Studium licencjackie

Kierunek: Metody Ilociowe w Ekonomii i Systemy Informacyjne

Forma studiów: Stacjonarne

Imie i nazwisko autora: Kacper Mordarski

Nr albumu: 101247

<tytu>

Praca licencjacka napisana

w Katedrze Matematyki i Ekonomii Matematycznej

pod kierunkiem naukowym

dr hab. Michaa Ramszy

Warszawa 2022

Contents

1 Introduction

1.1 General Introduction

This paper aims to replicate, to an essential degree, the seminal paper on the emergence of cooperation, cf. **?**. The secondary goal of this research is to provide the publicly available code allowing replication of the original results.

Tutaj jest drugi akapit.

—

either prove or disprove the research paper with title **"Scale-Free Networks Provide a Unifying Framework for the Emergence of Cooperation"** written by F.C. Santos and J. M. Pacheco and published in Physical Review Letters nr 95 (later on reffered to as "The Paper"). Regardless of the outcome this work is going to be useful because the above mentioned authors did not provide the code that was used to conduct the necessary simulations. Therefore goal is not only to disprove the above mantioned paper but also to provide readers with clear and easy to follow code.

1.2 Description of The Paper

The Paper was published in "Physical Review Letters" on 26th August 2005 and according to Google Scholar has been since cited over 1600 times. It clearly shows the magnitude of said paper and its groundbreaking character. This paper is presenting the results of simulations conducted by the authors and its implications for evolutionary game theory.

The crucial thing to understand is that the authors of The Paper changed the approach to modeling such games by applying Scale-Free Networks of Contacts (later on referred to as "SF NOCs"). Its innovativeness lies in the never used before degree distribution of said graph. Before being used graphs had a degree distribution with a single peak. It means that every "player" could interact only with a fixed number of other "players". SF NOCs are said to perform better at modeling the actual, existing societies and networks. It complies with the rules of growth and preferential attachment (rich gets richer). When we analyze some actual networks, for example, the Twitter network, we observe that those with a bigger count of "followers" are more likely to gain new ones than accounts with a low count of followers.

One of the goals of The Paper was to compare the results of simulations on different kinds of graphs. According to The Paper, players that occupy vertices of SF NOC are much more likely to cooperate than on any other graph. Those results came up both in the Snowdrift game (later on referred to as SG) and Prisoners Dilemma game (later on referred to as PD).

1.3 A brief description of Snowdrift game and Prisoners Dilemma game

Prisoners Dilemma Game . The Prisoners Dilemma game is a widely known problem in game theory and decision analysis. It shows that there exist situations in which the outcome is not optimal, even though players act in their own best interest. In the scope of our analysis, it is important to note that in the PD game, the best strategy is to defect, regardless of the opponent's choice. The game is parameterized as follows:

	C	D
C	R,R	S,T
D	T,S	P,P

where the values are given as follows:

$$T = b > 1,$$

$$R = 1,$$

$$P = 0,$$

$$S = 0,$$

$$1 < b \leq 2.$$

We see that the above restrictions order the parameters as follows:

$$T > R > P = S$$

Snowdrift Game. The Snowdrift game represents a metaphor for cooperative interactions between players. Contrary to PD, the Snowdrift game stimulates cooperative behavior amongst players. It doesn't make the deflection strategy inapplicable, but payoffs of this game favourize cooperative behaviors more than the PD. The optimal strategy is to cooperate when the other defects and to defect when the other cooperates.

	C	D
C	R, R	T, S
D	S, T	P, P

Where the parametrization is as follows:

$$T = \beta > 1,$$
$$R = \beta - \frac{1}{2},$$
$$S = 1 - \beta,$$
$$P = 0.$$

We see that the above restrictions order the parameters as follows:

$$T > R > P > S$$

1.4 Simulations description

Because of the nature of this paper (an attempt to clone the results of The Paper), our simulations must be conducted in strict accordance with the methods outlined in The Paper. Therefore it is only natural that we must follow each step with the utmost care and diligence. According to The Paper, we must conduct 100 simulations for each parametrization. Each simulation is performed following those steps: Where the parametrization is as follows:

1. Setting up the parameters which are needed to create SF NOCs (such as the number of final vertices (population size), the average connectivity, etc.).

