Università di Torino

FACOLTÀ DI ECONOMIA

Corso di Laurea Magistrale in Economics

SIMULATION MODELS AND AGGREGATION

Tesi di Laurea in Simulation Models for Economics

Relatore: TERNA, P.
Correlatore: MARGARITA, S.

Presentata da:
LUCA BARONE

Anno Accademico
2011-12

Simulation Models and Aggregation

Luca Barone

Master Degree in Economics

Department of Economics and Statistics

University of Turin

July 5, 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 "UML language" and "ODD standard protocol", allowing us to clarify the internal processes of our model.

Il collegamento tra livello micro e livello macro è sempre stato difficile da tracciare, anche quando le variabili presentano forti caratteristiche omogenee. Cosa succeda quando componenti eterogenee e fattori casuali interagiscano è ancor più difficile da definire.

Adottando un approccio agent-based abbiamo trovato un risultato che non rispecchia i metodi classici di quantificazione di un'economia. Questo può essere interpretato come un campanello di allarme segnalante una descrizione non corretta dello scenario economico che vogliamo spiegare. Illustriamo l'efficacia della "macchina di ragionamento agent-based" e ne deriviamo un modello da confrontare con i metodi classici di aggregazione.

Una descrizione più comprensibile del modello è attribuita al linguaggio "UML" e al protocollo standard "ODD", che ci permette di rendere più chiari i processi interni del nostro modello.

"Maybe there is in human nature a deep-seated perverse pleasure in adopting and defending a wholly counterintuitive doctrine that leaves the uninitiated peasant wondering what planet he or she is on".[1]

[1] Robert Solow

To:

E.S.
W.C.
J.N.
C.M.
A.C.
M.B.
P.B.
L.M.
T.M.
A.M.
V.M.
L.D.
A.B.
P.S.
C.B.
B.M.

for your ivaluable support done with loving care.

Contents

Preface

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 (2008) 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), which in some cases may be canceled, causing a perfect equality between the two levels under study. Because 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.

In a nutshell, we do not know the nature of the error of aggregation, and especially its sign. 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.

Our tool was the production function, using a simple model we created. This model was analyzed through three different approaches: *Plain, Micro Simulation* and *Agent Based*. The first two methods 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. The last hand 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 between 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 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 *UML language* 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 sec-

ond 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.

1 Introduction

This work focuses on a discussion about the issue of aggregation. The starting point is the link between aggregate result and its components.

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

We refer to the theory of Vartia (2008), professor at the University of Helsinki. He discusses the issue by applying different approaches to the study, at first very theoretical and more practical then. An important practical case is attributed to the example of the consumption function and the marginal propensity to consume.

Although we retrace the key concepts of his theory, we will develop our analysis on a different theme: the aggregation is analyzed mainly under the point of view of the production function, the aggregate domestic product.

Our tool to aggregate is *R*. It receives as input a dataset of individual production functions and, through a quantitative process, outputs the aggregate result.

The distinctive character of our work is attributed to the composition of individual production functions. They are derived from an agent based model (ABM) invented by us.

Normally, they 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 a function of individual production.

We compare three different methods, starting from the simplest to the most complete, and we derive that changes on aggregate results are substantial.

Giving a brief explanation about the reason for these differences and how they originated, we conclude by providing our interpretation on findings.

In order to allow readers an interpretation too, we explain the details of the model using both the UML language and the ODD standard protocol, in addition to providing a thorough explanation on ABMs and what their pros and cons are.

Listed below are the following chapters and the subdivision of the arguments. We will not describe in detail the arguments of each chapter as an introduction is already present at the beginning of every chapter. We will provide only a general description.

Chapter 2 is a review concerning some of the most relevant concepts of economic theories. For this reason the title is "Pillars". Such a brush-up is needed because every topic of the chapter is used in our study.

This review is critical to quickly understand our analysis. From the model used to understand the functioning of NetLogo, to the model allowing us to derive the dataset for aggregation, we will cover each topic that is part of this chapter. For this reason, it seemed appropriate to start with a review of the tools we will use later.

We derive the formulas of production and consumption functions and we discuss about related concepts. Figures and graphs are provided to enable a quick review.

Chapter 3 is crucial to better understand the issue of aggregation. It first develops a distinction between individual and aggregate result, then shows how to make a connection between these two quantities. The nomenclature and technique used are very meaningful and straightforward. The third part (3.3) of the chapter applies the Vartia's theory of aggregation to a consumption function and derives significant results.

In other words, the chapter starts by giving a general understanding on the theme of aggregation and on the development of studies in recent years, it proceeds to explain how Vartia motivated and interpreted the connection between micro and macro through his technique and concludes by applying the analysis to a case concerning the consumption function, more precisely, the marginal propensity to consume.

Chapter 4 develops a detailed description on agent-based models (ABMs). First on what they are, second their history and evolution, third we illustrate types and goals that allow a classification, then we see how you can apply economic theories to agent-based methods and finally we conclude by motivating our choice: why we chose an agent-based method.

The chapter ends by introducing NetLogo. This is the program that we use for our agent-based analysis. This chapter, along with Chapter 4, will give a detailed explanation of NetLogo.

Chapter 5 is devoted to an easy example of ABM that represents a consumption function in a closed economy with a limited number of agents.

The purpose of this chapter is to understand the functioning of NetLogo in the first place, and see how you can apply economic theories to an agent-

based method.

After introducing the operation of NetLogo, we analyze the programming language and create a model. This model is used to assess the consumption function.

Then, we proceed with a simulation. The latter is used to compare outputs. The normal procedure is to run the model with different parameters so you can see how the final result changes with the change of internal variables. A commentary is provided for each parameter change. Further, a chart, graphically illustrates these distortions from the optimal result.

Chapter 6 is the main part of our work. It is divided into three major phases.

The first is the illustration of the model that has ben created with NetLogo, by which we construct our dataset of individual production functions. In the first stage we have an overview of the model and an explanation of the code in detail.

The second step is the aggregation with *R*. This is explained by direct analysis of the programming language of *R*. The results that has been found by *R* are attached, together with plots, and are discussed. Important are the simulations that allow us to calculate an average result rather than just one. This is necessary because the agent-based method emits different results each time.

A paragraph (6.3) to delineate the transition between the first and the second phase is entered. This helps us to understand how the extraction of the dataset by NetLogo and the inclusion of it in R have been managed.

The third phase, the most important, concerns the comparison of three different methods of aggregation.

The first way is the one discussed so far in this chapter, called "ABM for Aggregation". It is the most complete and detailed out of the three.

The second and third are two of the classic ways of aggregation used by the public authorities to aggregate, as the System of National Accounts (SNA). The most obvious of these is an average result of the production function that is used to aggregate, the name is "Plain Model for Aggregation". The other one is called "Micro Simulation Model for Aggregation" and is an average weighted according to a criterion for distinguishing agents.

All three methods are widely explained and discussed. Our interpretation allows to connect the theory of Vartia discussed in Chapter 3. A further comment is also present in the general conclusions of Chapter 8.

Chapter 7 develops a very detailed explanation of the model of the previous chapter. Since the model of Chapter 6 is the starting point of our analysis on the aggregation, there is a need to understand how it works in detail.

ABMs can be described by two mechanisms: UML language 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.

Chapter 8 is characterized by general conclusions. They outline a connection between our analysis to the theory of Vartia (2008) of Chapter 3. The findings relate mainly to Chapter 6, which, thanks to the support of

the remaining chapters, provide a clear analysis concerning the ABM aggregation.

In addition, an advance on future work is provided, along with an invitation for readers to continue and develop our analysis.

Appendix A reviews the key concepts of the theory of Vartia coming from Chapter 3 and describes the properties of the mathematical tool of "inner product".

Appendix B outlines the Basic Lemma of Aggregation (BLA), which is the weapon applied by Vartia. Thanks to BLA, individual functions turn to aggregate ones. The BLA has also be proven mathematically.

Appendix C lists the programming code in NetLogo referred to the example model of Chapter 6.

Appendix D lists the NetLogo programming code of the model reported in Chapter 7. That is the code of the tool that allows us to create the dataset of individual production functions.

Appendix E produces the *R* code that is used to aggregate individual production functions obtained with the model of Chapter 7.

2 Pillars of Economic Theory

The aim of first section is to provide a review concerning some of the main concepts in economics. We call them "Pillars" due to the fact that they are some of the most relevant starting points of many economic theories. We will soon come to discover why we need such a brush-up and how we can link them with our analysis.

The first wipe-over goes to the Keynesian Consumption Function, we reel off all its components by deriving the shape of the function firstly and hooking it up to the autonomous and induced factors subsequently.

Secondly, we outline the Average Propensity to Consume (APC), providing a numerical example of a decreasing APC.

Thirdly, we define what Budget Constraint means and how it obstacles the consumer consumption.

Then, we differentiate the concept of Capital from Income, in order to make clear the distinction between the two meaning.

Finally, we provide the definition of Production Function and we give explanations concerning its Capital and Labor factors. Further, we go through the concept of isoquants, returns to scale and we conclude by listing the major measures of productivity.

We also attach same of the main graphical proves of the above-mentioned definitions and derivations in order to better understand the reasoning.

2.1 Keynesian Consumption Function

The consumption function is a mathematical formula laid out by the famous economist John Maynard Keynes in his book *The General Theory of Employment, Interest, and Money*[10]. Such a function is designed to show the relationship between real disposable income and consumer spending. The real disposable income is what Keynes considered the most important determinant of short-term demand in an economy. The consumption function can also be seen as the relationship between consumer spending and the various factors determining it. At the household or family level, these factors may include income, wealth, expectations about the level and riskiness of future income or wealth, interest rates, age, education, and family size. The consumption function is also influenced by the consumer's preferences (e.g., patience, or the willingness to delay gratification) and by the consumer's attitude toward risk. The characteristics of consumption functions are important for many questions in both macroeconomics and microeconomics.

The standard version of the consumption function emerges from the "life-cycle" theory of consumption behavior articulated by economist Franco Modigliani. The life-cycle[14] theory assumes that household members choose their current expenditures optimally, taking account of their spending needs and future income over the remainder of their lifetimes. Modern versions of this model incorporate borrowing limits, income or employment uncertainty, and uncertainty about other important factors such as life expectancy. However, economist Milton Friedman advocated a simplified version of this model, known as the "permanent income hypothesis"[4], which abstracts from retirement saving decisions.

The formula, in the simple case, is shown as an affine combination of *autonomous consumption* that is not influenced by current income and *induced*

consumption that is influenced by the economy's income level

$$C = C_0 + c_1(YD) \tag{2.1.1}$$

where

- C is the consumer expenditure
- C_0 is a constant that does not depend on Income

  (i.e. the autonomous consumption)

  it is the level of consumption that would take place even if income was zero. If an individual's income fell to zero, some of his existing spending could be sustained by using saving: it is called dis-saving
- $C_0 > 0$
- c_1 is the marginal propensity to consume

  (i.e. the induced consumption)

  it is the change in consumption divided by the change in income. Simply, it is the percentage of each additional euro earned that will be spent
- $0 < c_1 < 1$
- YD is the current disposable income
- YD represents income - taxes $= y - T$
- $T = T_0 + t_1(y)$
- T_0 is lump sum taxes
- $0 < t_1 < 1$

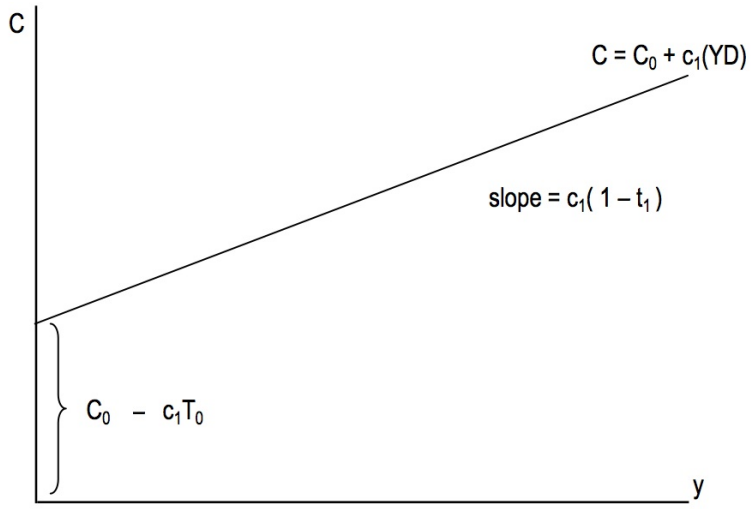

Figure 1: Keynesian Consumption Function

thus, we can rewrite

$$C = C_0 + c_1(y - T) \tag{2.1.2}$$
$$= C_0 + c_1(y - T_0 - t_1 y) \tag{2.1.3}$$
$$= C_0 + c_1 y - c_1 T_0 - c_1 t_1 y \tag{2.1.4}$$
$$= C_0 - c_1 T_0 + c_1 y - c_1 t_1 y \tag{2.1.5}$$
$$= C_0 - c_1 T_0 + c_1(1 - t_1)y \tag{2.1.6}$$

and it can be described graphically as in Figure 1.

The latter gives us a significant intuition: if income $= 0$, then $C > 0$ because people dis-save. We see this by the intersection of the consumption function with the vertical axis. At zero income, aggregate consumption is still positive. The only way to still have positive consumption when income is zero is to dis-save.

The fact that the consumption function is flatter than the 45° means that as income rises, consumption rises, but not as rapidly as income. Out

of every euro of income received, a portion is consumed and a portion is saved. So as income rises, savings, which is first negative (dissavings) and then is positive, rise. We call the amount of each additional dollar of income that is consumed the marginal propensity of consumption: MPC. Since the amount consumed out of each additional dollar of income is smaller than a dollar, then we know that $MPC < 1$.

2.1.1 The Average Propensity to Consume

The Average Propensity to Consume (APC) can be computed by dividing both sides of Keynesian consumption function by disposable income:

$$APC = \frac{C}{YD} = \frac{C_0}{YD} + c_1 \frac{(YD)}{YD} \tag{2.1.7}$$

$$= \frac{C_0}{YD} + MPC. \tag{2.1.8}$$

The consumption function based on Keynes' absolute income hypothesis has the following important properties:

- The MPC is between 0 and 1 (not inclusive), i.e. $0 < MPC < 1$;
- The MPC is less than the APC, because of the existence of some autonomous consumption. In diagrammatic terms, because the consumption function does not go through the origin;
- The APC falls as YD rises. Consider Table 1, it can clearly be seen that the lower the APC the greater the level of disposable income. In other words, the absolute income hypothesis implies that households spend a smaller proportion of their income as it increases. (Correspondingly, the fraction saved - the APS or Saving Ratio - must increase with income.) The declining APC results form the second property listed above, i.e MPC for the above example is 0.75 which is less than the APC

YD	C	APC	Saving (YD-C)
500	875	1.75	-375
1000	1250	1.25	-250
1500	1625	1.08	-125
2000	2000	1	0
2500	2375	0.95	125
3000	2750	0.92	250
3500	3125	0.89	375
4000	3500	0.88	500

Table 1: Numerical example of decreasing APC

in the table. Thus, given an increase in disposable income, households spend an additional 0.75 of this on extra consumption. But this is less than the current fraction of total income being spent (the APC). Hence the APC will decline as income rises, as Figure 2 shows graphically and (9) satisfies:

$$APC_1 = \frac{C_1}{y_1} > APC_2 = \frac{C_2}{y_2} > APC_3 = \frac{C_3}{y_3}. \tag{2.1.9}$$

2.2 Budget constraint

A budget constraint describes all the combinations of goods and services that a consumer can afford.

It is important to understand that the budget constraint is an accounting identity, not a behavioral relationship. An agent may well determine his behavior by considering his budget constraint, but his budget constraint is a given element of the problem he faces.

For simplicity, economists tend to consider only two goods (Figure 3), consequently the bundle of goods is just composed by two types of goods

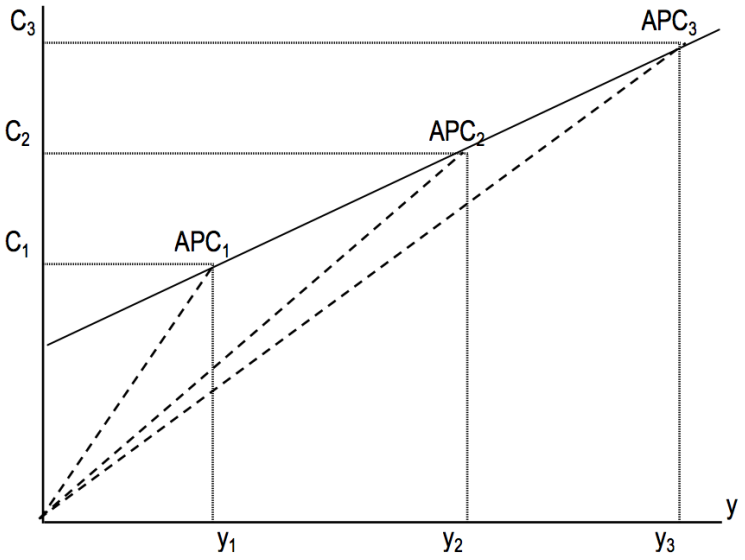

Figure 2: Average Propensity to Consume

(in our example good A and good B).

In general, budget constraints can be written as

$$P_A \cdot Q_A + P_B \cdot Q_B \leq I \tag{2.2.1}$$

where, the price of the good on the x-axis (good A) times the quantity of the good on the x-axis plus the price of the good on the y-axis (good B) times the quantity of the good on the y-axis has to equal the total amount of disposable income (the budget).

It also states that the slope of the budget constraint is the negative of the price of the good on the x-axis divided by the price of the good on the y-axis. (This is a little weird since slope is usually defined as change in y divided by change in x).

$$Slope = -\frac{P_A}{P_B} \tag{2.2.2}$$

Intuitively, the slope of the budget constraint represents how many of the good on the y-axis the consumer must give up in order to be able to afford

one more of the good on the x-axis.
The example in Figure 3 is built on the following assumptions:

- $I = 100$ euros
- $P_A = 50$ euros
- (initial) $P_B = 5$ euros
- (increased) $P_B = 10$ euros

The graph provides a clear picture of all the opportunities an agent has to choose:

i) spending all his income in good A, hence getting 2 units, and none of good B;

ii) using all income in 20 units of good B and zero of good A or

iii) all the combination of the two goods that lie between i) and ii).

Note that, being the solid line in the graph our budget constraints, all the combinations of goods that are above this line are not affordable, so our income will only allow us to purchase a combination of goods that are below this line (we do not consume all the income, hence the l.h.s. of (1) is $< I$) or on the surface of the line (we consume all the income, hence the l.h.s. of (1) is $= I$).

Shall we now ask the following question: *What if the price of the good on the y-axis happen to change?*

In the event that the price of good B would increase (e.g. from 5 euros to 10 euros), while maintaining the same income, say 100 euros, we could no longer afford 20 units of good B, but only 10. No change will be on the amount of the good A, whose price remains unchanged.

Figure 3 shows that there is a shift of the curve that intersect the y-axis to the origin, that confirms that we can now only have 10 units of the

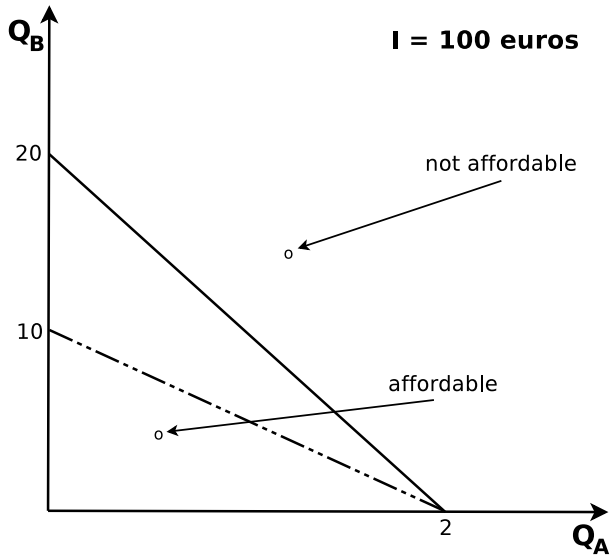

Figure 3: Budget constraint

good B. As a conclusion, the dashed line becomes the new budget constrain bound.

