Modeling and Markov chains

Philippe Cohard

Modeling and Markov chains

Philippe Cohard
Montpellier Research in Management – MRM, University of Montpellier,
Montpellier Management, Labex Entreprendre, Montpellier, France
philippe.cohard@umontpellier.fr

Abstract

Modeling and simulation are now topics with major interest for researchers and practitioners. Markov chain is a powerful approach of modeling that could be used in multiple areas. For example, in management this approach can give numerous opportunities to better understand phenomena. Thus we ask the question: "How can Markov chains be useful for modeling processes? ". This paper is our answer. With Markov chains we show the possibilities to better understand management project. We also propose two programs which explain how to implement these models based on Markov chains. The results show that Markov chains approach of modeling can be useful for teaching and explaining processes. Modeling gives data to analyze what can be compared with empirical data or tested for coherence. Modeling with Markov chains can be interpreted and give insights on real causes of complex phenomena.

Keywords*: Markov chains, modeling, process, simulation, program*

1. Introduction

Markov chain is a probabilistic approach that gives computationally fast results. Modeling can be done with the Markov chains. This kind of approach has been used in many fields, for explanation of behaviors, cycles of a machine, biology, etc. Creating models from Markov chains is simple, the matrix is just an integration of all values of the model. Markov chains make it possible to model probabilistic systems as changes from one state to the next where the probability is only dependent on the current state, not the previous ones (Grinstead and Snell, 2003).

There is an important number of libraries for Markov chains in PYTHON. The "pip search markov" command gives a great number of outputs (more than 100). There are dozens of libraries specifically on Markov chains. Many are for specific use like speech generation. But in this paper we will use only numpy library. We intend to use these chains for calculus because, we want to explain the capacities of these chains with PYTHON. In this paper we shall use PYTHON 2.7.

Andreï Markov studied the chains of events around the 1900s and published a first paper in 1906 (Senata, 2006). These events are now named Markov chains because he established laws and proposed the foundations of the Markovian process. A characteristic of a Markov chain is that: *"it is historyless in that the next state depends only on the current state, not on any prior ones"* (Hefferon, 2017, p.306). Representing Markov chains can be done in the

form of a diagram or in the form of a matrix. This kind of diagram is interesting to easily show how the process works. This is why we have some diagrams, called transition diagrams in our paper. But in our program we need to use the matrix form which is another way to present the same thing.

We present three types of Markov chains: (1) regular Markov chain, (2) absorbing Markov chain and (3) ergodic Markov chain. For the regular Markov chain: *"long-range predictions are independent of the starting state"* (Grinstead and Snell, 2003, p.407). Therefore, after numerous steps the probability distribution is stable (convergence). We propose the example below that shows the probabilities of the weather in a simulated world. This chain has 3 states: rainy, cloudy and sunny.

$$P = \begin{pmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 0.0 & 0.5 \\ 0.25 & 0.25 & 0.5 \end{pmatrix}$$

To simulate the next days, we calculate the successive powers of P from 1 to 9 (the program is provided in appendix 1). Then we can say that after nine days the predictions are independent of the starting state (today weather).

$$P^6 = \begin{pmatrix} 0.316 & 0.263 & 0.421 \\ 0.316 & 0.263 & 0.421 \\ 0.316 & 0.263 & 0.421 \end{pmatrix}$$

This way we found the limiting vector but it can be calculated solving the following equations:

$$\begin{aligned} v1 + v2 + v3 &= 1 \\ (1/4)v1 + (1/2)v2 + (1/4)v3 &= v1 \\ (1/4)v1 + (1/4)v3 &= v2 \\ (1/4)v1 + (1/4)v2 + (1/2)v3 &= v3 \end{aligned}$$

The limiting vector is in accordance with the previous results with the algorithmic approach of successive powers.

