreprice []
  ask fishers with [who = whopre?][if length recordpreciousinteraction > 3
   [if item 0 recordpreciousinteraction=1 and item 1
    recordpreciousinteraction = 1 and item 2 recordpreciousinteraction = 1
      [set minpreciousPrice minpreciousPrice * (1 + rate-of-change-inaday)]
    if item 0 recordpreciousinteraction=0 and item 1
    recordpreciousinteraction = 0 and item 2 recordpreciousinteraction = 0
      [set minpreciousPrice minpreciousPrice*(1-rate-of-change-inaday)]]]]]
end
```

All the consumers are interested either in precious fishes or in common ones except the buyers characterized by the breed-own *maychange?*=1. These consumers first try to buy the precious fishes and if they cannot, the negotiation is classified as unsuccessful and then they try to obtain the common ones. In the model, the changes of the prices involve only the first preference of the fish because this is the main target of the consumers; buying common fishes does not make them very satisfied because they prefer the precious ones and in the future they will change the precious price in order to buy them. It is important to note that the precious price is different from the common one and for this agent the last one does not change during the execution.

```
ask consumers with
 [fishespreferences = 0 and loyal? = 0 and bought? = 0 and maychange? = 1]
  [if count fishers with [commonnow > [commonQ] of myself] > 0
   [let sellingfisher one-of fishers with [commonnow > [commonQ]of myself ]
    let diff maxcommonPrice - [mincommonPrice] of sellingfisher
    ifelse diff >=  -1 * max-diff-accepted
     [set commonprice fput [mincommonPrice] of sellingfisher commonprice
      set listofsettledcommonPrice
       fput [mincommonPrice] of sellingfisher listofsettledcommonPrice
      set commonprice fput [mincommonPrice] of sellingfisher  commonprice
      repeat commonQ [set listcommonbought
       fput  [mincommonPrice] of sellingfisher listcommonbought]
      set settledcommonPrice [mincommonPrice] of sellingfisher
      set boughtcommon? 1
      ask sellingfisher
       [set commonnow (commonnow - [commonQ]of myself)
        set recordcommoninteraction fput 1 recordcommoninteraction]]
     [set boughtcommon? 0
      ask sellingfisher
       [set recordcommoninteraction fput 0 recordcommoninteraction]]]]
```

At the end of the day, both the consumers and the fishers check the results of the negotiation and they correct their own price. In particular, a fisher that has unsold fishes for two consecutive days , decides to ask less the following day and conversely a consumer that cannot buy fishes for two consecutive

days, makes higher his bid price. The changing percent of the consumers' prices and of the fishers' ones is respectively established by the slider *rate-of-change-maxprice* and *rate-of-change-minprice*. Below only the precious part of the code that is the same as the common one.

```
to change-min-precious-price
 if ticks > 2[ask fishers
 [if item 0 listofpreciousremainders<2 and item 1 listofpreciousremainders<2
     [set minpreciousPrice (1 + rate-of-change-minprice) * minpreciousPrice]
  if item 0 listofpreciousremainders>=2 and item 1 listofpreciousremainders>=2
     [set minpreciousPrice (1 - rate-of-change-minprice)*minpreciousPrice]]]
end
to change-max-precious-price
  if ticks > 2[ask consumers with [loyal? = 0]
   [if item 0 recordbought? = 0 and item 1 recordbought? = 0
     [set maxpreciousPrice (1 + rate-of-change-maxprice) * maxpreciousPrice]
    if item 0 recordbought? = 1 and item 1 recordbought? = 1
     [set maxpreciousPrice (1 - rate-of-change-maxprice)*maxpreciousPrice]]]
end
```

Plots

In the model there are four kinds of plots: one about the record of the number of the precious and common fishes remained in the sea (*fishes' population*), one about the daily precious and common settled prices (*Q-fishes*), one about the record of the daily mean settled price and the daily deviation standard (*precious-price* and *common-price*) and others about the supply and demand schedule of the precious and loyal fishes, with and without the loyal consumers (*d/o curves*).

