University of Padova

Department of Mathematics

Master Thesis in Data Science

PyDiM: a new Python library for Diffusion Model Analysis

Supervisor
Mariangela Guidolin
University of Padova

Master Candidate
Carlo De Dominicis

Academic Year
2022-2023

Dedication.

Abstract

Innovation diffusion analysis is an important tool for understanding how new ideas, products, and technologies spread through time. While there are existing packages in R for conducting this kind of analysis, there is a growing demand for similar tools in other data science-oriented programming languages, particularly Python. Python is a popular language for data analysis and machine learning, with a large and active community of users and developers. Having an innovation diffusion analysis library in Python would allow researchers and practitioners to leverage the language's strengths in data processing, visualization, and modeling. It would also provide a more accessible and user-friendly option for those who are more comfortable with Python than with R. This new Python library offers a comprehensive set of tools for conducting innovation diffusion analysis, including data preparation, visualization, and modeling. It is designed to be easy to use, with clear and intuitive functions and documentation. Additionally, it offers flexibility and customization options to meet the needs of a wide range of users. Overall, this new Python library for innovation diffusion analysis fills a gap in the current landscape of data analysis tools, providing a valuable option for those who prefer Python and opening up new opportunities for research and innovation in this important area

Contents

Listing of figures

Listing of tables

Listing of acronyms

BM . . . . . Bass Model

GBM . . . . . Generalized Bass Model

GGM . . . . . Guseo-Guidolin Model

UCRCD . . . . . Unbalanced Competition Regime Change Diachronic

LOC . . . . . Lines Of Code

OS . . . . . Operating System

1 Introduction

The concept of innovation plays a central role in modern society, in particular, after the 20Th century, it became very popular as the definition of technological innovation by people's common understanding but also from the literature itself. However, the etymology and the history of this concept is much broader. Innovation is generally understood as commercialized innovation because of the close relationship between technology and firms' marketing, but other types of innovation are rarely discussed. To a certain extent, every individual is innovative: artists, scientists, and so on. In his project series, starting from [1], Godin tries to bring to light, through various hypotheses, the possible genealogical history that brought the actual concept of innovation as intended nowadays. The first appearances are tracked back to the thirteenth century with "*novation*", with the means of renewing (an obligation). Since then, this word has been rarely used, since "create" and "invent" were preferred words for man's productive power and creative ability. Some use of the term as such are being cited from very few individuals, some relevant examples are N. Machiavelli in *The Prince* (1513) and F. Bacon in *Of Innovation* (1625). It seems that the word in se appeared with negative connotations in the Middle age since intended as "*change*" in a time in which traditions were the central point for politics and religion. Just in the late 20th century, passing through sociological theories about "invention" [2], the concept got the meaning of "creativity process" or "newness", as a consequence of two recurrent sequential steps: imitation and invention. To the point that "innovation" and "invention" got often interchanged, Figure 1.1 shows the trends of the two words in the last two centuries.

Figure 1.1: Frequency in the use of the words "invention" and "innovation".

The theory about the diffusion of innovation can be traced back to the early 20th century with Schumpeter's Theory of Innovation, he explains that the process of *creative destruction*, referred to the technological change, in a free market consists of three parts: invention (conceiving a new idea or process), innovation (arranging the economic requirements for implementing an invention), and diffusion (whereby people observing the new discovery adopt or imitate it). However, it became popular after the first publication of Diffusion of Innovations [3]. In his book, Rogers defined the diffusion of innovation as the result of the spread of information over time and through specific channels among a population of individuals. It is perhaps possible to determine the key factors influencing the diffusion process of these new ideas: the innovation itself, the communication channel, the time window in which the innovation fulfills its life cycle, and the adopter's social system.
Rogers gave the following definitions and characteristics to the 4 key factors.
Innovation. "An innovation is an idea, practice or project that is perceived as new by an individual or other unit of adoption", this means that the newness of an idea, practice or object is given by the adopter's perception of it more than its effective being new if the idea seems new to the individual than it is an innovation. Also, the adoption rate of innovations can be a lot different from each other, depending on the characteristic of the innovations, which are:

1. *Relative advantage*. It is the degree to which an innovation is perceived as better than the idea it exchange.
2. *Compatibility*.It is the degree of perceived consistency of the innovation with the existing values, potential past experience, and needs of the eventual adopters.
3. *Complexity*. It refers to the perceived difficulty to understand and use the innovation by potential adopters, an higher complexity can bring a slower adoption.
4. *Trialability*. It represents the degree to which an innovation may be experimented with on a limited basis.

5. *Observability*. This is the last one and represents the degree of visibility of the innovation's results.

Communication Channels. For Rogers, communication is "a process in which participants create and share information with one another in order to reach a mutual understanding", and yet, "A channel is the means by which a message gets from the source to the receiver". With that, he states that diffusion is a specific kind of communication that includes these elements: an innovation, two individuals or other units of adoption, and a communication channel. A communication channel can have a different nature, depending on the relation with the social network, two main examples are Mass media which are classified as external sources, and interpersonal communication (which will be also referred to as word-of-mouth) classified as internal sources as well as the most representative form of interpersonal communication.
Time. Rarely an adoption happens instantaneously, the passage of time is necessary for a diffusion process since the spread of communication itself relies on time and also on the innovativeness of an individual (as earlier to later adopter) compared to the other members of the social system.
Social system. Rogers defines this last element as "a set of interrelated units engaged in joint problem solving to accomplish a common goal", he also states this is the main criterion for categorizing the adopters since the nature of the social system affects individuals' innovativeness. This is, in general, the result of internal and external social influence.

The history of innovation diffusion modeling finds its origin between the 19th and early 20th century with the logistic model proposed by [4], in which he study demographic growth considering a maximum value for the population, his study will be discovered just in 1920. In a similar way, in [5] the authors proposed a logistic function to model the growth in bacterial cultures. Also, in [6], Mansfield used a logistic model to justify the spread of new techniques between firms considering the spread of innovation just by the imitation point of view. It is just in 1969, with the publication of the Bass Model (Bass 1969), that the literature regarding innovation diffusion modeling started its main growth. In his work, following Rogers' wave, Bass contributed the mathematical ideas of the concepts with the most successful modifications of the logistic model. Thanks to the profound impact it brought, many new models and literature has emerged, the most as an extension of the former, to cover an ever-increasing complexity of new product growth given by factors such as new and fastest communication channels, mixed market trends caused by globalization, increased competition, and so on.

The overall literature about innovation diffusion keeps increasing over the years [7], and the theory is applied to further and further research areas: from emerging economies (i.e. electronic

communications, services, pharmaceutical industry, etc...) to individual decision-making studies, important to enterprise management decision; but also, studies regarding the diffusion over two dimensions of space and time, to understand factors such as multi-country diffusion; or empirical researches about innovation diffusion, thanks to the decreasing difficulty in acquiring data given by the development of networks in which companies store datasets and customer relationship management systems. These reasons lead us to the definition of this library, which aim is to start filling a gap in the landscape of these kinds of data analysis tools.

PyDiM was born as the Python version of R's package DIMORA, to provide an extra tool for ever-increasing professional figures such as data scientists and analysts, helping them in data analysis without the need to change programming languages. Thanks to the wide range of tools available, the library can speed up work and be a viable alternative for those who prefer Python to R, since the former is already a widely used language for data analysis and machine learning and can rely on a large community of users and developers. In addition, our library can also be of great help to novices who are approaching the field of data analysis and modeling, speeding up their learning process.

The next chapters of this thesis work will present the implemented models and their mathematical features (Chapter 2), a technical explanation of the differences between Python and R, highlighting the criticism encountered during the development and the differences in the running time between the two coding languages (Chapter 3), some analysis of real-world data, with the aim of show the functioning of the models on topics having different nature (Chapter 4). Then, an overall comment on the work and the future perspectives.

2

Diffusion Models

The literature about diffusion models is becoming quite vast in the last 50 years. Its history finds roots in the form of logistic models used in biology to study epidemic spread, but also in fields such as sociology, chemistry, economics, and so on. In recent years it finds its uses in almost all those fields in which there is a need to study diffusion processes, which are present in many real-life stochastic systems between the heavy uses we can find physics, chemistry, biology, finance, sociology, economics, and marketing. The interdisciplinary nature of diffusion modeling, and the opportunities it offers make it worthy to be a very hot topic nowadays.
As mentioned, in this work we focused on implementing some of the models deployed in the context of marketing science which aims to study the diffusion of innovation, starting from the Bass Model [8] in Section 2.1, which defined the baseline of the literature in this field, and then presenting three generalizations of the former: the Generalized Bass Model [9] in Section 2.2 which integrates traditional economic variables not considered in the formulation of the BM, but still retaining the properties of the former; the Guseo-Guidolin Model [10] in Section 2.3 which considers the communication process of the innovation as a key factor influencing the market capacity, referred as *market potential*; lastly, in Section 2.4 will be presented a multivariate approach for competition between products, the Unbalanced Competition Regime Change Diachronic Model [11] which relies on BM to analyze couples of trends to find parameters and describe the diffusion of them and detect the effect of the competition.

2.1 Bass Model

The Bass model [8], originally developed in the context of marketing science, describes the life cycle of innovation by capturing the typical phases involved in this process: launch, growth, maturity, and decline. Its purpose was to model the growth over time of a new product as a result of the purchases by two classifications of adopters: the innovators, and the imitators which differ from one another by the degree of interpersonal communication occurring in the adoption phase. The former describes the portion of adopters influenced by external (to the social network) information sources, such as mass media or advertisements. The latter, as the name suggests, describes the portion influenced by the higher level of social interaction, that is, by internal information, often referred to as imitation or word-of-mouth. Although the role of the innovator was already present in the literature (Rogers defines these as early adopters), the Bass Model is the first model that accounts for their presence. Differently from the logistic approach proposed by Mansfield [6], Bass took into account the communication efforts realized by firms considering, at the launch of the product, a constant level of innovators buying the product. The Bass Model consists of a simple first-order differential equation:

$$z'(t) = \left[p + q\frac{z(t)}{m}\right][m - z(t)], \;\; t > 0 \tag{2.1}$$

In Equation 2.1, the variation of adoption over time, $z\prime(t)$, is proportional to the number of consumers who have not adopted yet, named *residual market*, $m - z(t)$, where m is the market potential (or size), and $z(t)$ is the cumulative number of adoptions at time t, i.e. $z(t) = \sum_{i=0}^{t} z\prime(i)$. Bass assumed the market potential to be constant over the entire life cycle and represents the maximum number of possible adoptions. The factor $p + q\frac{z(t)}{m}$ influencing the residual market represents the likelihood of purchase by a new adopter at time t, it is perhaps in this factor that the adopters play the key role: p is called innovation coefficient and represent the effect of exogenous variables, that is the external influence brought by external sources of information (e.g. mass media and advertising), q is the imitation coefficient and represents the portion of adopters which rely on internal influences to adopt the innovation, this influence is given by the ratio $\frac{z(t)}{m}$ representing the capacity of the market in a given time t. The component $q\frac{z(t)}{m}$ defines the spreading of internal information among imitator adopters, it represents the *word-of-mouth*. It is important to notice that innovators adopt at the very beginning of the process, while imitators are assumed to adopt in a second stage (Figure 2.1), since $q\frac{z(t)}{m} = 0$ when $z(t) = 0$, named, there are no cumulative sales when the product enters the market at

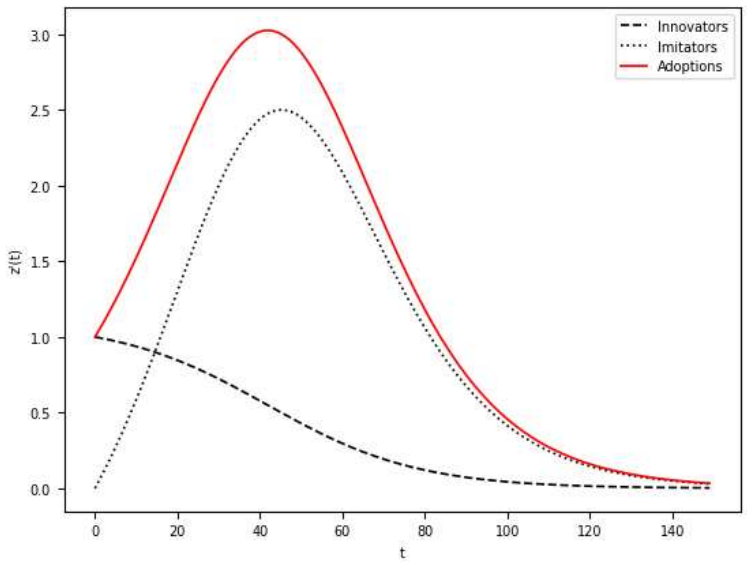

Figure 2.1: New adoptions according to the BM

time $t = 0$, and the equation is therefore reduced to $z'(0) = pm$.
Equation 2.1 can be rearranged as a duration model for survival analysis. By posing $\frac{z(t)}{m} = y(t)$ so that the model can be re-written as

$$y'(t) = [p + qy(t)][1 - y(t)], \quad t > 0 \tag{2.2}$$

Then, by rearranging Equation 2.2 it is possible to express the model as a hazard function of the form

$$\frac{y(t)'}{1 - y(t)} = p + qy(t), \quad t > 0 \tag{2.3}$$

Equation 2.3 can now be seen as the hazard rate, the conditional probability, of adoption at time t, where $y'(t)$ is the density function, $1 - y(t)$ is the survival function with $y(t)$ as a cumulative distribution function.

2.1.1 Closed-From solution

By applying the basic definitions of survival analysis and some transformations to Equation 2.2 it is possible to derive the closed-form equation of the Bass Model for product adoption, that

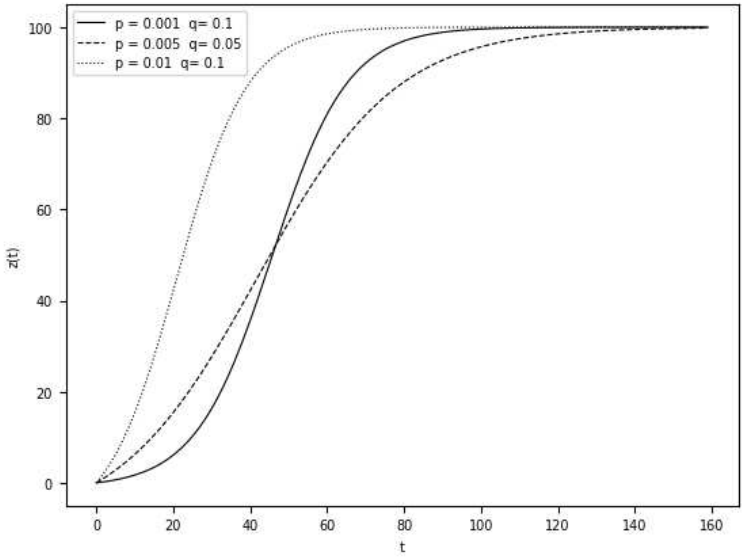

Figure 2.2: Cumulative adoption on different magnitudes with $m = 100$

is

$$y(t) = \frac{1 - e^{-(p+q)t}}{1 + \frac{q}{p}e^{-(p+q)t}}, \ t > 0 \tag{2.4}$$

The function $y(t)$ in Equation 2.4 takes values in $0 < y(t) < 1$, and directly depends on p and q to determine the speed of growth until saturation. Because $z(t) = my(t)$, the closed form described in Equation 2.4 can be re-written as

$$z(t) = my(t) = m\frac{1 - e^{-(p+q)t}}{1 + \frac{q}{p}e^{-(p+q)t}}, \ t > 0. \tag{2.5}$$

Just as in the previous equation, in Equation 2.5 parameters p and q act on the speed of diffusion, while the market potential m is, again, a scale parameter that allows for modeling the diffusion process in absolute terms. Typical values for the innovation and imitation coefficients are:

- Between 0.01 and 0.03 for p, on average 0.03
- Between 0.3 and 0.5 for q, on average 0.38

The higher the values the higher the fastest the adoption.
In Figure 2.2 we can see the effect of the coefficients on the cumulative adoption number, which