$$V = \begin{pmatrix} 0.316 & 0.263 & 0.421 \end{pmatrix}$$

Another kind is the absorbing Markov chain, which has at least one absorbing state with probability at 1. So when this state happens, it is impossible to leave it. Grinstead and Snell (2003) precise that in an absorbing Markov chain, a state which is not absorbing is called transient. This kind of Markov chain is particularly useful to answer the question of the

probability that the process eventually reaches an absorbing state. The following example is about a mission order. The proposition of mission order is sent by the employee to his supervisor. If this latter doesn't agree, he sends the order back to the employee (0.7). But if he agrees, he sends the order to the director which signs the order (0.3). There is no further step in this process (example below).

$$P = \begin{pmatrix} 0.0 & 1.0 & 0.0 \\ 0.7 & 0.0 & 0.3 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$$

We calculate the successive powers of P. The values of the matrix converge to a steady state value by P^{44} (Markov chains converge on a value, but computationally, what seems to be no change is frequently that the convergence is within the digits of precision that is printed or that the computer is capable of).

$$P^{44} = \begin{pmatrix} 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$$

As the state "1.0" appears, it represents a long run absorbing state.

Another kind of Markov chain is the Ergodic chain in which it is possible to go from every state to every state. Note that every regular chain is ergodic but every ergodic chain is not mandatory a regular one. We propose an example (corrected 11/07/2018): the tourist walk, inspired of Grinstead and Snell (2003). The tourist strolls in the city, without any precise goal. He continues walking until he reaches the tourism office (4), or his hotel (0). Where he can stay or come back.

The transition matrix is then:

$$P = \begin{pmatrix} 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.5 & 0.0 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.0 & 0.5 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \end{pmatrix}$$

With Markov chains we can model numerous systems: naturals systems like viruses or artificial systems like computer programs, machines, supply chains, etc. But we can model behaviors too. For example, the behaviors of peoples, of managers/workers or the processes in a company. Modeling is one of the uses of Markov chains. Once you have a good model of a process, you can use it to explain something or use it in a simulation engine. This kind

of engine can be used in computer games and of course in serious games as well. Thus we ask the question: “How can Markov chains be useful for modeling processes? “. We particularly focus on the reason why this can be useful for the manager. This article is our answer to the question and gives an insight using PYTHON.

2. The process for setting Markov chains

In this section we first present modeling and Markov chains (2.1), and then the settings that can apply Markov chains (2.2). The interest of the manager is to model the changing state of a system based on probabilities.

2.1. Modeling and Markov chains

Simulation is a well-known method for research based on abstraction models and computer programs. Harrison et al. (2007, p. 1231) explain that “*in the leading management and social science journals, about 8 percent of the published papers used simulation methodology”*. The simulation approach is quite developed in “the academy of management review”. Davis et al. (2007) explain how to use simulation methods for theory development. These authors suggest a 7 points road map for developing theories with simulation: (1) begin with research question, (2) identify simple theory, (3) choose a simulation approach, (4) create computational representation, (5) verify computational representation, (6) experiment to build novel theory and (7) validate with empirical data. Simulation and modeling can give useful insights to researchers about events, social systems, processes in an exploratory approach. Furthermore, Harrison et al. (2007) proposed a process of management theory and simulation modeling which shows the phases of such a research (figure 1).

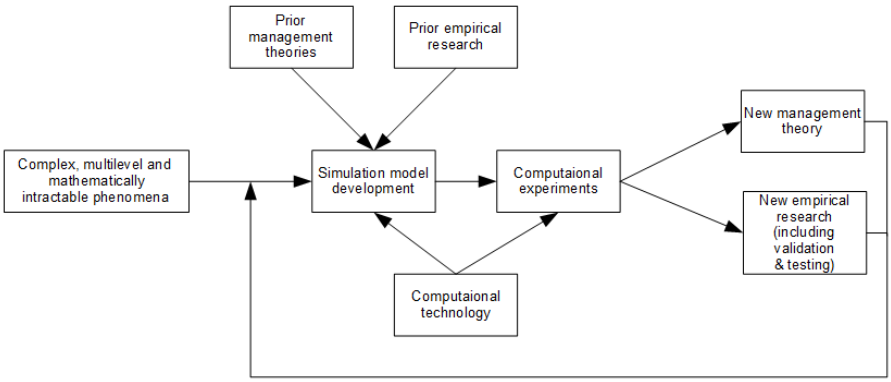

Figure 3. The interactive process of Harrison et al. (2007)

For example, Leroux and Berro (2010) in their exploratory simulation analyze the mutual adaptation dynamics developed by the agents of bioclusters.