2.2.1 Capital vs. Income

People are often confused by terms income and capital. Part of the problem is that such terms have several meanings and it is, therefore, important to understand the context in which they are used.

The main difference between capital and income is that capital is a fund, instead income is a flow. A fund of property existing at an instant of time is called capital. A flow of services provided by that capital by the payment of money from it or any other benefit in relation to such fund, through a period of time, is called income. It is now more meaningful to state that capital is wealth, while income is the service of wealth.

Income can also be associated, on the one hand, with the amount of money that the majority of us receive for doing work. On the other hand,

it is also an operating cost to the production cycle.

Capital is what is invested into the production cycle to get it started. Income to a business is synonymous with profit after all operating costs are paid, and capital investments are compensated.

Income has a bit of flexibility to it and so it is able to fluctuate. Capital is more rigid and is a fixed cost. Technically, income could go away and the business would still operate (basically a non-profit model) but if capital goes away the business will shut down.

An interesting approach to better outline the differences between capital and income is considering Nature. Schumacher in his book[17] provided a very good example of "natural capital": he talked about fossil fuels. Schumacher states:

"No one, I am sure, will deny that we are treating them as income items although they are undeniably capital items. If we treated them as capital items, we should be concerned with conservation: we should do everything in our power to try and minimize their current rate of use; we might be saying, for instance, that the money obtained from the realization of these assets - these irreplaceable assets - must be placed into a special fund to be devoted exclusively to the evolution of production methods and patterns of living which do not depend on fossil fuels at all or depend on them only to a very slight extent."

2.3 Production Function

The production function (Figure 4) is a very important tool that can be explained both by microeconomics and macroeconomics theory. For that reason, it is a function that specifies the output either of a firm, an industrial sector or an entire economy for all different combination of inputs.

In other words, since we know that a given output can be produced with

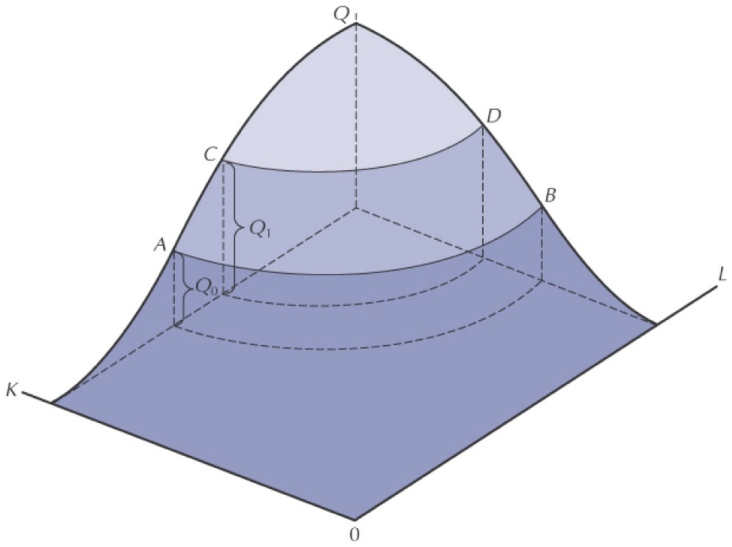

Figure 4: Production Function

many different combinations of factors of production (land, labor, capital and organization) that are inputs, the output, thus, is a function of inputs. This functional relationship that can be found between physical inputs and physical output is called production function. It does not represent the result of economic choices, but rather it is externally given entity and can influence any economic decision-making. It is described as a technological relationship based on the current state on engineering knowledge. As a consequence, It is important to keep in mind that the production function describes technology rather than economic behavior.

A firm may maximize its profits given its production function, but generally firms take the production function as a given element of that problem. (e.g. in specialized long-run models, the firm may choose its capital investments to choose among production technologies.)

In abstract term, it is expressed in a functional form:

$$Q = f(X_1, X_2, X_3, ..., X_n) \tag{2.3.1}$$

where Q is the quantity of output and $(X_1, X_2, X_3, ..., X_n)$ are the quantities of inputs. Usually, for convenience, production function is expressed as follows:

$$Q = f(K, L), \tag{2.3.2}$$

where K represents the Capital and L the Labor.

i) At each level of K, production increases with L

ii) At each level of L, production increases with K

iii) Different combinations of inputs (K, L) are unable to obtain the same production

2.3.1 Isoquants

An isoquant is a curve that shows all the technologically efficient combinations of two resources, such as labour and capital, that produce a certain amount of output. Iso, from the Greek word meaning 'equal', and quant from 'quantity', means 'equal quantity'. Along a particular isoquant, the amount of output produced remains constant, rather, the combination of resources varies (Figure 5). To produce a particular level of output, the firm can employ resource combinations ranging from capital-intensive combinations (much capital and little labour) to labour-intensive combinations (much labour and little capital).

The short run is defined as a period of time where at least one factor of production is assumed to be in fixed supply i.e. it cannot be changed. We normally assume that the quantity of capital inputs is fixed and that production can be altered by suppliers through changing the demand for variable inputs such as labour, hence,

$$Q_{short-run} = f(K_{fix}, L). \tag{2.3.3}$$

In the long run, all factors of production are variable.

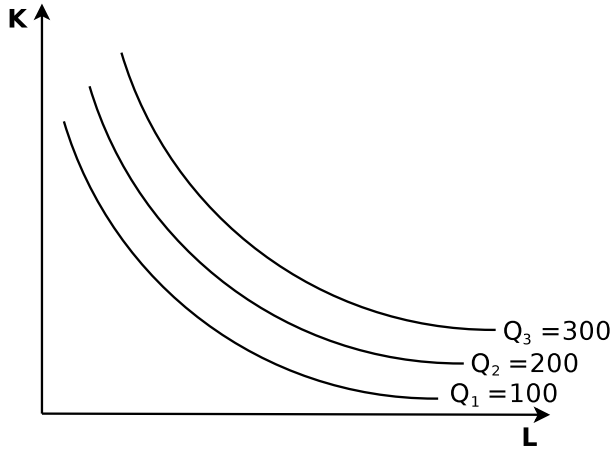

Figure 5: Isoquant Map

2.3.2 Returns to scale

Returns to scale[9] are technical properties of the production function. If we increase the quantity of all factors employed by the same (proportional) amount, output will increase. The question of interest is whether the resulting output will increase by the same proportion, more than proportionally, or less than proportionally. In other words, when we double all inputs, does output double, more than double or less than double? These three basic outcomes can be identified respectively as *increasing returns to scale* (doubling inputs more than doubles output), *constant returns to scale* (doubling inputs doubles output) and *decreasing returns to scale* (doubling inputs less than doubles output).

Marshall 1890 used the concept of returns to scale to capture the idea that firms may alternatively face "economies of scale" (i.e. advantages to size) or "diseconomies of scale" (i.e. disadvantages to size).

After Marshall, there has been a long debate to better define the concept of returns to scale in a technological sense by Wicksell (1900, 1901, 1902),

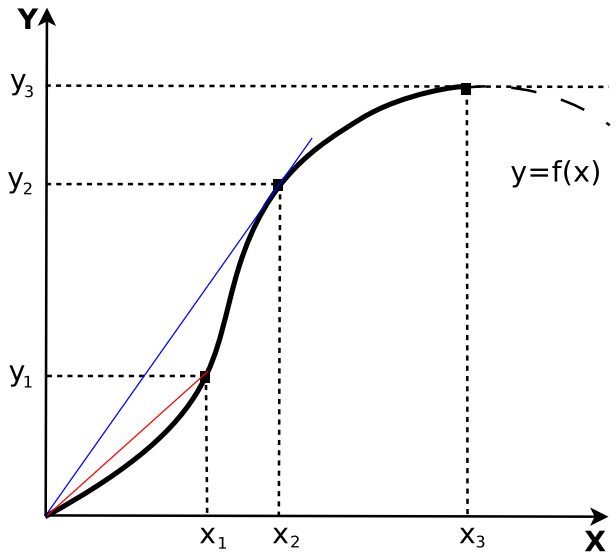

Figure 6: Returns to scale

Wicksteed (1910), Sraffa (1926), Robinson (1932) and Hicks (1932, 1936); Although it was a common proposition that a single production function would have different returns to scale at different levels of output.

The simplest case one-input/one-output production function $y = f(x)$ can be shown graphically in Figure 6, where, as all our inputs (in this case, the only input, x) increase, output (y) increases, but at different rates. At low levels of output (around y_1), the production function $y = f(x)$ is convex, thus it exhibits increasing returns to scale (doubling inputs more than doubles output). At high levels of output (around y_3), the production function $y = f(x)$ is concave, thus it exhibits decreasing returns to scale (doubling inputs less than doubles output).

2.3.3 Measures of productivity

- *Total product* is simply the total output that is generated from the factors of production employed by a business. In most industries, it is

straightforward used to measure the volume of production from labour and capital inputs that are emplyed. But in many service or knowledge-based industries, where much of the output is "intangible" or perhaps weightless, it is harder to measure productivity.

- *Average product* is the total output divided by the number of units of the variable factor of production employed (e.g. output per worker employed or output per unit of capital employed).
- *Marginal product* is the change in total product when an additional unit of the variable factor of production is employed. For example marginal product would measure the change in output that comes from increasing the employment of labour by one person or by adding one more machine to the production process in the short run.

3 Theory of Vartia: Integration of Micro and Macro

3.1 The general motivation

Many mathematicians, statisticians and economists have tried to make sense in the aggregation problem during roughly 40 years, Klein (1946), Leontief (1947), Nataf (1948), Gorman (1959), Theil (1954), Arrow (1963), Green (1964) and other almost Nobel-level classic experts of aggregation in economics have studied the matter, so that inputs allocated to these questions are considerable. After 40 years of intensive research on the area, writers gave up and withdrew from it roughly 30 years ago. Evidently, the problem was considered as unsolvable or too difficult to solve. Most of the "solutions" were negative ones: aggregation is normally impossible. It is possible only under conditions, which are usually described as very restrictive.

Vartia, University of Helsinki, has been studying the aggregation problem for many years and he built up his own theory concerning the topic. He believes that economists went on assuming representative agents and behaviors as if that would have been the outcome and suggestion of the researchers. Their suggestions were exactly the opposite, thus, there must have remained something important undiscovered in the issue. Problems of similar complexity have been considered in statistical physics but without heterogeneity of behavioral functions. The following subsections "Whole and its Parts: Micro and Macro link" and "Aggregation of Marginal Propensity to Consume" will show how micro can be decomposed and aggregated in order to provide two meaningful views towards micro explanations of macro results.

We will only focus on the Vartia's general theory that will be soon applied in the Agent Based Simulation in order to examine the outcomes. For the sake of convenience I will use the Vartia's notations to explain his theory.

Vartia's interpretation of aggregation lead to an integration of parts and whole simply by mathematical approach given results that have been certified e.g. by simulations and they have worked in all empirical problems they have been applied to.

The scheme (1) states that the actual topic of research area is the macro level (M), chosen some micro level (m) and (μ) denotes the mathematical-logical methods in their integration

$$m \xrightarrow{\mu} M. \tag{3.1.1}$$

Ketonen (professor of philosophy), expressed his definition about the "whole" and "parts": *the whole is equal to or else than the sum of its parts*; Suppose 11g of hydrogen gas is burned in 89g of oxygen, we get 100g of water. In that case we are able to say that the whole (water), is the sum (in respect to its weight) of its parts but, in term of the properties, it is else than the sum of its parts (hydrogen and oxygen). The whole, in the example, has essentially different properties from its parts.

3.2 Whole and its Parts: Micro and Macro link

[23]

3.2.1 Introduction

Micro foundations are now associated with the Analysis-operator and the Macro behavior with the Synthesis-operator. On the one hand, the former gives the detailed idea of all the individual micro outputs in terms of their behavioral function and inputs; On the other hand the latter describes the macro output (here the weighted mean of micro outputs) in terms of the same arguments. They both allow us to *"see the forest from the trees"*.

As a final result the two operators are decomposed and studied in the details in order to extract the meaningful explanations of the agents behavior both at micro and at macro level.

3.2.2 Parts vs. Whole

We start by defining the micro units, the "parts" (e.g. households) as a subset $A(i)$ of the macro set A, the "whole"

$$A = \bigcup_{i=1}^{n} A(i) = \sum_{i=1}^{n} A(i) \tag{3.2.1}$$

and the output, considered as a reaction

$$y_i = y(A(i)), \tag{3.2.2}$$

then, the sum constitutes the total output

$$Y = y(A(1)) + ... + y(A(n)). \tag{3.2.3}$$

Inserting the relative frequencies $q(A(i))$ as a *counters*

$$Q = q(A(1)) + ... + q(A(n)), \tag{3.2.4}$$

we can now refer to Y, for instance, as the total value of consumption and Q as the amount of households. Thus, considering means, we can get the per capita values

$$\overline{y} = \frac{Y}{Q}. \tag{3.2.5}$$

If all subsets are singletons or sets with one unit only,

$$A(i) = \{a(i)\} \tag{3.2.6}$$

then,

$$A = \{a(1), ..., a(n)\} \tag{3.2.7}$$

and

$$q(a(i)) = 1. \tag{3.2.8}$$

That is the ideal case where all the units are measures separately and equally weighted (equally weighted mean or unweighed average). Moreover, the (1) is considered as a sampling scheme where *proportional allocation* is observed.

3.2.3 Analysis-operator

As we define previously, the Analysis-operator or A-operator includes all the details concerning the micro structure, the parts, and, by fixing the behavioral function f_i with $(i = 1, 2..., n)$ of the n households we are able to define the whole micro system, where $y(a(i))$ is the output and $x(a(i))$ is the input for $a(i)$

$$y(a(i)) = f_i x(a(i)) \tag{3.2.9}$$

At this stage we can rewrite all the micro equations in term of vectors

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} f_1(x_1) \\ \vdots \\ f_n(x_n) \end{bmatrix}$$

and the A-operator

$$y = A \begin{bmatrix} f_1, x_1 \\ \vdots \\ f_n, x_n \end{bmatrix} = A(f, x).$$

The latter separates the function symbols from their arguments agent by agent and output their outcomes $f_i(x_i) = f_i x_i$, that is a product. Thank to the A-operator, the micro system can be expressed shortly and elegantly with

$$y = A(f, x), \tag{3.2.10}$$

called the Basic Micro View "Eight Symbol Formula". It shows how the outputs are explained in terms of its constituent parts: all the behaviors f_i and the inputs x_i, that are the *Analysis* of the micro system.

3.2.4 Synthesis-operator

We come to discover the Synthesis-operator or S-operator, that is also called Aggregator or Integrator and it practically is the weighted mean of y that we define initially. First, we have to define some new notation concerning the mean

$$\langle y \rangle = \overline{y} = \frac{1}{n} \sum_{i=1}^{n} y(a(i)), \tag{3.2.11}$$

now, if we substitute y with (10)

$$\langle y \rangle = \langle A(f, x) \rangle, \tag{3.2.12}$$

finally, we end up with the S-operator

$$\langle S(f, x) \rangle = \langle A(f, x) \rangle. \tag{3.2.13}$$

To conclude, the Synthesis-operator has the capability to map the inputs to the average output as follow

$$\langle S(f, x) \rangle = \langle y \rangle. \tag{3.2.14}$$

3.2.5 Linearity of A and S-operators

In this part, we do not consider the multitude of details that Vartia arose in his work, but will proceed to highlight the main results that allow us to draw important conclusions about the role of the aggregation and the relative accuracy concerned with the statement that the macro level is the synthesis of the micro level.

The theory of Vartia assumes the linearity of either the Analysis or the Synthesis operator, in that way it is easier to decompose the *Actual Behavior* $AB(x)$, that is, using the nomenclature of Vartia, the definition of S-operator.

As you can see below, it is possible to separate from the notion of $AB(x)$ its *Heterogeneous Effect* $HE(x)$

$$S(f,x)\rangle = AB(x) = CB(x) + HE(x) \tag{3.2.15}$$

where the *Common Behavior* $CB(x)$ can be further decoposed into *Representative Behavior* $RB(x)$ and *Non-linearity Effects* $NLE(x)$

$$CB(x) = RB(x) + NLE(x). \tag{3.2.16}$$

Note that $NLE(x)$ is vanishing in linear average behavior only.

Further, consider that the last expression looks nice but is very complicated instead: $CB(x)$ depends on all the behaviors, despite the fact that it seems simple.

The result of all the mathematical passages derived up to now is the great discovery that

$$NLE(x) + HE(x) \approx constant. \tag{3.2.17}$$

This can be verified by the Central Limit Theorem CLT, more precisely by the *Cramer's converse of CLT*:

If independent random variables produce for some n a sum (normalized or not), which is exactly normally distributed, then all these variables must themselves be normally distributed

This is a proven fact, although for large n the sum gets more and more normally distributed irrespective of the distribution of individual variables.

Then, by considering small changes h, $x \to x + h$, the actual differential changes of the macro output satisfy

$$\Delta AB(x, h) \approx \Delta CB(x, h) \tag{3.2.18}$$

3.2.6 Conclusions

Analysis and Synthesis operators have been defined and they enabled us to delineate the behavioral components of the macro function. Through the application of the central limit theorem we have discovered that the macro objects seem to behave as if they were made of equal parts that behave in the same way and the driving forces were the means of the input variables.

In the end, the heterogeneity of behaviors and all the differences of the agent-inputs at micro level might cancel away at macro level in the major part of the normal conditions.

3.3 Aggregation of Marginal Propensity to Consume

[22]

3.3.1 Introduction

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

Vartia has being looked for that answer for a long time and he interestingly finds that, when behavioral equations are expressed in terms of changes instead of levels, the connection between micro and macro formulations becomes a real possibility.

He showed that applying this rule to the Keynesian consumption function, the marginal propensity to consume can be aggregated. The effective macro level marginal propensity of consume MC is considered to be an affine combination of the uniform absolute changes and uniform proportional changes of income. The uniform absolute changes of income can be denoted with MC^{UA} and the uniform proportional changes of income with MC^{UP}. Concerning their weights, they strictly depend on the way and how much the income distribution is changing.

The problem of aggregation is illustrated by the Keynesian consumption function where the individual non-linear consumption function (2) can differ in arbitrary ways, even if there might be a million of them. We consider a set of agents (a), where their consumption $c(a)$ is a function f^a of income (x)

$$c(a) = f^a(x(a)), \tag{3.3.1}$$

the Marginal Propensity of Consume (MPC) is represented with

$$\overline{m}(a) \tag{3.3.2}$$

and the changes in income for households (a) with

$$\Delta x(a). \tag{3.3.3}$$

They map the micro level information to the macro level.

Vartia shows that, thank to the Basic Lemma of Aggregation (BLA), we can aggregate the MC directly by expressing the inner product

$$\langle \overline{m}\Delta(x) \rangle \tag{3.3.4}$$

in terms of meaningful macro level. The BLA is able to decompose the average of a product into the product of averages plus a covariance term that can be small depending on its weighting.

In the following sections we will consider:

i) the uniform absolute changes of income $\Delta x^{UA}(a)$ and we will prove that MC^{UA} is an evenly weighted average of the micro mpc's;

ii) the uniform proportional changes of income $\Delta x^{UP}(a)$ and we will prove that MC^{UP} is an income weighted average of the micro mpc's;

iii) the macro marginal coefficient MC and we will express it as an affine combination of the uniform marginal components MC^{UA} and MC^{UP}, capturing the effects of the relative income inequality RII on the macro consumption.

3.3.2 Micro and Macro equations in economics

The economic theory need to be explained at this stage. We consider a set of households, say a million of households

$$a \in \Omega \tag{3.3.5}$$

where the macro set of all the households

$$\Omega = \{a_1, ..., a_n\} \tag{3.3.6}$$

and we ask the following question:

How do incomes and consumption functions of "a" determine total consumption?

