UNIVERSITY OF TURIN

DEPARTMENT OF PHYSICS

MASTER DEGREE IN PHYSICS OF COMPLEX SYSTEMS

An evolving complex network in a stock market

Matteo Bertino

Supervisor: Prof. Pietro Terna

Examiner: Prof. Michele Caselle

A.A. 2014/2015

Contents

Abstract

A financial market is a very important example of complex system in modern science and standard financial theory does not offer effective models to analyze the full complexity of such world. Implementing an ABM enables to represent heterogeneity of agents and their irrational behavior. Anyway, beside the intention to monitor the dynamics of micro variables (as investors profits) and macro variables (as stock prices), we are also interested in discovering the properties of interaction among all agents. This aim can be achieved with an ABM which brings out a financial (temporal) network in the system, a new independent tool that evolves with the simulation and incorporates many additional information. In particular we are dealing with an informational network, in the sense that a single link expresses information sharing, between two agents, about the securities in the market. So the idea is that agents can interact and make decisions through a series of social interactions: the result is a realistic and imperfect learning process. Combining all these features with the real trading system of double auction market we can build an ABM which explores many complex phenomena in a stock market. All the simulation data regarding market itself and the agents can be integrated with mathematical and statistical properties of the emerging informational network, exploiting both Network Analysis and Agent-Based Modeling. In addition, this work presents real applications for trading operations because we can introduce fictitious agents, the arbitrageurs, able to align the simulated stock prices to real ones. The simulation so reflects the real market and you can test the actual performance of every trading strategy, discarding those who have bad performed and using those which guarantees significant mean profits. This makes such ABM an effective tool both for theoretical purposes and for real trading.

Acknowledgements

I would like to express my deep gratitude to Professor Pietro Terna, my supervisor, for his patient guidance, wise advices and useful critiques for this work. I want also to say a big thank-you to my examiner, Professor Michele Caselle, who was an essential reference point for my Master Degree path in Physics of Complex Systems.

Finally, I wish to thank my family and my girlfriend Valeria for their support and encouragement throughout my study.

1 Introduction

The fragility of the equilibrium models that we consider in finance and macroeconomics require new ways to investigate real complexity in economic systems. ABMs are one of the most promising tools to study such complex emerging patterns. Only being able to model heterogeneous agents and the interactions among them, we can understand how the nature of the system on micro scale can affect the dynamics of the important aggregate variables on the macro scale. In particular, in a stock market the role of investors expectations about future asset prices and about expectations of the other agents, is the drive of many irrational macro phenomenona such bubbles or crashes, which cannot be explained through a deterministic view of the market. Investors can have different information sets (*informational asymmetry*) and available information is processed in different ways; indeed agents can use different trading strategies according to different forecasting econometric models, or simply according to different irrational (and personal) moods. So, starting from an ideal environment in which investors share the same expectational model to compute their asset demands, we have to explore dynamic models in which heterogeneous expectations play their fundamental role, producing many of the real stock market features. In this sense the flexibility and computational power of ABMs are essential to create many models that look towards this direction. We are able to include several kinds of agents in the market, giving all of them different behavior rules and, if necessary, the capability to process information and learn from emerging data. This is equivalent to include agents who can elaborate information not in a static way, but performing new strategies in response to the current market situation, in what we can define as a continuos *evolutionary* learning process. In particular in our work we shall consider two main types of agents: the consultants, who are financial advisors who process data and make forecasts about future stock prices, and the investors, who operate on the market and can use the indications of a consultant to define their own strategy.

The existence of heterogeneous agents who interact and process information in different ways, even changing their own behavior rules through time, is an important basis to construct a good ABM. However we could be interested in exploring the intrinsic nature of all interactions among investors, linking it to the emerging market dynamics. From this point of view we are going to analyze the network emerging from the dynamics of our ABM, which is an informational network and not a trade network. The agent-based model (which aim to reproduce in the best way the real market) will evolve normally through time and, every time an interaction is established between two different agents, a link is created in the network, with the two agents involved becoming nodes of the network itself. Such approach is very powerful because the network which emerges is the direct result of the dynamics on the micro-scale among agents, and therefore is more realistic than a network model which wants to mimic a priori some features of the system. We can say that the network arises naturally from the system dynamics and this means that it incorporates all relevant complex patterns included in the agent-based representation. Moreover this method goes beyond the issue of making realistic network models in which network structure varies over time. The emerging network is a *temporal network*, where its links appear and disappear in relation with the *non-equilibrium state* of the system. In a financial market we are indeed not interested in static network analysis, but we want to explore how market dynamics can shape the network of social interactions among traders. So this kind of network can be seen as an evolving complex object which photographs, tick by tick in our simulation, the nature of the market. It is possible to compute several network metrics measures to discover some properties of the system which could not be inferred only with an ABM. Putting together simulation data relative to aggregate variables (in our case stock prices and profits of investors) with dynamic measures for the evolving network, we are able to monitor and study many complex properties of the market, both on micro and macro scale.

In order to better present the theoretical fundamentals for such work, we include in Chapter 2 some

interesting references about past works in the field of Agent-Based Models for stock markets. We shall introduce many theoretical concepts which are commonly discussed in this context and that we want to develop through our model. Moreover this chapter contains the reasons for joining ABMs with Network Analysis, explaining the advantages of such new challenging perspective.

The market model is presented in Chapter 3, with the investment plan for investors and the pricing models used by our consultant. We also add many transition mechanisms (between the two main classes of traders) that regard behavioral finance phenomena. The ABM has anyway to include the rules for the initialization and evolution algorithm of the informational network: the latter is reported as the final section of model description. The chapter ends resuming and highlighting the novelties of this ABM market with respect to the past ones, with a great focus on the role of informational network.

In Chapter 4 we shall present the programming framework used to build this ABM, which is SLAPP. The protocol of this framework is explained in detailed way since it can be useful for other ABMs in many other research fields.

Chapter 5 presents the structure of the program for this ABM. All the methods are briefly described with their role in terms of model dynamics. In particular we shall discuss how the control parameters can be set and which is the time generation system within the simulation.

The first outputs of the simulation are shown in Chapter 6. Here we also include a complete analysis of qualitative effect of all control parameters. Indeed in ABM with so many behavioral and learning phenomena, it is firstly necessary to understand the role of every parameter and, if possible, to make a sort of calibration for some of them.

This work walks towards two different directions. The first one is the construction of a new ABM market to stimulate the theoretical discussion of many important topics regarding complexity in financial markets; Chapter 6 and part of Chapter 8 has this goal. The second is instead related to use the ABM as a tool to test the performance of specific trading strategies. Chapter 7 therefore defines a method to keep aligned the simulated prices to real ones, such that the simulation reflects a real market.

The actual quantitative analysis of simulation data is reported in Chapter 8. We shall study the statistical properties of stock prices dynamics, checking if the simulation can present some features of real financial markets; in this sense we are use the ABM in the theoretical perspective of complex financial systems. Next we implement what we have made in Chapter 7 to compare some interesting agents properties (profits of traders and forecasting performance of consultants) in the totally simulated market to those in the market aligned to real data. This is very interesting because we can analyze the difference between a complex, but simulated, environment and a complex real market.

2 Theoretical framework and literature references for ABM markets

2.1 Some past and recent ABMs for stock market: strengths and weaknesses

The use of ABMs to study financial markets is a new and growing field. The idea to left an analytically formalization of market dynamics is the only possible way to move towards the framework of agents heterogeneity. *Rational expectations hypothesis* is of course a good benchmark to explore with analytical tools which is the emerging market behavior and indeed standard financial theory usually works under such assumption. What has to be clear is that Agent-Based financial models are conceived not to contrast the EMH and all its achievements from a theoretical point of view. Instead we are interested to leave some unrealistic assumptions or restrictions, moving forward a kind of financial markets in which a wide ecology

of trading rules can determine many macro empirical features not well explained by the EMH.

In order to get a list of past and recent Agent-Based financial markets you can consult LeBaron (2002)[1]. All such markets are created to represent a very large set of expectational models and trading strategies, in order to understand market dynamics outside the *rational equilibrium of homogeneous expectations*. The common goal is to build an ABM able to mimic some real complex features of financial systems and attempt to understand the origin of these phenomena as a multi-agent interaction effect.

2.1.1 The Santa Fe Artificial Stock Market

Among the financial ABMs, maybe the most famous is the *Santa Fe Artificial Stock Market*, born in the late of 1980's and early 1990's as desire of Brian Arthur and John Holland to build a financial market with an ecology of trading strategies[2]. The SFI market structure is quite simple because there are two tradable assets, one risk free bond (in infinite supply) and a risky stock paying a stochastic dividend simulated by an AR(1) process. Agents in the market form their demands according to a classic portfolio choice framework (myopic investor with CARA utility function) and use a classifier system to make their predictions on the first and second moments of the stock returns; the forecast rules are fixed as linear pricing models, with respect to prices and dividends. The emerging global price for the stock is obtained using a clearing price condition: on the basis of all asset demands the price is set to its equilibrium value such to satisfy all the demands (i..e *clear the markets*). This is a first important choice to take in mind if we want to compare such ABM to that one built in our work. Anyway one of the most interesting and brilliant tools used in the SFI market is the condition/forecast classifier to represent all the world of linear expectational models. Resuming in few words, every forecast rule (represented by particular values for the coefficients in the linear pricing function) is conditional upon a particular state of the market. A certain market state is symbolized by a bit-string and it can be possible to define a string which represents several states together. Every bit is used to carry a particular market condition, both in terms of technical and fundamental analysis. With great simplification, when an agent wants to set her stock demand she can choose among a set of rules; so she will pick the forecast with a condition rule which matches with the current state of the market. The process can be named as *inductive learning* because a single trader does not worry about which are the expectations of the other investors about stock price (*deductive learning*). Testing every time a particular forecast rule, agents will decide to drop the rules which have badly performed and maintain those who have well performed. Such learning process is enriched by natural selection and mixing of market predictors (the bit-strings to represent states of the market) thanks to the use of a *Genetic Algorithm*[3]. The role of the GA is to operate on a lower time scale with respect to the choice made by the traders: all the nonperforming predictors are discarded through time, while new predictors (created as a combination of the performing existing ones) are produced. This is a very graceful and sophisticated learning mechanism which is maybe the great novelty with respect to the previous works in ABM for financial markets.

The results of the SFI market model are very interesting since we can individuate an equilibrium solution of rational homogeneous expectations and this is a theoretical strength for the work, since there is a regime which agrees with the standard financial framework. But most important, beside such solution there is a different regime which instead shows several complex patterns empirically observed in real stock markets: trading volume and price volatility different from zero, significant returns autocorrelations, as main results. So the SFI market opens up a new perspective in building an artificial agent-based stock market and it can

[1] *Building the Santa Fe Artificial Stock Market*

[2] If you desire a complete description and discussion of SFI market you can see LeBaron both in *Agent-based computational finance: Suggested readings and early research* (1999) and in *Building the Santa Fe Artificial Stock Market* (2002)

[3] For an introduction and complete explanation on GAs see Beltratti et al. in *Neural networks for economic and financial modeling.* (1996)

be a very interesting benchmark, both in theoretical terms and for its computational tools, to deal with topics regarding complexity in financial systems. In our work we shall try to compare the hypothesis or results of our ABM to those of the SFI market, every time discussing which can be the common goals and tools and which are the great differences.

Since we adopt the methodology of ABMs in this work it is also very important to turn the table and illustrate the potential weaknesses of Agent-Based works, in particular for the SFI market. One of the first problem is the sensitivity to some control parameters in the model. When we define a control parameter we are simply creating a button which allows us to manipulate in some sense the simulation. It is often a good idea to define parameters with a theoretical basis from the economic point of view because in this way we can know a priori their effect on the simulation and their possible relationship with other variables (as the macro aggregate variables that we want to monitor). However sometimes we need to introduce control parameters which are more artificial and whose meaning is not so immediate. This is a common trick adopted in several ABMs and a simple but important example comes from the SFI market, where we introduce the learning rate at which the GA changes the market predictors. The sensitivity to such parameter is strong and indeed there are two different regimes according to the initial choice[4]. The problem of sensitivity to some control parameters can be faced using calibration through real financial data. Anyway a control parameter which has a difficult interpretation with respect to real markets, is very difficult to calibrate. We shall see in our simulation similar problems, which will be solved in some cases through calibration with a real data sample. In general we have to be aware of the fact that sensitivity to control parameters and their relative tuning is a very important field when we are using an ABM. If the tuning is not possible we have a very large parameters space and this, though can be seen as a source of higher information, implies a more complicated work. We would like to not explore regions of parameters space which corresponds to unrealistic market dynamics, so the task is to try to calibrate all control parameters in a reasonable range, if possible. In Chapter 6 we shall see a very intelligent technique which can be adopted to be sure of having realistic outcomes from the simulation. The question of how to calibrate control parameters is common to all ABMs, not only in the financial context. During the construction of an Agent-Based market the modeler has to made several important choices about the kinds of agents, the market structure, the tools used to monitor and study the learning process. With respect to these three categories it is now our purpose to analyze some of the usual assumptions made in the known ABM markets (particularly the SFI market). So we can easy understand which are the strengths and the weaknesses of such works.

2.1.2 Agents classification: rational and noise traders

First the agents classification is a very important degree of freedom in modeling an artificial stock market. The baseline point is to adopt a fundamental dichotomy between *rational* and *noise* traders. Rational traders can be initially intended as those agents who share identical expectations of an asset's future price, and who instantaneously and rationally discount all market information into this price, as standard financial theory assumes; in contrast, noise traders are some investors who do posses expectations different from those of rational traders[5]. Of course this is a too simple classification and we have to think about a more complex variety in trading strategies for the investors. The SFI market is really powerful in this sense because every agent has its own forecast rule (so it can be considered as rational) but heterogeneity in expectational models is guaranteed in a very elegant way. What is actually relevant to note is however that the SFI market is

[4] See again LeBaron in *Building the Santa Fe Artificial Stock Market* (2002)

[5] A more detailed discussion is contained in *Asset Pricing Under Endogenous Expectations in an Artificial Stock Market*, W. B. Arthur, S. N. Durlauf, D. Lane

just a market in which all investors can be considered as rational (i.e. they all have an asset pricing model to get their asset demand): we are then in presence of rational heterogeneous expectations. In real markets however there will be always many investors which adopt trading strategies totally uncorrelated to some well documented expectational model. Many traders will invest according personal and irrational strategy, but their role in the market is so important as that of rational traders. This could be seen as a first weakness for the SFI market. Anyway the objective in that work was to explore the contrast with rational homogeneous expectations, then the idea to leave the standard equilibrium solution, yet maintaining rational expectations, was the core element. Indeed the role of noise traders is that of counterparts of rational traders; in a market in which rational expectations are so heterogeneous, as in the SFI market, there are always counterparts (if we are working in the complex regime with high learning rate for the GA) and noise traders are no more essential. In other works however it can be more realistic introduce traders who operate without some rational strategy, such to compare the performance of a random strategy with that one of particular asset pricing models.

2.1.3 Market structure: clearing price condition

Another very important question is the kind of market structure we introduce in our ABM. In other words, which is the mechanism which brings out the price of all securities in the market. This is a very essential topic, particularly from the the complex systems science point of view. For example, in the perspective of statistical physics, stock prices can be seen as the macro variables which are the results of micro interactions among agents. So which is the nature of these interactions? All the main ABM markets, even the SFI market, uses the clearing price condition to get the stock price which we observe in the market. The clearing price condition is used in modeling many economic systems: it is a condition very effective from a mathematical point of view, but it can be not so realistic in a stock market. The reason is that in real financial markets stock prices arises through a double auction system. The latter is the true form of interaction among agents, who bid or ask a price for every asset, making an order with a specific size. The double auction market is a structure which entails complexity because the stock prices arises not only according to asset demands of all investors, but also on the basis of the temporal order on which they send their envelopes in the electronic system. Such mechanism is really realistic and go beyond the clearing price condition. In building an ABM market we have to consider if a realistic trading system can be useful to pursue our research goals or if the clearing market assumption is enough to analyze complex patterns in stock markets.

2.1.4 Learning process for the agents

Once we have made a particular agents classification it is essential to decide how agents can learn during the simulation. Several choices are possible, but the delicate theme is not only how agents can adapt their forecast to new emerging data, but what kind of learning tools we use to explore the space of expectational models. Here we remind two important classes of learning algorithms used in ABM markets [6]. The first is the use of a GA to make selection/discarding operations among a wide ecology of expectational models. This is indeed the approach used in the SFI market where the GA is the basis of the learning process, as previously illustrated. The parameters space explored by the GA is, in that ABM, the space of conditional forecast rules, since what does matter are the market predictors associated to particular parameter values for the linear pricing models. The GA is very effective and unbiased in the learning process because we

[6] Both classes are discussed in relation with economic and financial systems in *Neural networks for economic and financial modeling*, Beltratti et al. (1996)

do not have any problem relative to underfitting or overfitting: the expectational rules space is explored without any bias and mixing the existing performing predictors in a process which is the same of natural selection. The other class of learning tools are the Artificial Neural Networks (ANN) that are used by agents to make forecast about future asset prices, as was done in the work of Beltratti and Margarita (1992) and Beltratti et al. (1996). The difference with respect to the SFI market is that the ANN allows to construct *non linear* pricing models so opening a more complex class of expectational models which can better individuate stock prices trend. However the use of Neural Networks implies several choices to calibrate the out-of-sample performance of these learning tools and we have to be careful to not construct too complex forecast models (represented for example by too many hidden units in the network), which will could fit very well the training set but badly perform out-of-sample.

2.2 A first outlook to the stock market model

We are trying to simulate a stock market in which investors can interact with each other and with some special agents, the consultants. Indeed any investor has to allocate her financial wealth among N possible assets and, to do this, she can pay an analyst/consultant who will worry about the optimization of her portfolio. The consultant elaborates the information he has at each trading day and, using some econometric model, computes an estimate for future asset prices; then she optimizes the portfolio of her client (according to her degree of risk aversion), i.e. determines the optimal asset composition for each security. After receiving these values, the investor will try to buy or sell assets, asking or bidding a price for each asset in the corresponding auction; the double auction mechanism will complete the possible trades and the new asset prices will emerge in the market. *So the investors will try to construct the optimal portfolio, but, actually, the interaction with the asking or bidding prices of other investors can often lead to realization of an imperfect portfolio. This effect is one of the most realistic consequence in terms of complexity: in a market in which there is a clearing condition for prices, every investor gets all the shares of stocks he has demanded (or sells all the shares he has offered*). Here instead, the complex interactions between investors in the auction limits the possible trades, depending on how investors have chosen asking or bidding prices, and, as a consequence, how consultants perform forecasts about asset returns.

2.3 Why ABM with Network Analysis?

The attempt to join agent-based models with network analysis is not so frequent in the recent literature[7]. This work operates in such new research context not simply to study a complex financial market, but also to provide a policy making tool for real applications. In the field of complex systems we have a straightforward necessity to consider the set of interactions among agents as an independent entity which can be actually quantified. However, the process usually followed is to build a mathematical model of the network, which is defined a priori on the basis of some regularities observed in real data. In this way we use some simple assumptions to mimic social interactions among agents in mathematical terms: the result can be interesting and useful to define global property of static networks, but is not clearly enough to describe a set of dynamic interactions which are very difficult to model with a fixed mathematical rule. So the mathematical formalism, which for example defines the generation algorithm of our network, limits our representation of real interactions among agents.

The solution is to use an ABM which controls the generation and evolution of the network itself. This approach solves several problems of the classical framework of network analysis. Firstly the ABM

[7] A recent paper that deals with the topic is *From Agen-Based Models to Network Analysis (and return): The Policy Making Perspective*, Fontana et. al. (2015)

is intrinsically dynamic and this will make the network an evolving tool (i.e. a temporal network) which reacts together with the simulation. As a second effect, the rules which defines network dynamics can be built exploiting specific micro behavior of agents. The latter can interact on the basis of both micro and macro variables and the ABM can easily monitor such interactions and form new links or drop some of the old ones. This is equivalent to build a generation/evolution algorithm which depends on the state of the system; if we change important settings in the simulation world, also the emerging network will change, even maintaining the same network rules in the program. There are no more mathematical conditions that fix a priori the emerging properties of the graph and the network becomes an evolving complex object which reflects the specific dynamics of your system.

We can also see the network as a sort of elegant bridge between micro variables of agents and macro aggregate quantities of the system. Indeed you can compare local network properties of a single node with the features of the corresponding agent. On the opposite you can analyze global network properties under different regimes in the ABM. Both these directions in network analysis are considered in this work.

Finally, using Agent-Based Network Analysis we can look for effective policy making actions in a complex economic system. There are significant parameters or rules (think for example to limitations regarding short-selling in financial markets) that can be easily modified in an ABM to test what is the effect on the system. This operation is reinforced using a network which evolves in dynamically way over time: the network indeed does not reach an equilibrium condition in the sense of General Equilibrium theory, but continuously changes together with agents behavior. In other terms we have a flexible tool which can represent the non-equilibrium state of an economic system, according to the perspective of economics of complexity. Quantitative measures allow to study many important network effects which are essential in policy making interventions. Every economic system has to be conceived not as a sum of agents who determine aggregate variables, but as an evolving entity in which all the parts are linked together through dynamic interactions. Only an ABM which implements a dynamic network can catch all such features.

2.4 Themes we are going to face with this work

The main goal of this work is to investigate if many real aspects of a stock market can naturally emerge from our model, in contrast with some important pillars of standard economic theory about financial markets. Here we report some of the main interesting topics we can deal with; during the work we will speak about each of these themes using simulation data previously obtained, as a benchmark of discussion.

- Is the EMH perfectly valid? In other terms, asset prices follow a random walk process or there is a significative correlation effect in returns series? Is this effect linked to some properties in the kind of interactions among agents?
- Does the simulation show some real market phenomenons, such bubbles or crashes? And, if so, do these effects arise naturally from some important parameters or mechanisms in the simulation?
- What is the nature of asset return distributions? Can we check some non zero skewness or kurtosis? What can kind of stochastic process is able to fit stock prices dynamics?
- Does the model bring out some interesting behavior rules according to which investors naturally maximize their profits changing their consultant? Does the learning and decisional process of consultants and investors naturally converge to some equilibrium condition (as, for example, in the homogeneous expectations framework) or not?
- Can we find significant investors profits (and then corresponding trading strategies) in the market?

- What are the main properties of interaction which emerges from simulation dynamics? Are the network structure and metrics correlated with the evolution of asset prices and investors profits?

Obviously these questions are often linked together and they well express the emergence of complex patterns in the market. Combining stock price series of the simulation with the analysis of the network which arises and evolves over time, we can answer these questions, and, maybe, we can better understand how a complex system such a stock market works. Moreover, starting from this work, there is an appealing idea to pursue for future researches: it is possible to identify policy-making actions which can effectively stabilize the market behaviors or *moods* or there is completely no chance to manipulate with some efficacy such a complex environment?

3 The model

We shall consider a stock market in which M investors can purchase and trade N risky assets (we will consider only stocks paying no dividend); all their trading operations can follow the advice of one of of the H consultants. In this market the risk-less rate (the rate on a T-bill) is denoted by r_f,. The time evolution of the market is structured according to a continuous auction (for all stocks) which defines every single day of trading. In our market we do not worry about investment on T-bills and our aim is to investigate how the *risky portfolio* is constructed during the simulation. So the knowledge of r_f needs us only to take the optimal risky portfolio shares for the stocks. From now on (if not specified in other way) we will talk about investor's portfolio as the risky portfolio, while when we will refer to financial wealth this corresponds to the financial wealth owned in stocks (equity).

A clever and schematic way to present the model is to explain the features, the actions and the behavior rules for the two different types of agents, the investors and the consultants. Here we anticipate a general scheme for agent classification to take in mind for the next paragraphs.

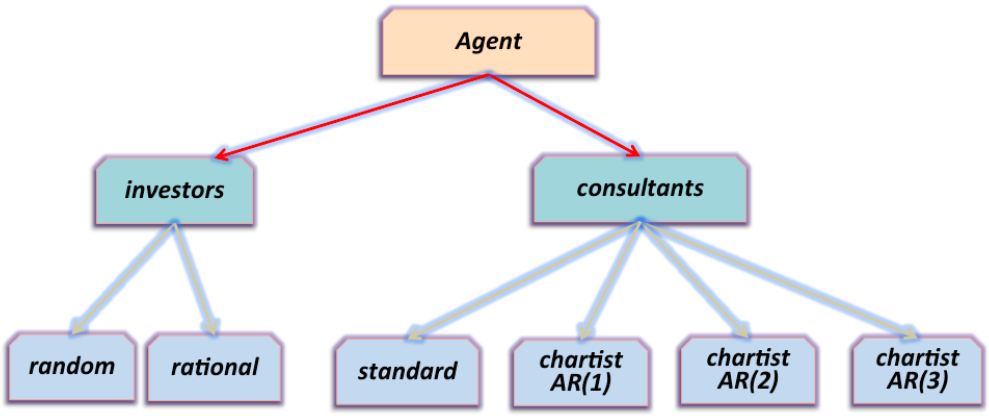


Figure 1: Agents classification in our ABM: random investors act on the market from their own, while rational investors will consider the optimal portfolio estimate, taken from their consultant, to buy or sell stocks. The four classes of consultants represent four kind of expectational models, leading to different optimal portfolio estimates.

3.1 Investors

The first question is to understand if investors in a stock market are always rational, i.e they trade according to some financial model which relies on new information about assets, or, conversely, if they are also irrational, buying and selling stocks only on the basis of personal and temporary moods and beliefs not well supported by market knowledge. The reality is that the contribution of irrational investors is essential to bring out many complex features in a financial market. The standard financial theory, which assumes that all investors share rational expectations of an asset future price (discounting all market information into this price), fails in proving these characteristics. So it is crucial to introduce many investors in the market who do not consult any specialist to trade, but simply invest in response to personal feelings or according to some self-made strategy. We call this kind of agents as *random* investors, in contrast to the other class

of investors who set their asset demands with the help of a financial consultant.

All investors are characterized by a degree of relative risk-aversion γ^j (comprised between 2 and 4), a cash account S^j and a financial wealth (i.e. the current value of all the owned stocks) W^j. We start from a situation in which every investor yet owns a portfolio of stocks; this portfolio is determined in relation with the nature of the investor we are considering (see next). However the idea is that the initial state of the market (and therefore of the simulation) presumes that all investors have a well determined financial wealth: there are no agents who begin the simulation without stock ownership. This implies that we have agents who traded stocks in the past, constructing their personal risky portfolio. At each trading day (even during the auction) investors can choose a fraction of their cash account to perform their trading strategy in order to correct their portfolio by purchasing stocks or they can sell the stocks they yet have without any further investment in the market. Also short-selling and leverage operations are allowed. All these actions require some decision making capabilities and we will see how to implement such decisions, taking into account the different two classes of investors. However let us give first of all some important elements we set in the creation of an agent of the investor class:

- we set randomly her degree of relative risk-aversion γ^j (using a uniform probability distribution between 2 and 4);
- we set randomly her cash account S^j as a random number drawn from a uniform distribution on a certain range. So here we can choose the possible range of equity value for the investors.
- we fix her financial wealth W^j, which is the product of the number of owned shares per their actual value. So we have to draw, for each stock, how many shares the investor owns: this is done defining the vector $\vec{n}$ that includes the number of owned shares for all stocks. Then it is possible compute the corresponding financial wealth relative to asset i, i.e.

  $$W_i^j = n_i p_i$$

  where n_i is the i^{th} component of $\vec{n}$ and p_i is the *current price for asset* i. As we will see investors use such prices to make some investment decision steps.
- finally we can compute the total financial wealth as $W^j = \sum_{i=1}^{N} W_i^j$.

These are the initial conditions for an investor of one of the two possible classes. We set the total number M of investors in the simulation and then we choose how many of these will be random investors (M_{rand}) and how many will be classified as rational (M_{rat}). For future analysis we call the relative fraction of random investors in the market as $n_{rand} = 1 - n_{rat}$, where n_{rat} is, on the contrary, the relative fraction of rational investors.

Once any investor has an initial portfolio the problem is to define how she will bid or ask a price for a security. This process follows a general scheme (both for random and rational investors) and is implemented before the beginning of a trading day. Here we illustrate the main steps:

1. on a certain trading day (indexed by t) every investor sets which is the *maximum* amount of money will invest for a single asset, i.e. a *maximum common* threshold for every order in the auction. This amount of money, denoted by ΔS_t^j, is set according the kind of investor we are considering and enables the trader to correct her portfolio, represented as a vector of shares for every security. First of all it is important to note that ΔS_t^j fixes an upper bound for both purchasing and selling orders and simulates the investor choice to set an *ideal* limit in orders magnitude, but *does not mean the investor will actually spend in the market a total amount of money bounded by* ΔS_t^j. The introduction of

such constraint in all trading operations is very important from a computational point of view and moreover is related to different *a priori* strategies for random and rational investors in the choice of ΔS_t^j.

2. once we know this upper limit to finance long (or short) positions on assets we have to consider what is the portfolio goal, i.e. which are the shares of stocks we fix as objectives. In the case of rational agents these shares will be well determined by the work of one of the consultants; for random investors will see instead that the mechanism to diversify the investment among different stocks is not directly related to the optimal portfolio shares.

3. having well in mind the portfolio goal, the investor can now bid or ask prices (through ΔS_t^j), always showing the related number of shares.

3.1.1 Introducing to rational investment scheme

It is possible to put this problem on a mathematical framework: this is very useful for investment plan of rational investors, while, as we will show later, random traders act on different behavior rules, not being focus on the optimal portfolio goal to reach. However we include here such mathematical discussion because we need to introduce several essential quantities for investment plan of both kinds of investors; moreover this can allows us to start to define some important differences between the two class of investors.

Let us consider the portfolio at the beginning of some trading day as $\overrightarrow{w_t}$: fixing the portfolio goal as $\overrightarrow{w_t'}$, the objective is to reach such portfolio with investment operations. An investor tries to complete this task buying and selling stocks, always respecting the constraint given by ΔS_t^j. In order to get the new optimal portfolio $\overrightarrow{w_t'}$ we have to apply a variation of financial wealth ΔW_t^j. It is important to note that ΔW_t^j can be positive, negative or even equal to zero since we can augment our financial wealth through purchasing stocks but we can also decrease it through selling stocks; so the two effects can perfectly balance each other leaving unchanged the total equity. This is equivalent to decompose ΔW_t^j as the sum of positive terms, relative to the purchase of shares, and negative terms, relative to the sale of other stocks:

$$\Delta W_t^j = \sum_{i,buy}^{Nbuy} \Delta W_{i,t}^j + \sum_{i,sell}^{Nsell} \Delta W_{i,t}^j$$

Dropping again the indexes t and j, we now try to express the terms ΔW_i as a function of their corresponding w_i', which are the new optimal portfolio shares for every stock. It is suffice to start from the expression of w_i'

$$w_i' = \frac{W_i + \Delta W_i}{\sum_{i=1}^N W_i + \sum_{i=1}^N \Delta W_i}$$

we can simplify such equation taking into account that $\sum_{i=1}^N W_i = W$ is the equity value (i.e. the total financial wealth) at the beginning of the trading day

$$w_i' = \frac{W_i + \Delta W_i}{W + \sum_{j=1}^N \Delta W_j} \Rightarrow w_i'(W + \sum_{j=1}^{N} \Delta W_j) = W_i + \Delta W_i$$

$$\Rightarrow w_i'(W + \sum_{j\neq i}^{N-1} \Delta W_j) = W_i + \Delta W_i - w_i' \Delta W_i$$

$$\Delta W_i = \frac{w_i'(W + \sum_{j\neq i}^{N-1} \Delta W_j) - W_i}{1 - w_i'}$$

where $\sum_{j\neq i}^{N-1} \Delta W_j$ is the sum on all terms ΔW_j different from the equity variation of the considered asset i. This is a system of N equations with infinite solutions for ΔW_i: *there are infinite possible choices of* ΔW_i *to reach the optimal portfolio goal this is what we actually expect from our model*. It is possible to explain why this system has infinite solutions using linear algebra theory. A linear system $AX = B$ is studied on the basis of the matrix A for the associated homogeneous system, and of the vector B containing all the constant terms. In this case the matrix A has $N \times N$ dimension and the vector B has N components (N is the number of assets in the market). Let us describe matrix terms for A and components in B.

$$A_{ii} = 1 - w_i'$$

$$A_{ij} = -w_i', j \neq i$$

$$B_i = w_i'W - W_i$$

Now we have to study the augmented matrix $A|B$, which describes the solution of our system. The Rouchè-Capelli theorem states a linear system has one possible solution only and only if $rank(A) = rank(A|B)$. So in our case we expect that

$$rank(A) < rank(A|B)$$

In this way we have infinite solutions which depend on the number of free parameters in the system: this number is equal to the difference $rank(A|B) - rank(A)$. It is possible to check, in any dimension (i.e. with any number N of assets in the market), that

$$rank(A|B) = N, rank(A) = N - 1$$

So there is only a single free parameter in the system to find the solution we want. Such result can be also thought in other terms: *there is a* ΔW_i *value which is totally (almost, as we will see below) arbitrary and we can fix it in a random manner to determine all the remaining* ΔW_i terms. This process will be illustrated in detail in the section of rational investors. Here we start showing that the solution of this system can be represented by $N - 1$ relations of this kind

$$\Delta W_{i,t}^j = f_i(\Delta W_{i,t}^{j*})$$

where $\Delta W_{i,t}^{j*}$ is the free ΔW_t^j term (corresponding to the free parameter) and f_i is a linear function. So these $N - 1$ linear functions of $\Delta W_{i,t}^{j*}$allow us to compute one of the infinite solutions of the system having fixed $\Delta W_{i,t}^{j*}$: here show the actual final solution of the system, making explicit all f_i

$$\Delta W_i = \frac{(\Delta W_i^* + W_i^*)w'_i}{w'_i*} - W_i$$

with indexes j and t dropped for simplicity. *This is the analytical final solution for all* ΔW_i, on the basis of the free term ; note instead that both W_i^* and $w_i'*$are not arbitrary but all determined in the simulation. With these $N-1$ available equations we can get *one of the possible infinite solutions for* ΔW_t^j: such process is included in the section relative to rational investors.

Random investors will directly compute all ΔW_i without using w_i', but we can always use the last equation to get w_i' from ΔW_i in order to see its value. Anyway there are further constraints on ΔW_i to consider. The ΔW_i values will be used to send an order on the auction and so they have to be not too small in magnitude (remind that they can be negative or positive to represent purchasing or selling intentions): in particular the price of a single share can fix a sort of lower bound for $|\Delta W_i|$ and we decide to fix this lower limit to the current price for asset i, denoting it with p_i. *The latter is then the exe price in the auction.* Moreover there is a natural upper bound given by the choice of ΔS_t^j: an investor cannot buy stocks for an amount greater than the cash amount set as maximum threshold in investments, i.e. $|\Delta W_{i,t}^j| < \Delta S_t^j$. Note that, according to the meaning of ΔS_t^j, this constraint is valid also for negative $\Delta W_{i,t}^j$ values and so is perfectly symmetric. We combine all these considerations to get the following inequalities for $\Delta W_{i,t}^j$.

$$|\Delta W_{i,t}^j| > p_i \wedge |\Delta W_{i,t}^j| < \Delta S_t^j$$
$$\Rightarrow p_i < \Delta W_{i,t}^j < \Delta S_t^j \vee -\Delta S_t^j < \Delta W_{i,t}^j < -p_i$$

These two inequalities are very important because are valid for both random and rational investors. As we will see the closing price of each trading day is put into the data sample used by consultants and is completely available to all agents in the market. We can add these two inequalities to the solution of the previous systems of equation for $\Delta W_{i,t}^j$, so finding allowed $\Delta W_{i,t}^j$ values: this will be done speaking about investment scheme for rational traders.

It is important to underline that ΔW and ΔW_i are variations in the equity which we fix as a goal, and they are not the actual variations determined by the auction or by market fluctuations in stock prices. Anyway we set such values as guideline in investments, in order to get the final bid or ask prices.
Once we have fixed all ΔW_i we can try to obtain such variations buying or selling stocks in these quantities:

$$n_i = \frac{|\Delta W_i|}{p_i} - mod[\frac{|\Delta W_i|}{p_i}, 1] = floor[\frac{|\Delta W_i|}{p_i}]$$

where if ΔW_i is positive n_i is the number of purchase stocks, while if ΔW_i is negative n_i is the number of sold stocks. *The price p_i is the current last price for asset i, considered at the moment at which the investor decides to enter in the auction.*

The solution shown above implies that the purchase or the sale of n_i shares leaves a remainder part of ΔW_i untouched, which is $|\Delta W_i| - n_i p_i$. Let us explain how this remainder term is used in the case of purchase or sale of stock, extending the expression for n_i in both cases.

If we are purchasing stocks this part can be used to augment the final bid price. Moreover you can decide to buy a number of shares lower than the maximum allowed quantity (which is $floor[\frac{|\Delta W_i|}{p_i}]$): this is a strategy to make a bid price more competitive (higher on the book). We model such choice expressing

$$n_{i,buy} = floor[\frac{|\Delta W_i|}{p_i}] - U(0, \triangle n_{trade} floor[\frac{|\Delta W_i|}{p_i}])$$

So we are assuming orders with demanded quantities comprise between the maximum possible value ($floor[\frac{|\Delta W_i|}{p_i}]$) and a random quantity represented by the uniform distribution $U(0, \triangle n_{trade} floor[\frac{|\Delta W_i|}{p_i}])$. The control parameter $\triangle n_{trade}$ is a fraction (real number between 0 and 1) which sets the maximum possible deviation of the demanded number of shares from its maximum value. Let us note that $\triangle n_{trade}$ is not an absolute number of shares chosen as maximum variation, but a relative portion of the maximum order size: in this way we apply a global maximum variation, in percentage terms, to all types of orders, which

could be relatively small or very big in volume. An investor can buy this amount of shares bidding a price which is higher than the current one, in this way

$$Bid \rightarrow p_{bid,i} = p_i + \frac{\Delta W_i - n_{i,buy} p_i}{n_{i,buy}} = \frac{|\Delta W_i|}{n_{i,buy}}$$

Lower is $n_{i,buy}$ and higher is the bid price, with the consequence of an order which is then more competitive on the auction market (a higher bide price occupies a better position in the book).

If we are going instead to sell stocks we have to use the quantity $|\Delta W_i| - n_i p_i$ in a different way. Indeed if we applied the same reasoning of the purchase case, we should increase the ask price to reach the entire amount ΔW_i through our selling order: but this kind of choice will entail a worst position of the order in the book, so lowering the probability of completing the operation. Since one of the main goal of an investor is to make an order which can have good probability to be processed, this operation is counter-productive (even for a random investor). On the contrary, the strategy adopted will be to increase by one the number of shares to sell and simultaneously decrease the ask price of the order, so reaching the total amount $|\Delta W_i|$; in other terms we redefine the number of shares to sell as

$$n_{i,sell} = \frac{|\Delta W_i|}{p_i} - mod[\frac{|\Delta W_i|}{p_i}, 1] + 1 = floor[\frac{|\Delta W_i|}{p_i}] + U(0, \triangle n_{trade} floor[\frac{|\Delta W_i|}{p_i}])$$

Here $\triangle n_{trade}$ controls the deviation of the supplied number of shares from the minimum value ($floor[\frac{|\Delta W_i|}{p_i}]$). *So globally $\triangle n_{trade}$ defines the overall dispersion of order sizes, and, as main consequence, the overall volatility of bid and ask prices. This is of course a great source of freedom for all investors (both rational and random) in the trading process: even having the same portfolio goal, two investors can try to reach them through different orders, in terms of quantities and prices. A $\triangle n_{trade}$ really close to zero leads to low prices fluctuations; increasing $\triangle n_{trade}$ we are then guaranteeing more volatility in the market.* Once we have determined the number of shares we are trying to sell we will define the final ask price of the order

$$Ask \rightarrow p_{ask,i} = p_i + \frac{|\Delta W_i| - n_{i,sell} p_i}{n_{i,sell}} = \frac{|\Delta W_i|}{n_{i,sell}}$$

So we have now a main scheme to compute the ask and bid price of an order, with their related number of shares. Remind that this phase of investment plan is the same for both random and rational investors.

We want only make clear the most important question in all the investment scheme: *ΔW_i is not the actual equity variation for two reasons. First it corresponds simply to the total cash value that we can get from our order, but if the order is actually processed the actual equity variation (in terms of the asset i) is not always the goal variation ΔW_i . You can check this on the basis of the matching mechanism in a double auction market: when our order is processed the stock price is set to our bid or ask price only if we entered on the auction before our counterpart; otherwise the new stock price will be that chosen by our counterpart. This leads to an immediate actual equity variation that can be different from ΔW_i (lower for buyers and higher for sellers). Anyway a second and most important factor, which partially contrast such effect, is that we can have an actual equity variation ΔW_i only if the trading operation regarding the stock (purchasing or selling) is completely executed in the auction (i.e. all the shares demanded is purchased or all the shares offered are sold). This is a straightforward consequence of the complex interaction which occurs through the auction mechanism: only if an investor managed to complete all her trading operations (furthermore in a brief period of time) she would reach actually the portfolio goal. Such assertion is obvious when we present a stock market model based upon a clearing condition for asset prices, but here we are exploring a real form of market interaction as the auction: we expect then several deviations from the portfolio goal, which will*

be set randomly for random investors and it coincides with the optimal portfolio for rational investors. So these two kinds of market imperfections lead to an imperfect portfolio and our work is intended to analyze if an investor can really construct, over a short trading period, a portfolio which is close to the optimal one.

3.1.2 Temporal collocation

This is the main process used in our simulation to set purchasing or selling orders in the auction. What is the temporal collocation in the simulation for this process? We set randomly the moment in which a generic investor decides to enter in the auction: at this moment all the ΔW_is are assumed to be fixed because the investor has used the previous time to choose them (on a random basis or according to the Δw_is she has). So, the number of shares which can be traded and the values for ask and bid prices, will be at last determined by the exact instant of time at which the order will be sent in the auction.

Following this pattern for investments in the auction market, *we are assuming that every investor can only make a unique order on the auction, for a single asset, during one trading day.* This is a choice which simplify the simulation program structure and does not affect significantly the realistic impact of our work. Moreover we could ask ourselves what happens to unexecuted orders in the auction at the end of a trading day. They could be inserted again in the book of the following day but this would complicate the steps in investment decision: so *we assume that the every order has only daily validity and at the end of the trading day it will be eliminated from the electronic book.*

3.1.3 Short-selling and leverage operations

In all our discussion we did not speak about any constraint in investment operations: this means we are considering a market in which both short-selling and leverage trading operations are allowed. In mathematical terms then we have no bounds for the shares w_i' in the optimal portfolio vector. The possible introduction of any constraint on such terms will affect the portfolio choice procedure (which will be explained later). Anyway it is important to start from a simulation environment in which investors can make any kind of trading operations, since we are very interested to study the effect of short-sales or leverage strategies on the market.

Now we are ready to illustrate the two different classes of investors, reminding the main mechanism to form a purchasing or selling order on the market.

3.1.4 Random investors

In our model we assume that getting real time information about assets and processing it according to some financial model or strategy is very complicated and expansive for an usual investor. This kind of work can be done only by consultants of firms who are specialized in financial investments. So all the other agents who invest in the market are considered to trade in a random manner, because we cannot reproduce and know how they choose to invest, but also because they can be strongly irrational. We present the main behavior rules which define how a random investor operates on the market:

- during each trading day a random investor j sets the amount of money designed for correcting her portfolio, ΔS_t^j: this amount is set according to a uniform distribution because we do not want to privilege some values. However the range of allowed values for ΔS_t^j (i.e. the support of the uniform distribution) varies according to the relative degree of risk-aversion γ^j. In particular we are using the

following uniform distribution

$$p(\Delta S_t^j) = U(\max_i\{p_i\}, \frac{W_t^j}{\gamma^j})$$

where p_i is the current price for asset i and W_t^j is the current value of the portfolio, where both are considered in real time during the auction (using the opening price values if we have no yet trading operations completed). The index t here represents not simply the trading day at which we compute W_t^j but also the actual instant of time during the auction in which we are considering the equity value. Obviously we could choose a different range (the support of the distribution) for ΔS_t^j, this is simply one possible reasonable choice: for the most risk-adverse random investor (with $\gamma^j = 4$) the maximum allowed value for ΔS_t^j is 25% of the value of the portfolio, while the least risk-adverse investor (with $\gamma^j = 2$) will choose a maximum value for ΔS_t^j equal to 50% of W_t^j . The choice of a lower bound for ΔS_t^j equal to $\max_i\{p_i\}$ is essential as we will see later speaking about the choice of the quantities $\Delta W_{i,t}^j$. Indeed ΔS_t^j defines a ceiling amount for trading operations and if we had $\Delta S_t^j < p_i$ this would imply that we cannot operate on the auction relative to asset i; therefore we avoid such situation for every stock, choosing a lower bound equal to $\max_i\{p_i\}$.