Markov chains are used in different types of researches. Al-Sayed et al (2016) used a Markov chain model monitoring the state of cloud resources. In another kind of research, Andernsen et al. (2017), used Markov chains models to optimize ward resources and patients’ relocation.

The Markov chains theories are clear and easy to access. It should be an interesting approach which is not protected by imposing technicalities and provides computationally fast results (Brémaud, 1999). The chain can be represented by a diagram of transitions. In the diagram, the sum of the probabilities from one state to each of the other states must equal 1. The process starts in one state and moves, by step, from this one to another. *"If the chain is currently in state s_i, then it moves to state s_j at the next step with a probability denoted by p_{ij}, and this probability does not depend upon which states the chain was in before the current state."* (Grinstead and Snell, 2003, p.405-406). We will study a fictional example with diagram: the stream pattern of students on internet. It should be noted that this chain is a regular Markov chain.

In a course of computer science, researchers observe the stream pattern of students working on a case study. They find the following diagram:

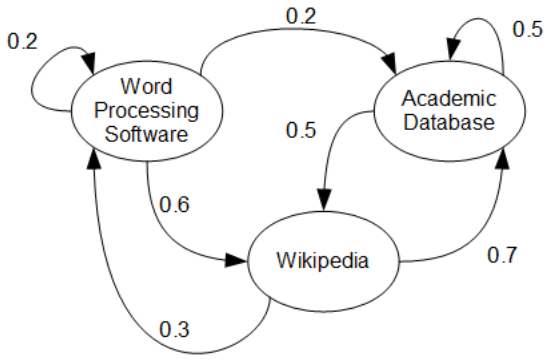

Figure 2. Markov chain of students working on case studies

When the students begin, they start on the word processing. They have a probability of 0.2 to stay on this software, 0.6 going on Wikipedia and a probability of 0.2 visiting the academic database to find research articles. When students are on Wikipedia they have a probability of 0.7 accessing the academic dataset because they found references, and 0.3 returning to the word processor software to develop their answer. When students are on the academic database there is equi-probability 0.5 that they stay on the academic database or that they access Wikipedia to have an explanation of a concept. Word processing: 0.2+0.6+0.2 =1; Wikipedia: 0.3+0.7=1; Academic database: 0.5+0.5=1).

We represent this diagram in the form of a matrix, where rows represent the current state and columns represent the next period state. The first row and column represent use of Word Processing software, the second use of Wikipedia, and the third, the use of academic databases.

$$P = \begin{pmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0.0 & 0.7 \\ 0.0 & 0.5 & 0.5 \end{pmatrix}$$

To see the state of the Markov chain after 5 steps we just have to calculate the 5-th exponent of the P matrix. Now we can do a PYTHON program to simulate the Markov chain based on the matrix. This was our introductive example. In the next section we are working with a more developed one.

2.2. Markov Chain project management example model

To start with we have to decide of the rules of the world we want to simulate. To do so in the research process we have to search over the literature or use our main research results. We are testing a model on project management. The outcome is to have simulation inspired from reality to simulate possible worlds. We can possibly use it for educational ends or else to develop thinking about the project cycle.