We define the micro consumption function to be non-linear, stocastic and agent-specific and we add an error term $\varepsilon(a)$:

$$c(a) = f^a(x(a)) + \varepsilon(a) = f^a(x_1(a), ..., x_K(a)) + \varepsilon(a), \tag{3.3.7}$$

the error terms ε are supposed to have vanishing conditional expectation, finite variances and small correlation across households, further, their mean is zero for the Law of Large numbers:

$$E(\varepsilon(a)|x(a)) = 0. \tag{3.3.8}$$

Despite the fact that the System of National Account (SNA) refers the total consumption as a sum of all the individual micro consumption function over households Ω

$$\widetilde{c} = \sum_{\Omega} c(a) = \sum_{i=1}^{n} c(a_i), \tag{3.3.9}$$

we will set the average consumption per household as the macro output in the following way:

$$\langle c \rangle = \frac{1}{n} \sum_{\Omega} c(a). \tag{3.3.10}$$

Considering the number of households n close to constant (say in time) or slowly changing, the variables $\widetilde{c}$ and $\langle c \rangle$ are practically constant multiples of each other.

3.3.3 Integration of micro and macro: The MicMac-function

The macro output in term of micro inputs can be formulated by plugging (7) into (10), we get

$$\langle c \rangle = \frac{1}{n} \sum_{\Omega} f^a(x(a)) = \frac{1}{n} \sum_{\Omega} f^a(x_1(a), x_2(a), ..., x_K(a)) \tag{3.3.11}$$

and the average of the error terms $\langle \varepsilon \rangle$ is zero for a million of households. The (11) is considered the MicMac-function, since it maps micro information to the macro level.

In the case of an individual household, the change in output $\Delta c(a)$ caused by input change $\Delta x(a)$ are derived as follow by the mean value theorem:

$$\Delta c(a) = \sum_{k=1}^{K} D_k f^a(\xi(a)) \Delta x_k(a) = \sum_{k=1}^{K} \overline{m}_k(a) \Delta x_k(a); \tag{3.3.12}$$

where

$$D_k f^a(\xi(a)) = \overline{m}_k(a). \tag{3.3.13}$$

In the case of macro level, (12) becomes

$$\langle \Delta c \rangle = \frac{1}{n} \sum_{\Omega} \sum_{k=1}^{K} \overline{m}_k(a) \Delta x_k(a) = \sum_{k=1}^{K} \frac{1}{n} \sum_{\Omega} \overline{m}_k(a) \Delta x_k(a) = \sum_{k=1}^{K} \langle \overline{m}_k \Delta x_k \rangle. \tag{3.3.14}$$

The (14) is still called the MicMac-function: its output is a macro variable but its inputs are micro vectors $\overline{m}_k$ and Δx_k; it is easier to aggregate than the level equation (12).

From now on we concentrate only on one explanatory variable, that is the income effect of the consumption function, thus (14) simplifies into the following inner product [2]

$$\langle \Delta c \rangle = \langle \overline{m} \Delta x \rangle. \tag{3.3.15}$$

[2]Inner product's properties are outlined in Appendix A

3.3.4 Decomposition of the consumption function

BLA [3] allows to decompose (15) into

$$\langle \overline{m}\Delta x\rangle = \langle \overline{m}\rangle\langle \Delta x\rangle + s(\overline{m})s(\Delta x)corr(\overline{m},\Delta x) \tag{3.3.16}$$

where

$$s(\overline{m})s(\Delta x)corr(\overline{m},\Delta x) = cov(\overline{m},\Delta x) \tag{3.3.17}$$

and (17) is zero because all the vectors Δx the great majority lies in the orthogonal complement, or, because there is a lack of correlation between $\overline{m}$ and Δx.

Another meaningful way of decompose (15) reveals the affine combination of the marginal coefficient and the input changes (19), where $\langle \Delta x\rangle \neq 0$

$$\frac{\langle \overline{m}\Delta x\rangle}{\langle \Delta x\rangle}\langle \Delta x\rangle = \langle \overline{m}\rangle_{\Delta x}\langle \Delta x\rangle \tag{3.3.18}$$

hence,

$$\langle \overline{m}\rangle_{\Delta x} = \frac{\langle \overline{m}\Delta x\rangle}{\langle \Delta x\rangle} = \sum_{\Omega}\frac{\Delta x(a)}{n\langle \Delta x\rangle}\overline{m}(a) \tag{3.3.19}$$

and consent us to examine the different effects of uniform absolute and uniform proportional changes:

- Uniforme absolute changes, where $\Delta x(a) = A$ is constant for all agents

$$\langle \overline{m}\rangle_{\Delta x} = \frac{\langle \overline{m}A\rangle}{A} = \langle \overline{m}\rangle; \tag{3.3.20}$$

- Uniforme proportional changes, where $\Delta x(a) = \Delta x^{UP}(a) = Px^0(a)$

$$\langle \overline{m}\rangle_{\Delta x} = \langle \overline{m}\rangle_{Px^0} = \frac{\langle \overline{m}Px^0\rangle}{\langle Px^0\rangle} = x^0. \tag{3.3.21}$$

[3] BLA is explained and proved in the Appendix B

It can be shown that in the case of uniform absolute changes, (20) is an unweighted average, in the other case of uniform proportional changes, (21) is a weighted average. [4]

3.3.5 Uniform absolute changes

All the steps derived so far become useful to express the connection between the macro aggregates $\langle \Delta c \rangle$ and $\langle \Delta x \rangle$, thus, by applying again the BLA to (15) we write

$$\langle \Delta c \rangle = \langle \overline{m} \Delta x \rangle = \langle \overline{m} \rangle \langle \Delta x \rangle + cov(\overline{m}, \Delta x), \tag{3.3.22}$$

since we know that $\Delta x(a) \approx A$, thus, it is constant, the covariance term vanishes identically; The same result can be obtained in the case where income does not change at all.

For *Uniform Absolute* changes where $\Delta x(a) = A$ for all the agents the *Marginal Coefficient* MC equals

$$MC^{UA} = \langle \overline{m} \rangle \tag{3.3.23}$$

and it is equally weighted aritmetic average of $\overline{m}(a)'s$ and hence

$$\langle \overline{m} \Delta x^{UA} \rangle = \langle \overline{m} \rangle \langle \Delta x^{UA} \rangle = MC^{UA} \langle \Delta x^{UA} \rangle. \tag{3.3.24}$$

3.3.6 Uniform proportional changes

More interesting changes are the uniform proportional ones

$$\Delta x^{UP}(a) = P x^0(a) \tag{3.3.25}$$

[4] State income taxation provides a good example: if large positive input changes are connected to large marginal coefficients (in this case large marginal taxes), then the macro marginal coefficient $\langle \overline{m} \rangle_{\Delta x}$ is larger than if all the income changes had been the same. Income weighted marginal tax rate is much higher than the unweighted one.

where the relative changes

$$\frac{\Delta x^{UP}(a)}{x^0(a)} = \frac{\Delta x^{UP}}{x^0} \tag{3.3.26}$$

can be associated to some constant P, say $0.02 = 2\%$.

If we divide and multiply by the positive $x^0(a)$ the (14) and we apply the BLA we end up with

$$\langle \Delta c \rangle = \langle \overline{m} \Delta x \rangle = \langle \overline{m} \rangle_{x^0} \langle \Delta x \rangle + \langle x^0 \rangle cov_{x^0}(\overline{m}, \frac{\Delta x}{x^0}), \tag{3.3.27}$$

where the covariance term varies around zero because usually relative changes of income are constant, hence, it vanishes identically if either tits variables are constant or are not correlated with each other. Further, the covariance term is negative in case of consumption because large absolute changes in income are connected with small marginal propensity to consume.

To conclude, for all *Uniform Proportional* changes where $\Delta x^{UP}(a) = Px^0(a) = P$ and P is constant, the covariance term also vanishes identically and the *Marginal Coefficient* MC equals

$$MC^{UP} = \langle \overline{m} \rangle_{x^0} \tag{3.3.28}$$

and

$$\langle \overline{m} \Delta x^{UP} \rangle = \langle \overline{m} \rangle_{x^0} \langle \Delta x^{UP} \rangle = MC^{UP} \langle \Delta x^{UP} \rangle. \tag{3.3.29}$$

3.3.7 Relative income inequality *RII*

So far we went trough the effects of the two different kinds of absolute changes, but what happen if we combine them? If we express the effective marginal coefficient MC in terms of MC^{UA} and MC^{UP} plus a non-systematic (or random) term u [5], we can apply the regression analysis rules:

$$\Delta x(a) = \Delta x^{UA}(a) + \Delta x^{UP}(a) + u(a) \tag{3.3.30}$$

[5]The non-systematic component is orthogonal: $u \perp \Delta x^{UA}$ and $u \perp \Delta x^{UP}$

thus at the macro level we get

$$\langle \Delta x \rangle = \langle \Delta x^{UA} \rangle + \langle \Delta x^{UP} \rangle \tag{3.3.31}$$

and dividing both side by $\langle \Delta x \rangle$, it gives us the weights that sum to unity

$$\frac{\langle \Delta x^{UA} \rangle}{\langle \Delta x \rangle} + \frac{\langle \Delta x^{UP} \rangle}{\langle \Delta x \rangle} = w^{UA} + w^{UP} = 1. \tag{3.3.32}$$

The weights are closely connected with the relative income inequality RII:

- if $w^{UP} = 1$, then incomes changes proportionally and RII stays constant;
- if both of the weights are positive, $0 < w^{UP} < 1$, relative changes of incomes of "poorer" increase more than incomes of the "richer" and RII decreases;
- if $w^{UP} > 1$ and $w^{UA} < 0$, RII decreases.

3.3.8 The effective marginal coefficient MC

We can now express the effective marginal propensity to consume MC as an affine combination [6] of MC^{UA} and MC^{UP}, the following is the final macro equation

$$\langle c \rangle = MC \cdot \langle \Delta x \rangle + cov(\overline{m}, u) \tag{3.3.33}$$

where

$$MC = MC^{UP} \cdot w^{UP} + MC^{UA} \cdot (1 - w^{UP}) \tag{3.3.34}$$

is the effective macro marginal coefficient. Nevertheless, the most of the income change normally comes from the proportional part and its relative weight $w^{UP} \approx 1 \rightarrow MC \approx MC^{UP}$.

[6] it is a weighted average only for positive weights and it lies between them.

3.3.9 Conclusions

Firstly, analyzing the economic meaning of our previous derivations, we attain that an income change can have different influences to our aggregate consumption. Suppose that a variation of income is associated to a simple transfer from high income earner (with low marginal propensity to consume) to low income earners (with higher marginal propensity of consume), even if the the total average income remains constant for every individual, the consumption does increase. That is

$$\langle \overline{m}\Delta x \rangle > 0 \iff \langle \Delta x \rangle = 0$$

and it does not occur if and only if all the marginal coefficient are the same, but it is a very restrictive condition.

Normally such effect cannot be explain by text-book model, where income is represented by total or mean income: a meaningful example is the System of National Account (SNA) that we have seen in section 3.2, more precisely in the equation (10).

Another important implication of these results can be verified in the case where the tax system adopts a progressive income taxation. Supposing again that the variation of income is a transfer from high income earners to low income earners, the mean effect is again zero but in this case the tax effect is negative. The negative tax effect is due to the fact that every high income earner has a larger marginal tax than the income receiver.

Secondly, the Keynesian consumption function allow us to examine the effective marginal propensity to consume MC in (34), that is not time-invariant but varies systematically from one situation to another one, hence we can replace its notation with MC^t. The latter decomposes into uniform absolute and uniform proportional marginal coefficient and is an affine combination of these. We stated that any change of MC^t and therefore

on consumption strictly depends on the income distribution, especially on changes of relative income inequality RII.

Finally, as a final results, we can generalize (33) and (34) for any output and input variables:

$$\langle \Delta y^t \rangle = \sum_{k=1}^{K} MC_k^t \cdot \langle \Delta x_k^t \rangle + \varepsilon^t \tag{3.3.35}$$

$$MC_k^t = MC_k^{t,UP} \cdot w_k^{t,UP} + MC_k^{t,UA} \cdot (1 - w_k^{t,UP}) \tag{3.3.36}$$

$$\varepsilon^t = \sum_{k=1}^{K} cov(\overline{m}_k^t, u_k^t) \tag{3.3.37}$$

and underline that $MC_k^t \cdot \langle \Delta x_k^t \rangle$ is the macro effect of a single variable. Further, an important consideration needs to be done in the case where the effective marginal propensity to consume distributes symmetrically around zero, $MC_k^t \approx 0$, therefore, a variable has a significant effects on the micro level but practically none on the macro level.

4 Agent Based Model (ABM)

I will devote this entire section to the detailed explanation of the Agent Based Models (ABMs), their pros and cons, the goals that drive us to use them, their different purposes and much more.

Firstly, I would provide a definition of what ABMs are and the their strengths.

Secondly, I will briefly outline the computer simulation fore-runners that have contributed to their birth and in which way we can find some similarities between them.

Thirdly, I will highlight all the main ABM features that allow us to divide them according with the characteristics or the goals they have in common.

Then, I'll show how ABMs can be meaningful to explain results dictated by economic theories.

Finally, I will conclude by motivating my decision to use the ABM approach (NetLogo language) to find out about the Micro-Macro link and why we obtain different results by applying the aggregation theory of Vartia to "Mathematical models" instead of ABMs.

I found the recent Squazzoni's book[19] very interesting and intuitive, for that reason I would like to base this section on his book mainly and I will also adopt the same ABMs classification he used in social research area.

4.1 What are ABMs?

Gilbert (2008)[5] stated that an ABM can be defined as a "computational method that enables a researcher to create, analyze, and experiment with models composed of agents that interact within an environment" but, if we want to give a less general explanation we should say that it is a class of computational models for simulating the actions and interactions of autonomous agents (both individual or collective entities such as organizations or groups) with a view to assessing their effects on the system as a whole. It combines elements of game theory, complex systems, emergence, computational sociology, multi-agent systems, and evolutionary programming. Technically, ABMs are a purely application of artificial intelligence to social science. Thank to the last statement it is easy to conclude that they can be employeed in a wide variety of fields of study. The most important features that distinguish the ABMs from other computer models are

- Interactions (nonlinear), they study how agents have the power to modify their choices with respect to somebody else choices and output a completely unexpected result.
- Heterogeneity, we do not have here an homogeneous representative agents, being always on the base of every economic theory.
- Explicit representation, agents and environment are represented with all the satisfactory features that mark them out.
- Emergence, complex systems develop over time, 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.
- Micro-Macro relationship, all the links that occur, either from Micro to Macro level or on the way around, are considered with meticulous

attention.

- Correspondence, models used in different research areas can be discussed and related to different subjects. One may create a team work which leads to a faster development of the model.
- Visualization techniques, they provides a sophisticated interface that allows us to investigate and observe the complex interaction.

A necessary step is defining what ABMs are used for. Axtell (2000)[1] identified two different approaches according to "what does a researcher need".

A first group of researchers engage the computational power of ABMs to support and complement their analytic models. In that contest ABMs are able to provide solutions where differential equation systems fail in their attempt. In this way they are considered as an extension of math models, with object-oriented programming languages used to reproduce or translate equation-based objects. Usually, the first approach is often associated with game theory that represents a conventional theoretical framework that leads to a better explanation and formalization of a given field of study, such as social science.

A second group of researchers, according with Liebrand (1998)[13], substitute analytic models with the object oriented programming languages in order to model complex system with autonomous and heterogeneous agents. In that contest the Macro behavior of the system is not known in advance and no assumption of equilibrium and restrictive other implication are given. Only by running the model that has been expressed in ABM language, the simulation tells us what the system behavior is. Thank to this potentiality we do not have to fit the model with strict interpretation and, in that sense, ABMs are the perfect tools to build sophisticated and empirically grounded models.

To have a better idea of what ABMs are, we should definitely have a look back at the past, the "history", in order to understand what the main steps that contributed to their birth and growth are.

4.2 History

The idea of agent-based modeling was developed as a relatively simple concept in the late 1940s. Since it requires computation-intensive procedures, it did not become widespread until the 1990s. Starting from 1990s the rise of distributed artificial intelligence and the diffusion of the object oriented programming sped up the diffusion of this technology, on which ABMs are built on.

The following are the main computer simulation fore-runners, they are listed in terms of similarity to the ABMs, from the most different to the most similar:

- *System Dynamics* was employed in understanding industrial processes and became a valid support to underline the policy analysis and management either in the private or in the public sector. Its strength is the capability to evaluate a nonlinear interaction, thus, any change of each variable is correlated with other variables change. The assumption that lie on the base is that the system has represented by a circular and time-delayed relationship between structural components, factors and variables. It is not trivial trying to estimate the result for a change of even one single variable. Its disadvantage is the lack of heterogenous component, it only provides interdependence and feedback among variables, describing the system structure *ex-ante*, on the contrary of the ABMs.
- *Microsimulation* is considered a unit-record model based, units can be associated with individuals, let's say household, that are treated as a

records containing a set of all the attributes and a unique identifier. Records are, for instance divided depending on age, sex, marital status and so on, and they come from dataset or survey. The idea is that the model is based on certain deterministic transition probabilities that have the power to change the behavior of each unit and reflect the impact they have on certain aggregate variables of interest. For that reason, heterogeneity in term of distribution parameters (not in term of behavior) is under observation. The drawback is that interactions between single units are missing, hence it cannot explain a macro common behavior depending on social interactions.

- *Cellular Automata* are used to model local interactions the most. These interactions are mainly between units, but, they are able to output a macro effects. They are built on a grid of cells that own a finite number of states, such as "on/off". The states change over time according with other cells (let's say the neighboring cells). The problem here is that interactions between micro units are associated with a single homogenous parameter that leads to a poor approximation in understanding complex social interactive systems.

Having analyzed pros and cons of the previous models, one should not be surprised if the advent of ABMs have been able to fill the gaps and implement innovative techniques for the study of complex systems.

We shall now highlight all the main ABM features that provide the keys to categorize them into different groups.

4.3 ABMs: different types, different goals

They can be distinguished in five types according with their purposes and the link they would represent between model and empirical reality. Before

we outline the five categories, first of all, we need to assign them a roughly subdivision: *Analytical, Synthetic* and *Applied models.*

4.3.1 Synthetic models

Synthetic models can provide "what-if" scenarios which might reveal non-obvious features of reality. From the name "synthetic", they are the best models to synthesize any component of social life and derive intuitions as much realistic as possible. They model aspects of evolution of any behavior (especially social behavior) and any kind of "structure" that are found difficult to study by using empirical or experimental models. For the underlined reasons, they are considered the most explorative approach to computer simulation.

1) *Artificial societies* models are surrogates of empirical or experimental models and provide sensitive results where the latter cannot explain particular phenomena, for instance, due to the lack of data or the existence of restrictive constraints such as ethical, time or budget limitations. A surrogate model allows a detailed re-enactment of the real system, therefore by reproducing the reality in an "artificial life", researchers are able to explore intuitions and significant aspects concerning the evolution of agents behavior in the real system. More often than not, these kinds of models are defined "trans-disciplinary" models, because of their feature of including aspects that are usually common to different fields and disciplines. Nevertheless, however, nowadays artificial models are not so popular and rather poorly developed.

The awkward point is represented by the difficulty of transforming the result one obtains into empirically testable finding. That is worthy of note.

4.3.2 Analytical models

Analytical models include models that: are of paramount importance for the progress of science but abstract research can easily lose sight of empirically relevant aspects (Abstract models); can do a great deal to connect evidence and theory and to focus on well-specified social mechanism but both their empirical validation in single important cases and their theoretical generalization are extremely challenging to achieve (Middle-range models); might help to refine the validity domain of general theories and provide histories that might inspire theory building, but their theoretical generalization requires an enormous and well-organized collection of evidence, not always at hand (Case-based models).

2) *Abstract models* are usually adopted for estimating phenomena at general range. The aim here is to support theory building and their developments, any generalizations about the matter are very common. The interesting fact is that it might happen to come to discover that some models, at the end, present non-obvious properties related to the interactions, being at odds with our expectations. There is, thus, a need to formalize such properties in a theoretical framework, useful for empirical studies.

   The bad point of Abstract models is that we easily lose the sense of the real empirical explanation.