What is important to understand in every possible choice for the support of the uniform distribution is that we want a decreasing range when the investor has a higher γ^j: when an agent is less disposed to bear risk she can invest a higher amount of money to buy stocks. However we note that this distribution is independent from the risk-less rate: this, as we will see, is a difference between a random investor and a rational one.

- once the random investor has chosen the maximum amount she will invest in a trading operation for the current day, she has to compute the values of $\Delta W_{i,t}^j$ (which can be positive or negative or even null). We have seen how we can determine such variables starting from the knowledge of the changes in portfolio weights Δw_i. But here we are talking about random investors and so we have no information about the objective Δw_is: we could set them randomly but it is more realistic assume that a random investor will not fix a Δw_i as a goal in her investments, but rather she will choose every single $\Delta W_{i,t}^j$ according to some irrational moods or to some strange and unknown strategy. So we have to draw the value of $\Delta W_{i,t}^j$ from a new distribution and then get the corresponding value for Δw_i (it is suffice to invert the equation used for $\Delta W_{i,t}^j$): such process is more effective since the investor does not think in terms of objective shares in her portfolio, but she simply decide to invest a certain amount of money for every several stocks.
  In order to construct a proper distribution we have to consider that we have to always satisfy the conditions $p_i < \Delta W_{i,t}^j < \Delta S_t^j \vee -\Delta S_t^j < \Delta W_{i,t}^j < -p_i$. So we can use for instance a uniform distribution on two opposite ranges, $(p_i, \Delta S_t^j)$ and $(-\Delta S_t^j, -p_i)$: to be more precise we have first to randomly draw what will be the range in which $\Delta W_{i,t}^j$ lies (so deciding if the investor intends to buy or sell such stock, with equal probability) and then use the corresponding distribution

$$p(\Delta W_{i,t}^j) = U(p_i, \Delta S_t^j), p_i < \Delta W_{i,t}^j \leq \Delta S_t^j$$

$$p(\Delta W_{i,t}^j) = U(-\Delta S_t^j, -p_i), -\Delta S_t^j \leq \Delta W_{i,t}^j \leq -p_i$$

  In this way we cannot have $|\Delta W_t^j| > \Delta S_t^j$ or $|\Delta W_t^j| < p_i$. So far we have assumed that $\Delta S_t^j > p_i$ for all stocks: this is not always true in principle since ΔS_t^j is drawn starting from zero. So our algorithm

will have to deal with this particular case.
Having defined the correct distribution we will use it in the following way. An investor chooses at random (with the same probability) the first stock (among the N possible assets) for which she will set $\Delta W^j_{i,t}$, then she draws the actual value using $p(\Delta W^j_{i,t})$. As next step she will choose (always at random) a second stock and she will draw again their corresponding $\Delta W^j_{i,t}$.
Once we have determined all $\Delta W^j_{i,t}$s we could get the corresponding $\Delta w^j_{i,t}$, but this is an unimportant task from the point of view of a random investor.

- with the knowledge of all $\Delta W^j_{i,t}$s it is now possible to determine the bid or ask price for every stock according to general criterions illustrated above.

3.1.5 Rational investors

Investors who choose to set their asset demands according to some external specialistic agents are called rational investors. They are willing to pay an analyst to get a portfolio goal which can guide their trading operations. The cost of such service is one of the control parameter of the simulation and if it is too high it can be possible that investing through the support of consultants becomes not profitable, even if the forecasts performed are reliable. Here we present the behavior rules of a rational investor, in the process of determining a purchasing or selling order on the auction:

- first of all rational investors set the amount ΔS^j_t using this uniform distribution

$$p(\Delta S^j_t) = U(\underset{i}{max}\{p_i\}, \frac{W}{(\gamma^j)^{r_f}})$$

  In this way the maximum value for ΔS^j_t is inversely proportional to the relative degree of risk-aversion powered to the risk-less rate in the market. A highly risk-adverse rational investor, in a market with a high risk-free rate, will be disposed to invest a small amount of money to buy stocks because investing in the risk-less asset can be more fruitful. Our choice for the upper bound in the range is only conventional and we could change its expression, in order to get different maximum and minimum numerical values for it. Anyway, whatever is the choice made by the modeler, it is essential that the upper bound is inversely proportional to both γ^j and r_f. While the random investor does not consider the risk-less rate, i.e. the benchmark return useful to evaluate the risk premia on the market, the rational investor includes such rate to fix the cash ceiling to buy stocks.

- to fix the several values of ΔW^j_t , we have first to solve the related system of equations, i.e. we have to use the $N-1$ relations$\Delta W_i = \frac{(\Delta W_{i*}+W_{i*})w'_i}{w'_{i*}} - W_i$. In addition it is essential to remind the constraints on $\Delta W^j_{i,t}$, summarized with the overall condition $p_i < \Delta W^j_{i,t} < \Delta S^j_t \vee -\Delta S^j_t < \Delta W^j_{i,t} < -p_i$, which has to be satisfied for all stocks. This corresponds to solve a system of inequalities of this kind

$$p_i < \frac{(\Delta W_{i*} + W_{i*})w'_i}{w'_{i*}} - W_i < \Delta S$$

$$-\Delta S < \frac{(\Delta W_{i*} + W_{i*})w'_i}{w'_{i*}} - W_i < -p_i$$

  where we have four inequalities for each stock. Solving this system means we can get the allowed ranges for ΔW_{i*} such that all the remaining ΔW_i satisfy the conditions $p_i < \Delta W^j_{i,t} < \Delta S^j_t \vee -\Delta S^j_t <$

$\Delta W^j_{i,t} < -p_i$. Now we solve separately these inequalities

$$1]\frac{(\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i - w'_{i*}\Delta S}{w'_{i*}} < 0$$

$$2]\frac{(\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i - w'_{i*}p_i}{w'_{i*}} > 0$$

$$3]\frac{(\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i + w'_{i*}p_i}{w'_{i*}} < 0$$

$$4]\frac{(\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i + w'_{i*}\Delta S}{w'_{i*}} > 0$$

and we have to remind that we have to satisfy one of the two couples $[1-2]$ or $[3-4]$ at the same time, i.e.

$$1 \wedge 2 \quad \vee \quad 3 \wedge 4$$

The solutions can be found considering both numerator and denominator

$$1]true \Leftrightarrow \begin{array}{c} (\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i - w'_{i*}\Delta S > 0 \\ w'_{i*} < 0 \\ \vee \\ (\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i - w'_{i*}\Delta S < 0 \\ w'_{i*} > 0 \end{array}$$

$$2]true \Leftrightarrow \begin{array}{c} (\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i - w'_{i*}p_i > 0 \\ w'_{i*} > 0 \\ \vee \\ (\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i - w'_{i*}p_i < 0 \\ w'_{i*} < 0 \end{array}$$

$$3]true \Leftrightarrow \begin{array}{c} (\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i + w'_{i*}p_i < 0 \\ w'_{i*} > 0 \\ \vee \\ (\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i + w'_{i*}p_i > 0 \\ w'_{i*} < 0 \end{array}$$

$$4]true \Leftrightarrow \begin{array}{c} (\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i + w'_{i*}\Delta S > 0 \\ w'_{i*} > 0 \\ \vee \\ (\Delta W_{i*} + W_{i*})w'_i - w'_{i*}W_i + w'_{i*}\Delta S < 0 \\ w'_{i*} < 0 \end{array}$$

and so we get

$$1]true \Leftrightarrow \Delta W_{i*} < -W_{i*} + \frac{w'_{i*}}{w'_i}W_i + \frac{w'_{i*}}{w'_i}\Delta S$$

$$2]true \Leftrightarrow \Delta W_{i*} > -W_{i*} + \frac{w'_{i*}}{w'_i}W_i + \frac{w'_{i*}}{w'_i}p_i$$

$$3]true \Leftrightarrow \Delta W_{i*} < -W_{i*} + \frac{w'_{i*}}{w'_i} W_i - \frac{w'_{i*}}{w'_i} p_i$$

$$4]true \Leftrightarrow \Delta W_{i*} > -W_{i*} + \frac{w'_{i*}}{w'_i} W_i - \frac{w'_{i*}}{w'_i} \Delta S$$

now, calling the common term $-W_{i*} + \frac{w'_{i*}}{w'_i} W_i \equiv s_i$, and resembling the first two conditions

$$1-2]true \Leftrightarrow \underbrace{s_i + \frac{w'_{i*}}{w'_i} p_i}_{x_i^2} < \Delta W_{i*} < \underbrace{s_i + \frac{w'_{i*}}{w'_i} \Delta S}_{x_i^1}$$

$$3-4]true \Leftrightarrow \underbrace{s_i - \frac{w'_{i*}}{w'_i} \Delta S}_{x_i^4} < \Delta W_{i*} < \underbrace{s_i - \frac{w'_{i*}}{w'_i} p_i}_{x_i^3}$$

This is the solution for the inequalities relative to a single stock, but we have to solve the complete system of inequalities. This can be done imaging a graphical representation of the solutions, where we have to compute, for each asset, the four values $(x_i^1, x_i^2, x_i^3, x_i^4)$. First of all we have to note that all such solutions assume $\Delta S_t^j > p_i$, which is a condition always satisfied by construction through the selection technique of ΔS_t^j. In this case it is clear that

$$x_i^1 > x_i^2 > x_i^3 > x_i^4$$

though we cannot be sure of the sign of these four quantities. The solution of the system of inequalities is the following

$$\max_i\{x_i^2, p_i\} < \Delta W_{i*} < \min_i\{x_i^1, \Delta S\}$$

$$\vee$$

$$\max_i\{x_i^2, -\Delta S\} < \Delta W_{i*} < \min_i\{x_i^1, -p_i\}$$

where we have included also the constraints on ΔW_{i*} itself, i.e. $p_i < \Delta W_{i*} < \Delta S \vee -\Delta S < \Delta W_{i*} < -p_i$.

However implementing the solution of this system of inequalities in our simulation program is not so easy and entails different problems, both related to programming steps and to mathematical restrictions. Then it is more straightforward and convenient to build a selection algorithm which uses the solution of the system of $N-1$ equations ($\Delta W_i = \frac{(\Delta W_{i*} + W_{i*}) w'_i}{w'_{i*}} - W_i$), but does not explicitly take into account the solution of the system of inequalities. The technique we have conceived is quite simple: first choose randomly which is the stock $i*$ and set (with equal probability) if ΔW_{i*} is positive (purchasing) or negative (selling); then draw ΔW_{i*} according the corresponding distribution

$$p(\Delta W_{i,t}^j) = U(p_i, \Delta S_t^j), p_i < \Delta W_{i,t}^j \leq \Delta S_t^j$$

$$p(\Delta W_{i,t}^j) = U(-\Delta S_t^j, -p_i), -\Delta S_t^j \leq \Delta W_{i,t}^j \leq -p_i$$

which are the same used by random investors. The difference is that here, the remaining ΔW_i are computed through the deterministic solutions $\Delta W_i = \frac{(\Delta W_{i*} + W_{i*}) w'_i}{w'_{i*}} - W_i$. Once we have got all the

ΔW_i terms we have to check if the constraints $p_i < \Delta W_i < \Delta S \vee -\Delta S < \Delta W_i < -p_i$ are satisfied for all stocks: if this is true you can accept the solution; if this is false draw again ΔW_{i*} (maintaining the same chosen sign) and compute once more the other ΔW_i s. The algorithm should continue until we have an acceptable solution. The problem is that there are cases in which our system of inequalities has no solutions for ΔW_{i*}, at most for one of the two branches ($\Delta W_{i*} > 0$ or $\Delta W_{i*} < 0$): looking at the found solutions it is possible to see that this happens because $\min_i\{x_i^1, \Delta S\} < \max_i\{x_i^2, p_i\}$ (no positive acceptable solution) or $\min_i\{x_i^1, -p_i\} > \max_i\{x_i^2, -\Delta S\}$ (no negative acceptable solution). Therefore it is essential to fix a maximum number of trials in the determination of all ΔW_i s, that is a maximum number of times we draw ΔW_{i*}. If we reach such number of trials we change sign to $\Delta W_{i*}$and apply the same algorithm on the other part of the solution. This algorithm is clearly not sophisticated from a mathematical point of view but it is very strong from a computational perspective because it explores directly the set of all solutions (using the system of equations) for ΔW_i and then verify if the generated solution satisfies all the constraints.

- now it is possible to fix bid and ask prices using as a benchmark the corresponding $\Delta W_{i,t}^j$s.

We have not yet specified how actually rational investors get information from a consultant or decide which consultant they will call. Such choices will be illustrated later, first when we will explain the time frequency for this process, and then when we will speak about the rules in network generation.

3.1.6 Transitions from rational to random investors and vice-versa

The different behavior of rational and random investors is the basis of the heterogeneity in the model, and, at the same time, is one of the main sources for complex features in the market. However, in reality, the border between a rational and a random investor is not absolute. Irrationality can appear in some investors who usually used to rely on consultants; conversely, some random investors, who used to follow some self-made strategy, can choose to apply to financial analysts. So we can think about transition processes between the two classes of investors, which are important to make flexible and more realistic the evolution of agent actions in the simulation. Here we explain the two different transition processes we will include in the model; some technical details will be better understood when we will speak about the actual scheduling in the simulation.

From rational to random investor: herd effect One of the most interesting feature of a stock market is that an investor can be influenced in trading operations by the average behavior of the other agents in the market. Such phenomenon is called *herd effect* and can affect every investor, even a rational one, who founds her decisions on the information obtained by the consultant. A random investor operates with an unknown strategy, or simply, according to personal and irrational moods: so the behavior of this agent is totally unpredictable and we do not worry about herd effect because the actions of a random investor can yet simulate such phenomenon. So, in our model, this phenomenon is limited to rational investors, who can suddenly change their strategy according to the following scheme.

When an investor decides to enter in the auction of a certain asset, she looks at the book before actually making her order. In this way she can be irrationally influenced by what sees in the auction. For instance, a rational investor who is going to buy a stock and make a bid price, looks at the current status of the book and sees that on the selling side there are so much more orders than those in the purchasing side: many other investors are trying to sell that security and this can strike the rational investor, who, with

a certain probability, can overturn her investment decision making an ask price and trying to sell! (The same phenomenon can occur in the opposite situation with an investor who suddenly chooses to buy a stock which should be sold according her current information). This is the basis of the herd effect: a rational investor can follow the general trend in investment decisions, ignoring the rational strategy she is pursuing through the advice of her consultant [8]. We can think about this phenomenon as a chain reaction which starts when a side of the book is much longer than the other: some rational investors, who should make an order on the shorter side, are frightened and invert their order following the herd of investors. Here we have considered that this phenomenon is only dependent from the relative length (number of orders) of a side of the book with respect to to the other, but what is also important is the relative trading volume between the two sides (i.e. the relative number of all shares in the selling orders with respect to all the shares in the purchasing side of the book, or vice-versa). These two factors can trigger the transition from rational to random investor, since she will make an order on the opposite side of the book according to the investment decision rules of a random agent. It is clear how such process can lead to bubble or crashes, because when a side of the book becomes really prevalent on the other (in terms of relative length and volume) we have a reinforcement effect which tend to attract other rational investors, so increasing further the number of investors who are operating on that side.

Now we try to model such irrational behavior using a particular distribution which will depend on two parameters: the relative length (l_{side}) and the relative trading volume (V_{side}) of the dominant side of the book (w.r.t the other side). In other words we have to count how many orders are on both sides of the book and how many shares are included in the orders of both parts. When the rational investor decides to make her order (having yet set the amounts ΔS_t^j and all the quantities $\Delta W_{i,t}^j$), she has first of all to look at the book and check the current situation. We draw a positive random number (which will call ϵ) from the distribution which depends from l_{side} and V_{side}: if the number exceeds a certain threshold this means the investor was influenced, in an irrational manner, looking at the book and therefore the transition is occurred. The investor has to reset her initial order, making a new one on the opposite side of the book, respecting the usual rules for a random agent. A good distribution we can use for such random selection is a gaussian function of this kind

$$p(\epsilon) = N(0, \frac{l_{side} + V_{side}}{n_{base}})$$

The mean value of the normal distribution is 0, in order that the most likely values of ϵ are near zero and will not exceed the chosen threshold. The standard deviation is the sum of the two parameters, which capture the relative dominance of one side of the book on the other, divided by a rescaling factor; n_{base} is the minimum number of shares which every investor owns at the beginning of the simulation, i.e. is a lower bound in random initialization of the number of shares. The use of a rescaling factor is simply due to reduce the scale of the control parameter which manages the transition; in our work we have used $n_{base} = 1000$. Note that in order to have only positive numbers we simply compute the absolute value of the drawn ϵ value. Now the problem is what threshold we set for the transition test: we call the threshold as ϵ_{test} and we will see that this is a very important control parameter in the model. A high ϵ_{test} means that the transition of a rational investor towards the random class is less likely and then herd effects are less important. It is possible to compute the actual probability transition by herd effect

$$P(|\epsilon| > \epsilon_{test}) = Erf(\frac{n_{base}\sqrt{2}\epsilon_{test}}{2(l_{side} + V_{side})})$$

Plotting this function at different values of ϵ_{test} we get this figure

[8] See Kirman in *Complex Economics: Individual and collective rationality* (2011)

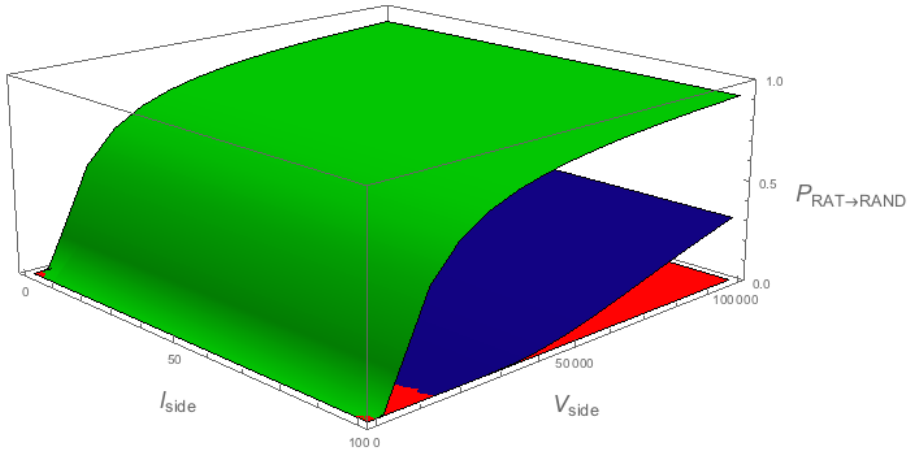


Figure 2: Plot for the transition probability through herd effect, as a function of l_{side} and V_{side}. The green surface is for $\epsilon_{test} = 10$, the blue is for $\epsilon_{test} = 100$, the red is for $\epsilon_{test} = 1000$.

We want to underline again that this transition is relative to a single stock and the transition test has to be executed for all stocks (with$\Delta W_{i,t}^j$ different from zero) when the rational investor enters in the auction (i.e. just before bidding or asking price): so it is possible that the test will result positive for some stocks and negative for other ones. It is suffice to have one positive test, namely for a single stock, to say that the transition is occurred and the rational investor has become a random agent; indeed changing the investment plan on a single security produces a considerable deviation from a strategy which aims to reach the correct new portfolio shares vector$\overrightarrow{w_t'}$ (or in other terms which aims to reach the portfolio goal). So we distinguish the two situations for the result of the transition test:

- when the transition test is negative for a stock, the rational investor continues to use the normal rules to make an order on the market (on the basis of the vector $\overrightarrow{w_t'}$). Assuming she has yet fixed ΔS_t^j and all the $\Delta W_{i,t}^j$s, the purchasing or selling order is normally computed through the ordinary technique.
- when the transition test is positive for a stock, the rational investor does not process an order according the normal scheme for her class. Indeed she will change all the $\Delta W_{i,t}^j$ values (corresponding to all the stocks for which the transition is positive) using the selection algorithm of a random investor. First $\Delta W_{i,t}^j$ will be automatically set as positive (negative) term if the opposite dominant side of the book is the bid (ask) side: then you can draw $\Delta W_{i,t}^j$ with the standard distributions used in the scheme of a random trader and considering all the relative constraints. This is repeated for all stocks which are positive to the change behavior test.

Once we have defined how the process occurs the question is how long the transition lasts. The transition is only temporary because is an irrational pulse of the moment. After having processed the order on the opposite side of the book, the rational investor will continue to follow the ordinary investment rules, even if she is going to invest again during the same trading day. At every operation the transition test is always performed, but the only thing that can actually change is the method which allows the determination of bid/ask prices with their corresponding volumes.

From random to rational investor: new clients for consultants It is possible that some random investors, performing their self-made strategies and following some momentary moods, can see their profits (in terms of their financial wealth) decreasing over time. This can lead a random investor to ask herself if it would not be better to take advice from an external specialist, in order to support their trading operations on the basis of some knowledge of the market. This implies a transition from the class of random agents to the class of the rational traders. Once again we have to construct some probability distribution to make a transition test, using some reasonable hypothesis. The transition phenomenon is driven by the number of trading days in which a random investor records a negative variation of her profits.

It is necessary to briefly present the definition of investors daily profits, before to illustrate the transition mechanism. An investor is characterized by her equity fund (i.e. the portfolio value) and by the total cash amount of money. The sum of both these variables is what defines the global wealth of our investor: then the daily profit is the relative variation of this sum, computed before and after the daily auction market.

$$\pi_t^j = \frac{(W_t^j + S_t^j) - (W_{t-1}^j + S_{t-1}^j)}{W_{t-1}^j + S_{t-1}^j}$$

where the time index t represents the end of trading day t, while the index $t-1$ can be seen both either as the end of trading day $t-1$ or as the beginning of trading day t (i.e. before the auction is opening). Investors daily profits are the important variables on the micro scale, i.e on the agents level, which are interested in.

Let us start denoting $n_{\downarrow days}$ as the number of *consecutive* days in which a random investor observes her profits decreasing; we have to define a distribution for a random number ϕ which determines the result of the transition test. So it is useful to fix again a threshold ϕ_{test}, such that if the drawn value of ϕ is higher than ϕ_{test} the transition is occurred and the random investor will *permanently* become a rational trader. This kind of transition occurs at the end of the trading day, when the random investor looks at the fall in her profits, and decides to change her approach to trading operations. Therefore, the following day, if the transition test is positive, the random agent will behave as a rational investor. We take a distribution similar to that used in the opposite kind of transition (rational to random)

$$p(\phi) = N(0, n_{\downarrow days})$$

As in the previous case if we increase the threshold value the transition is less likely and then ϕ_{test} is a relevant control parameter. Now let us compute the probability for this kind of transition, which is

$$P(|\phi| > \phi_{test}) = Erf(\frac{\phi_{test}}{\sqrt{2}n_{\downarrow days}})$$

and plot this function at different levels of ϕ_{test}

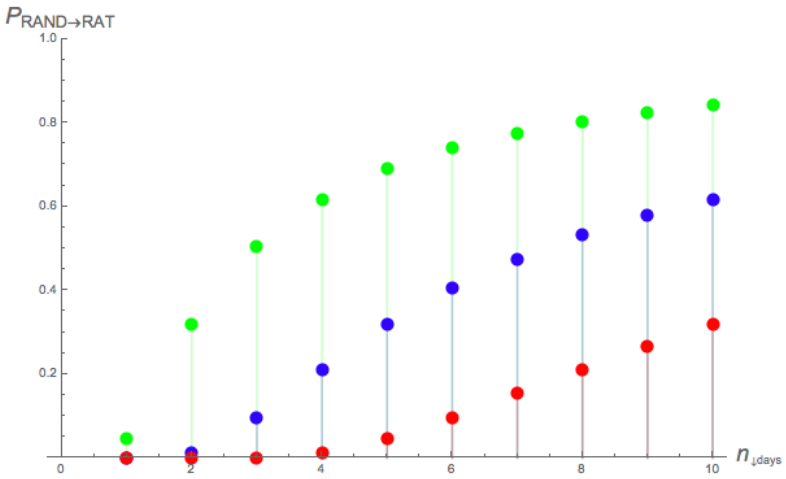


Figure 3: Discrete plot of transition probability from random to rational class, as a function of $n_{\downarrow days}$. The green points is for $\phi_{test} = 2$, the blue are for $\phi_{test} = 5$, the red are for $\phi_{test} = 10$.

When the transition occurs, at the end of the auction, the random investor will follow every investment rule (starting from the choice of ΔS_t^j) of a rational trader in following days. Let us highlight that, once the random investor has become rational, a temporary transition back to random class, as described above, is always possible.

Resuming some differences between the two transition processes (from now on we will define the transition rational to random as $RAT \to RAND$ and the inverse transition as $RAND \to RAT$)

1. the transition $RAND \to RAT$ is not limited to some asset but entails a change in investment rules on all security auctions; conversely the transition $RAT \to RAND$ is relative to single assets (though we assume the transition is occurred even if is relative to only one stock).

2. the transition $RAT \to RAND$ only changes investment rules about the determination of the corresponding $\Delta W_{i,t}^j$s: the choice of ΔS_t^j is however untouched. The transition $RAND \to RAT$ modifies instead all the investment process of an investor, both for ΔS_t^j and for all $\Delta W_{i,t}^j$s.

3. the transition $RAND \to RAT$ is permanent; the transition $RAT \to RAND$ has on the contrary an instantaneous effect.

4. since the transition $RAND \to RAT$ is permanent, the new rational investor (who was a random trader) can be affected from temporary transitions $RAT \to RAND$ during the auction, but her nature remains now intrinsically rational.

The transition mechanism of the kind $RAT \to RAND$ presented above does not consider the possible local interactions among investors. When we will introduce the rules for the emerging network we will see other rules which can make possible these transitions, on the basis of the current links in the market.

Consultancy frequency for rational investors Rational investors, as we have seen, can access directly to the optimal portfolio goal shares w_i'. Before the beginning of each trading day (i.e. between the closure of the auction of the previous day and the opening of the new one) the investor can call a consultant and pay her to get such values. However, after the first tick in the simulation (the first trading day), a rational investor could decide to retain the values of w_i' used in the past auction (or auctions) and not consult, for some time, an analyst. This can reflect the fact that the rational investor do not desire to pay every day a specialist because this can be too expansive. In addition, forecasting methods used by a consultant are based on a statistical sample which is not so enriched from one or very few trading days. So we have to construct a mechanism for which rational investors decide to call a consultant or, conversely, decide to use the same w_i' to perform new orders in the auction. The question is subtle because, as we have discussed, the portfolio of the investors could be very far from the optimal one. Maintaining the same vector of portfolio goal shares, the rational trader can try to continue her investment path without changing the optimal portfolio composition.

The choice to ask consultancy or not has to be modeled again in a random manner because is personal, even talking about rational investors. There are two driving factors in such choice: from one hand we have to consider the price to pay for every consultancy operation, which is the same for all consultants and is described by a control parameter named as cost of information (p_{info}); from the other side we have the number of consecutive days for which the investor has used the same portfolio goal share vector, a value which we denote as $n_{\vec{w'}}$. As we have done before in the case of transitions between the two classes of investors, we can construct a test to verify if the rational investor has decided to consult the analyst or not. So let us define a distribution to draw a random number λ and compare it with a certain threshold λ_{test} : if $\lambda > \lambda_{test}$ the test is positive and the investor will call a certain consultant.

$$p(\lambda) = N(0, \frac{n_{\vec{w'}}}{p_{info}})$$

The gaussian distribution shown above has a standard deviation which increases in the number of trading days passed with the same values of portfolio goal shares, while decreases with respect to the cost of the consultancy price. Since we want only $\lambda > 0$ it is suffice to consider the absolute value of λ. The choice of the threshold value λ_{test} is once again a parameter control of the simulation and it fixes the probability in asking consultancy for rational investors. If we compute this probability we get

$$P(|\lambda| > \lambda_{test}) = Erf(\frac{\lambda_{test} p_{info}}{\sqrt{2} n_{\vec{w}'}})$$

where we can explicitly see how p_{info} affects negatively the probability for a new consultancy, while $n_{\vec{w}'}$ has a positive effect on it: the following plot makes clear the situation

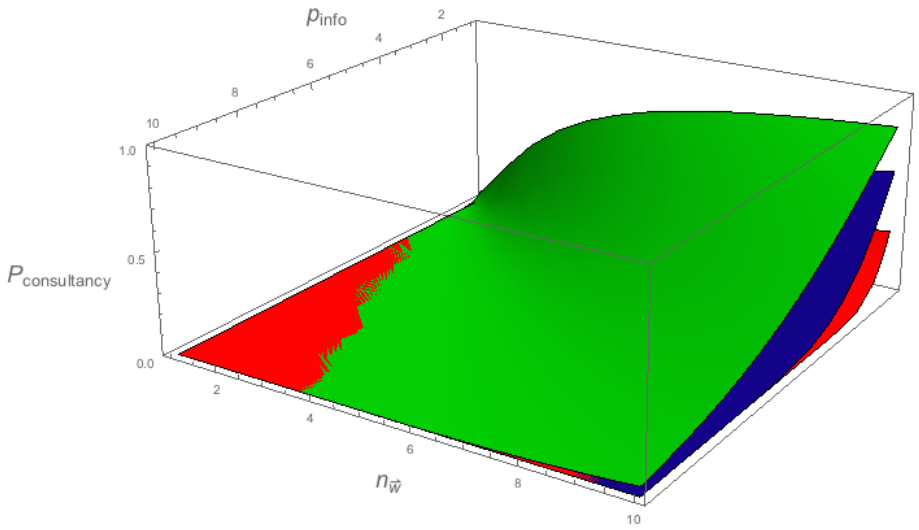


Figure 4: Plot for the probability of new consultancy, as a function of $n_{\vec{w}'}$ and p_{info}. The green surface is for $\lambda_{test} = 2$, the blue is for $\lambda_{test} = 5$, the red is for $\lambda_{test} = 10$.

The next question is to define how investors can choose which particular consultant they will call and how they can change consultancy during the simulation. Introducing the idea of the emerging network in the system we will also deal with such topic.

Budget investment constraints for investors Now we deal with a question not yet faced: what are the budget constraints for an investors in our simulation? This problem is very delicate since the freedom in investment operations for a trader, with respect to her budget, affects the interactions with the other traders in the market. In this model we assume that *no one investor can borrow money to continue to invest when her cash account S^j is no more sufficient to perform trading operations.* We have seen how the amount of money ΔS_t^j used at each trading day (as *ideal* threshold) to buy, sell or short-sell stocks (or at most trying to do that), is set (for both rational and random investors) on the basis of the current portfolio value W_t^j and does not depend on the cash account S_t^j. However if $\Delta S_t^j > S_t^j$ it is obvious that the investor cannot find such amount of mone during the new trading day. In this situation we set automatically$\Delta S_t^j = S_t^j$, so using up all the cash account available to the investor. Afterwards the investor will continue to follow the usual investment steps.
Anyway another important budget constraint is related to the definition of ΔS_t^j itself: we know that ΔS_t^j does not fix an upper bound for the total expense during a trading day, but only a limit in the magnitude of every order. So it is possible that, even if $\Delta S_t^j < S_t^j$, the trading operations determined by our selection algorithm will require a higher amount of money with respect to the current cash account S_t^j. This can occur within the same trading day: an investors begins with $S^j > 0$ and then, some purchasing (or short-selling) operation requires more money than the current S^j. In this case all operations which cannot be executed are not included in the auction: this is done, from the point of view of our simulation program, inside the electronic book system, which applies a filter to all orders and takes into account the books of all securities (because we have to monitor the total amount of money required to perform trading operations). The

choice to not include this constraint in the investment decision guarantees a good flexibility in the selection algorithm for all $\Delta W^j_{i,t}$s: we maintain (for both random and rational investors) the usual investment decision steps and we eliminate only afterwards the orders which are not allowed. This, in some sense is equivalent to an investor who makes an ideal investment plan and then understands how some orders cannot be possible: so she decides to drop them because cannot sustain such investments.

However selling operations are always possible and thanks to them the account S^j can come back again to a higher value which allows to buy or short-sell stocks again. Finally we note that an ABM market can simply assume that investors can borrow money without any limit, since only profits (which are relative variations of market gains and cash account) does matter. The introduction of constraints on investment operations is anyway useful to make more effective the simulation in terms of applications to real trading.

3.2 Consultants

The role of consultants in the simulation is to try to make reliable forecasts about future asset prices, on the basis of data relative to past prices. At the beginning of the simulation every consultant has yet a series of data for all stocks: this can given according to real market data, or on the basis of some stochastic process to model the past trend of every asset. However our purpose is to create a market in which there a great heterogeneity in expectational models used to process data and this corresponds to create many different kinds of consultants, who operates through different learning mechanisms and adopt various strategies to make forecasts. All types of consultants exploits a series of data emerging from the simulation dynamics (the past asset prices), but when the simulation starts there are obviously no data. So, we used external data series taken by real market archive to have a realistic sample from which any consultant can begin her analysis: we denote such series as $\{p^i_{t_i}\}_{t_i=-t_0,\dots,0}$, where the time is considered as negative since we fix the time origin when the simulation starts, and i is the index over the stocks. Once the simulation begins, a new asset price value (first tick in the simulation) emerges and this is added to the initial exogenous sample; in this way, at time t, the new data determined through the simulation join the first external series, producing the final data series $D_t = \{p^i_{t_i}\}_{t_i=-t_0,\dots,t}$. We will discuss in more detail the exact origin of our external (and real) samples in the paragraph relative to the simulation settings. It can be crucial to highlight that consultants take, for each trading day, the closing price at the end of the auction (which in our model coincides with the opening price of the following day).

In the ABM we call n_{consul} as the relative fraction of consultants in the market *with respect to the total number of rational investors.* The choice to consider only rational investors in computing such value is due to the fact that consultants have an impact only over rational traders. In addition, if some transition $RAND \to RAT$, the number of rational investors actually change and the fraction n_{consul} so varies over the simulation.

3.2.1 Portfolio choice framework

What kind of information a consultant has to get from the analysis of past asset prices data? The answer is related to what type of portfolio optimization framework we adopt in the model. As final aim, the consultants have to provide the investors the vector $\overrightarrow{w^j_t}$, which defines what are the objective changes in portfolio shares. The optimization process is related to the maximization of the utility function of every investor, therefore the problem is to decide which utility function we can use in our model. First of all we realize that a myopic approach in terms of portfolio choice is all we need for our simulation. In our work indeed a consultant has to elaborate, at each trading day (we can define this process as performed between the closure of a daily auction and its opening the next day), all available information to get the

vector $\overrightarrow{w_t'}$; so the investment horizon is only the following trading day and we are not in presence of a multi-period investing framework. The myopic element in portfolio optimization decisions is a consequence of the dynamics we have chosen in the model, but we could extend the work giving investors the possibility of invest over long periods of time. Anyway a rational investor wants to take advice from her consultant during each trading day, having as personal goal the maximization of her portfolio value at the end of the day (the outcome of the optimization process is evaluated when the auction is closed). In this framework it is reasonable to choose a power utility function, since it is the most realistic from a risk-aversion perspective: we report the expression for such function

$$U_t(W_t^j) = \frac{W_t^{1-\gamma_j} - 1}{1 - \gamma_j}$$

With this utility function the degree of relative risk-aversion γ_j is just equivalent to $-\frac{WU"(W)}{U'(w)}$, as in the standard portfolio choice theory. The important properties of such utility function is that the degree of relative risk-aversion is independent from wealth, while the coefficient of absolute risk-aversion $(-\frac{U"(W)}{U'(w)})$ is decreasing in wealth: these two aspects are essential because they well model the behavior of an agent in an economic system. The use of a power utility function in the portfolio optimization problem requires the assumption of log-normally distributed returns for every asset. Such assumption is enough realistic, though it presumes that consultants know the a priori distribution for asset returns. However we have to remember that the model is investigating a complex system, for which the asset return distribution is unknown and, in fact, we are willing to discover, through the simulation, what kind of distribution emerges from market dynamics. In other terms assuming that asset returns are log-normally distributed does not mean that assets will show actually such distribution. we are simply making a reasonable hypothesis to fix the optimization method. The next question could be the following: can an a priori distribution for asset returns in portfolio optimization directly influence the actual emerging distribution? Our work is intended to answer to this question. *We expect anyway that the complex interactions in the market are the main drive for the actual distribution for asset returns, while the single choice in optimization problem are less relevant in this context: this is one of the core topic of this financial economic system.* If we want to get the optimal portfolio we have to maximize the expected utility, or equivalently, the logarithm of the expected utility

So the optimization problem can be expressed by

$$max[logE_t(U_{t+1})] = max[logE_t(\frac{W_{t+1}^{1-\gamma_j}}{1-\gamma_j})] = max[logE_t(W_{t+1}^{1-\gamma_j})]$$

where we dropped the scale factor $(1 - \gamma_j)$ because does not influence on the result. If the financial wealth is also a log-normally random variable, the following expression is valid

$$logE_t(W_{t+1}^{1-\gamma_j}) = E_t(logW_{t+1}^{1-\gamma_j}) + \frac{1}{2}var_t(logW_{t+1}^{1-\gamma_j})$$

because the product of a lognormal variable powered to some value is again a lognormal variable. The maximization problem is

$$max[E_t(logW_{t+1}^{1-\gamma_j}) + \frac{1}{2}var_t(logW_{t+1}^{1-\gamma_j})] =$$

$$= max[(1-\gamma_j)E_t logW_{t+1} + \frac{1}{2}(1-\gamma_j)^2 var_t(logW_{t+1})]$$

and dividing by $(1-\gamma_j)$ we find

$$max[E_t w_{t+1} + \frac{1}{2}(1-\gamma_j)var_t(w_{t+1})]$$

where the lower-case letters mean the logarithm of the corresponding quantities. It is useful to express again the problem through the budget constraint $w_{t+1} = r_{p,t+1} + w_t$, with $r_{p,t+1} = log(1 + R_{p,t+1})$ as the continuously compounded portfolio return (*here we are considering the complete portfolio, i.e. the portfolio which also presents the risk-less asset*).

$$max[E_t(r_{p,t+1}) + \frac{1}{2}(1-\gamma_j)var_t(r_{p,t+1})]$$

This corresponds to a mean-variance problem in portfolio optimization and this is clear reminding that $E_t(r_{p,t+1}) = logE_t(r_{p,t+1}) - \frac{1}{2}var_t(r_{p,t+1})$: substituting such expression in the argument to maximize we have

$$max[logE_t(r_{p,t+1}) - \frac{\gamma_j}{2}var_t(r_{p,t+1})]$$

which clearly defines the usual trade-off between expected return and variance of the investor's portfolio. The solution of this problem when we have one risk-free asset and more risky assets is mathematically expressed as

$$\overrightarrow{w'_{p,t}} = \frac{\Sigma^{-1}(\vec{E_t}(r_{t+1}) - r_{f,t+1}\vec{1} + \frac{\vec{\sigma_t^2}}{2})}{\gamma_j}$$

where $\overrightarrow{w'_{p,t}}$ is the vector of stock weights *for the complete portfolio (and not for the risky)*. Indeed the scale of such weights depends on the degree of relative risk-aversion of the investor j, while the relative composition among stocks is the same for every investor. Σ is the covariance matrix of asset log returns in the market, $\vec{E_t}(r_{t+1})$ is the vector of expected asset log returns and $\vec{\sigma_t^2}$ is the vector of the log return variances (i.e. the vector with the diagonal elements of Σ). Since our work is intended to investigate the risky portfolio of investors and not its complete version, we would like to get the optimal values for the vector $\overrightarrow{w_t}$, which incorporates the weights of all stocks in the risky portfolio. The solution is quite simple and here we immediately present the result[9] (in our portfolio choice framework)

$$\overrightarrow{w'_t} = \frac{\Sigma^{-1}(\vec{E_t}(r_{t+1}) - r_{f,t+1}\vec{1} + \frac{\vec{\sigma_t^2}}{2})}{\vec{1}\Sigma^{-1}(\vec{E_t}(r_{t+1}) - r_{f,t+1}\vec{1} + \frac{\vec{\sigma_t^2}}{2})}$$

The terms w'_i in the vector $\overrightarrow{w'_t}$ are just the values that a rational investor wants to get applying to a consultant. So we can finally answer to the initial question: what kind of information a consultant has to get from data series? To perform the mean-variance myopic optimization consultants have to compute, at the end of the trading day (i.e. when the auction is closed), the expected returns for all stocks and also the covariance matrix (which photographs the risk of every security and the correlations among returns), for the next day (i.e. when the new daily auction will close). This task is reached using different expectational models, or in other words, applying different asset pricing methods. First a consultant uses her model to compute the expected returns and then such values are used to get the covariance matrix, according to the ordinary statistics expression. This is equivalent to say that:

[9] For details see Campbell et al. in *Strategic Asset Allocation: Portfolio Choice for Long-Term Investors* (2001), Chapter 2: 25-29

- the expected returns are computed in different ways according to the kind of asset pricing model used from the consultant;
- the covariance matrix is obtained in the same way for all consultants, though it is derived from the expected returns and then is biased on the basis of the expectational model chosen to make forecasts. We briefly remind the expression for a term in the covariance matrix (knowing the expected returns), computed at time t:

$$\sigma_{ij,t} = \frac{1}{t - k_0 - 1} \sum_{k=k_0}^{t} (r_k^i - E_t r_{t+1}^i)(r_k^j - E_t r_{t+1}^j)$$

  Here k is the index which runs over time and starts from the beginning of the considered sample; i and j define the two stocks for which the covariance is evaluated. If $i = j$ we obtain the variance term for the stock i. This expression is the sample covariance because we simply use the past values in the sample (and the expected returns previously computed through some model) and we do not need to know the probability joint distribution for the two stocks (that would be impossible!).

It is important to note that a mean-variance (Markowitz) approach to portfolio optimization can be considered limited and too simple because we are ignoring further moments in asset returns distribution. However assuming that all the consultants in the model use such approach to compute the portfolio goal is anyway a good hypothesis because the complexity in the model arises from many other factors, as we have yet announced previously. Since the optimization process is implemented through log returns, in the following data processing operations performed by consultants we will always have to compute log returns.

Finally we can describe the different classes of consultants we include in the model. Every class is composed by agents who used the same type of expectational model to forecast future asset return. There are so many possible choices and extensions to the kinds of agents here presented, but we have chosen to include only consultants who perform *technical analysis*, i.e. who exploit just price series to make forecasts. This is quite natural in a Agent-Based simulated stock market because we can easily monitor stock prices evolution, but the knowledge of external (real or simulated) *fundamentals* for pricing assets is really another problem. Of course we could think to build a dynamic model that reproduces the evolution of some macroeconomic variables able to affect the *fundamental value* of the securities: this is very challenging because we would link our simulated market to some other stochastic dynamic model and this is not the purpose of this work. So *fundamental analysis* is not covered by our classes of consultants.

Standard analyst A *standard* analyst is a consultant who implements the most trivial asset pricing method: using the available sample she computes the expected log return of every stock as the sample mean, so getting the most simple estimator to apply Markowitz analysis. Obviously, from a machine learning point of view, the sample mean is a very weak estimator to make forecasts, since it represents just the average value over the available data set and does not catch any possible trend or feature in asset returns. In general, the sample mean can be a useful estimator when we are considering data which follows a symmetric distribution. If, as we did in our portfolio choice framework, we assume that returns are log-normally distributed, then the log returns are just normally distributed and the mean is a good value to represents the possible data set. However the assumption of log-normally distributed returns is just a hypothesis introduced to reach mathematical integrity and it needs to be tested *a posteriori* from the simulation.

Chartists A *chartist* is a consultant who implements an econometric model to forecast future asset log returns, on the basis of the entire sample of past asset prices. The econometric models used by *chartists* are dependent only by past values of the stock prices and do not depend from any other economic variable: this is equivalent to apply only technical analysis. We can introduce many stochastic process to model security prices evolution, but in our work we have included maybe the most simple class of time series model, which is the AR process, which is very effective and simple to implement technical analysis. Therefore it can be interesting define three classes of consultants, which one with a different order in the AR process used to price assets.