We use a partly iterative project management cycle proposed by Duncan (1993): initiating, planning, executing, controlling, closing. The planning, executing and controlling are iterative phases. The number of phases and their name can vary. But they generally stay in the same concepts like for example Khang and Moe (2008): conceptualizing, planning, implementing and closing. For our model the iterative nature of these kind of project is particularly interesting because the end of the process is not clearly defined at start. Iterative project life cycle is perceived as more robust because of the possibility of reorganizing the project (de Blois, Lizarralde and De Coninck, 2016), particularly on design phase. In this proposition, the ratios are from a study named CHAOS research that *"encompasses 18 years of data on why projects succeed or fail representing more than 90,000 completed IT projects"* (Standish Group, 2013, p. IV). Please note that this model is an important simplification of reality, but that can give useful insights. There are great differences between ratios of small and big projects. We work only on the small projects in this article, but comparing modeling between big and small projects could be interesting too. The settings for small projects are: 76% succeeding (on time delivery, on budget, with required features), 20% challenged (late, over budget, or with less required features), and 4% failed (cancelled before completion or delivered but never used) (Standish Group, 2013).

The proposed Markov chain is presented in figure 3. A project can be reorganized and then it can be a success or be cancelled. This adjustment is perceived in our modeling as a change on planning, then the project is challenged but not closed. Only success or cancellation can capture the token and stop the process (the sum of every row must be equal to 1): it's an absorbing Markov chain. Initiating, planning and executing are controlled phases in the model (marked with 1).

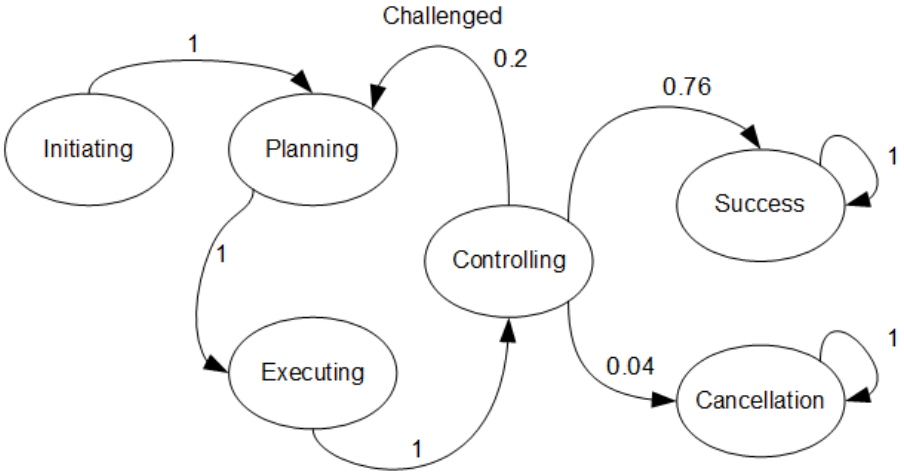

Figure 3. Proposed Markov chain for Management of small projects

Challenge, success and cancellation respect the ratios proposed by the study previously mentioned.

The matrix of the proposed Markov chain of the project management cycle is the following:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0.2 & 0 & 0 & 0.76 & 0.04 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

An initial state vector (probability vector) is used in our program to consider the long term behavior of the Markov chain starting in a chosen state. A probability vector is a row vector of non-negative values which sum to 1 (Grinstead and Snell, 2003). We set it to start at the initiating phase. The variable of the initial state vector is `initia` in the program. This way we consider the long-term behavior of the Markov chain in a chosen state. Note that the first state is mandatory for this kind of chain: we need to start by initiating the project.

We are showing now how to program this model with PYTHON.

3. Implementing the project management Markov Chain model

We are using PYTHON 2.7 for programming our Markov chain (it works with PYTHON 3 too). We are using the library numpy which allows us simple manipulation of the matrix. We use the IDLE code editor to code in this paper.

The steps to go from the mathematical description of a Makov chain to the python program are as follows. First, create the matrix `P` as a numpy object. Then in the same way create the initialization matrix `initia`. Each iteration varies the value of `step` which corresponds to the value of the exponent. Finally execute the calculation: raise `P` to the `step` power and multiply by `initia`.

We suggest the following code to analyze the results after 11 steps for 1 project.

