UNIVERSITÀ
DEGLI STUDI
DI TORINO

ALMA UNIVERSITAS
TAURINENSIS

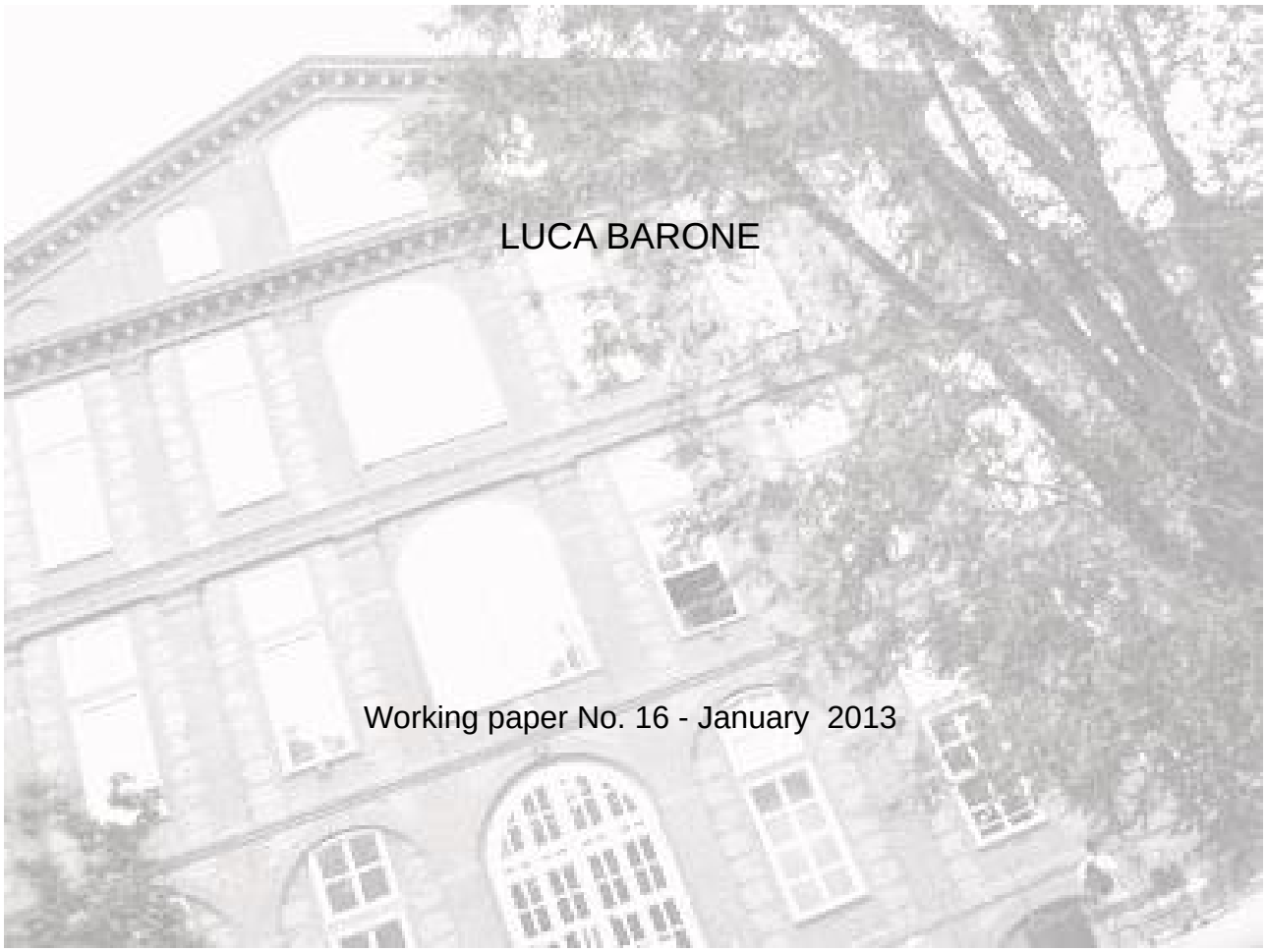
Founded in 1404

DEPARTMENT OF
ECONOMICS AND STATISTICS
WORKING PAPER SERIES

Quaderni del Dipartimento di Scienze
Economico-Sociali e Matematico-Statistiche

ISSN 2279-7114

AN ABM FOR ECONOMICS: MICRO EXPLAINS MACRO



LUCA BARONE

Working paper No. 16 - January 2013

An ABM for Economics: Micro explains Macro

Luca Barone (University of Turin)

24th November 2012

Abstract

The link between micro and macro level has always been difficult to trace, even when variables have strong homogeneous characteristics. What happens when heterogeneous components and random factors interact is even more difficult to define.

By adopting an agent-based approach we found a result that does not reflects the classical methods of quantification of an economy. This can be interpreted as an alarm bell signaling a wrong description of the economic framework we want to explain. We illustrate the effectiveness of the "agent-based reasoning machine" and we derive a model to compare with classical methods of aggregation.

A more comprehensible description of the model is given by "Unified Modeling Language (UML)" and "ODD standard protocol", allowing us to clarify the internal processes of our model.

Keywords: Aggregation, NetLogo, Simulations, Micro-Macro link, Agent Based Models (ABMs), Unified Modeling Language (UML), ODD standard protocol

Contents

1 Introduction **3**

2 Simulation Model for Aggregation **6**
2.1 NetLogo Model 7
2.1.1 Extraction of dataset 8
2.2 *R* results and plots 9
2.3 Simulations 11
2.4 Aggregation techniques 12
2.4.1 Plain Model for Aggregation 14
2.4.2 Micro Simulation Model for Aggregation 14
2.4.3 ABM for Aggregation 15
2.4.4 Comparison 15

3 General conclusions **17**

A Appendix A: NetLogo code **20**

B Appendix B: *R* code **33**

C Appendix C: Representations of Simulation for Aggregated Consumption and Production function **38**
C.1 Unified Modeling Language (UML) 38
C.1.1 Class diagram 39
C.1.2 Sequence diagram 41
C.1.3 State diagram 42
C.1.4 Activity diagram 42
C.2 ODD Standard Protocol 43
C.2.1 Purpose 47
C.2.2 Entities, state variables and scales 47
C.2.3 Process overview and scheduling 48
C.2.4 Design concepts 48
C.2.5 Initialization 49
C.2.6 Input data 50
C.2.7 Submodels 50

1 Introduction

The problem of aggregation is one of the fundamental economic steps that has not been completely solved so far. How to connect an individual to an aggregate function is not clear yet. In other words, we have not found the function that allows us to connect the micro to the macro level and/or vice versa.

Vartia has tried to construct a theory that tied these two variables significantly. From the beginning, there has seemed like an important interpretation of the problem, and we thought that could fill the void left by the economic theory of reference. In part, our impression was correct. He attributes the difference between micro and macro level variables to a covariance between the terms constituting a function under consideration (e.g. the consumption function, Godley (1999)), which in some cases may be canceled, causing a perfect equality between the two levels under study.

Do the parts determine the whole or is the whole more than the sum of its parts?

We refer to the theory of Vartia, professor at the University of Helsinki, building up a significant practical case attributed to the example of the consumption function and the marginal propensity to consume. Vartia's contribution to the problem of aggregation can be found through his main papers:

- On the Aggregation of Quadratic Micro Equation, Vartia (2008b);
- Whole and its Parts Micro Foundations of Macro Behaviour, Vartia (2009);
- Analysis and Synthesis of Wage Determination in Heterogeneous Cross-sections, Suoperä & Vartia (2011);
- Integration of Micro and Macro Explanations, Vartia (2008a);
- Relative Changes and Index Numbers, Vartia (1976);
- How should relative changes be measured? Törnqvist *et al.* (1985).

Since we believe that these cases are restrictive, we decide to focus on the concept of equal sign between the two levels plus a covariance term that explains precisely the differences that emerge. Subsequently, we found that the theory produces a valid interpretation of the problem of aggregation but something is missing. No mention about the sign of the covariance is made: we do not know if covariance term is increasing or decreasing or whether it is negative or positive.

So how can we know if the macro level overestimates the micro level or the micro level is underestimated by the macro level?

Unfortunately, we still have no answer. Our task, through this work, was groped to make a clarification about the sign of covariance and, hence, about the nature of the aggregation error.

Although we retrace the key concepts of the theory of Vartia, we will develop our analysis on a different theme: the aggregation is analyzed mainly under the point of view of the production function, Friedman (1957), the aggregate domestic product, Burbidge & Cuff (2005), using a simple Agent Based Model we created through NetLogo. Hence, the distinctive character of our work is attributed to the composition of individual production functions, since they are derived from an agent based model (ABM). Normally, individual production functions refer to a sample mean of a given period of time with reference to some place. In our case, they are calculated using a model created in the computer. We build our own model, we define the parameters and variables of the model, we decide how long the time-series dataset should be and how many agents we want to consider. In this way, we can give a greater breadth to the range of individual production functions, by evaluating the factors that emerge from the interactions of agents, the factors of imitation behavior and all the random factors.

The central part of our discussion is the calculation method to derive the aggregate results from an individual production function. Three different methods are shown, starting from the simplest to the most complete. Giving a brief explanation about their differences and how they originated, we conclude by providing our interpretation on findings and by proving that changes on aggregate results are substantial.

The first two methods are named *Plain* and *Micro Simulation*. Such approaches can be combined with the practice of aggregation used by the System of National Accounts (SNA) or other entities that deal with the problem of aggregation. It refers to the simple summation of the individual functions (Plain) that can sometimes be weighted (Micro Simulation) according to methods of calculation.

In the last method, the Agent Based Model, our tool to aggregate is *R*. It receives as input a dataset of individual production functions from NetLogo and, through a quantitative process, outputs the aggregate result. That is an innovative method that could prove a key tool for reducing the error of aggregation or explaining its causes.

Tied to the agent-based methodology, we obtain and quantify, through various simulations, the covariance among variables of individual production functions. That value of covariance between individual behaviors,

which in our case turns out to be negative for the production function, allows us to explain the error of assessment between micro and macro levels. In our Agent Based Model the production function is lower where individuals are not covered as independent entities but act by interfering with one another. As a consequence, it is completely wrong to attribute to each individual either the same distinctive features or the equal propensity to preferences in behavior (i.e. do not consider the covariance term).

Unfortunately, it is still difficult to understand in detail the reason for which the function is less than the Plain and Micro Simulation if not attributing a technical meaning resulting from the assumptions incorporated during the construction of the model or from the agents' interactions. In order to allow readers to their own interpretation on the results obtained, we opted to represent the model either with the *Unified Modeling Language (UML)* or the *ODD Standard Protocol*; being the first the most effective under the point of view of the representation of the "mechanical sequence of behavioral course", i.e. it is closer to the practical and technical ABM technique, while the second more theoretical and useful for readers that are not interested in the technicalities of the model.

Probably, our results can not be defined as definitive yet, since our model is an easy simplification of an economic scenario. However, we can say that our "*Agent Based Reasoning Machine*" provides some interesting results and helps to outline a step forward in the discussion on the issue of aggregation.