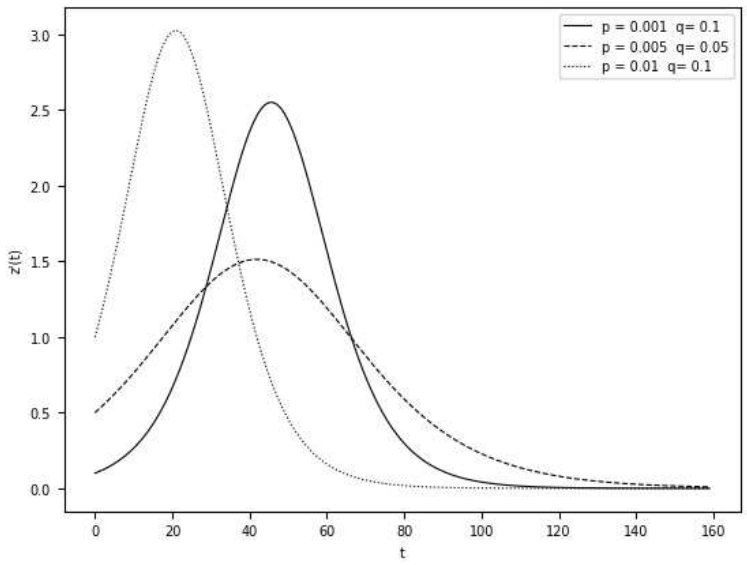

Figure 2.3: Instantaneous adoption on different magnitudes with $m = 100$

reaches the saturation of the market potential, m, with different speeds. The corresponding instantaneous adoptions, $z'(t)$, are defined as

$$z'(t) = m\frac{p(p+q)^2 e^{-(p+q)t}}{\left(p + qe^{-(p+q)t}\right)^2}, \ t > 0. \tag{2.6}$$

In Figure 2.3 we can see the instantaneous adoptions of Equation 2.6 on different coefficients magnitude, considering a market potential of 100. The figure highlights a positive start of the process given by the initialization $z'(0) = pm$, but also the presence of a maximum peak which indicates the maximum instantaneous expansion of the diffusion process at time

$$t^* = \frac{\log(q/p)}{p+q}, \tag{2.7}$$

and should indicate the moment of maturity of a product in its life cycle. The time t^* in which the process reaches its peak is described as follows And the instantaneous function takes the value

$$z(t^*) = \frac{m}{2} - \frac{p}{2q} \tag{2.8}$$

Taking into consideration the peak of a diffusion process is very important from a marketing perspective since, in strategic terms, it represents the moment before the decline of the life cycle

for which the firm can, for example, take action to try to change the trend on time.

2.2 Generalized Bass Model

The Generalized Bass Model [9] comes from the same F.M. Bass to overcome the lack of the Bass Model over the combination of contagion effect with traditional economic variables, such as price and marketing strategies. So they implemented a generalized version of the BM that, based on certain circumstances, includes turning points and irregularities in the penetration curve, has a closed-form solution, and reduces to the standard BM if there exist plausible regularity conditions for the decision variables. This model has been built seeking to preserve the fundamental character of its predecessor, in which p and q are permitted to vary over time, thus, the variation in adoption is given by

$$z'(t) = \left[p + q\frac{z(t)}{m}\right][m - z(t)]\,x(t)\,,\ \ t > 0,\ \ x(t) > 0. \tag{2.9}$$

In Equation 2.9, they define $x(t)$ as "*current marketing effort*", his role is to reflect the current effect of dynamic marketing variables on the number of adoptions at time t, which the authors define to be variation in pricing and advertising. As it is possible to notice in the last property listed earlier, the reduction of GBM to BM, is possible in case $x(t) = 1$ or $x(t) = c$, named, when no changes occur in the market during the time. In the other cases, it influences the speed of the diffusion process, if $0 < x(t) < 1$ the process slows down, whereas if $x(t) > 1$ it accelerates. In the next sections, we will see how the model gets his closed form, and how $x(t)$ can map some sort of "*carryover effects*" over the lags, possibly given by marketing variables and other forms of strategies, for which some examples of the implemented ones will be shown.

2.2.1 Closed-From solution

As mentioned, one interesting property of this model is to have a closed-form solution. Just as the Bass Model, it is achieved by studying the model under survival analysis assumptions. We have already seen how this generalization preserves the fundamentals of the BM by allowing p and q to be variable over time. This allows the following generalization on the hazard function described in Equation 2.3, recalling that we posed $\frac{z(t)}{m} = y(t)$, defined as

$$z'(t) = \left[p + q\frac{z(t)}{m}\right][m - z(t)]\,x(t)\,,\ \ t > 0,\ \ x(t) > 0. \tag{2.10}$$

Equation 2.10 shows how the right-hand side of the equation remains unchanged and, perhaps, the coefficients do not get affected by $x\,(t)$ which, instead, serves to shift the hazard function upward or downward depending on marketing variables, as the mentioned price and advertising, or other strategies that are not set to control the timing of a diffusion process. By skipping all the transformations on Equation 2.10, the resulting closed-form equation for the model is given by

$$y\,(t) = \frac{1 - e^{-(p+q)\int_0^t x(\tau)d\tau}}{1 + \frac{q}{p}e^{-(p+q)\int_0^t x(\tau)d\tau}},\ \ t > 0. \tag{2.11}$$

Thus it is possible to re-write this closed-form solution in terms of cumulative adoption

$$z\,(t) = my\,(t) = m\frac{1 - e^{-(p+q)\int_0^t x(\tau)d\tau}}{1 + \frac{q}{p}e^{-(p+q)\int_0^t x(\tau)d\tau}},\ \ t > 0. \tag{2.12}$$

Equation 2.12 can then be differentiated with respect to t to get the corresponding density function

$$z'\,(t) = m\frac{p\,(p+q)^2\,x\,(t)\,e^{-(p+q)\int_0^t x(\tau)d\tau}}{\left(pe^{(p+q)\int_0^t x(\tau)d\tau} + q\right)^2},\ \ t > 0. \tag{2.13}$$

2.2.2 Mapping carryover effects trough $x(t)$

As mentioned above, the *current marketing effort*, $x\,(t)$, introduced by this generalization of the Bass Model could be mapped to represent possible *carryover effects* on the lags of the trend. In [9] the authors proposes a possible solution to use this to model the trend over common marketing variables, such as price and advertising, with a function considering a percentage variation of the form:

$$x\,(t) = 1 + \beta_1\frac{Pr'\,(t)}{\mathrm{Pr}\,(t)} + \beta_2\frac{A'\,(t)}{A\,(t)},\ \ t > 0 \tag{2.14}$$

In Equation 2.14, $\mathrm{Pr}\,(t)$ and $A\,(t)$ represent price and advertising at time t, and $Pr'\,(t)$ and $A'\,(t)$ are, respectively, the rate of changes in price and advertising. In [12] the author proposes a set of "*structured shock*", mapped on $x\,(t)$, which aim is to generalize the use of the Bass model, thought to be just a strategic marketing analysis tool, to describe diffusion processes referring to different applicative context also a lot far from the cited one, such as epidemiological or technological migration phenomena. In the book, the author describes three kinds of shocks:

rectangular, *exponential*, and *mixed*. The three of them are being deployed in the GBM implementation of this library and will be described in the following. The first two kinds of shock formulas are presented as general cases which consider an indefinite number of shocks since in this way are being implemented in the library.

Rectangular Shock.

This is an easy representation of the function $x\left(t\right)$ as having a transient stationary behavior on the trend over a given period. This kind of shock is formalized as follows

$$x\left(t\right) = 1 + \sum_{i=1}^{n} c_i I_{t \geq a_i} I_{t \leq b_i}, \;\; t > 0, \;\; a_i < b_i \tag{2.15}$$

Equation 2.15 describes the most general case in which n rectangular shocks are considered, parameters a_i and b_i represent interval circumscribing the shock, c_i identifies the intensity of the local effect of the shock, be it positive or negative. The Indicator functions I contribute to the selective activation of the shock, being 1 if the event verifies at time t in the described domain, and 0 if not.
The corresponding integral for this function gets the form

$$\int_0^t x\left(\tau\right) d\tau = t + \sum_{i=1}^{n} c_i \left(t - a_i\right) I_{t \geq a_i} I_{t \leq b_i} + c_i \left(b_i - a_i\right) I_{t > b_i}, \;\; t > 0, \;\; a_i < b_i \tag{2.16}$$

Exponential Shock.

In some cases, the stationary behavior of the function $x\left(t\right)$ can be altered by intense instantaneous shocks characterized by subsequent uptakes happening at different speeds. This kind of shock is defined as

$$\left(t\right) = 1 + \sum_{i=1}^{n} c_i e^{b_i (t - a_i)} I_{t \geq a_i}, \tag{2.17}$$

In Equation 2.17, parameters a_i describes the timing of the shocks' outbreak; b_i describes the speed of the uptake for the shock to return to the stationarity of the function, this is typically negative, suggesting an exponentially decaying behavior; parameters c_i represents the intensity of the shock and can be positive or negative; and I is as described for the rectangular shock, the indicator function, it is 1 if the starting point of the shock, a_i is observed after the time t.

The respective integral of this function is given by

$$\int_0^t x(\tau)\, d\tau = t + \sum_{i=1}^{n} c_i \frac{1}{b_i}(e^{b_i(t-a_i)} - 1) I_{t \geq a_i}. \tag{2.18}$$

Mixed Shock.

In some cases, $x(t)$ should be able to consider local events having different natures and different causes. A mixed shock combines the structures of the shocks described above to model a more complex behavior of the trend during its lifetime. A mixed structure of this kind can be defined as follows

$$x(t) = 1 + c_1 e^{b_i(t-a_1)} I_{t \geq a_1} + c_2 I_{t \geq a_2} I_{t \leq b_2}, \tag{2.19}$$

Differently from the solely rectangular and exponential shocks, in the case of the mixed shock, it is considered just the simpler case that includes a single couple of shocks, the first exponential, and the second rectangular. That is because empirically, it has been observed that too complex functions tend to poorly estimate, endangering the renowned parsimony of the Bass and Generalized Bass models. The respective integral for this last function is given by

$$\int_0^t x(\tau)\, d\tau = t + c_1 \frac{1}{b_1}(e^{b_1(t-a_1)} - 1) I_{t \geq a_1} + c_2 (t - a_2) I_{t \geq a_2} I_{t \leq b_2} + c_2(b_2 - a_2) I_{t > b_2}. \tag{2.20}$$

2.3 Guseo-Guidolin Model

Another main feature that makes BM's structure so simple is the assumption that the market potential remains constant during the diffusion process, and considers a certain number of adopters from the very beginning of the process. This can be reasonable under particular circumstances, such as the diffusion of the next generations of already existing products already known by possible customers. In the other cases, literature [13] observes that theoretically there is no rationale for a constant population of adopters, so a dynamic nature of the market potential need to be considered in several situations, some examples can be: the so-called *incubation period* of innovative products, when it is still unclear if they will be a success or a failure; or the difficulty in the adoption of high complexity innovations. Both the previous cases highly rely on the effect of information spreading for the good outcome of the process, delineating a consequent dynamic behavior in the market potential. These are the considerations behind the

GGM [10], which considers the adoption as a two-stage process where the actual purchase/acquisition of the innovation is a direct consequence of the "awareness" spreading, named, the communication process, this same can vary over time and so indirectly reflect the same effect on the market potential. Before going deep into the presentation of the GGM, a general overview of this kind of BM generalization will be given. In general, these kinds of models consider a dynamic market potential $m(t)$ in the BM equation, such that

$$z'(t) = m(t)\left[p + q\frac{z(t)}{m(t)}\right][m(t) - z(t)] + m'(t)\frac{z(t)}{m(t)},\ t > 0 \tag{2.21}$$

Equation 2.21 simply adds to the instantaneous adoption described by the standard BM a factor $m'(t)\frac{z(t)}{m(t)}$ that is, a portion, given by the growth rate $\frac{z(t)}{m(t)}$, of the variation of the market potential $m'(t)$. This means that the variation $m'(t)$ will cause fluctuation on the instantaneous adoption, and this will be directly given by the size of market potential: the larger the $m(t)$ the most positive and reinforcing the effect on $z'(t)$, as a consequence, it gets a negative effect the smaller the $m(t)$, expressing the outcome of the adoption process as dependent from either the expansion or the declining of the market.
Also in this case, Equation 2.21 can be rearranged to express the formula as a hazard function:

$$\frac{z'(t)\,m(t) - z(t)\,m'(t)}{m^2(t)} = \left[\frac{z(t)}{m(t)}\right]' = \left[p + q\frac{z(t)}{m(t)}\right][m(t) - z(t)] \tag{2.22}$$

Then, by substituting $\frac{z(t)}{m(t)}$ with $y(t)$ we get

$$y'(t) = [p + qy(t)][1 - y(t)],\ t > 0 \tag{2.23}$$

which is the same hazard function of the BM, from which can be derived the closed-form equation related to the generalization:

$$z(t) = m(t)\,y(t) = m(t)\frac{1 - e^{-(p+q)t}}{1 + \frac{q}{p}e^{-(p+q)t}},\ t > 0. \tag{2.24}$$

2.3.1 Structure of $m(t)$ in the GGM

Equation 2.24 shows that $m(t)$ is an uninterpreted function that multiplies the dynamics of the adopters of a diffusion process over time. Then, $m(t)$ can take several structures depending on the hypotheses made on the market potential development. In [10], as mentioned before, the

authors hypothesize that the effective adoption phase is preceded by a communication process that is needed to pave the way for the market. For this purpose, they define $m(t)$ as

$$m(t) = K\sqrt{\frac{1 - e^{-(p_c+q_c)t}}{1 + \frac{q_c}{p_c}e^{-(p_c+q_c)t}}} \tag{2.25}$$

Equation 2.25 describes the communication process under some concepts taken from the network science theory that we will see more in detail later. However, in the equation, p_c describes the behavior of innovative consumers which act as communicators, whereas q_c acts as a receiver, and then, a means of spreading this information helping make it viral. The parameter K indicates the asymptotic behavior of $m(t)$, that is, the limit of this communication phase when all the possible customers are informed about the innovation and will eventually become adopters. Then, by applying that kind of dynamic market potential structure to Equation 2.24 we get the cumulative form of the GGM, but with some changes:

$$z\left(t\right) = K\sqrt{\frac{1 - e^{-(p_c+q_c)t}}{1 + \frac{q_c}{p_c}e^{-(p_c+q_c)t}}}\frac{1 - e^{-(p_s+q_s)t}}{1 + \frac{q_s}{p_s}e^{-(p+q)t}}, \;\; t > 0 \tag{2.26}$$

In the Equation 2.26, we can perhaps notice that the cumulative adoption function, $z(t)$, can be described as the product of two distinct factors, one indicating the communication phase characterized by parameters p_c and q_c, and the other as the adoption phase, which in Equation 2.24 was used to describe the dynamics of the adopters, characterized by parameters p_s and q_s. Note that, GGM can reduce to the standard BM in the case in which $m(t)$ immediately reaches K, named when the spread of information is immediate.

2.3.2 The assumptions behind $m(t)$

To formulate the function $m(t)$, authors in [14] extended the concept of *absorptive capacity* discussed by Cohen and Levinthal [15], in this last, the authors argue that a *prior related knowledge* defines the ability to assimilate and exploit a novelty, both on an individual and in a social context. That represents the main contribution to the hypothesis lying behind $m(t)$, since the adoption of an innovation in a specific social context may be viewed as direct evidence of an existing absorptive capacity. In Cohen and Levinthal [15], authors highlight the importance of designing a communication structure of an organization in order to better understand its absorptive capacity, and for this purpose, they assumed a cross-sectional model to describe such

structure. In Guseo and Guidolin [10], the authors redesigned the model, considering an evolutionary perspective based on a stochastic Cellular Automata model, inspired by Boccara [16], it considers a communication structure involving a set of informational linkages among the unit of the system in order to develop a collective knowledge through a cellular Automata Network. The reflection of this kind of network on a socio-economic domain on knowledge of this kind can be interpreted like that: the unit of analysis is represented by *edges* representing interpersonal links, they can be of two kinds, *standard edges* between two different agents, or *reflexive edges* representing a single agent which "auto-communicate" the information; the state of an edge can be active on inactive, and the activation occurs in case information passes or is passed, without accounting the means nor the direction of the vertex.