The codes in this article are experimental, given under BSD license, with no warranty.

```
import numpy as np      #Numpy library
#Set the P matrix
P = np.matrix([[0.0, 1.0, 0.0, 0.0, 0.0, 0.0],
               [0.0, 0.0, 1.0, 0.0, 0.0, 0.0],
               [0.0, 0.0, 0.0, 1.0, 0.0, 0.0],
               [0.0, 0.2, 0.0, 0.0, 0.76, 0.04],
               [0.0, 0.0, 0.0, 0.0, 1.0, 0.0],
               [0.0, 0.0, 0.0, 0.0, 0.0, 1.0]])

# starting state : initiating
initia = np.matrix([[1.0, 0.0, 0.0, 0.0, 0.0, 0.0]])

# Treatment of the data for 11 phases
for step in range(11):
    result = initia * P ** step
    print(result)
```

At the end of the execution, the printed results are:

```
[[1. 0. 0. 0. 0. 0.]]                          # 1 st month
[[0. 1. 0. 0. 0. 0.]]                          # 2 nd
[[0. 0. 1. 0. 0. 0.]]                          # 3 rd
[[0. 0. 0. 1. 0. 0.]]                          # 4 th
[[0.   0.2  0.   0.   0.76 0.04]]              # 5 th
[[0.   0.   0.2  0.   0.76 0.04]]              # 6 th
[[0.   0.   0.   0.2  0.76 0.04]]              # 7 th
[[0.    0.04  0.    0.    0.912 0.048]]        # 8 th
[[0.    0.    0.04  0.    0.912 0.048]]        # 9 th
[[0.    0.    0.    0.04  0.912 0.048]]        # 10 th
[[0.     0.008  0.     0.     0.9424 0.0496]]  # 11 th
```

We can interpret these results. If we have a fix duration of one month per phase we can say: the first month there are no doubts that the project is in Initiating phase. Likewise, we are almost sure that the project will be in Planning phase the second month, in the Executing phase the third month and in the Controlling phase the fourth month. The fifth month everything can change just like it has been coded in our model. We retrieve the probabilities linked to the ratios. It's what happens next, the sixth month, which is interesting: we stay again in the loop of "planning, executing, controlling" or more likely the project is caught by Success (0.9424) or Cancellation (0.0496). It is possible that the project loop into Planning but it is not likely (0.008). That's consistent with our model and its settings.

With this program we can test a large number of steps. What happens if we test on 1000 steps? We can see that the algorithm converges, the result becomes stable and do not vary. With round at 3 decimals we see the convergence on the 17th step, with 0.95 for success and 0.05 for cancellation (figure 4). To do so you may change the number of steps and add: `result=np.asarray(result)` and `result=np.round(result, decimals=3)`before printing result.