Chartist AR(1) Let us start from a *chartist* who uses an AR(1) process as expectational model

$$AR(1) \Rightarrow r_{t+1} = \alpha r_t + \gamma + \eta_{t+1}$$

where α is the autocorrelation coefficient, γ is the drift term, while η_{t+1} is a white noise process with variance σ_η^2. Noise terms are distributed according to a normal distribution (i.e $\eta_{t+1} = N(0, \sigma_\eta^2)$) and, though normality for asset log returns is rejected as we have previously repeated, this assumption works well in evaluating portfolio performance for a mean-variance investor. How does this type of *chartist* analyst exploit this model to make forecasts? At time t (and more precisely at the end of the trading day denoted by t) she uses the data series of the stock i, $\{p_{t_i}^i\}_{t_i=-t_0,\dots,t-1}$, to perform a linear regression (OLS technique) through one of the AR processes: in this way the estimates of α,γ, and of the variance σ_η^2 can be easily obtained. The choice to not include in the data sample the latest log-return, i.e. r_t, is due to learning theory. Indeed we are interested about the expected log returns of the stocks, computed with the following expressions

$$AR(1) \Rightarrow E_t r_{t+1} = \alpha r_t + \gamma$$

Here we can immediately see how the AR(1) process is a Markov process (*it has no memory*), because $E_t r_{t+1}$ does not depend on r_{t-1} or previous log returns. The estimated values of α and γ are computed taking into account the penultimate log return in the data series (i.e. that one at time $t-1$): so it is possible to apply such model to the last log return of the sample (i.e. at time t) in order to get the expected log return for the next trading day, denoted by $t+1$. If we used all the data sample $\{p_{t_i}^i\}_{t_i=-t_0,\dots,t}$, considering also the last value of the series, we would break a very important notion of the Machine Learning theory: we cannot use a data contained in the training set (which is the data set used to compute the estimators of the parameters α and γ) to test the model. So training set and test set cannot overlap, i.e. in-sample performance has no coincide with out-of-sample performance. Obviously we are interesting in the second one: a significant forecast is made out-of-sample and then we have to evaluate the out-of-sample performance of our model. Such theme is faced in the Appendix regarding Machine Learning[10] .

Chartist AR(2) A more complex model, just in the sense of learning theory, is obtained with an AR(2) process. The idea is to consider the second lagged value for log returns

$$AR(2) \Rightarrow r_{t+1} = \alpha r_t + \beta r_{t-1} + \gamma + \eta_{t+1}$$

where α and β are the autocorrelation coefficients, with respect to the two lagged log return values, γ is the drift term and η_{t+1} is the white noise gaussian process with variance σ_η^2. While the AR(1) process is

[10] As main reference look at Yaser S. Abu-Mostafa, Malik magdon-Ismail, Hsuan-Tien Lin (2012) *Learning from data*

a Markov process, the AR(2) process is a stochastic process with *memory*: indeed the expected log return value is

$$AR(2) \Rightarrow E_t r_{t+1} = \alpha r_t + \beta r_{t-1} + \gamma$$

and depends not just by the last value of the process (i.e. r_t), but also by the penultimate one (i.e. r_{t-1}). As in the previous case we can get the estimates of α, β and γ through linear regression. In addition, once again, we have not to overlap training set and test set: so, for an AR(2) process it needs to use a training set without the last two log returns in the series since these values will be used to make the forecast for the upcoming trading day.

Chartist AR(3) Finally we consider a chartist who exploits an AR(3) model

$$AR(3) \Rightarrow r_{t+1} = \alpha r_t + \beta r_{t-1} + \delta r_{t-2} + \gamma + \eta_{t+1}$$

where α, β and δ are the autocorrelation coefficients, with respect to the three lagged log return values, γ is the drift term and η_{t+1} is the white noise gaussian process with variance σ_η^2. OLS technique ensures the estimation of all parameters α, β, δ and γ. The expected log return for the next trading day is

$$AR(3) \Rightarrow E r_{tt+1} = \alpha r_t + \beta r_{t-1} + \delta r_{t-2} + \gamma$$

showing that this is not a Markov process. In this case we need to exclude from our training set he last three log return values to make reliable forecasts.

All these three AR(q) models used by our chartists are assumed to be *weakly stationary*, i.e. they have finite and time-independent unconditional mean and unconditional variance, while the auto-covariance depends only on the lag time[11]. This implies that the autocorrelation coefficients have to be strictly smaller than one:

$$|\alpha| < 1 \wedge |\beta| < 1 \wedge |\delta| < 1$$

and this can be naturally verified in the simulation, as done in Chapter 8. The relevant question is however which of these AR models can better perform in making forecasts. Using the estimates of expected log returns, every consultant will build an optimal portfolio and we are interested in evaluating the performance of such portfolios. As always reminded in the course of our work, the optimal portfolio will be difficulty reached by the investors since the traders operate in a complex interaction environment, dominated by the double auction system. In Chapter 7 we analyze the performances of portfolios created by the four kinds of consultants, first of all computing out-of-sample error of the forecast log returns. Machine Learning notions will lead us towards weaknesses and strengths of all the models.

3.3 Emerging network rules

Once we know all agents features and behavior rules, we have to establish some basic mechanisms that define how the informational network emerges in the system. When the simulation runs how do the links among agents are set? Links are social interactions between two agents and can be seen as an information trade. We distinguish two kinds of links in our temporal network: those between two investors and those

[11] See Marno Verbeek (2004) *A guide to modern Econometrics*

between a consultant and an investor. So, in other words we can think about two separated networks to be analyzed in different ways and with different properties.

In both kinds of network agents interact most of the time on the basis of investors daily profits. The mechanism of network generation has then to look at these variables, since they are the reflection of both personal trading strategy and stock market behavior. We shall analyze in quantitative way the features of daily profits in Chapter 8, also making a parallelism with network properties of all investors/nodes.

3.3.1 Network with links investor$\longleftrightarrow$investor

Investors can interact among each other in order to exchange information about the market. With information about the market we intend any possible suggestion, available data or opinion relative to the securities in the market. In other words the term information has not to be confused with *rational information* since the latter is just a minimal part of the overall information available to the agents through the contact with other investors.

Boundary conditions for initial network generation The main idea is that a single investor starts the simulation being acquainted with some other investor in the market. When we make start the simulation we select, for each investor, her acquaintances (beside her consultant) in a random manner. There are many possible initialization algorithms but we choose one which mimics the social phenomenon in setting the first contacts in the market.

The main intuition is that every investor *can* enters in the market through some initial acquaintance (or acquaintances) who is (are) yet operating on that market. So there are some default initial neighbors for every investor: this number can be comprise between zero and a maximum threshold chosen by the modeler. If the default value is zero this means that the agent has decided to enter in the market from its own and not because she was influenced by other investors. On the opposite there can be multiple investors who have convinced the agent to start to invest. Anyway the default number of initial neighbors is just a portion of the actual initial number of neighbors (i.e. the actual starting degree). Indeed, since every investor uses the same mechanism to become part of the market, there is the possibility that an agent with its default neighbors can convince another agent to enter in the market, so augmenting its default degree. The entire mechanism is common to many social phenomenon in which we have to set initial edges, simulating the entry in a new social or economic system.

Now we present the algorithm for network initialization, taking in mind the phenomenon above illustrated. The number of default initial acquaintances is set as a random number between a zero and a maximum threshold, with uniform probability: in our work we set such threshold to one, with the meaning that only one investor can convince an agent to enter in the market. Nevertheless the possibility of zero default neighbors has always to be included in the model. Once that you have drawn the actual number of default acquaintances, if this number is not zero, choose randomly (every time with equal probability) the default neighbors. Since you are repeating the process for all investors it is possible that setting an agent who has yet set her default initial degree can become one of the default neighbors of another agent. This, as previously explained, implies the possible extension of initial neighbors with respect to the default ones; for instance, an agent with zero default contacts could convince a new agent to become an investor, so having an initial degree no more equal to zero. Once we have drawn which is the actual number of default neighbors, denoted as k_0, it is possible to compute the initial degree distribution, i.e the probability that an investor has a degree k at the beginning of the simulation. First the probability of setting k_0 default neighbors is

$$p_0^{default} = \frac{1}{k_{0,max} + 1}$$

where $k_{0,max}$ is the maximum number of default neighbors allowed in the informational network: in our work $k_{0,max} = 1$, then $p_0^{default}(k_0) = \frac{1}{2}$. Now you can compute the initial degree distribution, which is a conditional distribution on k_0 itself: it simply defines all the contributes of other investors to become default neighbors and coincides with a binomial distribution

$$p_0(k|k_0) = \begin{pmatrix} M-1-k_0 \\ k \end{pmatrix} (\frac{1}{M-1-k_0})^k (1 - \frac{1}{M-1-k_0})^{M-1-k_0-k}$$

where the single event probability is $p = \frac{1}{M-1-k_0}$; indeed this is the probability that an agent in the network can be chosen (in uniform way with respect to all the other traders) as a default neighbor, so contributing to the degree of the investor who has chosen her.

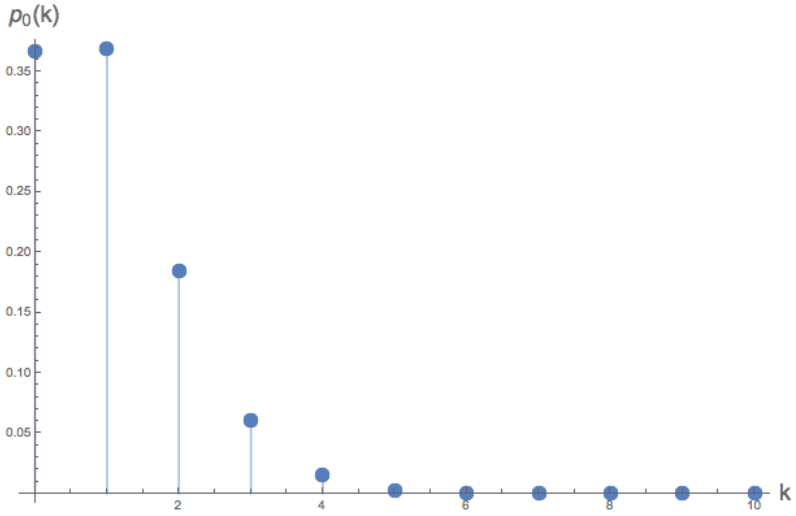


Figure 5: Theoretical initial degree distribution with $k_0 = 0$ according to our initialization algorithm.

Finally it is possible to compute the expected value and the variance of this distribution since we are simply considering a binomial distribution

$$E(k|k_0) = \frac{1}{M-1-k_0}(M-1-k_0) = 1$$

$$var(k|k_0) = \frac{1}{M-1-k_0}(M-1-k_0)(1 - \frac{1}{M-1-k_0}) = \frac{M-k_0}{M-1-k_0}$$

Network evolution algorithm How these links are set during the simulation? The investors could want to increase their acquaintances in order to have more information and the intermediaries in such process are just the investors who are yet acquainted with them. Indeed, in terms of network theory, two investors who know each other are two neighbor nodes in the network (this is equivalent to say there is a link between them). During the simulation an investor can then set new links with the investors who are

acquainted with her neighbors (the neighbors of her neighbors). This mechanism is not random but depends on market dynamics itself, in a sense that we explain now. So we are adopting an algorithm for the network evolution which is not defined *a priori*, as usually happen in many network models and this is the great novelty in such work. The ABM has a specific dynamics which controls how the network evolves over time. Before to present the generation algorithm, take into account the most important assumption in all such mechanism: *an investor wants to increase their acquaintances not in a random manner, but considering only the investors who have realized better daily profits in the market, since this is the most clear and simple criterion to evaluate traders reliability.* Now we resume the steps of the algorithm for network evolution, which occur at the end of every trading day and are applied to any investor in the market:

- first an investor decides if she wants to increase her acquaintances. This choice will depend by her recent portfolio performance and by the performances of her neighbors. If all her neighbor traders will have get worst performances (even if positive) during the day, the investor will not need to augment her information network, i.e. she will not search a new acquaintance in the market through one of its neighbors: the algorithm can be stopped. If, on the contrary, one or more of her neighbors will have performed better with their portfolios, the investor will choose the neighbor with the best daily profits, which is called the *prototype neighbor*: looking for the best profits means looking for the higher reliability. The *prototype neighbor* will become the intermediary to search a new reliable acquaintance in the market.

- next, the investor looks for a new edge in terms of information sharing and she wants to consider once again only investors who have performed better than her in the last trading day. Look at the list of neighbors of the *prototype neighbor* and consider only those who have realized better profits than our investor and that are not yet among her neighbors: indeed it is possible that some neighbor of the *prototype neighbor* is yet a neighbor of the investor (a clustering effect). The new acquaintance will be the trader with the best profits and you can set a new link with her.

- once you have set the new link prune the old edge with the worst of your neighbor, i.e. the neighbor who have realized the worst daily profits. This operation represents the idea of an updating of your informational network. New reliable acquaintance can bring new information about the market and instead those who have poorly performed can be discarded from your set of useful acquaintances

We are simulating the common phenomenon for which an acquaintance of ours becomes the intermediary of a new interaction, for example because she suggests us a new person to contact to get additional information about the market. However this is not a random process but we assume is driven by the daily profits in the market, which are of course related to the global emerging market dynamics. So finally we can understand why this is not an *a priori* network model: the evolution of our informational temporal network is strongly dependent by personal trading strategies of the investor (which can determine their profits) and by asset dynamics in the market.

3.3.2 Network with links investor$\longleftrightarrow$consultant

The other kind of edge in the network occurs between a rational investor and her consultant. At each trading day a rational investor decides if she seeks the advice of some consultant or continues to perform trading operations with the same portfolio goal. When she calls a consultant we can imagine that a new link appears in our network. We have not yet explained how an investor can choose which specific consultant to call. At the first tick of the simulation this decision is completely random because no one rational investor has particular information about the work of the consultants. The first call is then performed drawing one

of the consultants in the market with uniform probability: so, in other words, we have a uniform prior about analysts work and this is extended to all several categories of consultants. This first choice sets the initial state of the network, for the links investor-consultant: it can be important to understand that such initial state of the network is completely random and does not depend on any possible data series or other agents features.

After this first decision, rational investors can continue to use the same vector $\overrightarrow{w_t'}$ for some days, but, sooner or later, they will want to ask new information to consultants, in order to update the optimal portfolio goal for their trading operations. Then they can get new information from the same consultant or call a new one in the market: it is quite easy to understand that such choice is based on the recent portfolio performance. When the investor decides to search new consultancy information (the test for λ is positive), she looks at the profits made in the previous day until the last consultancy and she compare them to the profits of the neighbor investors. In particular she will check if one of her neighbors has an *additional* value equal or higher than a relative fraction x_{change} of her own daily profits; for example $x_{change} = 1$ means a trader will consider neighbors who have realized two times her daily gains. If exists one or more neighbors with these features, the investor can make two different decisions (note that if there are more neighbors with better profits we choose obviously that one with the higher portfolio performance):

- if the neighbor investor who has obtained the best profits is also a rational investor, she will choose the consultant of her acquaintance to obtain the new $\overrightarrow{w_t'}$ for the upcoming trading day. However this mechanism is not applied in one particular case: if at the end of the first trading day with the new $\overrightarrow{w_t'}$, the investor has not completed all her orders, than the portfolio performance cannot be correctly evaluated and the change of consultant for the next day is not allowed (but anyway there is the possibility of a new consultancy from the same analyst). In other words it is not possible to change consultant after only one trading day if all orders are not processed.

- if the neighbor investor who has obtained the best profits is a random agent clearly the investor cannot change consultant because her neighbor has no used any consultancy. In this case, instead, the rational investor can become a random trader, but not following the rules presented above for transitions due to the factors l_{side} and V_{side}. Indeed we are simulating the possibility that a rational investor could be influenced by the better portfolio performance of some acquaintance, who follows a random strategy, and decides to *definitely* stop her rational behavior. So this is, in contrast to the transition $RAT \rightarrow RAND$ we have seen before, a *permanent* transition to random investor class. However the transition is not stochastic as in the other case but is only related to the existence of one or more neighbor random investors with additional profits equal or higher than x_{change} of her own profits. If there are one of more neighbors with such features the rational trader will become a random agent immediately. Of course it is possible to build another scheme with stochastic transition (for instance on the basis of the specific profits of all the neighbor random traders) but here we decide to apply a simple *logistic* transition awarding a straightforward meaning to x_{change}.
  Following the rules in our ABM we remind that, as a random investor, usual transitions $RAND \rightarrow RAT$ are again possible in future trading days.

A link between the investor and the consultant remains fixed until the investor changes analyst, even if she is using the same $\overrightarrow{w_t'}$ for several trading days. When the consultancy edge is modified, the old link disappear and a new one will appear in the network, involving the new consultant. Moreover we note that transitions of the kind $RAND \rightarrow RAT$ are not driven by other investors performances or choices. A trader who invests on the basis of personal strategies or moods will decide to become a rational agent only as personal choice. So it is more simple to influence a rational investor to become a random one!

It is very important to underline that a link between a consultant and an investor is only temporary, while a link between two investors, once it is set, is permanent in the simulation. Indeed we assume that an investor will continue to exploit all her acquaintances during the simulation in order to achieve the best possible degree of information about the market.

3.4 A final discussion on the model: which are the novelties with respect to other ABM markets?

Now that we have illustrated all the elements in our work it is important to fix its position in the context of past and recent ABM markets, always considering the main topics which characterize such class of simulations. It can be useful, in particular, getting a look to the SFI market and compare its features with that ones introduced in our ABM.

First let us deal with agents classification: the choice to create well separated classes of agents it is very efficient both from theoretical point of view and on the programming framework. The SFI market considers just many traders who used several expectational models and there is no division between agents who process data and make forecast and agents who directly invest on the market. If we separate the task of optimization and forecasting (which concern consultants) from actual trading operations (relating to investors) we get a more realistic framework. Indeed in reality investors difficulty have the proper knowledges about the market to process data, and, as a further complication, there is a considerable cost of time in doing such operations. So most of the time, traders will decide to consult some specialized analyst who works for example for an asset management business or for a bank. A rational investor is then some one who decides to trust in a financial advisor and pays her to get a strong guideline for trading. In the SFI stock market model instead all investors build and manage a pricing model to decide their asset demand and this is totally acceptable to explore many theoretical problems in such a complex environment, but is very few realistic. So we propose this new scenario considering the consultants as auxiliary agents who just analyze data and make forecasts. What about the existence of two classes of investors with opposed investment strategy? Random traders, as repeated many times, are essential as counterparts of rational investors, but their existence is also due to verify if, in a market in which many traders operate with self-made or irrational strategies, there is the chance for those who trade with a rational scheme to make significant profits, i.e to *beat the market.* Therefore you can think about the role of rational traders as a sort of background to make more realistic the market itself (indeed they are usually referred as *noise* traders): in a real stock market the most part of investors are actually trading without the use of an econometric model to make quantitative forecasts and this is another point which supports our work. Finally the different classes of consultants used in this ABM surely represents a too poor collection, with respect for example to the wide ecology of expectational models used in the SFI market. However the fact that even with the same portfolio goal there are infinite ways to operate on the market, helps to reduce the problem; the number of expectational models are limited but the actual trading strategies can be very different, so leading again to heterogeneous expectations framework.

The second topic of discussion is the mechanism of price formation. The most part of previous ABM markets used the clearing price condition to define the stock price value at each instant of time in the simulation. Using such mechanism we have *equilibrium* prices which satisfy asset demand of all the agents. However in a real stock market the situation is more complicated because there exists a double auction electronic system that manage all the purchasing and selling orders on the basis of their bid and ask prices: the electronic book checks, in real time, if there is a possible matching and executes all the corresponding operations. The implementation of such system is one of the main novelties with respect to previous artificial stock markets and this will generate many complex features which otherwise will be difficult to analyze: one of them is that is not always possible for investors to reach their optimal equity fund composition in

just one trading day and this is one of the forms of *market imperfections* due to the real trading system.

The final question to take in great consideration is the learning process for the agents in the simulation. The informational network is the evolving tool through which a trader can learn and update her strategy and her decisions and so its role is to simulate learning possibility in terms of social interactions among agents. We are then going beyond a more realistic learning process with respect to the GA used in the SFI market. The latter tool is clearly more efficient and elegant in the exploration and construction of pricing models, but this sort of elegance makes it just unrealistic. We can imagine a market in which investors interact with a limited number of other traders and through these acquaintances they can modify their behavior in terms of investment strategy: it is a more elaborate and imperfect learning mechanism if compared to a GA which mixes all the pricing models and produces new ones in a continuous way. Certainly the emerging network rules and decision-making processes can be changed or improved, but this approach can be a first step to consider social interactions among traders as an instrument of learning for the agents in the market.

3.5 List of control parameters in the simulation

Here we resume the complete list of control parameter which can use in the model, explaining their meaning and their potential role in the simulation.

n_{rand}: the relative fraction of random investors with respect to the total number of investors in the market. Remind that we can define also $n_{rat} = 1 - n_{rand}$ as the relative fraction of rational investors.

n_{consul}: the relative fraction of consultants with respect to the total number of rational investors.

$\triangle n_{trade}$: the relative maximum allowed variation, with respect to the baseline value ($floor[\frac{|\Delta W_i|}{p_i}]$), for the number of demanded/supplied shares. Augmenting $\triangle n_{trade}$ is equivalent to increase stock prices volatility.

ϵ_{test} : the threshold value for transition test $RAT \rightarrow RAND$. Increasing ϵ_{test} we reduce the probability of such transitions and then we decrease the magnitude of herd effects.

ϕ_{test} : the threshold value for transition test $RAND \rightarrow RAT$. Increasing ϕ_{test} the probability of such transitions is lowered.

p_{info} : the price which a rational investor has to pay when she applies to a consultant. This cost of information is fundamental because can affect the consultancy frequency for an investor.

λ_{test} : the threshold value for deciding to make a new call to a consultant (who can be the same of the last call or a new one). Increasing λ_{test} the probability to update the values of w_i' decreases.

x_{change}: the relative portion of additional financial wealth of a neighbor (with respect to the investor's portfolio value) which is needed to follow her advice. Augmenting x_{change} implies less competition among consultants and less permanent transitions towards random traders class.

4 Introducing SLAPP

SLAPP[12] (Swarm-Like Protocol in Python) is a simplified flavor of Swarm, which is *a multi-agent software platform for the simulation of complex adaptive systems* (Minar, 1996). While Swarm was originally written in Objective C, SLAPP is written in Python and therefore is a simplified version from a programming language point of view. Thanks to Python we can easily exploit the advantages of an object-oriented language to create a flexible framework for agent-based simulations. The main idea of Swarm was to provide object oriented libraries for building models and making experiments on them. The *swarm* is a synonym of collection, i.e. a collection of agents which is considered as the fundamental unit of every model. In order to get a more complete introduction to the relationship between SLAPP and Swarm you can consult a recent work of Terna et al[13]. We limit to illustrate the structure of SLAPP, describing its protocol and the general functionalities of its libraries.

As in Swarm, the protocol used in SLAPP is the creation of the simulation model on two different levels:

- the lower level is that of the model (the *modelSwarm*) where the agents live and interact each other. Here we create the instances of the agents (or *bugs*) and we prepare the actions to be executed. Moreover there exists a schedule (defined according to the clock of the model) that allows all actions are performed by all agents in the collection or by specific ones.
- the higher level is that of the observer (the *observerSwarm*), where we can run the model and execute other external actions, according another schedule and another clock. On such level we can monitor and analyze the outcomes of the model itself.

From the perspective of ABMs such protocol is very effective because when we build an agent-based simulation we want to manage it as a reusable experiment that can be launched every time we want and how many times is needed to make statistical analysis. On the level of the model we can control the dynamics of the simulation and the behavior of the agents: this is possible through the model schedule, a file which includes all actions to execute. We can see such schedule as a clock which opens a series of boxes, where every box represents an *action group* that has to be executed from a particular set of agents. We can associate to every action a probability or we can even allow an agent to add or eliminate some actions. This dynamical structure is based on the idea of modeling events as objects, in the sense of object-oriented programming language, and represents one of the key points recovered from the original Swarm system.

[12] SLAPP is a work by Pietro Terna

[13] *Agent-based Models of the Economy: From Theory to Applications*, Boero et al.

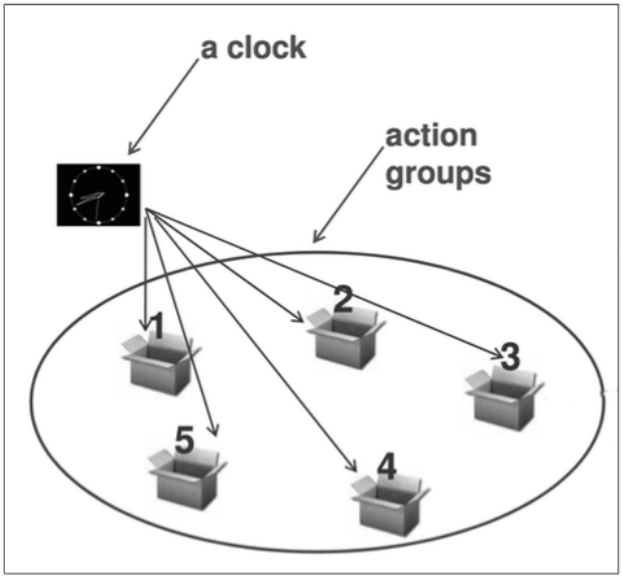


Figure 6: A simple sketch which shows the original idea of Swarm: the clock, according to the time schedule, opens a series of boxes, each of them containing an action group, which can be related to the Observer level or to the Model level.

The novelty of SLAPP with respect to Swarm is the possibility to read the schedule from a spreadsheet (a *txt* file or a *xls* file): this is one of the features of the scheme AESOP (Agents and Emergencies for Simulating Organizations in Python), an ideal layer of the models created in SLAPP[14]. It is possible to download SLAPP with all its folders and tutorials at the following web page[15].

In order to better analyze SLAPP structure we illustrate all its files (*.py*) with their main functionalities. We distinguish files included in the general folder of SLAPP (*$$slapp$$*), from those which are specific of your project; every project is simply a folder to put in the directory *6 objectSwarmObserverAgents_ AESOP_ turtleLib_ NetworkX* . So you can assemble your project creating a folder with all the files needed in this framework, plus any other file (which can be data file to read or a new file *.py*) to elaborate your work. Such short introductory guide is intended for who is willing to create an ABM in SLAPP for the first time. Every class or file can obviously include many other files (beside external but fundamental libraries as *random, matplotlib, numpy, datetime, networkx*) and so we have a structure which is a *puzzle* of libraries that help the user to build an ABM according to her own personal necessities.

ModelSwarm.py (SLAPP generic): the model level Every model we want to create is an instance of the *ModelSwarm* class. Here we summarize the main methods of the class

- the constructor of the class sets the number of *bland* agents (i.e. the number of non-specific agents in the model) and the size of the simulation world (i.e. the boundary coordinates in the plane used if we have to make move the agents); it reads the files *AgTypeFile.txt* and *AgOperatingSets.txt* (if the last one exists) and then sets all types of specific agents (or groups of agents) of our model;

[14]See Chapter *Introducing SLAPP* in *Agent-based Models of the Economy: From Theory to Applications* , Pietro Terna.
[15]https://github.com/terna/SLAPP/

- the methods *buildObjects* creates all the agents in the model (*bland* or specific);
- the method *buildActions* creates the *action groups* as instances of the class *ActionGorup* and builds methods through which we can read all such actions and then execute them. The method *buildActions* indeed uses another important method of the class *ModelSwarm*, which is *applyFromSchedule*: reading the schedule of the model, all actions are performed exploiting a general method contained in the file *Tools.py.*
- the method *step* executes the *action groups* on the *model* level, using the methods created with *buildActions.* So this is the method which actually implement the dynamics of the model at a single cycle.

Tools.py (SLAPP generic): general methods in SLAPP In this file we have many generic methods that allow us to make execute a specific method in the model *on* all agents or on a single one. We say that we execute a method *on* the agents because the specific methods of the model are included in the class *Agent.py* and we can call them using instances of that class. So the methods in *Tools* take as argument a collection of agent and a method which they will apply on such collection.

ActionGroup.py (SLAPP generic): actions as instances This class is that of an instance *action group*, in the original perspective of Swarm. We want only to underline that an instance *ActionGroup* has only a data member which is a string used as the name of the *action group.* Such name is taken from the file *modelActions.txt* which includes all relevant *action groups* to define in the model. The method *buildActions* of *ModelSwarm* class will create instances of *ActionGroup* and will build methods to execute them through other external methods, which can be model specific (contained in file *mActions.py*) or more generic (contained in file *Tools.py*). The file *modelActions.txt* has always to include a name

```
read_script
```

which represents the series of actions written in the schedule. So method *buildActions* of *ModelSwarm* class will define an *action group* that incorporates all the actions relative to the time schedule of our model. Other actions which can occur (in the mind of the modeler) out of the schedule will be defined as other instances of *ActionGroup*, according to what we have written in *modelActions.txt.*

convert_xls_txt.py (SLAPP generic): using excel to create the schedule Thanks to this file, SLAPP gives the possibility to the user to convert the file *schedule.txt* in the Excel version *schedule.xls.* The advantage is to provide a more readable version of the schedule, with rows and columns which have a particular meaning (see the paragraph relative to the reading of the *schedule.xls).*

Pen.py (SLAPP generic): implement *Turtle* class in your SLAPP applications The class *Pen* can be used for projects in which are interested to draw the position of our agents or to make other graphical operation in our simulation world; it uses the graphic class of Python *turtle.*

ObserverSwarm.py (SLAPP generic): the observer level When we want to run our model we create an instance of the class *ObserverSwarm.*

- the constructor sets the chosen project and initializes the instance of class *Pen.*

- the method *buildObjects* calls function *loadParameters* (from *parameters.py*) to set all parameters in the simulation; moreover it creates the instance of *ModelSwarm* and calls the method *buildObjects* (of *ModelSwarm* class) on it.
- the method *buildActions* apply on the instance *ModelSwarm* its namesake method; in addition it reads the file *observerActions.txt* (which has the same role of *modelActions.txt* on the observer layer) to build all *action groups* which are on the observer level. In particular in file *observerActions.txt* the *action group* which corresponds to the entire model dynamics is

  modelStep

- the method *run* executes all *action groups* previously created through *buildActions*: the

  modelStep

  action group is implemented calling the method *step* on the instance *ModelSwarm*. So *run* implements the complete dynamics on the observer level.

commonVar.py (project specific): initialize all simulation parameters This file contains all relevant control parameters for the simulation and any other useful variable which can be globally defined in the project. If we have parameters which affect the dynamics of the ABM (and then the behavior of the agents), we will include them here. All variables are simply initialized since they will be set at the proper time, using other functions which can be general methods of SLAPP or specific methods or functions defined in the project. If we want to remove some control parameters or add new ones, we can do this through this file.

parameters.py (project specific): fixing control parameters from interface The file *parameters.py* uses the function *loadParameters* to allow the user to set all control parameters of the simulation from the shell (or from the terminal). Moreover such function can be used to define the size of the simulation world (if it is relevant) and to decide how many cycles the simulation will run. So we have the possibility to choose the values of all control parameters before the simulation runs and this is of course an operation on the *observer* level.

WorldState.py (project specific): what kind of simulation world do you want to create? The method *buildObjects* in *ModelSwarm* creates an instance of class *WorldState.py*: such instance represents the state of our simulation world when we run the simulation. In this class we can create methods to set and get parameters (which are data members of the class) relative to the simulation world itself. The difference with respect to the parameters in *commonVar.py* is that we can directly act on the WorldState in the schedule, changing the corresponding parameters of the world. In other terms agents know environmental variables associated to the world state and they behave consequently.

Agent.py (project specific): specific model dynamics and agents behavior The class *Agent* is maybe the most important of your project because it contains all useful information to create the agents of the model and all relevant methods relative to the model schedule. Its constructor , which is called from the method *createTheAgent* in *mActions.py*, will initialize all data members of the agents. After its definition we can write any method relative to the model dynamics, which will be used from the agents (all of them or just a fraction). It is possible that such methods have to include other instances or methods

from other new classes specifically created for your project. So *Agent* class will have to include other files and this will extend the architecture of your project.

graphicalDisplayAndGlobalVarFunctions.py (project specific): functions for Network Analysis This file contains all the methods needed if you want to make network analysis on your project. Here you can create the graph that will arise from the simulation evolving with the system dynamics. All relevant methods used to make operation on the graph, to perform network analysis measures and finally to draw the graph itself, have to be written here. Clearly every method you create here can even be exploited inside the dynamics of the model itself, i.e. inside other specific methods of the class *Agents.* We have to highlight that such file imports the *networkx* module, which is a very powerful package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks[16].

mActions.py (project specific): methods on the model level This file contains methods which are used by *ModelSwarm* methods. The first one is *createTheAgent*, which is a specialized constructor (it uses the constructor of *Agent*) to create the specific type of agent in our model collection. The remaining methods are copied on *ActionGroup* instances to create all the methods to execute the *action groups* in *modelActions.txt.* Such choice is intended to avoid the existence of too many methods, relative to different projects, for the class *ActionGroup* (which instead has no specific methods). So in this way *ActionGroup* remains on the lower level of the SLAPP architecture and we have not to change it creating a new project.

oActions.py (project specific): methods on the observer level The same idea of *mActions.py* is here implemented for the observer. We can find methods that will correspond to the *action groups* in the file *observerActions.txt.*

penPosition.py: graphical interactive methods In this file we have the function *setPen* which creates the *Pen* instance and apply on them the proper method (of *Pen* class) to use it in the project. If the project does not need the use of class *Pen*, its instance address is initialized to zero and no methods are called on it. Note that *setPen* is called inside the constructor of *ObserverSwarm*.

All the files and classes above presented for the specific project you are working on, are mandatory in SLAPP: this means that you have to have all such files in your project folder. The following scheme represents the SLAPP structure in terms or relationships among all files and classes previously explained. The arrow starts from the class or file imported towards that one which imports it.

[16] See http://networkx.lanl.gov/index.html

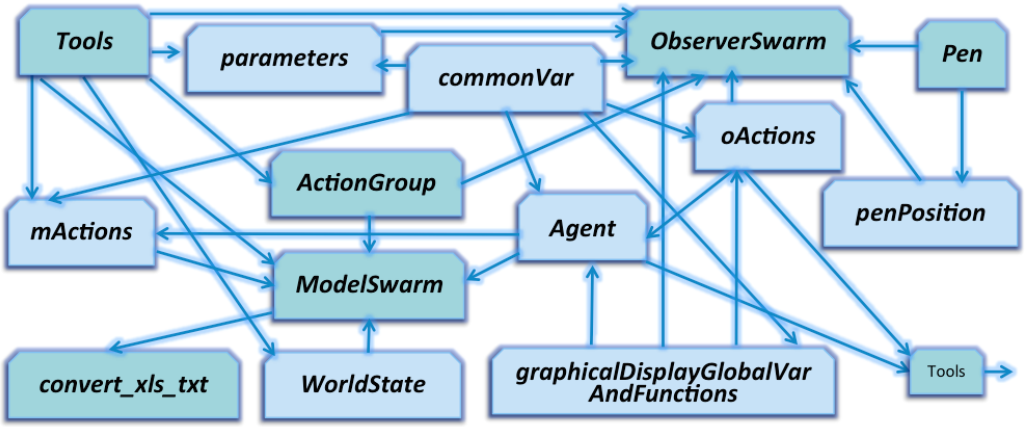


Figure 7: A network of relationships among the main files/classes used in SLAP: the arrows are directed towards a file/class that imports that one from which the arrows has started. The file colored in turquoise green are those included in the directory *$$slapp$$*, i.e. defining the basic structure of SLAPP. The light blue files/classes are instead contained in the folder of your specific project.

Beside the specific files and classes of the project shown in the scheme (which are mandatory in SLAPP), we can include any other Python file according to our personal needs in the project. The new files will be used for instance from the class *Agent* or from other specific classes of the project, but the general layer (contained in the folder *$$slapp$$*) of SLAPP can remain unchanged. In this sense we have a very flexible framework which allows the creation of any new ABM, maintaining a basic structure of files and classes which are build according the SLAPP protocol. It is possible to adopt little changes even in the general files of SLAPP, but if it is more efficient to change only the specific layer of SLAPP, so building many ABMs which operates with the same tools and with the same protocol.

4.1 How to use the model schedule

One of the most interesting topics in SLAPP is the capability to read the time schedule of the model from a simple spreadsheet. Here we give the explanation, taken again from Terna[17], of how to read the schedule (and then to create a new one for your specific project).

In column A, we have the sign # or the word macro or a name identifying a group of agents (the number of the agents in the group can vary from 1 to any value):

- with #, we state that, when the clock reaches the time (in units of the conventional internal clock of the model) set in column B, the content of the rows following that containing the # sign and until the next similar sign, have to be read and executed;
- with a macro name, we indicate that at a given time, SLAPP will activate the set of instructions reported in a specific sheet; with a name identifying a set of agents8 in column A, we send to this/these agent/s the method (as an action to be done) set in column B in a deterministic way; if in column B we have a number, this is the probability (expressed upon 1) of execution of the method, in this

[17] chapter *Introducing SLAPP* in *Agent-based Models of the Economy: From Theories to Applications* (Terna et al., 2015)

case reported in column C. The probability can be interpreted both as the share of the set of agents (recalled in column) to which we ask to do the action, or, which is quite the same, as the probability of each individual of the same list to be asked to execute the action. If the number in column B is both less then "0" and integer, exactly that number (multiplied times -1 to have it positive) of agents is asked to execute the actions; these agents are randomly extracted from the list.

5 The program structure

Having now visualized the main structure of SLAPP and its functionalities, we can write all the classes and files to implement our ABM. We will document the code in every class or file, explaining the scope of every method and the role of the main variables: it is very important to compare the code with what we have previously said about agent behaviors, trading mechanisms and emerging network rules. Moreover, through the construction of the code, and considering SLAPP's perspective, we will discuss in great detail the time schedule for all the actions in the model, a topic which we have previously (deliberately) evaded.

5.1 Initialize the control parameters (commonVar.py)

Let us start from the file which includes all the control parameters of our simulation.

Here we have a very useful memo to visualize immediately all the variables which can be considered *common* to all classes and files of the project.

Beside the control parameters which have presented above with their role in the simulation, there are other important variables to comment.

- *g* is the address of the instance *Graph* which will be created to define our network. All network operations (performed using the library *networkx*) will be executed on *g*.
- *g_labels* contains the address respectively of the labels in the graph.
- *SeriesAnalysys* is a boolean control variable used to ask the user if she wants to exploit the Observer tools to make data series analysis.
- *NetworkAnalysys* is a boolean control variable used to ask the user if she wants to exploit the Observer tools to make network series analysis.
- *orderedListOfNodes* is the list which contain the list of nodes in the network, where the nodes are instances of the class *Agent*.
- *clear* is a control variable which is set True before the beginning of each trading day (or better at the end of the previous day) to indicate that all book lists have to be clean.
- *verbose* is a control variable that is set from the shell (or from the terminal) at the beginning of the simulation: if it is *y* (i.e. *yes*) the program prints several comments about the execution of some important steps.
- *Arbitrage* is the boolean variable to allow arbitrage operations or not (see Chapter 6).

5.2 Set control parameters and simulation trading period (parameters.py)

Through this file we can set all control parameters in the model using the method *loadParameters*. In particular we can set the number of cycles (i.e. trading days) the simulation will last. *The time in our simulation (from the model perspective) flows on two different plans: on higher scale, we have the ticks (which are called cycles in SLAPP) that corresponds to trading days, while on lower scale we have the continuous time in the auction (i.e. the time used to complete the trading operations within a single trading day).* Here we report the main code of the function *loadParameters*

First it is important to talk about the *seed* of the simulation. The *seed* is an integer number (positive or negative) chosen to determine the random sequence used internally by our model. We briefly remind that a sequence of random numbers is only *pseudorandom*: indeed the *random* generator is simply a deterministic map which is chaotic from a mathematical point of view and then reproduces a sequence of numbers that seem random. If we repeat the execution with the same seed, the results will be the same. We enter 1 to start a different sequence in each execution, being the current value of the computer clock used as seed. Next question to focus on is that there are no *bland* agents in our project: this is not surprising because we want to well distinguish the role of investors and consultants in the market. However, for future extensions, we could think to use *bland* agents as further market operators who can execute other generic actions, being neither an investor nor a consultant. Finally we can note how the size of the simulation world (in sense of boundary coordinates on the plane) is not relevant here. Our market is represented by many investors or consultant who can interact from different countries. *The electronic book and any other informational medium can link an agent to the others: no geographical or space variable has to do with our model.*

5.3 Observer and model levels (observerActions.txt - modelActions.txt)

Let us see how we implement SLAPP protocol for our project, showing the *action groups* first on the observer level (in *observerActions.txt*) and then on the model level (in *modelActions.txt*).

Starting from the observer layer we find in *observerActions.txt* the following line

```
visualizeNet modelStep pause PlotDailyPrices pause
checkPortfolio pause checkConsultancy pause
DailyProfits pause LogReturnsDist pause
PlotDegreeDist pause DegreeVsProfits pause
Clustering_Assortativity pause
AutoCorrel pause clock
```

- *visualizeNet* is the method which draws the graph, having cleared its version of the previous day. Here we can see the links among investors and consultants (any particular agent is a node with a specific color as we will speak in the next paragraph) observing what have been the relationship changes among agents with respect to the past trading day. Indeed, at the end of the past trading day some investors can have changed their consultant and can have made new acquaintances in the market.
- *modelStep* orders to the model to make a step, i.e. to follow the sequence reported in *modelActions.txt* (see below).
- *PlotDailyPrices* makes see the plots of stock daily prices during a trading day.
- *pause* makes the program waiting until we hit the key Enter. The pause allows us to observe the network modifications step by step.
- *checkPortfolio* allows the user to verify if, during the last trading day, there were some investors who have reached the optimal portfolio, or at least who have reached a proxy of it (for instance fixing an acceptable error in percentage terms for the portfolio shares).
- *checkConsultancy* compute daily return of the optimal portfolios built by every consultant. This is an easy measure of consultancy performances, on the basis of the different expectational models used by the analysts.

- *DailyProfits* says the user which investors has outperformed or underperformed in terms of daily profits with respect to the rest of the investors. Moreover at the end of the simulation a histogram of the average daily profits is plotted to show which is their final distribution.
- *LogReturnsDist* shows the histograms, one for each security, with the sample distribution for log returns at the end of the simulation. We consider only the returns with respect to closing prices, so to have a reliable sample it needs to run the simulation for many trading days. Moreover such method prints the moments of the distributions as the sample mean, the standard deviation and the sample skewness and kurtosis.
- *plotDegreeDist* shows the plot of the degree distribution for the informational network at the end of the day. Here we can observe in a very explicit way the evolution of such temporal network.
- *DegreeVsProfits* is a powerful tool which looks for relationships between the degree of a node/investor and her mean daily profits (mediating over all the simulation trading days). The method run a linear regression on such quantities (the degree is the regressor) and shows the data point and the OLS line obtained. Then we are able to monitor the evolution of such relationship during market evolution.
- *Clustering_Assortativity* offers several network measures to investigate the global properties of our informational network. It computes the number of connected components, the average clustering coefficient and assortativity coefficient, all of them at each trading day. In addition it shows the plots with all the daily average clustering coefficients and assortativity coefficients. Together with the time-varying degree distribution, this is an effective way to see how the network evolves during the simulation.
- *AutoCorrel* is called at the end of the simulation (i.e. at the end of the last trading day): it plots (and shows numerically if required) the auto-correlations and partial auto-correlations using the returns of each stock at the end of every trading day (taking into account the closing prices). The daily case all the autocorrelations (both partial and not) are plotted at different time lags or can be print if required. All such operations are performed by R, which is indirectly used in Python through *Rserve*[18]
- *clock* tells the observer to make ahead the time on the observer layer. In this work such time corresponds to the specific trading day of the simulation.

All the methods which execute such actions are included in *oActions.txt*. The list of actions which defines the actual model are printed within just the following line in *modelActions.txt*

```
reset read_script
```

- *reset* is there as future use if we needed to reset the variables at every cycle.
- *read_script* is the command which invokes the content of *schedule.txt* (or *.xls*) file and looks to the # blocks, if any, related to the current value of the clock. So here we have all specific actions of the model schedule, which are methods written in class *Agent.*

5.4 The *Agent* class (Agent.py): agent behaviors and external data

In order to define correctly the specific model dynamics we have to write the constructor of *Agent* class and all its other methods.

[18] You can see what *Rsevre* is and how can you get it at

5.4.1 Constructor: initialize all data members and create the graph

The constructor takes 11 arguments (beside the mandatory *self*):

1. the number of the agent, i.e. its ID written in files *investors.txt* and *consultants.txt*.
2. the current state of the simulation world: this is the instance of *WorldState* class, created by the method *buildObjects* of *ModelSwarm*.
3. the coordinates in the plane of the agent. As we have previously said our project do not use any space variable: however, in order to visualize the network it is convenient to create coordinates of the nodes, where the nodes are the agents of the model. Since the actual location of agents in the plane is irrelevant, we generate such coordinates in a random manner according to a uniform distribution in the allowed range: this is directly implemented in the function *createTheAgent* of *mActions.py*.
4. the border coordinates to limit the space in which we represent the agents in the plane. Once again the space distribution of our agents is not important in this model, but we use such variables to get a good representation of the network.
5. *agType* is a string variable corresponding to the type of agent we can create the in the model and it can be *investor* or *consultant*. The constructor automatically initializes it to a null string, because the type is defined for each agent reading the files *investors.txt* and *consultants.txt*.
6. *kind* is a string variable for the particular class which an agent of some type belongs to. So *kind* can be *random* or *rational* for *investors*, while it can be *trend, chartist, fundamentalist* or *Bayesian* for consultants. In other terms while *agType* is the main classification for agents in our program, *kind* represents the specific class for a single type. Both variables are data members of *Agent* class used to classify every agent.