To prepare the plot *Q-fishes* it is necessary to make a vector *listpreciousbought* or *listcommonbought* containing the settled price of each successful interaction. The prices are referred to a single unit of fish so in these vectors each price is repeated as many times as many quantity of fishes is sold at a certain price. The vectors are sorted and then plotted.

For the demand and offer curves are just the same. After making the vectors of the ask and bid prices of all the agents or all except the loyal consumers, depend on the graph, these one are sorted and then plotted. Because of the great difference between the number of the consumers and the fishers, this graph is not meaningful but it gives an idea of the distributions of the prices.

The graphs *precious-price* and *common-price* are about the record of the daily mean price and the daily deviation standard of precious and common settled prices and it need the vectors of the daily settled prices that are used to value the daily mean and the daily deviation standard.

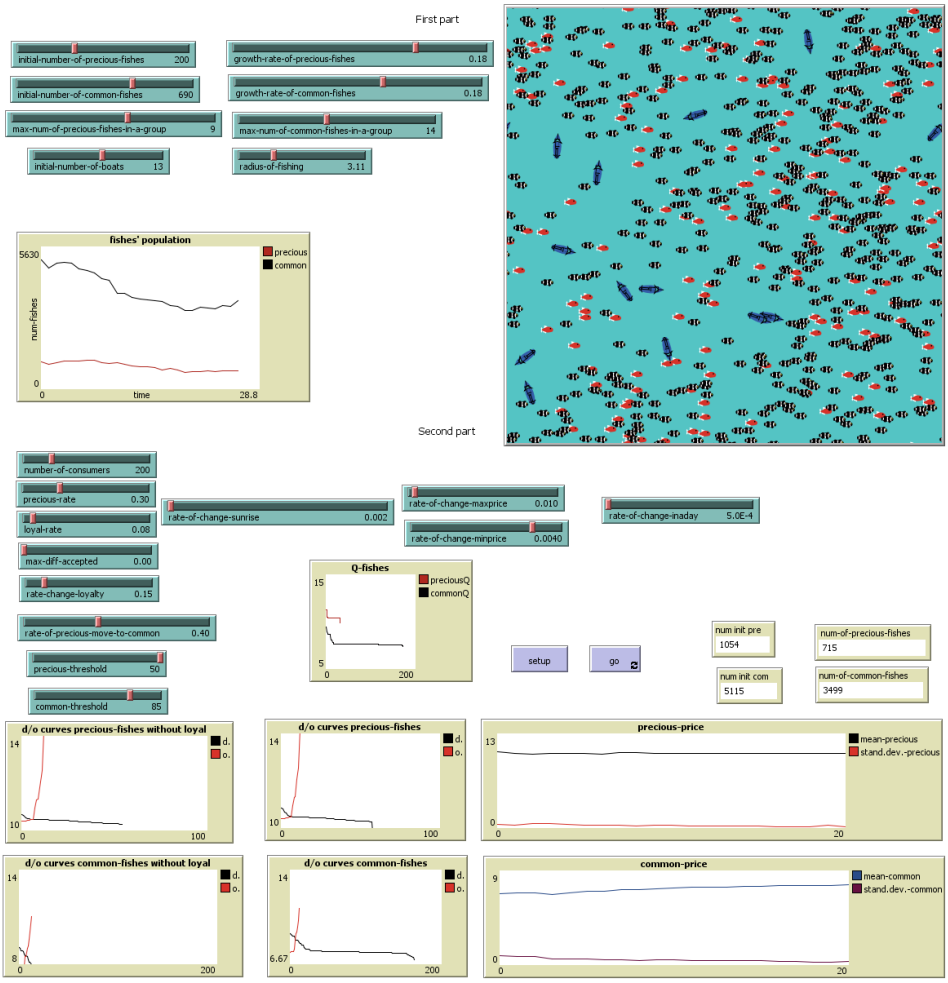


Figure 1: The interface of the program

3 Results

The first evidence is that a simulation is not mainly reproducible because of the first part, the interaction fishes-fishers, that is random and then the number of caught fishes is not the same restarting the simulation with the same parameters. So it is not possible to make the same experience with different value of the sliders *loyal-rate* or *precious-rate* or *number-of-consumers*.