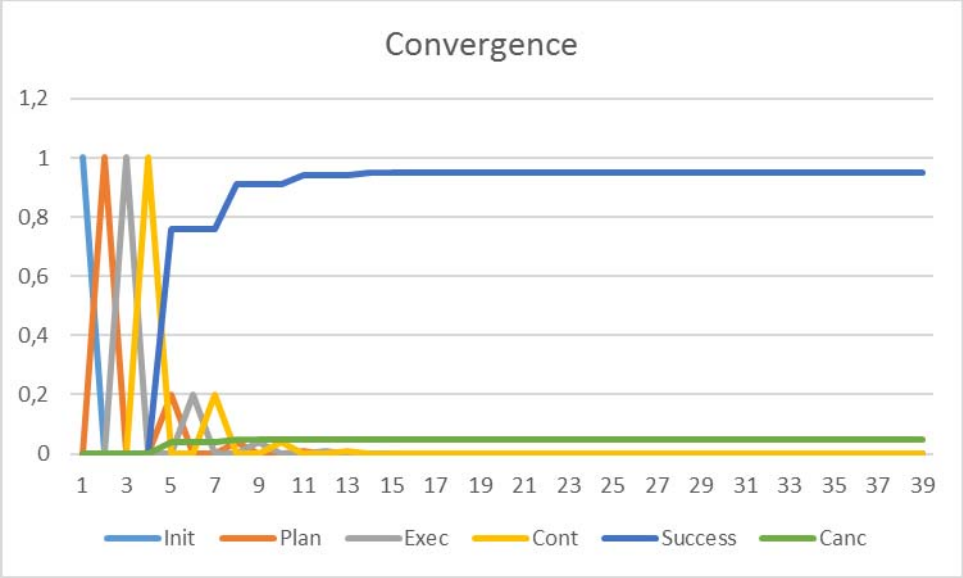

Figure 4. Convergence on the 17th step

With this modeling we can see that at the end there are only two possibilities: success or aborting. After a certain amount of times things become binaries at the 17th steps. It is likely that the project ends. This shows that most of Markov chains of this kind tend to converge to a long run steady state probability. Moreover, we can use that simulation in software and particularly in serious games or for educational purpose.

4. Using Markov chain model to analyze output

Now we have to generate data and show something. We propose to simulate the draws: generating the probabilities and showing the different values.

4.1. Design of the solution

We propose to do 20 steps in our Markov chain, to simulate draws. The step in which you are gives the entire set of possibilities for this step. This draw is done by a list of a thousand values that represent the probabilities of the line of the matrix.

In this list we load the line needed and draw a value. These consecutives draws can be represented as a tree. At the end the token can iterate in the planning phase or can be captured definitively in success or cancellation. We choose this approach because it seems to be more realistic.

To show the results we will create a CSV (Coma Separated Values) file with all values of the draws, 20 for a project (represented by the steps) and simulate it for 100 projects. There is a great number of combinations on the 20 steps by project. To do the draw we select randomly a number inside the list. Then we use the random library to generate this random number.

Using the CSV results can be interesting for educational purpose. The students may interpret the various possibilities. This puts in light the fact that a complex and long project can fail in the end. Although a long project can be adjusted during the process and finally can be successful. For a serious game use, two approaches are possible: the dynamic one and the static one. With the dynamic one we can integrate the program into the game and each time generate a “story of the project”. So that the story is generated dynamically and it may be different each time. With the static approach we can generate a large number of solutions, select the many best ones, and store them into a file. This way you don’t have to integrate the program, just a load CSV file. That last manner can be a solution for placing the learner in a specific situation.

4.2. Programming the designed solution

We develop two functions: `createEns` that use a matrix in input and `executeDraw` which use in input the number of STEPS, of type integer. This program developed in PYTHON 2.7 uses the numpy and the random library. We define two global variables STEPS for the number of steps which represents the draws for a project and ITERA for the number of iterations which represents the number of projects. The function `createEns` is designed to return a set `macroEnsemble` that is a list which contains lists of draws. The function `executeDraw` proceeds to the draw between the 1000 values. That function formats the result in CSV in the return variable `cumul`.

```
import numpy as np
import random as random
STEPS=20 #number of STEPS
ITERA=100 #number of iterations

def createEns(mat):
    i=0
    sizeP=np.shape(mat) #size of the square matrix
    ensemble=[]     # to stock the 1000 values
    macroEnsemble=[] # list of lists
    for val in mat:
        while i < sizeP[0]:
            valeur =val.item(i)* 1000
            ensemble += int(valeur) * [i]
            i=i+1
        #use of list reference byval
        macroEnsemble.append(list(ensemble))
        i=0
        del ensemble[:]
    return macroEnsemble

def executeDraw(STEPS, macroEns):
    j=1
    draw=0 #related to initial state vector
    cumul=""
    while j<=STEPS :
        draw = random.choice(macroEns[draw]) #start at 0
        print("STEPS "+str(j)+" : "+ str(draw))
        if j == STEPS:                # manage last separator
            cumul= str(cumul) + str(draw)
        else :
            cumul= str(cumul) + str(draw) + ";"
        j=j+1
    cumul=cumul+"\n"
    return(cumul)

# main program
if __name__ == '__main__':

    P = np.matrix([[0.0, 1.0, 0.0, 0.0, 0.0, 0.0],
                   [0.0, 0.0, 1.0, 0.0, 0.0, 0.0],
                   [0.0, 0.0, 0.0, 1.0, 0.0, 0.0],
                   [0.0, 0.2, 0.0, 0.0, 0.76, 0.04],
                   [0.0, 0.0, 0.0, 0.0, 1.0, 0.0],
                   [0.0, 0.0, 0.0, 0.0, 0.0, 1.0]])

    initia = np.matrix([[1.0, 0.0, 0.0, 0.0, 0.0, 0.0]])
    macroEns=createEns(P)
    k=0
    f = open("Output.csv","w") #open file
    while k<ITERA:
        retour=executeDraw(STEPS, macroEns)
        f.write(retour)
        k=k+1
    f.close()
```