3) *Middle-range models* gather empirical phenomena with same features in a group (e.g. common resource management in local communities), having the intent to investigate specific mechanisms. The name "Middle-range" is due to the fact that they create a bridge between theory and empirical evidence and allocate finding in the proper specific empirical fields. Willer and Webster (1970)[25] declared that thanks

to their heuristic[7] and pragmatic value, these models should not fully correspond to empirical reality, which is the target of their explanation. To emphasize, the further these models are from the empirical phenomena which class refer to, the stronger their heuristic value.

The good point is that they develop theories to derive testable findings and articulate theoretical explanations about the behavior of real systems; the bad point instead, is the difficulty to generalize their findings in order to test them again.

4) *Case-based models*'s aim is to achieve knowledge of the empirical situation, with accuracy, precision, and reliability. They are the good tool to appreciate complexity rather than achieving generality (Ragin 1987)[15]. They are based on pre-constituted theoretical hypotheses and often exploit general modeling frameworks. Perts of theoretical findings or well-known theories are often used both to approach the empirical puzzle and to build the model.

   Even if they can provide insights for theory building, they suffer from theoretical generalization.

4.3.3 Applied models

Applied simulations are replications of real systems that help researcher to solve important problems, outline potential re-engineering options, point out different policies in order to assist planners and decision makers, improve any knowledge of individuals that are part of real system where they are involved

[7] Heuristic, from the Greek "heuriskein" meaning "to discover", pertains to the process of gaining knowledge or some desired result by intelligent guesswork rather than by following some pre-established formula. Heuristic can be used for (1) describing an approach by "learning-by-trying", that is, the "trial-by-error" learning or (2) applying the general knowledge gained by experience, sometimes expressed as using a "rule-of-thumb".

day by day. The focus here is on what the consequences of those agents are with respect to the simulations that represents their typical behavior.

5) *Applied simulations* consist in building models that should map real systems as closely as possible. For that reason that they are *ad hoc* models. They allows researchers to implement or adjust pre-existing theories that can be either working or unattainable theories. Being a completely different approach comparing with the mathematical ones, they are useful to evaluate "new sides" of investigating and solving problems that occur in real systems.

The drawback of these models is that they suffer from a theoretical generalization and sometime it is very difficult to transform them in an applicable theory.

Note that not all these five types of models have been equally explored. Moreover, we should consider that the added value of each type of model and its potential for the development of any discipline need to be taken into account and implement much more, in and for the closest future, for giving a better opportunity to exploit their potential.

The following section will stress how ABMs can be used to explain economic theories. In a way, it is what I will represent soon, as an example, in my first simulation concerning the basic consumption function, linked with the production function and the different endowments that do limit the perspectives that have been considered by the simple model.

4.4 Why can ABMs explain results dictated by economic theories?

The section will deal with the applications of ABMs to economic theories in order to build up meaningful economic explanatory model that are able

to illustrate complex system, especially the link between micro and macro level.

Liebrand (1998) stated that ABMs languages and their logic are used to model a system of autonomous and heterogeneous agents, where the macro system behavior is not known in advance, and no fixed close and equilibrium solutions are attainable.

His statement is full of significance: in a sentence only, he went through the leading meaning that lies on the base of object oriented programming language.

(i) autonomous and heterogeneous agents

Here the famous dialogue from the Monty Python's Life of Brian:

"BRIAN:	Look. You've got it all wrong.
	You don't need to follow me.
	You don't need to follow anybody!
	You've got to think for yourselves.
	You're all individuals!
FOLLOWERS:	Yes, we're all individuals!
BRIAN:	You're all different!
FOLLOWERS:	Yes, we are all different!
DENNIS:	I'm not."

The base of every economic theory has always been the representative agent, but what exactly is a representative agent?

4.4.1 Representative agent

The majority of the economic models, that we studied and that have been developed so far, can be considered representative agent models.

A representative agent model is a model in which all agents act in such a manner that their cumulative actions might be the best set of actions of one agent maximizing its expected utility function (for instance). Economists construct representative agents in order to deal with the complicated issue of aggregation. Regard that, it is much easier to model the behavior of one person given some preferences and constraints than the behavior of a group of people, or an entire economy.

Since aggregation is what economists call the summing up of individuals' behavior to derive the behavior of a market or an economy, every model that aggregates individuals using a representative agent device does not consider any heterogeneous effect of a given group of people and, hence, their interactions. We can now better understand why an ABM that allow heterogeneity components is so precious.

Kirman (1992)[11] asserted that a representative agent models simply ignore valid aggregation concerns. He provided an example in which the representative agent disagrees with all individuals in the economy and concluded that the reduction of a group of heterogeneous agents to a representative agent is not just an analytical convenience, but it is "both unjustified and leads to conclusions which are usually misleading and often wrong." In his view, the representative agent "deserves a decent burial, as an approach to economic analysis that is not only primitive, but fundamentally erroneous."

A stronger statement was made by Chang, Kim, and Schorfheide (2011)[16] in the context of a DSGE model, they estimate a representative-agent DSGE model on the basis of the aggregate data implied by their heterogeneous-agent economy, and show that the estimated coefficients are inconsistent with the true parameters of the heterogeneous economy. They point out that "Since it is not always feasible to account for heterogeneity explicitly, it is important to recognize the possibility that the parameters of a highly-

aggregated model may not be invariant with respect to policy changes."

(ii) macro system behavior is not known in advance

What is the key tool to show how the micro is linked with the macro level?

4.4.2 Micro links Macro

The Nobel Prize winner Simon (1969)[18], one of the most prominent social scientists of the last century, influenced a wide range of disciplines, from artificial intelligence to organization science and psychology. He claimed that there is no *isomorphism*[8] between the complexity that social systems show at the macro level and their complexity at the micro level. Therefore, in many cases, the former is nothing more than the result of interaction between simple micro processes.

Moreover, by following his point of view, what is crucial is the *organization* that the parts arise. For the underlined reasons, computer simulation is fundamental to simplify and model complex social systems by a micro–macro approach. Having the chance to evaluate the interactions in such a way, it allows researchers to omit detailed knowledge of the behavior of each individual component in order to explain the micro-macro link. As a conclusion, there is no assumption to be made regarding the causal autonomy of macro level in explaining micro behavior course, rather, there is a need to study the latter by analyzing organization starting from micro processes.

(iii) no fixed close and equilibrium solutions are attainable

[8] Isomorphism, from the Greek iso, meaning "equal", and morphosis, meaning "to form" or "to shape", in mathematics is a one-to-one correspondence between the elements of two sets such that the result of an operation on elements of one set corresponds to the result of the analogous operation on their images in the other set.

Further, there is another issue that is strictly associated with the Simon's conclusion: in such environment there is no need to force economics complex systems by imposing the presence of equilibrium (or equilibria) in advance, otherwise, it will only cause restrictions.

4.5 Why did I choose an ABM approach?

It is now clear, from the whole section, why i decided to employ ABMs to confront the results we obtain with simulation to Vartia's theory findings. To a greater extent, I would also add the speech of Trichet, President of the ECB, Opening address at the ECB Central Banking Conference (Frankfurt, 18 November 2010):

"First, we have to think about how to characterize the homo economicus at the heart of any model. The atomistic, optimizing agents underlying existing models do not capture behavior during a crisis period. We need to deal better with heterogeneity across agents and the interaction among those heterogeneous agents. We need to entertain alternative motivations for economic choices. Behavioral economics draws on psychology to explain decisions made in crisis circumstances. Agent-based modeling dispenses with the optimization assumption and allows for more complex interactions between agents. Such approaches are worthy of our attention."

In the speech he stressed that there is a strong necessity of a change in our models nowadays. Could that change mark the starting point for the mounting exploitation of object oriented programming language to built new meaningful models?

From my personal point of view, it is time to change, it is time to give a chance to ABMs.

Subsequently, I will provide a briefly introduction to one of the most popular language, the one I choose for my simulations: NetLogo.

4.6 NetLogo

NetLogo is an agent-based programming language and it has a programmable modeling environment for simulating natural and social phenomena.

It was authored by Uri Wilensky, director of Northwestern University's Center for Connected Learning and Computer-Based Modeling, in 1999 and has been in continuous development. It is free and open source software, under a GPL license, It is written in Scala and Java and runs on the Java Virtual Machine.

The NetLogo environment enables researchers to explore emergent phenomena. One of the most valued feature is the possibility to create models in a different domains of study, such as economics, biology, physics, chemistry, psychology, system dynamics and many other natural and social sciences.

NetLogo takes part of the most updated generation of the series of multi-agent modeling languages. The following are the main historical passages that have contributed to its birth:

- LOGO (Papert & Minsky, 1967) is based on the theory of education based on Piaget's constructionism ("hands-on" creation and test of concepts), it is a simple language derived from LISP having a turtle graphics and exploration of "micro-worlds".
- StarLogo (Resnick, 1991), MacStarLogo and StarLogoT are all built on agent-based simulation language that allows exploring the behavior of decentralized systems through concurrent programming of numerous turtles.
- NetLogo (Wilensky, 1999) is an extension of StarLogo distinguished by the presence of continuous turtle coordinates, cross-platform, network-

ing, etc. It is the most popular program between the existing ones, due to the fact that it has a very growing library of models.

NetLogo is particularly well indicated for modeling complex systems, especially if they develop over time. Modelers can give instructions to hundreds or thousands of independent "agents" that can interact either by operating concurrently or cooperating, depending on the situation that is required by researchers.

Its potential will allow us to explore the connection between the micro-level behavior of individuals and the emerging macro-level results, that is the purpose of our work: the link that maps the relationship between micro and macro level, in other words the aggregation.

The following are the simulations codified by NetLogo language.

5 Simulation model through an explanatory example

I will devote the entire section to explain how you can apply an economic theory to an ABM.

This section is used to understand the basics of NetLogo. That is why the example is extremely simple. Starting from an example of poor grade will help us to understand the next simulation, being the tool to derive the final results and comments about the aggregation.

In order of appearance: we start by seeing what the main features of NetLogo are, how it works and how can be programmed from scratch. I would also explain the meaning of each line of code.

The next section will show how to approach NetLogo, then we'll go through the programming language and finally we would evaluate the analysis of results coming from simulations.

5.1 NetLogo overview

The NetLogo world is made up of agents. Agents are beings that can follow instructions. Each agent can carry out its own activity, all simultaneously.

In NetLogo, there are four types of agents:

- *Turtles*, they are agents that move around the world. The world is two dimensional and is divided up into a grid of patches.
- *Patches*, each patch is a square piece of "ground" over which turtles can move.
- *Links*, they are agents that connect two turtles and
- *Observer*, he doesn't have a location, thus, you can imagine him as looking out over the world of turtles and patches.

When NetLogo starts up, there are no turtles yet. The observer can make new turtles. Patches can make new turtles too. Patches can't move, but otherwise they're just as "alive" as turtles and the observer are. Patches have coordinates. The patch at coordinates (0, 0) is called the origin and the coordinates of the other patches are the horizontal and vertical distances from this one. We call the patch's coordinates pxcor and pycor. Just like in the standard mathematical coordinate plane, pxcor increases as you move to the right and pycor increases as you move up (depending on where you set the origin). The total number of patches is determined by the settings min-pxcor, max-pxcor, min-pycor and max-pycor. When NetLogo starts up, min-pxcor, max-pxcor, min-pycor and max-pycor are -16, 16, -16, and 16 respectively. This means that pxcor and pycor both range from -16 to 16, so there are 33 times 33, or 1089 patches total. You can change the number of patches with the Settings button.

Turtles have coordinates too: xcor and ycor. Patch's coordinates are always integers, but turtle's coordinates can have decimals. This means

that a turtle can be positioned at any point within its patch; it doesn't have to be in the center of the patch.

Links do not have coordinates, instead they have two endpoints (each a turtle). Links appear between the two endpoints, along the shortest path possible even if that means wrapping around the world.

The way the world of patches is connected can change. By default the world is a torus which means it isn't bounded, but "wraps". Hence, when a turtle moves past the edge of the world, it disappears and reappears on the opposite edge and every patch has the same number of "neighbor" patches; if you're a patch on the edge of the world, some of your "neighbors" are on the opposite edge. However, you can change the wrap settings with the Settings button. If wrapping is not allowed in a given direction then in that direction (x or y) the world is bounded. Patches along that boundary will have fewer than 8 neighbors and turtles will not move beyond the edge of the world.

NetLogo allows you to define different "breeds" of turtles and breeds of links. Once you have defined breeds, you can go on and make the different breeds behave differently. Note also that turtles can change breeds.

Please consider now Figure 7 that is the "Interface of NetLogo" where the number having a green background allow us to describe every single part of the interface.

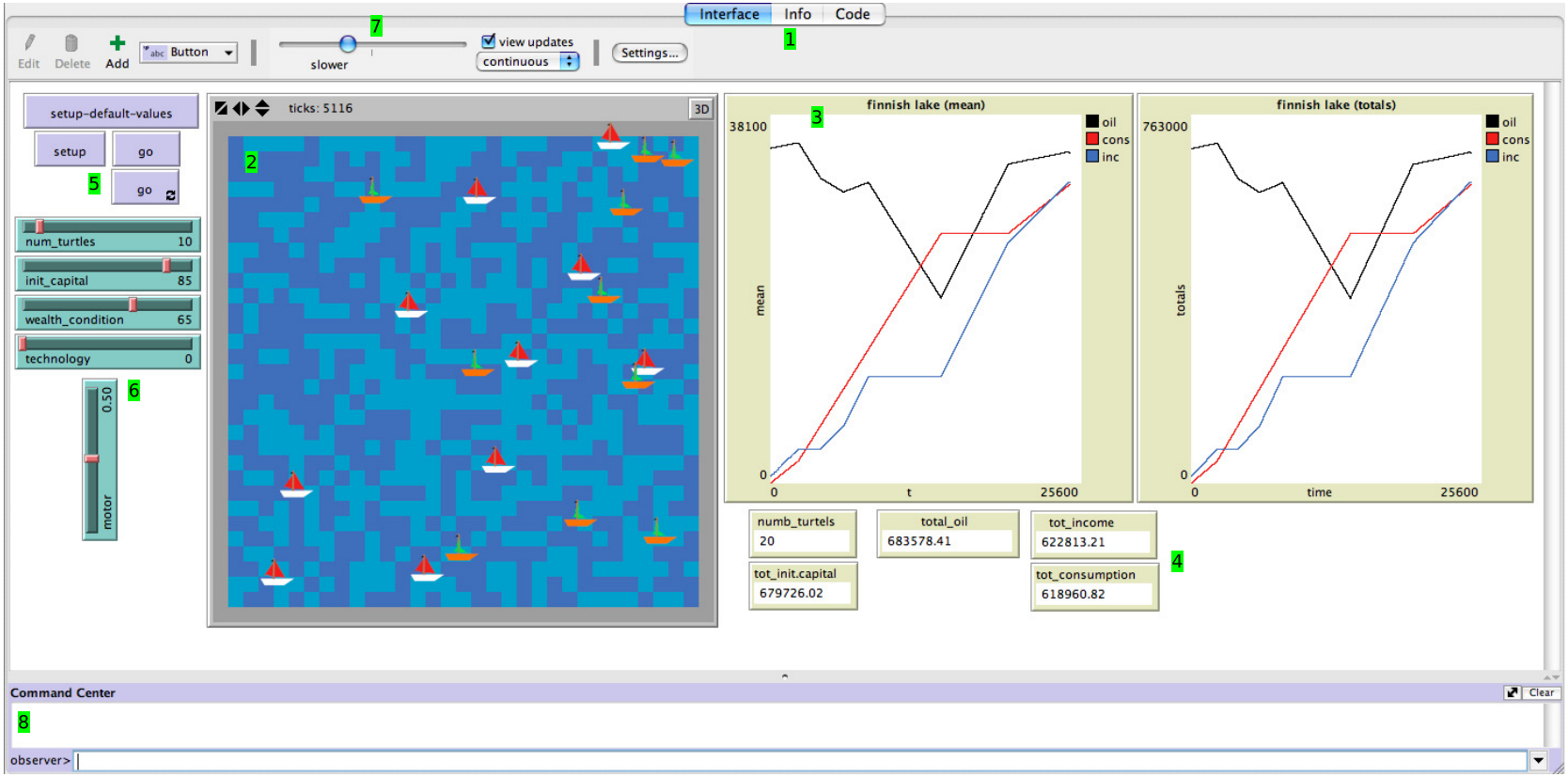

Figure 7: NetLogo interface

Number 1: shows different displays we can choose: *Info* is where the model is usually described in every feature; *Code* is the place where you program (digit the language code) and *Interface* is the graphical display that gives the results of the code and allow simulations.

Number 2 is the word, where all the patches and turtles are.

The "agents" (called turtles) can represent individuals (e.g. people), social groupings (e.g. firms), and/or physical systems (e.g. transport systems). In my model, agents are a group of fishermen. Fishermen are divided into two types: fishermen who produce and fishermen who do not produce.

In every model, the modeler provides the initial configuration of a computational economic system that is the "world" where all the actions are taken. In our case, the world is represented by a small and simple reality: a lake, a Finnish lake, where fishermen can fish and sail. Interactions with the outside are not considered. One can associate an example of a closed economy for instance.

In spite that it is a closed world, it does not mean that the final results are predictable. Actions are taken by multiple interacting agents that give rise to the emergence of a complex system based on non-linear functions and difficult mathematical derivation.

We shall move to understand how the *Buttons* on the display do work. We can see from the top left of the Figure that we can add as many buttons as we need, but what are they used for?

Buttons provide an easy way to control the model, normally a model has at least a "setup" button, to set up the initial state of the world, and a "go" button to make the model run continuously. Some models will have additional buttons that perform other actions. Every button contains different NetLogo code. That code is run when you press the button.

A button may be either a "once button", or a "forever button". Once

buttons run their code once, then stop and pop back up. Forever buttons keep running their code over and over again, until either the code hits the stop command, or you press the button again to stop it. If you stop the button, the code doesn't get interrupted. The button waits until the code has finished, then pops up. In my model the "go" button is either a once or a forever button (*Number 5*); that can be seen from the small symbol of repetition that marks the distinction of the two. Normally, a button is labeled with the code that it runs. For example, a button that says "go" on it usually contains the code "go", which means "run the go procedure". Procedures are defined in the Procedures tab, but, you can also edit a button and enter a "display name" for the button, which is a text that appears on the button instead of the code. You might use this feature if you think the actual code would be confusing to your users.

When you put code in a button, you must also specify which agents you want to run that code. You can choose to have the observer run the code, or all turtles, or all patches, or all links. (If you want the code to be run by only some turtles or some patches, you could make an observer button, and then have the observer use the ask command to ask only some of the turtles or patches to do something.)

Number 3 are called *Plots* and they are a couple of graphs that go along the model path. The one on the left shows the mean of our values, the one on the right instead is the total. Both of them have on the x-axis the time; in NetLogo, we should say that they have ticks (the ticks are also enumerated on the top left of the world). On the y-axis they have different values: the means and the totals respectively. The legend on the top right of both graph describes what the colored lines in the graph represent.

Number 4 is located around five buttons that are called *monitors*, they count variables that are under our consideration. In my model they rep-

resent: *numb_turtles* is the initial number of agents, *tot_init.capital* is the initial capital sum of all the turtles, *total_oil* is the total oil over the agents, same for *tot_income* and *tot_consumption*. (we will see in the next section what do they mean)

Number 6 marks five green buttons called *Sliders*. Their structure enables to set a min and a top value. We might also set the amount of increment we would like to have (e.g. 1 or $0,0003$). All those values are variables that let our model to change. For that reason, they are in the interface where we display the results by modifying the values of some variables and see what happen. The model has five sliders, the first, *numb_turtles* defines the number of each breed we would like to have from the beginning[9], *init_capital* is a variable that allows to set the initial capital of every agent, *wealth_condition* is a supplementary variable that boosts the initial capital, *technology* is linked with the production function and precisely it defines the technology of production of every agent. The last, *motor*, is associated with the consumption function and it sets the degree of consumption. Simply, think about sailing with motor, hence consume oil, or sailing without, (e.g. by exploiting the wind) thus, no oil is consumed.