The second evidence is that there is an initial transient in the graph *precious-price* and *common-price* because of the convergence of the ask and bid prices. This occurrence could be also observed in the *Q-fishes* graph (Fig.9). During the first day there is a great discordance in the daily settled prices both for precious and common fishes that decreases during the evolution of the model. This evidence is pursued initializing very different the ask and bid prices in order to have confirmation of the truthfulness of the model.

3.1 Simulations

First, the sliders are set in order to preserve the number of the fishes in the sea during the simulation and to guarantee enough, nay a bit less, number of caught fishes respect to the demand. In this way, the competition between the consumers is encouraged. After some attempts, a reasonable setting of the sliders could be seen in the figures 2 and 3.

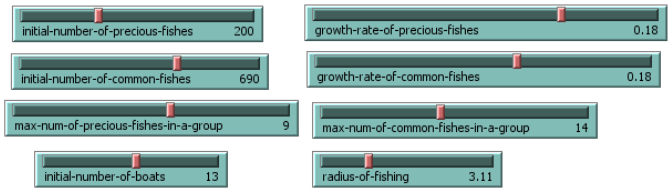


Figure 2: Sliders' settings of the first part of the program

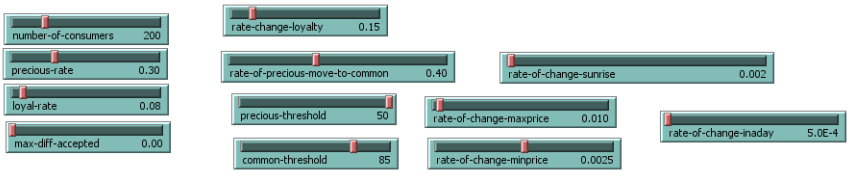


Figure 3: Sliders' settings of the second part of the program

The first simulation is made using these values (Figs. 2, 3) and after about 200 cycles the result is shown in figure 4. In the graph *Q-fishes* there is only

the common curve because the precious fishes are under the threshold so the precious part of the program is ended. This could be noticed looking the graph *precious-price* in which at the end there are a zero and then a constant line that represent the break of the simulation. Could be also noticed that the prices increase when the amount of fishes decreases.

The second simulation is made using the same value of the first simulation except the slider *number-of-consumers* that is set 600 instead of 200. The demand is so increased and then the prices increase (Fig. 5).

The third simulation is made using the same value of the first one except the slider *precious-rate* that is set 0.7 instead of 0.3. In this simulation there is conversely an increase of the population of the fishes, the offer is greater then the demand so the prices decrease (Fig.6). The rise of the number of the precious consumers does not produce evident changes. In fact to observe any alterations it is necessary to compare the same simulation with different values of the sliders but it is not possible because of the randomness of the first part of the program.

The fourth simulation is made using the same value of the first one except the slider *loyal-rate* that is set 0.4 instead of 0.08. This simulation ends when both the types of fishes go under the threshold so the offer decreases at the end and then the prices increase (Fig.7). The rise of the number of the loyal consumers does not produce evident changes in the graphs for the reasons stated above.

The fifth simulation is made using the same value of the first one except the slider *growth-rate-of-precious-fishes* that is set 0.9 instead of 0.18. This simulation is done to underline the importance of how the sliders are set. In fact, a reasonable setting avoids that the simulation end soon (Fig.8).

In the graphs of the previous simulations the standard deviation is not viewed because the graphs are focused on the record of the mean price in order to see them more accurately. So, the first simulation is then repeated splitting into two different graphs the record of the mean prices and the deviation standard. Besides it is added the graph *catch fishes* about the record of the quantity of the caught fishes at each cycle. These temporary graphs (Figs.10, 11) show us how the standard deviation, so the heterogeneity between the prices, is high at the beginning of the simulation or when the amount of fishes becomes too large or too small. Moreover, the fact that the graph of the number of the fishes in the sea has the same trend of the graph of the caught ones provides a proof of the truthfulness of the fishing action.