The principal program is operationalizing the matrix and coordinating the two functions presented above. The results of the simulation of the second program is a spreadsheet file which contains 20 columns that represent the successive draws and 100 rows that represent the projects.

5. Results

The first program, the long run behavior of the Markov chain, started in a chosen state: an initial state vector. This way we can simulate the future on the basis of a model. The model under the Markov chain is a reduction of reality and that is a limit. But it can capture important probabilities between the elements of the chain. We should consider our approach as reflecting possibilities rather than reality. Although a well configured chain could be a very useful decision support tool or an interesting learning tool.

In the second program, the simulation shows with our model that projects can finish with different results and moreover at different speeds. We can have projects that need more phases on planning – executing – controlling (A and D) or project that are successful very quickly (like E). There are projects that fail quickly (B) or take more time (C). We tested them with 5000 samples and we have found coherent ratio for projects succeeded 95.04% and failed 4.96% at the end (theoretical 76/80 and 4/80, 95% and 5%).

Project	**Draws**																			
A	1	2	3	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4
B	1	2	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
C	1	2	3	1	2	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5
D	1	2	3	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4
E	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

Table 1. Extract form the spreadsheet file of results.

6. Discussion

The main question of our research is: “How can Markov chains be useful for modeling processes? “. For research the utility can be seen from the validity of the results and their use. Cartier (2007) precise that validity of the results of a simulation is based on three elements: intern validity, ability of the model to be conform with the reality, and coherence of the model with other models. Moreover, the question refers to the epistemology and more precisely to the status of the researcher.

From a first point of view on simulation and modeling, the research can be epistemologically positioned in post-positivism, more precisely in critical-rationalism of Karl Popper (1979). Hypothesis and theories from the model can be seen as the reflect of the objective reality

outside of the subjects. In an exploratory research we can do the model from the literature and then propose a theory. This latter can be tested like demanded by the falsificationism. For this school of thought there is no way to find whether a theory is true but we can prove that a theory is false. Theories can never be accepted but corroborated.

From a second point of view on simulation and modeling, the research can be epistemologically positioned in constructivism, constructing an artificial world of interpretation that gives useful insights. The hermeneutic approach can give significations from the simulated material to suggest possible underlying causes. That can be enriched by empirical material like suggested in the roadmap of Davis et al. (2007) and the process of Harrison et al. (2007).

7. Conclusion

Modeling and simulation are now topics with an increasing interest for researchers and practitioners. Markov chain is a powerful approach of modeling that could be used in multiple area. How can Markov chains be useful for modeling processes? It's the question answered in this paper. Markov chain is a quite easy way for modeling a complex phenomenon and simulate it through a simple computer program. We showed that Markov chains can be represented graphically and quickly converted in a matrix and after programmed with PYTHON. The quality of the model is based on prior phenomena theories and prior empirical researches implemented in the program. Modeling is a simplification of reality in an artificial world. We gave an example for project management. Markov chains can be useful to treat data (modeling) and to design simulations games or serious games.

References

Al-Sayed, M. M., Khattab, S. and Omara, F. A. (2016). Prediction mechanisms for monitoring state of cloud resources using Markov chain model. *Journal of Parallel and Distributed Computing*, 96, pp. 163–171.

Andersen, A. R., Nielsen, B. F. and Reinhardt, L. B. (2017). Optimization of hospital ward resources with patient relocation using Markov chain modeling. *European Journal of Operational Research*, 260(3), pp. 1152–1163.

de Blois, M., Lizarralde, G. and De Coninck, P. (2016). Iterative Project Processes Within Temporary Multi-Organizations in Construction. *Project Management Journal*, 47(1), pp. 27–44.