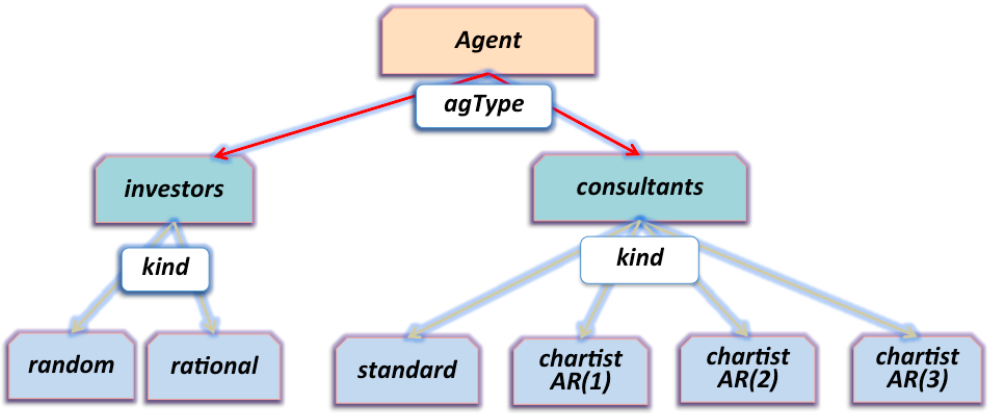


Figure 8: Agents classification in terms of programming language. An instance of agent is defined by two data members: *agType* tells us if the agent is an investor or a consultant; *kind* is indeed the specific class of investor/consultant.

7. the address of the created instance *modelSwarm*, which is copied here to be used in function *createTheAgent* (which indeed has to call the constructor of *Agent*).

All important data members or (agent attributes) are initialized by the constructor, according to the type of agent we are considering. So we have:

Investors: degree of risk-aversion γ_j; total cash account S^j; vector $\vec{n}$ of owned shares for every stock; all different portions of the equity, relative to every single stock (i.e. all W_i^j), computed using the last prices in external data lists (see next function *create_initial_prices_lists*); ΔS_i^j and all ΔW_i (which they are all initialized to zero); the optimal portfolio vector $\overrightarrow{w}$ and the chosen consultant (respectively initialized to empty list and to zero); the list of acquaintances in the network (empty list) and the number of days without consultancy $n_{\vec{w}}$ (which is initially zero); the daily profits due to trading operations and market fluctuations; the list of all daily profits.

Consultants: the list of past returns (which considers both external initial data prices and the new ones coming from the simulation); the list of the expected returns for the next period for every asset; the optimal portfolio vector $\overrightarrow{w}$ (initialized to null list).

Moreover all new *Agent* instances are appended to the list *orederdeListOfNodes* which is also used in file *graphicalDisplayAndGlobalVarFunctions.py* for network manipulations. Then we assign a label to each node in the graph, which is the corresponding number of the agent: in this way we can see on our screen her specific network behavior after each trading day. In order to distinguish from different types and kinds of agents we give them different colors[19]:

- *LightGray* = random investors;
- *ChartReuse* = rational investors;
- *Maroon* = standard consultants;
- *DarkBlue* = chartist AR(1) consultants;
- *Cyan* = chartist AR(2) consultants;
- *DarkMagenta* = chartist AR(3) consultants;

Finally we have to say that when the *Agent* constructor is called for the first time (first agent created) the constructor calls the function *createGraph* from *graphicalDisplayAndGlobalVarFunctions.py*: this initializes the graph g as instance of class *Graph* (thanks to networkx).

5.4.2 The other methods

Now we can describe the general functionalities and scopes of the methods in class *Agent*. For every topic we have a corresponding method applied on a type of agents (indicated in square brackets).

[19] See at http://www.w3schools.com/html/html_colornames.asp

create_initial_prices_lists [] First of all we have to think about the initial external data for stock prices, which is essential to compute the first forecast returns and then to make the simulation starts. All data are taken from the web page of yahoo.finance[20]. We consider a sample which starts from 09/01/2014 to 01/30/2015, which corresponds to almost five months of trading: look at the section *Running the simulation* to see which stocks are considered in this work. It is suffice to download the *csv* files (with daily prices) from the web and include it in the folder *6 objectSwarmObserverAgents_AESOP_turtleLib_NetworkX* where the main file *start.py* is located. The function *create_initial_prices_lists* generates the initial lists of stock prices (from the oldest to the most recent). It is also possible (using the library *datetime*) to define the lists of dates for all prices, which include the corresponding day, month and year: such choice could be useful if we intend to give our simulation an historical context. This function is used inside the constructor, only when we create the first agent: the reason is that we have to get all p_i^{last} from external series.

compute_optimal_portfolio [consultants] The method *compute_optimal_portfolio* is used from all kinds of consultants: taking the closing price of the day for all stocks, the consultant update the lists of stock prices (and then of returns) to elaborate the new expected returns for the next day. Remember that different kinds of consultants apply different asset pricing models to compute expected returns; once the list of such values (one for every stock considered in the simulation) is computed we can get the new covariance matrix. Finally the same algebraic rule is applied, in the portfolio framework used for our work, from every consultant to get her own optimal portfolio. At the first trading day obviously we have no closing auction price and so the consultants elaborate the prices lists yet initialized in the constructor applying the function *create_initial_prices_lists* (indeed the global price lists are initialized to the external lists). In this method we include all relevant asset pricing methods used for the four kinds of consultants considered in this work. If we decide to modify the expectational models used by a particular kind of the consultants, or even to include a new kind, we have to add here possible modifications and extensions.

ask_advice [investors] After the consultants have performed their forecasts, we can apply the method which implement consultancy operations. We use such method also to initialize edges among investors in the network and this is done at the first trading day. When a new link is set we immediately call the function to from *graphicalDisplayAndGlobalVarFunctions.py* to show how the graph is changed in order a dynamically view of network interactions, both for links investor↔investor and investor↔consultant. Anyway the remaining part of the method allows the rational investors to call a consultant and get the new optimal portfolio shares: such process is also controlled by the consultancy test which uses the control parameters λ_{test} and p_{info} (which are variables of *commonVar.py*). The possibility to change consultant is here included on the basis of criterions discussed above.

prepare_orders [investors] Before to send their orders in the auction, investors have to decide how much money (denoted by ΔS_t^j) will invest in current trading day and what will be the different cash amounts used in every auction to perform an order (i.e. the several $\Delta W_{i,t}^j$). All such choices are implemented according the type of agent on which the method *prepare_order* is called: so we can see here the main difference between a random or rational investor. Remind that such method is called inside the main method *trading_day* because the investor needs to know the current situation in the market to choose the terms $\Delta W_{i,t}^j$.

[20] https://it.finance.yahoo.com

set_orders [investors] With all the known values $\Delta W_{i,t}^{j}$ all investors can perform their orders in all the books: this process is the same for both random and rational traders. The idea is that a single order is a vector of this kind

[price quantity quantity ID−number order−number]

where:

- the price can be a bid or an ask price;
- the second component of the order vector is the quantity which the investor want to buy or sell, i.e. $n_{i,buy/sell}$.
- the third component is set equal to the second one and is used as benchmark on which compute the minimum volume the investor want to accept to process her order. In other words if the order can be process with a volume lower than some x percentage of this third component, the investor prefer the order is delete from the book. We will discuss in more detail such question when we explain the class *Book*.
- the fourth component is the number of the investor, which is used as ID in the book and which corresponds to the number used in our *investors.txt* file to identify the investor. This can be useful to analyze all the investor operations on the book.
- the last component is the number which defines the time order in the book for every investment order. So, for example, the first order which enters in the book has an order-number equal to one and so on.

Every order is a vector of this kind and the method method *set_ order* fill it for every investor and return such vector: so we have defined a *getter* method which will be used to perform a real investment order during a trading day.

clean_Books [investors or consultants] In our model we have assumed that all orders which remain unexecuted at the end of the day will be eliminated. So, before opening all the auctions, we need to clean all past lists in the books of every asset. This method operates such cleaning, not only on the two sides of a book, but also on the lists of past daily prices (see more detail in the next section related to *Book* class). Of course it is suffice that only a single agent performs this operation and so the method is actually applied by the first investor which is called on. It does not matter which type of agent executes this operation, since it is simply a fictitious trick to clean all the books: then it is possible to call the method on the consultants too.

trading_day [investors] The actual double auction market dynamics is implemented in the method *trading_ day* and acting on instances of *Book* class (see later). Reminding that all *Agent* methods are executed on a shuffled version of the list containing all agents (in this case only the investors), at each trading day, the time sequence of sending orders in the book always changes. When the method *trading-day* is applied on an investor, the investor first uses the method *prepare_ orders* and then we can get the vector order of that investor through method *set_ order* (which is a *getter* method).

update_profits [investors] When all the auctions are closed it is necessary to update the portfolio value of all investors. The *Book* class sets every money and securities exchange in trading operations, but the overall value of the portfolio at the end of the day is computed according to the closing prices for each stock. So this method update financial wealth of every investor and then defines which are their daily market gains. The profits are computed as percentage variations with respect to the equity value at the beginning of the day and will be set as data members of each investor.

search_new_acquaintances [investors] At the end of the trading day investors can extend their acquaintances using their network neighbors. So they can call acquaintances of their acquaintances to extend the number of edges in the informational market network. This process is done in the perspective of augmenting the external information: only in this way we can better decide if there are better consultancy opportunities or even if there is more profitable stopping to ask consultants. Moreover such method implements the transition test $RAT \rightarrow RAND$.

append_prices [investors or consultants] Every tick in the simulation model ends with the essential updating of all stock price lists. Every list of this kind is a global variable from a programming perspective since it can be accessible to all agents. The idea is that a single consultant or investor will append the closing price of the past trading day to the corresponding stock price list. As in the case of *clean_Books* it is suffice to call this method just one time, independently by which is the agent who operates on the lists.

5.5 The class *Book* (Book.py): continuous time during the trading day

The great advantage in using SLAPP is that we can easily extend the main structure of our model adding new specific classes to the the main files usually needed. In our case we are going to create a specialized class for the electronic double auction mechanism in trading operations: in other words we have to write the specific class which represents the *book* of some asset. The creation of a *Book* class is one of the main pillars in an Agent-Based Model intended for the simulation of real financial markets. We know that the actual trading mechanism represented by the electronic double auction market is one of the most important determinants for all complexity in a system such a stock market. So the specific code for this class is one of the key points in the entire program structure.

An instance of the *Book* class represents the book for a specific asset. Every instance *Book* is created inside the file *WorldState* at the beginning of the simulation and we maintain the same instance for all the simulation time. Using the common rules in a real double auction market we write two main methods for this class: one for purchasing orders and one for selling ones. Every time an investor wants to enter in the auction one of these two methods is called and the state of the book is modified. If some trading operation is possible the method immediately executes it: the new price for the security is included in the list of daily prices (which is a data member of the *Book* class) and there is an update in money (given to the seller) and shares (delivered to the buyer) for the investors who completed the operation. So the book evolves dynamically as in a continuos auction market and the emerging prices depend by the order in which investors send their envelopes. Among the other class methods, the most useful allow to know the size or the price list for both sides of the books (and they will be used to make transition test related to herd effect).

5.5.1 Simulation system for continuous time

Another important question from a simulation perspective is how the continuos time inside the auction is controlled. The list of daily prices is related to every single operation realized in the auction, but the *real* time during the day has to be simulated in some way. We include in our class a specific data member to represent the time (expressed in *h-min-s*) at which a new order is send in the auction (and processed if possible): this tuple value is added to the order vector of the book. We are assuming a trading day which starts at 9.30.00 and ends at 16.00.00 The algorithm to simulate the real trading times takes into account the actual sequence in which investors can send their orders. It is suffice to draw two numbers, one for the seconds and one for the minutes; they will be both added to the *min-s* tuple value for the previous order, such to determine the new *h-min-s* values. This is equivalent to draw random numbers within a specific window in terms of time variation among two consecutive orders. The algorithm has to be constructed in response to the actual number of investors in the market. The maximum time variation (in particular for the minutes) has to be lowered if we augment the number of traders: in this way we avoid order times out of the maximum limit, i.e. out of the closure time for the daily auction (which we set to 16.00.00). Note that the actual trading time is a property of the book and every book is related to a particular asset: in this way we generate trading instants in a independent way for each security, which well mimics what happens in reality.

Now we explain in more detail how the total time available during the trading day, which is usually of 390 minutes, is partitioned among all agents intervention in the auction. As we shall see in Chapter 6, one of the main goals in such work is to keep aligned the simulated market to a real one and analyze the emerging features in the model in this scenario. To do this we shall get a real data sample for intra-day prices: for every stock there is a series of list, each one containing all the daily prices with time frequency of one minute. *Day by day the list of real daily prices is a sort of guideline to make time partition, also if we are running the simulation without an actual alignment to the real data. So, denoting* $l_{i,t}$ *as the length of the list of daily prices for stock i at trading day t, we create an average time window of intervention with the following width*

$$\overline{\Delta\tau}_{i,t} = \frac{l_{i,t}}{M}$$

where we remind that M is the total number of investors in the market. The reason to consider this value as an average time window is that we shall add a stochastic fluctuation to fix the actual time variation between two consecutive orders. Here $\overline{\Delta\tau}_{i,t}$ is computed as a real float number and so it has to be converted in minutes and seconds, in order to use it in the time system for the auction: the conversion is made taking into account that time frequency for real data is of one minutes, in this way

$$minutes(\overline{\Delta\tau}_{i,t}) = floor(\overline{\Delta\tau}_{i,t})$$

$$seconds(\overline{\Delta\tau}_{i,t}) = [\overline{\Delta\tau}_{i,t} - floor(\overline{\Delta\tau}_{i,t})] \cdot 60$$

So we can express

$$\overline{\Delta\tau}_{i,t} = (minutes(\overline{\Delta\tau}_{i,t}), seconds(\overline{\Delta\tau}_{i,t}))$$

For instance if $\overline{\Delta\tau}_{i,t} = 1.5$ we get $minutes(\overline{\Delta\tau}_{i,t}) = 1$ and $seconds(\overline{\Delta\tau}_{i,t}) = 30$. Now, starting from 9.30.00, every investor entering in the auction moves the time counter in this way

$$\triangle\tau_{i,t} = (minutes(\overline{\Delta\tau}_{i,t}), randint(seconds(\overline{\Delta\tau}_{i,t}) - a, seconds(\overline{\Delta\tau}_{i,t}) + a))$$

This is the actual time variation between two investors who send consecutively a buying or selling order in the auction. Note that the stochasticity of this process is controlled by the parameter a, which fixes the possible fluctuation for the seconds, while the minutes of $\triangle\tau_{i,t}$ are always fixed in order to exclude too high fluctuations. If we set $a = 0$ the time flow becomes deterministic. A value of $a = 1$ is what we adopt for this work because we want a very stable time generation system and such request will be clear in Chapter 6 speaking about arbitrage operations.

Finally it is important to say that the value of continuous time within a trading day is absolutely not important if we run the simulation without alignment to real data. What does actually matters is the order in which buying or selling intentions arrive in the auction. Nevertheless we decide to show continuous time also in the totally free simulation scenario, so to provide a pretty feature for the auction. The importance of time partition and simulation within a trading day is instead essential if we want to guarantee a real time alignment for simulated stock prices to real ones, as we shall illustrate in Chapter 6.

5.5.2 Further extension: introduction of realistic constraints for investing operations

The *Book* class is also intended for short-selling or leveraging. In our market we assume that such operations are always allowed such to maintain the portfolio optimization process totally free. Anyway these two trading operations actually need to open an account with a broker, maintaining a percentage margin above a required limit. If we want to take care about this element in the simulation it is possible to extend the code in the *Book* class to represent the current situation of the account with the broker and apply margin calls when is needed (i.e. when the margin account falls below the necessary threshold). In our work we do not add such possibility because this would not modify significantly our results: however the extension is simple to implement in the program.

In real markets limitations on short-selling and financial leveraging could be introduced in order to try to reduce overall systemic risk. *Once again, using a such flexible ABM, we can include every constraint on such operations, implementing the limitations in the* Book *class. So there is the chance to write a stock market simulation with all the rules you need or that you want: you can explore new scenarios, changing the actual limitations with new rules, or you can evaluate different regimes of policy making interventions (for example different actions of the Central Bank).*

5.6 The model schedule (schedule.txt)

Now that we have described all *Agent* methods, we can finally show the actual model dynamics contained in the file *schedule.txt*. We report the lines relative to the first trading day: the lines are repeated for all the others days (cycles)

```
#1
consultants compute_optimal_portfolio
investors ask_advice
investors -1 clean_Books
investors trading_day
investors update_profits
investors search_new_acquaintances
consultants -1 append_prices
```

1. The first step in the model schedule occurs when the auctions are closed (the trading day is end). The consultants use the closing prices for every stock to compute the expected returns for the next

day and update the covariance matrix: in this way every consultant will get a new optimal portfolio, available to all investors who will call her in the upcoming trading day. So, on a temporal scale, this operation is executed between the closure of the past trading day and the opening of the current one: that is the reason why the method could be also put at the end of the model schedule.

2. When the current trading day starts rational investors are asking advice to some consultants in the market: such operation, as we had announced describing the model, can occur in any moment during the day (before the auction is closed) and always precedes any order sent in the auction. We assume that this is the first action in the model schedule because is logically the first necessary operation for rational investors to continue to act on the auction market.

3. Before opening all the auction markets we have to clean all book lists and this is done through method *clean_ Books* which is executed only one time per day; the first investor on which the method is called performs the cleaning operations (note the -1 in the schedule).

4. The next task for all investors is to define their orders on the auctions for all the stocks: this can be done just before send the order in the electronic book for every asset, using the *current* stock price. So the method *trading day*, which inserts and processes investors' orders in the book, exploits the methods *prepare_ orders* and *set_ orders*. Such methods are not included in the model schedule because every investors uses them when it enters in the auction market. The continuos time at which the investor sends her order is determined through methods of *Book* class and is the only temporal information we monitor about investor operations during a single day. The actual dynamics, which is executed through the rules in all the auctions, is then represented here thanks to *trading day*.

5. With all the auctions closed, it is possible to update the equity of all investors, computing their daily profits. Such step is very important because the remaining steps in the model schedules requires the knowledge of daily market gains and several quantitative methods (contained in *oActions.py*) on the Observer level exploit such values.

6. The final steps in a market day are related to the new possible links set by investors. All traders try to extend their informational network using their neighbors as intermediaries, but also considering the acquaintances who have best performed during the day. So the main changes in the evolution process for the informational network will occur at this step.

7. The simulation at each tick has to end updating the global list of prices for each stock: a single agent (we choose an investor) append the closing price of the previous day for each stock price list.

6 Running the simulation

Now we are ready to run our simulation looking at the first results. In every ABM is essential to start from a qualitative comprehension of how the control parameters can affect system dynamics. Moreover we have always to remember that the initial seed set by the user will determine a particular evolution of the system, which is intrinsically stochastic. Several choices of the seed will lead to several outcomes in stock market evolution and this is one of first problem to take care if you want to a statistical analysis. In addition, having set many control parameters, we are exploring a multi-dimensional parameter space: a single point in such space is a specific parameters setting which can produce a particular output in the simulation (with a fixed seed). A complete quantitative analysis will have to consider both such elements. In the following section we limit ourselves to exploring the outputs of the simulation and the role of control parameters.

All the past daily stock prices used as initial sample for the simulation are taken, as anticipated before, from the web page of yahoo.finance[21]. We are considering the closing price of every trading day. It is possible to choose any number and kind of stocks, even from different stock exchanges. For simplicity in this work we will simply consider four stocks, in order to have just few estimates in terms of portfolio optimization and expected returns. Here we report the four chosen stocks and the corresponding stock exchange on which are priced:

- APPLE Inc. (AAPL) on *NasdaqGS*;
- GOOGLE Inc. (GOOG) on *NasdaqGS*;
- FACEBOOK Inc. (FB) on *NasdaqGS*;
- International Business Machines Corporation (IBM) on *NYSE*;

All the initial data samples for stock prices are taken from the period 01-1-2015 to 05-15-2015.

6.1 First outputs

Let us start showing the outputs of the simulation, considering, day by day, which are the plots and the numerical measures available to the user. In this sense is really important to remind all the methods contained in *oActions.py.* which are all the tools used on the Observer level to evaluate the results of the simulation.

When you start the simulation the first thing which appears on the screen is the informational network among investors and consultants. You can see how the links are set, according to the generation algorithm included in *ask_ advice*: link by link the network arises in its initial version.

[21] https://it.finance.yahoo.com

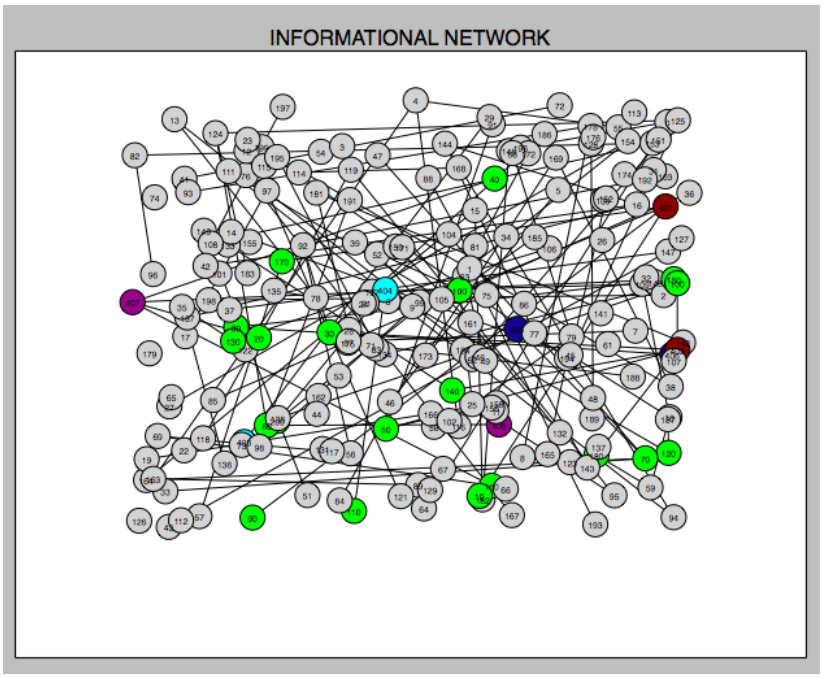


Figure 9: Informational network with colors to identify agent kind

Here we can see all the nodes in the network with the ID number for each investor or consultant: see the color labels to distinguish all types of agents. Once the initial network is set, the agents begin to operate on the market: consultants make their forecast for the daily stock returns and pass their optimal portfolio estimates to rational investors who have called them; investors fix their orders and send them in all the auctions, determining stock price movements during the day. It is possible to watch the daily price series for all stocks thanks to *PlotDailyPrices*

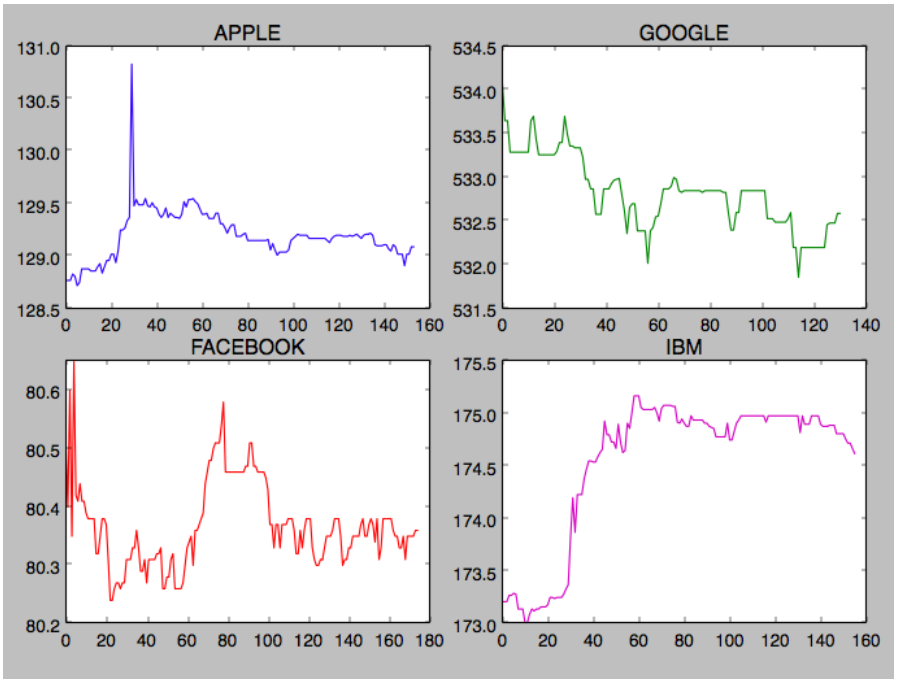


Figure 10: A plot of daily prices for the four chosen stocks: at each trading day you can observe a plot of this kind through method *PlotDailyPrices*.

So we are able to observe the general daily trend for every stock price. Next step is the evaluation of the global properties of the informational network. The first useful tool is the plot of the degree distribution, which is a very powerful indication of the features of any network; thanks to *PlotDegreeDist* we can visualize such plot and see if the network is heterogeneous or if it shows some specific trend (for instance a power-law behavior). Since the method is called at each trading-day the degree distribution will evolve during time, being affected by the market dynamics: then it is possible to follow such evolution and try to get the general properties regarding the dynamics of network itself. Here we show a figure of the degree distribution

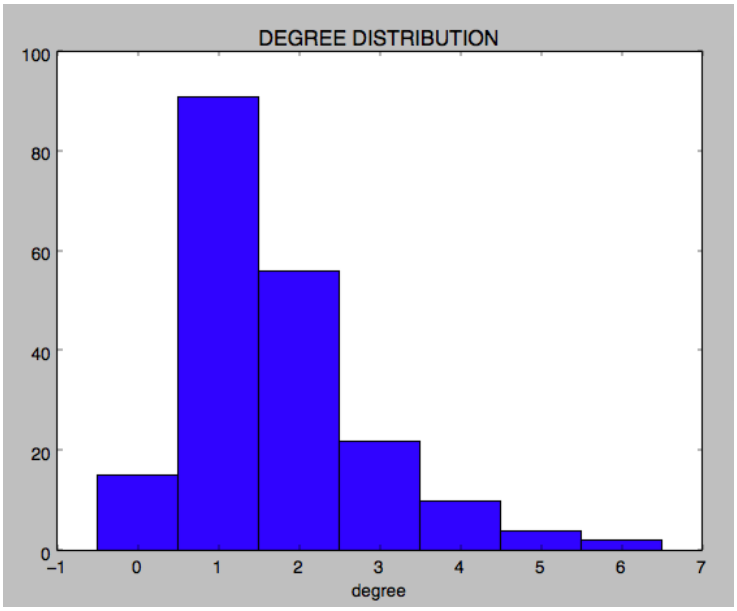


Figure 11: Degree distribution at some trading day: here there is no normalization.

Normalization can be required and shown if desired, both for this histogram and for other ones built in the simulation. Other important outputs regards the performance of investors (in terms of their profits) and of consultants (in terms of return of their advised optimal portfolio). You can get both numerical information through the terminal and plot to visualize the results.

The great flexibility of SLAPP has no limits in all the possible outputs (both numerical and graphical) you can obtain from the simulation. It is possible to analyze what happens at each trading day, but also analyze the final data sample emerging from the entire simulation. For all the other specific outputs offered by methods in *oActions.py* we shall refer to Chapter 7 for the quantitative analysis.

6.2 Discussion of qualitative effect of control parameters

The first necessary step to analyze the result of our ABM is to look at the qualitative effect of control parameters. Though we know in theoretical terms which is the local effect of any control parameter, we do not know how their variations can affect the global properties of the market and of the informational network. A complex system indeed is characterized by unexpected global pattern obtained by some parameters configurations. Here we look at every control parameter previously introduced, showing its effect and discussing from a theoretical perspective its role in the simulation. The idea is to try to calibrate all the control parameters in a reasonable range in which simulation outputs are realistic, so reducing the parameters space.

For the rest of our discussion (and also in the next chapter) we shall run our simulation with the following baseline values

r_f	M	n_{rand}	n_{cons}
0.01	200	0.9	0.5

Table 1: Usual settings for total number of agents in the model: applying the definition for n_{rand} and n_{rat}, we finally have 180 random investors, 20 rational investors, 10 consultants.

The effect of the risk-free rate r_f on the simulation is negligible since is the same for the investment decisions of all the rational agents; moreover we are considering only equity funds for the investors and so there is no trade-off between the risky portfolio and the risk-free asset.

6.2.1 n_{rand} and n_{rat} : heterogeneous expectations and market volatility

In this work we do not deeply investigate the role of n_{rand} in the model. Other works in the literature of complex financial systems[22]consider the impact of different expectational models in a stock market. In the framework of standard financial theory (*Efficient Market Hypothesis*) the homogeneous expectations about future asset prices cannot produce complex patterns usually observed in real markets (bubbles or crashes, high price volatility, significant trading volume), while heterogeneous expectations can offer realistic outcomes. In our ABM we are clearly in the field of heterogeneous expectations since every random agent invest according to her personal strategy; the rational agents, from the other side, can consult and change their analyst, so getting different optimal portfolio estimates (driven by different expectations about future asset returns according to the consultants). Even having the same consultant for the current trading day (i.e. the same optimal portfolio goal), two rational investors can follow two different trading strategies to (try to) reach the same equity fund composition. This was shown in mathematical details in Chapter 2, explaining the rational investment scheme. In conclusion in our work, beside heterogeneous expectations, we have a great degree of freedom in trading strategies for investors (as it should be in a real market). If we run the ABM in an absurd market in which there are only rational investors who take advice from a unique consultant, there will be anyway significant price volatility and trading volume.

[22]see in particular *Asset Pricing Under Endogenous Expectations in an Artificial Stock Market* in W. B. Arthur, S. N. Durlauf, D. Lane (1997) *The economy as an evolving complex system II.*

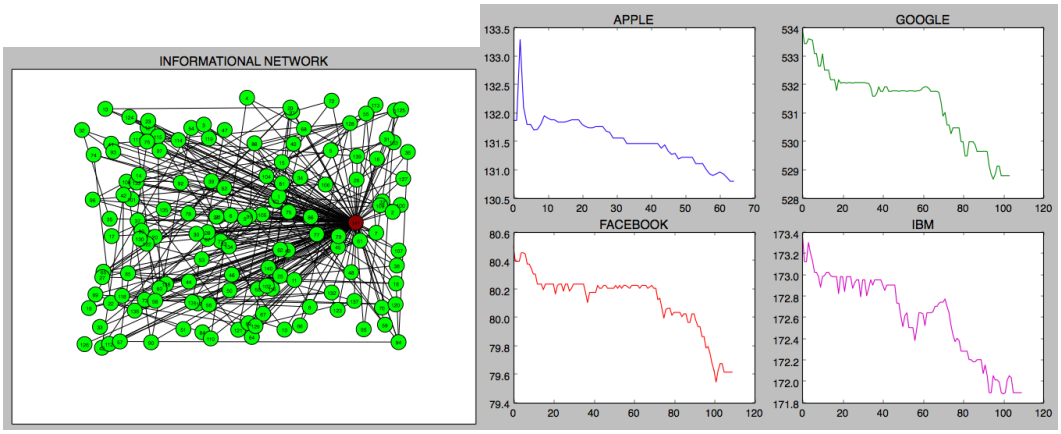


Figure 12: Informational network and daily stock prices in a *rational homogeneous expectations* market.

The role of n_{rand} has to be therefore more subtle and all its implications are difficult to illustrate before a complete analysis of this ABM. So we just limit our discussion to see what kind of general effect can have, for example, on global market volatility, which is a relevant property to measure.

In the method *PlotDailyPrices* contained in *oActions.py* we can add the possibility to compute the mean standard deviation of daily stock prices (the mean is over all the stocks). At every day we get this mean value appending it to a global list and, at the end of the simulation, we can compute the mean over all such daily average standard deviations, getting the final value

$$\sigma_{market} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{i=1}^{N}\sigma_i$$

where T is the total number of simulation trading days, while σ_i is the standard deviation of daily prices of stock i. σ_{market} is then an average both over time and over all securities and is a good measure to catch global prices volatility when we run a simulation. In the following analysis we shall use different times this estimator in relation to different control parameters. Here we simply verify if does exist some relationship between n_{rand} and σ_{market}. So we lunch the simulation on a horizon $T = 10$ trading days, maintaining $M = 200$ investors, and changing every time n_{rand} to sign the corresponding σ_{market} value. Note that the choice of the other parameters will be clear later in the next sections: it is suffice to understand that we choose to operate in a no transitions scenario, running the simulation with a horizon of $T = 10$ trading days.

σ_{market}	0.5561	0.5072	0.7310	0.5489	0.6723	0.5874	0.4539	0.5716	0.7866
n_{rand}	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9

Table 2: Table with σ_{market} ($seed = 10$) vs n_{rand} with the following parameters setting: $\Delta n_{trade} = 0.1$; $\epsilon_{test} = 10^7$, $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no transitions scenario and fixed consultancy).

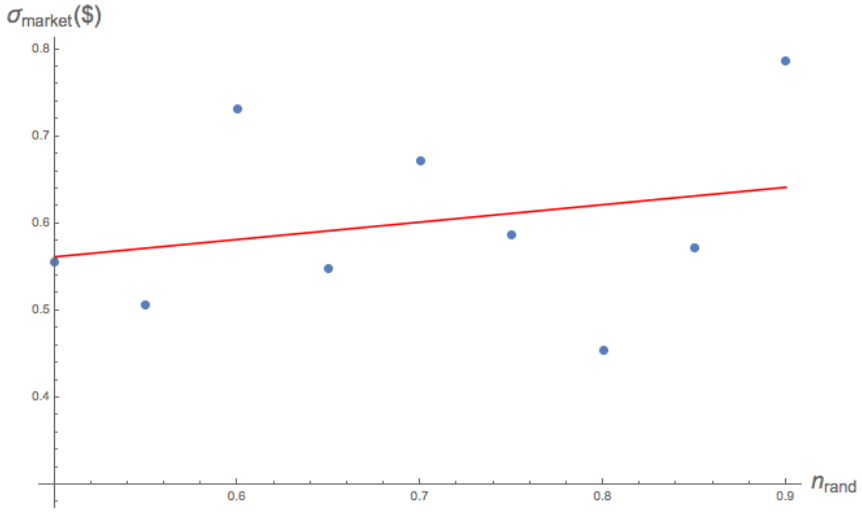


Figure 13: Plot with σ_{market} ($seed = 10$) vs n_{rand} with the following parameters setting: $\Delta n_{trade} = 0.1$; $\epsilon_{test} = 10^7$, $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no transitions scenario and fixed consultancy).

From this plot we can see how there are statistical fluctuations in σ_{market} , due to the stochastic nature of stock price dynamics, with a fixed seed. In order to have a better relationship with less errors fluctuations we should consider an average value of σ_{market} over different seeds; this is not our goal here, since we are simply interested in catching the qualitative behavior of market volatility with respect to n_{rand}. Therefore the linear fit is sufficient to see that *increasing n_{rand}, which corresponds to reduce the fraction of rational investors, market volatility tends to increase. The presence of rational traders helps to reduce mean prices fluctuations and this is in some sense related to the benchmark situation of homogeneous expectations as in the EMH. To well motivate such assertion we have to underline that the effect of n_{rand} on σ_{market} depends also by the relative number of consultants in the market, n_{consul}, and by their composition. If the heterogeneity of our analysts is limited then augmenting the number of rational investors is the same to increase the number of traders who have similar expectational models, so restricting the expectations space and lowering market volatility* (see Chapter 1 about the SFI market). *The problem is that in our ABM there is the possibility that multiple investors can have the same consultant, so investing with the same pricing model and the same portfolio goal; in addition there can be many consultants of the same kind. For both these reasons rational traders can share similar trading strategies, so reducing heterogeneous expectations: we are then getting a little closer to homogeneous expectations framework.*

Presenting the list of consultants used in our ABM we can better know why the above considerations are reasonable.

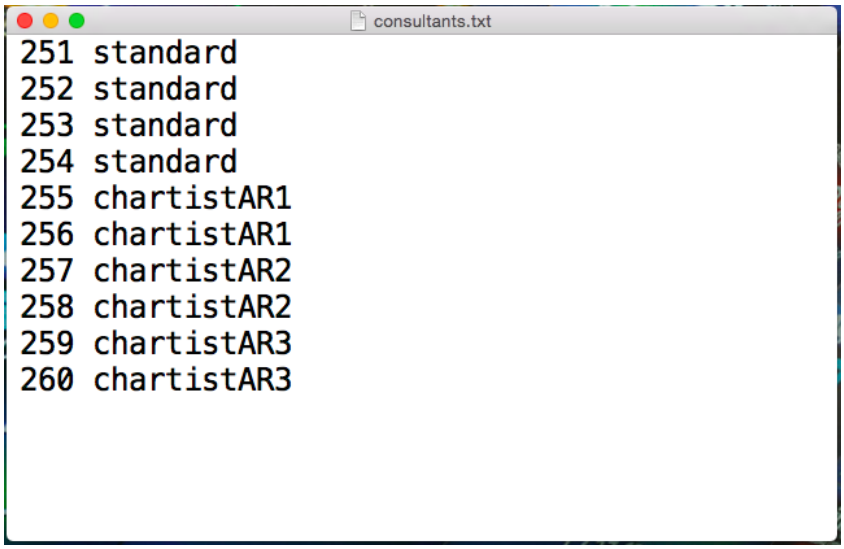


Figure 14: The list of all consultants in the simulation is contained in the file *consultants.txt*. In this work we choose 10 analysts (half of the rational trader, using $n_{rand} = 0.9$): 4 standard consultants and 6 chartists, two for each kind of AR process.

Heterogeneous expectations is guaranteed by the existence of the four classes of consultancy, while the presence of multiple analysts of the same kind go instead towards the idea of homogeneous expectations. We appreciate the existence of some analysts with the same pricing method because in this way we give higher chance to rational investors to choose a particular consultant (or changing kind of consultant if the parameters setting favors this possibility).

For the rest of the work we maintain such consultants list and, from the other side, we decide to include a tenth of the investors as rational traders ($n_{rat} = 1 - n_{rand} = 0.1$, i.e. $n_{rand} = 0.9$ as mentioned at the beginning of the discussion). This is a good benchmark to guarantee heterogeneous expectations and, at the same time, to test if there are advantages from consultancy.

6.2.2 $\triangle n_{trade}$: bid/ask price flexibility and market volatility

One of the most important element in this ABM is the source of freedom, for every investor, to set the bid/ask price. The baseline size of an order, given its magnitude $|\Delta W_i|$, is $floor[\frac{|\Delta W_i|}{p_i}]$: if you are going to buy stocks you can drop a random quantity drawn by $U(0, \triangle n_{trade} floor[\frac{|\Delta W_i|}{p_i}])$ from such size and increase the bid price; if you are going to sell stocks you can instead add such quantity and reduce the ask price. So $\triangle n_{trade}$ determines the flexibility of bid/ask prices and, in a market which is greatly determined by the interaction double auction system, such volatility affects daily prices fluctuations, i.e. global market volatility. If we set a too high $\triangle n_{trade}$ we could find unrealistic fluctuations in stock prices and then it is important to calibrate such parameter in an appropriate range.

In particular σ_{market} depends on $\triangle n_{trade}$ and we analyze their relationship in the following way: let us fix the seed and the time horizon T and run the simulation at different $\triangle n_{trade}$ values, computing at each time the corresponding σ_{market}. The results are showed below, with $seed = 10$ and time horizon $T = 10$.

σ_{market}	0.0.1994	0.3626	0.8584	3.3173	7.1329	9.1026	13.2770
$\triangle n_{trade}$	0	0.0005	0.001	0.005	0.01	0.015	0.02

σ_{market}	9.0925	16.0884	17.2852	25.4404	38.1321	20.1581
$\triangle n_{trade}$	0.025	0.03	0.035	0.04	0.045	0.05

Table 3: Table with σ_{market} vs $\triangle n_{trade}$ ($seed = 10$) with the following parameters setting: $n_{rand} = 0.9$; $\epsilon_{test} = 10000$, $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no transitions scenario and fixed consultancy).

The choice to not extend our search beyond $\triangle n_{trade} = 0.05$ (which means a 5% maximum variation in the number of shares to buy or sell) is due to the desire to have realistic values for σ_{market} , as shall will discuss below speaking about calibration. Now let us plot the data and show a ordinary least squares line to better visualize the trend.

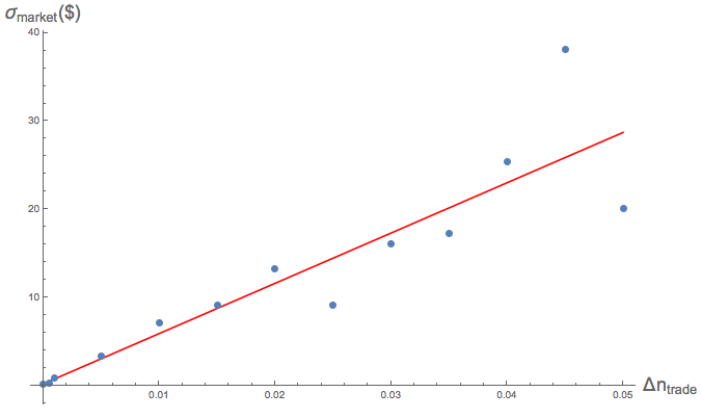


Figure 15:
Plot with σ_{market} vs $\triangle n_{trade}$ ($seed = 10$) with the following parameters setting: $n_{rand} = 0.9$; $\epsilon_{test} = 10000$, $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no transitions scenario and fixed consultancy).

There again (as previously seen in relation with n_{rand}) statistical fluctuations in σ_{market}: augmenting $\triangle n_{trade}$ we are no always sure that prices volatility (i.e. σ_{market}) rises, with a fixed seed. These fluctuations becomes more significant when we go toward higher values of $\triangle n_{trade}$ (as can be seen from the plot) and this is another reason for which we have no reported other data beyond $\triangle n_{trade} = 0.05$. In order to achieve a more general and accurate representation, so reducing statistical fluctuations, we could extend the time horizon T and also taking an average value over different seeds. This work is not performed here because there is no need to get a more accurate relationship.

Calibration using a real sample The important request is instead to calibrate the control parameter $\triangle n_{trade}$ such to find realistic stock prices volatility (represented by σ_{market}): this work can be done observing daily prices fluctuations from real markets. So this time we are looking at intra-day prices data

for our four stocks[23]: the sample is taken from the most recent history and the frequency is minute by minute (almost 390 daily stock price values for a single security). It is suffice to repeat the same algorithm previously used to get a real estimate of σ_{market}: using the same horizon ($T = 10$ trading days) we got

$$\sigma^{real}_{market} = 0.61\$$$

This is a σ_{market} value specific for this sample and, in principle, we should use several samples (i.e. different series of ten trading days, without overlapping) to get a complete range for σ^{real}_{market}. Anyway in our work, as we shall explain in Chapter 6, we are really interested to align our market to a specific real sample. For this reason, even maintaining the simulated market as totally independent by the real one (while in Chapter 6 we shall try to bind the two worlds), we desire a simulation in which σ_{market} is close to our sample-specific σ^{real}_{market}. To do this we have to use a calibration curve in order to define an allowed range for $\triangle n_{trade}$. The previous estimated OLS line can be a good calibration curve, so let us show the two regression coefficients

$$a = 0.218431$$

$$b = 570.011$$

where a is the intercept and b the angular coefficient. Suppose we want σ_{market} in a range $(0.50 - 1.00)\$$, which is a good benchmark if we look to different real data samples (considering also the same four stocks): it is suffice to solve these two separate equations

$$a + b\triangle n_{trade} = 0.5 \Rightarrow \triangle n_{trade} = 0.0005$$

$$a + b\triangle n_{trade} = 1 \Rightarrow \triangle n_{trade} = 0.0014$$

to get a reasonable range $\triangle n_{trade} = (0.0005 - 0.0014)$. There is also the possibility to calibrate $\triangle n_{trade}$ such to have $\sigma_{market} = \sigma^{real}_{market} = 0.61\$$

$$a + b\triangle n_{trade} = 0.61 \Rightarrow \triangle n^{real}_{trade} = 0.0007$$

so giving our market a statistical property which coincides with that of the real stock market. *From now on we choose to consider only $\triangle n_{trade}$ values within the estimated range* $(0.0005 - 0.0014)$.