3.2 Conclusions

The model is effective and truthful but it can still be improved. For example, the initial position and the movement of the boats could be optimized or a more difficult modality of fishes' breeding can be used or it can be periodically introduced a season when fishing is prohibited that allows the fishes to grow

and to reproduce. The problem of the model is its high randomness, mainly in the first part. A solution could be repeat the first part many times and then take the average quantities for the second part. An alternative might be separating the first and second part. First the interaction fishes-fishers runs and produces the record of the caught fishes. Then, from these data the second part of the program runs many times with different values of the slider in order to understand the significance of the *loyal-rate* and the *precious-rate*. Besides, the model and the results of the article [2], that is about the wholesale fish market, could be used to make a very complete fish market. In fact normally, the fishers are not both seller and fishers as we have considered in this model.

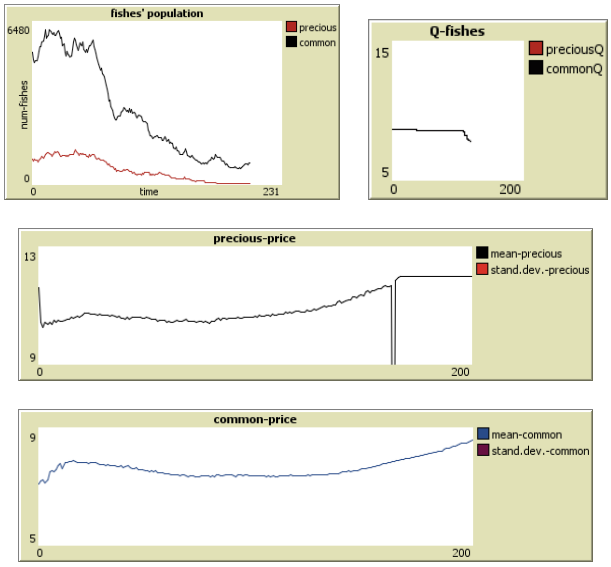