By putting these concepts in a more intuitive way, let us define them from a network science point of view. Let $G = (V, E)$ be a graph with a set of nodes $V = \{1, ..., N\}$ representing the *individuals*, and a set of edges E defined as ordered pairs (i, j) such that $E \subseteq \{(i, j)\} : i, j \in N, E \subset V^2$, that represents the possible relationships between the nodes V.
By representing the network using an adjacency matrix Y having 3 nodes:

$$Y = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

We get a square matrix of size $N \times N$, with edge (i, j), written as $Y_{i,j}$, equal to 1 if there is a connection in the pair (or a self-connection in case $i = j$), otherwise equal to 0. Under a perfect communication regime, all pairs $Y_{i,j} = 1$, however, in a realistic situation not all nodes will be connected, implying that the size U of the network, will be $U \leq N \times N$.

The GGM assumes that the market potential $m(t)$ is formed by all the individuals aware of the product, this same awareness process is assumed to be a communication diffusion process that evolves over time, thus, can be described using a BM:

$$h(t) = m_c v(t) = m_c \frac{1 - e^{-(p_c+q_c)t}}{1 + \frac{q_c}{p_c} e^{-(p_c+q_c)t}} \tag{2.27}$$

In Equation 2.27 coefficients m_c, p_c, and q_c are the parameters of BM referred to the communication process, therefore p_c, and q_c represent innovative and imitative behaviors respectively, while m_c is the total amount of possible connections in the network. Then, the quantity $h(t)$ represents the number of active edges of a network, the size U mentioned above, which for the definition of the market potential $m(t)$, accounts just for the *informed individuals* (i.e.

$V_{i,j} = 1$). The approximation of this value is thus obtained in a straightforward way, by considering the square root of $h(t)$

$$\sqrt{h(t)} = \sqrt{m_c}\sqrt{v(t)} = K\sqrt{\frac{1 - e^{-(p_c+q_c)t}}{1 + \frac{q_c}{p_c}e^{-(p_c+q_c)t}}} \tag{2.28}$$

2.3.3 Compact form of GGM

In "*Innovation Diffusion Models: Theory and Practice*" [7], the author rearranges GGM in a compact way for possible useful usages, such as this implementation, as follows:

$$z(t) = KS(t; p_c, q_c, p_s, q_s) = K\sqrt{F(t; p_c, q_c)}G(t; p_s, q_c) \tag{2.29}$$

In Equation 2.29 $S(t; p_c, q_c, p_s, q_s)$ is the product of the two cumulative distribution functions corresponding to the communication and adoption phases, respectively $\sqrt{F(t; p_c, q_c)}$ and $G(t; p_s, q_c)$.

The corresponding instantaneous process $z'(t)$ may be defined accordingly

$$z'(t) = KS'(t) = K\frac{1}{2\sqrt{F(t)}}G(t)f(t) + \sqrt{F(t)}g(t) \tag{2.30}$$

In Equation 2.30 $f(t)$ and $g(t)$ are respectively the derivative of $F(t)$ and $G(t)$, and are given by:

$$f(t; p_c, q_c) = \frac{(p_c(p_c + q_c)^2 e^{t(p_c+q_c)}}{(p_c e^{t(p_c+q_c)} + q_c)^2}$$

$$g(t; p_s, q_s) = \frac{(p_s(p_s + q_s)^2 e^{t(p_s+q_s)}}{(p_s e^{t(p_s+q_s)} + q_s)^2}$$

By rearranging $S'(t)$ in a more compact notation it is possible to highlight the presence of the two distinct phases of the diffusion process described by $k_1(t)$ and $k_2(t)$ in the following equation

$$S'(t) = \frac{1}{2\sqrt{F(t)}}G(t)f(t) + \sqrt{F(t)}g(t) = k_1(t) + k_2(t) \tag{2.31}$$

2.4 Unbalanced Competition Regime Change Diachronic Model

UCRCD is the last model implemented in PyDiM, developed by Guseo and Mortarino [11], it is part of a series of models that uses multivariate approaches to analyze the innovation diffusion by considering a crucial dynamic characterizing almost all commercial and technological markets: the competition. It is indeed a factor that can determine the trend of diffusion, representing an obstacle that can eventually cause its failure in the market, or bring benefits.

In general, the literature about these kinds of approaches extends the structure of the BM and focuses on bivariate models more than multivariate, due to the complexity of the differential equations systems involved, having an increasing number of parameters to estimate. In [17], authors make a review of the literature regarding these types of models, all the approaches seem characterized by a common thread concerning the introduction of a complex two-part imitation component that comprehends a *within-product* imitation, which accounts for the product's specific sales, and a *cross-product* imitation, which provides information on the effect of the competitor's sales on the product's life cycle. Another component we are interested in, and that characterizes these models, is the timing of market penetration of the two products, there are two possible scenarios that can present: the products enter the market at two different times, in this case, the competition is said to be *diachronic*; otherwise, in case the products penetrate the market simultaneously, the competition is said to be *synchronic*. As the name suggests, UCRCD is specialized in the former and thus assumes that the diffusion process is characterized by a first stage in which there is only one product in the market, and a second stage starting from the entrance into the market of the second product which rises the competition.

$$
\begin{aligned}
z_1'(t) = &\left\{ \left[p_{1_a} + q_{1_a} \frac{z(t)}{m} \right] (1 - I_{t>c}) \right. \\
&\left. + \left[p_{1_c} + (q_{1_c} + \delta) \frac{z_1(t)}{m} + q_{1_c} \frac{z_2(t)}{m} \right] I_{t>c} \right\} [m - z(t)], \\
z_2'(t) = &\left[p_2 + (q_2 - \gamma) \frac{z_1(t)}{m} + q_2 \frac{z_2(t)}{m} \right] [m - z(t)] I_{t>c},
\end{aligned}
\tag{2.32}
$$

Where

$$
\begin{aligned}
m &= m_a(1 - I_{t>c}) + m_c I_{t>c}, \\
z(t) &= z_1(t) + z_2(t) I_{t>c}.
\end{aligned}
$$

Parameters	Description
m_a	market potential of 1 before competition
p_{1_a}	innovation of 1 before competition
q_{1_a}	imitation of 1 before competition
m_c	market potential in competition
p_{1_c}	innovation of 1 in competition
$q_{1_c} + \delta$	within imitation of 1 in competition
q_{1_c}	cross imitation of 2 on 1
p_2	innovation of 2
q_2	within imitation of 2
$q_2 - \gamma$	cross imitation of 1 on 2

Table 2.1: UCRCD parameters and relative description

The two components of the model can be described by the differential equations expressed in System 2.32, where, as it is possible to notice, the market potential is given by different factors depending on the considered phase: in the first one, when there is no competition, we consider m_a; in the second, the competition phase, m_c is added to the equation. The cumulative adoption is given by the sum between the first product (which equation accounts for the two phases) and the second product, i.e., $z(t) = z_1(t) + z_2(t)I_{t>c}$. The residual market, $m - z(t)$, is common to the two equations, and terms δ and γ are added to the imitation coefficients and are needed to define the within and cross imitation mentioned before. In addition, $I_{t>c}$ is the indicator function, it is equal to 1 when $t > c$.

In the first phase of no competition, the life cycle of the first product, z_1', is described by parameters m_a, p_{1_a}, and q_{1_a}, modeled according to a standard BM. In the phase of the competition, when $t > c$, z_1' and the new product z_2' are still described according to BMs, the parameters returned by the models, defined by the tuples (m_1, p_{1_c}, q_{1_c}) and (m_2, p_2, q_2), are used to define the parameters of UCRCD in the competition phase. For instance, we get new parameters: the market potential under competition, $m_c = 2(m_1 + m_2)$; the *within* imitation coefficient $q_{1_c} + \delta$; and the *cross* imitation coefficient given by $q_2 - \gamma$.

In $z_1'(t)$, factor $(q_{1_c} + \delta)\frac{z_1(t)}{m}$ describes imitative behavior given by internal dynamics, while $q_{1_c}\frac{z_2(t)}{m}$ provides a measure of the influence of the sales of the second product on the first. The second product, $z_2'(t)$, describes in a symmetric way the competition phase of the former product: $(q_2 - \gamma)\frac{z_1(t)}{m}$ describes the influence of sales of the first product on the second, and $q_2\frac{z_2(t)}{m}$ describe the imitative behavior in the internal market.

Table 2.1 summarizes the parameters of the model. In general, the relationship between

q_{1_c}	$(q_2 - \gamma)$	Effect on the diffusion
$-$	$-$	The competition results unfavorable for both the competitors
$-$	$+$	Competition given by Product 2 results unfavorable just for Product 1
$+$	$-$	Competition given by Product 2 results favorable just for Product 1
$+$	$+$	Competition benefits both the Products

Table 2.2: Interpretation of the cross imitation coefficient signs

the first and second products is controlled by the cross imitation coefficients, q_{1_c} and $(q_2 - \gamma)$. While it would be straight to think about competition as a factor bringing negative effects on the competitors, empirical studies (also carried by using the same UCRCD, see [18, 19]) demonstrate that other scenarios are possible, as described in Table 2.2. Another aspect we must consider refers to the parameters δ and γ. According to what proposed by [11], when they are assumed to be equal, i.e., $\delta = \gamma$, the UCRCD is said to have *standard* form, in this case, the model can get closed form. The implication behind that choice is that we are assuming symmetry between the two competitors, which does not apply in realistic scenarios. Instead, by imposing no constraint, $\delta \neq \gamma$, the UCRCD takes a more flexible form which cannot allow a closed form. In [11] authors define it *unrestricted* UCRCD.

3 Implementation

To render PyDiM the most similar to its elder brother DIMORA [20], a lot of arrangements had to be done on different levels, starting from all the methods needed to display statistical summaries and plots for which Python has a lack of built-in libraries such as R's summary and plot (for the last one in particular, even though Python offers libraries, e.g. Matplotlib [21], there are no dedicated methods such as R's plot capable to handle multiple trends, in our case from competition models such as UCRCD, in an intuitive way). In any case, by the nature of the coding languages, PyDiM results 3 times faster in the computations compared to DIMORA, moreover, this Python implementation has been written more efficiently, with almost 400 lines of code versus 900 of the R version.

In the next sections, we will compare the two programming languages by first presenting their technical differences, in Section 3.1, then, in Section 3.2 will be presented some glimpse of the Python implementation and the difference in running time between it and R.

3.1 Main Differences Between Python and R

In data science and analytic, it is well known the hot debate about which one, between Python and R, is more suitable in the field. The reality is that they are two different languages providing different features and tools which make one a better fit for some specific use cases compared to one another. In general, they both have strengths and weaknesses that should be considered. In many ways they are very similar: open source, well suited for data science tasks such as data

manipulation, exploitation, and modeling, they are both easy to learn and execute. Python is a general-purpose programming language that emphasizes code readability. It supports data science tasks with libraries such as Numpy [22] for handling large dimensional data structures, Pandas for data manipulation and analysis, and Matplotlib [21] for building data visualization, but also libraries for statistics like Scipy [23], and machine and deep learning deploying. Python allows us to take advantage of all that in scalable production environments, for example, it is possible to use it to build machine learning applications and put them on mobile APIs. Moreover, Python is a high-level language, that ensures a smooth approach to object-oriented programming and higher readability, since it is more like the natural language, it also allows data rendering at a much higher speed compared to a low-level language like R, thanks to the shortest codes. R is a programming language optimized for statistical analysis, data preparation, and visualization, it can count on a rich ecosystem of complex data models and tools for data visualization (at least count, there are more than 13,000 packages on the Comprehensive R Archive Network, CRAN). R is particularly popular among scholars and researchers since it permits us to deploy deep statistical analysis using few lines of code and beautiful data visualization. As said above, R is a low-level language, it means longer codes for computations and more time for processing

3.2 Implementation and Running Time

This section will present the main insight of 's implementation with snippets of code for the functions related to the formulas presented in 2 organized by methods. Then, a comparison in running time between Python and R will be presented in most of the subsections.
Before going forward, note that:

- The models shares some parameters having the same use, hence they will be described there and not repeated later, if not necessary, to avoid redundancies:
  - *series*: data vector containing the series to be fitted
  - *prelimestimates*: vector containing the starting values used by the model
  - *alpha*: the confidence interval's significance level (default is always 0.05)
  - *oos*: the number of predictions after the last observed value in the series, if not specified (default will be set as the 25% of series length)
  - *display*: a boolean value, if "True" allows displaying the fitted values for cumulative and instantaneous observed data and oos (default is "True")

- The comparisons are being made on two main instances for the models: parameters estimation, named, the computation of $z(t)$; whole module running time; statistics computation and summary-displaying time of the *summary* module for each model; and predictions and plottings time for the *plot* module (this last process is timed using Python Jupiter's notebook since using the standard script does not allows for inline plotting).
- all the timings are given by the average running time computed on three code executions;
- PyDiM, at its first version (0.1.0), is implemented in Python v3.10.11 and ran on Visual Studio Code;
- for the comparison it is been used DIMORA version 0.3.5 executed in R v4.2.2 on RStudio;
- Both the libraries/packages are being run on a machine that mounts an Intel(R) Core(TM) i5-6200U 2.30GHz-2.40 GHz processor, and Windows 10 as OS.

The following subsections will present the generic methods of the library, useful to understand some logic implemented in the models, and all the methods implemented for each model, in the same order they are being described in Chapter 2.

3.2.1 Generic modules

Apart from the models, additional modules are being implemented in the library to simplify the use of generic functions between the models' methods, some examples are the computation of the statistics, the plots, and the prediction for the models.