Some comments on the calibration process are really necessary before continue in the discussion of other parameters. In a more general ABM framework we could make calibration using, as previously reminded, more samples to make a set of estimates for $\triangle n^{real}_{trade}$ (or an average among all them). Anyway, as will become more evident from the next chapter, the key element in all this work is looking for the parallelisms or contrasts between a totally simulated market and simulated market kept aligned to a real one. So the sample used to make such alignment is unique and control parameters have to be fixed, even in a totally simulated framework, on its basis.

6.2.3 ϵ_{test}: herd effect in the simulation

In the previous paragraph we have worked in a no-transition regime, which with parameters settings such to avoid any possible transition $RAT \rightarrow RAND$ and $RAND \rightarrow RAT$: this corresponds to high values for the corresponding control parameters (ϵ_{test}, ϕ_{test}, x_{change}). Here we start our discussion regarding herd effect, in a market in which instead we do *not* allow:

[23] See Chapter 6 for the web reference used to get such data

- transitions of kind $RAND \rightarrow RAT$: so we maintain very high values for ϕ_{test} (we used $\phi_{test} = 100$);
- transitions of kind $RAT \rightarrow RAND$ due to network effects: this means high values for x_{change} (we used $x_{change} = 10$).

So in this environment there are no permanent transitions between the two classes of investors.

Herd effect on global market volatility As a first step in the discussion of herd effect we have to look at the global role of ϵ_{test} in the market. This can be possible monitoring once again σ_{market} at different values of ϵ_{test}. *The question is if increasing the probability of herd effect, we are actually increasing stock prices volatility, or if this effect is simply local with respect to single assets and to single trading days.* Now let us check if stock prices volatility can be affected: run the simulation on an horizon of $T = 10$ trading days and compute the corresponding σ_{market} values, with respect to the chosen threshold ϵ_{test}. However, beside the chosen value for ϵ_{test}, we have also to consider the actual herd effect during the simulation, which can be quantified by the overall number of herd transitions during all the simulation period. Therefore it is useful to add such data every time you launch a new experiment, denoting it as n_{tot}^{herd}.

σ_{market}	0.6251	1.1347	0.8243	0.9395	0.6850	0.9515	0.6021	0.6117	0.5818	0.8584
n_{tot}^{herd}	39	42	43	38	9	4	1	2	1	0
ϵ_{test}	1	5	10	50	100	200	500	1000	2000	10000

Table 4: Table with σ_{market} vs $Log_{10}(\epsilon_{test})$ and $Log_{10}(n_{tot}^{herd})$ ($seed = 10$) with the following parameters setting: $n_{rand} = 0.9$, $\triangle n_{trade} = 0.001$; $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no network effects and fixed consultancy).

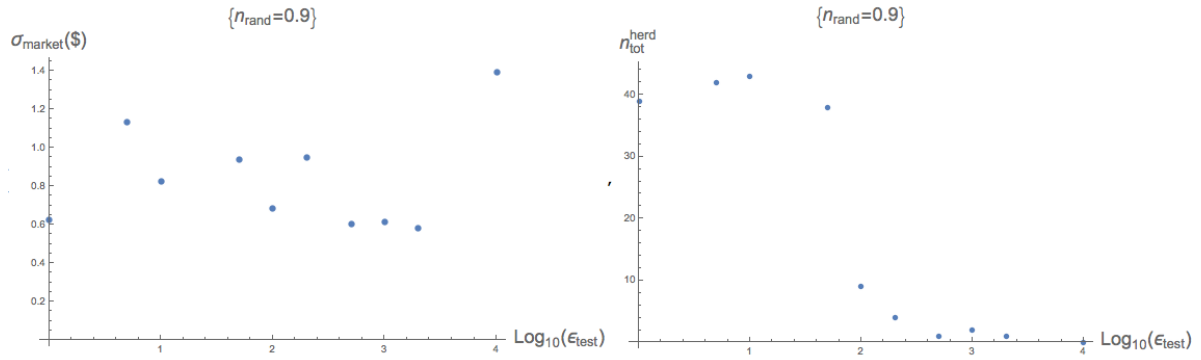


Figure 16: On the left: Plot with σ_{market} vs $Log_{10}(\epsilon_{test})$ ($seed = 10$) with the following parameters setting: $n_{rand} = 0.9$, $\triangle n_{trade} = 0.001$; $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no network effects and fixed consultancy). On the right: Plot with σ_{market} vs $Log_{10}(n_{tot}^{herd})$ ($seed = 10$) with the following parameters setting: $n_{rand} = 0.9$, $\triangle n_{trade} = 0.001$; $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no network effects and fixed consultancy).

The two plots above show two important things to take in mind for the rest of our work: the first one is that the global role of the herd effect on the mean prices volatility during the simulation is negligible *if we work with such agents compositions settings*, as you are going to explain; the second is that, as greatly predictable according its function, increasing ϵ_{test} reduces the number of herd transitions n_{tot}^{herd}, *but not in a deterministic way.* Let analyze these two implications.

Why market volatility, expressed by σ_{market}, does not show any kind of relationship with ϵ_{test}? A possible answer is that the number of herd transitions actually occurred is always too low (with respect to the global number of orders made), even with small values of ϵ_{test} , since we have just few rational investors in the market; so the observed number n_{tot}^{herd} does not have a significant impact on stock prices history. If we want to investigate such possibility we have to change the settings of the simulation in this way

M	n_{rand}	n_{consul}
200	0.5	0.2

σ_{market}	4.6606	1.2166	3.0298	3.8688	1.5825	1.9177	1.1545	1.5421	0.7808	1.3935
n_{tot}^{herd}	841	335	714	526	220	221	185	210	167	107
ϵ_{test}	1	5	10	50	100	200	500	1000	2000	10000

Table 5: Plot with σ_{market} vs $Log_{10}(\epsilon_{test})$ and $Log_{10}(n_{tot}^{herd})$ $(seed = 10)$ with the following parameters setting: $n_{rand} = 0.5$, $\triangle n_{trade} = 0.001$; $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no network effects and fixed consultancy).

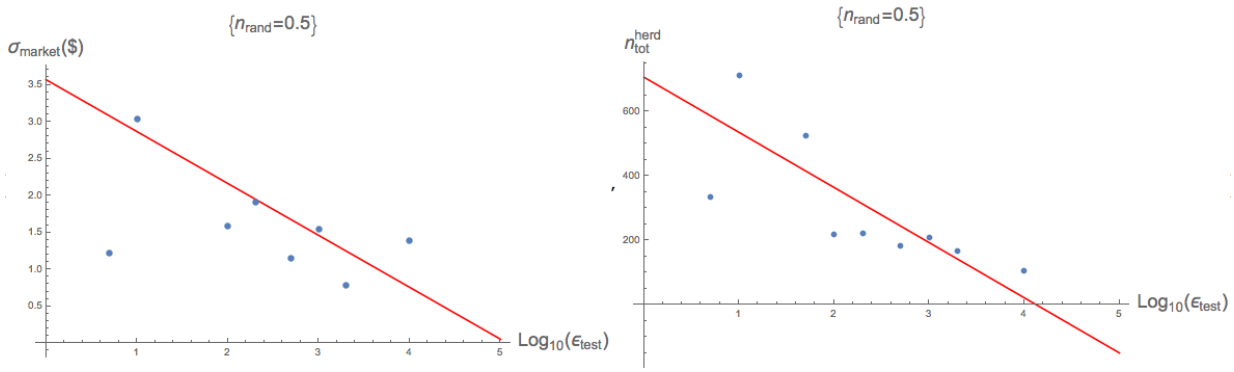


Figure 17: On the left: Plot with σ_{market} vs $Log_{10}(\epsilon_{test})$ $(seed = 10)$ with the following parameters setting: $n_{rand} = 0.5$, $\triangle n_{trade} = 0.001$; $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no network effects and fixed consultancy). On the right: Plot with n_{tot}^{herd} vs $Log_{10}(\epsilon_{test})$ $(seed = 10)$ with the following parameters setting: $n_{rand} = 0.5$, $\triangle n_{trade} = 0.001$; $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no network effects and fixed consultancy).

The magnitude of σ_{market} values are now higher than in the benchmark case with $n_{rand} = 0.9$ and the reason is that the herd transitions are more frequent, as a direct consequence of having more rational

investors. No herd transitions scenario can now be reached at very high ϵ_{test} values (typically we can use $\epsilon_{test} = 10^7$), not included in the table for simplicity.

Moreover, looking at the two corresponding plots above, we can see a more definite relationship for σ_{market} with respect to ϵ_{test} and the latter is a decreasing trend: as we could expect, augmenting ϵ_{test} implies less herd phenomena so stabilizing the overall market volatility during the simulation. Such behavior was hidden in the regime $n_{rand} = 0.9$ because the actual number of transitions is too small to affect global market volatility. So we can imagine there is s sort of threshold (not deterministically computable) for n_{tot}^{herd} beyond which ϵ_{test} has an effect on σ_{market} . *So what really matters is not ϵ_{test} values but the overall magnitude of herd transitions, resumed by n_{tot}^{herd}.* Indeed in both settings, $n_{rand} = 0.9$ and $n_{rand} = 0.5$, σ_{market} always increased if the actual total transitions rise: we do not plot such relation because it can be quite evident from previous data.

Now the other question is why augmenting ϵ_{test} we have a no deterministic relation with respect to n_{tot}^{herd}: we have found in both situations that the actual transitions decreased rising ϵ_{test} but with some noise. What is the origin of such noise? In order to get a good answer it is important to think about the overall simulation dynamics. Using the same seed (as done for all the paragraph comparing the regimes $n_{rand} = 0.9$ and $n_{rand} = 0.5$) the series of pseudorandom number drawn in Python should be the same: such series is used to perform all random operations related to network evolution, investment decisions and execution of transition tests. However when some herd transition occurs at the first trading days, the daily prices realization changes and so this also modifies the stock prices history, i.e. the closing prices included in global lists of consultants. If such value has changed, in the next trading days also the investment strategies of rational investors will be affected and, in particular, the selection algorithm (which explores the set of solution in a stochastic way) for all ΔW_t^js deviates from its previous direction. The series of drawn pseudorandom numbers is modified and this can alter the entire simulation, included, as an indirect consequence, the actions of random investors and finally the realization of stock daily prices. So there is the possibility, that even with higher ϵ_{test} values the total number of herd transitions n_{tot}^{herd} will be greater, since new scenarios arise during the simulation. This is a very complex effect, difficult to analyze in great detail since it entail the entire simulation history. As always if we took average values of n_{tot}^{herd} with different seeds we could reduce significantly the noise in this process. This is not our task since we are making just a qualitative discussion and such analysis will require too much time, without giving us new important results.

Impact of herd effect on single assets $RAT \to RAND$ due to herd effect is relative to every single asset because it occurs on the basis of the current status of the book. Then we are really interested to see the particular effect that a certain number of transition phenomena can have on a single stock price during a trading day. For the rest of the paragraph we show just in qualitative way the results, without a large statistical analysis: future works can look at the quantitative role of the parameter ϵ_{test} on single asset dynamics.

Since we have shown how herd effect can be consistent only if we have a consistent fraction of rational investors in the market, we work again with $n_{rand} = 0.5$ and we look the plot of daily stock prices changing ϵ_{test}. We show two regimes of herd transitions, with $\epsilon_{test} = 1$ and $\epsilon_{test} = 100$, always maintaining the same seed and comparing the same trading day. To summarize what happens on a particular day we use a list containing the number of herd transitions occurred for any of the asset: we use a parenthesis with letter B if the transitions have occurred on the buying side, letter S if the transitions have occurred on the selling side; if there are mixed transition we report both.

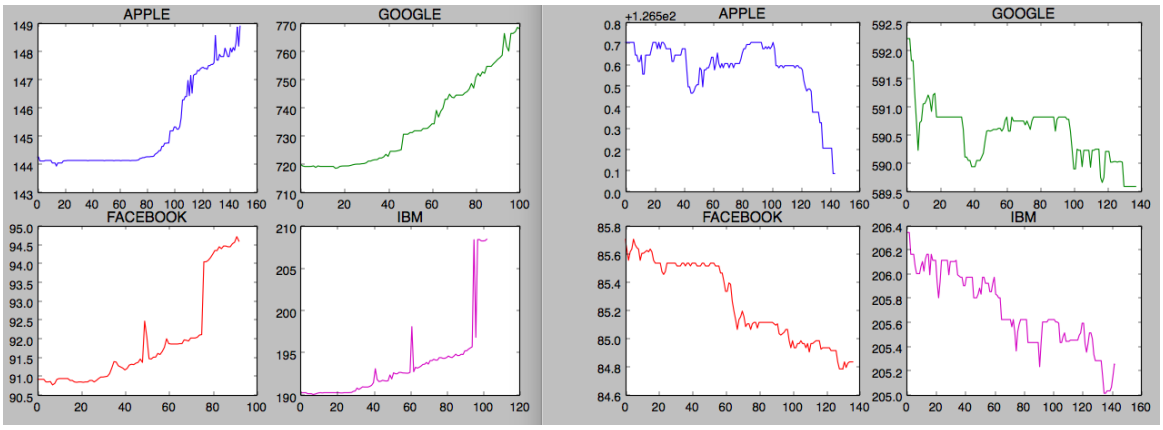


Figure 18: Daily stock prices at trading day $t = 10$ ($seed = 10$) with the following parameters setting: $n_{rand} = 0.5$, $\triangle n_{trade} = 0.001$; $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no network effects and fixed consultancy). On the left panel ($\epsilon_{test} = 1$) there have been the following transitions: $41(B)$ $45(B)$ $40(B)$ $41(B)$; on the right panel ($\epsilon_{test} = 100$) there have been the following transitions: 0 0 $1(S)$ 0 .

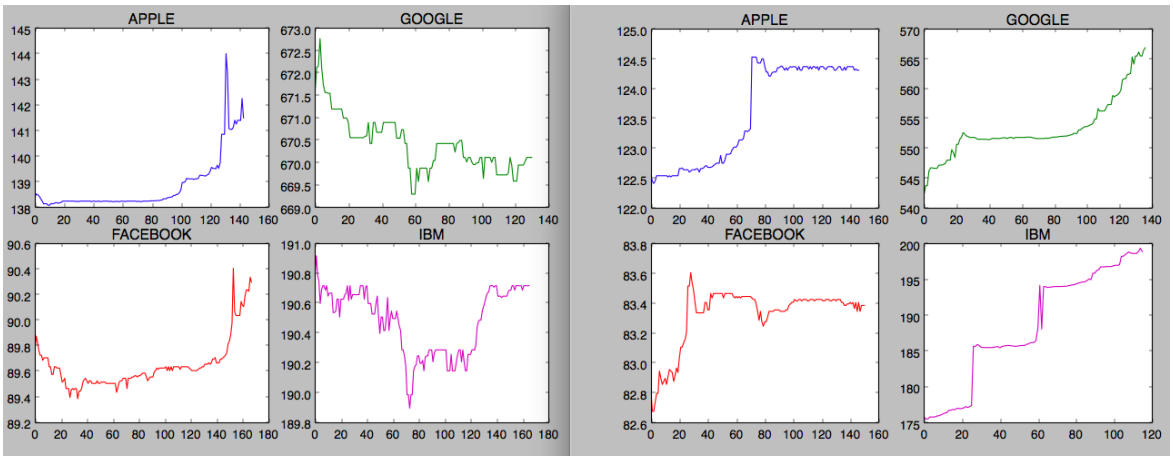


Figure 19: Daily stock prices at trading day $t = 8$ ($seed = 10$) with the following parameters setting: $n_{rand} = 0.5$, $\triangle n_{trade} = 0.001$; $\phi_{test} = 100$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no network effects and fixed consultancy). On the left panel ($\epsilon_{test} = 1$) there have been the following transitions: $27(B) - 1(S)$ $1(B) - 5(S)$ $46(B) - 3(S)$ $5(S)$; on the right panel ($\epsilon_{test} = 100.$) there have been the following transitions: 0 16 $5(B)$ $30(B)$.

Here we have shown the plots for two different trading days and this is suffice two observe some interesting feature about this phenomenon:

- the occurrence of a significant number of herd transitions correspond always to a bull (bear) trend when the transitions are on the buying (selling) side. More there are transitions and more straightforward is the trend. You can see such evidence from both plots.

- during a trading day mixed transitions on an auction market, i.e. for the same stock, are typically rare but can occur. Moreover their relative existence is imbalanced, namely the number of transitions on one side of the book is significantly higher than those on the other side. The second plot above reported is a good example for this situation.

- the total number of daily herd transitions does not always reflect the parameter ϵ_{test}. At a certain trading day we cannot say a priori that the total daily transitions will be greater in a market setting with higher ϵ_{test}: the case above illustrated in the second plot is of course infrequent in the horizon of the entire simulation, but we have included it here to explain such eventuality.

Now let us try to comment and give a good explanation to all such observations.

The first point is what we have been always desired from the introduction of a herd mechanism. A rational trader can be influenced from the current imbalance of a book side with respect to the other, so changing her investment decision. *The imbalance is initially due simply to the actions of random traders, but if other rational investors will be affected by herd behavior they can themselves fuel the effect: so herd behavior displays a self-fulfilling relation (a sort of positive feedback). If we have a sufficient number of herd transitions a chain reaction is triggered and the dominant side of the book is further enforced, so augmenting the probability of new transitions.* This is just an emerging property from collective behavior and is related to a real form of trading interaction, i.e. the electronic book. Note that the self-fulfilling effect is clearly consistent augmenting the number of rational investors.

Regarding mixed transitions, if we pay more attention to the latter considerations we convince us that these situations are possible, though rare. It can happen that some initial transition occurs on one side of the book but the operations of the other traders do not lead to a significant bull (or bear) trend in that direction; on the contrary the trend arise on the opposite sense and then herd transitions will occur on the opposite side with respect to the original ones. This can be verified through the visualization of time order for the transitions.

The most surprising effect is however the possibility that, at some trading day, the number of total daily herd transitions is higher in a regime with lower ϵ_{test} value. As we have tried to explain in previous paragraph showing the results about market volatility, herd transitions can change the sequence of daily investment strategies adopted by rational investors. Therefore, if the simulation lasted for a single day the number of daily transitions would be always lower increasing ϵ_{test} : we can note such intuition looking at the first trading step. Starting from the same investment plan for all rational traders (the seed is indeed the same) the drawn values of ϵ are the same, but the threshold ϵ_{test} has changed, so lowering the actual transitions according to the expected manner. Anyway from the second trading day the stock prices history was modified and the future strategies of rational traders can change, together with other processes related to the drawn pseudorandom numbers. This can lead to such strange phenomenon and you can convince of all these happenings running the simulations with different seeds.

Finally we want to note that the bubbles or crashes here observed are *intra-day phenomena*, which can be not necessarily related to bubbles or crashes on a wider time horizon of several trading days. Of course herd transitions will tend to favor bull or bear trends over more trading days, but there is no deterministic relation which ensures this. The opposite can also occur, i.e we can observe bubbles or crashes during some trading days, without the existence of actual herd transitions and you can easily check this running the simulations at different seeds and over a very long time horizon. We do not show explicitly such results

because are quite simple to verify and maybe future works can specifically study the relationship between herd phenomena and long-run bubbles or crashes.

Disruption of rational investment strategy Herd effect has not only a direct impact on stock prices dynamics. A rational trader who is affected by herd behavior her trading strategies are upset by the unexpected transition. The agent was investing with one possible strategy which leads to the optimal portfolio (or which can get close to it) but the herd transition cancel the strategy, even if it occurs in relation with just one stock. So we can imagine that, as a mean effect, increasing ϵ_{test} implies to disrupt optimal strategy of rational agents and then *can* obstruct the construction of the optimal portfolio.

This hypothesis can be checked using the output of method *checkPortfolio* in *oActions.py*: you can monitor, at each trading day, which investors have reached or proxied their optimal portfolio goal. If you operate with a particular seed and count the number of investors who have proxied the optimal portfolio, at different ϵ_{test} values and on the horizon of many trading days, you can verify all what we have said here. Once again it is better to look a t the relationship between the actual number of herd transitions n_{tot}^{herd} and the number of investors who have got close the optimal fund composition. Here we do not show the data relative to this phenomenon just because is not one of our priorities. *Indeed what is really interesting are the mean profits of our investors and the latter can be good, i.e. above the global mean in the market, even adopting a sub-optimal strategy during the simulation: that is another effect of a complex market.*

Having explained in a qualitative way all the effects of herd behavior in the simulation, which is the appropriate regime for ϵ_{test} in our simulation? Or, in other words, how many average herd transitions we want to accept in our stock market? Allowing too many herd transitions is not a good idea if we want to analyze the important results of the ABM. As we have seen herd effect is a sort of noise effect because disrupts rational strategies and distorts stock prices trend, leading, as an extreme effect, to bubbles or crashes during the simulation horizon. In other works it could be very interesting pledging time to explore all the outcomes of the market in a regime of high herd effect. However for the remaining analysis we shall operate in a no herd transitions environment, so giving the chance to rational investors to always perform all the trading operations needed to the construction of the optimal portfolio goal.

6.2.4 ϕ_{test}: new rational agents in the market

When a random trader observes for many days that her profits are decreasing she can think about starting to call an analyst as guidance for trading. First we set the parameters setting such to remove:

- herd transitions $RAT \to RAND$: so we maintain very high values for ϵ_{test} (we used $\phi_{test} = 10^7$);
- transitions of kind $RAT \to RAND$ due to network effects: this means high values for x_{change} (we used $x_{change} = 10$).

We are simply consider $RAND \to RAT$ transitions, so having a possible increase in the number of rational investors. Let us run the simulation for a horizon $T = 10$ trading days at different ϕ_{test} values and look at the corresponding σ_{market} and n_{tot}^{rat} measures.

σ_{market}	0.4639	0.6374	0.6114	0.5817	0.6722	1.0120	0.9315	0.7867
n_{tot}^{rat}	180	178	177	167	99	18	2	0
ϕ_{test}	0.1	0.5	1	2	5	10	15	50

Table 6: Table with σ_{market} vs ϕ_{test} and n_{tot}^{rat} ($seed = 10$) with the following parameters setting: $n_{rand} = 0.9$, $\triangle n_{trade} = 0.001$; $\epsilon_{test} = 10^7$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no network effects and fixed consultancy).

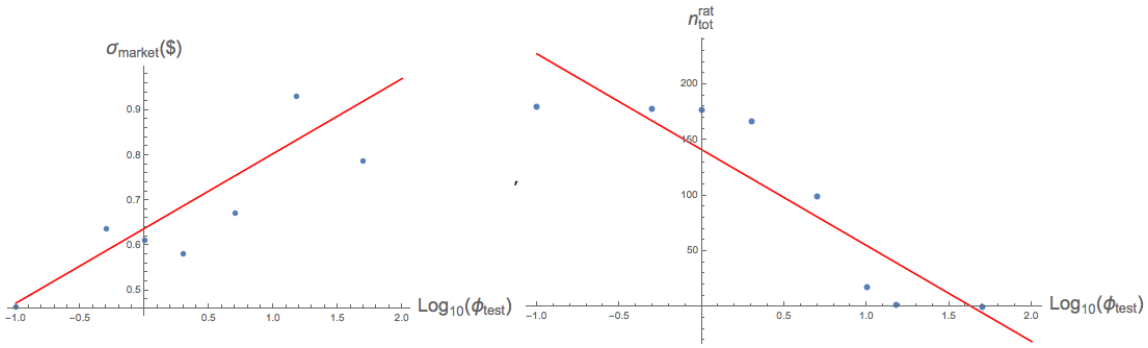


Figure 20: On the left: Plot with σ_{market} vs $Log_{10}(\phi_{test})$ ($seed = 10$) with the following parameters setting: $n_{rand} = 0.9$, $\triangle n_{trade} = 0.001$; $\epsilon_{test} = 10^7$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no network effects and fixed consultancy). On the right: Plot with n_{tot}^{rat} vs $Log_{10}(\phi_{test})$ ($seed = 10$) with the following parameters setting: $n_{rand} = 0.9$, $\triangle n_{trade} = 0.001$; $\epsilon_{test} = 10^7$, $x_{change} = 10$, $\lambda_{test} = 10$ and $p_{info} = 50$ (no network effects and fixed consultancy).

The plot shows how an increase in ϕ_{test} rises market volatility and lowers the total number of $RAND \rightarrow RAT$ transitions. Once again statistical fluctuations are significant in σ_{market} with a single seed but the simple linear fits catch the main trend. Instead n_{tot}^{rat} decrease in a deterministic way with respect to ϕ_{test}: this is in contrast to what happened for n_{tot}^{herd} with ϵ_{test}. We can then assert that the number of $RAND \rightarrow RAT$ transitions is a more controllable variable than the number of herd transitions, as compared to their corresponding control parameters. A thing to notice is the possibility to reach the maximum value of $RAND \rightarrow RAT$ transitions, which corresponds to the total number of random investors initially included in the simulation (which is 180 in our settings): as we decrease the values of ϕ_{test} we get closer and closer to this value for n_{tot}^{rat} , until the saturation is perfectly reached and we shall have only rational traders at the end of the simulation.

The fact that market volatility increases augmenting ϕ_{test}, i.e. augmenting the number of $RAND \rightarrow RAT$ transitions, agrees with all the previous results about the role of n_{rand} in the simulation. Indeed we have seen understood that an increase in the relative number of rational investors can help to reduce the level of heterogeneous expectations, then stabilizing market fluctuations. So if n_{tot}^{rat} increases, more and more random traders will become rational and market volatility decreases. This is just the opposite effect of herd transitions in some sense, although n_{tot}^{rat} is a less stochastic variable than n_{tot}^{herd}.

6.2.5 λ_{test} and p_{info}: consultancy frequency to update rational strategy

The construction of a rational strategy taking an optimal portfolio estimate from her own consultant is an attempt to realize persistent profits, namely to *beat the market*. At the first tick of the simulation the choice of the consultant is totally random, reflecting the lack of knowledge about the performances of the entire analysts collection. However, in our ABM, we have considered the possibility that a rational trader decides to update their optimal portfolio composition, asking advice for a new consultancy. This chance is controlled by the parameter λ_{test} which fixes the probability to make a new call to some consultant in the market (the same of the previous day or a new one) and such probability is a function of $n_{\vec{w}'}$ (the number of days spent with the same portfolio goal vector) and p_{info} (the price paid for a single consultancy). Which is the effect of varying the consultancy frequency in the market? If a rational investor decides to call again a consultant, she can either consult the last analyst who has relied on, or she can call a new analyst in the market, according to the influence of her neighbors in the informational network (network effects). In both cases this choice entails an updating of her portfolio goal and a consequent learning mechanism.

If the investor consults again her past analyst, she will use the same expectational model as a guideline in her investment operations, but, with new available data, the forecasts are better calibrated in relation to new market dynamics: we do not know if this could be an advantage or not, because in a complex market, even having the expectational model which is trained on better and most recent data, the actual performance in investment strategy is not guaranteed.

If, as opposite chance, the investor calls a new consultant (advised from some of her acquaintances), she will adopt a new expectational model for the portfolio goal composition. Once again, it is very difficult to know a priori, which is the model which performs better (we try to face the problem in Chapter 7); even if the new pricing model is better, the trader must manage to construct the new optimal portfolio (or at least a proxy of it) and this is not always possible.

So the lesson is that both new consultancy possibilities could not necessarily lead to an improvement of actual profits. This is the challenge of complexity in a stock market, for most part due to the double auction trading system. Anyway we can look to a decrease in λ_{test} and p_{info}, which rises the probability portfolio goal updating, as an increase in the learning rate for rational investors in that market. Making a parallelism with our introduction to ABM markets, *you can compare the role of λ_{test} (and p_{info}) to that one of the exploration rate used by the Genetic Algorithm in the SFI market. Rising the rate at which the GA works we had an increase in the explored volume of expectations space; in our ABM the effect λ_{test} about the class of expectational models used is more complicated. Applying for new consultancy, rational agents could become clients of the same consultant and this would reduce the belief space; but it could happen also the opposite situation, with traders who call new different analysts, so enlarging the class of expectational models applied in the market. Even updating the optimal strategy maintaining the same consultant is an operation that involves the use of a new pricing model (with different parameters but of the same class).* The mechanism used in the SFI market is obviously more elegant and clear and the reason is that is not modeled on social interactions and personal choices of the traders. Instead our artificial stock market considers all such phenomena.

Too high or too low learning rate it is possible to imagine that, in the long-run horizon (as some months), updating the optimal portfolio goal can become a winning strategy because the training sample in the learning process is significantly augmented and the forecasts about asset returns have substantially changed. So too high λ_{test} and p_{info} limit learning activity and, in the long-run, can reduce portfolio performances. The downside of reducing λ_{test} and p_{info} (i.e rising the probability of new consultancy) is that a too frequent optimal portfolio updating can represent an obstacle and not a learning source: in

our complex market building the optimal portfolio is difficult and can require some trading days, so if we change too many times the objective in portfolio shares, investors will never reach a satisfactory proxy of the optimal portfolio.

In conclusion too high or too low learning rates (represented by consultancy frequency) both take their advantages. In what kind of regime we want to run our simulation? *Our goal is to observe if investors can realize persistent profits in the short-run horizon (until two weeks of trading) and so it is better to operate with low consultancy frequency, using high values for λ_{test} and p_{info} : in other words we want that a rational investor maintain her starting portfolio goal composition for the rest of the simulation, or, in the worst possible scenario, who can update its strategy just one time.* This is sufficient to test if there is an advantage to modify the rational investment plan during few trading days: this work shall be displayed in Chapter 7.

Now we check what can be a good choice of λ_{test} to have first a scenario with zero new consultancies (on a horizon of $T = 10$ trading days), and second a scenario with an average of one new consultancy per investor (which corresponds to a total number n_{tot}^{cons} almost equal to the total number of rational investors). Running the simulation and reporting the results we have

n_{tot}^{cons}	41	16	0	0
λ_{test}	0.01	0.1	1	10

Table 7: Table with n_{tot}^{cons} vs λ_{test} ($seed = 10$) with the following parameters setting: $n_{rand} = 0.9$, $\triangle n_{trade} = 0.001$; $\epsilon_{test} = 10^7$, $\phi_{test} = 100$, $x_{change} = 10$ and $p_{info} = 50$ (no transitions scenario).

We can immediately note that the value $\lambda_{test} = 10$ used in all the previous settings is within the scenario of no new consultancies, also called *no learning update* scenario. Decreasing λ_{test} n_{tot}^{cons} always rises, without any fluctuations. With $\lambda_{test} = 0.1$ there are 16 new consultancy calls (and no multiple calls for a single trader), which means that almost every rational investor has decide to make a single update to her trading strategy (remember that we have 20 rational investors in the simulation). *This is just what we can accept if we work on short-run horizon: an average update frequency of one call approximately every 5 trading days.* So rational traders have the proper time to try to build a good proxy of their initial portfolio goal and then they can attempt to update the portfolio composition. The regime $\lambda_{test} = 0.01$ leads instead to a too high learning rate, with all rational investors who ask, at least, for two new consultancies during the simulation. The risk in adopting such rate in the learning process is that is difficult to well evaluate the actual performances of trading strategies; there is few time to complete the proxy process of optimal portfolio and, in addition, is difficult to judge its actual result on the market if the reached composition is tested for just one or few days. In other words we want the investors can freely try to build their portfolio goal, without any interruption, and we want that the built proxy for this optimal portfolio can become more stable to be correctly evaluated from a statistical point of view.

Learning rate as a consequence of network effects The last comment is about what actually happens in the simulation if the test for new consultancy is positive: in this case network effects will become relevant since the decision of changing consultancy or not is related to which were the profits of your neighbors. In this sense is very relevant the control parameter x_{change} which fixes the threshold for accepting the advice of one of your acquaintances. All the previous considerations were assuming not too high x_{change}

values, such to enable that network effects occur and guarantee an actual updating of rational strategies. Anyway the data regarding n_{tot}^{cons} considered above present both cases in which there are network effects or not. Therefore it is important to underline that deciding for a new consultancy is just the first step in the learning process, while the second step is indeed related to network effects as we shall see below. The latter effect is triggered by the first one but does not necessarily happen even is there is a good consultancy frequency.

6.2.6 x_{change}: controlling network effects

When a rational investor decide to update her portfolio goal she will look at her neighbors in the informational network in order to decide what will be her next choice. This is defined, both in economics and complex systems science, as *network effects*. The behavior of an agent is influenced by the other agent who interact with her, i.e by her neighbors in the network which defines the complex system. This ABM market wants to mimic the social interactions of agents through the powerful tool of the informational network. Now it is interesting to consider in more detail such effects, which was explained in Chapter 2 when we have illustrate the rules of links *investor* $\rightarrow$ *consultant*.

Remind that if you have chosen to update your investment strategy you have to speak with all your acquaintances asking which were their recent daily profits. In particular you will consider only the advice of your neighbor with the best profits, since it can appear as the most reliable among all your current acquaintances. Now the control parameter x_{change} plays its role verifying if the investor will consider the advice of your best neighbor or not. In particular x_{change} expresses how much the daily profits of your best neighbor has to exceed your daily profits. If you best acquaintance satisfy this request, you have to look what kind of investor she is: if she behaves herself as a rational trader you will call her consultant (which could be a new one or the same previously called); if instead she is a random agent there occurs a permanent $RAT \rightarrow RAND$. The latter kind of transition is very different from the herd $RAT \rightarrow RAND$ transition because is global choice for investment behavior and can last for all the remaining simulation. Here we decide to work, as usual, in a parameters setting such to exclude all the other transition phenomena, so removing:

- herd transitions $RAT \rightarrow RAND$: we maintain very high values for ϵ_{test} (we used $\phi_{test} = 10^7$);
- transitions of kind $RAT \rightarrow RAND$ due to network effects: this means high values for x_{change} (we used $x_{change} = 10$).

As anticipated before, network effects are triggered by the choice of new consultancy and so the same is for permanent $RAT \rightarrow RAND$ transitions. So we monitor, at a given seed and on a simulation horizon of $T = 10$, the number of such transitions, n_{tot}^{rand}, in relation to different values for x_{change}; we also report the corresponding value of n_{tot}^{cons} to compare the two count measures.

n_{tot}^{rand}	10	10	10	9	9	9	7	6
σ_{market}	0.6187	0.6187	0.6187	0.6164	0.6164	0.6164	0.5943	0.5974
n_{tot}^{cons}	16	16	16	15	15	15	15	16
x_{change}	0.01	0.1	0.2	0.5	1	2	5	10

Table 8: Table with n_{tot}^{rand} vs x_{change} ($seed = 10$) with the following parameters setting: $n_{rand} = 0.9$, $\triangle n_{trade} = 0.001$; $\epsilon_{test} = 10^7$, $\phi_{test} = 100$, $\lambda_{test} = 0.1$ and $p_{info} = 50$ (no transitions scenario and new consultancy rate approximately fixed to 1 per investor during the simulation horizon of $T = 10$). We also consider n_{tot}^{cons} and σ_{market}

What we can evidently observe is that we have always $n_{tot}^{rand} < n_{tot}^{cons}$. This result is a direct consequence of all we have said in this paragraph and in the previous one: new consultancy requests trigger network effects and, as a part of them, also trigger permanent $RAT \rightarrow RAND$ transitions, but both will ultimately depend on x_{change} values. Moreover it is a good result to see that when x_{change} augments n_{tot}^{rand} always rises or at least remains constant: so there are no fluctuations in this sense because the permanent $RAT \rightarrow RAND$ transitions are deterministic transitions, i.e. they certainly occurs if the threshold represented by x_{change} is overtaken.

The interesting fact is instead the behavior of n_{tot}^{cons} with respect to variations in x_{change} . Though we are maintaining the same value of $\lambda_{test} = 0.1$, i.e. the same probability for a new consultancy, n_{tot}^{cons} is varying just a little at different x_{change} regimes; this is in some sense due to noise in network effects. When a permanent $RAT \rightarrow RAND$ transition occurs there is a fewer rational trader in the market, so a fewer trader which can require a new consultancy. If the transition has affected a particular investor who was close to a new consultancy (i.e. in one or two trading days will be sure to make a new call), this can deny a new consultancy decision which was virtually certain in the next trading days. So there is a source of noise related to the particular state of the network (in particular think about all the neighbors who influence a rational trader) that can introduce little fluctuations in n_{tot}^{cons}.

Another effect we can observe is a sort of *saturation* phenomenon in n_{tot}^{rand} when we reduce the values for x_{change}. Here we can easily imagine what happens in the market. Let us consider a fixed number of rational investors and an almost (for what said a moment ago) fixed n_{tot}^{cons} value (i.e. λ_{test} is fixed). For each rational agent the number of neighbor random traders is limited since an agent has a limited degree: therefore it will emerge a natural *saturation* for n_{tot}^{rand}, which is a very realistic effect in a finite size informational network.

Finally we comment the relation between σ_{market} and n_{tot}^{rand}. Market volatility, as we know, tends to increase when the relative fraction of random investors augments and such effect can be clearly verified from our table where an increase in n_{tot}^{rand} leads in general to higher σ_{market} values. There can be also fluctuations in the process (as you can see passing from $n_{tot}^{rand} = 6$ and $n_{tot}^{rand} = 7$) which can be explained by what can be the specific contribute in terms of trading operations (and consequent market volatility) of the rational investors who become random traders. It is anyway a marginal noise effect with a little impact on σ_{market} and can be easily removed mediating over more seeds.

6.2.7 Combining all transitions: balancing effects

In previous sections we have analyzed the qualitative effect of every single control parameter, stopping the other transitions, i.e working in a scenario in which just one phenomenon can affect the simulation. Now it is time to say few comments about a scenario in which all kinds of transitions are possible (which is the same to say that also network effects are consistent).

First it is essential to remind how an increase in the relative fraction of random investors means a higher market volatility, which is, from a theoretical perspective, a consequence of a reinforcement in heterogeneous expectations. Taking such fundamental concept in mind we summarize the qualitative effect of the three kind of transitions on σ_{market}, which is our naive estimator for market volatility.

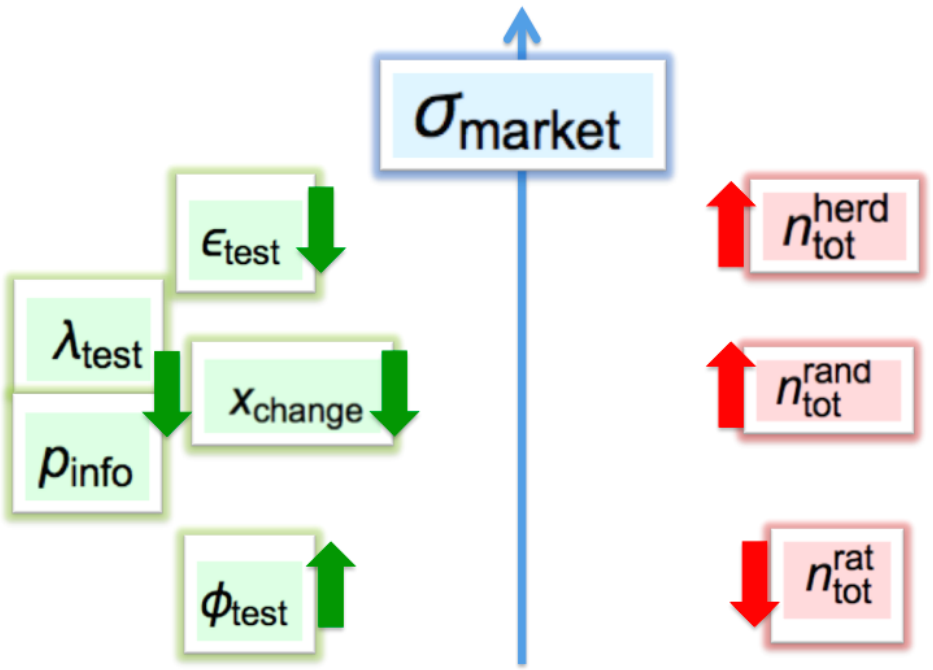


Figure 21: Qualitative scheme to summarize the effect of all transition phenomena on market volatility. On the left you have the control parameters which affect the three kind of transitions; on the right there are the actual total number of transitions during the simulation horizon.

Therefore the simple lesson to learn is that transitions towards random class implies an increase in market volatility and this is valid both for herd and permanent transitions of this kind. Instead $RAND \to RAT$ transitions has the contrary effect with a stabilization in market fluctuations.

In a scenario in which all transitions are possible the two pulses can balance each other: the idea is that the most likely kind of transition dominates on the other one. However the contrast is not so trivial because, for example, if there are many $RAND \to RAT$ transitions (and so market volatility would reduce), herd effects will become more frequent (since there are now more rational traders) and so there is an opposite force which tries to restore a general level of market volatility. This happens because herd transitions are just local transitions on the time scale of the simulation. From the other hand $RAND \to RAT$ transitions are quite independent from permanent $RAT \to RAND$ transitions due to network effects.

7 Alignment to a real market: introduce the arbitrageurs

All the previous considerations about the simulation are simply related to the rules set in our ABM. While the goals of works of this kind are similar and the knowledges about quantitative and behavioral finance are also the same, there are of course several degrees of freedom for the modeler in implementing a stock market simulation. Think for instance to all the random tests used to describe agents behavior or choices during trading days: there are many ways to conceive all these phenomena, not only from a mathematical point of view (we could use different distributions) but also from a practical perspective (it could be evenly valuable consider other parameters for behavioral tests). The structure of the informational network is itself related to the emerging rules set in the model (different initial networks or different algorithms to set new links at each trading days). Then, though the construction of such ABM is very interesting from a theoretical perspective, a real and totally reliable application of the simulation is possible if we find a way to make more realistic the evolution of stock prices. This can be achieved through the introduction of *fictitious arbitrageurs*. An arbitrageur is a trader who exploits temporarily misalignments between the price of an asset on two different markets: she buy the asset on the market where the price is lower and then sells it on the other market, at higher price, so realizing a risk-less profit without any net investment. According to the EMH arbitrage operations are not possible (rational market hypothesis) and in this framework the *no-arbitrage condition* is one of the most important relations in financial models. It is not our purpose discuss here the actual degree of rationality in real financial markets and the actual chances to realize an arbitrage strategy. Indeed we are simply interested in construct a series of arbitrageurs in our ABM market and verify what can be their effectiveness in keep aligned the simulated prices to the real ones in our sample.

7.1 Real data sample for intra-day stock prices

If we want to introduce arbitrage operations we need real data of intra-day stock prices, since we desire a simulated market aligned to the real one within avery single trading day. The data sample used can be downloaded editing *http://www.google.com/finance/getprices?i=60&p=13d&f=d,o,h,l,c,v&df=cpct&q=STOCK-ID*, for all four stocks[24]. Here you can note that the time frequency is of 60 seconds, i.e. *we consider the opening price of every window of one minute*. The number of trading days covered is 13, from 05-19-2015 to 06-5-2015: this correspond to three weeks of trading and is a very good horizon to test the feature of the model with the intervention of arbitrageurs.

7.2 The arbitrageurs in the program

It is very interesting to decide what is the place of arbitrageurs in the program, considering SLAPP structure. In an ABM which exploits a real trading systems arbitrage operations are not so trivial. If we were just interested in making arbitrage with respect to closing prices for a trading day the question would be quite simple. The arbitrageur(s) would look at the real price expected for that trading day and would operate buying or selling shares on the auction in response to the difference between the two prices, so reducing their gap. But in our work we desire to keep stock prices aligned within a single trading day, according to the time frequency of one minute as in our real data. This is really more challenging because trading frequency in the simulation is then very essential and many complex features of the double auction system reveal as more important.

[24] http://www.google.com/finance/getprices?i=60&p=13d&f=d,o,h,l,c,v&df=cpct&q=AAPL
http://www.google.com/finance/getprices?i=60&p=13d&f=d,o,h,l,c,v&df=cpct&q=GOOG
http://www.google.com/finance/getprices?i=60&p=13d&f=d,o,h,l,c,v&df=cpct&q=FB
http://www.google.com/finance/getprices?i=60&p=13d&f=d,o,h,l,c,v&df=cpct&q=IBM

The main idea, however, is that a single arbitrageur looks at the time at which she wants to operate and compare two prices: from one hand she observes the last daily price in the auction and from the other she considers the real external price which corresponds to the same trading minute of her live intervention. If the real price is higher of that internal to the simulation the arbitrageur makes a buying order on the auction, choosing a bid price comprise between the two misaligned prices; if the opposite situation occurs the arbitrageurs will make a selling order with an ask price comprise once again between the two different prices. In both case, if the operation is then executed, the effect is to reduce the gap between real and simulated price, so (partially) guaranteeing the alignment.