Our call is to employ this tool for reasoning in the future, implementing the model we constructed to make it closer to reality, in order to obtain further explanations about the covariance between macro and micro level, i.e. the error of aggregation.

The following section is the main part of our work. It is divided into four major phases.

The first covers either the illustration of the model that has ben created with NetLogo, by which we construct our dataset of individual production functions or the extraction technique used to output the dataset.

The second phase treats our aggregation tool: *R*. Results that have been found by *R* are attached, together with plots, and are discussed.

The third is very relevant. It deals with simulations that allow us to calculate an average result rather than just one. This is necessary because agent-based models emit different results each time.

The last phase, the most important, concerns the comparison of the three different methods of aggregation: *Plain, Micro Simulation* and *ABM*.

All three methods are widely explained and discussed. A further comment is also present in the general conclusions.

Appendix A lists the NetLogo programming code of our model. Such Appendix can be used for the detailed explanation of model code that allows us to create and output the dataset of individual production functions discussed in section (2.1).

Appendix B produces the *R* code that is used to aggregate individual production functions whose findings are shown in section (2.2).

Appendix C outlines two mechanisms of describing ABMs: UML and ODD standard protocol. The first structures the model in several classes that interact with different order. Each interaction is drawn through diagrams which show the type of relationship. Diagrams can draw divisions between classes, timelines or individual and / or aggregate behaviors. The second is more verbal and describes in words what happens within the model by dividing arguments according to a standard protocol for reading and writing. The two mechanisms combined give a complete view of the model and allow for every type of reader to understand how it works.

2 Simulation Model for Aggregation

This chapter develops an agent-based model that will help us to discuss the problem of aggregation. This model is derived from the fusion of two other models taken from the library of NetLogo: "Sugarscape, Epstein & Axtell (1996) and Wilensky (1999)" and "Urban Suite - Economic Disparity, Felsen & Wilensky (2007) and Wilensky (1999)". The main feature of our model is an additional code that allows us to extract a dataset to be processed for our aggregation. Despite that, the model will be an easy simplification of reality. For simplicity we decided to use *R* for data processing, having excellent ability to communicate with NetLogo.

We split the session by starting with a general description of the model, the main features and what are the agents' behaviors. Secondly, we describe the code, emphasizing its economics meaning rather than the technical content of the NetLogo language. Then, we explain our technique of extraction of the dataset that was created by NetLogo. We come to understand as *R* can use the dataset extracted and how is possible to derive aggregated results. Finally, we compare the *R* aggregated results of NetLogo individual production functions dataset with other 2 ways used for aggregation. We thus outline the differences among the three methods, pros and cons and we discuss concerning the best technique.

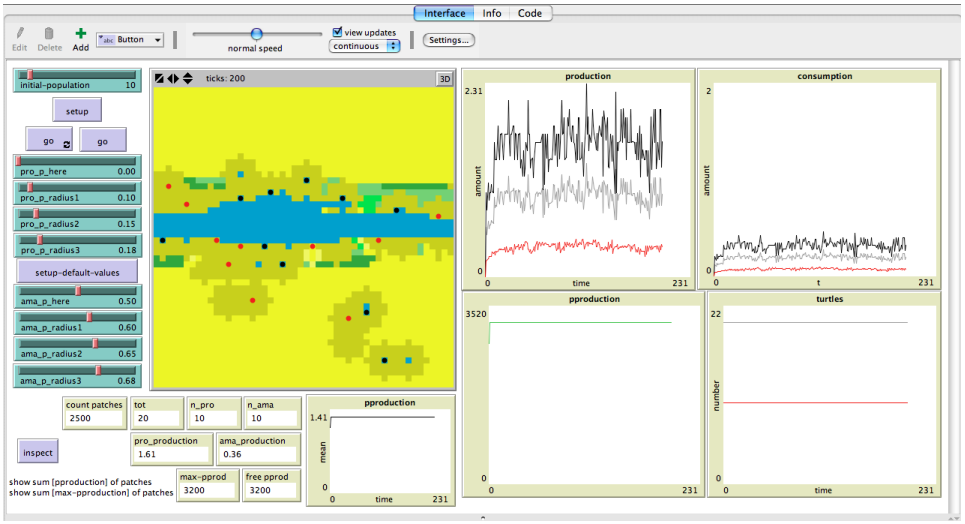


Figure 1: NetLogo interface (after 200 ticks)

2.1 NetLogo Model

Figure 1[1] shows the NetLogo interface of the model.

We set up two breeds of turtles, graphically the blacks and the reds, that have different peculiarities. Same for the patches that contain resources. According with the amount of resources, they have different colors, for instance the blue means no resources, the green the maximum amount of resources and the yellow the lowest level of resources. Think about a field where blu is water, yellow is a parched land and green is a rich soil. Agents can move on the ground searching for the best areas where to exploit resources. As soon as they find the best area, they start to produce, hence, to extract resources from the soil. At the same time they produce, they have to consume a part of their production. For this reason, same of the resources are needed for the energy requirement and the left over is considered as the net turnover. Naturally, when an agent exploits resources from a field, time has to pass to restore the initial condition of resources. In addition, agents have the power to exploit also the neighboring patches and not the patch where they are only. Imagine as they spoil the surrounding area as a result of their activity on one patch. At last, every agent can recognize the best areas where to move on. This is an important skill that helps them to distinguish areas where other agents have already located on to the empty ones or, basing the decision directly on the quantity of resources, the area having the maximum amount of resources from the parched lands.

[1]Figure 1 shows the interface after 200 ticks have past

To sum up, by locating on patches, agents gain resources, increasing either their production or consumption counter, and decreasing the amount of resources hold by patches. They can see where they have better go and their strategies strictly depend on other agents' actions.

The idea of this model is to build up some exogenous factors that affect the production function, in order to have a non trivial results. Different factors can affect the agent's production. First of all, we equip agents with the capability to single out the best areas from the rest of the others. They can see the amount of resources that a patch has. Secondly, they affect both the patch where they are located and the surrounding ones causing a deviation to other agents' choices. Then, we set the maximum amount of resources of each patch limiting the range of actions of the population. Finally, we assign different technologies of production to our agents. Blacks are more efficient than reds. They produce more, having a more advance technology that also leads to a higher exploitation of the soil and a greater impact of the surrounding areas.

To conclude, here interactions play an important role and provide non-trivial results concerning both the individual and the aggregate level.

2.1.1 Extraction of dataset

The NetLogo model has the main aim to create a dataset that is used for aggregation. Usually, dataset comes from surveys or historical records own by public institutions. In our case, it comes from an ABM: by running simulations, it provides the dataset we need for the aggregation analysis. This is a very innovative way of evaluating the aggregation issue, since the existent literature does not supply any meaningful example that is linked with such an ABM approach.

Technically, it saves 20 text files for both breed of agents having the production data of each agent over time. Note that we are assuming that the population is composed by 20 agents, 10 *pro* and 10 *ama* in order to keep the extraction and importation of the dataset simple. However, It is not a big deal to extend our analysis to a larger population. NetLogo simplifies our task by assigning to each agent a number starting from 0 to the number of agents we would have in the model (in the example the total number is set to 10, hence, we will have 10 *pro* and 10 *ama*). Due to the fact the the enumeration starts with agent *pro*, the *pro* will be enumerated from *pro0* to *pro9*. *ama* instead, will be from *ama10* to *ama19*.

2.2 *R* results and plots

We now attach the *R* outputs. Figure 2 and Figure 3 draw the regression graph and the density plot of the above-mentioned *prod_average* respectively, outlining the deviations from the general mean. Further, Figure 4 and Figure 5 furnish details regarding the individual production function densities of every agent belonging to each breed (*pro* agents first and *ama* subsequently).

Representations in Figure 4 and Figure 5 show that agents are extremely heterogeneous with respect to each other. However, agents do present many common features and opportunities to have the same values of production.

Why such results present so different values if agents have mostly the same shapes?

Fundamentally, two reasons need to be outlined. On the one hand, defining agents with "the same shape" is completely wrong. While building up the NetLogo model, we imposed some different characteristics to agents. For instance, the capability to find out resources and the technology of production. In addition, further, the locations, where they were distributed on, were completely random. All of these features do contribute to vary the production function values. On the other hand, we should recall the concept of *Interaction* and *Emergence*, Squazzoni (2012). The former, can be defined as the agents' power to modify their choices with respect to the somebody else's choices and to output a completely unexpected result. The latter, instead, is the idea to the evolution of complex systems that develop over time. The two are strictly linked and lead to different agents' behaviors. Normally, hundreds or thousands of independent "agents" interact by operating concurrently or cooperating and, as a result, we end up with non-obvious properties, contrasting with any expectation.

To conclude, it is not completely correct to award ABM technique the all merit of furnishing unexpected results, rather, we do stress that some assumptions were made in order to create heterogeneity among individuals.

Remember that our model does not consider many of the differences that distinguish a real economic scenario. Imagine what would happen if the model tried to explain a more complicated framework: results cannot be anticipated in any way.