Some conventions are being used in the following implementations, which include "private" modules and "protected" functions: in Python, a private/protected object (module, class, function) is an object that should neither be accessed outside a class nor by any base class or method, it is used to hide its inner functionalities from the outside. A private object is defined by a double underscore "__", a protected object by a single underscore "_", as a prefix of the name. Pay attention that this is just a convention used to communicate to other developers that those objects are not meant to be accessed, since Python objects are public by default.

summary.py

The module *summary*, as the name suggests, is used to display the statistics related to models' coefficient estimations to the user. This module contains just one callable function named *print_summary* which presents as follows

```
def print_summary(model):

    if model['type'] == "UCRCD model":
        stats = model['estimate']
        __ucrcd_summary(stats)
    else:
        stats = lib.get_stats(model['optim'], model['data'],
                model['prelimestimates'], model['method'], model['alpha'],
                model['type'], model['df'], model['residuals'])
        __standard_summary(stats)
```

As it is possible to notice, *print_summary* requires a model in input then, depending on whether it is a UCRCD model or not it retrieves and prints the statistics in two possible ways: in case the input is a UCRDC model, the statistics are retrieved from the model itself, since its parameters and statistics are computed internally to the model's module as we will see later, just as made in the R version; whereas the other models' statistics are computed externally through the *get_stats* function implemented in the *__lib* module, that will also be presented later. The private functions *__ucrcd_summary* and *__standard_summary* are built to actually display the statistics the more similar to R's *summary* function, they are very similar with the only difference in the logic, in fact, the former is thought to be able to handle multiple series input.

```
def __standard_summary(stats):
    print('')

    print('Residuals:')
    print('Min.         1st Qu.     Median    Mean       3rd Qu.    Max.')
    res_stat = st.describe(stats['Residuals'])
    print('% 5.6f   % 5.6f  % 5.6f  % 5.6f  % 5.6f  % 5.6f\n' % tuple([res_stat
    [1][0], \
        np.percentile(stats['Residuals'], 25), np.median(stats['Residuals']),
    res_stat[2], \
            np.percentile(stats['Residuals'], 75), res_stat[1][1]]))

    print('Coefficents:')
    print("    Estimate    Std. Error    Lower    Upper    p-value")
    significance = __assign_significance(stats['p-value'])

    if type(stats['Std. Error'][0]) is str:
        string = "% s % .4e   % s     % s     % s      % s  % s"
    else:
```

```
        string = "% s % .4e   % .4e   % .4e   % .4e    % .4e % s"

    for i in range(len(stats['Param'])):
        print(string \
            % tuple([stats['Param'][i], stats['Estimate'][i], stats['Std. Error'][i
    ], \
            stats['Lower'][i], stats['Upper'][i], stats['p-value'][i], significance[
    i]]))
    print('---')

    print("Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1\n")
    print('Residual Standard Error: % 5.6f on %i degrees of freedom' %
            tuple([stats['RMSE'], stats['Df']]))
    print('Multiple R-squared: % 5.6f Residual sum of squares: % 5.6f' %
            tuple([np.sqrt(stats['R-squared']), stats['RSS']]))
    print('\n')
```

An example of the output of the above function, for a BM summary on CD sales in the United States (Chapter 4 will present the full case study), would be

```
Residuals:
Min.          1st Qu.     Median     Mean       3rd Qu.    Max.
-266.057614   -115.811925  -41.746993  -39.515035   78.083947   180.558900

Coefficients:
    Estimate      Std. Error    Lower         Upper          p-value
m   1.4814e+04    4.9642e+01    1.4717e+04    1.4911e+04     0.0000e+00 ***
p   2.1919e-03    1.0573e-04    1.9847e-03    2.3991e-03     0.0000e+00 ***
q   2.5062e-01    3.5423e-03    2.4368e-01    2.5757e-01     0.0000e+00 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual Standard Error:  136.208147 on 37 degrees of freedom
Multiple R-squared:  0.999600 Residual sum of squares:  686448.396186
```

For the summary style, it is possible to notice that we have chosen to fully reply R's *summary* function since the result is displayed very clearly.

plot.py

As the name suggests, *plot* is the module implemented to handle the plotting of the models, it relies on plotting libraries such as *Matplotlib*[21] and *Statsmodels* [24] to display all the graphical information to understand the models fit: cumulative and instantaneous adoptions, residuals and residuals' autocorrelation. Just as in *summary*, it has just one callable function: *dimora_plot*, which implements two different logic depending on whether a model is a UCRCD or not, the only difference is that for UCRCD the module uses an inner protected function that displays bivariate series and fit, instead of univariates as for the other methods. The call for the function is the following

```
def dimora_plot(model, plot_type = 'all', oos=0, legend=None, index_as_label=
False)
```

the input parameters are pretty simple since the method is thought to be of easy use, and are: the *model* given in output by the model's implemented method; *plot_type* with which the user can choose to plot the fit of the model using "fit", the residual and autocorrelation plot by choosing "res", or both the fit and residual plots using "all"; *oos* requires an *int* value indicating how much points, out of the sample, he/she wants to predict; *legend* is used for stylistic choices, it allows the user to give a name to the series he is going to plot; the last parameter is *index_as_label*, a *boolean* parameter also added for stylistic purposes, if false it uses a time step index for the plot's x label, otherwise it uses the index of the original dataset triggering a sequence of checks implemented to recognize the kind of series it is handling (especially in case of time series).

The coding sequence for univariate series is implemented as follows

```
def _plot(len_series, data, len_w_oss, new_pred, model_pred, title, ax, labels,
legend):
    ax.plot(len_series, data, 'k.-', linewidth = .7, markersize=2.)
    ax.plot(len_w_oss, new_pred, 'g--')
    ax.plot(len_series, model_pred, 'r')
    ax.set_xlabel(labels[0])
    ax.set_ylabel(labels[1])
    ax.set_title(title)
    ax.legend(legend)

def _plot_res(t, res, ax, title_res, title_acf):
    ax[0].stem(t, res, markerfmt='k.', basefmt='k-')
    ax[0].set_title(title_res)
    ax[0].set_xlabel('t')
    ax[0].set_ylabel('Residuals')
```

```
        plot_acf(res, title = title_acf, ax=ax[1], color= 'black', marker='.')
        ax[1].set_xlabel('Lags')
        ax[1].set_ylabel('ACF')

    # product refers to the number of series
     if model['type'] == "UCRCD model":
        product = 2
     else: product = 1

     if product == 1:

        cumsum = np.cumsum(model['data'])
        res = model['residuals']

        # retrieving fit and predictions
        z_fit, z_prime_fit = predict.dimora_predict(model, t)
        z, z_prime = predict.dimora_predict(model, xlim)

        if plot_type == 'fit':
            fig, ax = plt.subplots(1,2)
            _plot(ind, cumsum, xlim, z, z_fit, 'Cumulative', ax[0], ['', 'z(t)'], legend)
            _plot(ind, model['data'], xlim, z_prime, z_prime_fit, 'Instantaneous', ax[1], ['t', "z'(t)"], legend)

        elif plot_type == 'res':
            fig, ax = plt.subplots(1,2)
            _plot_res(t, res, ax, title_res, title_acf)

        else:
            fig, ax = plt.subplots(2,2)
            _plot(ind, cumsum, xlim, z, z_fit, 'Cumulative', ax[0,0], ['', 'z(t)'], legend)
            _plot(ind, model['data'], xlim, z_prime, z_prime_fit, 'Instantaneous', ax[0,1], ['', "z'(t)"], legend)
            _plot_res(t, res, ax[1], title_res, title_acf)
```

An example of output for the below code using a (plot_type = “fit”, oos=20, legend “CD sales”, index_as_label = True) configuration is shown in Figure 3.1

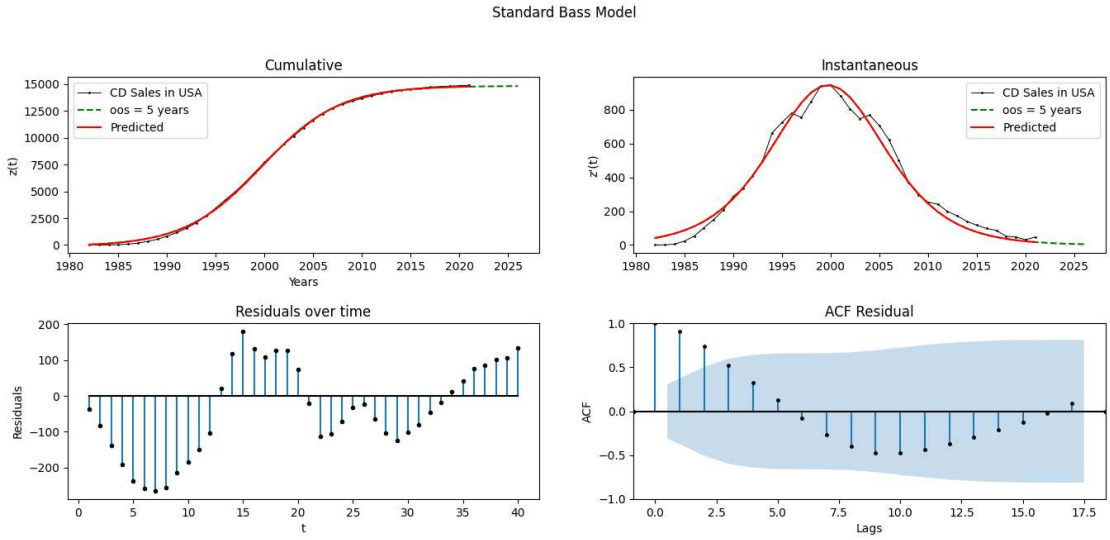

Figure 3.1: BM on CD Sales plots

__lib.py

The *__lib* module is one of the two private modules of the library, it is meant to handle generic computations of the library: the statistics (already mentioned in *summary.py*), or series starting with long sequences of zeros. The function *get_stats* computes the statistics for the model and represents the most prominent function of the method, defined as follows:

```
def get_stats(ls, series, prelimestimates, method, alpha, model, df = None, res=None
    ):
    parameters = _lib.set_params(model)

    if df != None: df = df
    else: df = len(series) - len(prelimestimates)
    # print(df)
    y_mean = np.mean(prelimestimates)
    TSS = np.sum((series-y_mean)**2)

    if method == "nls":
        if df == None:
            df = len(series) - len(ls[0])
        # Get the parameters
        parmEsts = ls[0]
        # Get the Error variance and standard deviation
        res = ls[2]['fvec']
```

```
        RSS = np.sum(res**2)
        MSE = RSS / df
        RMSE = np.sqrt(MSE)
        # Get the covariance matrix
        cov = np.abs(MSE * ls[1])
        # Get parameter standard errors
        parmSE = np.diag(np.sqrt(cov))
        # Calculate the t-values
        tvals = parmEsts/parmSE
        # Get p-values, 2-sided test
        pvals = (1 - st.t.cdf(np.abs(tvals), df))*2
        # Get biased variance (MLE) and calculate log-likehood
        s2b = RSS / len(series)
        logLik = -len(series)/2 * np.log(2*np.pi) - \
            len(series)/2 * np.log(s2b) - 1/(2*s2b) * RSS
        # Get R-squared
        R_squared = 1 - RSS/TSS
        # Get Lower & Upper bounds
        lower = [(parmEsts[j] + -1 * np.dot(st.norm.ppf(1-alpha/2), parmSE[j]))
                for j in range(len(parmSE))]
        upper = [(parmEsts[j] + np.dot(st.norm.ppf(1-alpha/2), parmSE[j]))
                for j in range(len(parmSE))]

    elif method == "optim":
        parmEsts = ls[0]
        parmSE = ['-' for i in range(len(parmEsts))]
        lower = ['-' for i in range(len(parmEsts))]
        upper = ['-' for i in range(len(parmEsts))]
        tvals = ['-' for i in range(len(parmEsts))]
        pvals = ['-' for i in range(len(parmEsts))]
        RSS = np.round(ls[1], 4)
        MSE = RSS / df
        RMSE = np.sqrt(MSE)
        R_squared = 1 - RSS/TSS

    stats = {
        'Residuals': res,
        'Param': parameters[:len(parmEsts)],
        'Estimate': parmEsts,
        'Std. Error': parmSE,
        'Lower': lower,
```

```
        'Upper': upper,
        't-value': tvals,
        'p-value': pvals,
        'RMSE': RMSE,
        'Df': df,
        'R-squared': R_squared,
        'RSS': RSS
    }
    return stats
```

The need to implement such a function is given by the lack of built-in methods able to handle the statistics of the minimization approaches implemented in SciPy [23], so, instead of looking for alternatives we decided to implement our own simple method, that although being a pretty long function it results very straightforward.

The other mentioned function is very easy and it is implemented to take care of the series starting with more than two zeros:

```
def handle_zeros(series):
    i=0
    while series.iloc[i+1] == 0:
        i += 1
    return series.iloc[i:]
```

It cut all the zeros but the one before the effective start of the growth of the series, that is because it can happen, for example, handling NaNs and replacing them with 0, that the series will present a long tail of zeros which does not help in describing any sort of life cycle that the implemented models are meant to describe. This function is typically called by the models in a preliminary phase of series and parameters checking.

__predict.py

This second private module takes care of the model's predictions, it contains just one simple function, called *dimora_predict* which retrieves models' internal functions to compute cumulative and instantaneous adoptions using the same models' parameters, it presents as follows

```
    def dimora_predict(model, t = None):
    if model['type'] == "Standard Bass Model":
        m, p, q = model['optim'][0]
        fit = model['functions'][0](t, m, p, q)
        instant = model['functions'][1](t, m, p, q)

```

```
        return fit, instant

    elif model['type'] == "Generalized Bass Model":
        params = model['optim'][0]
        fit = model['functions'][0](params, t, model['shocks'][0], model['
    x_functions'][0])
        instant = model['functions'][1](params, t, model['shocks'][0], model['
    x_functions'][0], model['x_functions'][1])

        return fit, instant

    elif model['type'] == "Guseo-Guidolin Model":
        params = model['optim'][0]
        fit = model['functions'][0](t, params, model['market_potential'])
        instant = model['functions'][1](t, params)

        return fit, instant

    elif model['type'] == "UCRCD model":
        raise KeyError("UCRCD does not allows for predictions in this implementation
    ")

    else:
        raise KeyError("Model type not recognized, be sure to input a PyDiM's model"
    )
```

As it is possible to notice, all the univariate models call the functions from the model's instance to make the predictions, except UCRCD that, as for now cannot allow predictions.

3.2.2 Bass Model

This is the first implemented method of the package since it is the simpler and the base of most of the other methods' implementation, the implementation required just 49 LOC versus almost 107 of R. The call for the method presents like that:

```
def bm(series, method="nls", prelimestimates=[], alpha=0.05, oos=None, display=True)
```

Apart from the default attributes mentioned before, BM requires an optional parameter *method* used to select the minimization method for the adoption function, default is "nls", but "optim" can be chosen as an alternative, they respectively minimize the function using nonlinear least-squares with the Levenberg-Marquardt algorithm [25], or Unconstrained minimization using

a Limited Memory BFGS [26]. Both the algorithms used to minimize the function are implemented by the SciPy library [23] and applied to our code as follows

```
    if method == "nls":
        optim = opt.leastsq(func=_residuals, x0=prelimestimates, args=(t),
    full_output=1)
        res = optim[2]['fvec']

    elif method == "optim":
        mass = np.sum(series) + 1000
        min = opt.minimize(fun=_f, x0=prelimestimates, args=(t), bounds=[(1e-10,
    mass), (1e-10, 1), (1e-10, 1)], method='L-BFGS-B')
        res = _residuals(min.x, t)
        optim = [min.x, min.fun, res]
```

Two main differences can be noticed in the declarations of the two options: parameter *optim* is defined in different ways, that is because of the nature, and so the output, of the two minimization algorithms; the functions given in input for the minimization algorithms are related to the residuals and residuals sum of squares of the cumulative adoption function, then these, and the instantaneous adoption function, are defined as follows

```
    def _z(t, m, p, q):
        return (m * (1 - np.exp(- np.multiply((p + q), t))) / (1 + q / p * np.exp(-
    np.multiply((p + q), t))))

    def _zprime(t, m, p, q):
        return (p+q*_z(t, m, p, q)/m)*(m - _z(t, m, p, q))

    def _residuals(par, t):
        return cumsum - _z(t, par[0], par[1], par[2])

    def _f(par, t):
        return np.sum(_residuals(par, t)**2)
```

As it is possible to notice, *_z* and *_zprime* are the exact implementations of Equations 2.5 and 2.1 defined in Section 2.1:
As the last step, the BM method returns a dictionary containing all the useful parameters for successive scopes, and eventually plots the results, if selected.

```
    model = {
        'type' :"Standard Bass Model",
        'functions' : [_z, _zprime],
```

```
        'data' : series,
        'prelimestimates' : prelimestimates,
        'method' : method,
        'alpha' : alpha,
        'df' : None,
        'optim' : optim,
        'residuals' : res,
    }

    if display:
        plot.dimora_plot(model, 'fit', oos)

    return model
```

Table 3.1 contains the running times of the method applied to a time series about CD sales in the USA (Subsection 4.1, this and the next models' example will be analyzed in depth in the next section.

	Python (s)	R (s)
Parameters optimization	0.00598	0.03101
Whole module	0.00747	0.20463
Summary computation	0.01944	0.01559
Prediction	0.0002	0.0012
Plotting	1.45731	0.06379

Table 3.1: Running Times for the BM

3.2.3 Generalized Bass Model

GBM is the second implemented method and the one that deserves some clarifications as we will se later, the call for the method presents as follows

```
def gbm(series, shock, nshock, prelimestimates, alpha=0.05, oos=None, display=True)
```

Among the parameters, those that characterize the method are *shock* and *nshock*. The former's input must be a string, between "exp", "rett", or "mixed" indicating the kind of shock(s) to map the trend; and the latter must be an integer type, that indicates the number of shocks of the kind defined in the previous parameter. In this case, *prelimestimates* has no default argument since it is up to the user to indicate the approximate parameters for the BM and for each shock. GBM, in its actual implementation, is not able to automatically find the shocks' parameters.