7.2.1 The problem of time frequency for arbitrage intervention

We could conceive the arbitrageurs as true investors in the market, i.e. instances of *Agent* class which are included in the file *investors.txt*: in this sense we would have exactly the arbitrageur as a third kind of investors beside those rational and random. This would be not difficult to realize in the program because we could easily adjust all the methods regarding the investors, just allowing the arbitrageurs to operate through method *trading_ day*.

However a choice of this kind present a series of problems related to the management system for the continuos time inside the auction. Indeed we remind that time within a single trading day flows thank to the arrival of a new investor in the auction. When a trader send her envelope in the electronic book, i.e. when she uses the methods of *Book* class to buy or sell stocks, the time counter is updated with a certain stochastic degree of freedom. If we build the arbitrageurs as true new investors in the market, their actions will be alternated with respect to those of the other investors. This is a consequence of the fact that SLAPP works on a shuffled list of agents, selecting for example an actual investor (random or rational) and then an arbitrageur who is the subsequent trader; of course we can also also observe either a series of consecutive arbitrageurs who operate in the auction or more consecutive random/rational traders who are sending their orders. Therefore the order in which SLAPP has shuffled the list of all investors can determine a situation in which there are time windows (remind that each of them corresponds to a minute) without the intervention of an arbitrageur in the auction. The effect is that in such time window arbitrage operations are neglected and the stock price move away from the real corresponding behavior: if the number of minutes not covered by the arbitrageurs is significant the alignment can no more be maintained and stock prices drift away from the real behavior.

This problem does not simply depend on how SLAPP shuffles the list of investors and arbitrageurs, but also on time partition within a trading day. Looking back at Chapter 4 we have seen that the frequency at which every investor interacts in the auction (i.e the difference in time counter between two consecutive interventions) is related to the number of total investors in the market: more there are the traders (usual investors and arbitrageurs) included and less wide is the time frequency of intervention, such to guarantee that the simulated time does not exceed the limit horizon. Thanks to this system we are sure that all trading operations occur in a window of time which has the corresponding real price in live time: arbitrage operations are then always guaranteed. Though this system for continuos time management is very effective is intrinsic weakness is now evident if we think about the intervention of arbitrageurs. If we want to achieve a mean frequency of trading equal to one minute we should operate in a market with almost 390 investors (consider random and rational traders plus arbitrageurs); in other words one trader per each minute represented in the real daily prices list. Even in this framework, which is yet very heavy in terms of computational time per single simulation, there is the concrete possibility that no one arbitrageur can act with a certain minute. The solution would be increase both the number of actual investors and of the arbitrageurs, for example doubling their amount to almost 800, with a frequency of two traders per minute of time. However also this

choice, which would produce anyway too long simulations, entails a non zero probability for the absence of single arbitrageur intervention in time windows of one minute. In an ideal framework in which there would be so many traders and arbitrageurs the latter probability would become zero and there would be the certainty of having at least one arbitrageur with every trading minute: clearly this scenario cannot be reached for the limits in terms of computational power and time and so we have found a concrete reason to work with arbitrageurs as true investors.

Another obstacle to true arbitrageurs in the ABM is due to the complexity of double auction system. Even if we aways had a single arbitrageur every trading minute, her action remains limited and can have a lagged effect. Indeed, first of all, if the arbitrageurs succeeds to complete her trading operation she will moved just partially the simulated price near the real external one. Second her trading operation can be executed not immediately (the existence of a counterpart is not always guaranteed) and this will generate a (partial) alignment with some delay. The partial alignment and the time delay are two fundamental factors that show us that a single or few arbitrageurs per each time window are not enough to guarantee a constant link between simulated and real market.

So finally the conclusion is that if we consider arbitrageurs as true investors who operate on the same level of random and rational traders, we cannot achieve a global and strong alignment with real data. The solution is to put arbitrage operations on an inner time scale, so giving another dimension in the program to all arbitrageurs.

7.2.2 Arbitrageurs as shadow agents

If the arbitrageurs stop to be true investors, i.e actual instances of *Agent* class, what can be their representation in the program? As we have explained the problem is that we would want a higher frequency of intervention for arbitrageurs and this is almost impossible if we continue to consider them on the same plan of other investors in the model. So what we can do is to place arbitrageurs on a denser trading scale with respect to all the other traders. Their actions are included once again in method *trading_ day* but are indirectly launched by those of rational or random investors, since this time an arbitrageur is no more an *Agent* instance and cannot call any more method *trading_ day. So the arbitrageurs become a sort of shadow agents because they exist only in response to trading operations of actual investors.*

The scheme for this framework presents two kind of arbitrageurs on the basis of their time of intervention: those who operate, in series, after the intervention of a random or trader investor but without moving the time counter (so immediately after the last operation), are called *infra arbitrageurs*; those who operate in a range of time comprise between two consecutive random or rational traders are instead the *step arbitrageurs.* The picture below can photograph which is the actual time plan of intervention for all kinds of investors in the auction.

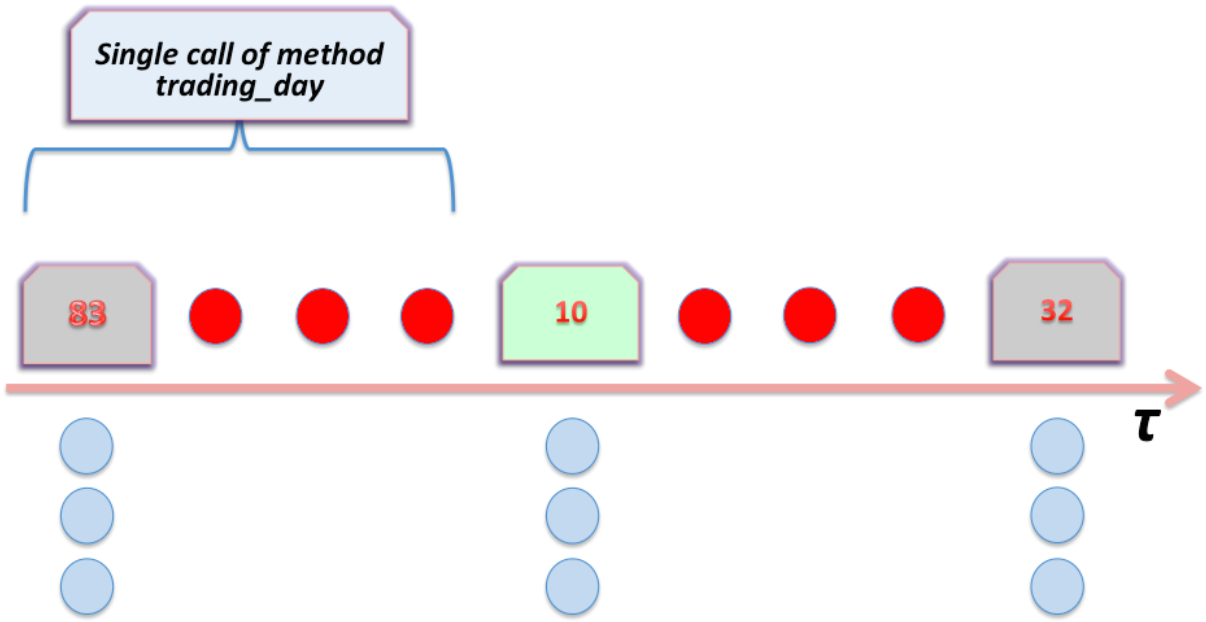


Figure 22: A sketch of the basic time plan of intervention for the arbitrageurs at each trading day. The blue circles are the *infra arbitrageurs*, while the red circles are the *step arbitrageurs*. Only the *step arbitrageurs* move time τ within a trading day.

When a true investor calls method *trading_ day* and enters in the auction, the arbitrageurs are activated by her action operating inside the *scope* of the method just called. Immediately after the order sent by a random or rational investor, the *infra arbitrageurs* enter in the auction and look at the current time yet set by the last true investor; they check in which time window (i.e. in which minute) are operating and make a purchasing or selling order in order to approach simulated and real price. *What is really relevant to note is that infra arbitrageurs does not move time counter, but simply operate using the last temporal value contained in the book.* Using more *infra arbitrageurs* helps to guarantee a better alignment baacuse, as we have discussed above, the presence of a single arbitrageur is not enough.

After the first series of *infra arbitrageurs* (i.e. those who operate immediately after the true investor) time counter is moved by the *step arbitrageur*. Indeed we had required that arbitrageurs can operate on a denser time scale with respect to rational or random investors. Every *step arbitrageur* moves the time counter according the scheme adopted and illustrated in Chapter 4. However, since we have now many more agents who act on different instants of time, we need to define again time partition in the following way

$$\overline{\Delta\tau}_{i,t} = \frac{l_{i,t}}{M \cdot n_{arbStep}}$$

where $n_{arbStep}$ is the chosen number of *step arbitrageurs* acting within a single call of method *trading_ day*: for each random or rational investors who send her purchasing or selling order there are $n_{arbStep}$ who operate before the new true investor, so updating the time counter. The introduction of *step arbitrageurs* implies

a denser time partition, but without any change to actual trading frequency of true investors: the number of rational and random traders is not modified and the average temporal distance between two consecutive true investors remain the same. The role of *step arbitrageurs* is therefore to provide a denser mapping of time flow to perform arbitrage operations at a higher frequency.

Such structure is effective because we have found the way to solve the problem of partial alignment and of too low arbitrage frequency, although the parameters $n_{arbStep}$ and $n_{arbInfra}$ have to be calibrated to obtain the desired result of actual and constant alignment to real data. But is this investment plan sufficient to guarantee total alignment?

7.2.3 Adding ad hoc counterparts for the arbitrageurs

In order to test the effect of arbitrage operations with the scheme previously conceived, we vary $n_{arbStep}$ and $n_{arbInfra}$ such to observe, day by day, if the simulated market (i.e. all the security prices) remained aligned to the real one. Also the order size of the arbitrageurs can be an interesting parameter because is a measure of the effectiveness of their intervention. *If the arbitrage is robust it could be valid for any seed and for all the* $T = 13$ *trading days of the real sample.*

Running several simulations with different parameters settings we realize there is no combination which guarantees the alignment over the entire horizon of $T = 13$ trading days. The typical situation is the following

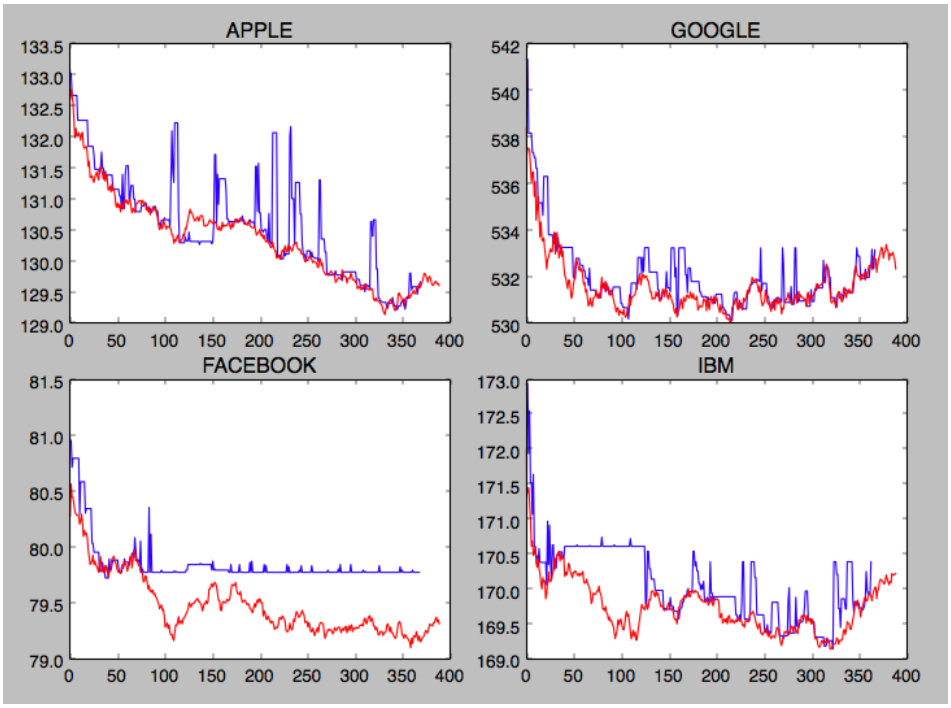


Figure 23: When arbitrage operations fail the reason is due to lack of counterparts for the arbitrageurs: so stock prices deviate from real data and remain in a quite steady state.

Why arbitrage does not work even with our intervention plan? The reason is due to one of the elements previously discussed, i.e time delay in arbitrage operations. Though there are many *step arbitrageurs* who maintain a very high arbitrage frequency and many *infra arbitrageurs* who reinforce the alignment, the lack of actual counterparts in the market may frustrate their work. *Indeed if an arbitrageur does not find an immediate counterpart in the book, her order remains suspended and her action reveals worthless: since an arbitrage operation is related to real time price values, it will be effective only if the order is immediately executed in the book.*

The latter is, as continuously repeated, an effect which stems both from the complexity of double auction trading system and from the limited number of traders (in relation to the total number of trading minutes). What can be the solution to such issue? Every time an *infra* or *step arbitrageur* makes her order there is another fictitious trader who operate as her natural counterpart, making a bid/ask price which matches with the ask/bid price made by the arbitrageur. *So we are defining ad hoc counterparts for each arbitrageur because in this way we are sure that their order is immediately executed: the new stock price will be that proposed by the arbitrageur (because the counterpart enters in the auction after her) and so the simulated price and the real one will always get close to each other.*

There could occur a situation in which the *ad hoc* counterpart is worthless. When the first *infra arbitrageur* makes her order after the true investor it can be possible that the latter becomes the counterpart of the arbitrageur. If this happens what does matter is the order size of both traders: if that of the real investor is greater than that of the arbitrageur, the latter order is cancelled. The arbitrageur does no more appear in the auction and the intervention of her *ad hoc* counterpart is no more necessary: it can even reveal as a damn to alignment because we are introducing an agent who operate in the opposite direction with respect to tha arbitregeur. If we neglect such element we could find that after some trading days the alignment collapses. In order to avoid such situation we use the *ad hoc* counterpart *only* when the order of her corresponding arbitrageur is yet in the book. A final obstacle is represented by a true investor with a very large order size, who can become the counterpart of many consecutive *infra arbitrageurs*: in this case the *ad hoc* counterparts will not be introduced and the stock price will be continuously set to that proposed by the investor. The consequence is once again a disalignment between simulated and real market. The trick is to assume that the first *infra arbitrageur* will fix her order size equal to that of the true investor who has come before her. So if there is a matching both orders are removed by the book, there is no need of the *ad hoc* counterpart and a new *infra arbitrageur* can operate without any problem. All these modifications ensures that the time plan of intervention previously defined is actually efficient for the entire simulation period. Here we show the final sketch with the counterparts.

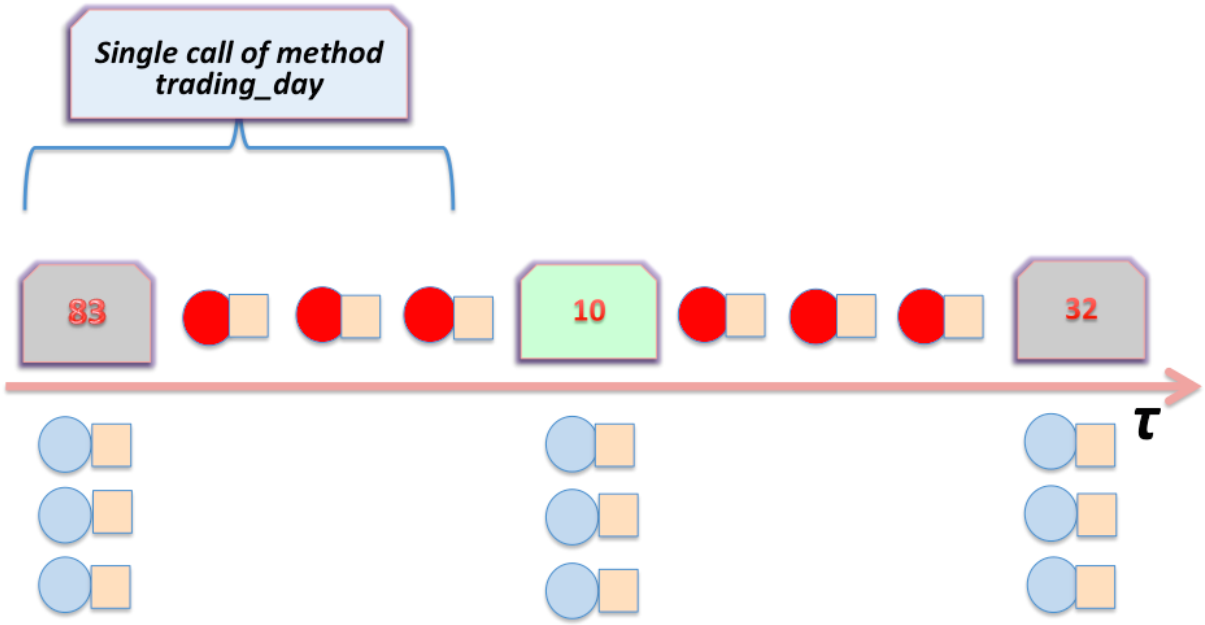


Figure 24: A sketch of the final time plan of intervention for the arbitrageurs at each trading day, with the introduction of *ad hoc* counterparts; the latter are represented by the orange squares.

It is the moment to test the effect of the arbitrageurs on the market, having adopted the new modifications in the time plan of intervention. We simply show the final result in the case of $n_{infra} = 10$ and $n_{step} = 10$.

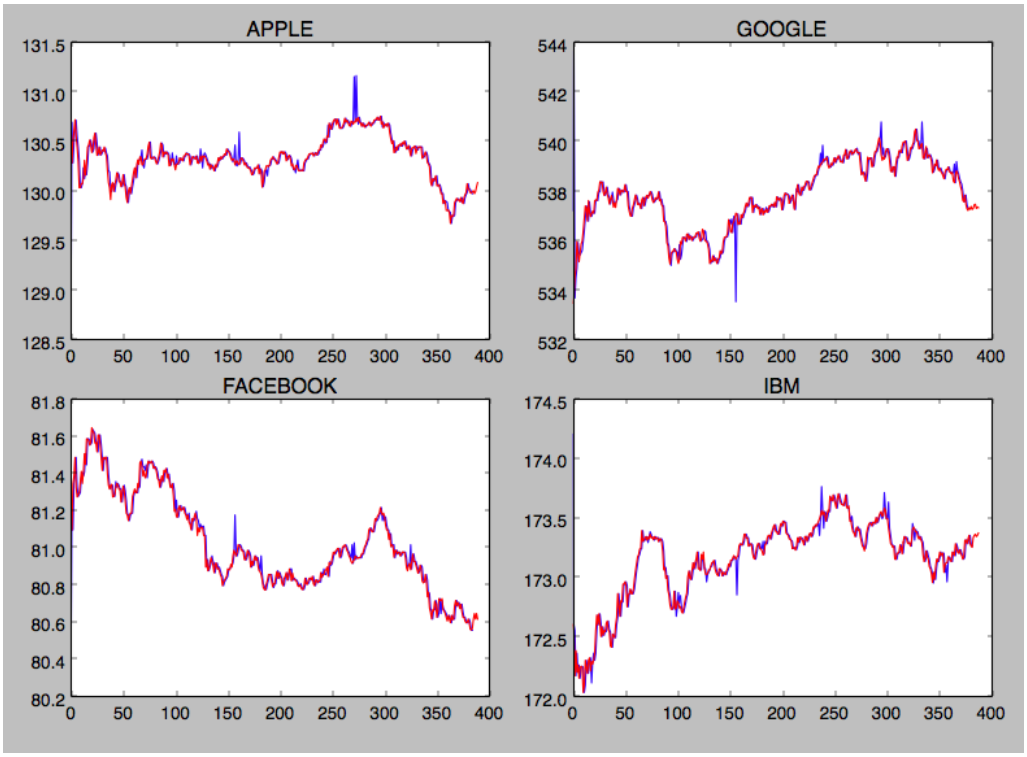


Figure 25: Plot of daily prices applying the final intrevention plan of the arbitrageurs: the alignment is really strong with $n_{infra} = 10$ and $n_{step} = 10$.

The simulated market is really aligned to the real one with the new scheme of intervention for the arbitrageurs. However n_{infra} and n_{step} both play a fundamental role and if we reduce their values the alignment becomes more weak, though the two price series remain close to each other. For the rest of the work, when we speak about arbitrage scenario we are always adopting the configuration $n_{infra} = 10$ and $n_{step} = 10$, which guarantees a very stable condition for arbitrage intervention. Indeed switching seed and running again the simulation the alignment is not altered and so we have build a robust scheme to operate in a stock market which is actually the same of the real one.

All the rules and tricks adopted to construct an effective intervention plan for the arbitrageurs can seem quite unrealistic. Anyway this does not matter because the idea is simply to ensure that random or rational investors can trade in a market in which stock prices are constinuously kept aligned to real ones. The arbitrageurs remain artificial agents who operate only for our purposes but do not own any relevant feature. Even their existence in the book does not influence true investors: this is for example the case of herd transitions. In the arbitrage framework rational investors, when are looking at the current size or length f the opposite side of the book, does not consider nor the arbitrageurs neither their *ad hoc* counterparts. Herd effect is just moved by other real investors, while arbitrageurs and their *ad hoc* counterparts simply operate in the auction to guarantee the alignment.

8 Simulation data analysis

In Chapter 6 we have studied the role of control parameters to investigate a series of interesting learning or irrational phenomena in the field of behavioral finance. Now, on the contrary, it is time to analyze some properties which are typical of quantitative finance, in order to compare them to the theoretical benchmark of the EMH and to real stock market features. In addition we can explore the performance of the agents in the market and match the results with some local and global properties of the informational network. The analysis can be divided in three main blocks which entail different problems and so require a different approach for the illustration of quantitative results.

The first part is dedicated to statistical properties of stock returns in the simulated market: we look in particular at skewness, kurtosis and autocorrelations of log returns. Here the simulation horizon is generally high because we desire a large enough sample (formed by daily returns) to get reliable statistical estimators. In this part of the analysis we are trying to discuss the EMH and reject its strong form in our simulated stock market. There is no need to include arbitrage operations because the latter could destroy the natural statistical market features emerging from the simulation.

The second part aims to presents the most important variables on the micro scale, i.e. which belong to the agents in the market. *In this case arbitrage is essential since the important question is if many properties of the agents operating as part of the simulated market are preserved with the alignment to real data. The arbitrage scenario offers a realistic application of our ABM to a real stock market. What can really matters is not a comparison regarding market properties as stock prices and returns, but a debate about the performance and behavior of agents in both scenarios.* So the simulation horizon has to now to match with that of the real sample used for arbitraging.

The final part of the analysis is instead always focused on the general properties of the informational network: arbitrage operations are not considered here because we are simply studying general features of the network which are preserved also with the alignment. The idea is to illustrate many global network measures but also have a look to a very interesting parallelism between profits and degree of investors in the market.

We shall run the simulation at different seeds taking a final average for all the desired estimators, even when arbitrage is allowed. Indeed, though the market is kept aligned to the real one independnetly from the seed, the series of pseudorandom numbers, which will define the specific trading strategies of the agents, will vary with the seed. Therefore many simulations are required to catch the true statistical properties.

Note that we can associate an error to all mean values over more simulations: the latter is the standard deviation of the sample mean, i.e. the standard deviation of all the measures divided by the square root of the number of measures. This is a basic formula of statistical analysis applied to the context of ABMs and it reveals very useful to evaluate the reliability of our quantitative results through the implementation of simple hypothesis tests.

All the analysis performed here is relative to the scenario of no transitions for the ABM. The choice to avoid transitions of any kind is necessary to define an effective benchmark to data analysis: if some control parameters and behavioral phenomena would affect our results it would be very complicated to test the basic properties of this ABM. Here there is the list of the configuration used for control parameters for the rest of the chapter; we shall not show any more such parameters setting.

M	n_{rand}	n_{consul}	$\triangle n_{trade}$	ϵ_{test}	ϕ_{test}	λ_{test}	p_{info}	x_{change}
200	0.9	0.5	0.001	10^7	100	10	50	10

Table 9: Parameters setting used for the rest of the analysis: no transitions are allowed and there is no update in rational strategies.

Also remind that when we allow arbitrage operations, the configuration used is $n_{infra} = 10$ and $n_{step} = 10$.

8.1 Log returns distribution

The first topic to analyze is the distribution of returns in the simulated market. In particular, by construction due to our portfolio choice framework and also to simplify some statistical measure, what we shall consider is the distribution of log returns, which we remind are defined as follow

$$r_t = log(1 + R_t) = log(1 + \frac{P_t - P_{t-1}}{P_{t-1}}) = log(\frac{P_t}{P_{t-1}}) = p_t - p_{t-1}$$

where we have adopted the common convention of lower case letters for the logarithmic variables. In this formula t is the index which defines a particular trading day because what we actually study is the distribution of log returns at the end of every trading day, i.e. tick by tick in the simulation. The distribution of daily returns would be too specific for a single trading day, but what is really important is to investigate market properties in terms of stock returns on the horizon of many trading days. This kind of analysis then drifts away from the idea of a very short simulation horizon because we need a wide enough data sample. The method *LogReturnsDist* will display the four histograms of log returns distribution for all the stocks in the market and it will also provide the four sample moments of such distribution as useful indicators of the sample.

The possible comparison with the aligned market through arbitrage operations is, in this case, not what we want for the analysis. If we align the market to the real one, returns distribution will result very similar to that one yet available from historical data and so we would not get any new information. So here we limit our discussion to the free simulation framework and we run several simulations to get the averages over more seeds for all the statistical indicators. At every simulation we compute four statistical indicators for the log returns distribution: the sample mean, the sample standard deviation, the sample skewness and the sample kurtosis, all computed using the Python library *Scipy*[25]. Remember from statistical analysis that these indicators are just *estimators* of theoretical moments of the distribution and so they have intrinsically their error because they can be seen as random variables. Indeed every simulation we run can be always seen as a single experiment which produce a random sample for log returns. For our purposes it is however uninteresting to consider the error on a single value of the estimator, which is the result of a single simulation. Instead we need to run more simulations (i.e. with different seeds) and get every time the final value of the estimators. At the end we can compute the mean estimator over all the seeds and use as its error the

[25] See http://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.skew.html and http://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.skew.html and https://docs.scipy.org/doc/scipy-0.7.x/reference/generated/scipy.stats.kurtosis.html to verify which are the formulas used to compute all the estimators

standard deviation of the mean. This process helps in reducing possible statistical fluctuations related to a single seed. We show the results for all four stocks, every time reporting a list of this kind: $\begin{matrix} APPLE \\ GOOGLE \\ FACEBOOK \\ IBM \end{matrix}$.

$seed$	$\overline{r}$	$\hat{std}_r$	$\hat{skew}_r$	$\hat{kurt}_r$
2	0.0068787	0.0200404	5.1767980	32.616677
	0.0072455	0.0016945	4.6084938	27.719475
	0.0031988	0.0088778	0.9850042	2.2359852
	0.0049737	0.0077554	0.1555409	−0.5839528
3	0.0033224	0.0092022	0.365892	0.674538
	0.0079487	0.0203961	4.824523	28.58797
	0.0029747	0.0102330	1.247728	1.760738
	0.0035855	0.0084141	1.620051	5.117114
4	0.0023930	0.0090717	0.833189	2.139300
	0.0053438	0.0092904	0.076307	0.086686
	0.0049468	0.0131666	2.135379	7.409232
	0.0051659	0.0094909	1.136712	2.520838
5	0.016874	0.071815	6.857178	47.64790
	0.019294	0.073567	7.060778	49.82472
	0.009831	0.042295	6.943308	48.68444
	0.003860	0.007459	0.251779	−0.536329
6	0.0058963	0.0218604	5.047748	30.79119
	0.0050790	0.0131442	2.630529	11.78199
	0.0034024	0.0098622	0.009862	1.396571
	0.0141505	0.0720896	7.223621	51.40449
7	0.0082795	0.0119655	0.562833	0.9338712
	0.0082446	0.0141589	1.260453	1.9942652
	0.0020761	0.0092072	1.100538	2.4241624
	0.0051572	0.0149015	3.657671	17.071039
8	0.0074747	0.014892	3.465567	16.69379
	0.0163019	0.041626	5.212617	28.25545
	0.0042015	0.017645	5.143135	29.78239
	0.0066880	0.015675	4.981838	28.00130
9	0.0053892	0.0088155	0.542090	−0.236957
	0.0055201	0.0088039	1.092742	0.9455018
	0.0016287	0.0075519	1.169379	2.5094085
	0.0026144	0.0069089	1.002175	0.6886551
10	0.0052944	0.0172609	3.780343	15.380802
	0.0101440	0.0153822	1.947621	3.4842991
	0.0033508	0.0093966	1.487669	3.2752270
	0.0052671	0.0078494	0.935132	0.9362563
11	0.0070547	0.0095671	1.4660561	2.9137962
	0.0078924	0.0143495	2.6848616	8.4056828
	0.0017129	0.0060912	0.8065554	0.1493118
	0.0080038	0.0156456	2.4457105	6.9595717

Table 10: Table with all the values obtained for the four moment estimators for log returns distribution. The values are reported for each seed and for every stock in the market; the simulation horizon is $T = 50$.

Ten different seeds are sufficient to achieve good values for the estimators of distribution moments because the simulation is running for many trading days and this reduce the importance of stochastic fluctuations due to a single seed. It is important to remind that all such sample estimators have their error, but we decide to neglect them because the latter result so less important if we take an average over all the simulations. Before to proceed to the final analysis let us also show, for one single seed, the histogram of the distribution.

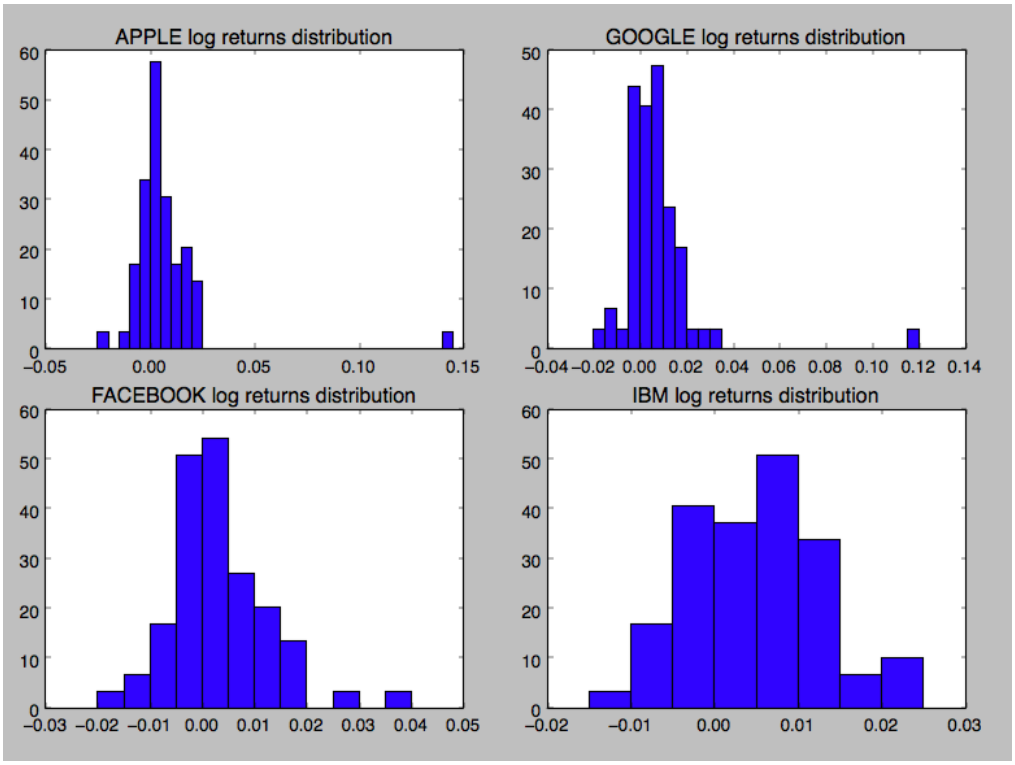


Figure 26: Plot with the four histograms (with normalization) of log returns distribution after $T = 60$, at $seed = 20$.

From these plots we can immediately see how the hypothesis of normality for log returns (log normality for total returns) is totally rejected, as we previously expected although we have used such assumption in the portfolio optimization process. The existence of tails in the distribution and in particular of *fat* tails, i.e. tails which have a significant wight in terms of probability, is what we actually observe in real stock returns distribution. Now taking, for each estimator and each stock, the average over the ten different seeds we get more stable statistical estimates regarding the moments of these distributions. Remind that the standard error here shown is the standard error of the mean, which corresponds to the standard deviation (computed among the ten estimator values at different seeds) divided by the square root of the ten (i.e. the number of considered measures).

SECURITY	$avg(\overline{r})$	$avg(\hat{std}_r)$	$avg(\hat{skew}_r)$	$avg(\hat{kurt}_r)$
APPLE	0.0068854±0.0012501	0.0194491±0.0060123	2.80977±0.74715	14.9555±5.34925
GOOGLE	0.0093014±0.0015148	0.0212413±0.0066928	3.13989±0.69651	16.1086±5.26849
FACEBOOK	0.00352193±0.000838	0.0134326±0.0033619	2.10286±0.690979	9.98275±5.011259
IBM	0.00594661±0.001031	0.0166189±0.0062659	2.34102±0.726014	11.1579±5.32904

Table 11: Table of final average estimators over the five seeds, with their standard errors.

In order to check if the existence of asymmetry and fat tails in all returns distributions is relevant we perform some *Student's t-tests*. In particular we want to check if the skewness and kurtosis of stock returns are significantly different from zero: both these estimators are defined such to be zero for a perfectly symmetric distribution without fat tails (which is just the normal distribution) and therefore this is our *null hypothesis* and we test it for all stocks.

SECURITY	$t(\hat{skew}_r)$	$t(\hat{kurt}_r)$
APPLE	$\frac{2.80977}{0.74715} = 3.76 \Rightarrow reject$	$\frac{14.9555}{5.34925} = 2.79 \Rightarrow reject$
GOOGLE	$\frac{3.13989}{0.69651} = 4.51 \Rightarrow reject$	$\frac{16.1086}{5.26849} = 3.06 \Rightarrow reject$
FACEBOOK	$\frac{2.10286}{0.690979} = 3.04 \Rightarrow reject$	$\frac{9.98275}{5.01259} = 1.99 \Rightarrow reject$
IBM	$\frac{2.34102}{0.726014} = 3.22 \Rightarrow reject$	$\frac{11.1579}{5.32904} = 2.09 \Rightarrow reject$

Table 12: Results for all the *t-tests* regarding skewness and kurtosis. When $t > 1.68$ the *null hypothesis* is rejected and skewness and kurtosis can be considered statistically different from zero.

The hypothesis of zero skewness and zero kurtosis is always aborted for all securities, as we expect in a real market, according to such *t-tests*.

So normality is rejected for stock returns and this is a very important sign of complexity in our simulated stock market. The existence of skewness and kurtosis quite different from zero means that extreme events (i.e extreme negative or positive returns) are not so rare in market dynamics, as we have shown looking at bubbles and crashes. Fatter tails and asymmetry in stock returns distribution are very few times considered in many financial models and this ABM brings out naturally such complex features.

8.2 Autocorrelations in return series

If from one hand is important to look at the sample of stock returns in terms of probability distribution, from the other side we have to analyze what kind of stochastic process can well fit our simulated data. This implies time series analysis and in particular we want to estimate the Auto Correlation Function (ACF). Finding the values of autocorrelation coefficients is essential to discuss the basic framework of the EMH, which we remind that assume, in its strong form, that stock returns would follow a white noise process. So the ideal assumption in an efficient market is that there should not be any correlations between two stock returns at different time.

In this section of data analysis we have, as in the case of returns distribution, no need to consider also the arbitrage scenario: our goal is to investigate the statistical properties of stochastic processes regarding simulated stock returns and the alignment to a real market would destroy such features emerging from the

ABM. We briefly remind the definition of autocorrelation coefficient at lag k

$$\rho(k) = \frac{cov(r_t, r_{t+k})}{\sqrt{var(r_t)}}$$

where r_t is the log return of the stock at time t; the autocorrelation coefficients are always comprised between -1 and 1. Taking the autocorrelation coefficients at different lags we construct the autocorrelation function for a certain stock and this is one of the most interesting tool in the field of technical analysis. Indeed significant auto correlations (either positive or negative) at some lag can imply the possibility to predict future asset prices starting from past observed values: this is just the final aim of technical analysis, which is denied by the EMH in its strong form.

Once again we can exploit the specific method *global_AutoCorrel* in *oActions.py* to have both plots and data about auto correlations in stock return series. Here we show a plot of the four correlograms; we remind that all the plots and the numerical estimates are obtained thanks to R which is called in Python through *Rserve*.

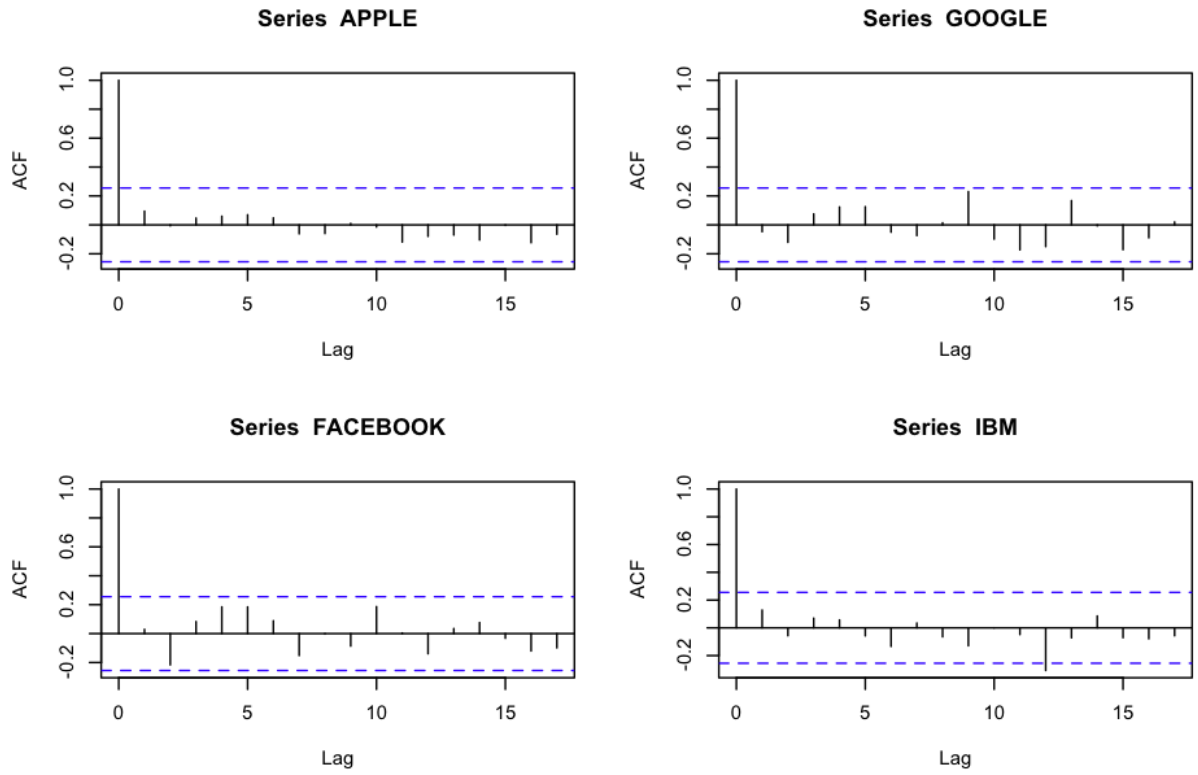


Figure 27: Correlograms of the four stocks at time $T = 50$ and with $seed = 2$.

The correlograms are very effective to see the autocorrelation function of all stocks. Remind however that the these are simply sample estimates of autocorrelation terms and so a correlogram does not perfectly

coincide with the theoretical autocorrelation function. As required by our protocol of analysis, we have to run several simulations and report every time the autocorrelation coefficient at the first three lags, for each stock.

seed	*first lag coefficient*	*second lag coefficient*	*third lag coefficient*
2	0.0947653	−0.00654034	0.04699794
	−0.0471224	−0.12071084	0.07589424
	0.0293465	−0.21650041	0.08316923
	0.1281105	−0.05680620	0.0696779
3	0.2316339	0.01220066	−0.02761343
	0.0350706	−0.10232072	0.00376836
	−0.07976039	0.04995115	0.15889485
	0.10406792	−0.04028296	0.00968335
4	−0.00198622	0.10214642	−0.17533732
	−0.01433347	−0.05193951	−0.13133695
	−0.11208756	−0.0015034	0.0400106
	0.03506098	−0.10913646	−0.00619845
5	0.01033549	−0.01629427	−0.00134489
	−0.01639105	−0.02490294	0.0286404
	−0.04426887	0.02132076	−0.0236143
	0.11249018	−0.12285452	−0.0041835
6	0.13162423	−0.01341794	−0.09331024
	0.23643289	−0.0176305	−0.0519811
	0.13085654	0.05384506	−0.073712
	0.01720319	−0.02238405	−0.04229209
7	0.06517722	0.01715156	0.06352964
	0.12520284	−0.11824404	−0.01809475
	0.08006286	0.0337679	0.05465449
	0.00047235	−0.23974658	−0.056979647
8	0.11362271	−0.0116292	0.16997919
	−0.05455658	−0.06309181	0.30897942
	−0.04478641	−0.07589929	−0.07569153
	0.07297459	−0.06547189	0.03977377
9	−0.00472509	0.20890658	0.10675701
	−0.09833715	0.14931471	−0.16209287
	0.14893342	−0.1092886	−0.02132928
	0.0550758	0.12776553	0.0133733
10	0.20403492	0.00651641	0.03303922
	−0.0577929	0.05376497	0.20742237
	0.10324745	0.17663907	0.11959451
	0.17267242	0.12447321	0.22539648
11	0.08763367	−0.15839161	0.14830163
	0.30311953	−0.03481737	0.08074693
	0.2279538	−0.09383619	−0.18655126
	−0.14015884	−0.11441677	−0.12371397

Table 13: Table of autocorrelation terms at the first three lags for all securities; the simulation period is $T = 50$.

As estimators the autocorrelation coefficients above computed are intrinsically affected by significant

errors; the latter are not reported here because we want instead to take the average, over the ten simulations, of every autocorrelation term. The results are shown here, with their standard errors, for all securities.

SECURITY	*avg first lag coefficient*	*avg second lag coefficient*	*avg third lag coefficient*
APPLE	0.0932116±0.0257503	0.0140648±0.0294947	0.0270999±0.0337458
GOOGLE	0.0411292±0.0430055	-0.0330578±0.0262681	0.0341946±0.0455176
FACEBOOK	0.0439497±0.0353984	-0.0161504±0.0347957	0.00754253±0.0329078
IBM	0.0557969±0.0275092	-0.0518861±0.0352935	0.0124537±0.0291278

Table 14: Table of average lag coefficients in auto correlations; the simulation period is $T = 50$ and the standard error is always reported.

Now you could use the average estimators for autocorrelation coefficients to test the hypothesis of no correlations in stock returns which is typical of the EMH in its strong form: *stock prices should follow a random process and then returns should not exhibit serial correlations.* However in this case the intrinsic error on every single estimator has a significant effect because leads to a very volatile series of estimators using more seeds. If we performed the *t-test* on the lag coefficients we would have to accept the hypothesis of zero autocorrelations. Before to actually assert that this is true we follow another more efficient approach which does not depend so significantly by stastical errors.

A more elegant and common test used in quantitative finance to check predictability is computing the *variance ratio*[26]. This variable considers the ratio between a composed return and its corresponding single period value; in the case of log returns the composed return on k periods is the sum of the k single period log returns. If there are no autocorrelations in the process, the variance of the composed return is equal to k times the variance of single period return; otherwise autocorrelation do appear and we have some degree of predictability. The variance ratio test then exploits the linear growth of the variance in the following way

$$VR(k) = \frac{var(\sum_{i=1}^{k} r_{t+i})}{k \cdot var(r_t)} = 1 + 2\sum_{i=1}^{k-1}(1 - \frac{i}{k})\rho(i)$$

where the second expression is obtained assuming that the process is stationary (in weak sense), i.e. the variance does not depend on time t. There are three main situations to consider:

- $VR(k) = 1$: there is no autocorrelations in stock returns;
- $VR(k) > 1$: positive autocorrelations in stcok returns;
- $VR(k) < 1$: negative autocorrelations in stock returns.