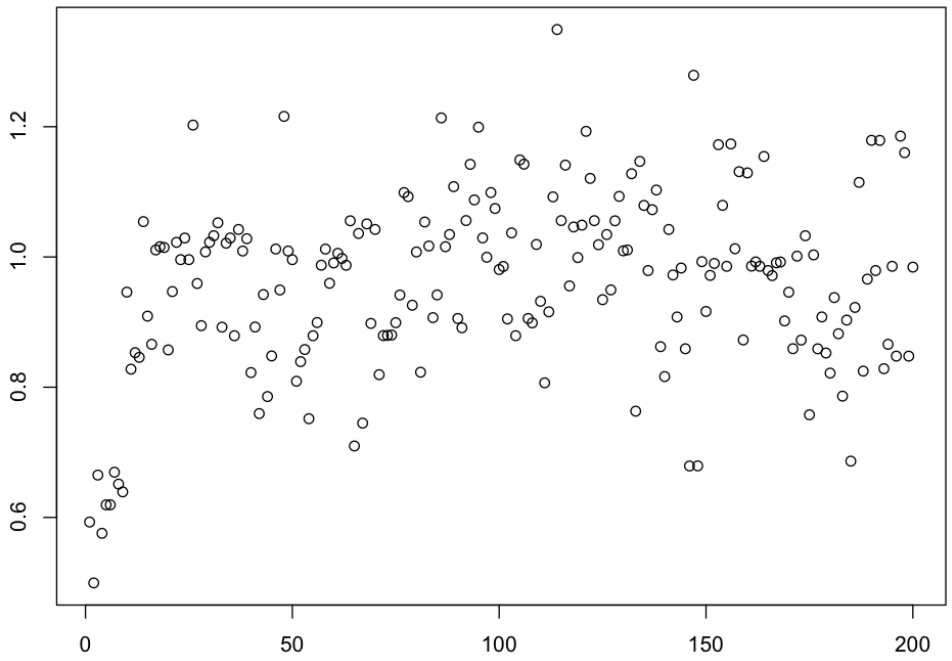


Figure 2: *prod_average* regression plot

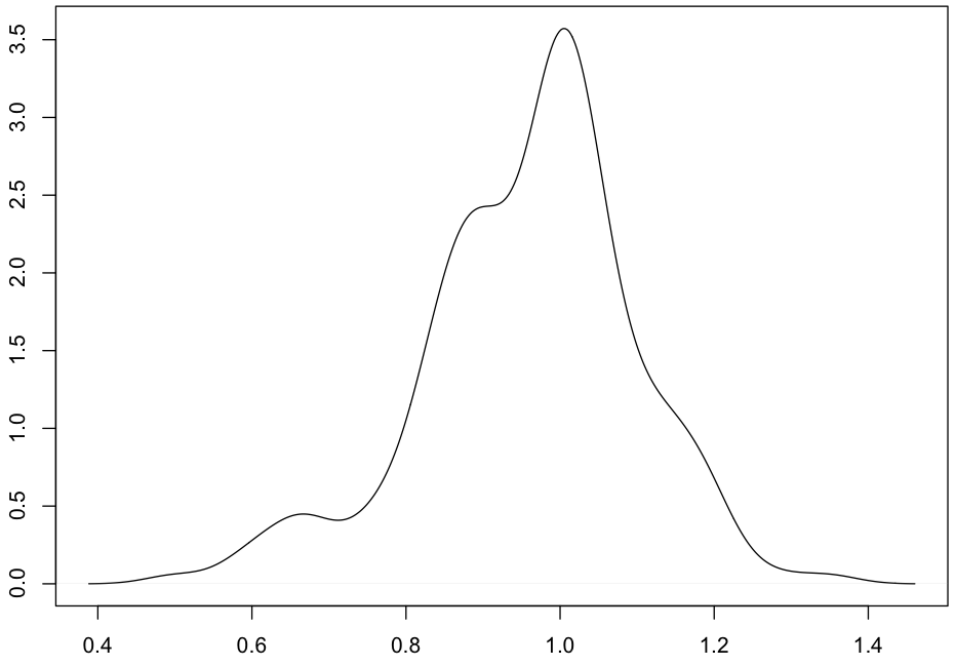


Figure 3: *prod_average* density plot

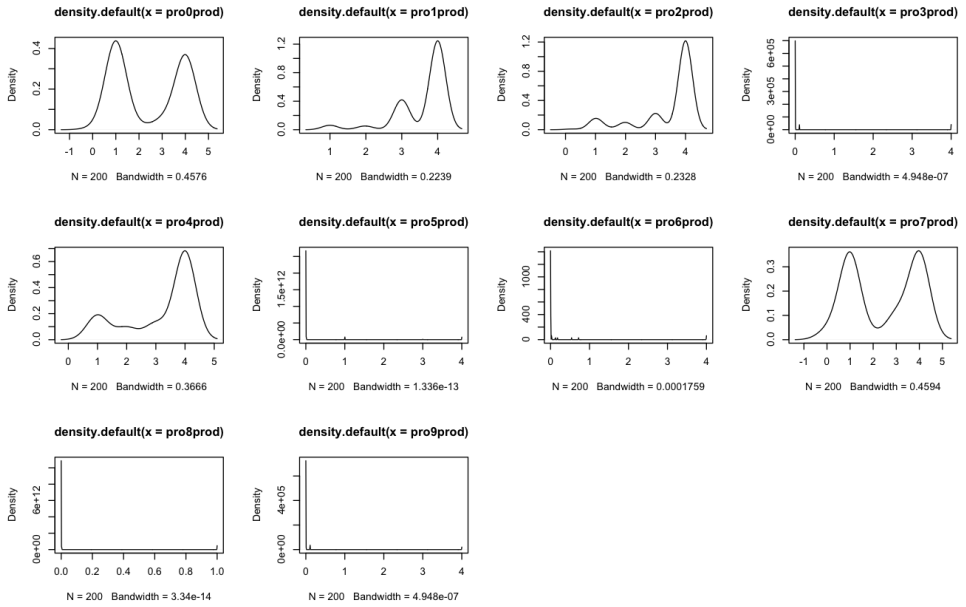


Figure 4: *pro* production densities

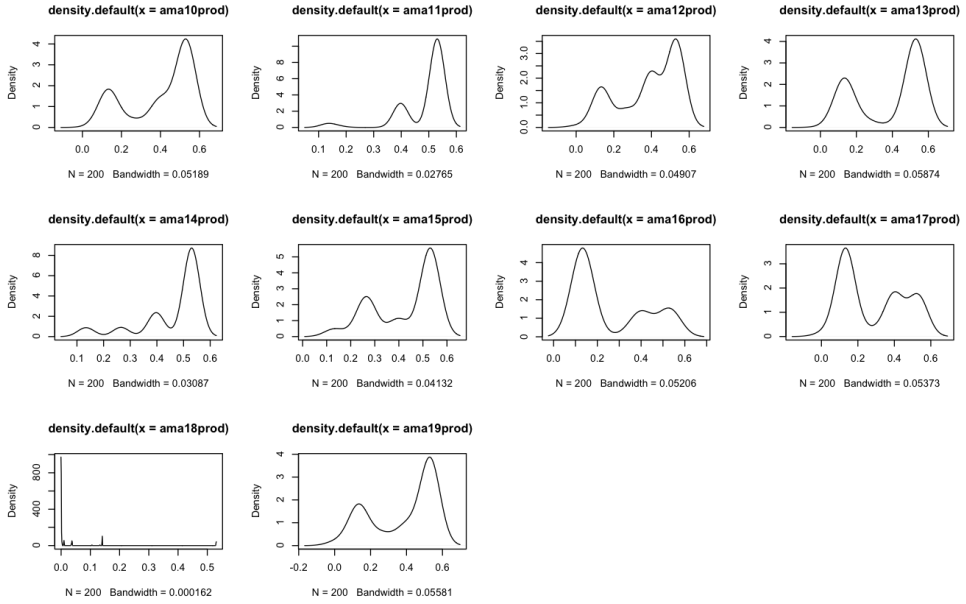


Figure 5: *ama* production densities

2.3 Simulations

Simulations are depicted in Table 1 that furnishes values concerning the *prod_average* and then the relative *pro* and *ama* contributions to the general

Simulation	*prod_average*	*pro_prod_average*	*ama_prod_average*
1^{th}	0.96	1.57	0.35
2^{th}	1.13	1.99	0.27
3^{th}	1.34	2.49	0.20
4^{th}	0.73	1.15	0.30
5^{th}	1.28	2.19	0.37
6^{th}	1.16	2.02	0.30
Average (6^{th} excl.)	1.32	2.28	0.36
Average (6^{th} incl.)	1.10	1.90	0.30

Table 1: Simulations

average within the considered period. First column depicts the simulation whose values refer to, the second the general *prod_average* and the third and fourth the two breeds' contributions. The first simulation is the one we attached the matrix and plots in the above section.

All the simulations have run for 200 ticks apart from the sixth where we wanted to see the effect of leaving the simulation running for 1000 ticks. For this reason, we wanted to separate the average of the two rows on the bottom of the table, to allow us to consider what would happen in the case we wanted to extended simulations for a longer period. However, table shows that there is not a huge difference, hence, time is not a relevant parameter affecting the production function.

Despite the fact that there is a difference in the average including the sixth simulation and the one excluding the simulation, one should pay attention to the smaller number of simulations in the table (five simulations), meaning that the averages might have big variances if another simulation is included in the calculation (since it represents a share of around the 20% of dataset).

To conclude we can state that ABM approach leads to a *prod_average* value of around 1.20 composed by 2.10 of *pro_prod_average* and 0.30 of *ama_prod_average*. That is a relevant statement that will be compared with other two approaches outlining different results concerning the aggregation.

2.4 Aggregation techniques

Here, the most relevant section: the discussion on aggregation emerges and leads us to develop different techniques to decipher. The dynamics of our model that describes the evolution of the production function over time can be read in three different ways. These ways of reading the model

	Plain	*Micro Simulation*	*Agent Based*
diversity factor		√	√
interaction factor			√
emergence factor			√
prod_average	1.28	1.28	1.20
pro_prod_average	1.28	1.71	2.10
ama_prod_average	1.28	0.85	0.30

Table 2: Models for Aggregation

are associated to real approaches through which the aggregation is derived by taking various interpretations.

The following are the three methods for our aggregation, that are presented, discussed and confronted, starting from the easiest to the most complicated one. They are: *Plain Model for Aggregation*, *Micro simulation for Aggregation* and *Agent Based for Aggregation*. The first two approaches are very similar to the aggregation techniques used by public authorities responsible for aggregating, such as the System of National Accounts (SNA) for instance. This is because the calculation is mainly based on sums or weighted averages. The third method, though, is a computational tool that we call "*Agent Based Reasoning Machine*" that is based on a procedure agent-based. It seemed appropriate to compare the techniques used today with an innovative method structured on agent-based technics. Through the specification of heterogeneous characters of agents and different behaviors adopted during the production function, we could map the aggregated results by individual values.

Table 2 summarizes, on the upper part, the key distinguishing features of the three methods, on the lower part instead, provides the average values obtained from the three aggregation techniques.

2.4.1 Plain Model for Aggregation

The first approach is the easiest. From the name "*Plain Model*" we have already an idea concerning the complexity of its structure. It is based on the availability of resources. In our example the *World* was composed by 50 times 50, thus, 2500 patches. Each patch owes a quantity of resources, differently set according to the external map where the maximum amount of resources was establish exogenously.

In this model, we aggregate the resources finding that the global availability of resources was 3200. Since we know that patches are 2500 and agents gain an amount of resources depending on the place where they are located on, we assign to every agent the same value of resources. In this way we do not consider either the heterogeneity among agent themselves or the variables coming from the interactions.

The assigned value is 1.28. Moreover, we wanted to motivate how we derive that value but, since no rule is imposed for defining such value, one could pretend to assign 2 instead of 1.28 due to the fact that some resources are easier to obtain or, in the opposite, to assign 1 because of the difficulty of exploiting some soils.

The main idea here is that resources are assigned by, say, an external entity that takes care of the equalitarianism among the population.

2.4.2 Micro Simulation Model for Aggregation

The second approach is considered in between the other two.

Its name "*Micro Simulation Models*" means that a diversification among micro units is done and assortment of agents are now taken into account. There is thus an improvement with respect to the previous method even if resources availability plays again an important role and the interaction are absent. Here, the *World* resources are still 3200 with an average of 1.28 each patch.

Differently from the previous model, we assign an amount of resources strictly dependent on agent technology of production. For this reason *pro* will have a larger value compared with the *ama* one. Since technology of production of agents *pro* is more efficient, more precisely it is doubled efficient, we assign resources for 1.71 to *pro* and 0.85 to *ama*.

The central concept here is that the external entity assigns resources depending on the diversity of agents. In this example, shares of resources are positively correlated with the quality of agents' technology of production.

2.4.3 ABM for Aggregation

The third approach is the most complete out of three.

Builded up on the resources availability as well as the other two, it considers either the heterogeneity factor among agents or their interaction and emergence components.

Since we have already explained how the values are derived in this model, we only provide them: *pro* collect 2.10, *ama* 0.30 and the average is 1.20.

The reasoning here is not linked on the external entity any more. One should think that agents now, through an agent based model underlining different features among individuals, non equal efficiency of their actions and casualness, are able to gain or lose resources depending on their behaviors and non foreseeable factors.