The function for the method is minimized just as BM, with a Non-Linear least squared based on the Levenberg-Marquardt algorithm [25], on the residuals on the cumulative adoption. The functions $z(t)$ and $z'(t)$ for this method are defined as

```
1   def _z(shock_par, t, nshock, intx):
2       z_part = 0.
3       m = shock_par[0]
4       p = shock_par[1]
5       q = shock_par[2]
6
7       for i in range(1, nshock+1):
8           z_part += intx(t, i, shock_par)
9       z_prime = z_part + t
10      z = m * (1 - np.exp((-(p+q)*z_prime), dtype= np.float64)) / (1+(q/p)*np.exp
    ((-(p+q)*z_prime), dtype=np.float64))
11
12      return z
13
14  def _zprime(shock_par, t, nshock, intx, xt):
15      xi = 0
16      m = shock_par[0]
17      p = shock_par[1]
18      q = shock_par[2]
19
20      for i in range(1, nshock+1):
21          xi += xt(t, i, shock_par)
22
23      x_t = 1 + xi
24      z_t = _z(shock_par, t, nshock, intx)
25      z_prime = (p + q * (z_t/m)) * (m - z_t) * x_t
26
27      return z_prime
```

From the code above it is possible to notice that the functions *_z* and *_zprime* are, again, the exact codification of Equations 2.12 and 2.10. At rows 7 and 21, the loops are being implemented to express the function $x(t)$ (and its respective integral) as a sum of the shocks, as defined in Equations 2.15, 2.17, and 2.19, and the respective integrals. In addition, the parameters "xt" and "intx" refers to the current marketing effort function and its integral, they are given in input as functions defined depending on the selected kind of shock as follows

```
    '''EXPONENTIAL SHOCKS GENERALIZED FUNCTIONS'''
    def _exp_intx_gen(t, index, shock_par):
        a = shock_par[3*index]
        b = shock_par[3*index+1]
        c = shock_par[3*index+2]
        intx = c*(1/b)*(np.exp(np.dot(b,(t-a)))-1)*(t>=a)

        return intx

    def _exp_xt_gen(t, index, shock_par):
        a = shock_par[3*index]
        b = shock_par[3*index+1]
        c = shock_par[3*index+2]
        xt = (c*np.exp(np.dot(b, (t-a))))*(t >= a)

        return xt

    '''RECTANGULAR SHOCKS GENERALIZED FUNCTIONS'''
    def _rett_intx_gen(t, index, shock_par):
        a = shock_par[3*index]
        b = shock_par[3*index+1]
        c = shock_par[3*index+2]
        intx = np.dot(c,(t-a))*(t>=a)*(t<=b) + c*(b-a)*(t>b)

        return intx

    def _rett_xt_gen(t, index, shock_par):
        a = shock_par[3*index]
        b = shock_par[3*index+1]
        c = shock_par[3*index+2]
        xt = c*(t>=a)*(t<=b)

        return xt

    '''MIXED SHOCKS GENERALIZED FUNCTIONS'''
    def _mix_intx_gen(t, index, shock_par):
        if shock[index-1] == 'exp':
            intx = _exp_intx_gen(t, index, shock_par)
        else:
            intx = _rett_intx_gen(t, index, shock_par)
        return intx
```

```
42
43      def _mix_xt_gen(t, index, shock_par):
44          if shock[index-1] == 'exp':
45              xt = _exp_xt_gen(t, index, shock_par)
46          else:
47              xt = _rett_xt_gen(t, index, shock_par)
48          return xt
49
50      if shock[0] == 'exp':
51          intx = _exp_intx_gen
52          xt = _exp_xt_gen
53      elif shock[0] == 'rett':
54          intx = _rett_intx_gen
55          xt = _rett_xt_gen
56      elif shock[0] == 'mixed':
57          shock = ['exp', 'rett']
58          intx = _mix_intx_gen
59          xt = _mix_xt_gen
```

There we have to make a clarification: as noticeable, the variable "shock" here is a list, and not a string anymore, that is because the variable is meant to be more prone to changes in the future, especially in the case of mixed shocks having more than 2 shocks, so that the kinds of occurring shocks can be listed and treated according to the clauses expressed in the declaration of the functions.

Just as the BM, also GBM exits the execution by returning a dictionary containing the useful parameters, and eventually displaying the plot of the fitted model on the series:

```
1       model = {
2           'type' :"Generalized Bass Model",
3           'functions' : [_z, _zprime],
4           'data' : series,
5           'prelimestimates' : prelimestimates,
6           'method' : "nls",
7           'alpha' : alpha,
8           'df' : None,
9           'optim' : optim,
10          'residuals' : res,
11          'shocks' : [nshock, shock],
12          'x_functions': [intx, xt],
13          }
14
```

```
    if display:
        plot.dimora_plot(model, 'fit', oos)

    return model
```

The running times for a GBM with one rectangular on the birth rate in Japan[27] time series (Case study analyzed in Subsection 4.2) are shown in Table 3.2

	Python (s)	R (s)
Parameters optimization	0.01795	0.02711
Whole module	0.01795	0.70096
Summary computation	0.0728	0.01041
Prediction	0.0002	0.005
Plotting	1.39327	0.10783

Table 3.2: Running Times for the GBM function with 1 rectangula shock

Running times for GBM with mixed shock on the trend of monthly interest in Facebook* (the relative case study is presented in Subsection 4.2) are shown in Table 3.3

Topic	Python (s)	R (s)
Parameters optimization	0.10771	0.24512
Whole module	0.10871	0.29825
Summary computation	0.13264	0.00813
Prediction	0.0001	0.0058
Plotting	1.04022	0.10664

Table 3.3: Running Times for the GBM function with 2 mixed shock

3.2.4 Guseo-Guidolin Model

The GGM is the third and last univariate model implemented in PyDiM, its implementation, as we will see, is quite clean and straightforward with its 68 LOC. The call is very similar to the first two methods and defined as follows

```
    def ggm(series, mt = None, prelimestimates = None, alpha = 0.05, oos=None,
    display = True)
```

*According to Google Trends (n.d.), monthly interest in Facebook June, 2023

The parameters are exactly the same as the BM, presented in 3.2.2, apart from the presence of a parameter *mt* in charge of the definition of the *dynamic market potential* function. The method is, in fact, thought to describe market dynamics as described by Guseo and Guidolin in [10] (and explained with Equation 2.25) by default, however, it is also able to get other functions able to describe market dynamics based on time. That turns the method into a general-purpose BM with dynamic market potential behaviors. The default $m(t)$ is defined as in Equation 2.25:

```
    def _mt_func(t, K, pc, qc):
        mt = K * np.sqrt(np.abs((1 - np.exp(-(pc + qc) * t)) / (1 + (qc / pc) * np.
    exp(-(pc + qc) * t))))
        return mt
```

and the two (cumulative and instantaneous) functions are described by the following Python functions

```
    def _z_base_mt(t, par, mt):
        K, ps, qs, pc, qc = tuple(par[0:])
        z = mt(t, K, pc, qc) * (1 - np.exp(-(ps + qs) * t)) / (1 + (qs / ps) * np.
    exp(-(ps + qs)*t))
        return z

    def _z_defined_mt(t, par, mt):
        K, ps, qs = tuple(par[0:2])
        z = K * mt(t) * (1 - np.exp(-(ps + qs) * t)) / (1 + (qs / ps) * np.exp(-(ps
    + qs)*t))
        return z

    def _zprime(t, par):
        K, ps, qs, pc, qc = tuple(par[0:])
        F_t = (1 - np.exp(-(pc + qc) * t)) / (1 + (qc / pc) * np.exp(-(pc + qc) * t)
    )
        G_t = (1 - np.exp(-(ps + qs) * t)) / (1 + (qs / ps) * np.exp(-(ps + qs) * t)
    )

        ft = (pc * (pc+qc)**2 * np.exp(t*(pc+qc))) / ((pc * np.exp(t*(pc+qc)) + qc)
    **2)
        gt = (ps * (ps+qs)**2 * np.exp(t*(ps+qs))) / ((ps * np.exp(t*(ps+qs)) + qs)
    **2)

        k1_t = (1/2)* F_t**(-1/2) * G_t * ft
```

```
        k2_t = np.sqrt(F_t) * gt

        return K*(k1_t + k2_t)
```

As it is possible to notice, two $z(t)$ functions are implemented, the first (*_z_base_mt*) use the GGM approach to fit the cumulative data, the second one takes care of an eventual $m(t)$ defined by the user. The choice between the two is handled by a conditional statement (if/else):

```
    if type(mt) == 'function':
        _z = _z_defined_mt
    elif mt == 'base' or mt == None:
        _z = _z_base_mt
        mt = _mt_func
    else:
        raise KeyError("'mt' parameter must be either a function or None/left blank
    ")

    def _residuals(par, t, mt):
        return cumsum - _z(t, par, mt)

    optim = opt.leastsq(func=_residuals, x0=prelimestimates, args=(t, mt),
    full_output=1)
```

Regardless of the user's choice, the cumulative function is allocated on a variable _z, and optimized via nonlinear least-squares on the residuals

In table 3.4 are shown the running times of GGM applied to Japan's birth rate analyzed in the case study in Subsection 4.3

Topic	Python (s)	R (s)
Parameters optimization	0.04089	0.02779
Whole module	0.05386	0.08108
Summary computation	0.14062	0.015594
Prediction	0.001	0.005
Plotting	1.8361	0.11229

Table 3.4: Running Times for the GGM

3.2.5 Unbalanced Competition Regime Change Diachronic model

UCRCD is the last method implemented in this library, unlike the previous implementations, this results to be a little more difficult since it requires fragmented computations on different parts of the time series. Moreover, even though we presented this as a *diacronic* model, as we will see, the implementation allows to handle *synchronic* competition too. The call for the method is the following

```
    def ucrcd(series1, series2, par = "double", prelimest_series1 = None,
        prelimest_series2= None, alpha=0.05, delta=0.01, gamma=0.01,
        display=True)
```

This method requires the following specialized parameters: *par*, it accepts a string between "double" or "unique", they are needed to set constraints on terms *delta* and *gamma*, if "unique" is chosen then a *standard UCRCD* (see Subsection 2.4), i.e., $\delta = \gamma$, will be used, otherwise, *unrestricted UCRCD* will be used; parameters *delta* and *gamma* are the respective preliminary estimates for the terms.

On a standard diachronic regime, the method act as follows: it applies a standard BM on the section of the first product trend to estimate the parameters under the non-competition phase

```
c2i = len(series1) - len(series2)
end = len(series1)

if c2i > 0 :
    s1 = series1.iloc[:c2i]
    series1 = series1.iloc[c2i : end]
    t = np.arange(1, c2i+1, 1)
    s2 = np.zeros(c2i)
    Z1 = np.cumsum(s1)
    Z2 = np.cumsum(s2)

    BMs1 = BM.bm(s1, display=0)
    m1, p1a, q1a = BMs1['optim'][0]

    # making predictions on the non-competitive part of the series
    pred_BM1 = BMs1['functions'][1](t, m1, p1a, q1a)

    o_bass = pd.DataFrame.from_dict({'t': t, 's1': s1, 's2': s2, 'Z1': Z1, 'Z2': Z2
    })
```

```
    p_bass = pd.DataFrame.from_dict({'t': t, 'pred_1': pred_BM1, 'pred_2': s2})
```

Then, we define the model so that it can fit with or without constraint, just by posing in Equation 2.32 $(q_2 - \delta)$ instead of $(q_2 - \gamma)$

```
t = np.arange(c2, end, 1)
Z1 = data1.iloc[c2:end]
Z2 = data2

data = pd.DataFrame.from_dict({'t': t, 's1': series1, 's2': series2, 'Z1': Z1, 'Z2':
     Z2})

def _model(t, params, par):
    Z = Z1+Z2

    if par == 'unique':
        mc, p1c, p2, q1c, q2, delta = tuple(params)
        z2 = (p2 + (q2 - delta) * Z1 / mc + q2 * Z2 / mc) * (mc - Z)
    elif par == 'double':
        mc, p1c, p2, q1c, q2, delta, gamma = tuple(params)
        z2 = (p2 + (q2 - gamma) * Z1 / mc + q2 * Z2 / mc) * (mc - Z)

    z1 = (p1c + (q1c + delta) * Z1 / mc + q1c * Z2 / mc) * (mc - Z)

    return {'z1':z1, 'z2': z2, 't':t}
```

At this point, the parameters are minimized and used to produce the remaining part of the statistics, which will be stacked later.

```
fitval1 = opt.leastsq(func= _res_model, x0=params, args=(t, par), maxfev=10000,
    full_output=1)

df = len(series1) + len(series2) - len(params)

parest = fitval1[0]

estimates = _model(t, parest, par)
z_prime = pd.DataFrame.from_dict({'t':estimates['t'], 'pred_1': estimates['z1'], '
    pred_2': estimates['z2']})

data['t'] = np.arange(c2,end)
z_prime['t'] = np.arange(c2,end)

```

```
# computing the coefficients stats
stats1 = lib.get_stats(BMs1['optim'], BMs1['data'], BMs1['alpha'], lib.set_params(
    BMs1['type']))

no_competition_stats = np.row_stack([[stats1['Estimate'][i], stats1['Std. Error'][i
    ], stats1['Lower'][i],\
    stats1['Upper'][i], stats1['p-value'][i]] for i in range(len(stats1['Estimate'])
    )])

stats2 = lib.get_stats(fitval1, tot, alpha, lib.set_params('UCRCD'), df=df)

competition_stats = np.row_stack([[stats2['Estimate'][i], stats2['Std. Error'][i],
    stats2['Lower'][i],\
    stats2['Upper'][i], stats2['p-value'][i]] for i in range(len(stats2['Estimate'])
    )])
```

The last part of the UCRCD code is in charge to make the final adjustments, to return all the variables in a readable form

```
### Final adjustments and statistics ###
obs = pd.melt(data.iloc[:,:3], id_vars=['t'], var_name='product', value_name='
    consumption')
pred = pd.melt(z_prime, id_vars=['t'], var_name='product', value_name='consumption')

ss1 = obs['consumption'][0:end]
ss2 = obs['consumption'][end+c2 : 2*end]

pp1 = pred['consumption'][0:end]
pp2 = pred['consumption'][end+c2 : 2*end]

res = obs['consumption'] - pred['consumption']

res1 = res[0:end]
res2 = res[end+c2: 2*end]

tss = np.sum((obs['consumption']- np.mean(obs['consumption']))**2)
rss = np.sum(res**2)
r_squared = 1 - rss/tss

df = [len(ss1)-len(Estimate1[:,0]), len(ss2)-len(Estimate1[:,0])]
MSE = [np.sum(res1**2) / df[0], np.sum(res2**2) / df[1]]
RMSE = np.sqrt(MSE)
```

UCRCD applied to the competition between Covid-19 daily cases and daily vaccination, the relative case study is presented in Section 4.4

Topic	Python (s)	R (s)
Parameters optimization	0.35008	0.26772
Statistics computation	0.01795	0.026
Whole module	0.42885	0.30505
Summary computation	0.46187	0.015594
Plotting	1.91688	0.2342

Table 3.5: Running Times for the UCRCD

4

Case Studies

In this chapter of the thesis, we are going to explore some applications of the presented models and make a detailed analysis of the estimations by considering the statistics related to them and the predictions graphically shown in the plots. Moreover, as anticipated in the Introduction 1, we are going to demonstrate the efficiency of those models over events pertaining to different natures, giving, if necessary, the relative key to reading to understand the parameters.

4.1 Bass Model

In this section, we will see an application of the BM applied on the yearly Compact Disc sales in the USA's market*, a beautiful bell-shaped trend related to the intercourse of one of the most important products that characterized the music reproduction industry, and the perfect fit of this model for the purpose.