Number 7 set the speed of the model interactions, hence the simulations. Usually, the modeler, after having run the program (press the go button), steps back to observe the development of the system over time without further intervention. The speed of the evolution is set with that slider.

Number 8 is a grid where commands can be typed in, either observer code or general commands.

[9] *numb_turtles set at 10 means that we will have 10 agents for each breed. Hence, 20 agents: 10 fishermen and 10 ufishermen.*

5.2 Programming language

We will see soon how we can achieve the same economic results basing the economic functions on computer programming. The purpose of the model is to represent the consumption and the production function, that have been outlined in Chapter 2, in a ABM framework. In detail, system events are and should be driven by agent interactions without external imposition of equilibrium conditions. For that reason, despite the presence of budget constraints, we will allow for agents to exceed their initial equipment of budget and thus, the consumption action is not limited. This is one of the most important rule of ABMs.

We will now explain every single line of code that is, from our point of view, the best way to understand the language. We will attach the NetLogo language in dark grey color and we will comment on the bottom of every set of procedures.

1.

```
breed [fishermen fisherman]
breed [ufishermen ufisherman]


fishermen-own [
  init.capital consumption oil income
  ]
ufishermen-own [
  init.capital consumption oil income
  ]
```

Normally, the first part of every model is devoted to define the agents, precisely the breeds. We have already mentioned that this model has

two different kind of fishermen: the first type can produce, the second is not able to produce and hence, it can only consume. Shall we define the former as *fishermen* and the latter as *ufishermen* (u = unemployed fisherman). NetLogo requires to provide both the name of the whole group and the singular name of one agent. That is why every agent has a name that usually is, if it is not defined differently, the name of the breed and a number (related with the amount of total turtles).

Then, we define the variables of every breed. In this case, both have *init.capital*, *consumption*, *oil*, *income*.[10]

The first variable is the initial endowment of capital, the one that is set by the modeler.

The second, consumption counts every unit that has been consumed by the turtles observed.

Income is related with capital. Economic theory differentiates capital from income; we have already made the distinction in Chapter 2. We will now just underline that income can be seen as a result of the amount of capital multiplied by an interest rate. As a conclusion the "initial income" will only be the interest of the capital. Further, the model allows an increase in income by the production function. Since Fishermen can produce, they earn income.

Finally we have the *oil* variable that is a balance between all the variables previously illustrated.

2.

```
to setup-default-values
  set num_turtles 10
```

[10] Usually if all breeds have the same variables, modeler tends to blend agents in a unique procedure. As a conclusion we will not see *fishermen-own* and *ufishermen-own*, but only *turtles-own*.

```
  set init_capital 85
  set wealth_condition 65
  set technology 5
  set motor 0.5
end
```

Note that every procedure starts with *to* and ends with *end*. It is a feature of NetLogo language and it might be common to other languages too.

setup-default-values is always used to fix (*set*) the default values. In this case, we set the default values of each *slider* button. It is the procedure to define the *setup-default-values* button we saw on the top left of the NetLogo interface. By clicking the button we impose the program to run with these values.

3.

```
to setup
  ca
  setup-fishermen
  setup-ufishermen
  setup-world
  reset-ticks
end
```

Here we have the *setup* procedure for the homonym button on the NetLogo interface. It is one of the principal procedure allowing the initial setup of our "world". In this case, we setup both the breeds and the world (the patches). Setup procedures are linked with the following lines of code. Modeler usually prefers to divide the code in a group and

subgroups. (that is why I will specify later what they are about)

Then we have *reset-ticks* that resets the "time" starting from the beginning (e.g. tick = 0).

Further, *ca* is the abbreviation of *clear-all*.

4.

```
to go
   ask fishermen [
     sail
     trawl
     update.values
     ]
   ask ufishermen [
     sail
     update.values
     ]
   tick
end
```

It is the main procedure and, after having set the agents and the world, it represents the start of the simulation. *go* defines the code of the homonym button on the interface. It means "when i click the go button, ask fishermen to sail, trawl and update values". Same for ufishermen, but without trawling, since they cannot produce (we will see soon what does trawl mean).

tick allows NetLogo to put the "clock" forward one tick (for instance, a minute has past).

5.

```
to setup-fishermen
```

```
  create-fishermen num_turtles [
    set shape "boat"
    set size 3
    values
  ]
end
```

setup-fishermen defines the features of the breed fishermen, that are: the initial number, the shape of the turtles (NetLogo has many different shapes in order to allow the modeler to differentiate the agents in the interface), the size and finally the values. Since values are common to both breeds, they will be set subsequently.

6.

```
to setup-ufishermen
  create-ufishermen num_turtles [
    set shape "uboat"
    set size 3
    values
  ]
end
```

The only differences from the last set of procedures (5.) is that we are now defining the ufishermen and, for this reason, we need to set a different shape. It is not mandatory but we like to localize the movement of the different agents on the screen. (Figure 7 shows that fishermen have "topsails" and are red & white, while ufishermen have "courses" and are green & orange)[11]

[11]"topsails" and "courses" are part of the sailing language and they mean respectively:

7.

```
to setup-world
  set-patch-size 14

  ask patches with [
    pxcor > -16 and pxcor < 16 and
    pycor < 16 and pycor > -16
    ]  [set pcolor one-of [sky blue]]

  ask patches with [
    pxcor = -16
    ]  [set pcolor grey]

  ask patches with [
    pxcor > -16 and pycor > 15
    ]  [set pcolor grey]

  ask patches with [
    pxcor = 16
    ]  [set pcolor grey]

  ask patches with [
    pxcor > -16 and pycor < -15
    ]  [set pcolor grey]
end
```

setup-world are procedures that set the world. Previously, we men-

to be under sail or, in the second case, the opposite.

tioned that our world can be seen as a closed economy, for that reason we need to erect the borders. The latter here are the grey patches around the perimeter of the square on the interface. They differentiate from the blue patches that are the water of our finnish lake. By simply changing the patches color, we can now avoid agents to go in the grey patches and hence we limit the world.

We have discovered the *ask* procedure. It is composed of two square brackets because of the *with*, but it can also be with one only. It asks to every patch having a pxcor equal to 16 to become grey instead of blue or sky (it is referred to the second last procedure). *with*, in this case, allows to set a condition: if such condition is verified, *ask* runs the command in the last square brackets.

Finally, the first procedure *set-patch-size* defines the size of every patches.

8.

```
to values
  set init.capital 0
  set init.capital
                  init.capital
                 + random-float 1000 * init_capital
                 + random-float 10 * wealth_condition
  set income
            income + 0.02 * init.capital
  set consumption 0
  set oil
         oil + income + init.capital
end
```

We have already met *values*: it was either in (5.) where we set up the

fishermen or in (6.) for ufishermen. Such procedure sets all the values of each agent variables.

Firstly, it sets the initial capital at zero and then, it updates the initial capital with a *random-float* 1000, that is a random number that does not exceed 1000, multiplied by the slider *initial_capital*. Usually, *random-float* is used to allow agent to be heterogeneous, in that way, each agent will have a different amount of initial capital.

Thirdly, we set income by multiplying 0.02 with the slider *initial.capital*. We have already consider this point and we conclude by assuming that the initial income is basically the interest calculated over the initial capital,. For instance, in this example, we have $r = 2\%$ (where r is the interest rate).

Then, we set the variable of consumption equal to zero.

Finally, we impose *oil* to be equal income plus initial capital. *oil* will be the balance of all the values. For instance, consider *oil* as an account where we direct the positive income coming from production and where we withdraw resources to consume. It is not by change that it is called oil: fishermen need oil to consume (sailing) and they invest the income earned to buy oil, in order to keep on producing (fishing).

9.

```
to sail
  if pcolor = grey
    [set heading
                heading + random-float 180
      ]

  ;if oil > 0 [
    fd 1
```

```
    set consumption
                    consumption + random-float 4 * motor
  ;]
end
```

First of all, I would introduce the *if* procedure. As can be seen here, *if* considers a condition, that is the grey patch, and then, in the case the condition is satisfied, it commands what to do. More precisely, if the patch is grey, hence there is no water any more, it changes the course of sailing. In other words, it changes the direction of the turtles and sets its degree by a random number less than 180.

Then, we see a procedure that starts with a semicolon, when a semicolon at the beginning of the row is present, it invalidates all the row (semicolon can be found at the second last row, before *end*, being the square bracket part of the oil procedure). We will soon understand why semicolon is imposed here.

Now, we will come to discover the *sail* procedure entirely: it represents the consumption function. We have already discussed the consumption function economically, we stated that it is shown as an affine combination of autonomous consumption that is not influenced by current income and induced consumption that is influenced by the economy's income level (Section 2.2.1).

Here, consumption is free of budget constraints, in this way we do create an autonomous consumption.[12] Further, we have already argued

[12]In the case we need the consumption function to be the result of the disposable income only, we provide a line of code where we set the budget constraint ($oil > 0$). In this way the consumption will stop as soon as agents will finish their income (they stop sailing). It only needs to be enabled by eliminating the semicolon before the *if* procedure and the relative square bracket on the bottom of the code.

that our model supports the consumption function simply by allowing agents to sail. For that reason, being free of constraints, we only need for fishermen to sail. That is why the code set a *fd* command, that is, in other words, "go forward one position" since it is the abbreviation of the *forward* command. After that it updates the consumption balance adding a number randomly less than 4 multiplied by the value of our *motor* slider. The slider considers a random consumption among agents and hence heterogeneity.

10.

```
to trawl
  set income
            income + random-float technology
end
```

Here comes the production function that has the same features of the consumption function we have just analyzed.[13] There is no command but updating the income by adding a random number imposed by the *technology* slider. In such a way we imagine fishermen, once on a patch, are able to trawl. The result strictly depends on the technology they use. For instance, a bigger boat allowing a larger dragnet.

11.

```
to update.values
  set oil
         income - consumption
end
```

The last procedure updates our balance, *oil*, by adding the income

[13]Remember that *fishermen* only produce; *ufishermen* are considerate unemployed.

earned and subtracting the consumption carried out.

We now turn to comment the results coming from the simulation.

5.3 Simulation

This section, dedicated to the simulation, will explain how the results, arising from the aforesaid functions of consumption and production, will vary and how our budget will be composed at the end of the process analyzed. To better understand the important role of simulation, we would introduce a fitting example: Human Judgment vs. Simulation.[14]

5.3.1 Human Judgment vs. Simulation

Human Judgment, also known as Seat Of The Pants Analysis (SOTPA), is probably the most widely used alternative to simulation. SOTPA is making decisions by instinct and feelings rather than using objective analytical tools. With SOTPA you never actually have to get out of your chair, or even spend any significant time to reach a decision.

"I use SOTPA all the time and you probably do too".

"When I am in a hurry to go out and want to know if I should wear a jacket or bring an umbrella, I might take a quick look out the window, reflect on the season and yesterday's weather, then make a decision."

That's SOTPA.

"I know that there is a high likelihood that I will be wrong, but I don't want to take the time to do the weather channel research to get more objective information."

Human judgment beat simulation in this case. But not always.

[14] Dave Sturrock (2009)

"When I am going on an all-day outside outing and have the same decision to make; the importance of being correct increases. In that situation I will take the time to consult at least two weather sources and even step outside for some direct research. With this objective analysis I can make a more informed decision."

Although such an analysis is never perfect, including objective data in my analysis dramatically increases the likelihood that my decision will be correct.

Now let's say I am a manager and my staff comes to me proposing purchase of a new piece of equipment to solve an important problem in my facility. They may give me technical specifications, maybe some manual or spreadsheet calculations, perhaps even show me a case study about how that equipment was used in another facility. The easy thing for me to do at that point is to make a SOTPA-based decision.

"After all, I must be pretty smart to get to be a manager, right? Right?? And I know "the big picture". So who better than me to make the decision? And why should I need more information?"

The problem is that my facility is a very complicated system.

"Why should I expect that I can predict the impact of adding this proposed equipment to my facility? I also don't have time to simulate."

I don't have time to research weather when the penalty of being wrong is low, but I make the time to do it when the penalties are higher.

Simulation beats human judgment when it matters.

We shall consider Figure 9 where a graph coming from the NetLogo interface is represented. In such graph we can see what the simulation drew and what the final results are.

On the x-axis we have the evolution overtime, on the y-axis the totals. As it can be seen from the legend on the top right of the figure, the red pen draws the total consumption, the blue pen the total income and the black pen the tradeoff between the two, the oil.

We have already said that the time is represented through the advancement of ticks in NetLogo. We stop the simulation every 2000 ticks in order to change some values among the variables. Thus, note that the difference between t_0 and t_1 is exactly 2000 ticks, say 2 years (same happen for t_5 and t_6).

Let us now explain what the effects are at each lag.

- t_0

  At t_0 we start with the values that are shown in Figure 8. We would see the effect of the individual consumption and production function, but , despite the number of turtles is set at 1 we know that NetLogo will give birth to one *fisherman* and one *ufisherman*. Individual consumption will be for each breed in this case.

  Note that the graph in Figure 9 depicts a consumption and a production that get through the origin. On the other hand, *oil* cuts the y-axis at an higher value because of the initial endowment of capital. As soon as ticks elapse, either *production* or *consumption* will step up and *oil* will show the balance between the two aforesaid variables. Consider that the values are the sum of both agents, hence totals.

  Values of our variables stay constant until t_1, where we set up different condition as follow.

- t_1

  What if the MPC happens to increase?

  It has been already considered that MPC is very determinant for the consumption function, we also stated that $0 \leq MPC \leq 1$ and if it

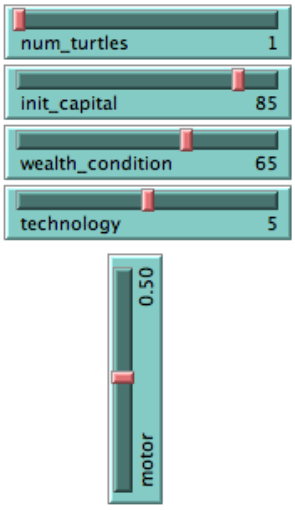

Figure 8: Starting values

happens to increase it will cause the consumption to increase as well.

We shall now see the response by setting MPC to 1. We set *motor*[15] at 1 and we run the simulation for 2000 ticks again. Not surprisingly, the graph shows that *consumption* tends to grow and, on the opposite, the *oil* falls. No change for the production that seems to be unaffected.

- t_2

  We set the *motor* at 0. For instance, think as we wanted to switch the motor off and we sailed just by exploiting the wind energy. In economics: we let the MPC to be zero. In such a case *consumption* should fall and hence *oil* has to rise as much as the income does. That is exactly what the graph shows.

- t_3

  What happen if we change the technology of production?

  Suppose now that we have a negative shock on technology factor.

[15] We set the *motor* slider vertically as it was the throttle control of our boat.

Hence, *technology* slider is set at zero, *production* should decrease and affect the *oil* negatively. *Consumption* here starts to increase because we reintroduce the default value of *motor* at 0.5 (with respect to the zero we had during t_2). Once again the graph does not disappoint our economic logic.

- t_4

  Here we have a positive shock on technology of production, thus, *technology* is set at 1. *Consumption* stays constant and *production* scores a significant increase; *oil* draws the trade off.

- t_5

  What if MPC is zero while positive shock on technology factor is affecting the market?

  On the one hand *consumption* has to decrease, on the other hand *production* should keep on increasing because of the shock and *oil*, in this case, will outline a sharp increment.

- t_6

  We conclude our simulation.

  We collect the data, say of the last 12 years of individual consumption, either of our *fisherman* or *ufisherman* and the production function of the former, being zero the one of the latter.

The model presented in this section would be a trivial example to better understand the NetLogo structure and its power.

Nevertheless, despite its simplicity, we have to assert that the results are not fully intuitive.

Through the interactions between agents and exogenous factors such as the non-dependence of the consumption function from disposable income

(autonomous consumption function), we can achieve the amount of consumption and gross domestic product (in a market with two agents) of the last 12 years.

Next section will deal with a more difficult simulation: we will aggregate the micro level in order to find out about the Micro-Macro relationship.

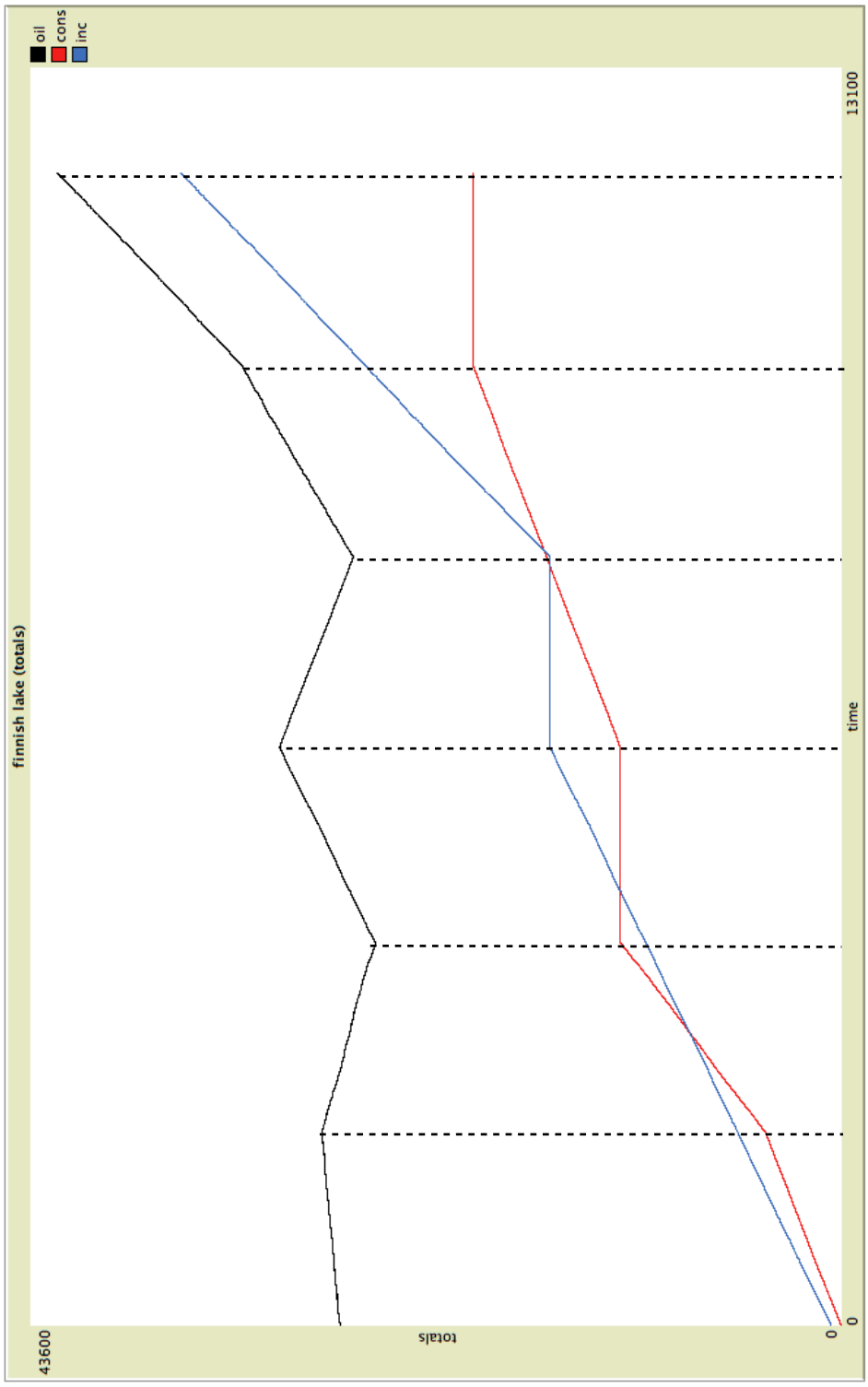

Figure 9: Graph

6 Simulation for Aggregated Production function

This chapter will focus on a new model that has been specially made to aggregate. For this reason, the complexity of the model will no longer be relatively simple, as the previous one, but at the same time, even excessively difficult.