2.4.4 Comparison

Even if the expected amount of resources exists for all the three methods, only the first two models are able to calculate it. This means that, in the last model, no rule can be followed to obtain the exact share of resources that agents *pro* and *ama* should have. The only way to have an answer is running the simulation and aggregate at the end of the process. The problem is that we will always have different values since simulations present different internal mechanisms with respect to each other, even if parameter values do not vary. In other words, despite the fact that we are aware of the amount of available resources that represent the production of individuals, we do not know how many can really be exploited and what are the real ability of agents to exploit them.

We need to stress once again their differences. The simplicity of calculations of the first method allows us to divide the resources according to the number of available lands that, knowing that will be exploited by individuals, represent the exact production of the same individuals. The problem is that no specification about the impossibility of certain land use has been made. Furthermore, no distinction between the ability of individuals to use land has been taken into care. Finally, no random variable which is not be expected has been taken into consideration.

How can we say that this technique is the best in order to derive aggregate quantities to compare with individual sizes?

Now move the attention to the second method which is not too far away from the first. Here, resources are allocated according to a criterion of productive capacity of the agents. Thus, agents with better capacity utilization will be entitled to more resources than their colleagues who

have less advanced production technology. The flaw in this approach, as in the first, is that it does not reflect the reality: any factor that contributes to the inability to produce or complicate the agents' actions of producing is not considered. According to this model all available resources are fully accessible and fully usable. No agent or external factor interferes in the production of any individual agent.

Therefore, is it a valid procedure, according to the readers, to determine the micro and macro link?

We now analyze the last method, that is based on differences between agents and random factors that affect individual production functions.

Can we thus state that is closer[2] *to reality?*

In our model, a factor of "instability" has been inserted. It refers to the effect of damage to surrounding areas after an agent has the resources extracted in a considered soil. This effect turns out to be one of those factors that can not be calculated and that are not evaluated by the first two methods. It depends on individual behavior from which creates different consequences on the final results. We know that an economic scenario as a whole presents many of these factors, our challenge is to identify and calculate them. Besides this, we know that a further difficulty lies in creating a method of calculation which can be used forever and that is adaptable to different situations. It would therefore be entirely wrong to believe that there is only one calculation tool used in different economic realities. Our research shows that it is wrong to attribute to all economic actors the same behavior. Attributing the *"Representative Agent"*, Kirman (1992) and Schorfheide (2011), as the answer to the heterogeneity of an economy is therefore very simplistic.

Why is the economy so difficult to calculate?

Assuming that

i) an economy has a total amount of resources available for use in the production process,

ii) agents have distinctive characteristics with respect to their production technology and

iii) agents interfere negatively with each other in their production process,

[2]We call it "closer" because unfortunately it remains a simplification of what happens in an economic environment as a whole.

our analysis provides us an important help concerning aggregation. To associate an aggregate result as the first two methods would be too restrictive, to consider an aggregation as the last method would be more complete indeed.

To conclude, we can say that trying to describe with more details an economy, the aggregate production function provides lower results than an approach with fewer details. Whether due to the effect of mutual damaging of agents or the different characteristics of each agent or the inability to fully exploit all available resources, we can not assert it. What we can affirm, however, is that each of these variables plays an important role that, along with random factors, produce the average results on the aggregation transcribed in Table 2 (column on the r.h.s.). The Agent Based approach shows a general average (*prod_average*) lower than the methods where a summation or a weighted average was used (*Plain, Micro Simulation*, i.e. the SNA technics), in addition to a greater differentiation between the two types of individuals (*pro_prod_average, ama_prod_average*).

Finally, can we align ourselves with the principle of Vartia attributing to such decrease in the production function a meaning of negative covariance between micro and macro level?

Our opinion is that the results lead us to believe that there is a connection and, through the agent-based approach, we obtain a negative covariance, in the case where aggregation of production function is considered. For covariance we mean the term coming from Vartia analysis (i.e. the aggregation error).

3 General conclusions

We come to understand how to create an agent-based model to describe an economy using NetLogo. Firstly, we build the model that will allow us to locate the individual functions of production. Secondly, we extract and load the dataset to another program: *R*. *R* is a mechanism able to aggregate what we have obtained with the model in NetLogo. Then, NetLogo and *R* define, at this point, one of the three methods we adopted to aggregate, it covers the agent-based method. Further, two other methods of aggregation are presented, they refer to the classic procedures of aggregations. These procedures are commonly used by public bodies who hold to aggregate.

The central part of our work is the comparison between the classical procedures and the new agent-based technique developed by us. The results of these comparisons are different and, for that reason, we believe we

need to make our agent-based model as clear as possible[3].

The comparison between the classical procedures and agent-based method reveals that Vartia is feasible and that something new has been discovered. Vartia, in his works, believes that the outcome difference between individual and aggregate functions is reflected by a covariance between the variables that determine such functions. According to his studies he states that this covariance, i.e. the error of aggregation, can be, in most cases, eliminated.

Although we support that this covariance term cannot be easily eliminated in a complex system like an economy, we reckon that something is missing. The flaw, in our opinion, is that the error of aggregation is not covered duly: no information on the nature, on the causes and on the consequences are mentioned. Our work is thus aimed at finding information on this covariance, so we can figure out how to treat it. According to the comparison of the agent-based and the classical method of aggregation, we find that the former is smaller than the latter. The error of aggregation is therefore defined by a negative covariance term. In our example of the production function, then, the whole is less than the sum of its parts.

How this is possible is difficult to define, may be due to internal processes of agent-based simulations, or to assumptions set during the construction of our model or to random factors. That is why we want to give a clear explanation of the model, in order to leave to readers a very personal interpretation of the results obtained.

With the invaluable help of readers and people interested in the subject we can proceed to study this issue that, through the agent-based approach, is an interesting starting point for a more precise analysis of the economic scenario surrounding us.

[3]Appendix C provides a clarification of the model through the use of two techniques of description: UML language and ODD standard protocol.

Acknowledgements

This article is a brief discussion of the more complex work which was the subject of my thesis research at the University of Turin, Master's Degree in Economics.

I would like to express my deep gratitude to Professor Pietro Terna and Professor Sergio Margarita, my research supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of this research work. I would also like to thank Dr. Weijie Chen, for his advice and assistance in keeping my progress on schedule. My grateful thanks are also extended to Dr. Ilari Ahola for his help in offering me the resources in running the *R-code*.

I also thank Professor Nigel Gilbert, editor of the Journal of Artificial Societies and Social Simulation (JASSS) and renowned expert on the subject, for his valuable help.

Finally, I wish to thank Dr. Eleonora Stero and my parents for their support and encouragement throughout my study.

Appendices

A Appendix A: NetLogo code

In this part we will provide a detailed description of the code focusing on the economic significance rather than the technique of the language of NetLogo. We divide the code into several parts which will be numbered and will comment at the end of the language code.

1.
```
globals [
  my-date-and-time
  ]

turtles-own [
  production
  consumption
  vision
  vision-points
]

patches-own [
  pproduction
  max-pproduction
]

breed [ pro a-pro ]
breed [ ama a-ama ]
```

Starting from the beginning, we define:

- *my-date-and-time*, a variable that will be used for extracting the dataset subsequently;
- the variables of our agents, *production, consumption, vision* and *vision-points* allowing us to create heterogeneous agents having distinct willing to consume or produce in the case of the former two variables, and capability to find (see) resources in the latter. *production* and *consumption* are also employed as a benchmark evolving over time where we can see the composition of individual functions with respect to the aggregate ones.
- the maximum amount of resources that a patch can have and the main variable: *pproduction* that is the productivity of a piece

of land. Imagine the amount of resources that a certain land can offer.

- the breeds, in other words two types of agent having different features intrinsically. This subdivision enables us to make a greater contribution to the diversity of attitudes and preferences among the agents, in addition to creating a real model that complicates any possible anticipation of the results. Here, the two breeds are *pro* from "professional" and *ama* from "amateur", and indicate the dissimilar capability to exploit the subsoil, hence the resources. It can be referred as different level of technology of production.

2.

```
to setup
  ca
  set my-date-and-time date-and-time
  setup-patches
  setup-pro
  setup-ama
  reset-ticks
end
```

Setup procedure sets out the agents and the patches as follow.

3.

```
to setup-patches
  file-open "map1.txt"
  ;file-open "map2.txt"
  foreach sort patches
  [
    ask ?
    [
      set max-pproduction file-read
      set pproduction max-pproduction
      patch-recolor
    ]
  ]
  file-close
end
```

Patches have a different composition of resources depending on an external file that consists of a matrix of numbers between zero and

four. Zero will coincide with a land devoid of resources, on the other, four will be a land rich of resources that can be exploited by the agents. The possibility of changing the composition of the environment is interesting from the point of evaluation the results. We can thus see how the values vary by changing the soil characteristics and how this affects the individual and aggregate behavior of population.

setup-patches allows the patches of charge in the number of resources contained in the external file and to be colored according to the quantity of resources available. If the patch is green it will have a greater amount of resources, otherwise it will be yellow. The color of the patches is useful for a graphic representation of the model, therefore, has no importance for a mathematical explanation.

4.

```
to setup-pro
  create-pro initial-population
  ask pro [
  set color black
  set shape "dot"
  set size 2
  move-to one-of patches with [
    not any? other turtles-here]
  values
  set vision-points []
  foreach n-values vision [? + 1]
  [
    set vision-points sentence vision-points
    (list (list 0 ?) (list ? 0)
      (list 0 (- ?)) (list (- ?) 0))
  ]
  exploit-p pproduction
  ]
end
```

Agent *pro* are graphically defined by the black and, when they are created, they are placed on a patch where there is no another agent.

vision and *vision-points* allow us to set the capability of catching resources of each agent, that, as mentioned above, will be different for everyone.

Command *exploit* will allow agent to immediately take advantage of

the patch where he is located. Soon we will see how such a procedure works and how the exploitation of resources of an agent can affect other agent's interests.

5.

```
to setup-ama
  create-ama initial-population
  ask ama [
  set color red
  set shape "dot"
  set size 2
  move-to one-of patches with [
    not any? other turtles-here]
  values
  set vision-points []
  foreach n-values vision [? + 1]
  [
    set vision-points sentence vision-points
    (list (list 0 ?) (list ? 0)
      (list 0 (- ?)) (list (- ?) 0))
  ]
  exploit-a_pproduction
  ]
end
```

Agent *ama* are defined with red and presents the same setup procedures as the *pro*.

6.

```
to values
  set production 0
  set consumption 0
  set vision random-in-range 1 10
end
```

values is a common feature of agents, for that reason has a separate code. It can be easily seen that both the above-mentioned setup procedures of the two types of agents are connected to this. The reference code allows us to reset the consumption and production and to develop the ability to see randomly among agents. Thank to the latter variable, we improve the factor of heterogeneity in our model.