4.1.1 CD Sales in USA

Recorded music is a very good example of a succession of innovations, in the last 60 years, almost six out of eight reproduction tools have served their purpose, leaving space for the successors, see for example the 8-Tracks, the cassettes, or the CDs. Others have fallen but are seeing a new rise, see the vinyl that is riding the wave by the streaming music side. One of the bigger

*https://www.statista.com/chart/12950/cd-sales-in-the-us/

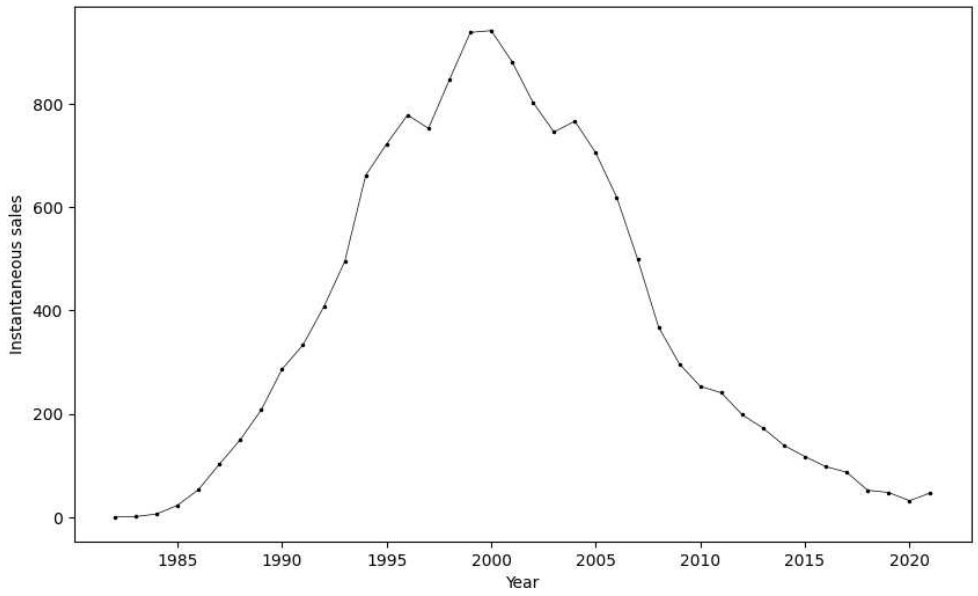

Figure 4.1: CD Sales in million units between 1982 and 2021

	Estimate	Std. Error	Lower	Upper
m	$1.4814e+04$	$4.9642e+01$	$1.4717e+04$	$1.4911e+04$
p	$2.1919e-03$	$1.0573e-04$	$1.9847e-03$	$2.3991e-03$
q	$2.5062e-01$	$3.5423e-03$	$2.4368e-01$	$2.5757e-01$

$R^2 = 0.9996$

Table 4.1: BM estimations on CD Sales

footprints in this market was for sure given by the advent of compact disks, they represent the beginning of the digital music era and the bigger source of revenues as of today. In this first case study, we analyze the Compact Disks' yearly sales in million units, between 1982 and 2021, in the United States. By looking at the trend in Figure 4.1 it is possible to notice that its life cycle seems complete, and the diffusion process followed almost exactly a Bass-like behavior, apart from some little perturbations near the peak, in 2000. Table 4.1 and Figures 4.2, 4.3 report the results of the fitting. The model is representative, with an $R^2 = 0.9996$ and all the parameters are significant, with *p-value* < 0.001. The innovation and imitation coefficients are $p = 0.0022$ and $q = 0.25$ suggesting that the spread of compact disks was not so much based on the innovation factor as on the imitation. Market potential of $m = 14,814$ (million units) helps understand the extent of what was said before about CDs as bigger sources of income in the music reproduction industry.

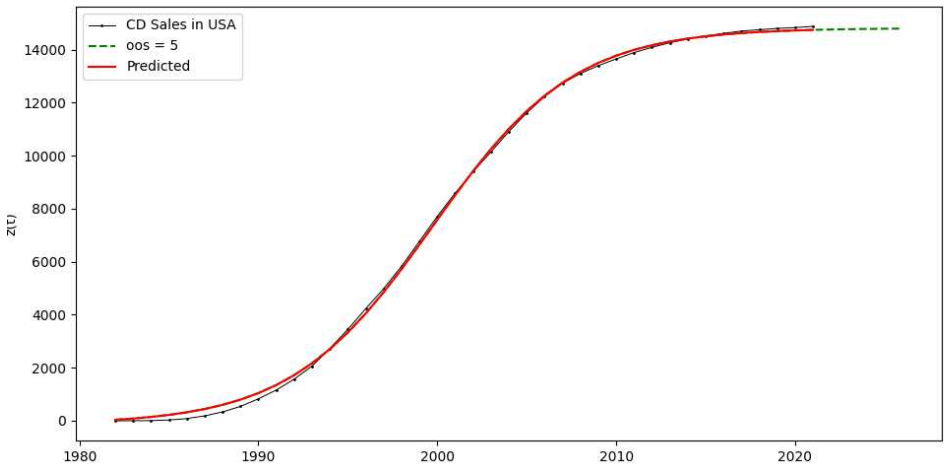

Figure 4.2: Cumulative BM on CD Sales

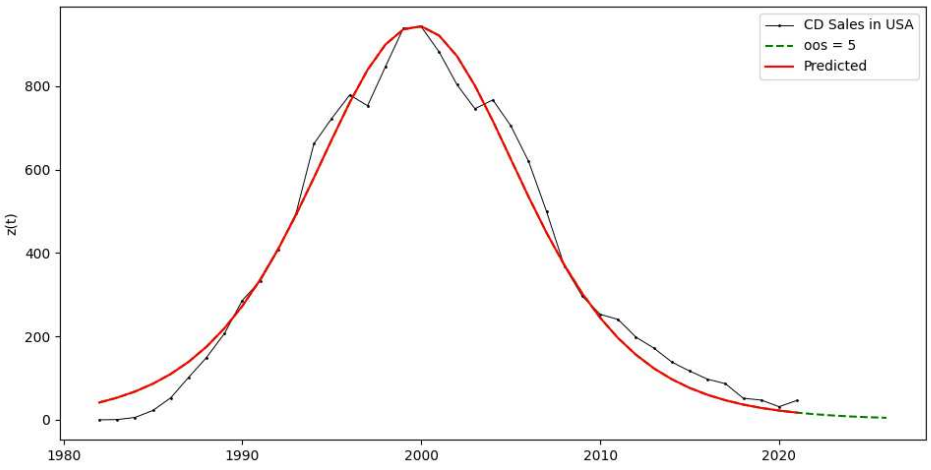

Figure 4.3: Instantaneous BM on CD Sales

4.2 Generalized Bass Model

For GBM we have two very different examples: the birth rate in Japan between 1872 and 2022, and the number of research on Google Search regarding Facebook. In the former example, we will explore and analyze how this kind of model can possibly analyze a trend highly affected by economic, political, and social changes. While in the former we will adopt a point of view always aimed at change, but understood as the interchange of technological means in our era.

4.2.1 Birth rate in Japan

An interesting trend we wanted to study, refers to the national birth rate, Japan's case seemed a good fit for a GBM example, as we will see below. But first, let us analyze some of the main possible reasons that relate this kind of trend to economic and social factors. Individuals' economic conditions, such as disposable income, employment, economic security, and cost of living, can influence decisions about procreation, it is not unusual that in situations of poverty or economic uncertainty, people may delay or avoid having children because of financial concerns related to child nutrition, education, and health. On a social level, we can consider even more aspects, internal, such as the cultural norms, social expectations, the perceptions about the role of the family, and external, such as the availability of social policies and services designed to assist families, guarantee health services and social protection. Figure 4.4 displays a trend of the birth rate in Japan between 1872 and 2022, the rate is expressed as the annual number of births per person, the data from 1872 to 2009 taken from [27], 2010-2022 from the United Nations data-bank[†], and missing data for 1944-45-46 are estimations taken from [28].
As we can see, the trend in Figure 4.4 shows an interesting behavior that can be divided into three main phases. The first phase of growth, between 1872 and 1920, period in which Japan went through rapid industrial development and modernization that accelerated the economic growth and the rate of employment, and also through significant changes in the social structure and living conditions of people. Then, a second phase of decline between 1921 and 1946 has been influenced by a combination of factors, including urbanization, industrialization, cultural and social change, the evolution of women's roles, economic hardship, and improved access to contraceptive methods. The third phase regards the war and postwar period, until nowadays, where it is possible to notice a sudden peak in 1947, probably given by the economic recovery and a desire for reconstruction, followed by an exponential decay after 1949 that started slow-

[†]https://population.un.org/wpp/

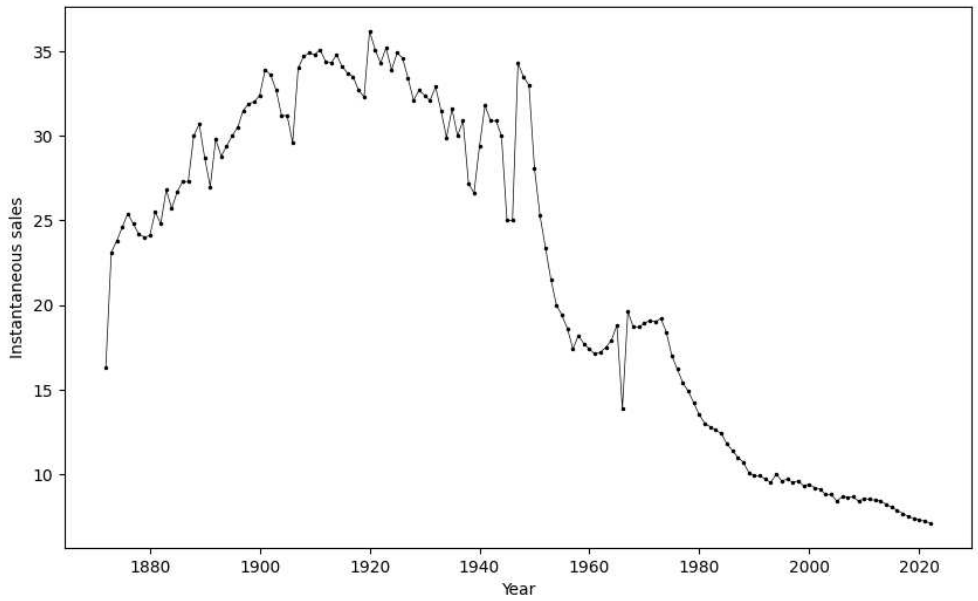

Figure 4.4: Japan birth rate per person between 1872 and 2022

	Estimate	Std. Error	Lower	Upper
m	$3.4216e+03$	$3.1171e+00$	$3.4155e+03$	$3.4277e+03$
p	$6.3789e-03$	$1.4244e-05$	$6.3509e-03$	$6.4068e-03$
q	$2.5888e-02$	$9.9558e-05$	$2.5693e-02$	$2.6083e-02$
$a1$	$8.3500e+01$	$8.1681e-01$	$8.1900e+01$	$8.5101e+01$
$b1$	$9.2686e+01$	$9.5245e-01$	$9.0819e+01$	$9.4553e+01$
$c1$	$-1.9375e-01$	$3.0178e-02$	$-2.5290e-01$	$-1.3460e-01$

$R^2 = 0.9999$

Table 4.2: GBM with 1 rectangular shock estimations on Japan birth rate

ing down after 1973, mostly given by an improvement of living conditions, urbanization, social changes, economic pressures, and the aging of the population. In Table 4.2 and Figures 4.5 and 4.6 are shown the results of a GBM with a negative rectangular shock starting at $a_1 = 83.5$ (1952) and ending at $b_1 = 92.69$ (2006) with intensity $c_1 = -1.94$. All the coefficients are significant in the confidence interval. The model reaches an $R^2 = 0.9999$. BM's parameters p and q assume, in this context, different meanings compared to a marketing or economic context, a possible interpretation could be: p is the rate of people having a sort of innate will to procreate, so people that do not accounts to external factors but rather to ideals and needs they have; and q as the people who are more prone to have children based on the social and economic

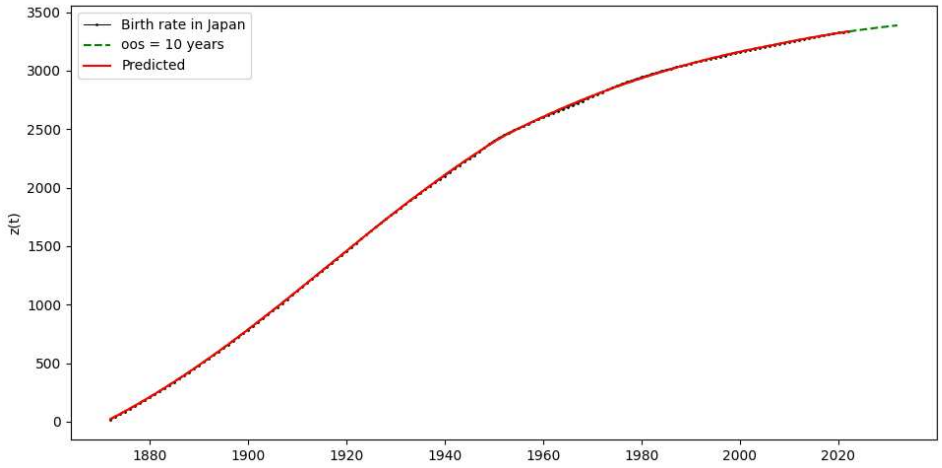

Figure 4.5: Cumulative GBM 1 rectangular shock on Japan birth rate

context, in this case, we speak about people using a more rational way of thinking in this sense. Based on this interpretation of $p = 0.0064$ and $q = 0.026$ we can say that the behavior of the trend is attributable to the industrial and economic growth, westernization, and social changes we were talking about earlier, rather than a cultural factor.
The choice of the use of a shock of the rectangular kind can be justified by the nature of this phenomenon. If it is true that history is cyclic, then it is very likely that the birth rate of the nation will eventually start to grow again, ending a transient period of decreasing birth rate that, by definition (Section 2.2), this kind of shock should be able to fit well.

4.2.2 Interest in Facebook

With almost 3 billion monthly active users, Facebook is the most-used social network nowadays. Founded in 2004 by M. Zuckerberg, as the main founder, to connect university students. Started spreading worldwide after September 2006, and in just one year it entered the top 10 visited websites, in march 2010, just for a week, traffic on Facebook surpassed Google's search engine, and in August 2015 the website reach a record of 1 billion active users on the platform simultaneously, then, in July 2018 it is been announced the first drop in active users in Europe and a global slowdown.
In this case study we are going to analyze the trend of the interest rate in Facebook during time[‡], see Figure 4.7: the interest rate expressed by Google Trends is given by a numeration between

[‡]According to Google Trends (n.d.), monthly interest in Facebook June 2023

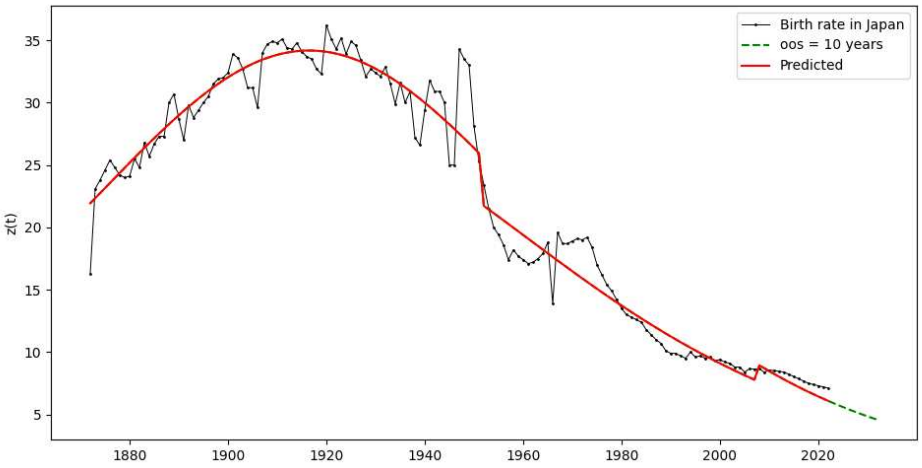

Figure 4.6: Instantaneous GBM 1 rectangular shock on Japan birth rate

0 and 100, where 100 indicates the higher frequency in term search and 0 means that no sufficient data for the term are being found.