Figure 4: a) *fishes' population*, b) *Q-fishes* c) *precious-price* d) *common-price*

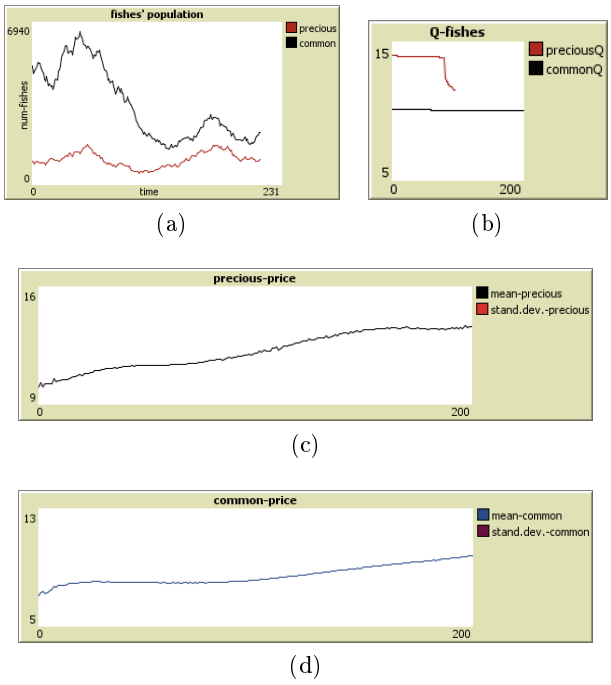
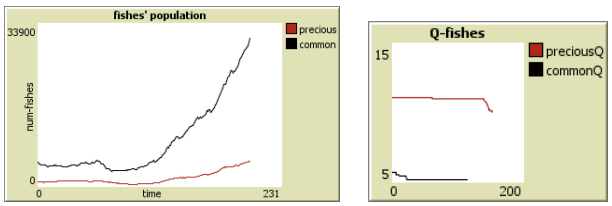
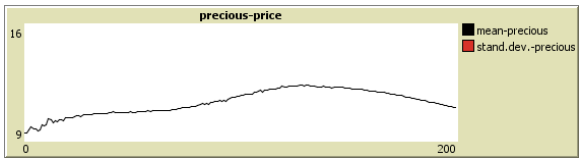


Figure 5: a) *fishes' population*, b) *Q-fishes* c) *precious-price* d) *common-price* *number-of-consumers* = 600



(a)

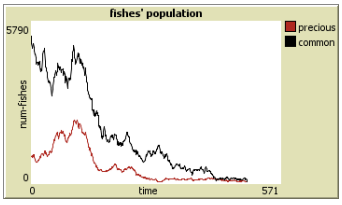
(b)



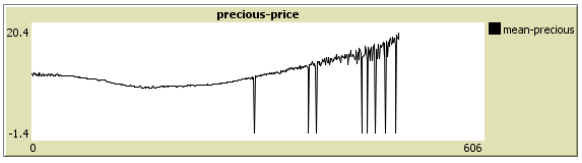
(c)

(d)

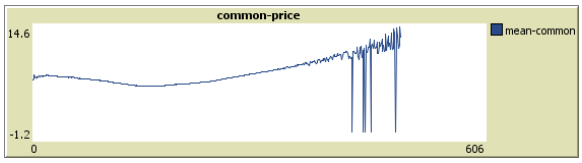
Figure 6: a) *fishes' population*, b) *Q-fishes* c) *precious-price* d) *common-price* *precious-rate* = 0.7



(a)

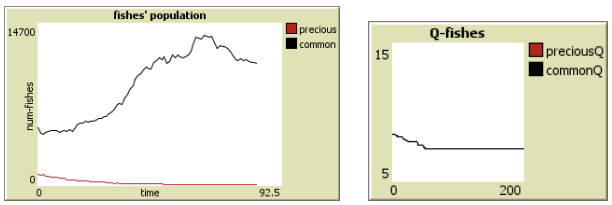


(b)



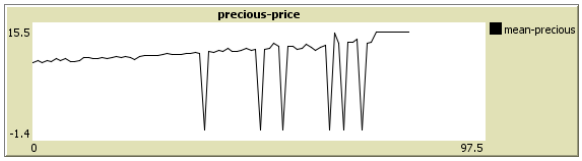
(c)

Figure 7: a) *fishes' population*, b) *precious-price* c) *common-price* *loyal-rate* = 0.4

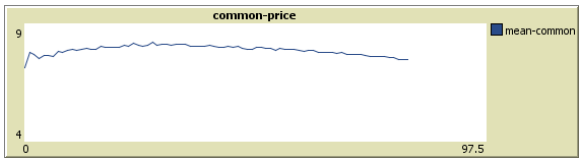


(a)

(b)

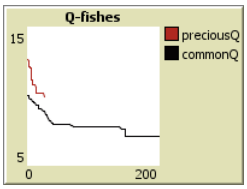


(c)

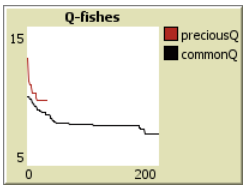


(d)

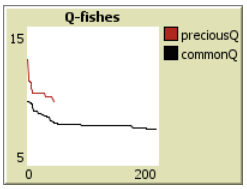
Figure 8: a) *fishes' population*, b) *Q-fishes* c) *precious-price* d) *common-price* *growth-rate-of-precious-fishes* = 0.9



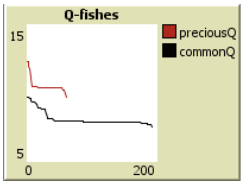
(a)



(b)