7.

```
to exploit-p_pproduction
  ask patch-here [
    set pproduction ( pproduction * pro_p_here ) ]
  ask patches in-radius 1 [
    set pproduction ( pproduction * pro_p_radius1 ) ]
  ask patches in-radius 2 [
    set pproduction ( pproduction * pro_p_radius2 ) ]
  ask patches in-radius 3 [
    set pproduction ( pproduction * pro_p_radius3 ) ]
end

to exploit-a_pproduction
  ask patch-here [
    set pproduction ( pproduction * ama_p_here ) ]
  ask patches in-radius 1 [
    set pproduction ( pproduction * ama_p_radius1 ) ]
  ask patches in-radius 2 [
    set pproduction ( pproduction * ama_p_radius2 ) ]
  ask patches in-radius 3 [
    set pproduction ( pproduction * ama_p_radius3 ) ]
end
```

We have already run into *exploit* at points 4 and 5, where agents were set. We defined this procedure as the land exploitation by each agent. We have also said that this action affects other agent's interest, but how?

It can be seen that the two procedures are identical, the only things that change over the rows are the *pro* or *ama* and *p* or *a* suffixes that are used to differentiate the two kinds of agents. *exploit* command defines the capability of agents to exploit the soil through the different sliders called *p_here, p_radius1, p_radius 2* and *p_radius3*. In economic terms, it has to be associated to the technology of production that differs from one agent to the other. We see that agents in this model do not restrict the exploitation on the patch where they are situated only (*p_here*), but have the power to affect also the patches in radius 1, 2 and 3 respectively. In other words, when an agent goes on a patch having the aim of exploiting resources, he will also cause to all the neighboring patches (in radius 1, 2 and 3) to reduce the amount of resources own. The significance of such command is undoubted: since a agent strategy is strictly dependent on another

agent action, model acquires the glamor to output a non-trivial results, linking any micro behavior with different macro explanations. Running the simulation with the same setting will provide a completely different outcome. That is what in ABMs programming are called *interactions* in a complex system.

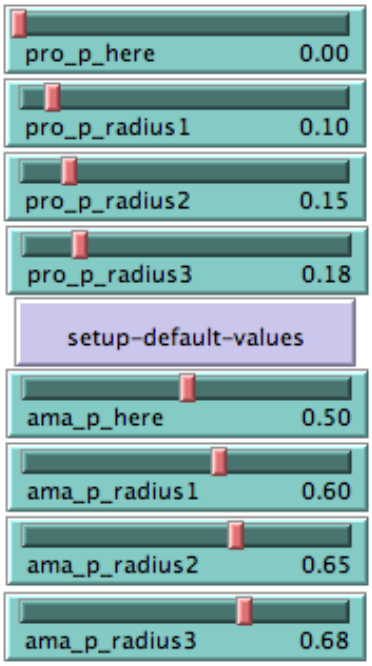


Figure 6: Technology of production parameters

Further, as Figure 6 proves, we arrange the above-mentioned sliders to allow us a variation of the parameters of the production function: by decreasing the values (shifting the slider to the left) we improve the technology of production, hence the soil will be better exploited.[4] Finally, a better technology of production, say of agents *pro*, will cause agents *ama* to reduce the production drastically. For this reason agents *ama* have to move to a new area where *pro* do not stay around.

8.

```
to go
  if not any? turtles [
    stop
```

[4]The values in the sliders work as a percentage: if the values is set at 0, it means that the resources of the soil will be completely exploited, otherwise, if set at 0.50, resources will be exploited only for half of their total amount. It can be seen that *pro* have a way better technology of production with respect to *ama*.

```
  ]

  ask turtles [
    turtle-move
    a-eat
    p-eat
  ]

  ask patches [
    patch-recolor
    patch-growback
  ]

  extract-data

  tick
end
```

We come to discover the main procedure of the model: *go* command. Three different parts can be outlined here: *ask turtles*, *ask patches* and *extract-data*. The remaining parts are needed to *stop* the simulation when there is no turtle any more and, at the end, *tick*, used to count the advancement of the program clock, in other words it counts the time that passes between one period to the following one.

i) *ask turtles* contains *turtle-move* and *eat*. The former allows agents to search for patches having more resources to exploit, the latter instead, is the consumption function and the production function. Soon we will see in why they can be considered in such a way.

iI) *ask patches* implies *patch-recolor* and *patch-growback* commands that will be described at points 11 and 12 respectively. It orders patches to restore their original amount of resources and to color according to the amount of resources own.

iii) *extract-data* houses the long procedure at point 15 that is needed to extract the dataset from NetLogo.

9.

```
to turtle-move
  let move-candidates (
    patch-set patch-here (patches at-points vision-points)
    with [not any? turtles-here])
```

```
  let possible-winners move-candidates with-max [pproduction]
  if any? possible-winners [
    move-to min-one-of possible-winners [distance myself]
  ]
end
```

tutle-move set the capability of agents to find the most interesting patches depending on the amount of resources that can be exploited on them. According with the agents dexterity to see and the quantity of resources, turtles move to the best area where to produce. Best area can be interpreted either as an area where there is not any influence from other agents or where resources are as much as possible.[5] Moreover, we impose that agents cannot move to patches where other agents are already located on.

10.

```
to p-eat
   set production (pproduction)
   set consumption (production * 0.2)
   ask pro [
    exploit-p_pproduction
    ]
 end

to a-eat
   set production (pproduction)
   set consumption (production * 0.2)
   ask ama [
    exploit-a_pproduction
    ]
 end
```

eat commands technically set the production and the consumption function of both the *pro* agents and the *ama* agents. By gaining the same amount of resources that a patch has (*pproduction*), we set the period t production function of an agent; on the other hand, by imposing a lump-sum of 20% of production as a consumption, we assign the consumption function. In this case, a consumption can be

[5]Remember that we have previously uploaded a "Map" where we defined the maximum amount of resources of every patch. Thus, there may be area with low resources exploitation and area with high resources exploitation.

associated either as a cost of producing (fix cost of exploiting resources) or the amount of consumption an household wants to carry, given his income coming from his work. Both the analysis are correct, the important is to distinguish weather we prefer to focus on a "firm analysis" rather than a "consumer analysis". That is possible only because our model is a simplified vision of the reality, hence, both the points of evaluating the matter can be associated with the outcome.

11.

```
to patch-growback
   set pproduction max-pproduction
end
```

patch-growback restores the amount of resources that every patches initially owned. They are set according with the variable *max-pproduction* that we have already met. Usually, it can be interpreted as a regeneration of resources after the agents work. Note that it requires some time, for that reason every *tick* contributes to restore the initial condition. A tick can be seen as the necessary time that enables the resources restoring.

12.

```
to patch-recolor
  if pproduction <=  0  [set pcolor 95]
  if pproduction >   0   and (pproduction <=   .5 ) [
    set pcolor 44]
  if pproduction >   .5  and (pproduction <=   1  ) [
    set pcolor 45]
  if pproduction >  1    and (pproduction <=   1.5) [
    set pcolor 46]
  if pproduction >  1.5  and (pproduction <=   2  ) [
    set pcolor 54]
  if pproduction >  2    and (pproduction <=  2.5 ) [
    set pcolor 55]
  if pproduction >  2.5  and (pproduction <=  3   ) [
    set pcolor 56]
  if pproduction >  3    and (pproduction <=  3.5 ) [
    set pcolor 64]
  if pproduction >  3.5  and (pproduction <=  4.5 ) [
    set pcolor 65]
  if pproduction >  4.5 [set pcolor 66]
```

```
end
```

recolor is a simply way to assign a color depending on the amount of resources own by patches. It is used for the graphical representation. Despite the fact that many variables are set in the model, somehow, it helps us to understand agents' actions, in addition to give a better idea of the *patch-growback* procedure.

13.

```
to-report random-in-range [low high]
  report low + random (high - low + 1)
end
```

It is a technical procedures that can be associated as an utility to derive the model outcomes.

14.

```
to setup-default-values
  set initial-population 10

  set pro_p_here 0
  set pro_p_radius1 .10
  set pro_p_radius2 .15
  set pro_p_radius3 .18

  set ama_p_here .50
  set ama_p_radius1 .60
  set ama_p_radius2 .65
  set ama_p_radius3 .68
end
```

It sets the default values of our variables. Usually default values are needed to define the optimal condition of the model. We can decide to simulate deviations from the optimal condition of the model, for instance by making variation of the initial values of the variables. In that way we can see, through running the simulation, how the outcome changes by changing the parameters of the different variables. A click on the *setup-default-values* button on the NetLogo *interface* will bring back the values to the original, the optimal ones.

15.

```
to extract-data
```

```
file-open (word "pro0production.txt")
ask a-pro 0 [file-print production]
file-close

file-open (word "pro1production.txt")
ask a-pro 1 [file-print production]
file-close

file-open (word "pro2production.txt")
ask a-pro 2 [file-print production]
file-close

file-open (word "pro3production.txt")
ask a-pro 3 [file-print production]
file-close

file-open (word "pro4production.txt")
ask a-pro 4 [file-print production]
file-close

file-open (word "pro5production.txt")
ask a-pro 5 [file-print production]
file-close

file-open (word "pro6production.txt")
ask a-pro 6 [file-print production]
file-close

file-open (word "pro7production.txt")
ask a-pro 7 [file-print production]
file-close

file-open (word "pro8production.txt")
ask a-pro 8 [file-print production]
file-close

file-open (word "pro9production.txt")
ask a-pro 9 [file-print production]
file-close

****    ****    ****    ****    ****
```

```
file-open (word "ama10production.txt")
ask a-ama 10 [file-print production]
file-close

file-open (word "ama11production.txt")
ask a-ama 11 [file-print production]
file-close

file-open (word "ama12production.txt")
ask a-ama 12 [file-print production]
file-close

file-open (word "ama13production.txt")
ask a-ama 13 [file-print production]
file-close

file-open (word "ama14production.txt")
ask a-ama 14 [file-print production]
file-close

file-open (word "ama15production.txt")
ask a-ama 15 [file-print production]
file-close

file-open (word "ama16production.txt")
ask a-ama 16 [file-print production]
file-close

file-open (word "ama17production.txt")
ask a-ama 17 [file-print production]
file-close

file-open (word "ama18production.txt")
ask a-ama 18 [file-print production]
file-close

file-open (word "ama19production.txt")
ask a-ama 19 [file-print production]
file-close
```

```
end
```

The last, is a technical procedure to extract the dataset, creating for each agent a text file of his production over time (firstly for *pro* and subsequently for *ama*). This tool is needed for aggregation with *R*.