The new model will still be an easy simplification of reality as well as the first one. It has some implementations on the structure of the production function that changes our individual and aggregate results. Furthermore, it has an additional part in the code that allows us to extract a dataset to be processed for our aggregation. 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 differences with the previous model, the main features and what are the agents' behaviors.

Secondly, we describe, as before, the line of 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.

6.1 Model overview with NetLogo

Figure 10 and Figure 11[16] show the NetLogo interface of the new model; it can be easily noticed that it has completely different features with respect to the previous one, especially concerning the sliders and graphs. The *world* monitor presents a different composition as well.

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.

[16] Figure 10 shows the interface at the beginning, after the setup; Figure 11 shows the interface after 200 ticks have past. Graphs and values in the monitors of Figure 11 are updated by the simulation with respect to the ones of Figure 10.

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. In the starting model, our fishermen were free of risk during the action of fishing: nothing could prevent them from reducing their production. Now, 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.

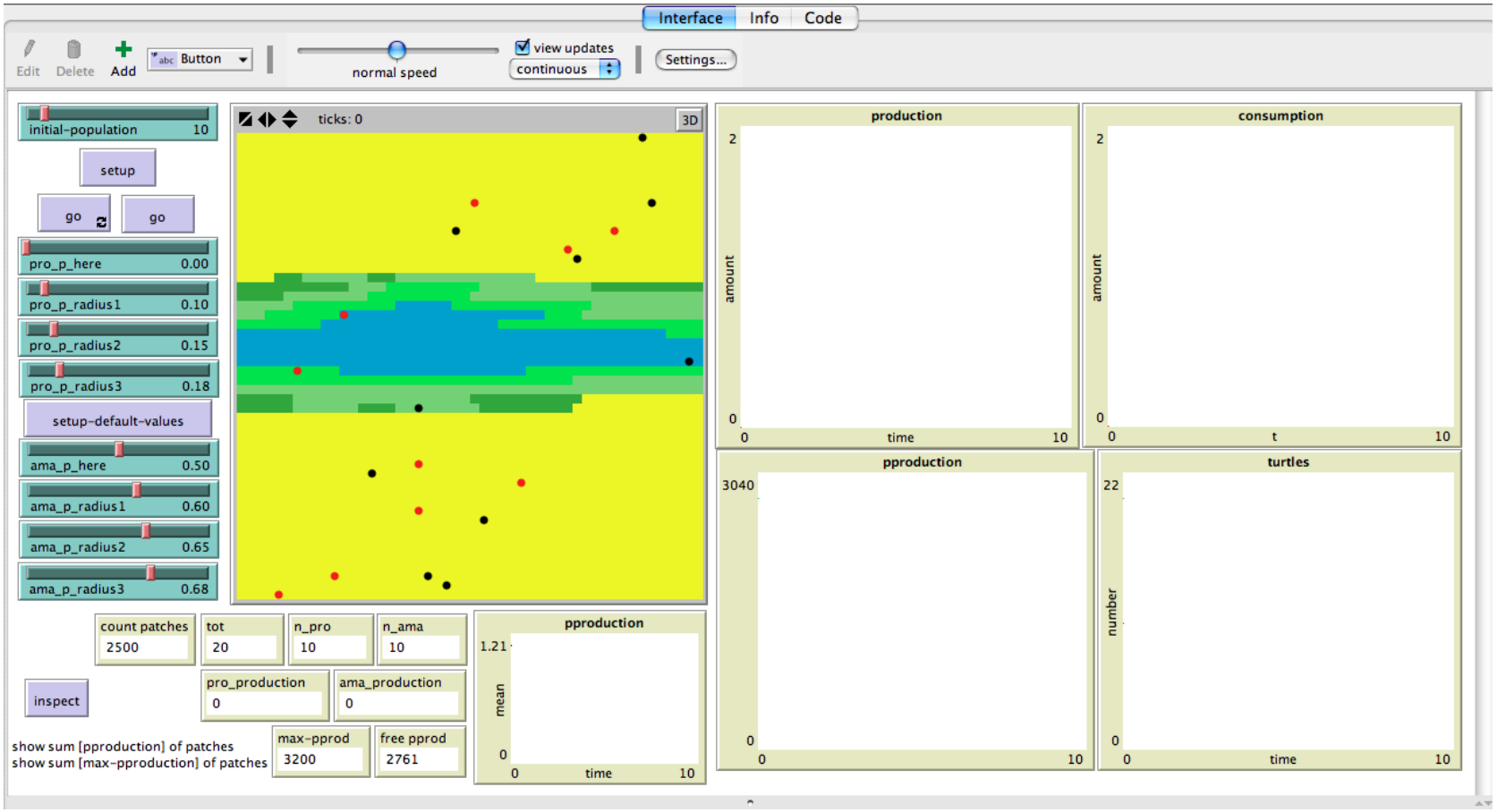

Figure 10: NetLogo interface (at setup)

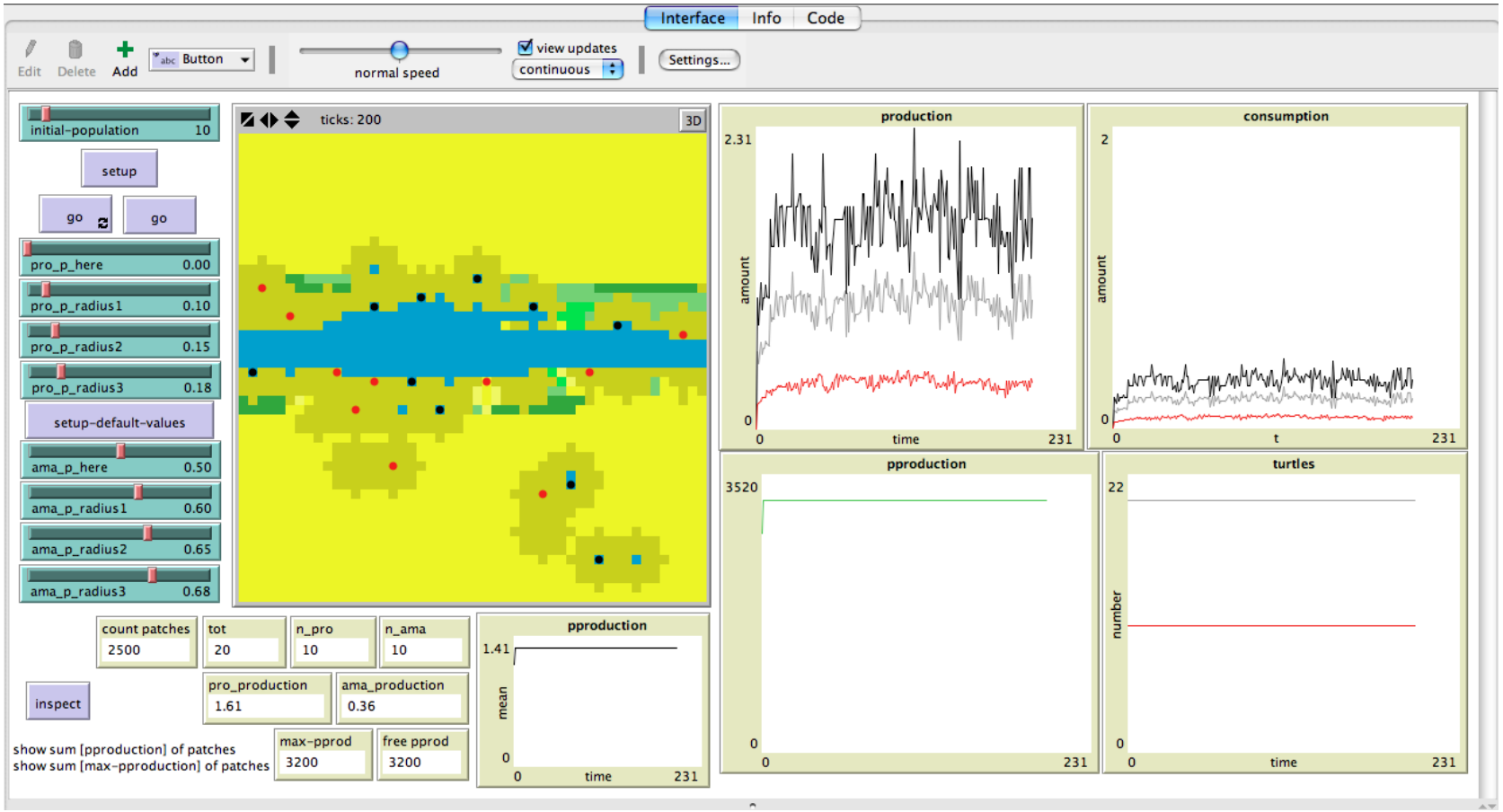

Figure 11: NetLogo interface (after 200 ticks)

6.2 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 that will be dark gray.

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 quan-

tity 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 re-

sources 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

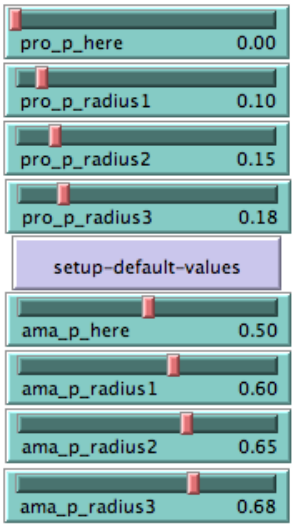

Figure 12: Technology of production parameters

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.

Further, as Figure 12 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.[17] 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.

[17]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*.

8.

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

  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.[18] Moreover,

[18]Remember that we have previously uploaded a "Map" where we defined the maxi-

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 associated either as a cost of producing (fix cost of exploiting resources) or the

mum amount of resources of every patch. Thus, there may be area with low resources exploitation and area with high resources exploitation.

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.[19]

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]
```

[19] *setup-default-values* button can be also seen in Figure 10

```
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 the next session "Aggregation with *R*". The following section "Extraction of dataset" will provide a better intuition on the importance of that passage.

6.3 Extraction of dataset

The model that has been created by NetLogo in this section, in addition to give an example of a more complicated framework compared with the previous model of the finnish lake, 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 instead. We create a model that, by running simulations, 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 the ABM approach.

Technically, point 15 of the previous section extracts the dataset created by the simulation. It saves 20 text files for both breed of agents having the production data of each agent over time.[20] 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.

In details, the files that have been created can be easily differentiated in the following way. The quotation marks bound the name of the file, for instance "pro5production.txt" will be the production of the agent *pro5*. 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*.

Figure 13 depicts how one of the text files looks like. It is a list of values, a time-series, describing what is the *pro5* records of production over time.

[20] *to extract-data* saves 10 files for *pro* production and 10 files for *ama* production. In total 20 text files.

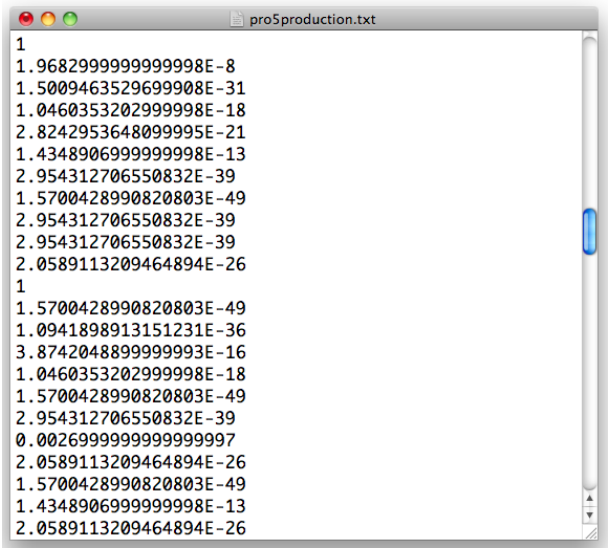

Figure 13: pro5production.txt

Every value is independent on each other and it is listed starting from time t_1 to t_{200}.

6.4 Aggregation with *R*

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 divide the session in two parts: the first will provide the R code with brief comments, the second will give some intuitions concerning the final results of aggregation that has been well explained by the R plots.

6.4.1 *R* code

In the following pages we want to 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 17 and Figure 18). 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 (Figure 14), its regression graph (Figure 15) and density plot (Figure 16).

```
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 individ-

ual 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 2 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)
```

6.4.2 *R* results and plots

We now attach the *R* outputs, starting with the *prod_average* matrix including the average of the production function for all agents (Figure 14). Note that the matrix contains 200 elements because, for sake of convenience, we decided to stop the simulation after 200 ticks have past.

Then, Figure 15 and Figure 16 draw the regression graph and the density plot of the above-mentioned *prod_average* respectively, outlining the deviations from the general mean.

Finally, Figure 17 and Figure 18 furnish details regarding the individual production function densities of every agent belonging to each breed (*pro* agents first and *ama* subsequently).

Representations in Figure 17 and Figure 18 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* we explained in Chapter 4. The former, was 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, was 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 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.

6.4.3 Simulations

Point "viii" of the above section (Section 6.4.1) provided a brief introduction of this section.

Table 2 furnishes values concerning the *prod_average* and then the relative *pro* and *ama* contributions to the general 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 soon compared with other two approaches outlining different results concerning the aggregation. The comparison will be covered in the next section.

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 2: Simulations

```
> prod_average
  [1] 0.5929555 0.4994500 0.6650806 0.5756285 0.6193400 0.6193728 0.6694637 0.6512688
  [9] 0.6392300 0.9459954 0.8277283 0.8530127 0.8458601 1.0542483 0.9091204 0.8659052
 [17] 1.0107728 1.0157832 1.0145202 0.8572922 0.9468786 1.0225037 0.9958603 1.0290664
 [25] 0.9958697 1.2025161 0.9592676 0.8945316 1.0078000 1.0225411 1.0326318 1.0524900
 [33] 0.8922718 1.0211500 1.0291453 0.8790193 1.0422724 1.0091205 1.0280721 0.8224530
 [41] 0.8924158 0.7595234 0.9422703 0.7856400 0.8480194 1.0122125 0.9493035 1.2158850
 [49] 1.0091225 0.9958601 0.8091271 0.8392648 0.8578910 0.7516004 0.8790625 0.8992936
 [57] 0.9873667 1.0121600 0.9595672 0.9908262 1.0055648 0.9977451 0.9872601 1.0555300
 [65] 0.7097467 1.0361348 0.7448371 1.0507653 0.8979616 1.0422701 0.8192702 0.8792331
 [73] 0.8795589 0.8801764 0.8990467 0.9415724 1.0989000 1.0926731 0.9259001 1.0076711
 [81] 0.8229879 1.0536740 1.0168490 0.9068314 0.9417593 1.2135547 1.0157542 1.0344100
 [89] 1.1079917 0.9055300 0.8912647 1.0559331 1.1422700 1.0876295 1.1993070 1.0292239
 [97] 0.9995403 1.0989000 1.0744230 0.9807282 0.9854359 0.9048635 1.0369438 0.8791412
[105] 1.1490464 1.1425371 0.9055339 0.8987506 1.0191931 0.9319000 0.8067017 0.9157568
[113] 1.0923079 1.3493032 1.0558858 1.1410408 0.9555341 1.0460740 0.9990357 1.0489028
[121] 1.1929104 1.1206593 1.0557415 1.0187928 0.9344100 1.0344448 0.9494328 1.0555406
[129] 1.0930243 1.0094299 1.0107830 1.1278148 0.7631779 1.1467400 1.0791452 0.9791310
[137] 1.0725992 1.1027591 0.8622753 0.8163260 1.0422729 0.9723800 0.9078671 0.9830334
[145] 0.8590204 0.6790430 1.2790168 0.6791582 0.9927659 0.9162912 0.9716900 0.9900638
[153] 1.1723800 1.0792762 0.9856440 1.1735470 1.0126402 1.1310102 0.8725150 1.1291286
[161] 0.9859958 0.9924158 0.9856401 1.1543520 0.9791558 0.9713624 0.9910400 0.9922700
[169] 0.9017500 0.9458948 0.8591200 1.0012707 0.8723802 1.0326008 0.7576858 1.0031851
[177] 0.8589700 0.9078900 0.8524992 0.8216368 0.9376404 0.8821088 0.7863031 0.9030302
[185] 0.6864452 0.9225258 1.1145200 0.8247128 0.9659624 1.1791412 0.9791650 1.1790102
[193] 0.8284443 0.8658982 0.9856511 0.8477291 1.1856400 1.1601257 0.8477060 0.9844100
```