Thus this is the trend in the search of the term "Facebook" on Google, it can be considered an indicator of public interest that provides information about trendy thematic over a given time span and can be used to study social phenomena or marketing campaign effects. As it is possible to notice, the trend in Figure 4.7 is characterized by: a sudden growth between April 2008 ad December 2010, a time window in which Facebook started to be translated into other languages and become accessible all over the world; then, the peak in interest is achieved in November 2012 and remained almost constant until July 2013, in this time window Facebook develops new features for mobile phones and holds a referendum over the users' privacy. After that, the rate of research on Google started dropping slowly, until reaching a value of 10 in May 2022, seen for the last time in June 2008. Anyway, the drop in Google's research does not mean any consequent drop in Facebook usage, in fact, it still is the most used social network worldwide, it may just indicate that the usefulness in the use in Google search for this topic dropped due to a multitude of factors such as the migration on mobile phones' app or the use of other sources to find Facebook's path and news. To fit the trend, a GBM with mixed shocks has been chosen, Table 4.3 and Figures 4.8 and 4.9 show the results. The parameters are all significant and the model is representative, with $R^2 = 0.9998$. Parameters $p = 0.0003$ and $q = 0.029$ indicates that the innovativeness factor poorly affected the trend compared to the imitative one, suggesting that the awareness about the Facebook phenomena has spread mostly through social interactions (by word-of-mouth, social media, or other sources of information).

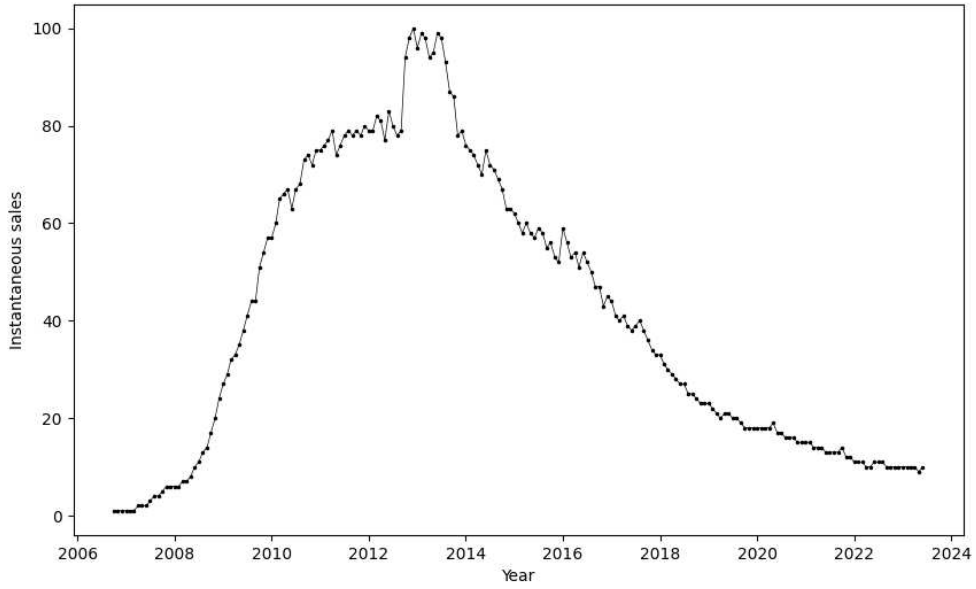

Figure 4.7: Facebook interest rate between 2006 and 2023

	Estimate	Std. Error	Lower	Upper
m	$8.4147e+03$	$7.6641e+00$	$8.3997e+03$	$8.4297e+03$
p	$3.0430e-04$	$2.0214e-05$	$2.6468e-04$	$3.4392e-04$
q	$2.9129e-02$	$2.6439e-04$	$2.8611e-02$	$2.9647e-02$
a_1	$2.3353e+01$	$4.8201e-01$	$2.2408e+01$	$2.4297e+01$
b_1	$-4.4364e-02$	$1.2534e-03$	$-4.6821e-02$	$-4.1907e-02$
c_1	$3.3255e+00$	$1.2195e-01$	$3.0865e+00$	$3.5646e+00$
a_2	$7.3105e+01$	$6.6341e-01$	$7.1804e+01$	$7.4405e+01$
b_2	$8.5560e+01$	$5.9050e-01$	$8.4403e+01$	$8.6718e+01$
c_2	$2.6464e-01$	$2.0635e-02$	$2.2420e-01$	$3.0508e-01$

$R^2 = 0.9998$

Table 4.3: GBM with 2 mixed shocks estimations on Facebook interest rate

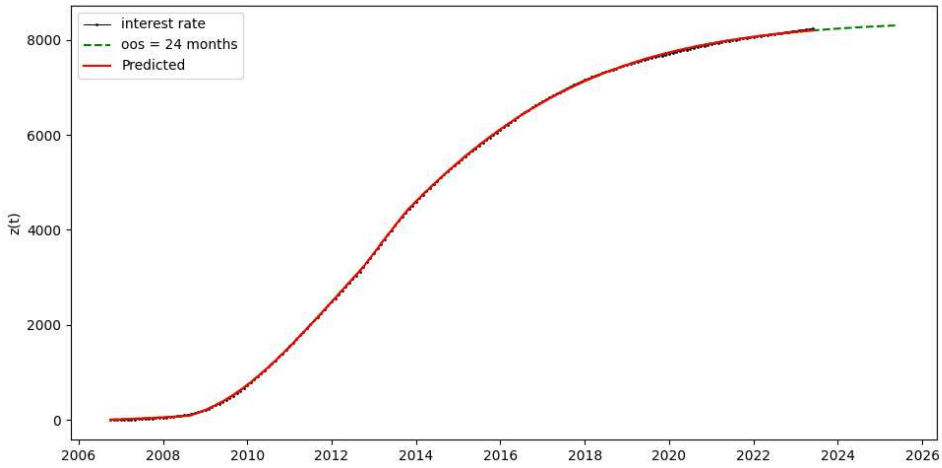

Figure 4.8: Cumulative GBM 2 mixed shock on Facebook interest rate

The trend is modeled using an exponential and a rectangular shock. The former is needed to describe the fast growth in the early stages of the life cycle, but due to the natural shape of the exponential shock it does not capture very well the outline of the trend at the beginning of the growth at $a_1 = 23.35$ (around late 2008), nevertheless, it is able to well describe the strong intensity of the shock $c_1 = 3.33$ and the slow overtake phase after the shock given by $b_1 = -0.044$, giving to the fitted data the left-skewed shape we can see in Figure 4.9. The latter shock is rectangular, with which we wanted to describe the peak between $a_2 = 73.11$ (late 2012) and $b_2 = 85.56$ (end of 2013) which is characterized by a fairly high intensity, i.e. $c_2 = 0.265$. In the next 24 months after June 2023, the model predicts the continuation of decay, till an interest rate of 3% in June 2025.

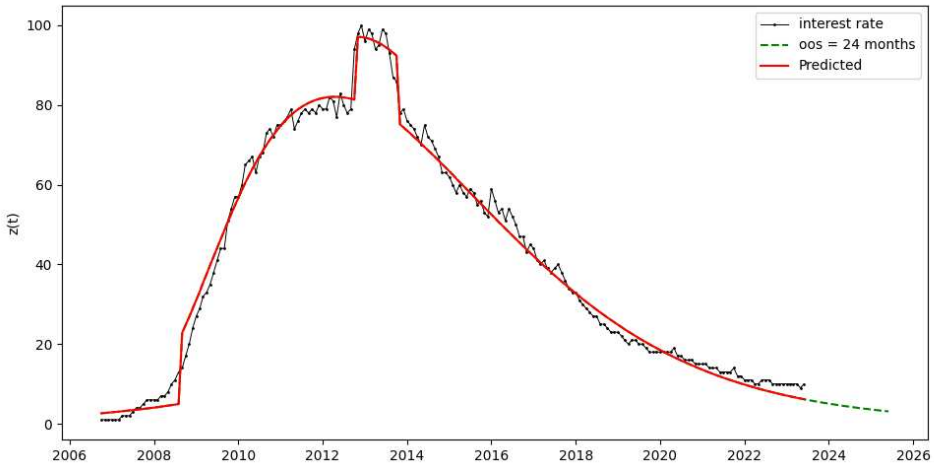

Figure 4.9: Instantaneous GBM 2 mixed shock on Facebook interest rate

4.3 Guseo-Guidolin Model

For the last univariate model, we go back into standard use to analyze the behavior of a notorious product that has struggled to achieve success, but that, thanks to effective marketing campaigns and successive word spreading has reached the top: the Apple iPhone, its trend perfectly represents the idea laying behind the creation of this model.

4.3.1 iPhone quarterly sales

In the history of smartphones, iPhone surely represented a turning point. However, even being a flagship product in this sector, its success was slow in coming, it was launched, in its version *2g*, in 2007 but it started gaining its very popularity with later versions, such as *iPhone 3g* and *4*. The success of the smartphone has been reached by a multitude of factors: its features were revolutionary at that time, such as the capacitive touchscreen interface, intuitive touch-based navigation, and sleek design, but it did not stop there, with the new iterations of the product, Apple packed every new product with substantial improvements; then, in 2008 it introduced the App Store that suddenly provided an abundance of apps, enriching the user experience; Apple's ecosystem was growing in the meanwhile, allowing the users to interconnect their devices; the last key factor it is been the effective marketing campaigns, compelling, to generate curiosity among consumers and creating an aura of exclusivity end desirability around the iPhone.
In this subsection, we analyze the quarterly sales of the Apple iPhone from its launch in 2007

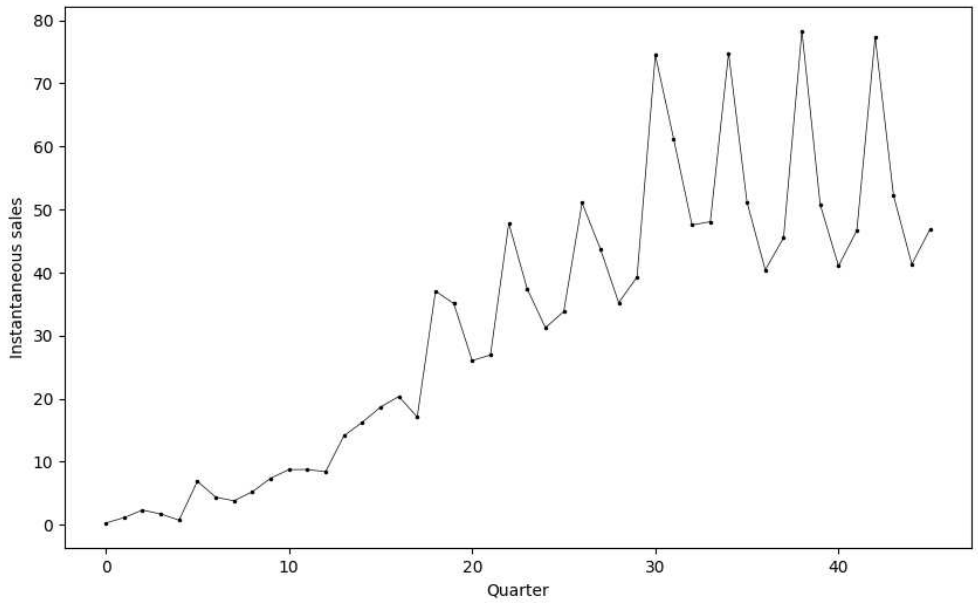

Figure 4.10: iPhone quarterly sales between 2007 and 2018

	Estimate	Std. Error	Lower	Upper
K	$1.9896e+03$	$1.2293e+02$	$1.7486e+03$	$2.2305e+03$
p_c	$9.4627e-03$	$1.3400e-03$	$6.8363e-03$	$1.2089e-02$
q_c	$1.4127e-01$	$1.8279e-02$	$1.0544e-01$	$1.7709e-01$
p_s	$3.8431e-04$	$1.5531e-04$	$7.9913e-05$	$6.8871e-04$
q_s	$1.3207e-01$	$1.5940e-02$	$1.0083e-01$	$1.6331e-01$

$R^2 = 0.9997$

Table 4.4: GGM estimations on iPhone quarterly sales

until the end of 2018, shown in Figure 4.10 in million pieces. As it is possible to notice, the trend in Figure 4.10, is characterized by slow growth until quarter 18, after that, it starts being characterized by seasonalities which peaks locates on 3rd quarter of each year. The instantaneous sales reach stationarity after quarter 30. In Table 4.4 and Figures 4.11 and 4.12 the results of a GGM over the iPhone's quarterly sales trend. This kind of trend, characterized by certain slowness in its growth, represents a good example of how the communication process affected the adoption by passing through the various versions of the product. All the parameters are significant and the model is representative, with an $R^2 = 0.9997$. In particular, it is possible to address the double effect the firm wanted to bring through its marketing strategies by looking at the innovation and imitation parameters. On the one hand, $p_c = 0.0094$ and

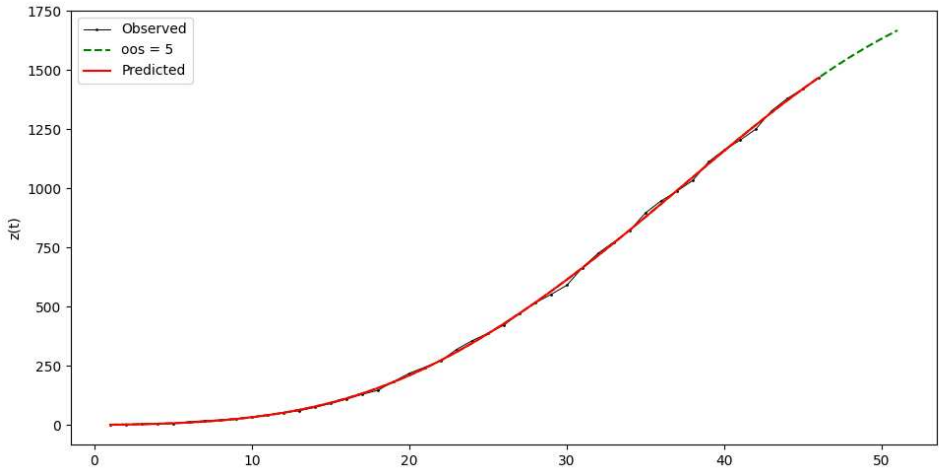

Figure 4.11: Cumulative GGM on iPhone quarterly sales

$p_s = 0.00038$ tell us that innovation occurred almost only during the communication phase, suggesting the effectiveness in creating curiosity before the launch of every product. On the other hand, it is possible to notice that imitations parameters are prominent, with $q_c = 0.14$ and $q_s = 0.13$ that give us a clue about the desirability that led people to will to be part of an exclusive group.

4.4 Unbalanced Competition Regime Change Diachronic model

For the UCRCD model, we choose an example that leverages the assumptions laying behind the creation of these models, the epidemiological diffusion. We are going to analyze the natural competition between a highly contagious virus, Covid-19 (one of its variants in particular), and the vaccines created to counter it.

4.4.1 The competition between Covid-19 and Anti Covid Vaccines

In this last case study, we are going to analyze the competition between the Covid-19 daily cases vs/ daily vaccination in Italy[29], in an eleven-month time window from August 1st, 2020 to July 1st, 2021, namely, from the month before the identification of the first main variant named Alpha (also called English variant, or B.1.1.7), characterized from the high virality, to the month in which the vaccinations reached the peak.