B Appendix B: *R* code

This section is for understanding the logic of importing the NetLogo dataset to *R* and giving some explanations concerning the aggregation of Micro data to meaningful Macro level. For simplicity we decided to use *R* for data processing, due to the fact that it has excellent functions to communicate with NetLogo. We will attach the *R* code providing a brief comments on its main parts.

i) We import the dataset concerning the production function referred to the agents *pro*. The following procedures assign to every variable of each agent a different name, in order to simplify the next step of aggregation. For instance, the production of *pro0* is called *pro0prod* by *R*. Same for all the other variables of every agent *pro*.

```
pro0production <- read.table("pro0production.txt")
pro0prod <- pro0production[,1]

pro1production <- read.table("pro1production.txt")
pro1prod <- pro1production[,1]

pro2production <- read.table("pro2production.txt")
pro2prod <- pro2production[,1]

pro3production <- read.table("pro3production.txt")
pro3prod <- pro3production[,1]

pro4production <- read.table("pro4production.txt")
pro4prod <- pro4production[,1]

pro5production <- read.table("pro5production.txt")
pro5prod <- pro5production[,1]

pro6production <- read.table("pro6production.txt")
pro6prod <- pro6production[,1]

pro7production <- read.table("pro7production.txt")
pro7prod <- pro7production[,1]

pro8production <- read.table("pro8production.txt")
pro8prod <- pro8production[,1]
```

```
pro9production <- read.table("pro9production.txt")
pro9prod <- pro9production[,1]
```

ii) The same procedures as the previous ones, with the only difference that now they assign the *R* names for agents *ama* dataset.

```
ama10production <- read.table("ama10production.txt")
ama10prod <- ama10production[,1]

ama11production <- read.table("ama11production.txt")
ama11prod <- ama11production[,1]

ama12production <- read.table("ama12production.txt")
ama12prod <- ama12production[,1]

ama13production <- read.table("ama13production.txt")
ama13prod <- ama13production[,1]

ama14production <- read.table("ama14production.txt")
ama14prod <- ama14production[,1]

ama15production <- read.table("ama15production.txt")
ama15prod <- ama15production[,1]

ama16production <- read.table("ama16production.txt")
ama16prod <- ama16production[,1]

ama17production <- read.table("ama17production.txt")
ama17prod <- ama17production[,1]

ama18production <- read.table("ama18production.txt")
ama18prod <- ama18production[,1]

ama19production <- read.table("ama19production.txt")
ama19prod <- ama19production[,1]
```

iii) We plot the individual densities of every single agent's production function (Figure 4 and Figure 5). Soon we will comment on those

outcomes.

```
par(mfrow=c(3,4))

plot(density(pro0prod))
plot(density(pro1prod))
plot(density(pro2prod))
plot(density(pro3prod))
plot(density(pro4prod))

plot(density(pro5prod))
plot(density(pro6prod))
plot(density(pro7prod))
plot(density(pro8prod))
plot(density(pro9prod))

par(mfrow=c(3,4))

plot(density(ama10prod))
plot(density(ama11prod))
plot(density(ama12prod))
plot(density(ama13prod))
plot(density(ama14prod))

plot(density(ama15prod))
plot(density(ama16prod))
plot(density(ama17prod))
plot(density(ama18prod))
plot(density(ama19prod))
```

iv) We calculate the average of *pro*'s production functions, we ask *R* to show its time-series matrix and we plot the regression graph.

```
pro_prod_average <- (pro0prod + pro1prod + pro2prod
+ pro3prod + pro4prod + pro5prod + pro6prod + pro7prod
+ pro8prod + pro9prod)/10
pro_prod_average

plot(pro_prod_average)
```

v) Same as the last procedure even if now it refers to the *ama*.

```
ama_prod_average <- (ama10prod + ama11prod
+ ama12prod + ama13prod + ama14prod + ama15prod
+ ama16prod + ama17prod + ama18prod
+ ama19prod)/10

ama_prod_average

plot(ama_prod_average)
```

vi) We calculate the general average of the production functions of all agents called *prod_average*. Then, we ask for the time-series matrix, its regression graph (Figure 2) and density plot (Figure 3).

```
prod_average <- (pro0prod + ama10prod + pro1prod + ama11prod
+ pro2prod + ama12prod + pro3prod + ama13prod + pro4prod
+ ama14prod + pro5prod + ama15prod + pro6prod + ama16prod
+ pro7prod + ama17prod + pro8prod + ama18prod + pro9prod
+ ama19prod)/20

prod_average

plot(prod_average)
plot(density(prod_average))
```

vii) We aggregate the individual production functions in order to find the macro level of production. In others words, by summing up the individual production functions, we find the general product (hence, GDP). Then, we ask for the time-series matrix, the regression graph and the density plot.

```
prod <- (pro0prod + ama10prod + pro1prod + ama11prod
+ pro2prod + pro4prod + pro6prod + pro8prod prod
```

```
+ ama12prod + ama14prod + ama16prod + ama18prod
+ pro3prod + ama13prod + pro5prod + ama15prod
+ pro7prod + ama17prod + pro9prod + ama19prod)

prod

plot(prod)
plot(density(prod))
```

viii) We conclude by asking the value regarding the average of the production functions of *pro*, *ama* and the general *prod_average* within the considered period (200 ticks) respectively. This will be the tool to compare different simulations (in the following section named Simulations). The aim is underlining the variances between interactions, basing the models with the same parameter variables. We will soon discover that results will be always different, despite the fact the parameters do not vary between models (Table 1 will show such differences). Notice that, in addition to the parameter values among simulations, we also need to consider the agents' features that do always change as soon as we press on *setup* button.

```
mean(pro_prod_average)
mean(ama_prod_average)
mean(prod_average)
```

C Appendix C: Representations of Simulation for Aggregated Consumption and Production function

We have two main problems concerning the descriptions of Agent Based Model simulations:

i) there is no standard way for describing them and

ii) ABM are often described verbally without a clear indication of the equations, rules, and schedules that are used in the model.

For such two reasons, in order to better explain the model, we both apply the UML language and the ODD standard protocol.

Despite the fact that we believe the UML approach is awkward especially for non computer sciences experts, we would apply either the UML or the ODD standard protocol. Since ODD standard protocol has a non quantitative nature and most of the explanations are simply verbal expositions, we presume it would be, in a sense, easier for readers.

The following are the main differences between the two overtures. We first start with the UML language, providing a general overview, depicting all its diagrams and describing them. Then, we conclude by outlining the main steps of the ODD standard protocol after having explained the general intuition of each stage.

C.1 Unified Modeling Language (UML)

Unified Modeling Language (UML), Bersini (2012), is a standardized general-purpose modeling language in the field of object-oriented software engineering; It can be seen as a language rather than a methodology, it is based on four main diagram:

i) Class diagram illustrates the classes and their relationships, such as association, composition and inheritance;

ii) Sequence diagram represents how objects interact and exchange messages over time. It allows developers to trace the program while it executes and to follow the way objects interact in memory;

iii) State-Transition diagram is able to follows the state-transitions of one complicated class of agent over its lifetime (for instance the dying transition as soon as the agent's internal energy goes below a minimum threshold). It always starts from an initial state (the birth of the object), and ends at a final state (the death of the object);

iv) Activity diagram is best understood as a throwback to the more traditional procedural types of diagram (called "flow charts"), it helps programmers when dealing with the more procedural instructions flow related parts of their code. This diagram is often seen as an alternative to the state diagram. The advantage of this diagram over the state diagram lies in its ability to cover the behaviors of collaborating elements (in some such cases it can also become an alternative to the sequence diagram).

Both the exact order of diagrams to be realized and the correct drawing of each diagram are not part of the language but rather the result of correct practice acquired through experience.

The benefit brought by UML becomes more marked as models grow in complexity (reflected by the number of classes). Note, however, that UML has been advocated in other scientific fields such as biology, chemistry or physics. Hence, it might allow us to extend our analysis on the aggregation issue.

C.1.1 Class diagram

"...Establishing classes and their interrelationships is more of an art than a science...requires a lot of training, development experience, the knowledge of some good recipes..." Bersini (2012)

The Class diagram is considered the most convenient way for classes and their relationships to be illustrated. Despite that, it is not suitable for establishing what the necessary classes are and how their interrelations should be.

Figure 7 depicts a Class diagram of our simulation model for aggregated production function. The classes of our model are: *World* that includes all the actors of the simulation, *Patch* that is the site where agents and resources (*pproduction*) are, *Resource* is own by every patch in a different quantity and is looked for by agents and *Agent* that, by moving and looking for resources, can produce and consume part of their production. A further subdivision of agents can be done by differentiating their capability to produce, hence their technology of production: *pro* are the most efficient breed and *ama* have exactly the same chances to find resources but a lower productivity.

It can be easily seen that plenty of associations appear. For instance, when we see a "*1 - 1*" association linking *Agent* and *Patch*, it means that

an agent can be placed in one and only one patch at one time. Such association is needed for agents to interrogate if the patch is free, in order to decide wether to move there or, in the case it is not available because another agent is already located there, to move to somewhere else place.

As can be notice, an association can be either unidirectional (*World-Agent*) or bidirectional (*pro-Resource*). In the first case, it means that the message between the two classes flows in one direction only, as the arrow indicates. A label is commonly associated with the message representing a name of the attribute referring to the class.

An association "*1 - 8*" of Patch is a directional auto-association meaning that each site is directionally associated to its eight neighbors. The eight sites around a patch can be all taken over in the only case where no agent is already located there.

The diamond of an association is called "composition" and indicates a stronger form of association. A composition between two classes states that the disappearance of the contained object leads to the elimination of the object as well. For instance, if *World* disappears, all its objects will do the same automatically.

A "*1 - 0..** " relationship between two classes means that any object of the first class is associated or composed with an arbitrary number of objects of the other class. Dashed line represents a dependency and it is a weaker form of association, where the arrow shows the dependency from the dependent object to the object it depends on. For instance, *pro* has a dependency on *Resource* but, since a single agent is not necessarily always associated with the same resource, we need a dashed line to depict a weaker relationship.

"Inheritance" is another type of association within the class diagram. It is drawn from *Agent* class to the two subclass of *pro* and *ama*. Inheritance allows for subclasses to have specific attributes, methods or the redefinition of some methods already present in the superclass.

The abstracts contained in the class, such as *black color* or *more efficient* of *pro* class, refer to different features between the two subclasses of agents, while the abstracts of agents are common to all. An example is the different technology to produce distinguishing the *pro* from the *ama*, while the way to move is the same for both.

Behavior class

A different kind of class diagram is represented by *Behavior classes* that is used to distinguish static objects from processes. The latter describes how objects change over time. Same as before, they are mostly adopted for explaining complex situation where an individual behavior becomes hard

to keep trace. Figure 8 provides a meaningful explanation concerning agent *pro1*'s behavior and the process that generates interactions in the system.

A *bridge design patterns* could be employed to attach the behavior class in Figure 8 to the class diagram in Figure 7. In such a case the behavior linked with the inheritance depicts the behavior of all agents providing a further information of the simulation model mechanisms.

Finally, another tool helps us to better understand: the *decorator design patterns* needed to assess the separation between of fundamental characteristics of a class diagram from a set of added functionalities (decorators) varying from one object to another. It can be seen as a representation of different subclasses and their main features in a more flexible way than simply providing a list of subclasses.

C.1.2 Sequence diagram

Despite behavior class represents a good explanation of the objects change over time, a more complete picture is given by the *Sequence diagram*. Interactions between objects and emerging messages coming from such relationships are the base of the diagram structure. Thank to this character, the representation allows developers to trace the program while it executes and to follow every single interaction of the observed objects.

Figure 9 draws the interaction of an agent starting from *world* where he moves to the *patch x* looking for a free place to stay. The query is thus *see(resources)* since the goal is to find both a free place (better is if free of agent also in its neighboring) and a site rich of resources. Next step is *do(patch y)* once resources have been exploited. Sequence diagram is the best tool to trace the sequential steps and the execution logical flow of the program. Its goal is to maintain simplicity and readability within the several steps portraying the system.