Figure 14: *prod_average* matrix

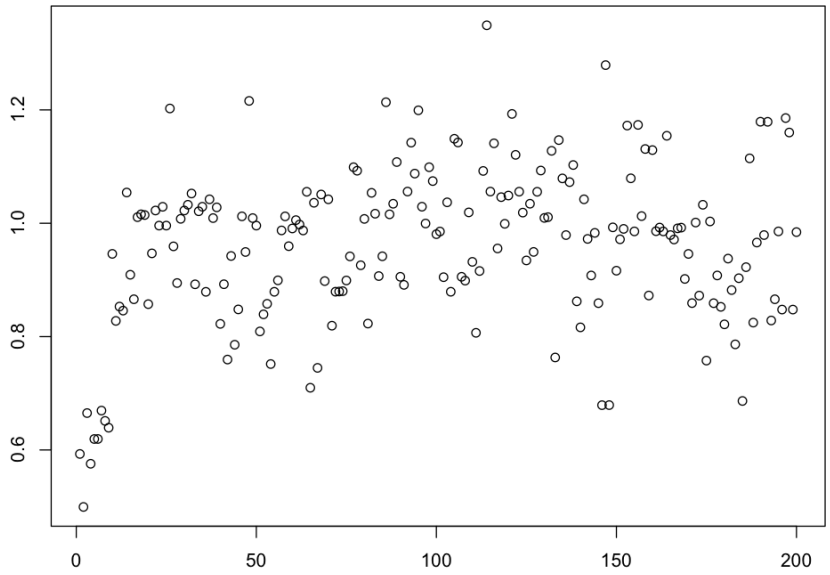

Figure 15: *prod_average* regression plot

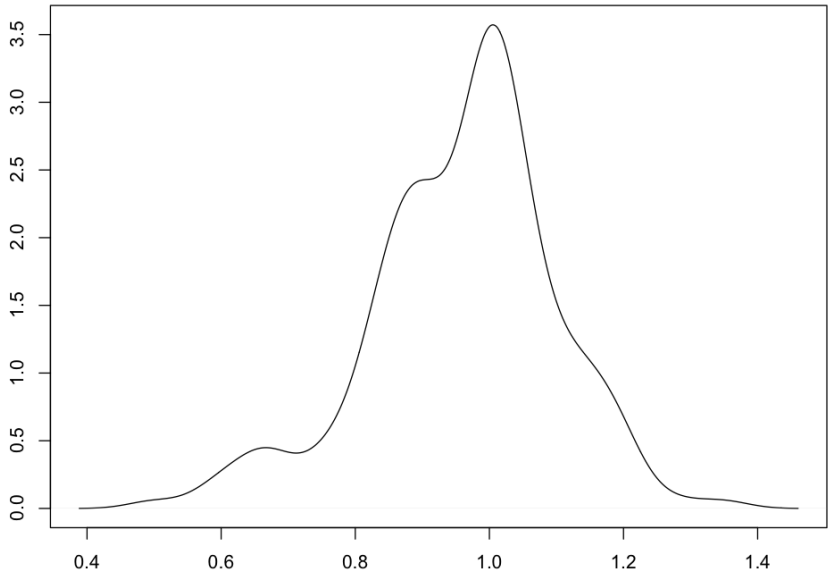

Figure 16: *prod_average* density plot

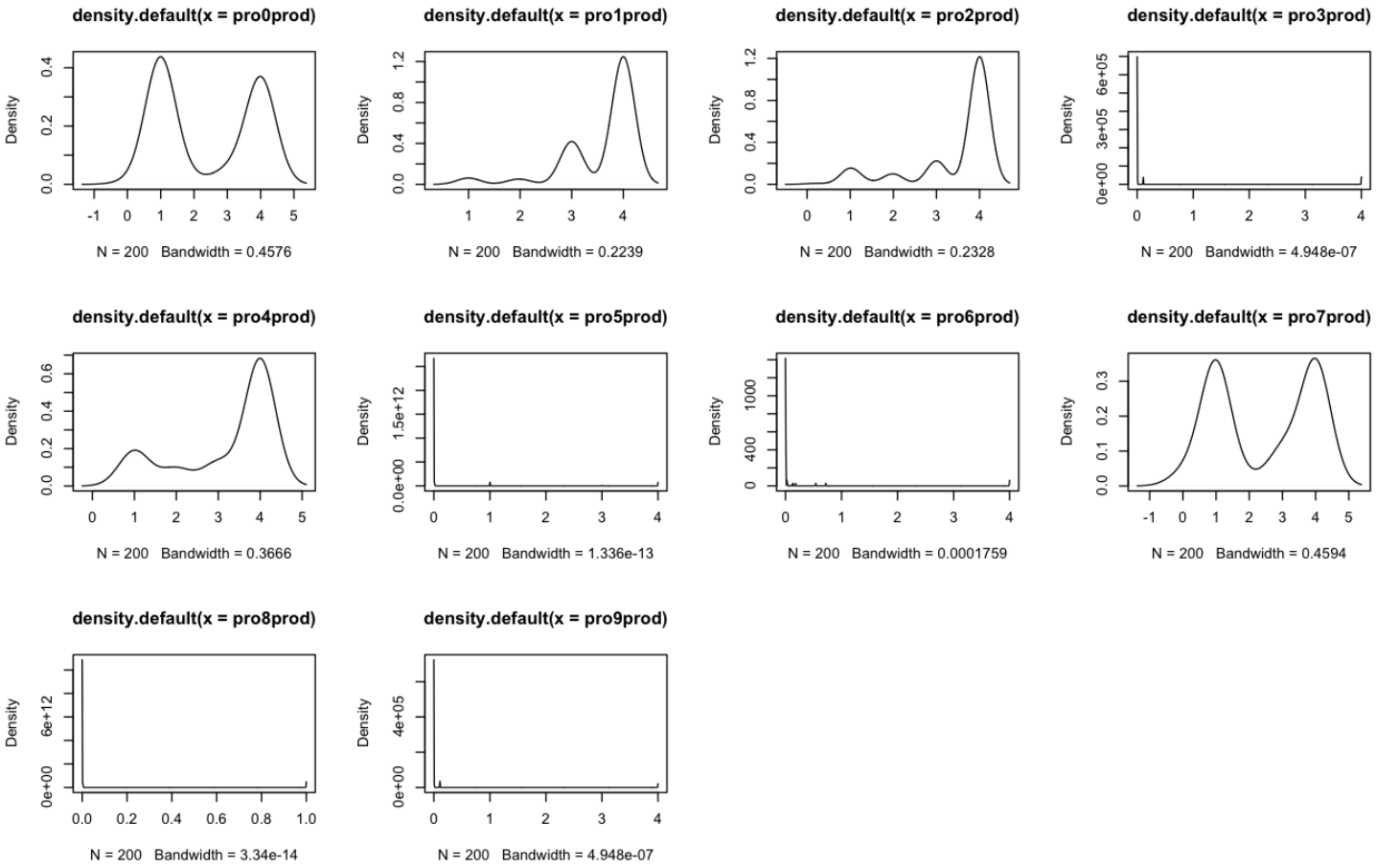

Figure 17: *pro* production densities

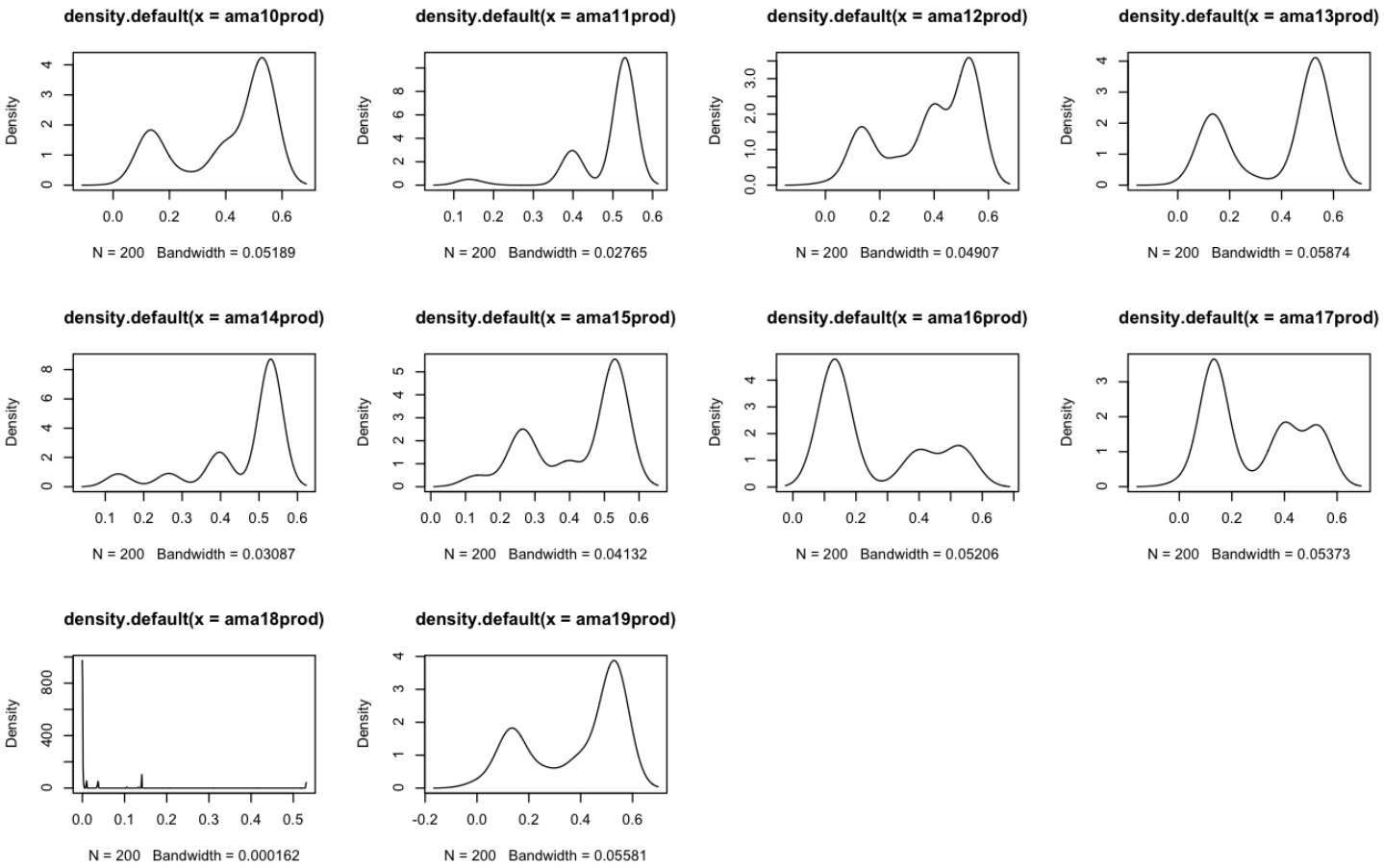

Figure 18: *ama* production densities

6.5 Comparisons

This is the most relevant section of the chapter where the discussion on the 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 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 3 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.

The following paragraphs will serve to motivate how these values have been obtained and, comparing them with each other, to conclude by interpreting what is the most appropriate way to aggregate and which one reflects an economic scenario more in depth.

	Plain Model	*Micro Simulation Model*	*Agent Based Model*
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 3: Models for Aggregation

6.5.1 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.

6.5.2 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 for

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.

6.5.3 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.

6.6 Conclusions

This chapter furnish a complete overview of an ABM used to aggregate.

We first build up an agent based model to extract a dataset concerning individuals production functions with NetLogo.

Secondly, we import such dataset into R, we aggregate, we calculate the average and we comment the final results.

Thirdly, we see what happen if we run others simulations having the same parameter values.

Then, we calculate the average of production function among such simulations.

At the end, we compare different ways to find out such average of production function.

The most relevant part is thus the last. Three tools to estimate the individual production functions through their general average are considered. The analysis starts from the individual level but moves to the aggregated level in no time. We name such tools "*Plain Model, Micro Simulation Model*" and "*ABM*" and we discuss their distinguishing features.

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 to reality? We call it "closer" because unfortunately it remains a simplification of what happens in an economic environment as a whole.

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.

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

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

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 3 (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 (2008) 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 in equation (3.3.37) of Chapter 3, coming from Vartia analysis (i.e. the aggregation error).

Moreover, another clarification needs to be done: we have been dealing with the average production values so far, but, since our *World* is composed by 20 agents, the aggregation results are really straightforward. In others words, the average production values represent per capita values of aggregated production function. For this motivation readers should think to the aggregate level as a sum of the average production values out of population (in our example 20 agents).

7 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.

7.1 Unified Modeling Language (UML)

Unified Modeling Language (UML)[2] 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.

7.1.1 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..."[2]

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 19 depicts a Class diagram of our simulation model for aggregated production function described in Chapter 6. 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 20 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 20 to the class diagram in Figure 19. 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.

7.1.2 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*. Inter-

actions 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 21 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.

7.1.3 State diagram

State diagram or *state-transition diagram* is designed in Figure 22. 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.

7.1.4 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 23 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.