2. Choosing the parameters of the game which is to be simulated (either PD or SG).

3. Creating the randomly generated SF NOC (we use Barabasi - Albert model to do that). The SF NOC must be created in compliance with preferential attachment and growth rules.

4. Randomly distributing strategies amongst the population (SF NOC in this particular case). Each vertex can either get a cooperation or deflection strategy.

5. Each pair of cooperator-deflectors engages in a round of a given game. In compliance with replicators dynamics, we keep track of cumulative payoffs for both strategies so that "players" can adjust their strategies throughout the population. This step is repeated 11 000 times, each time is called "generation". The first 10 000 is the so-called "transient time".

6. We collect results (equilibrium frequencies of cooperators and defectors) by averaging over the last 1 000 generations.

1.5 Replicatory dynamics

In our analysis we consider replicator to be a strategy in a game. The general idea is that replicators compete for dominance throughout the population. Payoffs of their strategies represent their "fitness". It is important to note that each player can alter their strategy through inheritance. The attempt of inheritance occurs whenever one of the sites is updated. For the sake of an example, let us say that the site that was just updated is site x. The procedure is as follows:

1. The site x is updated.

2. A neighbor y is drawn at random among all k_x neighbors

3. if cumulative payoffs of y (P_y) are greater than cumulative payoffs of x (P_x), the chosen neighbor takes over site x with probability (P_i) given below.

$$P_i = \frac{(P_y - P_x)}{Dk_>}$$

Where P_i is the probability of the chosen neighbor taking over the site x, P_y is a cumulated payoff of strategies y, P_x is a cumulated payoff of strategies x, $k_>$ is the largest between k_y and k_x (k_y is a number of neighbors with a strategy y, k_x is a number of neighbors with a strategy x), D depends on the game (it is equal to either T-S for PD or T-P for SG).

2 Algorithm

2.1 Introduction to the algorithm

In this section of the dissertation, I'm going to describe the key elements of the algorithm used to replicate the results obtained in The Paper. My understanding of the mechanisms on which the algorithms are based is limited to the rather vague and unclear description in The Paper.

2.2 Functions

In order to perform necessary calculations I had to define the following functions:

1. Transform — This function is used to map strategies onto a vector of arrays. As an input this function takes one of the edges of a SF NOC, as an output it returns a vector of length two (two vertices connected with an edge).

2. Strat — This function is used to map previously distributed strategies onto a vector of edges. As an input it takes an edge and as an output it returnes a vector of length two (two strategies previously attributed to the vertices).

3. Games — This function is used to evaluate the results of games played between players (vertices connected with an edge). As an input this function takes a vector of edges, vector of strategies and a vector of accumulated payoffs. It returns an adjusted vector of accumulated payoffs.

4. CheckStrat — This function fulfills a number of tasks. It takes as an input a randomly chosen vertex, vector of strategies, vector of accumulated payoffs, and the SF NOC. It identifies all neighbors of the previously mentioned vertex, then shuffles them, and then looks for a neighbor with a different strategy. Then it proceeds to change the strategy of the vertex with lower accumulated payoffs. The probability of the transition is described in section 1.5.

2.3 Description of the algorithm

The main algorithm is the fundamental element of this dissertation. It consists of 3 nested loops executing instructions necessary to conduct simulations, on which my research is based. I will be describing those loops in an inside-out order.

The first loop is responsible for evaluating the results of the games, potentially changing the strategies of players, and keeping track of the proportion of the strategies used by players. To run properly, it requires previously set parametrization and a set of edges of the Barabasi-Albert graph. As a result, it produces a vector of length 100 in which elements are proportions of coop/def strategies, measured after each generation (1001-1100) in the population of players. This loop is repeated 2 000 times, which gives us a total of 2 200 000 generations for one Barabasi-Albert graph parametrization for a given game.