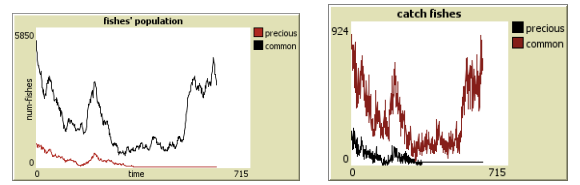


(c)



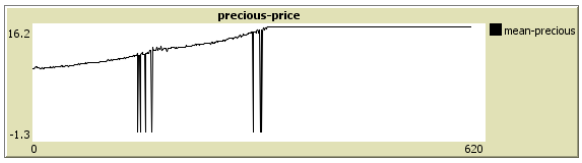
(d)

Figure 9: *Q-fishes* after a) 1 cycle, b) 5 cycles c) 10 cycles d) 20 cycles

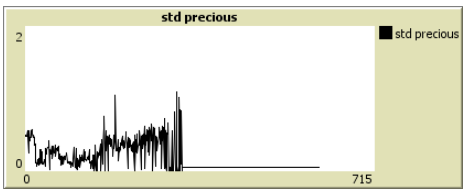


(a)

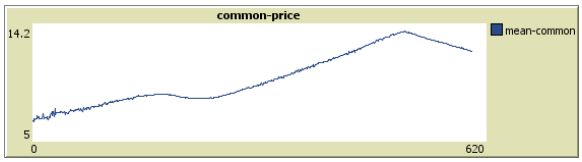
(b)



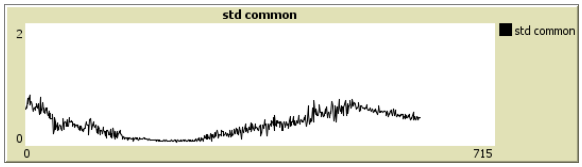
(c)



(d)

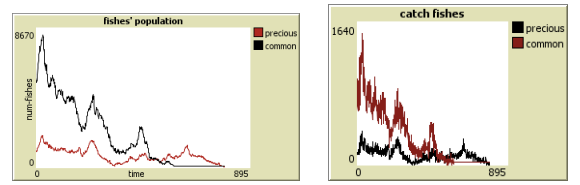


(e)



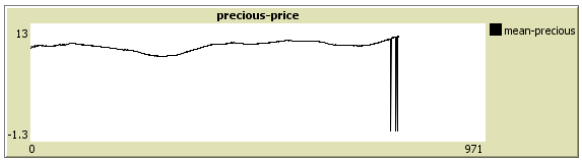
(f)

Figure 10: a) *fishes' population*, b) *catch fishes* c) *precious-price* d) *std precious* e)*common-price* f) *std common*

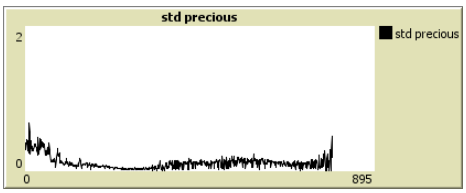


(a)

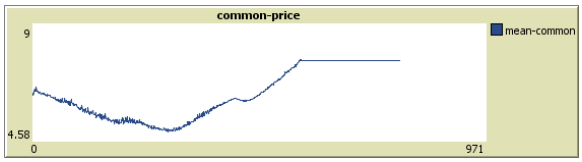
(b)



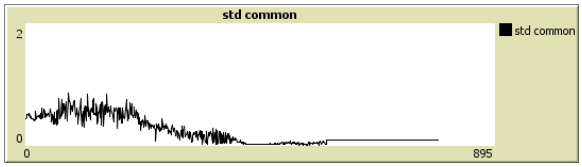
(c)



(d)



(e)



(f)

Figure 11: a) *fishes' population*, b) *catch fishes* c) *precious-price* d) *std precious* e)*common-price* f) *std common*

References

[1] NetLogo User Manual, https://ccl.northwestern.edu/netlogo/docs/

[2] Alan P. Kirman, Nicolaas J. Vriend, *Evolving market structure: An ACE model of price dispersion and loyalty.* Journal of Economic Dynamics & Control, 25 (2001) 459-502. http://econ2.econ.iastate.edu/tesfatsi/KirmanVriend.EvolvingMarketStructure.pdf