Brémaud, P. (1999) *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. New York, NY: Springer.

Cartier, M. (2007). Méthodes de simulation. In : R.A. Thietart, ed. 2007. *Méthode de recherche en management*. Paris: Dunod, pp. 143–172.

Davis, J., Eisenhardt, K. and Bingham, C. (2007). Developing Theory Through Simulation Methods. *Academy of Management Review*, 32(2), pp. 480–499.

Duncan, W. (1993). The process of project management. *Project Management Journal*, 24(3), pp. 5–10.

Grinstead, C.M. and Snell, J.L. (2003). *Introduction to Probability 2nd edition*, the American Mathematical Society. [pdf] Available at: www.math.dartmouth.edu [Accessed 18 September 2018].

Harrison, J.R., Carroll, G.R. and Careley, K.M. (2007). Simulation Modeling in Organizational and Management Research. *Academy of Management Review*, 32(4), pp. 1229–1245.

Hefferon, J. (2017). *Linear Algebra - third edition*. Colchester: Saint Michael's College.

Khang, D. B. and Moe, T. L. (2008) 'Success criteria and factors for international development projects: A life-cycle-based framework', *Project Management Journal*, 39(1), pp. 72–84. doi: 10.1002/pmj.20034.

Leroux, I. and Berro, A. (2010). Négociation public/privé et coévolution stratégique dans un biocluster. *M@n@gement*, 13(1), p. 38.

Popper, K. (1979). *Objective Knowledge: An Evolutionary Approach*. New York: Oxford University Press.

Senata, E. (2006). Markov and the creation of Markov chains. *MAM 2006 (Markov Anniversary Meeting)*, Charleston: Boson Books, pp. 1–20.

Standish Group (2013). *Chaos Manifesto 2013, Think big, act small*. Standish Group, p. 48.

Appendix 1 – Program of the first example

```
import numpy as np      #Numpy library
#Set the P matrix
P = np.matrix([[0.25, 0.5, 0.25],
               [0.5, 0.0, 0.5],
               [0.25, 0.25, 0.5]])

# Treatment of the data for 20 phases
for step in range(20):
    result = P ** step
    result= np.asarray(result)
    result=np.round(result, decimals=3)
    print(result)
    print("STEP =" + str(step) + "\n")
```

The outputs of the program are:

```
[[0.25 0.5  0.25]
 [0.5  0.   0.5 ]
 [0.25 0.25 0.5 ]]
STEP =1

[[0.375 0.188 0.438]
 [0.25  0.375 0.375]
 [0.312 0.25  0.438]]
STEP =2

[[0.297 0.297 0.406]
 [0.344 0.219 0.438]
 [0.312 0.266 0.422]]
STEP =3

[[0.324 0.25  0.426]
 [0.305 0.281 0.414]
 [0.316 0.262 0.422]]
STEP =4

[[0.312 0.269 0.419]
 [0.32  0.256 0.424]
 [0.315 0.264 0.421]]
STEP =5

[[0.317 0.261 0.422]
 [0.314 0.266 0.42 ]
 [0.316 0.263 0.421]]
STEP =6

[[0.315 0.264 0.421]
 [0.317 0.262 0.422]
 [0.316 0.263 0.421]]
STEP =7

[[0.316 0.263 0.421]
 [0.315 0.264 0.421]
 [0.316 0.263 0.421]]
STEP =8

[[0.316 0.263 0.421]
 [0.316 0.263 0.421]
 [0.316 0.263 0.421]]
STEP =9

[[0.316 0.263 0.421]
 [0.316 0.263 0.421]
 [0.316 0.263 0.421]]
STEP =10

[[0.316 0.263 0.421]
 [0.316 0.263 0.421]
 [0.316 0.263 0.421]]
STEP =11

[[0.316 0.263 0.421]
 [0.316 0.263 0.421]
 [0.316 0.263 0.421]]
STEP =12

Etc.
```