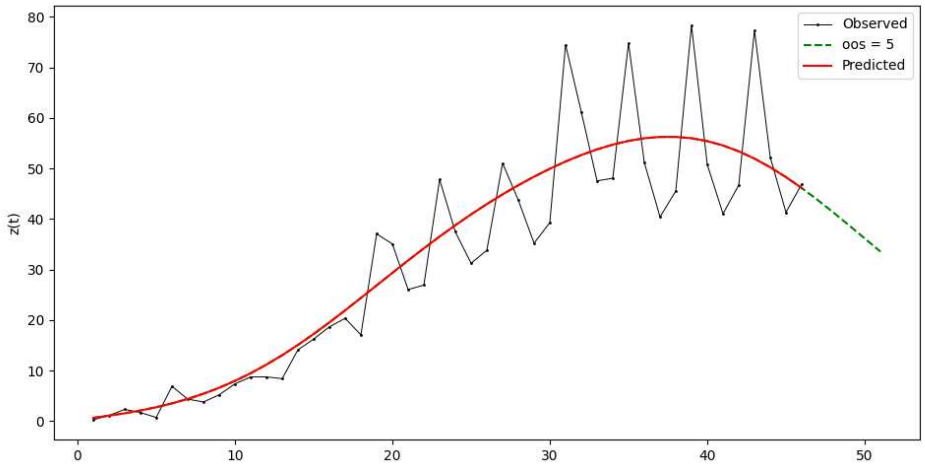

Figure 4.12: Instantaneous GGM on iPhone quarterly sales

The two series' trends are shown in Figure 4.13, a premise should be made. They are normalized on their respective maximum values on the considered time window, so that one could have an intuitive analysis, especially at the graphical level. This means that the values of the coefficients in Table 4.5 are slightly different in the estimation made on a proportional scale, yet they retain the same properties, significance, and signs on all the coefficients. The plot in Figure 4.13 shows the trends regarding the daily Covid-19 new cases (in black) and the daily vaccine doses administered (in grey), in Italy. Starting with the former, the first thing that can be noticed is the inconstancy of the trend, mostly given by the inconstant quarantine policies put in place and the people's behavior in the periods of non-quarantine the first growth is temporally placed after the summer reopening, characterized by high aggregations of people, after the first decrease of the cases there was a general loosening in restrictions, characterized by less aggregation, leading to the April peak. Since we considered a time window in which most of the cases are given by just the Alpha variant, so we consider an "almost-constant" virality. The latter series is characterized by a starting slow increase, until March, then a rapid growth until July 2021, which is like that because of the vaccination policies that provided for staggered vaccination starting with the highly frail, the elderly, and health personnel, and only then to the broad masses.

The effect of vaccinations is not fully shown in the plot, in fact, after the decrease of April-June, in Italy, there has been a reopening for the summer, characterized by a very low diffusion until January 2022 after the Beta variation diffusion. In Table 4.5 and Figures 4.14, and 4.15 are shown the results of the analysis over the series. The coefficients result to be all significant in

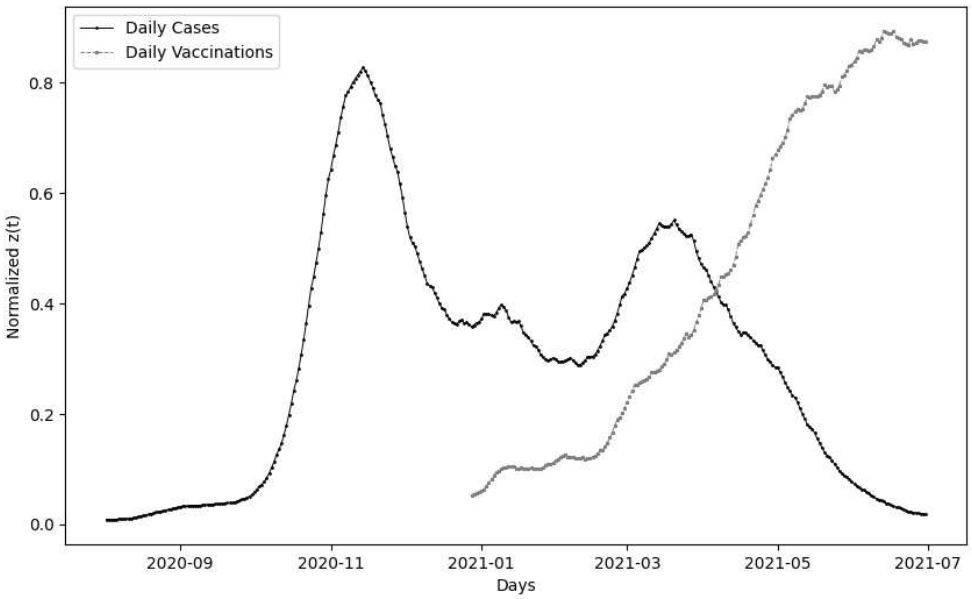

Figure 4.13: Italy Covid-19 Daily cases vs/ Daily vaccinations between Aug. 2020 and Jul. 2021

	Estimate	Std. Error	Lower	Upper
m_a	$4.2970e+01$	$2.3139e-01$	$4.2517e+01$	$4.3424e+01$
p_{1_a}	$2.3211e-05$	$1.2081e-06$	$2.0844e-05$	$2.5579e-05$
q_{1_a}	$7.5924e-02$	$6.7681e-04$	$7.4598e-02$	$7.7251e-02$
m_c	$2.3419e+02$	$3.7758e+00$	$2.2679e+02$	$2.4159e+02$
p_{1_c}	$-1.4371e-03$	$1.6836e-04$	$-1.7671e-03$	$-1.1071e-03$
p_2	$8.7556e-04$	$2.6192e-04$	$3.6220e-04$	$1.3889e-03$
q_{1_c}	$-1.7348e-02$	$7.1780e-04$	$-1.8755e-02$	$-1.5941e-02$
q_2	$4.6753e-02$	$2.4726e-03$	$4.1907e-02$	$5.1600e-02$
δ	$3.3854e-02$	$1.3792e-03$	$3.1151e-02$	$3.6557e-02$
γ	$5.0454e-02$	$3.6103e-03$	$4.3378e-02$	$5.7530e-02$

$R^2 = 0.9804$

Table 4.5: UCRCD estimations on Italy Covid-19 data

the confidence interval and the model is representative, with an $R^2 = 0.9804$. Since UCRCD is been created for marketing purposes, in this kind of analysis a different connotation of the parameters should be given.
Let us suppose that the disputed market potential ($m = m_a(1 - I_t > c) + m_c I_t > c$) represents humanity, then we have an initial phase in which the virus was free to diffuse, and a second phase when the vaccines penetrate the market, and have the aim to contrast the diffusion of the virus. Then, the significance and the behavior of the parameters make sense.
The first phase of no competition is characterized by an almost inexistent innovation $p = 0.00002$ and a very high imitation $q = 0.076$, a behavior that well describes the contagion diffusion. The second phase of competition, instead, shows an expansion of the market, $m_c = 234.19$, almost quintupled of the former phase, that is because in the former the infectious people were limited, thanks to restrictions and the use of protective equipment, while in this latter phase, vaccines contributed the most because of the implementation of a pressing vaccination campaign. The other parameters, then, describe perfectly the contrasting effects of the vaccines on the virus: the increasing adoption described by $p_2 = 0.0088$ and $q_2 = 0.047$, of the former, coincides with a negative adoption of the latter, which registers a $p_{1_c} = -0.0014$ and a $q_{1_c} = -0.017$, these last parameters actually describe the reduction of the viral load and the reduced transmission provoked by the vaccines. The within imitation coefficient confirms what just said, with $q_{1_c} + \delta = 0,0165$, indicating the slowdown effect on the virus. A negative cross imitation coefficient, given by $q_2\gamma = -0.0037$ wrongly suggests a negative effect on both trends, this is probably given by the slow increase described by the model (see Figure 4.15) in both until the moment when they cross.

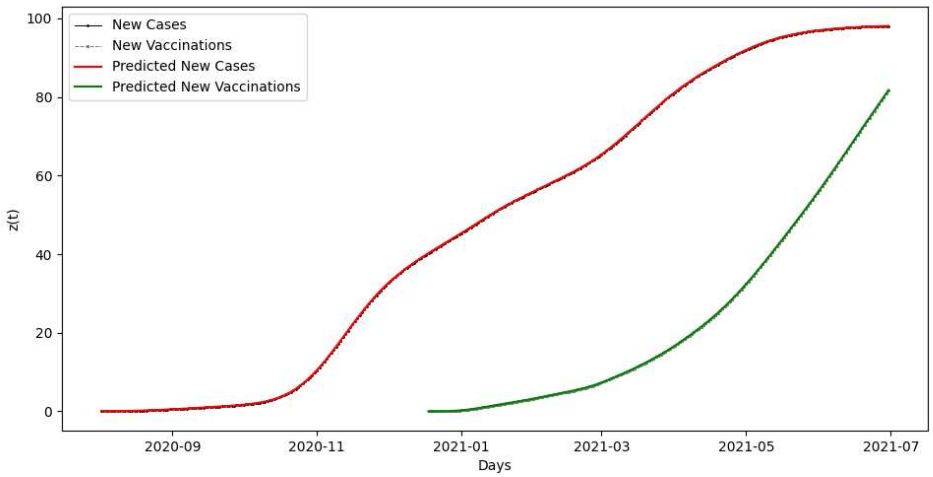

Figure 4.14: Cumulative UCRCD fit on Italy Covid-19 data

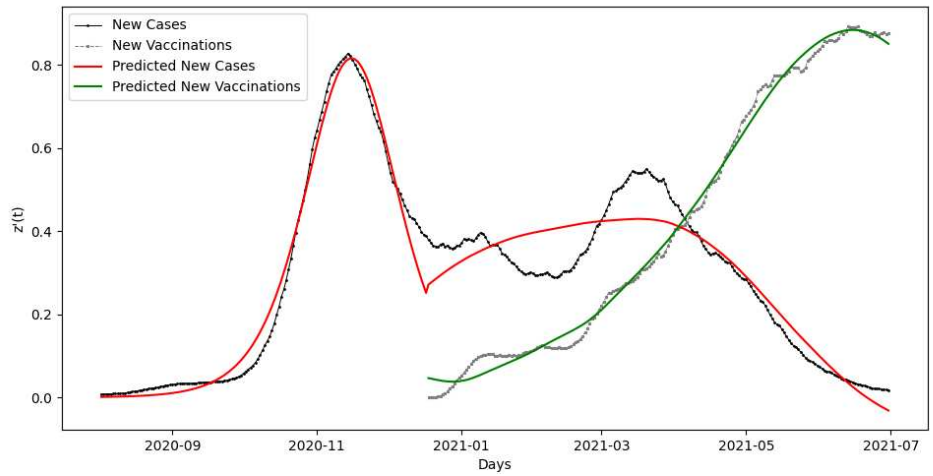

Figure 4.15: Instantaneous UCRCD fit on Italy Covid-19 data

5

Conclusion and Future work

The aim of this thesis project was to present the implementation of PyDiM, a Python library for Innovation Diffusion Analysis which takes inspiration from R's package "DIMORA" [20], to provide the Python community with a valuable option over these kinds of approaches, and filling a gap in the data analysis tools landscape.

At the time of this thesis writing, PyDiM implements four models: the standard Bass Model [3] who laid the foundations of modeling regarding innovation diffusion processes, giving inspiration for most, if not all, modeling approaches in this area developed to date; the Generalized Bass Model [9], a generalization developed by the same BM's author to overcome the lacking of the former over basic economic variables needed to describe, the so-defined, *carryover effects* that can present in the life cycle of any economic diffusion process; the Guseo-Guidolin Model [10], a model belonging from a branch of approaches which generalize the BM under hypotheses of dynamic market behaviors influencing the adoption of innovations, this in particular accounts for the presence of *structured shocks* having different nature, and that can suddenly influence the dynamics of a market with different intensities and duration; the Unbalanced Competition Regime Change Diachronic Model, the last implemented model, which differentiates from the others due to the fact that it is a bivariate model that analyzes the competition between two products on a diachronic regime.

The flow of this work started from the history and etymology of the innovation, the innovation diffusion until arriving at the modeling approaches, which aim to analyze and predict the process of the innovation diffusion, and the relative literature. Then, a theoretical and math-

ematical explanation of the implemented models has been given, followed by an overview of the practical implementations, highlighting the basic differences, between this library and R's DIMORA from a programming point of view. In the end, each model's functioning is displayed and commented on, demonstrating the usefulness of these approaches in various fields, also very different from marketing.

In the future, we aim to optimize and simplify the implementation even more, together with enlarging the number of provided approaches to render PyDiM an even more tools comprehensive library for Innovation Diffusion Analysis able to fill the gap, in the current Python landscape, of these kinds of modeling approaches and eventually become a landmark for the field.

Developing Data Science and Analysis tools is for all intents and purposes a very important task in the field, besides, "*tools do not determine success, but without the right tools, success is more difficult to achieve.*" - cit. Donald A. Norman.

References

[1] B. Godin, "Innovation: the history of a category," 2008.

[2] S. Gilfillan, "Colum. 1935. the sociology of invention," *Chicago: Follet*.

[3] E. Rogers, *Diffusion of Innovations*. Free Press of Glencoe, 1962. [Online]. Available: https://books.google.it/books?id=zwo-AAAAIAAJ

[4] P.-F. Verhulst, "Notice sur la loi que la population suit dans son accroissement," *Correspondence mathematique et physique*, vol. 10, pp. 113–129, 1838.

[5] A. McKendrick and M. K. Pai, "Xlv.—the rate of multiplication of micro-organisms: a mathematical study," *Proceedings of the Royal Society of Edinburgh*, vol. 31, pp. 649–655, 1912.

[6] E. Mansfield, "Technical change and the rate of imitation," *Econometrica: Journal of the Econometric Society*, pp. 741–766, 1961.

[7] M. Guidolin and P. Manfredi, "Innovation diffusion processes: Concepts, models, and predictions," *Annual Review of Statistics and Its Application*, vol. 10, 2023.

[8] F. M. Bass, "A new product growth for model consumer durables," *Management science*, vol. 15, no. 5, pp. 215–227, 1969.

[9] F. M. Bass, T. V. Krishnan, and D. C. Jain, "Why the Bass model fits without decision variables," *Marketing science*, vol. 13, no. 3, pp. 203–223, 1994.

[10] R. Guseo and M. Guidolin, "Modelling a dynamic market potential: A class of automata networks for diffusion of innovations," *Technological Forecasting and Social Change*, vol. 76, no. 6, pp. 806–820, 2009.

[11] R. Guseo and C. Mortarino, "Within-brand and cross-brand word-of-mouth for sequential multi-innovation diffusions," *IMA Journal of Management Mathematics*, vol. 25, no. 3, pp. 287–311, 2014.

[12] R. Guseo, "Strategic interventions and competitive aspects in innovation life cycle," in *Working Paper Series, N. 11*, 2004.

[13] V. Mahajan, E. Muller, and F. M. Bass, "New product diffusion models in marketing: A review and directions for research," *Journal of marketing*, vol. 54, no. 1, pp. 1–26, 1990.

[14] M. Guidolin, *Innovation Diffusion Models: Theory and Practice, First Edition*. John Wiley Sons Ltd., 2023.

[15] W. M. Cohen and D. A. Levinthal, "Absorptive capacity: A new perspective on learning and innovation," *Administrative science quarterly*, pp. 128–152, 1990.

[16] N. Boccara and N. Boccara, *Modeling complex systems*. Springer, 2010, vol. 1.

[17] F. Petropoulos, D. Apiletti, V. Assimakopoulos, M. Z. Babai, D. K. Barrow, S. B. Taieb, C. Bergmeir, R. J. Bessa, J. Bijak, J. E. Boylan *et al.*, "Forecasting: theory and practice," *International Journal of Forecasting*, 2022.

[18] M. Guidolin and T. Alpcan, "Transition to sustainable energy generation in australia: Interplay between coal, gas and renewables," *Renewable Energy*, vol. 139, pp. 359–367, 2019.

[19] A. Bessi, M. Guidolin, and P. Manfredi, "The role of gas on future perspectives of renewable energy diffusion: Bridging technology or lock-in?" *Renewable and Sustainable Energy Reviews*, vol. 152, p. 111673, 2021.

[20] F. Zanghi, A. Savio, F. Ziliotto, and A. Bessi. (2021) DIMORA: Diffusion models r analysis. [Online]. Available: http:CRAN.R-project.org/package=DIMORA

[21] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[22] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[23] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[24] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.

[25] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977*. Springer, 2006, pp. 105–116.

[26] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on scientific computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[27] M. Roser, "Fertility rate," *Our World in Data*, 2014, https://ourworldindata.org/fertility-rate.

[28] J. F. Steiner, "Japan's post-war population problems," *Social Forces*, vol. 31, no. 3, pp. 245–249, 1953. [Online]. Available: http://www.jstor.org/stable/2574222

[29] E. Mathieu, H. Ritchie, L. Rodés-Guirao, C. Appel, C. Giattino, J. Hasell, B. Macdonald, S. Dattani, D. Beltekian, E. Ortiz-Ospina, and M. Roser, "Coronavirus pandemic (covid-19)," *Our World in Data*, 2020, https://ourworldindata.org/coronavirus.

Acknowledgments

This is the acknowledgments section.