Unfortunately, the diagram does not provide an optimal solution for the object interactions but does evaluate the potential flows of execution and the successive responsibilities of every single object only. As can be seen, rectangles are placed on the life line indicating the method duration. They outline all the exchanged messages between objects.

Commands *loop* and *alt* enable objects to act in a sequential loop establishing a mechanism of sequential steps starting with the former command and ending with the latter. No detail concerning the nature of the agents selections is included, nor wether they are taken in a random or an ordered way.

The query *agent patch y = null* is used to define if the place is free, and in the case it does, an agent will move to the free site *set agent(here)*

(patch y), causing an updating of the site status that now turns to be busy. Otherwise, *else* is run.

C.1.3 State diagram

State diagram or *state-transition diagram* is designed in Figure 10. It refers to a single agent *pro1* and its aim is to follow the state-transition of one complicated class of agent over its lifetime. In the figure the agent deals with four actions only, called states, where transitions are traced between each other.

Diagram starts with the initial state, usually associated to the birth of an object and graphically shown with a black disk. It ends at a final state with a black disk inside a white disk. In our case, since we do not have any constraint, agents do not disappear from the *world*. For that reason, the final state is missing.

Transitions are based on *guards*, that is the UML name for transition conditions. A guard defines the motivation why an agent should move from the state of *Moving* to *Exploiting*. In the example, the reason is the presence of resources that have been caught off by agent while moving. The agent will go back to move as soon as resources on site have been exploited: *pproduction=beta(max-pproduction)*.

State design pattern can be also associated with the above-mentioned class diagram, furnishing a complete framework: each state is linked with every possible transition. Logically, that association is needed in a case of a complicated system having dependence between objects and their state. Modularization of behavioral blocks is thus required to have a meaningful explanation of such complexity.

C.1.4 Activity diagram

Activity diagram, frequently associated with his forefather *procedural diagram* called *flow charts*, consists a relevant aid when dealing with the more procedural instructions flows related parts of their code. Considered as an alternative with respect to the state diagram, it hosts the personal programmer perception of the model. It shows the dynamic of the objects through a succession of activities rather than a succession of states. The advantage is its capability to cover the behaviors of collaborating elements (same as the sequence diagram).

Figure 11 shows the partition of activities between two objects: *Agent (pro1)* and *Resources (pproduction)*, where the terms in parenthesis are the name used in the code. The two black bars mark the beginning and the end of an activity. For instance, while *pro1* exploits, he decreases the amount

of resources on site, increase both his consumption and production at the same time.

To conclude, UML language can be used in a multitude of ways and it represents one of the major tools allowing a better understanding of a simulation model, especially if complexity plays an important role and leads to difficulties coming up from the intrinsic mechanism of objects' interactions. The language is formal but, as we have seen, use-methodology is unrestrained and, for that reason, programmer has the chance to customize it and to use in the best way he would to.

The followings are the 5 Figures of the above-explained diagrams.

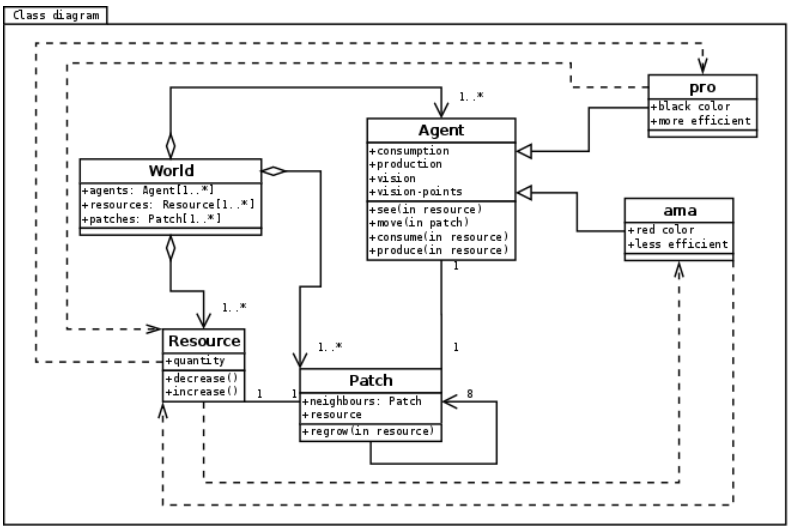


Figure 7: Class diagram

C.2 ODD Standard Protocol

"...there is no standard protocol for describing such simulation models, which can make them difficult to understand and to duplicate." Grimm *et al.* (2006)

The apt quotation above can be found in the abstract of the Grimm and Railsback paper of 2005, where they propose a standard protocol for describing ABMs. Their aim is to provide a simpler and quicker way to understand simulation models since no rule are set for a standard to follow in the representations of ABMs.

We give a general overview of the ODD protocol first in order to be better able go through each component later. Then, we apply the standard protocol to our model representation.

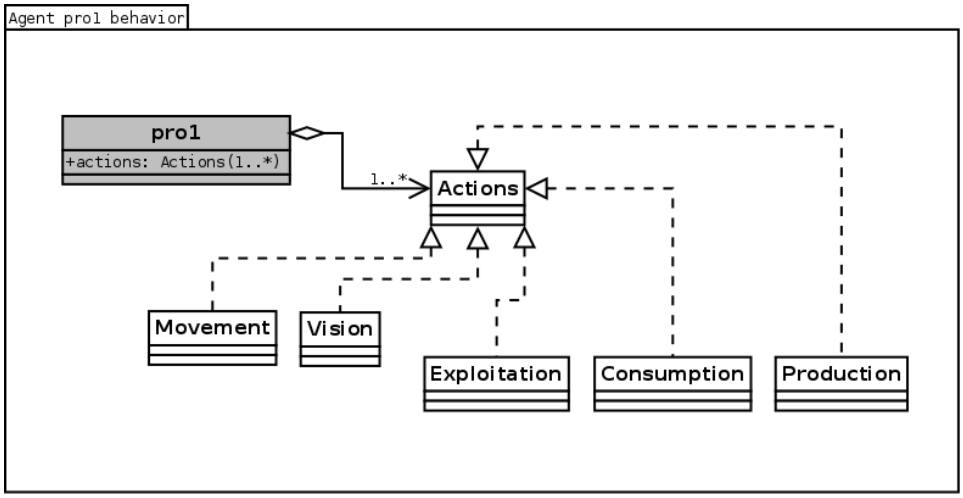


Figure 8: Behavior class

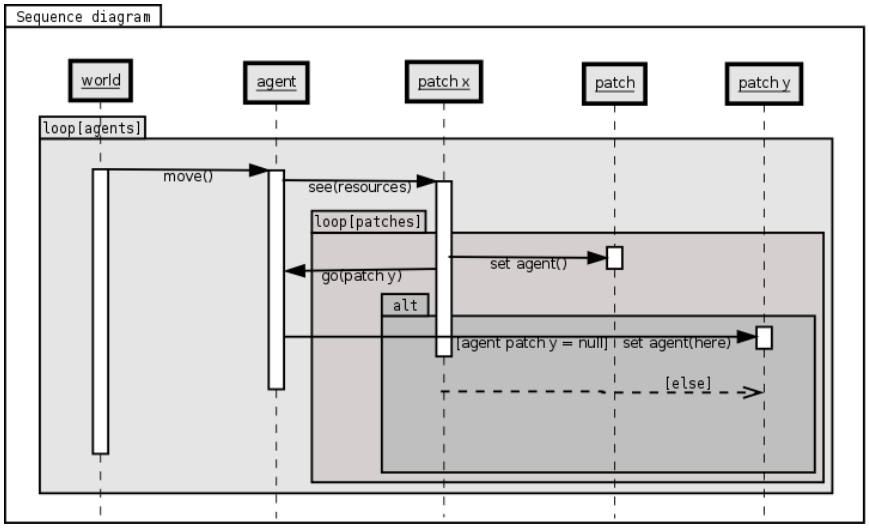


Figure 9: Sequence diagram

ODD overview

Odd protocol combines two elements:

i) a general structure for describing ABM, thereby making a model's description independent on its specific structure, purpose and form of implementation and

ii) separation of the verbal considerations from a mathematical description of the equations, rules, and schedules that constitute the model;

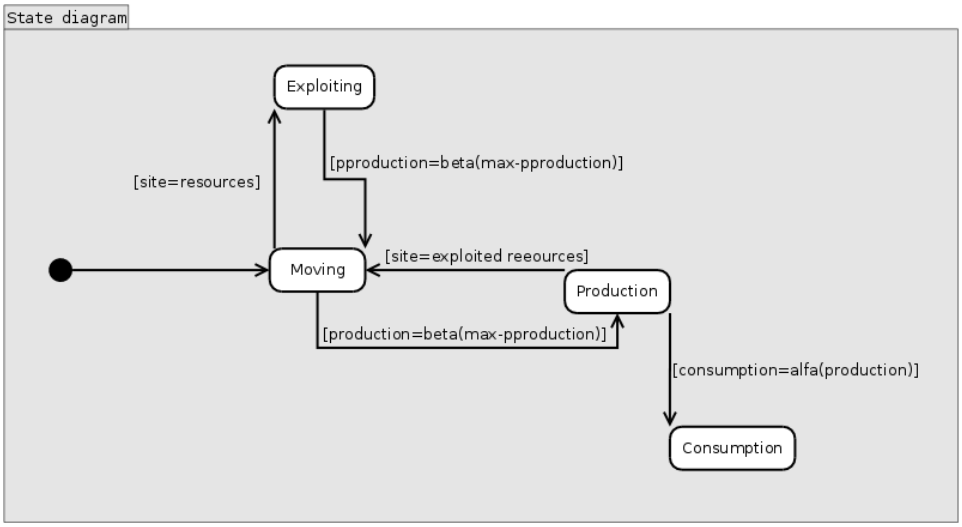


Figure 10: State diagram

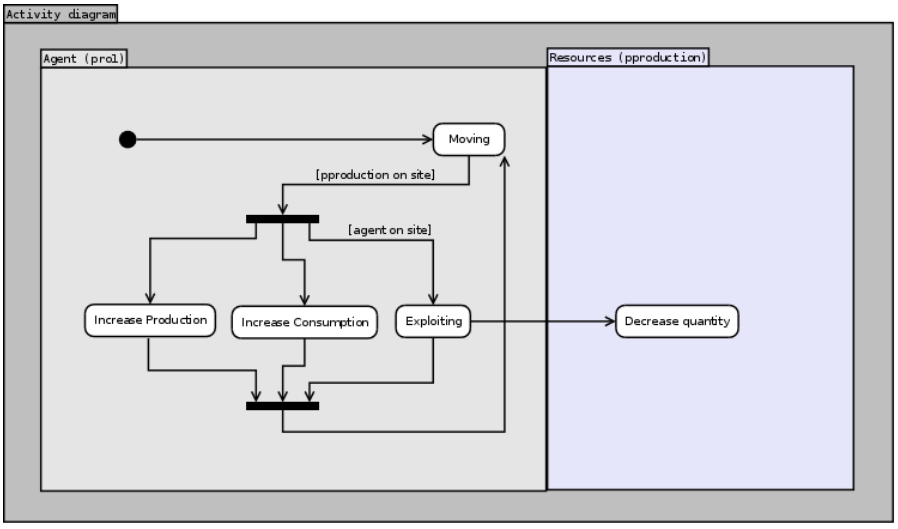


Figure 11: Activity diagram

Allowing both readers and writers to understand the main features of the model.