So the variance ratio is a very useful indicator to check the nature of aucorrelations in returns series.

Inside method *AutoCorrel* it is very easy to define the variance ratio as a function of k. We consider only $k = 2$ and $k = 3$ which are yet sufficient to test predictability, though the complete spectrum of *VR(k)* can be shown without difficulties. Here there are the results with ten different seeds, for all securities

[26] You can find a more complete introduction to some tests regarding market predictability in Chapter in *Financial Modeling of the Equity Market: From CAPM to cointegration* (2006).

seed	$VR(2)$	$VR(3)$
2	0.8302369	0.9814253
	0.4743831	0.4510684
	1.2004829	0.9982726
	0.9650108	0.5685957
3	1.1620379	1.131031
	0.8123939	0.725870
	0.6466633	0.637535
	1.1033076	0.873491
4	0.681806	0.756734
	0.304645	0.268595
	0.941589	1.046345
	1.148908	1.254821
5	0.774371	0.555863
	0.692866	0.208922
	0.681724	0.456029
	0.445842	0.254965
6	0.779537	0.402981
	1.4615996	1.762025
	1.4559741	1.610671
	1.4008418	1.696298
7	1.1331434	0.96665
	1.3800757	1.27021
	1.2742773	1.27738
	0.9124381	0.79704
8	0.892383	0.684542
	0.911134	0.820895
	0.550284	0.596662
	0.990740	0.977290
9	0.674872	0.373465
	0.573558	0.378069
	0.904878	0.653071
	0.559359	0.421709
10	1.05556	0.84035
	0.87124	0.85868
	0.99737	1.09967
	0.91966	0.87504
11	1.05735	0.81773
	0.51405	0.399427
	1.24994	1.353956
	1.01059	0.501363

Table 15: Table of variance ratio tests for all securities; the simulation period is $T = 10$.

And now let us compute again the average over all the ten simulations, reporting the standard errors.

SECURITY	*avg VR(2)*	*avg VR(3)*
APPLE	0.90413±0.05824	0.751077±0.0790667
GOOGLE	0.79960±0.11962	0.714376±0.155444
FACEBOOK	0.99032±0.09578	0.972959±0.119733
IBM	0.94567±0.08697	0.822061±0.134607

Table 16: Table of average variance ratios for $k = 2$ and $k = 3$; the simulation period is $T = 10$ and the standard error is always reported.

Next step is to execute a *t-test* for every asset and for both $k = 2$ and $k = 3$. The *null hypothesis* here corresponds to $VR(k) = 1$, i.e. to zero autocorrelations in stock returns.

SECURITY	$t(VR(2))$	$t(VR(3))$
APPLE	$\frac{0.90413}{0.05824} = 15.52 \Rightarrow reject$	$\frac{0.751077}{0.0790667} = 9.5 \Rightarrow reject$
GOOGLE	$\frac{0.79960}{0.11962} = 9.50 \Rightarrow reject$	$\frac{0.714376}{0.155444} = 4.83 \Rightarrow reject$
FACEBOOK	$\frac{0.99032}{0.09578} = 10.34 \Rightarrow reject$	$\frac{0.972959}{0.119733} = 8.13 \Rightarrow reject$
IBM	$\frac{0.94567}{0.08697} = 10.87 \Rightarrow reject$	$\frac{0.822061}{0.134607} = 6.11 \Rightarrow reject$

Table 17: Results for all the *t-tests* regarding the variance ratios. When $t > 1.68$ the *null hypothesis* is rejected and

All the tests reveal that autocorrelations are always different from zero. Such property is typically observed in real stock returns series and this supports once again the great realism of our ABM. The very relevant consequence is anyway that in this simulated market there is the possibility to predict stock prices. Can consultants actually exploit such autocorrelations to build an efficient optimal portfolio? And can rational traders actually use such portfolio to realize significant profits?

8.3 Mean profits distribution

After having analyzed statistical properties of stock returns emerging from the simulation, we need to consider the variables which characterize the agents in the market. One of the most important parts of the analysis for an ABM stock market concerns the profits realized by investors, reminding that such variables takes into account both market gains and cash flows deriving by executed trades. What can be actually interesting to consider are the mean daily profits for every investor. Daily profits are too volatile variables since reflect the particular trend of one trading day, but taking a time average over the simulation period (i.e. mediating over many trading days), ensures a more stable and realistic indicator. So, for an investor j, these are the mean profits over a simulation horizon T

$$\overline{\pi}^j = \frac{1}{T}\sum_{t=1}^{T} \pi_t^j$$

The possibility of obtain persistent profits implies significant positive mean profits and, if we check the latter is true for some investors, the great question is what kind of investors have managed to beat the market.

As significant auto correlations in stock returns implies a dispute with the EMH in its strong form, also the existence of traders who can persistently out perform (i.e. overcome the mean performance of all the other investors) is a sign that the market is not perfectly efficient. Which is the level of predictability for stock prices in a complex market based upon a real trading system? Do exist some trading strategies, based upon specific forecasting models, that guarantees higher profits? This work tries to solve such questions and our approach is quite simple.

Thanks to method *DailyProfits* we are able to see from the terminal which are the *outliers* in the mean profits distribution., i.e. which are the investors who have realized profits very higher or very lower than the average among all traders. From a mathematical point of view we decide to define *outlier* investors, indexed with j_{out} when they have outperformed and with j_n when they have underperformed, as those having realized the following mean profits

$$\overline{\pi}^{j_{out}} > \underbrace{\frac{1}{M}\sum_{t=1}^{M}\overline{\pi}^{j}}_{\hat{mean}(\overline{\pi})} + 2\hat{std}(\overline{\pi})$$

$$\overline{\pi}^{j_{in}} < \underbrace{\frac{1}{M}\sum_{t=1}^{M}\overline{\pi}^{j}}_{\hat{mean}(\overline{\pi})} - 2\hat{std}(\overline{\pi})$$

So the definition of outperforming, or also *beating the market*, is build through the sample mean and the sample standard deviation of the mean profits for all investors; the same is for underperformances. The choice to set two standard deviations as a threshold to discriminate outperforming or underperforming traders is just a benchmark for our analysis. In standard statistics all data beyond three standard deviations (from both sides of the distribution) are considered as statistical errors to remove from the sample itself. In our case we want instead to maintain the *outlier* investors within the sample because they have to contribute to alter the moments of the distribution of mean profits. Therefore two standard deviations allow to fix limit an effective but not too restrictive limit for *outliers* search.

In all such considerations the idea is to compare the totally free simulation scenario to the arbitrage framework. *If we operate with the simulated market aligned to the real one do significant profits yet appear? In other terms, are there any chance to beat a market which is very close to the real one?*

First let us plot the distributions of mean profits with one of the seeds used in the analysis, in both scenarios. Remind that, since we are interested in comparison with the aligned market, the simulation period has to coincide with the number of trading days covered by our real sample, which are 13.

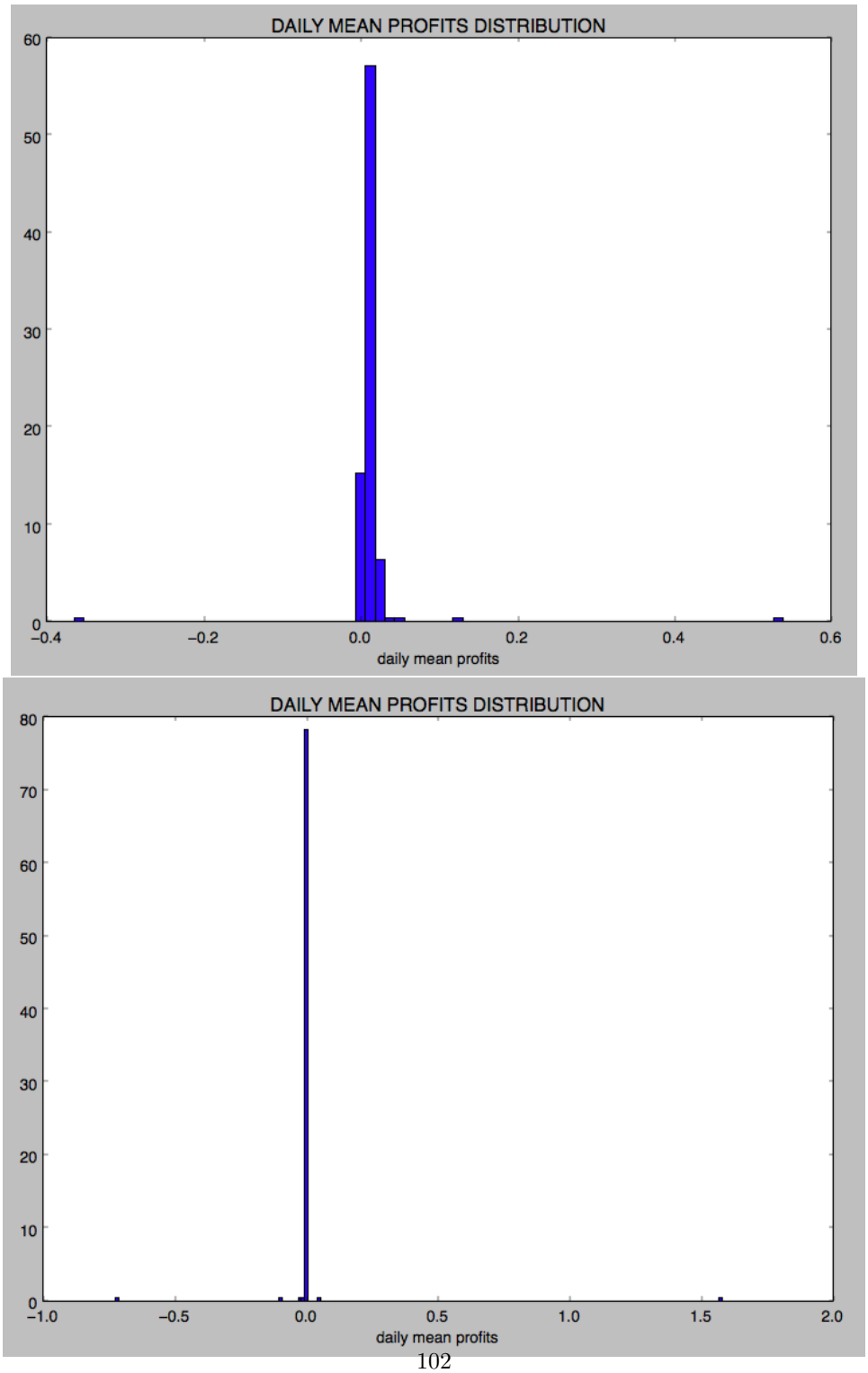


Figure 28: Plots of mean profits in the market (with normalization) with a simulation period $T = 13$ and *seed* $= 2$; the *outliers* are represented by the extreme bins in the histogram. On the left we have the no arbitrage scenario; on the right there is the histogram in the case of arbitrageurs operating on the market.

What is evident from such mean profits distribution is the existence of the *outliers* who have outperformed or underperformed during the simulation. The most part of investors have realized profits which are close to zero as we could expect in an efficient market. The situation is similar when the arbitrageurs operate on the auction, but the differences can be easily seen without any numerical analysis: the distribution is more localized at zero mean profits, but the *outliers* manage to realize better mean profits. *So in a market which replicates the real one, there are less chances to beat the market, but those who manage to do it will realize better mean profits with respect to the simulated market. This is a very interesting and not trivial result. In the simulated stock market there are more opportunties to get positive (or negative) mean profits because the market is less efficient. When we introduce the alignment with real stock prices, the fluctuations in mean profits distribution tend to decrease (you have to not consider the outliers if you want to check that), while the mean profits for the outliers become more extreme. Reducing the number of traders with no zero mean profits, the potential profits are more consistent if you are able to beat the market: this is a sort of competition mechanism which characterize such complex system.*

Now we run several simulations with different seeds and report every time the sample mean and sample standard deviation of mean profits; in addition we include which is the number of outperforming and underperforming *outliers* (n_{out} and n_{under}), specifying which is their consultant if they are rational traders. Indeed if some *outliers* exist it is so important to understand which is the trading strategy adopted during the simulation. *In particular are the outliers always rational traders or not? And if so, which forecasting method have used (indirectly through their consultant) to outperform or underperform?* Two tables summarize the results both in the free simulation scenario and in the aligned market. We specify both n_{out} and n_{under}, including the corresponding consultant if the *outliers* are rational.

seed	$\hat{mean}(\overline{\pi})$	$\hat{std}(\overline{\pi})$	n_{out}	n_{under}
2	0.00368	0.030279	2(standard) - 1(AR(3))	1(standard)
3	0.00442	0.00358	1(standard) - 1(AR(3))	2(standard)
4	-0.02740	0.27723	1(standard)	0
5	0.00567	0.00878	1(standard) - 1(AR(1)) - 1(AR(3))	2(standard) - 1(AR(3))
6	0.00628	0.05282	1(standard) - 1(AR(3))	2(standard) - 1(AR(3))
7	0.00242	0.058205	1(standard)	1(standard) - 1(AR(1))
8	0.00909	0.04451	1(AR(3))	1(standard)
9	-0.00090	0.26899	1(standard)	2(standard)
10	0.03932	1.71871	1(AR(3))	1(standard)
11	0.00411	0.01959	1(AR(3))	2(standard)

seed	$\hat{mean}(\overline{\pi})$	$\hat{std}(\overline{\pi})$	n_{out}	n_{under}
2	0.003518	0.122382	1(standard)	1(standard)
3	-0.336355	4.000818	1(standard)	2(standard) - 1(AR(3))
4	-0.250724	5.323477	1(standard)	1(standard)
5	-0.008539	0.814716	1(standard)	1(AR(3))
6	-0.020295	1.691549	1(standard)	2(standard)

Table 18: Table with the relevant estimates of mean profits distribution; the sample mean and sample standard deviation are used to visualize which are the *outliers*. All the values are considered on a simulation horizon $T = 13$. The higher table is for the no arbitrage framework while the lower one corresponds to the aligned market.

A first outlook to the table above bring some very interesting results. *The outlier investors are always rational traders, both in the case of outperformance or underperformance. Therefore rational trading strategy is actually a way to try to beat the market, but at the same time it can be also entail underperformance. So the situation in the market in terms of mean profits is the following: there are several random traders which usually get no significant mean profits and few rational traders who have positively exploited the consultancy, or on the contrary, who have made losses from it. It is quite normal that random investors will tend to realize null mean profits because a strategy without no market knowledge and which varies in a randomly way cannot produce significant advantages. A rational strategy instead tends to lead to extreme profits. These results are also valid in a market which is aligned to the real one.*

A way to proof is if $\hat{mean}(\overline{\pi})$ is consistent with the *null hypothesis* of zero: taking the average over all the ten simulations we execute a *t-test* in both scenarios.

	$avg(\hat{mean}(\overline{\pi}))$	π_0	t
no arbitrage	0.004669±0.005044	0	$\frac{0.004669}{0.005044} = 0.66 \Rightarrow accept$
arbitrage	-0.122479±0.050371	0	$\frac{0.122479}{0.050371} = 2.43 \Rightarrow reject$

Table 19: *t-statistic* to test the *null hypothesis* of zero for $\hat{mean}(\overline{\pi})$ in the no arbitrage framework.

For the no arbitrage framework we conclude that the hypothesis of zero $\hat{mean}(\overline{\pi})$ is statistically acceptable; the market is efficient if we look at the general mean profits distribution. However this does not exclude the possibility of some investors who can exploit some degree of inefficiency to get significant positive profits. If we allow arbitrage operations and the market is kept aligned to real data, the situation totally changes and the hypothesis of zero $\hat{mean}(\overline{\pi})$ is rejected. In particular we can see that $avg(\hat{mean}(\overline{\pi}))$ is negative, i.e. investors used to have financial losses through trading operations. *The real market is even more efficient in destroying any opportunity of actual profits and a random trading strategy will be always underperforming.*

Now the next step is to consider what kind of expectational model was used by *outlier* investors. More precisely if a portfolio has outperformed or underperformed in the market which is the econometric model which was based on? You can observe from the table that, in the no arbitrage context, almost all the *outliers* have taken advice from a *standard* consultant or from a *chartist AR(3)*, who use respectively the most simple expectational model and the most complex one to make forecasts about returns. However such two expectational model they can both determine outperformance or underperformance and we need a more specific and quantitative analysis which deals with the performances of all forecasting methods. In addition, when the market is aligned to the real one, we have only standard consultants to support the outperforming *outliers*. All such results have to be commented under a different perspective.

Until now we have just seen which is the performance of the outliers in terms of mean profits, but we have not considered the ideal performance of optimal portfolios advised by consultants. Next step is to look at returns of optimal portfolios and compare them to the results regarding *outlier* investors.

8.4 Performance of consultants

Actual performances of investors have to be judged in relation to which is the return of the optimal portfolios they have used during the simulation. This also justifies the choice to exclude learning update in rational

strategies because changing consultants during the simulation would avoid a comparison between ideal and actual performance of a trading strategy. *In this work we are very interested in the contrast between what can be the ideal return of optimal portfolio and what is the corresponding gains deriving from the attempt to construct such portfolio. This is just the hot topic to face speaking about complexity in a stock market.*

Let us use method *CheckConsultancy* to consider the daily return of optimal portfolios built by all consultants: at the end of every simulation we compute the average optimal daily portfolio return for each kind of consultant. *We repeat that the optimal portfolio is the portfolio created by a consultant and advised to her clients, but it is not the actual portfolio formed by rational traders.* Let us formally define the daily return of the optimal portfolio

$$r_t^{opt} = \sum_{i=1}^{N} w_i' r_t^i$$

Such value is simply the weighted sum of the four daily stock returns (according to the optimal shares w_i') and then is an estimator for out-of-sample performance. Clearly we are interested to consider the mean daily return of such portfolio

$$\overline{r_{opt}} = \frac{1}{T} \sum_{t=1}^{T} r_t^{opt}$$

This is a simple but robust estimator to use in classify the forecasting methods used by all consultants. The results are reported here for ten simulations both with and without arbitrageurs; in the arbitrage case we have run just five simulations because statistical fluctuations are less important (as you shall check from the standard error for the average over the simulations).

seed	*standard*	$AR(1)$	$AR(2)$	$AR(3)$
2	0.00430	0.00362	0.00239	0.00404
3	0.00772	0.00799	0.00965	0.00845
4	0.00891	0.00783	0.00860	0.00989
5	0.00767	0.00821	0.00778	0.00744
6	0.00796	0.00750	0.00632	0.00731
7	0.00427	0.00431	0.00462	0.00478
8	0.01030	0.00982	0.00979	0.01105
9	0.00625	0.00419	0.00319	0.00536
10	0.00467	0.00440	0.00393	0.00491
11	0.00487	0.00519	0.00456	0.0045

seed	*standard*	$AR(1)$	$AR(2)$	$AR(3)$
2 (arbitrage)	-0.000313	0.000010	0.000148	-0.000092
3 (arbitrage)	-0.000298	-0.000042	0.000059	-0.000051
4 (arbitrage)	-0.000287	0.000031	0.000081	0.000090
5 (arbitrage)	-0.000322	0.000017	0.000131	-0.000115
6 (arbitrage)	-0.000303	0.000026	0.000019	-0.000001

Table 20: Table of mean daily performance ($\overline{r_{opt}}$) for each kind of consultant with respect to their optimal portfolio; the simulation period is $T = 13$.

Now it is necessary to consider the average over all the simulations to have a robust estimator: we give the results in decreasing order as a sort of ranking of performances.

	$avg(\overline{r_{opt}})$		$avg(\overline{r_{opt}})$
$AR(3)$	0.006773±0.001091	$AR(2)$	0.000088±0.000017
$standard$	0.006692±0.000952	$AR(1)$	0.000008±0.000009
$AR(1)$	0.006306±0.000979	$standard$	-0.000305±0.000004
$AR(2)$	0.006083±0.001219	$AR(3)$	-0.000034±0.000026

Table 21: Average values of mean daily performance (computed as mean daily optimal portfolio return) for each kind of consultant, with their standard error. The list is in decreasing order to show the ranking for the four classes of expectational models

The final ranking here reported is just a simple way to classify the expectational models in terms of forecasting power. Since every estimator has its standard error, as shown in the list, the ranking is not absolute. Indeed *t-tests* could be made to compare each couple of consultant classes and so we could provide a sort of reliability degree for each position in the ranking. Anyway this is an enough robust classification and it could be eventually reinforced running more simulations to reduce the standard error.

In the free simulated market the AR(3) chartist is that with the best optimal portfolio return, the second one is the standard consultant: therefore the two best kind of expectational models are just the those mainly used by the outliers. What does this result mean for our simulated market? We have evaluated the mean out-of-sample performance of the four different optimal portfolios and we have prooved that only the traders who have used the best portfolios can achieve an outperformance in the market. Anyway there is the risk that some rational investor who is using such models can also undeperform: this seems strange because we should expect that better out-of-sample performances always imply better mean profits. In order to understand the origin of such phenomenon think about the rational investment scheme: once a rational trader has obtained the optimal portfolio estimate, she has to decide in which way can achieve the portfolio goal shares. The freedom in the choice of the several ΔW_t^j implies different trading stategies, even with the same optimal portfolio, so we can have significant differences in the actual mean profits. *The common point is however that a model with better out-of-sample performance entails higher fluctuations in the actual mean profits: more forecasting power is paid in terms of higher risk and this is also acceptable in the framework of standard financial theory.*

What about the aligned market? *Introducing arbitrage operations, the ranking order of the forecasting performances is reversed: the AR(3) chartist is the worst consultant and the second worst one is the standard analyst.* When stock prices are aligned to real ones, the pricing methods which account for autocorrelations at the first or second lag (i.e AR(1) and AR(2) processes) are the best ones in terms of out-of-sample performances. The most complex model (the AR(3)) totally fails in build a well performing portfolio: considering autocorrelations at too many lags is counterproductive for out-of-sample performance. In this context the most simple model, which is represented by standard consultant, is inferior to the those defined by AR(1) and AR(2) chartists, but performs better than the AR(3) pricing method.

Such classification of out-of-sample performances can be better explain in terms of Machine Learning theory [27]. The four classes of pricing models used in this ABM are conceived to be increasingly complex. According to Machine Learning, complexity of a model can be measured through the number of parameters needed to define the model itself: this number is related to what is called as *Vapnik-Chervonenkis dimension*

[27] See the Appendix regarding Machine Learning principles and also consult Abu-Mostafa et al. in *Learning from data*

(d_{VC}) and is a very useful measure to correctly evaluate the trade-off between in-sample error and out-of-sample error. If the learning model is very complex (high d_{VC}) the out-of-sample performance tend to be poor, despite a very low in-sample error; a simple model (low d_{VC}) is instead more effective in make more general forecasts, i.e. with lower out-of-sample error. The d_{VC} is minimum for the pricing method used by standard consulants and maximum for the AR(3) chartist.

When the market is just simulated, the most complex model manages to exploit the degree of predictability in stock prices. If we reduce the number of lag returns considered to make forecasts, using the optimal portfolio build by the AR(2) or the AR(1) chartists, the performance falls and it is better to use the most simple pricing model. So only with a very high d_{VC} we can catch some predictability in stock return series, otherwise it is better to apply a very naive forecasting method (with low d_{VC} and high in-sample error) to try to get significant mean profits. The *outliers* will reflect such considerations.

With the simulated market that coincides with the real one the situation is overturned. The AR(2) and AR(1) processes (which correspond to intermediate values for d_{VC}) lead to the best optimal portfolios. The most simple pricing method, used by standard consultants, is now too naive to predict future stock prices; only the AR(3) chartist get worst results, since her model is now too complex and produce too specific forecasts, with very high out-of-sample error. What did actually change introducing arbitrage operations? When stock prices are very close to real ones some level of predictability also exists and this is quite normal because autocorrelations are typically observed in real markets. Therefore technical analysis can be exploited to make effective forecasts an build a performing portfolio. Machine Learning principles however advise to use a not too complex model, since the latter is very efficient in fit the observed data (very low out-of-sample error), but is very weak in generalize its forecasts. At the same time the trivial model of standard analysts is too general and its very high in-sample error leads also to very high out-of-sample performances. The trade-off is solved using a model with an intermediate complexity, i.e. a not too low or too high d_{VC}: this is a common solution to model selection in Machine Learning.

But why is this no more valid if stock prices arise from the simulation without the intervention of the arbitrageurs? The reason can be foreseen using some experience which regards both ABM markets and machine learning. In the totally free simulated scenario the agents contribute to form stronger autocorrelations in stock prices with respect to the arbitrage framework, as we can also understand comparing the mean profits distribution in the two situations; arbitrage implies a more efficient market, which can be also autocorrelations coefficients which rapidly goes to zero after very few lags. Without alignment to real data, a more complex model is useful because autocorrelations are significant different from zero also after the two first lags: that is the reason why an AR(3) chartist can construct the best optimal portfolio. Anyway if you reduce the number of time lags in your forecasting model, the time series of stock returns is no more well fitted: the in-sample error rises significantly and this also augments the out-of-sample error, which is yet enough high. The solution is to consider an even simpler pricing method, which will entail a higher in-sample error but a better out-of-sample performance. So you can either use a very complex model with three different time lags, which allows to make elaborate forecasts, or it is better to choose the most trivial method. The simulation framework then can reinforce autocorrelations in stock prices, while the arbitrage operations lead to a limited level of predictability, as occurs in real markets.

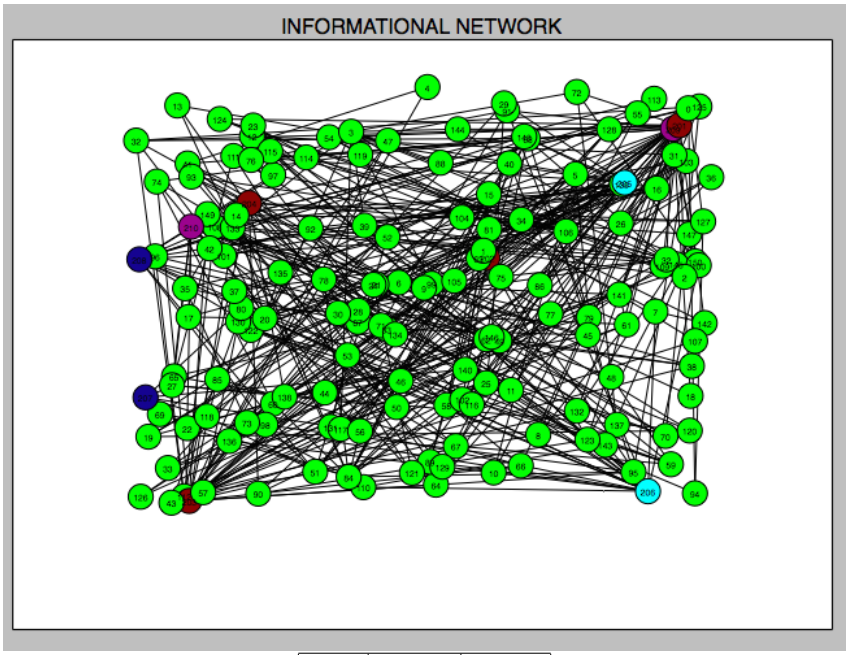
The last important comment regards the link between the best performances of consultants and the *outliers*, in the arbitrage scenario. AR(1) and AR(2) chartists are the best consultants, but the *outlier* traders are clients of standard analysts. This is quite strange because we expect that the best consultants will also advise the *outliers* (even in the underperforming case), as happens in the no arbitrage framework. If we want to solve this dilemma we have to change perspective with respect to the considerations made above. Machine Learning theory can help to explain the order ranking for the performance of consultants, but this is not the end of the story. *Indeed a very good out-of-sample performance of the optimal portfolio*

does not necessarily imply better mean profits for the traders who are using such portfolio goal. When the market is aligned to the real one this fact becomes really evident. AR(1) and AR(2) chartists manage to create good optimal portfolios, but the traders who are able to beat the market always have adopted the portfolio goal of a standard consultant. The reason is that when the market is very similar to the real one actual chances to exploit market predictability drastically fail. Being more precise, the market always presents some level of predictability and indeed AR(1) and AR(2) chartists display good performances, but the problem is that exploiting such level of predictability is very difficult when traders operate in the double auction system. The better solution is then to use the most simple and trivial model which can bring more profits. A system with so many agents and stock prices which reflect real dynamics is very complex and we have a clear difference between the strategy that makes the best forecasts and the strategy which guarantees the best profits.

8.5 Special case: learning framework in a completely rational market

All the results regarding mean profits of investors and performance of consultants were considered in a market in which traders maintain the same consultant for the entire simulation. We have remarked that such approach is due to correctly evaluate both performance of analysts and their implementation through trading operations of investors. However it could be interesting to present a special framework which allows investors to freely update their rational strategy and select a better consultant.

As we have repeated many times, the informational network is the tool through which every trader can learn about the market and modify her investment plan; this is essential for rational investors who wants to change consultant, so having a new optimal portfolio goal. In a market mainly composed by random traders, as in our case with $n_{rand} = 0.9$, network effects tend to privilege $RAT \rightarrow RAND$ transitions with respect to changes of consultancy: it is statistically easier to have a neighbor random investor than a rational one and then there are so less chances of being influenced to call a new analyst. So it is better to change our basic settings in the simulation, considering a completely rational market, i.e. formed only by rational traders. these are the modifications adopted for this new framework.



M	n_{rand}	n_{cons}
150	0	0.07

Table 22: Initial informational network in this completely rational market, together with the basic settings for the total number of agents: there are 150 rational investors and 10 consultants.

We have included only 150 and not 200 investors as done before because we want to reduce the simulation times, which are yet dilated in a completely rational market (more test which regard network and herd effects). The list of consultants remains instead the same used until now. We havae also decided to study this special case because it can be compared to the SFI market, where all tha agents were always rational and continuously adat their expectational model. Here network effects carry out the learning task, while in the SFI market the agents do not directly influence each other.

The next step is to ensure that the number of new consultancies in the simulation is quite significant and this is achieved setting $\lambda_{test} = 0.1$, which we remind entails a mean frequency of one new consultancy request per five days and for each investor (for the discussion see Chapter 5). In addition we fix $x_{change} = 0.1$ because in this way the reliability requirements of your neighbors, in terms of daily profits, are not so high and the number of actual changes of consultancy will be remarkable.

Now, with all these new settings, let us run a single simulation in the long-run period: it is suffice to explore the results with a single seed because we are interested in some global properties which must not depend from the choice of the seed. In particular we shall monitor the daily mean profits and the final number of clients of each single consultant. *The idea is that, after the initial random choice of an analyst, rational traders will tend to be influenced by investors with the best consultant: the market dynamics will naturally lead the network towards a stable state in which almost all the traders will consult the same analyst,*

i.e the financial advisor with the best forecasting method. In this case when we talk about the best forecasting method we do not intend the method which guarantees the best a priori performance, but that which ensures the actual and most performing portfolio in the market, or in the same terms that which leads to the best mean profits.

First let us show the mean profits distribution in our completely rational market.

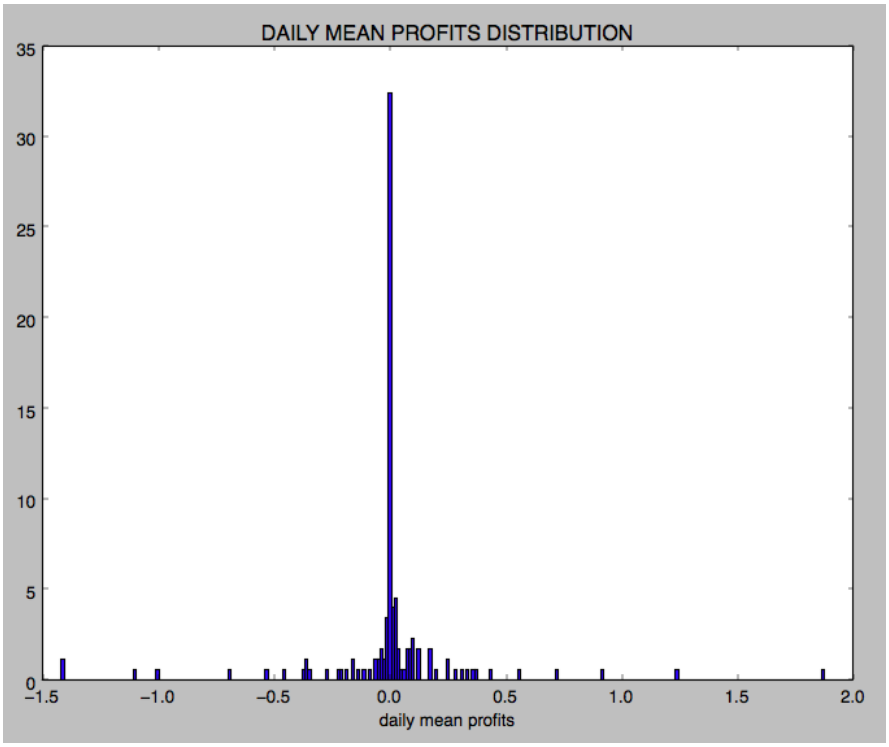


Figure 29: Plot of mean profits distribution (with normalization) in our completely rational market for $seed = 2$: the simulation period is $T = 40$ and we are not allowing arbitrage operations. The *outliers* are represented by the extreme bins in the histogram.

The great difference with respect to a market in which there are many random investors is the presence of many traders with mean profits significantly different from zero. The distribution contiue to be centered around zero, but the tails are now very consistent; this prevents the use of a gaussian distribution to fit such data. Rational traders can beat the market with higher frequency, but also significant losses become more likely. Clearly also the number of the *outliers* has risen in this framework. *Therefore a completely rational market is more volatile in the distribution of daily mean profits: rational strategies, as we have yet seen looking at the outliers in the standard case, can produce more extreme gains from trading operations. From the other hand random traders, who instead tend to realize null (or negative in the aligned market) mean profits, will reduce the opportunities of beating the market for many rational investors.* You could better test the latter sentence varying the number of random traders in the market, at fixed number of rational

agents: the percentage of rational outlier traders rises if the number of noise investors is lower. *Such effect is really important because in a real stock market random investors are usually more numerous than those who apply a rational trading strategy. These noise traders then make the market more efficient limiting the chances of outperformance, even if the market displays some level of predictability in its stock prices; the term noise is therefore very appropriate.*

We can also observe which analysts were called by the outliers. Here we report a symbolic result with a single seed.

n_{out}	n_{under}
2(standard) - 1(AR(3))	3(standard)

Table 23: Table with the *outliers* in a completely rational market with $seed = 2$: the changes in parameter settings are $\lambda_{test} = 0.1$ and $x_{change} = 0.1$. The sample mean and sample standard deviation are used to visualize which are the *outliers*; all the values are considered on a simulation horizon $T = 40$.

The standard consultant and AR(3) chartist have advised once again (as in the standard market) the *outliers*. Indeed all the conclusions regarding the effectiveness of the four forecasting methods are not changed in such framework. If you do not trust in this result because are referred to a single seed, you can run more simulations and see that statistical fluctuations do not affect the conclusions.

Finally we report a numerical estimate of the number of clients for every class of consultants; we sum the links (i.e. the clients) of analysts of the same kind.

	standard	$AR(1)$	$AR(2)$	$AR(3)$
number of clients	148	1	0	2

Table 24: The table shows which are the consultants with the higher number of clients in the market (degree in the informational network) in a completely rational market; the changes in parameter settings are $\lambda_{test} = 0.1$ and $x_{change} = 0.1$. The simulation period is $T = 40$.

Looking both at the network and at the table, we understand that almost all rational traders will follow the advice of standard consultants. *Network self-organizes to guarantee that all the traders, through social interactions, can progressively select the best analysts in the market, i.e. those who apply the most simple pricing method.* The fact that standard analyst become the most called by rational traders means that their optimal portfolio is the best performing. In the standard case, we have shown that the AR(3) process is the most accurate to produce good out-of-sample performances; with only rational agents, the situation has changed. The market is composed only by investors who perform a rational trading strategy: in particular many of them (i.e. those who consult the three kinds of chartists) invest on the basis of the existence of autocorrelations in stock returns. If many traders operate in such direction, stock prices (and returns) will progressively reflect these expectations and autocorrelations coefficients will be softened. For instance if

you foresee that prices will continue to rise, then prices will imediately augment as a consequence of many purchasing operations and in the future the positive autocorrelation is much less likely. Agent expectations affect stock prices and destroy chances of predictability: this is the Efficient Market Hypothesis. In such complex rational market the effect is similar in some sense. Clearly information is not immediately and totally reflected in stock market dynamics (as for the EMH in its strong form), but forecasting methods which exploit technical analysis will become fruitless if many traders will base their operations on them. On the opposite the method used by standard consultants is so trivial and not related to recent price dynamics, and this ensures a better out-of-sample performance for the corresponding optimal portfolio. If we added random traders in the market (so lowering the relative number of agents who follow technical analysis) the optimal portfolios built with chartist methods would guarantee higher returns.

It is possible to compare all these results to those obtained in the SFI market. Remember that there are two different regimes in that famous artficial stock market: if the learning rate of the GA is low the expectational models converge towards a sort of equilibrium (*homogeneous rational expectations*); if the learning rate is high expectations of agents remain quite various in the market (*heterogeneous rational expectations*). In our special case we could say that the network naturally converge towards the case of *homogeneous rational expectations*, but only the second regime of the SFI market is that which determines complex properties of a real market. However, it is essential to remind that in our work agents with the same expectational model can perform different trading strategies: realistic market features are preserved even if information dynamics lead all the traders to call the same kind of consultant.

Information dynamics, which is represented by the evolving informational network, naturally drives the set of expectational models towards a specific attractor. The latter is just the best performing pricing method, which corresponds, in this completely rational market, to that used by standard consultants.

8.6 Informational network analysis

The last section of the analysis wants to consider the properties of the informational network introduced this work. The series of statistical measures and tests that could be performed is very wide, but we limit our attention to some fundamental features. From the perspective of complex network theory we are facing a *temporal* network because its state changes over time together with market dynamics. This means that several network measures evolve over trading days, in particular in response to the daily profits of investors (which depend their self from market state). Every new trading day presents a new series of daily profits for investors and this determines an update in the network. In several situations what does really matter is the state of the informational network after many trading days, i.e. a sort of steady state for this temporal network. Therefore the relevant theme is to look at the properties of the network when it reaches such steady condition, which is not an actual steady state in the sense of dynamical systems but is just relative to some global properties. *Indeed the network reflects a non-equilibrium system as a stock market, where the agents and their interactions will never converge towards some equilibrium state.*

Firstly it is essential to understand that after many trading days there is a sort of *saturation effect* due to the finite size of the network. You can see a first sign of this effect after some trading days.

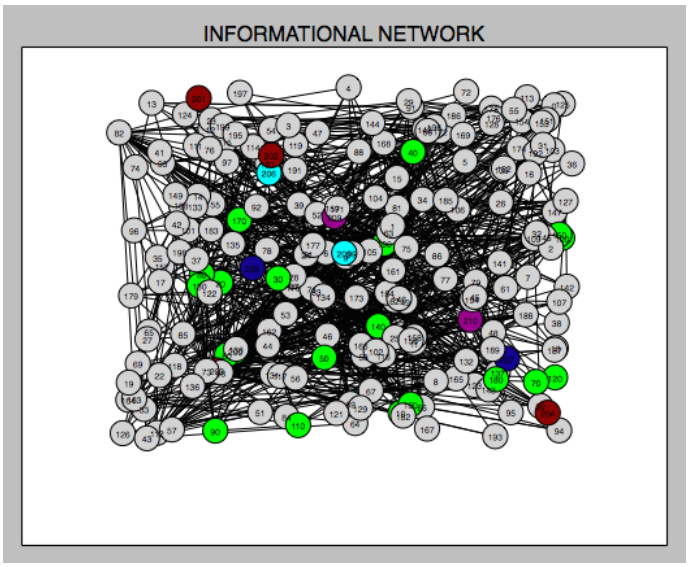


Figure 30: Informational network at day $T = 40$ with $seed = 2$.

The saturation occurs because more and more links are introduced in the network and this can be verified from this plot built at the end of the simulation horizon.

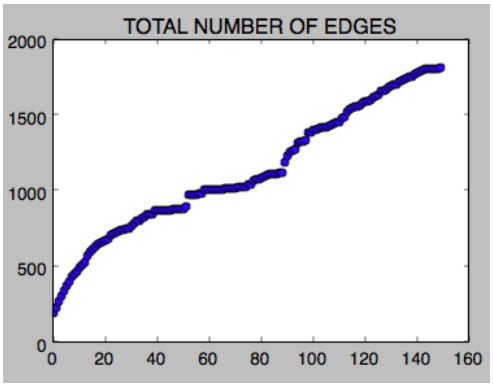


Figure 31: Time evolution of total number of edges in the informational network at day $T = 160$ with $seed = 2$.

Here the saturation is yet far from being reached, because the actual maximum number of links in an undirected network is $M(M-1)$ (which is equal to 39800 in our model). However it is easy to see that there are statistical fluctuations (and even jumps) in this increasing trend: the number of new incoming links is clearly stochastic. Why does the number of edges increase over trading days if we use an algorithm which add a new link (with the best neighbor of your best neighbor in terms of daily profits) and remove

an old one (with the worst neighbor)? The answer is in the the emerging network rules which concern the investors with no initial acquaintances in the market. Remember that such traders will continue to search their first contacts in the network using the initialization algorithm: so they always contribute to rise the total number of links. This phenomenon will persist over time because it can occur that some traders, those who have revealed as the worst neighbors of some other investor, have lost all their links, having once again a zero degree. The algorithm does not allow that an investor can remain with zero links because her tendency is to try to get new acquaintances as sources of information: the consequence is the increasing number of edges in the network, towards a complete saturation. This behavior of the network has to taken into account when we will speak of the global network measures.

8.6.1 Degree distribution dynamics

The first essential tool which photographs the state of the informational network is the degree distribution. As discussed in Chapter 2 dealing with the initialization algorithm for the network, the initial degree distribution is well proxied by a Poisson function, where the expected value is 1: *we remind that in this case we are considering just the network among investors and then consultants are not included here. At the beginning of the simulation the network is a random graph because represents the random nature of acquaintances in the market.* As the trading days go on the evolution algorithm tend to reward, in terms on new acquired links, the investors who have realized the best profits in the network: this selection process will affect the form of the degree distribution and we show its result below with a single seed.

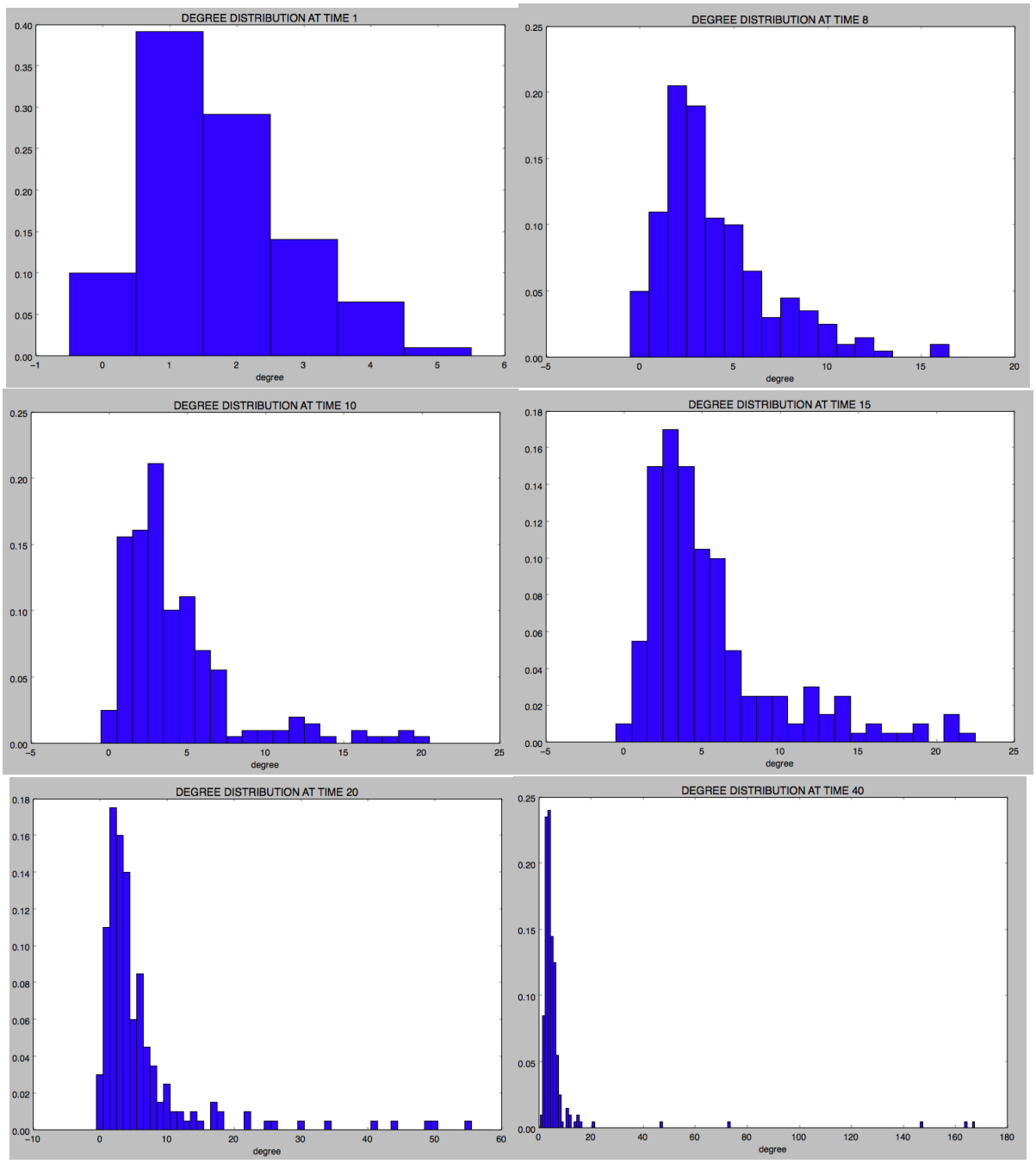


Figure 32: Degree distribution evolution (with normalization) until time $T = 40$, with $seed = 2$.