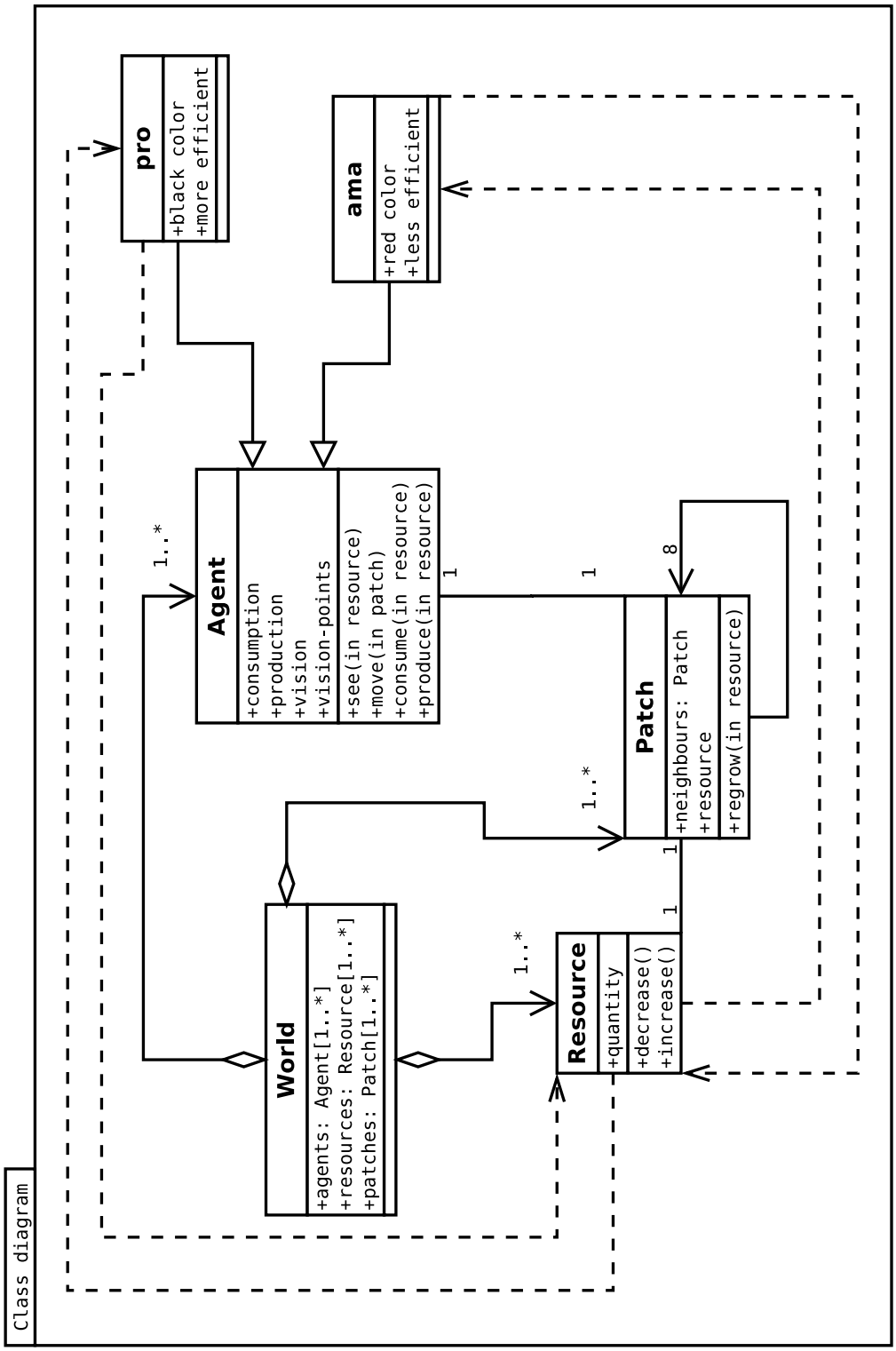

Figure 19: Class diagram

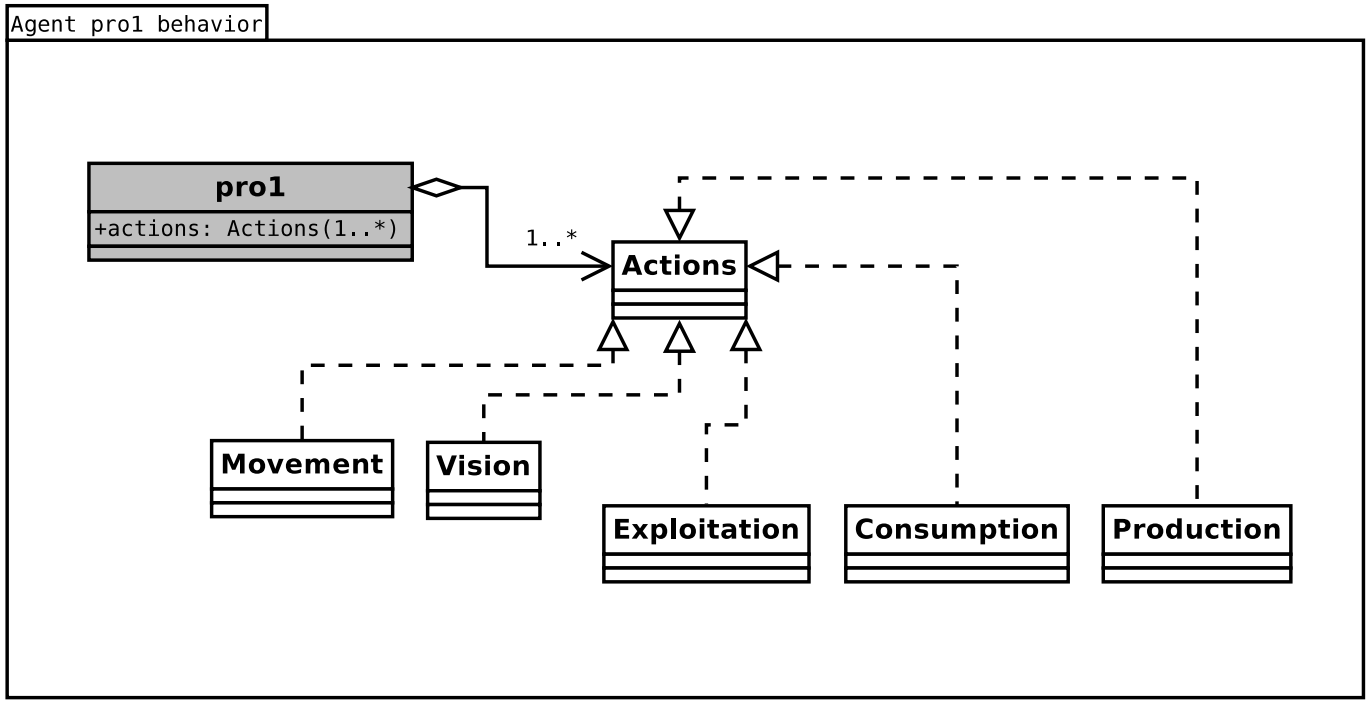

Figure 20: Behavior class

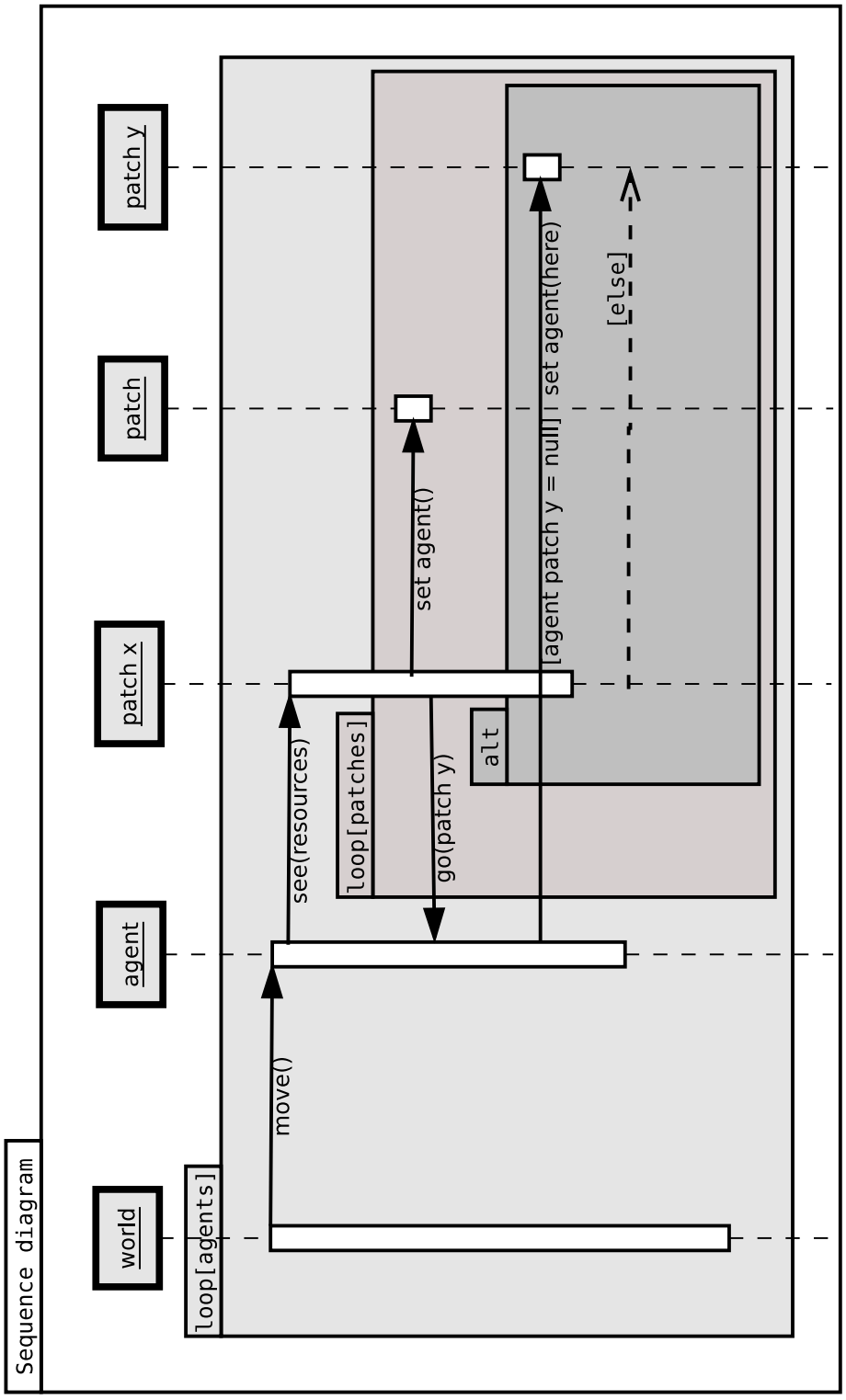

Figure 21: Sequence diagram

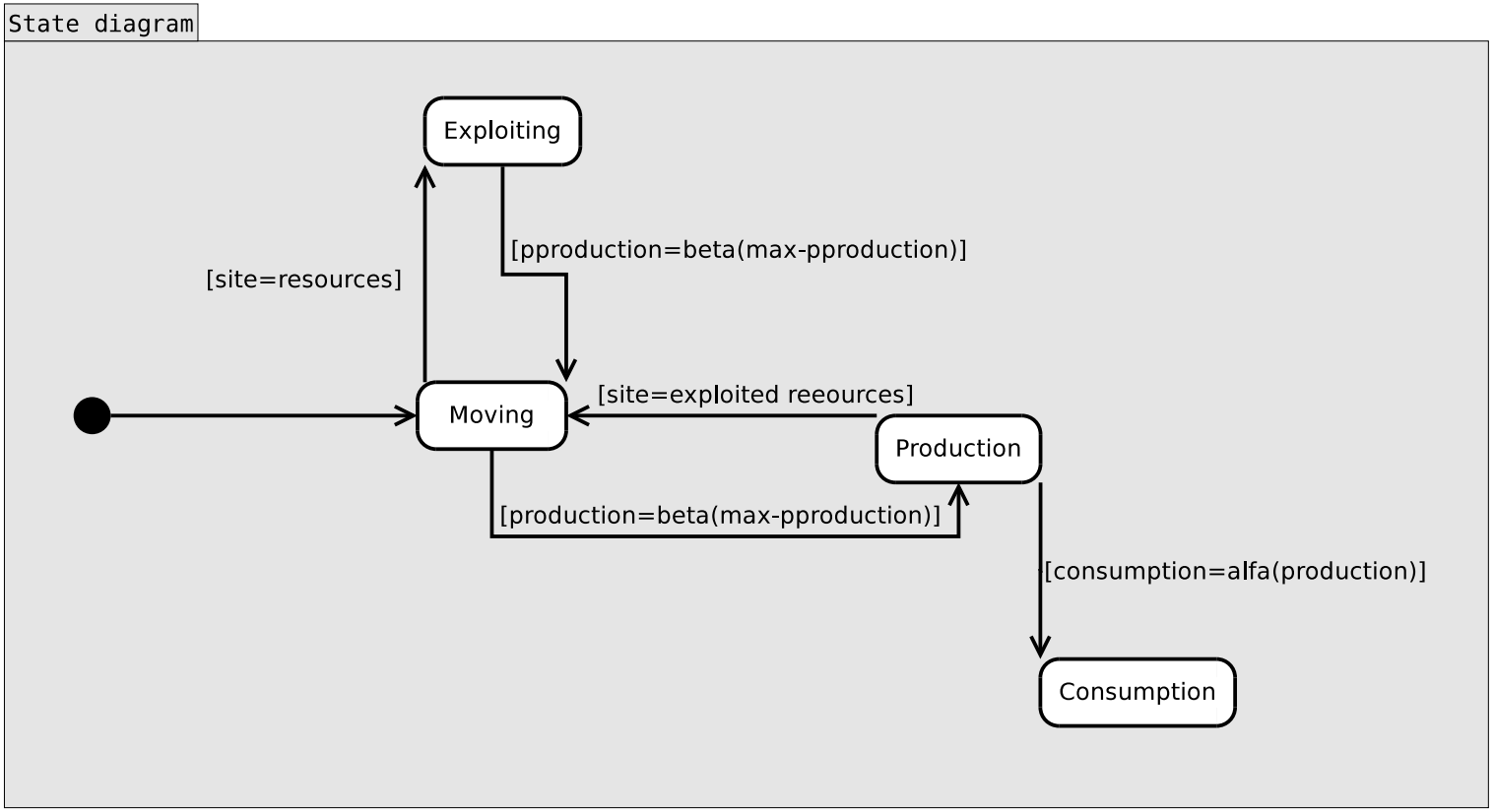

Figure 22: State diagram

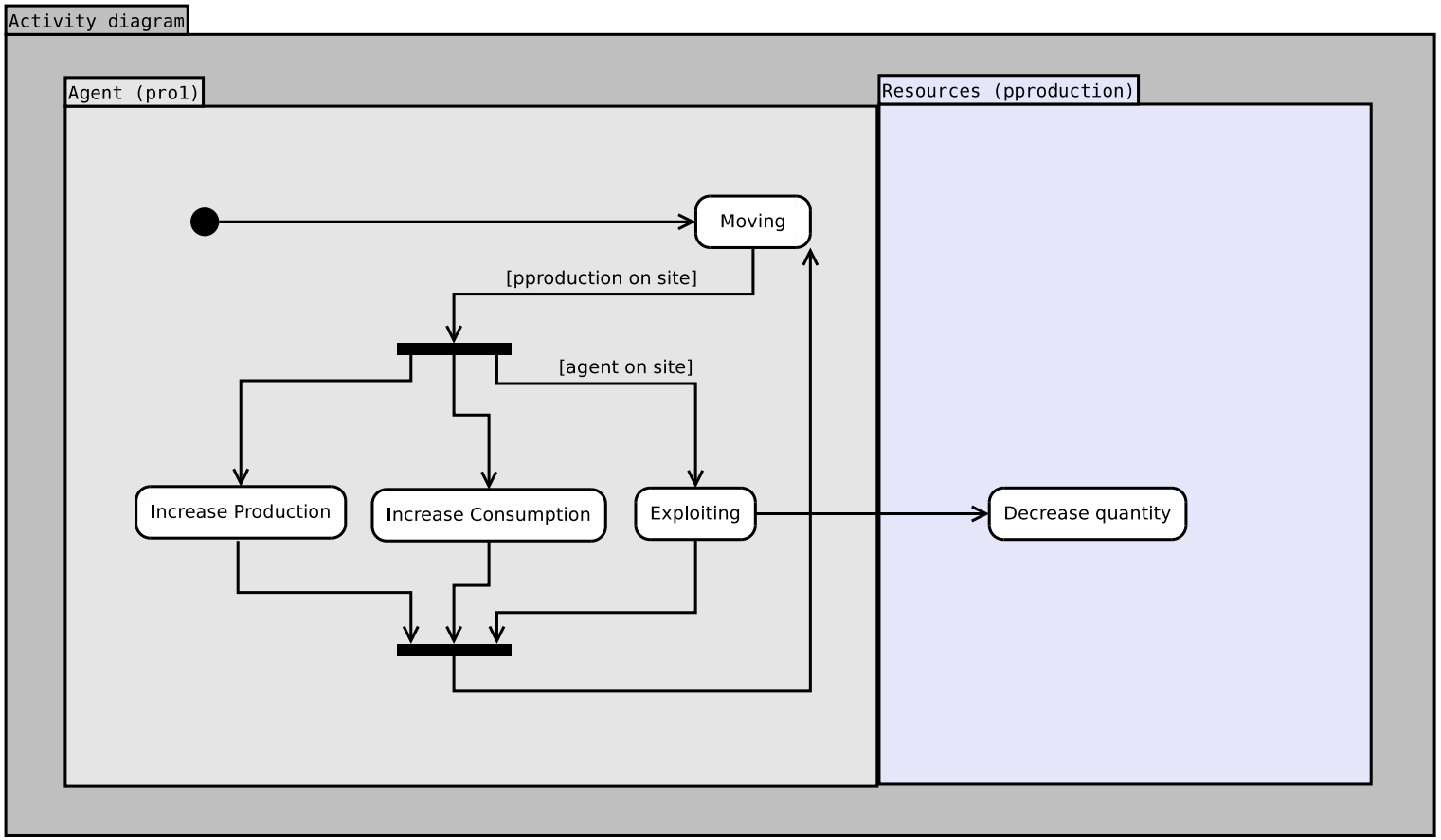

Figure 23: Activity diagram

7.2 ODD Standard Protocol

"...there is no standard protocol for describing such simulation models, which can make them difficult to understand and to duplicate."[7]

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.

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; 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"[8], which stands for the three blocks of elements "Overview", "Design concepts", and "Details" since the names of some elements have been changed.

Figure 24 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

7.2.1 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.

7.2.2 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.

7.2.3 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 23 covered in the previous section.

7.2.4 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 the 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 the 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.

7.2.5 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.

7.2.6 Input data

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

7.2.7 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*	**1**	*Purpose*
	2	*State variables and scales*	**2**	*Entities, state variables and scales*
	3	*Process overview and scheduling*	**3**	*Process overview and scheduling*
Design concepts	**4**	*Design concepts*	**4**	*Design concepts*
		Emergence		*Basic principles*
		Adaptation		*Emergence*
		Fitness		*Adaptation*
				Objectives
		Prediction		*Learning*
		Sensing		*Prediction*
		Interaction		*Sensing*
		Stochasticity		*Interaction*
		Collectives		*Stochasticity*
		Observation		*Collectives*
				Observation
Details	**5**	*Initialization*	**5**	*Initialization*
	6	*Input*	**6**	*Input data*
	7	*Submodels*	**7**	*Submodels*

Figure 24: ODD protocol

8 General conclusions

The work begins with an interesting review of some basic economic concepts underlyed

by many economic theories. Then, it describes rigorously Vartia (2008) on the aggregation and his application to the example of the marginal propensity to consume.

We proceed with the description of agent-based world, many different interpretations and ways of dealing with the agent-based tool are defined.

We come to understand how is possible to create an agent-based economic model, through an example, using NetLogo.

Just figured out how to do, we build the model that will allow us to locate the individual functions of production, once again with NetLogo. 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. NetLogo and *R* define, at this point, one of the three methods we adopted to aggregate, it covers the agent-based method.

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. This is undertaken through the use of two techniques of description: UML language and ODD standard protocol.

The comparison between the classical procedures and agent-based method reveals that Vartia (2008) 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.

A Appendix

Key concepts of Vartia's theory

Individual non-linear consumption function

$$c(a) = f^a(x(a)) \tag{A.0.1}$$

Average consumption per household (macro output in terms of micro input)

$$\langle c \rangle = \frac{1}{n} \sum_{\Omega} c(a) \tag{A.0.2}$$

MicMac-function

$$\langle c \rangle = \frac{1}{n} \sum_{\Omega} f^a(x(a)) \tag{A.0.3}$$

MicMac-function (after the mean value theorem)

$$\langle \Delta c \rangle = \frac{1}{n} \sum_{\Omega} \sum_{k=1}^{K} \overline{m}_k(a) \Delta x_k(a) = \sum_{k=1}^{K} \langle \overline{m}_k \Delta x_k \rangle \tag{A.0.4}$$

Inner product (MicMac-function with only one explanatory variable)

$$\langle \Delta c \rangle = \langle \overline{m} \Delta x \rangle \tag{A.0.5}$$

MC is Uniform absolute changes

$$MC^{UA} = \langle \overline{m} \rangle \tag{A.0.6}$$

MC is Uniform proportional changes

$$MC^{UP} = \langle \overline{m} \rangle_{x^0} \tag{A.0.7}$$

Decomposition into Uniform Absolute changes and Uniform Proportional changes

$$\Delta x(a) = \Delta x^{UA}(a) + \Delta x^{UP}(a) + u(a) \tag{A.0.8}$$

Decomposition at macro level

$$\langle \Delta x \rangle = \langle \Delta x^{UA} \rangle + \langle \Delta x^{UP} \rangle \tag{A.0.9}$$

Macro effective marginal coefficient MC

$$MC = MC^{UP} \cdot w^{UP} + MC^{UA} \cdot (1 - w^{UP}) \tag{A.0.10}$$

Macro equation

$$\langle c \rangle = MC \cdot \langle \Delta x \rangle + cov(\overline{m}, u) \tag{A.0.11}$$

A Appendix

Inner product's properties

The inner product or scalar product[12]

$$\langle xy\rangle = \langle (x(a))(y(a))\rangle = \frac{1}{n}\sum_{\Omega}(x(a))(y(a)) \tag{A.0.12}$$

of two n-vector x and y has the defining properties:

i) if and only if $x = 0$

$$\langle xx\rangle > 0 \tag{A.0.13}$$

$$\langle xx\rangle = 0 \tag{A.0.14}$$

ii)

$$\langle xy\rangle = \langle yx\rangle \tag{A.0.15}$$

iii)

$$\langle (\lambda x)y\rangle = \langle x(\lambda y)\rangle = \lambda\langle xy\rangle \tag{A.0.16}$$

iv)

$$\langle x(y+z)\rangle = \langle xy\rangle + \langle xz\rangle \tag{A.0.17}$$

B Appendix

Basic Lemma of Aggregation BLA

The Basic Lemma of Aggregation[21] is referred to the following identity:

$$\sum_{i=1}^{n} w_i x_i y_i = \overline{xy} + cov(x, y) \tag{B.0.18}$$

where

$$\overline{x} = \sum_{i=1}^{n} w_i x_i \tag{B.0.19}$$

$$\overline{y} = \sum_{i=1}^{n} w_i y_i \tag{B.0.20}$$

$$cov(x, y) = \sum_{i=1}^{n} w_i (w_i - \overline{x})(y_I - \overline{y}) \tag{B.0.21}$$

$$\sum_{i=1}^{n} w_i = 1 \tag{B.0.22}$$

$$\forall x, y, w \in R^n : \langle w \rangle \neq 0 \tag{B.0.23}$$

and x_I and y_i can be either arbitrary number or any variables.

This transforms an average of product to a product of averages plus a correction. Weights do not need to be positive or even non-negative; in the case where we have a negative weights, averages are affine combinations.

PROOF:

$$cov(x,y) = \sum_{i=1}^{n} w_i(w_i - \overline{x})(y_I - \overline{y}) \tag{B.0.24}$$

$$= \sum_{i=1}^{n} w_i(x_i y_i - x_i\overline{y} - \overline{x}y_i + \overline{xy}) \tag{B.0.25}$$

$$= \sum_{i=1}^{n} w_i x_i y_i - \overline{y}\sum_{i=1}^{n} w_i x_i - \overline{x}\sum_{i=1}^{n} w_i y_i + \overline{xy}\sum_{i=1}^{n} w_i \tag{B.0.26}$$

$$= \sum_{i=1}^{n} w_i x_i y_i - \overline{yx} - \overline{xy} + \overline{xy} \tag{B.0.27}$$

$$= \sum_{i=1}^{n} w_i x_i y_i - \overline{xy} \tag{B.0.28}$$

By plugging the last result into the initial identity, it is equivalent to BLA. QED.

C Appendix

NetLogo code of Chapter 5 model

```
 breed [fishermen fisherman]
 breed [ufishermen ufisherman]

fishermen-own [
  init.capital consumption oil income
  ]
ufishermen-own [
  init.capital consumption oil income
  ]



to setup-default-values
  set num_turtles 10
  set init_capital 85
  set wealth_condition 65
  set technology 5
  set motor 0.5
end



to setup
  ca
  setup-fishermen
  setup-ufishermen
```

```
  setup-world
  reset-ticks
end


to go
   ask fishermen [
     sail
     trawl
     update.values
     ]
   ask ufishermen [
     sail
     update.values
     ]
   tick
end


to setup-fishermen
  create-fishermen num_turtles [
    set shape "boat"
    set size 3
    values
  ]
end
```

```
to setup-ufishermen
  create-ufishermen num_turtles [
    set shape "uboat"
    set size 3
    values
  ]
end


to setup-world
  set-patch-size 14

  ask patches with [
    pxcor > -16 and pxcor < 16 and
    pycor < 16 and pycor > -16
    ]  [set pcolor one-of [sky blue]]

  ask patches with [
    pxcor = -16
    ]  [set pcolor grey]

  ask patches with [
    pxcor > -16 and pycor > 15
    ]  [set pcolor grey]

  ask patches with [
    pxcor = 16
    ]  [set pcolor grey]
```

```
  ask patches with [
    pxcor > -16 and pycor < -15
    ]  [set pcolor grey]
end


to values
  set init.capital 0
  set init.capital
                  init.capital
                 + random-float 1000 * init_capital
                 + random-float 10 * wealth_condition
  set income
            income + 0.02 * init.capital
  set consumption 0
  set oil
         oil + income + init.capital
end


to sail
  if pcolor = grey
    [set heading
                heading + random-float 180
      ]

  ;if oil > 0 [
```

```
    fd 1
    set consumption
                  consumption + random-float 4 * motor
  ;]
end


to trawl
  set income
            income + random-float technology
end


to update.values
  set oil
         income - consumption
end
```

D Appendix

NetLogo code of Chapter 6 model

```
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 ]


to setup
  ca
  set my-date-and-time date-and-time
  setup-patches
```

```
  setup-pro
  setup-ama
  reset-ticks
end


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


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


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


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


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


to go
  if not any? turtles [
    stop
  ]

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

  ask patches [
    patch-recolor
    patch-growback
  ]

  extract-data
```

```
  tick
end


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


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


to patch-growback
   set pproduction max-pproduction
end


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


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


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


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
```

E Appendix

R code

```
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]


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]

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))


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

plot(pro_prod_average)


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

plot(ama_prod_average)


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))


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

plot(prod)
plot(density(prod))


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

References

[1] Axtell, R. (2000): *Why agents?: on the varied motivations for agent computing in the social sciences*, Center on Social and Economic Dynamics.

[2] Bersini, H. (2012): *UML for ABM* , Journal of Artificial Societies and Social Simulation.

[3] Burbidge, J. and Cuff, K. (2005): *Capital tax competition and returns to scale*, Regional Science and Urban Economics.

[4] Friedman, M. (1957): *The permanent income hypothesis*, Princeton University Press.

[5] Gilbert, N. (2008): *Agent-Based Models*, Sage Publications, London.

[6] Godley, W. (1999): *Money and credit in a Keynesian model of income determination*, Cambridge Journal of Economics.

[7] Grimm, V. and Berger, U. and Bastiansen, F. and Eliassen, S. and Ginot, V. and Giske, J. and Goss-Custard, J. and Grand, T. and Heinz, S.K. and Huse, G. and others (2006): *A standard protocol for describing individual-based and agent-based models*, Ecological modelling Journal.

[8] Grimm, V. and Berger, U. and DeAngelis, D.L. and Polhill, J.G. and Giske, J. and Railsback, S.F. (2010): *The ODD protocol: A review and first update*.

[9] Hart, N. (1996): *Returns to scale and Marshallian economics*.

[10] Keynes, J.M. (1937): *The General Theory of Employment, Interest, and Money*, The Quarterly Journal of Economics.

[11] Kirman, A.P. (1992): *Whom or what does the representative individual represent?*, The Journal of Economic Perspectives.

[12] Kolmogorov, A.N. and Fomin, S.V. (1970): *Introductory real analysis*.

[13] Liebrand, W.B.G. and Nowak, A. and Hegselmann, R. (1998): *Computer modelling and the analysis of complex human behavior: retrospect and prospect*, in *Computer Modeling of Social Processes* (eds W.B.F. Liebrand, A. Nowak, and R. Hegselmann), UCL Press, London, pp. 1–14.

[14] Modigliani, F. (1966): *The life cycle hypothesis of saving, the demand for wealth and the supply of capital.*

[15] Ragin, C.C. (1989): *The comparative method: Moving beyond qualitative and quantitative strategies*, University of California Press, Berkeley.

[16] Schorfheide, F. (2011): *Estimation and evaluation of DSGE models: progress and challenges* , National Bureau of Economic Research.

[17] Schumacher, E.F. and McKibben, B. (2010): *Small is beautiful: Economics as if people mattered.*

[18] Simon, H. (1969): *The Sciences of the Artificial*, The MIT Press, Cambridge, MA.

[19] Squazzoni, F. (2012): *Agent-Based Computational Sociology.*

[20] Tobin, J. (1965): *Money and economic growth*, Econometrica: Journal of the Econometric Society.

[21] Vartia, Y. (2008): *On the Aggregation of Quadratic Micro Equations.*

[22] Vartia, Y. (2008a): *Integration of Micro and Macro Explanations in Economics: The Case of Marginal Propensity to Consume.*

[23] Vartia, Y. (2009): *Whole and its Parts: Micro Foundations of Macro Behaviour.*

[24] Vartia, Y. and Suopera, A. (2011): *Analysis and Synthesis of Wage Determination in Heterogeneous Cross-sections.*

[25] Willer, D. and Webster, M. (1970): *Theoretical concepts and observables*, American Sociological Review, 35, 748–757.