Further, a standard protocol would make reading and understanding the model easier because readers would be guided by their expectations.

The first idea of a standard protocol, that is PSPC + 3 protocol is referred to the initials of first four elements of the protocol (purpose, structure, process, concepts) and "+3" referred to the remaining three elements.

I would use for our model instead, the "revised PSPC + 3 protocol", called "ODD", Grimm *et al.* (2010), which stands for the three blocks of elements "Overview", "Design concepts", and "Details" since the names of some elements have been changed.

Figure 12 shows that this protocol consists of three blocks (Overview, Design concepts, and Details), which are subdivided into seven elements: Purpose, State variables and scales, Process overview and scheduling, Design concepts, Initialization, Input, and Submodels:

i) The Overview consists of three elements (Purpose, State variables and scales and Process overview and scheduling), which provide an overview of the overall purpose and structure of the model. Readers very quickly can get an idea of the model's focus, resolution and complexity;

ii) The Design concepts does not describe the model itself, but rather describes the general concepts underlying the design of the model and

iii) The Details, includes three elements (Initialization, Input and Submodels) that present the details that were omitted in the overview. In particular, the submodels, implementing the model's processes, are described in detail. All information required to completely re-implement the model and run the baseline simulations should be provided here. If space in a journal article is too limited, on-line appendices or separate publications of the model's details should be provided.

The logic behind the ODD sequence is: context and general information is provided first (Overview), followed by more strategic considerations (Design concepts), and finally more technical details (Details)

The benefits of the protocol become obvious in the test applications. The most important benefits are:

- The model description becomes easier to write. It is no longer necessary to waste a lot of time thinking about how to structure the text, because the protocol make those decisions for the authors so that they simply could follow the template;

- The model description becomes more complete because the protocol reminds the authors of important details that they might have otherwise forgotten to include in the documentation;

- The model description becomes easier to understand. In one case, for example, the protocol can suggest a context for describing a concept that is confusing to the reviewers of the original paper and

- The protocol is not only useful for individual-based or agent-based models, but for bottom-up simulation models in general, for example grid-based models.

A standard protocol for describing a simulation for aggregated consumption and production function

C.2.1 Purpose

The purpose of the model is to create a dataset of aggregated production functions of a set of agents that interact each other. There are two different breeds of agents: *pro* and *ama* distinguishing graphically with black, in the first case, and red, in the second. Production function is determined by the capability to see that each agent has, in addition to the technology of production. The latter is more efficient for agents *pro* with respect to *ama*. The two functions are bolstered by the agent actions of exploiting the resources own by the ground: the patches. Resources are set initially with an external map making out the composition of the field. After the agent exploitation, they regrow according to such a map. The unforeseeable determination of agents productions and consumption is due to the capability they have to affect the neighboring patches while they are exploiting resources from the ground. An agent will go exploiting a site as far as possible from another agent. The reason is to avoid any loss coming from the closeness. Many variations to parameter variables can be done allowing the analysis of system reactions.

C.2.2 Entities, state variables and scales

The structure of the model system is listed as following:

- *Agents:* There are 10 agents *pro* and 10 agents *ama* that are located randomly moving to look for resources. They cannot neither die nor reproduce, they do not have any age. The only counters they have are consumption and production. They present different capability to see, randomly set, and different technology of production. The *pro* production is more efficient than the *ama* one.

- *Landscape:* The *World* is composed by 50 times 50 squares: the patches. They have different color according with the amount of resources they own. Color blue means no resources, starting from yellow, that is the minimum amount of resources, they turn to different color until green that marks the maximum amount of resources. Such a quantity of resources is set by an external map providing the maximum amount of resources every patch can have. The map can be

changed to evaluate the different results that can be obtained. Resources are looked for agents that exploit them as much they can. Resources can regrow once a tick passes according with the external map.

- *Variables and parameters values:* Simulations can be tested by modifying the sliders on the NetLogo interface. The following variables and parameters values can be changed: the number of the population and the technology of production of both agents *pro* and *ama* regarding either the site where they are located on (*p_here*) or from all the patches set in radius 1 (*p_radius1*) to the ones in radius 3. The higher the technology of production the more agents gain from the exploitation and the more they want to stay far from each other while locating in a site. Further, the more the number of the population the less resources per capita, hence the less the production and consumption.

C.2.3 Process overview and scheduling

At this stage, a meaningful *flow charts* can provide the best scheduling and overview of the process. I would refer, thus, to the *Activity diagram* of Figure 11 covered in the previous section.

C.2.4 Design concepts

- *Emergence:* Resources of patches, Production and Consumption counter of agents depend on the degree of exploitation made by individuals. Such a composition is also linked either with the agent's capability to see or the external map with the imposed maximum amount of resources. The variation of the parameters values by the sliders can also affect the emergence.

- *Adaptation:* The improvement of the agent's capability to see can mark a positive betterment of consumption and production. Same for any rise in technology of production.

- *Fitness:* In the model the concept of fitness is represented with the amount of production each agent is able to make. Consumption is linked. Concerning the patches, it is the degree of resources own. It is also represented with the color a patch has. The more green the more the resources. The more yellow the less the resources. Blu means no resources at all.

- *Prediction:* The only prediction agents can have is the following: by imitating the action of other agents an individual is 100% sure about the reduction of his own production. He has better avoid reproducing someone else's actions.

- *Sensing:* Individuals need to consider the composition of the soil looking for the maximum amount of resources as possible and the presence of other agents in neighboring areas trying to skip any site already occupied.

- *Interaction:* In the model, an interaction can be identified into the consequence of the exploitation of resources from the soil. The action of exploiting a patch where an agent is locates, is also spread on the surrounding patches (until the patches in radius 3). In other words, while agent is located on *patch14* for instance, he exploits the resources either of *patch14* or all surrounding patches from radius 1 to radius 3.

- *Stochasticity:* Stochastic elements are the degree of agent's capability to see resources, agent technologies of production, the different maps of the resources composition of soil and the agent's movements.

- *Collectives:* Agents are not grouped into collective. The only possible distinction, that is far from grouping, can be associated with the two breeds: *ama* and *pro* present personal disjointed features.

- *Observation:* The aim of the simulation model is to collect a dataset of the agent's production over their lifetime, in order to make a comparison between the individual values and the aggregated ones.

C.2.5 Initialization

Environment is created by importing an external map of values identifying the maximum amount of resources of every patch. According to these values the color of each patch is defined.

The initial number of agents is set at 10 *pro* and 10 *ama*. The former are black and the latter red. They have different technologies of production set at the beginning according with "default values". As default values, the *pro* are able to exploit 100%, 90%, 85% and 82% the patch where they are located on, the patch in radius 1, in radius 2 and in radius 3 respectively. The *ama* instead are able to exploit 50%, 40%, 35% and 32% the patch where they are located on, the patch in radius 1, in radius 2 and in radius 3 respectively.

Initialization is always the same except for the stochastic elements described previously, such as the degree of agent's capability to see resources, the different maps of the resource compositions of soil and the agent's movements.

C.2.6 Input data

The model does not use input data to represent time-varying processes.

C.2.7 Submodels

Since a detailed description of the model code has already been provided within the preceding section (named NetLogo code), we do not reproduce the previous step, rather we invite readers to go through the section 6 again, in the case something is not completely clear.

	original ODD protocol (Grimm et al., 2006)	updated ODD protocol
Overview	***1*** *Purpose* ***2*** *State variables and scales* ***3*** *Process overview and scheduling*	***1*** *Purpose* ***2*** *Entities, state variables and scales* ***3*** *Process overview and scheduling*
Design concepts	***4*** *Design concepts* *Emergence* *Adaptation* *Fitness* *Prediction* *Sensing* *Interaction* *Stochasticity* *Collectives* *Observation*	***4*** *Design concepts* *Basic principles* *Emergence* *Adaptation* *Objectives* *Learning* *Prediction* *Sensing* *Interaction* *Stochasticity* *Collectives* *Observation*
Details	***5*** *Initialization* ***6*** *Input* ***7*** *Submodels*	***5*** *Initialization* ***6*** *Input data* ***7*** *Submodels*

Figure 12: ODD protocol

References

Bersini, H. (2012). Uml for abm. *Journal of Artificial Societies and Social Simulation* **15**(1), 9.

Burbidge, J. & Cuff, K. (2005). Capital tax competition and returns to scale. *Regional Science and Urban Economics* **35**(4), 353–373.

Epstein, J. & Axtell, R. (1996). *Growing artificial societies: social science from the bottom up*. MIT press.

Felsen, M. & Wilensky, U. (2007). Netlogo urban suite-economic disparity model.

Friedman, M. (1957). The permanent income hypothesis.

Godley, W. (1999). Money and credit in a keynesian model of income determination. *Cambridge Journal of Economics* **23**(4), 393–411.

Grimm, V., Berger, U., Bastiansen, F., Eliassen, S., Ginot, V., Giske, J., Goss-Custard, J., Grand, T., Heinz, S., Huse, G. *et al.* (2006). A standard protocol for describing individual-based and agent-based models. *Ecological modelling* **198**(1), 115–126.

Grimm, V., Berger, U., DeAngelis, D., Polhill, J., Giske, J. & Railsback, S. (2010). The odd protocol: A review and first update. *Ecological Modelling* .

Kirman, A. (1992). Whom or what does the representative individual represent? *The Journal of Economic Perspectives* **6**(2), 117–136.

Schorfheide, F. (2011). Estimation and evaluation of dsge models: progress and challenges. Tech. rep., National Bureau of Economic Research.

Squazzoni, F. (2012). *Agent-Based Computational Sociology*. Wiley.

Suoperä, A. & Vartia, Y. (2011). Analysis and synthesis of wage determination in heterogeneous cross-sections. *HECER Discussion Paper* **1**(331).

Törnqvist, L., Vartia, P. & Vartia, Y. (1985). How should relative changes be measured? *The American Statistician* **39**(1), 43–46.

Vartia, Y. (1976). *Relative changes and index numbers*, vol. 1. Research Institute of the Finnish Economy Helsinki.

Vartia, Y. (2008a). Integration of micro and macro explanations. *HECER Discussion Papers* **1**(239).

Vartia, Y. (2008b). On the aggregation of quadratic micro equations. *HECER Discussion Papers* **1**(248).

Vartia, Y. (2009). Whole and its parts: Micro foundations of macro behaviour. *HECER Discussion Papers* **1**(257).

Wilensky, U. (1999). Netlogo. *Center for Connected Learning and Computer-Based Modeling* .

DEPARTMENT OF ECONOMICS AND STATISTICS
UNIVERSITY OF TORINO
Corso Unione Sovietica 218 bis - 10134 Torino (ITALY)
Web page: http://esomas.econ.unito.it/