The plots show how the degree distribution evolve over time: starting from a Poisson distribution we can see how the tails becomes more and more important (the skewness becomes higher). The mean value of the distribution clearly moves forward because introducing new links in the network progressively reduce the number of investors with few acquaintances (low degree). *However the relevant property is that, as time goes on, the presence of nodes/investors with very high degree becomes relevant: network dynamics naturally brings out hubs. The latter effect is an emerging property of the system and it does not depend on some a priori algorithm which we know can determine arising of hubs, as for example in the case of the famous preferential attachment mechanism proposed by Barabasi* (1999). *On the contrary the informational network just evolves on the basis of market dynamics and according to trading operations of investors (which determine profits): so the existence of hubs is an unexpected feature which appears from the complexity of the stock market.*

Although there are hubs in such network we cannot speak about a scale-free graph, i.e. a graph with scale-free power-law[28] and this can be checked simply looking at the histograms above. The degree distribution is not totally decreasing but presents an evident maximum (the mode of the distribution) for very low degree. *So the network preserves some of its initial features of random graph: is this effect a consequence of the chosen initial network or the evolution algorithm always do not entail a complete convergence towards a power-law degree distribution?*

Both the hypothesis are actual drives in the formation of the almost stationary degree distribution. The random initial network fixes the the starting basin of acquaintances (neighbors) for the investors and so its essential to make the first pulse to the evolution algorithm; from the other hand the process to update the links in the network is strongly affected by market fluctuations because it looks at daily profits, which are very volatile: so in some sense there is an intrinsic stochastic component in the emerging network rules. *In conclusion the informational network has necessarily to maintain some features of a random graph but, at the same time, displays some interesting hubs, which are typical of scale-free distribution.*

In the context of complex network analaysis, we are in particular interested to verify the power-law trend for the decreasing part of the degree distribution. Therefore we need to discard the consider just the portion of histogram which starts from the mode of the distribution, i.e. the degree with the highest frequency (the most likely in the network). Looking just at this part we use perfrom a non linear fit with power law function

$$p(k) = Ck^{-\alpha}$$

with C as normalization constant and α as the exponent of the power law; the latter is typically included in the range $(2, 3)$ for real scale-free networks. Let us show the distribution with this fitted curve, using a single seed.

[28] See *Statistical mechanics of complex networks* (2002) for a complete discussion of random graphs and scale-free networks

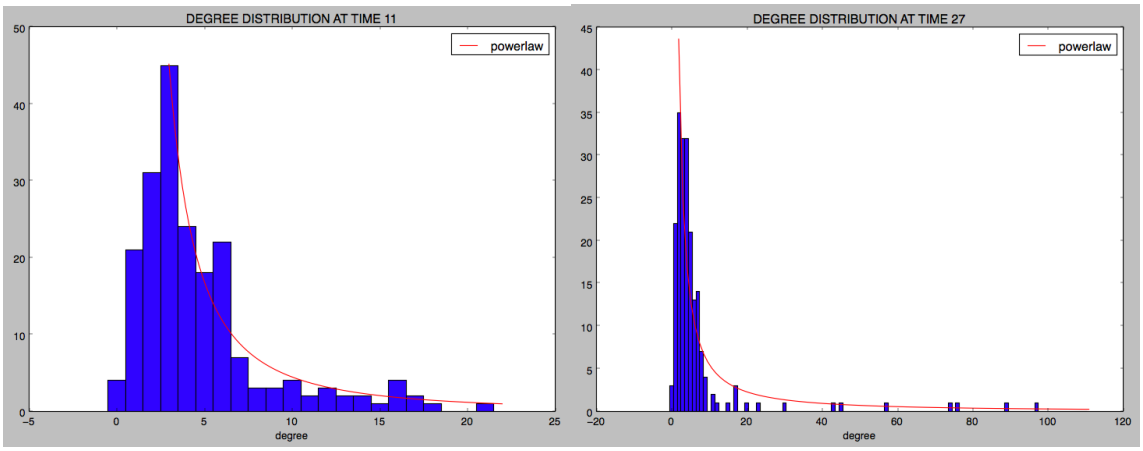


Figure 33: Degree distributions (without normalization) with the fitted power law at different trading days, always with $seed = 2$.

The power law trend seems very effective in fitting the tail of the distribution, but it reveals quite unsuitable for its main body (even starting to fit from the mode). The chi square values for the fit are then very high and we do not report them because we know that such degree distribution is not actually power law for its entire extension. The goal is instead to monitor the time evolution of the exponent for the power law, checking if it approaches the range usually observed in real scale-free network or not. Here we report a plot that shows this ; a single seed is really sufficient to catch the general trend.

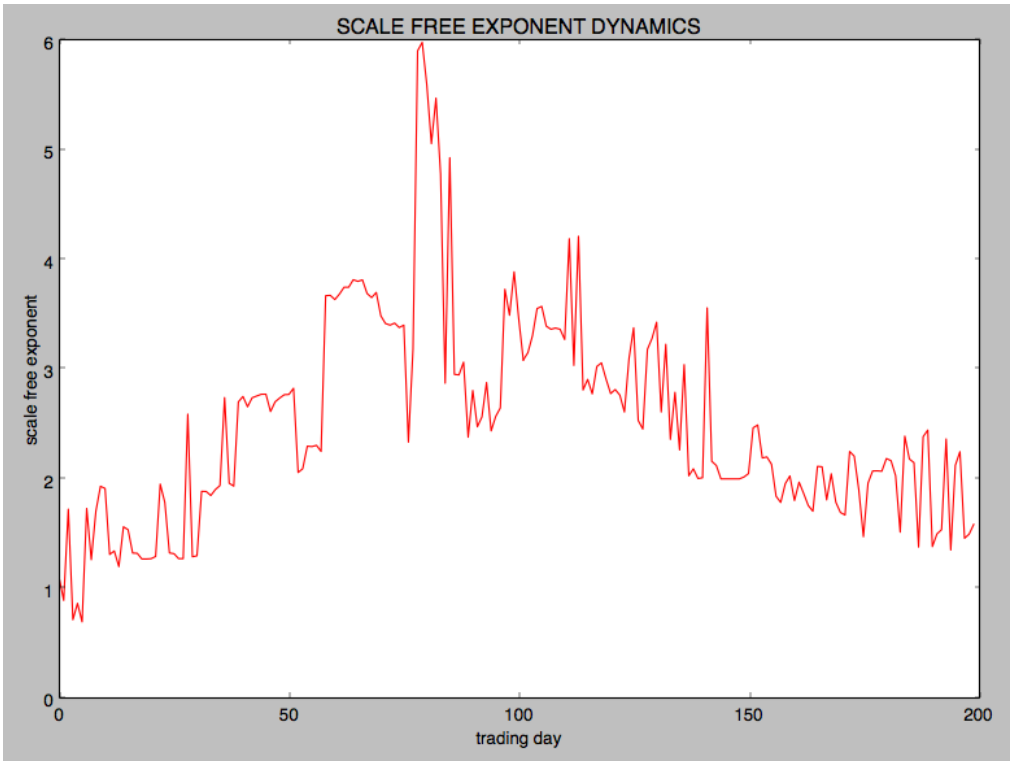


Figure 34: Time evolution for the scale-free exponent until trading day $T = 200$ and with $seed = 2$.

The observed dynamics clearly presents so many stochastic fluctuations due to the network which self-organizes over time in response to the market. However we are interested in the mean trend of the process. First the exponent of the power law tends to increase reaching, around 75-80 trading days, very high values (almost 6). Afterwards the effect is reversed and the coefficient will progressively decrease. Finally, approximately after 150 trading days (almost six months) the exponent starts to swing around a fixed value. The latter can be thought as a sort of asymptotic scale-free exponent and is around 2. *The fact that a global network property establishes fluctuations around some fixed value is a very general feature of our informational network*; we shall deal with such theme in next paragraphs. For now it is instaed more important to reflect about the specific dynamics which precedes the final fluctuations. When the scale-free exponent rises we could think that tails becom less important, but on the contrary we know that *hubs* are emerging in the network. However the fit also considers the central part of the distribution and this alters our intuition. A higher power law exponent means that the fitted curve for the distribution converges to zero more rapidly: the number of nodes with low degree (the main body of the histgram) is significantly higher than the number of emerging *hubs*. So an increase in such exponent means the network is converging to the scale-free behavior, starting from the initial random distribution. So in a first phase α rises because the distribution rapidly reaches low values for limited degrees, although *hubs* have appeared and also continues to have their importance. The increasing period leads to very high α values, beyond the typical range of scale-free networks: the contrast between isolated (with very few links) agents and *hubs* is

very pronounced, i.e heterogeneity is so high. Then the exponent starts to decrease because the number of agents with intermediate degree values augments; network returns towards a more realistic heterogeneity level. At the end of this dynamical process, fluctuations are established around $\alpha \simeq 2$, a value which proves that our informational network has stabilized towards scale-free behavior.

Now the remaining question is to understand what are the features of the *hubs-investors*: can we proof that they are just the traders which typically realize the best mean profits in the market? The next part of the analysis looks at such problem.

8.6.2 Mean profits vs degree of investors

Since our informational network evolves according to market state and to the profits realized by investors it is very important to look for some relationships between some network measures of a node and its corresponding variables in the market. Such approach corresponds to build a bridge between Network Analysis and Agent-Based Modeling: the agents are conceived both as part of the simulation, i.e. having some relevant micro variables which characterize them, and as nodes in the network. Once again there are several possible analysis if we look for such parallelism.

In this work we investigate the most important one for our kind of research. Can we find a straightforward and testable relationship between the number of acquaintances in the network and the mean profits obtained over time, , for a certain investor? Or instead there is no significant correlation between the two variables and mean profits will depend on other factors?

The comparison of degree and mean profits can be performed in two main ways: the first is to consider a data point as a single investor, with its degree and its final mean profits; the second regards the more elegant concept of spectral representation for the mean profits. The method which exploits single investors to build the sample to analyze can be too biased from the single performance of just few traders and so is more volatile with respect to the specific seed or market history. So we choose the method of spectral representation which divides all the investors in classes on the basis of her degree: every class includes all the investors with the same degree. Then it is suffice to take the average mean profits over all the investors of a class

$$\overline{\pi}(k) = \frac{1}{M_k} \sum_{j|k_j=k} \pi_T^j$$

where M_k is the number of investors with degree k. This expression determines the *spectrum of mean profits with respect to the degree*. At the end of a simulation we make such classification and compute, for each k, the corresponding $\overline{\pi}(k)$. Afterwards we can run a linear regression over the degree to estimate the intercept and the angular coefficient

$$\overline{\pi}(k) = a + b \cdot k$$

Such process is elegant and effective because it helps in reduce single fluctuations of some outliers, so removing part of the *deterministic noise* in the process. Here we show a plot with the fit line for a single seed.

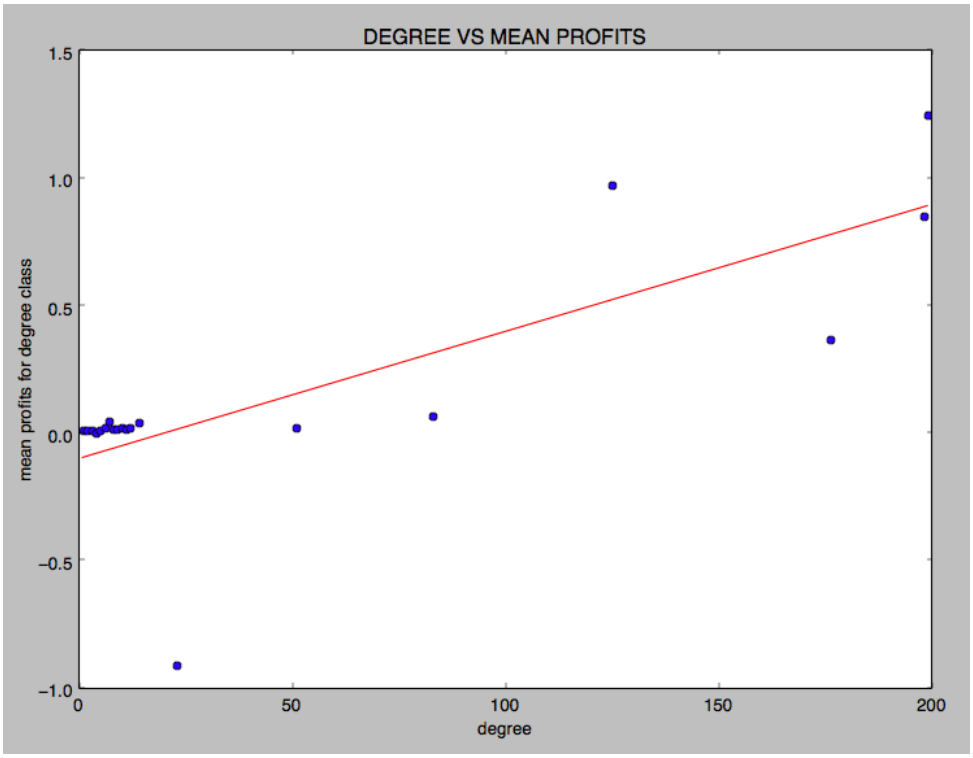


Figure 35: Linear relationship between degree and mean profits using the spectral representation, on a horizon $T = 50$ and with $seed = 2$.

The tendency of the network seems to . Let us run ten simulations choosing two different horizons, one for the short-run period ($T = 15$) and one for the long-run ($T = 50$)

seed	*a*	*b*	*seed*	*a*	*b*
2	0.00893485	-0.00014235	2	-0.09805	0.00498534
3	-0.0301551	0.00255158	3	0.0942108	0.01673452
4	0.00287802	0.0001861	4	0.43533537	-0.00049665
5	0.00071289	0.00029769	5	-0.06802736	0.00544437
6	0.00098385	0.00649118	6	0.12621377	-0.00297007
7	0.00853435	0.0001202	7	0.00778945	0.0020033
8	-0.27336922	0.04699603	8	0.28862357	0.00232469
9	-0.03024927	0.00670808	9	-0.04716175	0.00029655
10	0.00880401	0.00028981	10	-0.11268042	0.00732244
11	0.00368880	-0.00006703	11	0.06684628	0.00006049

Table 25: Tables of linear fit parameter estimates for spectral representation of mean profits vs degree; the simulation horizon is $T = 15$ on the left and $T = 50$ on the right.

In order to organize these data we have to take an average over all the simulations for the angular coefficient b: we are not interested in the intercept because only the angular coefficient can catch some relationship between mean profits and degree. As we have done previously in other sections of analysis we execute a *t-test* to check a statistically relevance for b: the *null hypothesis* assumes no correlation between the two factors, i.e. $b = 0$.

T	$avg(b)$	t
15	0.00634313±0.00459488	$\frac{0.00634313}{0.00459488} = 1.38 < 1.68 \Rightarrow accept$
50	0.0035705±0.00176147	$\frac{0.0035705}{0.00176147} = 2.07 > 1.68 \Rightarrow reject$

Table 26: *t-statistic* to test the *null hypothesis* of zero angular coefficient. The existence of some relationship between mean profits and degree in informational network is aborted at trading day $T = 15$ but accepted after $T = 50$ trading days.

The *t-tests* show that on the short-period, i.e. after $T = 15$ trading days, we have no definite relationship for mean profits vs degree, but extending the simulation horizon to $T = 50$ trading days the relationship becomes statistically significant.

Therefore in the short-run period there is no evident correlation between mean profits and degree of investors: the system (intended as the market together with the network) is yet too volatile and investors who have achieved the best mean profits may not necessarily have a higher degree with respect to the rest of the traders. Augmenting the number of trading days the selection algorithm is strengthened because there are more links in the network and so outperforming investors have the chance to get more and more acquaintances. The stochasticity in the market is overwhelmed and the informational network has made a self-calibration which awards, in terms of the number of acquaintances, the traders with best mean profits.

In the data with different seeds, you can observe some situations, though very rare, in which there is a negative relationship between mean profits and degree. In order to explain such events you have to remember that all the angular coefficients reported are fitting estimators and so have their statistical error which was not considered since the goal is to average over many simulations and directly use the mean angular coefficient; so some times a negative steepness of the fit line may not exclude an actual positive relationship for mean profits vs degree. However the relevant measure is just the average angular coefficient over the simulations, since the latter is able to reduce all stochastic effects of single market realizations at a specific seed.

8.6.3 Global network measures

Another fruitful field of analysis is relate to some specific global properties of the informational network, where this time we return to take into account both investors and consultants in the network. We have decided to monitor three relevant network measures to compare them to other typical complex networks: the number of connected components, the mean clustering coefficient and the level of degree assortativity[29]. All such variables will evolve over time and we report such evolution for a single seed because the general dynamic behavior does not vary from seed to seed.

[29] A theoretical description of these network properties can be found in *Dynamical Processes on Complex Networks* (2008)

Connected components Let us begin to see if the informational network becomes totally connected through time. The realistic initial condition implies an informational network composed by many connected components, which can be interpreted as groups of initial acquaintances (directed or undirected). We briefly remind that connected components are subgraphs of the global network which are totally connected: two different components have no common links, i.e. there is no path to move from a component to the other. In this work network dynamics should bring out a more and more connected graph; we consider a very large time horizon of $T = 60$ to be completely sure to observe the stationary property.

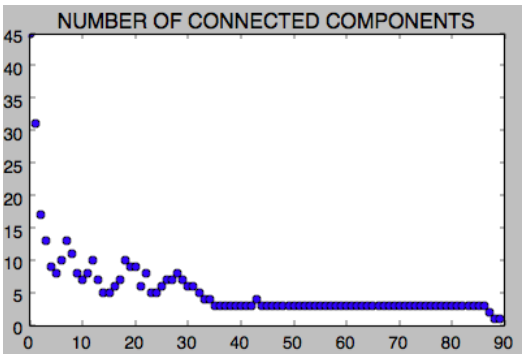


Figure 36: Time evolution of the number of connected components in the network on a horizon $T = 90$, with $seed = 2$.

The number of connected components decrease quite rapidly, as recognizable above, and in just few trading days reaches the stationary level. In order to evaluate if the graph is totally connected you have to be careful to the existence of consultants who were not called by any investor at the beginning of the simulation: indeed, since we are operating in a framework without new consultancies, the analysts not consulted at the first trading day will not be considered for the rest of the simulation and then they will remain isolated in the network, with a zero degree. So you have to find these consultants who do not participate to market dynamics and drop their number from the total number of nodes. If we find only such consultants with zero degree, the graph is totally connected and, in the language of network theory, a unique *giant component* has arisen[30]. Since the time horizon is very large, the probability of $RAND \to RAT$ transitions during the last days is not actually zero: some random investors could observe so many trading days with negative profits that they could decide to call a consultant; so there is the chance that an analyst without clients at the beginning of the simulation is called by some new rational traders. This explains the last little decrease in the number of connected components, which finally falls to one.

Beside this marginal effect due to our specific transition mechanism, the graph always converges towards the giant component and this is just what we expect from a network which entails information sharing. The convergence time could be estimated running more simulations and computing the first trading day at which the *giant component* appears, but this is not so relevant.

The interesting question is that the network naturally reaches a sort of steady state where all agents have at least one acquaintance. In this state any kind of news or expectations can reach every agent: indeed there is always a path on the network which joins two different agents. The network then evolves guaranteeing the highest level of efficiency in terms of information spreading and and network effects and this is just what

[30] See Chapter 5 and 6 of *Dynamical Processes on Complex Networks* (2008)

we desired from our model, although it naturally emerges from an algorithm which looks at specific market variables on the micro scale (i.e. the daily profits of investors).

Average clustering coefficient A very important property to consider is the *average clustering coefficient* in the graph, which is defined as

$$< C >= \frac{1}{M} \sum_{i=1}^{M} C(i)$$

where $C(i)$ is the clustering coefficient of node/investor which is the ratio of the number of links between the neighbors of i and the maximum number of such links. If the degree of node i is k_i and if these nodes have e_i edges between them, we have

$$C(i) = \frac{e_i}{k_i(k_i - 1)/2}$$

where it is worth remarking that this measure of clustering has a meaning for $k_i > 1$; if $k_i \leq 1$ we define $C(i) \equiv 0$. The property of clustering refers to the tendency of having neighbors which are also neighbors among them: a high clustering coefficient implies the formation of cliques in the neighborhood of any given node. Typically real complex networks exhibit high clustering coefficients.

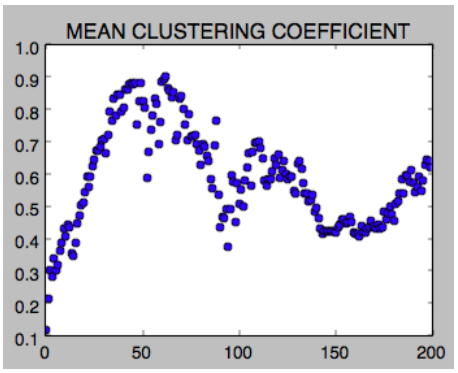


Figure 37: Time evolution of mean clustering coefficient on a horizon $T = 200$, with $seed = 2$.

You can observe how the average clustering coefficient increases during trading days and this is an effect which was strongly expected, partly due to the saturation of the network (the number of edges tends always to rise). As time goes by, the probability that two of your acquaintances can know each other is higher. *The growth of clustering effect in the informational network can be seen as a spread of information among agents. More and more traders tend to be connected each other, so the possibility of sharing information about the market increase consistently: the informational set available to each investor augments and learning through these social interactions is more efficient.* The delicate question, as future discussion, is if the higher clustering in the network can actually help investors to get better profits or not.

However, after many trading days (approximately 50), the average clustering coefficient reaches a maximum level before to start to decrease. The network approaches a state of very high clustering (close to the condition of totally clustered graph for $< C >= 1$) but then there is a reversing effect and average clustering coefficient dynamics displays periodic fluctuations. The evolution algorithm, which is based upon

daily profits, first leads to the highest state of clustering effect, also due to the saturation in the network. Once the number of edges can no more increase too much, the algorithm establishes periodic oscillations. The latter fluctuations occur around some definite value of $< C >$, which represent in some sense a steady value. However the informational network is an out of equilibrium system and we cannot have an actual limit for $< C >$.

Degree assortativity The last global network property which is investigated in this work is the *degree assortativity coefficient*, which defines the correlation between two nodes on the basis of their degree. Mixing patterns are very important in complex networks because agents/nodes can connect to each other on the basis of some of their property: if nodes are more likely connected to other nodes with similar properties we have an *assortative mixing*, while if the tendency is the connection among nodes with different properties or attributes we are talking about *disassortative mixing*. Testing assortativity with respect to the degree means looking for the correlation function $P(k'|k)$, investigating which is the probability that a node with degree k could link to another node with link k'. The assortativity coefficient is in general related to the concept of Pearson coefficient: here we show the formula used for the degree assortativity coefficient

$$r = \frac{\frac{1}{E}\sum_e j_e k_e - [\frac{1}{2E}\sum_e (j_e + k_e)]^2}{\frac{1}{2E}\sum_e (j_e^2 + k_e^2) - [\frac{1}{2E}\sum_e (j_e + k_e)]^2}$$

where j_e and k_e are degree of the extremities of edge e and E is the total number of edges. Now we follow the time evolution of this variable in the simulation, with a single seed and with a very large simulation horizon.

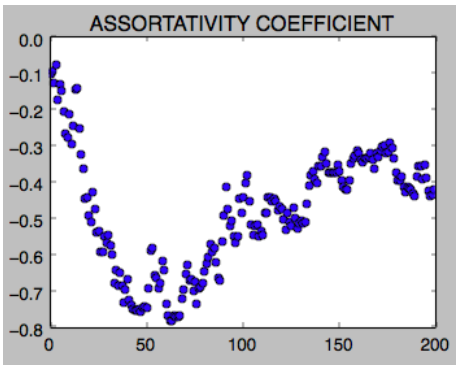


Figure 38: Time evolution of degree assortativity coefficient on a horizon $T = 200$, with $seed = 2$.

The first essential result is that our informational network presents *disassortative mixing* in terms of degrees. This is a strong and not trivial property since all the social network usually exhibit instead assortative mixing. How can we interpret this disassortativity of our informational network? First of all degree disassortivity means that highly connected nodes tend to connect to nodes with low degree, which corresponds to say that hubs tend to remain isolated among them. *In an informational network regarding news and expectations about stock market, the disassortativity can be a great advantage, since the nodes with so many acquaintances become a sort of control spot of the entire information. Two hubs which communicate are two agents who contrast each other in spreading information. So there could be interferences that would*

lead to a chaotic and contradictory flow of information and market dynamics avoids this possibility lowering the degree assortatvity level.

What we can see from the plot is that the disassortativity level tends to reach a minimum level and then slowly starts to settle on a sort of steady value, maintaining some statistical fluctuations over time. As in the case of the average clustering coefficient we could make a fit to get both such estimates, but this work would not add any more to the general understanding of the problem. The only thing that matters in our conclusions is that the network naturally develops an optimal value of degree disassortativity and this once again does no stem from some a priori mathematical law in the model.

9 Conclusions

Financial markets are complex systems in which agents expectations play a fundamental role in the emerging properties of stock prices. Standard tools offered by the Efficient Market Hypothesis do not manage to explain many of these features because they consider the market as an efficient system which naturally tends towards an equilibrium state. Instead stock market dynamics is always out of equilibrium because traders continuously change their strategies in relation to the available information. An Agent-Based Model is the way to include in our market agents heterogeneity, real trading system and real interactions among agents. In particular the great novelty of this work, with respect to previous ABM market, is the introduction of an informational network which mimics the set of social interactions among agents. The network evolves together with the simulation on the basis of daily profits realized by traders, so becoming an evolving complex entity which reacts to the current state of the market. This approach joins the Agent-Based Modeling flexibility with the tools of Network Analysis and offers two parallel ways to explore complex interactions and emerging phenomena.

Our simulated stock market is based on the difference between random traders, who operate without the help of a financial advisor, and rational investors, who try to get the portfolio goal proposed by their consultant. This contrast is not static because we have defined some control parameters which allow transitions between the two classes of investors, also taking into account network effects in the market. From the other side we are interested to observe which are the emerging statistical properties of stock prices dynamics. Therefore this ABM is also an attempt to jointly investigate some topics both related to behavioral and quantitative finance, which are typically two fields far from each other in the standard approach.

Qualitative results of Chapter 5 have explicitly shown which is the effect of all control parameters, facing complexity in financial markets from a theoretical perspective. We have seen how a market with a high fraction of random traders is more volatile, since this implies a wider space for expectational models. The consequence is that varying a control parameter that produces more actual transitions towards the random class, the market will become more volatile. Mean fluctuations in stock prices was controlled through a simple calibration of parameter $\triangle n_{trade}$; the latter operation is very elegant because we are able to reduce parameter space and guarantee realistic features for our simulated market. Among all kind of transitions we have proved that herd effect can trigger bull or bear trends in intra-day prices: irrational phenomena are so controlled by a specific parameter and can produce also bubbles or crashes on a longer time horizon. Another interesting theme is the role of parameter λ_{test} as a sort of learning rate for rational agents: asking for a new consultancy is the way to update the trading strategy, but what actually matter in this process are network effects. Agents can influence each other through social interactions modeled by the links in the informational network: this is the main difference with respect to the most famous ABM market, i.e. the SFI stock market. The decision making process for traders is inevitably affected by behavior and decisions of their acquaintances, which correspond to their neighbors in the network; thanks to this work we finally have defined the quantitative tools to monitor such behavioral effects. Clearly a too high learning rate does not ensure an efficient learning mechanism, but a too low learning rate does not provide enough time to get a good proxy of the optimal portfolio. Agents have to adapt their strategy, but also must have the proper time to construct the portfolio goal and test it in the market.

The investment plan for the intervention of arbitrageurs in the auction market responds to many of the obstacles which arise in a complex system, as illustrated in Chapter 7. The double trading auction system and the limited number of actual investors make difficult the work of arbitrageurs; through the introduction of *ad hoc* counterparts the alignment process becomes robust for each seed and then we have the possibility to run simulations with stock price dynamics equal to the real one. This is the essential step to use the

ABM for policy making applications.

Chapter 6 was useful to fix a precise control parameters setting such to avoid possible transition or learning mechanisms. In this way we can perform quantitative analysis in a basic framework without many complex micro behaviors that alter the results. Among all the statistical properties of stock prices dynamics, in Chapter 8 we have considered in particular the distribution of log returns and the autocorrelations in returns series. The distribution of all asset returns displays significant skewness and kurtosis, as in the case of real stock returns. Rare events as bubbles or crashes are actually observed in this ABM, so standard assumption of log normal returns is rejected. Moreover significant autocorrelations appear in the series of log returns, as the variance ratios have proved; this fact is a sign of predictability in stock prices and the question is if traders can actually exploit such level of predictability. However our simulation has naturally displayed all the features which commonly characterize a real market. Our ABM is therefore very realistic and this supports all the rules that define the stock market, both for agents behavior and for the trading system.

Next part of the analysis of Chapter 8 regards the comparison between the totally simulated market and the scenario with arbitrage operations. In particular we have explored the distribution of mean profits for the traders in both situations. The results show that only some rational investors can beat the market and realize significant profits. However under real market dynamics, gain chances are reduced, and, on the contrary the few *outlier* traders can achieve even better mean profits. Using some Machine Learning principles we have seen how, in the aligned market, the better forecasting methods are those which are an halfway between too complex and too simple methods, so exploiting just the few degrees of predictability in the market. In contrast to that, traders who actually manage to realize significant profits are those who follow the most simple trading strategy. Trying to exploit the degree of predictability (which exists) in a market which follows real dynamics is very difficult. So methods who adopt technical analysis loose their power when we pass to actual trading operations. The investment rules for the traders are not changed with respect to the totally free simulated market, but now traders are competing looking at real stock prices: complex interactions have destroyed in this case the chance to gain from technical analysis. The most trivial trading strategy ensures the best profits because is able to well generalize her forecasts. The latter results is very important because it explicitly shows how much is difficult to build a profitable trading strategy in a real market.

The last part of the work is focused on the properties of the informational network, starting from her degree distribution. We have observed that hubs naturally emerge, without any specific algorithm which takes into account some local network measures. This is also valid for all the other global properties of the network, as for the average clustering coefficient and for the degree assortativity. In particular the very interesting feature is the existence of a sort out of equilibrium steady state for the network. The latter definition may seem an intrinsic contradiction, but expresses the idea of an informational network which reflects a non equilibrium system such as a stock market and so cannot converge towards a state with fixed local properties. Anyway, after an initial period where the density of the network continue to increase, we can observe that many global network measures (as the scale free exponent, the average clustering coefficient and the degree assortativity) display fluctuations around some ideal steady value. The fluctuations are the consequence of the stochastic nature of the market and of the social interactions among agents (which are their self based upon variables that depend on market state, as the financial profits). *So only the mean time values of such global properties reach a sort of equilibrium state.* A conclusion of this kind should always be analyzed thinking about statistical non equilibrium systems, since a stock market and its related informational network are just systems of this kind.

9.1 Future works starting from this ABM

This work has shown some new ideas in the field of agent-based modeling for financial markets. However there is so much to do if we want to exploit this ABM both from the theoretical perspective and from real policy making applications.

The discussion of Chapter 6 just presents qualitative, although useful, results. Actual calibration is made only for $\triangle n_{trade}$, while the other control parameters are simply fixed such to avoid transitions or learning phenomena. A more quantitative exploration of the entire parameters space, using for example Bayesian inference techniques, is required for future works. Running the simulation for many seeds and collecting average results would ensure to well calibrate all the control parameters in a global scenario where all behavioral phenomena are allowed. Each phenomenon can be modified or extended in other future ABM markets, investigating for example the quantitative effect of herd behavior on the appearance of bubbles and crashes.

Data analysis performed in Chapter 8 can be repeated in a scenario which considers all kind of transitions or learning possibilities: this would be the actual bridge between behavioral and quantitative finance. Herd behavior can, for instance, modify profits of the traders or affect the performance of consultants. Moreover the most interesting topic, which our work had to omit for lack of time, is exploring the scenario in which few rational investors can update their trading strategy in a market mainly formed by noise traders. What is the optimal λ_{test} value which guarantees enough time to build a robust portfolio, but at the same time ensures an effective update on new emerging data? Of course we can try to answer this question also in the aligned market thanks to arbitrage operations. This is the first real application for this ABM. Indeed a portfolio manager or simply a trader who adopts some trading strategy can be interested to individuate the optimal frequency to update her strategy, maybe asking for new indications from her financial advisor or simply changing her personal forecast methods. The latter frequency is also related to the fees required by any financial advisor who manages trader's portfolio; so we can also think to p_{info} the percentage on your financial profits required by your portfolio manager. The ABM can therefore help a trader to reduce costs regarding financial consultancy.

Another interesting research line that can be analyze with greater attention looks at our informational network. Clearly other ABMs could modify some of the emerging network rules in this work, but what is surprising is the presence of many unexpected emerging features which are completely independent by some a priori network model. Such features can be brilliantly studied in the context of complex networks, exploring with more quantitative tools all the dynamical phenomena which regards information spread. Using more seeds and always maintaining a very large simulation horizon we can also fit the general dynamical pattern for many of the existing global network measures: for example we could find which are the asymptotic (i.e. mean time) values for the fluctuations of average clustering coefficient or degree assortativity. Another goal is looking for correlations among local network properties of agents/nodes with their corresponding model-specific variables; the example is obviously represented by the relationship between degree and mean profits of our investors, but many other properties (betweenness or closeness) could be studied in other works and on other ABMs. Maybe this work can encourage the idea to combining Network Analysis with Agent-Based Modeling, not only in the field of economics or finance, but also in other scientific contexts regarding complex systems.

9.2 Real trading applications and policy making tests with the aligned market

The greatest real application of this work involves the chance to test any forecasting model and trading strategy in a real complex market. The actions of arbitrageurs in all the books ensure that your simulated market is aligned to real data. So you can imagine to build a software platform with all tradable securities in

a given market (and also on many related markets): if you have access, through proper permissions, to live real data, you can apply this ABM to observe which would be the future performance of any type of trading strategy. Indeed if real time prices of all securities are linked and reflected in your agent-based market, thanks to the arbitrageurs, you have the possibility to look ahead and see which will be the daily profits obtained with a particular trading operation. It is suffice to run the simulation and observe the daily profits of traders who have used different strategies. The outcomes of your simulation can make more realistic if you monitor real data and use in your ABM the same number of traders involved in the real market. The latter operation requires a very high computational power for your platform because you have to handle a great amount of data in live time and running your model with so many agents who perfectly mimic each existing trader. Of course a certain seed in the simulation will correspond to a particular order in which traders interact in the auction: this generates specific daily profits and specific performances for trading strategies. The solution is launching parallel simulations and take a sort of average over several random events to estimate what is the mean outcome in reality. Thanks to such method you can discard those strategies who have bad performed at the end of the simulated day and adapt your real trading operations in live time.

From the policy making side we have also many interesting applications offered by this work. Indeed beside the possibility to maximize your profits as a trader, you can test the future evolution of the market in response to some consistent operations or new policies. For example if you work for the Central Bank, a new policy of intervention on the market can be tested thanks to this ABM. What will be the future market state if the Central Bank decides to purchase a very big amount of assets, so rising aggregate money supply? And what about the opposite policy? In this case the forecasts of your ABM can be calibrated taking into account specific macroeconomic indicators or new information available to the agents. Moreover the rules of the market can also be modified to check what is the result on asset price dynamics. An important example is the consideration of specific limitations for short-selling or leveraging operations. If there are some set of rules or constraints which make more stable the market, this ABM can try to find them.

This work, starting from other famous ABMs, is a new interesting tool in the world of financial forecasting because it deals with actual problems of a real complex market. Asset management businesses, banks and public policy institutions could adopt a software that supports an ABM of this kind in order to monitor the market, get reliable forecasts and set their strategies in live time.

10 Appendixes

10.1 Student's t-statistic

In statistical analysis a *t-statistic* wants to compare the mean value of a sample to some theoretical value which represents the *null hypothesis*, using the standard error of the sample mean. The *t-test* estimator is distributed according to the *Student's t-distribution*, which is similar to the gaussian one, but presents fatter tails.

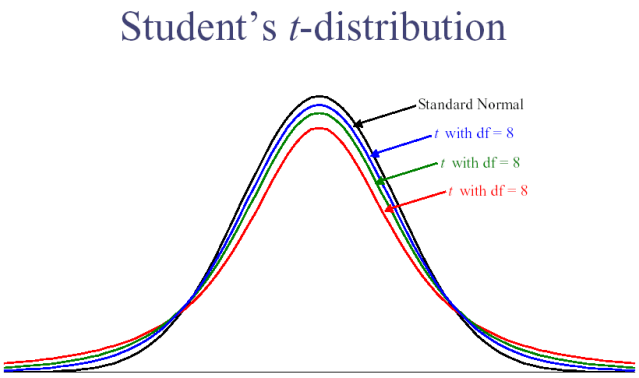


Figure 39: Student t distribution.

The use of *Student's t-test* is related to those situations in which we have a little statistical sample as in the case of repeated but limited simulations. Since we used to apply such test on the average estimators over many simulations, we can write

$$t = |\frac{\overline{estimator} - testValue}{error(\overline{estimator})}|$$

Then, using this standardized variable, we fix a significance level (usually equal to 5%) through which we have a threshold for rejecting the *null hypothesis*. In particular, as you can find in any book of basic statistics, if $t > 1.68$ you abort the theoretical hypothesis you are testing, otherwise you can accept it.

10.2 Some basic Machine Learning principles

Machine Learning theory comprises a series of mathematical theorems and statistical tools that explain how to extract information from available data[31]. When we have a data set to analyze, but we do not know anything about the problem, the question is to find a relationship which can fit the data yet available, but also can generalize and make forecasts on other new data. The learning problem can be presented with the following scheme

[31] A full presentation and description of Machine Learning approach can be found in *Learning from data*, Abu-Mostafa et al. (2012)

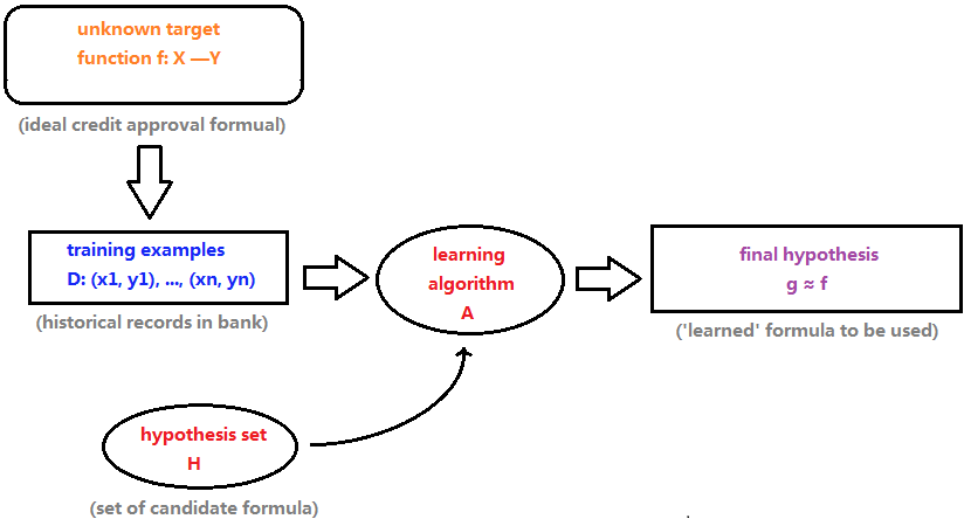


Figure 40: Machine Learning components.

When you are facing some data set, there is always a function which would allow you to make perfect forecasts about that kind of data: this unknown function is called *target function* and if you found it you would be able to make exact predictions about new data from the same problem, i.e. you would reach an absolute forecasting accuracy. All Machine Learning problems require to get a proxy for the unknown *target function* through the available data set. The data you use to get your *final hypothesis function* are defined as *training set*. The key point to reach such goal is what kind of *learning algorithm* you choose to implement in your program. The *learning algorithm* starts from an *hypothesis set* of candidate functions and select the *final hypothesis function* on the basis of the training sample; in our work the learning algorithm used is the simple linear regression method.

From a theoretical and practical point of view every learning problem entails a *trade-off* between two different kinds of error, both defined on a possible candidate function h. The first one is the *in-sample error*

$$E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N} e(h(x_n), f(x_n))$$

where N is the number of elements in the training set and $e(h(x_n), f(x_n))$ is the error measure (i.e. the metrics) used to evaluate the difference between the forecast with the candidate function h and the perfect forecast with the target function f. Of course we do not know the form of f but, if our learning problem is *supervised* (as in this work), we always know $f(x_n)$. A common point for learning algorithms is to select the *final hypothesis function* g such to minimize the *in-sample-error*, simply fitting your data. Our consultants made just this operation using as error measure the square of the difference between the two predictions, i.e. $e(h(x_n), f(x_n)) = (h(x_n) - f(x_n))^2$: this corresponds to the OLS method. The second kind of error is the *out-of-sample error*

$$E_{out}(h) = \mathbb{P}(h(x) \neq f(x))$$

which states the general probability of having constructed an hypothesis function which does not coincide with the target function. Therefore, while *in-sample error* quantifies how much our function can fit the training set, the *out-of-sample error* defines the probability of making an error in the generalization process. More you fit well the available data and less you can generalize your forecasts to new data: that is the essential *trade-off* in Machine Learning. The *learning algorithm* aims to look at such trade-off and many techniques are conceived to achieve a good *in-sample-error* (i.e. enough small) without a too high *out-of-sample error*.

References

[1] Blake LeBaron (2002), *Building the Santa Fe Artificial Stock Marke*t.

[2] Blake LeBaron (1999), *Agent-based computational finance: Suggested readings and early research.* Journal of Economic Dynamics & Control.

[3] Andrea Beltratti, Sergio Margarita, Pietro Terna (1996) *Neural networks for economic and financial modeling.* International Thompson Computer Press.

[4] W. B. Arthur, S. N. Durlauf, D. Lane (1997) *The economy as an evolving complex system II.* ABP.

[5] Magda Fontana, Pietro Terna (2015) *From Agen-Based Models to Network Analysis (and return): The Policy Making Perspective.* Department of Economics and Statistics Cognetti de Martiis.

[6] A. Kirman (2011) *Complex Economics: Individual and collective rationality.*

[7] John Y. Campbell, Luis M. Viceira (2001) *Strategic Asset Allocation: Portfolio Choice for Long-Term Investors.*

[8] Yaser S. Abu-Mostafa, Malik magdon-Ismail, Hsuan-Tien Lin (2012) *Learning from data.* AML-book.com

[9] Marno Verbeek (2004) *A guide to modern Econometrics.* John Wiley & Sons.

[10] Riccardo Boero, Matteo Morini, Michele Sonnessa, Pietro Terna (2015) *Agent-based Models of the Economy: From Theories to Applications.* Palgrave Macmillan.

[11] Frank J. Fabozzi, Sergio M. Focardi and Petterr N. Kolm (2006) *Financial Modeling of the Equity Market: From CAPM to cointegration.* John Wiley & Sons.

[12] Reka Albert, Albert-Laszlo Barabasi (2002) *Statistical mechanics of complex networks.* Review of Modern Physics.

[13] Alain Barrat, Marc Barthelemy, Alessandro Vespignani (2008) *Dynamical Processes on Complex Networks.* Cambridge University Press.