The loop itself operates as follows:

1. Each pair of connected players (vertices of a graph) engage in a single round of a given game. It means that function GAMES is applied throughout the entire array of edges, adjusting their accumulated payoffs.

2. One player is chosen randomly.

3. An attempt to change strategy is being embarked on. It indicates that the function CHECKSTRAT is applied to the player chosen in the previous step.

4. If the current iteration is higher than 1000, it calculates the share of cooperators in the entire population and then passes it onto a corresponding value of a vector TRACK.

5. After 1 100 iterations (generations), it ends, and then the next iteration of the outside loop is triggered.

The outer loop to the one described above is responsible for "resetting" the Barabasi-Albert graph. Since separate simulations are supposed to be carried out on a randomly generated SF NOC (but with the same parametrization), we need to conduct 100 of them (for each payoff parametrization), this loop is an indispensable element of the algorithm. The crucial fact to note is that this loop is also responsible for creating an object named ARRPATHS — vector of length 100, used to store the results of the simulations.

The procedure followed by this loop is as shown below:

1. It generates the random Barabasi-Albert graph with N vertices and average connectivity equal to Z. Both parameters are chosen deliberately at the beginning of the code.

2. The next step is constructing an array of strategies. This array is of length 1000 (number of players). The strategies are represented by numbers 1 (cooperation) and 2 (defection). The choice of these numbers is strict because later on, we use them to index the payoffs matrix. The process of creating such an array is following:

   (a) Creating two vectors of length 500, one filled with ones, and one filled with twos.

   (b) Concatenate those two vectors to get a vector of length 1000.

   (c) Shuffle the values in a vector and save them as STRATEGIES.

3. For the inner loop to conduct games, we need to transform our SF NOC into two arrays — one containing edges (tuples of players) and the other containing strategies for each edge. To achieve this, we apply the following steps:

   (a) We use function COLLECT on the function EDGES used on the Barabasi-Albert model, which results in an iterable, however not easily callable, array of edges.

   (b) Then we map a function TRANSFORM onto the previously obtained object. This produces us an array of tuples, each tuple representing one edge.

   (c) Lastly, we map function STRAT onto an array of tuples representing the edges.

4. The array of accumulated payoffs is created — initially with all values equal to zero (after each game, the results are added to the corresponding values in this array).

5. A global variable is created. This variable is named TRACK and it is a vector of length 100. This vector is used by the inner loop to store the proportion of cooperators in the population.

6. The final step before triggering the inner loop is creating ARRPATH, which will be used to store 100 TRACK variables. If it is the first iteration of this loop, we create a global variable named ARRPATH, which is an array. If, on the other hand, it is not the first iteration, we are using the function PUSH! to add the next TRACK to our existing array.

Last but certainly not least, we have the outer loop. This loop fulfills two tasks. First of all, it controls the parametrization of the games. It not only changes the values of crucial parameters but also updates the matrix of the payoffs. Since we want to generate 20 data points (one data point for each parametrization), said loop iterates 20 times. The range of parameters is given in the paper as follows. The b parameter in the PD game is in the range of [1,2), and the r parameter in the SG game is in the range of (0,1]. Its secondary task is to collect results obtained in the previous loops. Those results will then be analyzed and undergo data manipulations and statistical analysis.

The exact steps followed by this loop are shown below:

1. It creates a global variable called DATATOSAVE — an array of length 20. In its first iteration this variable is created. With every next iteration, it stores data as values in this array. Each element of this array is going to be ARRPATHS generated in the previously described loops. Ultimately it will contain all necessary data for further calculations and visualizations.

2. The next step is to adjust the parameter values. With each iteration, it adds a fixed value to b and r. Since we have 20 iterations and need to fill a distance of 1, the value-added each time equals 0.05.

3. Then it adjusts the payoffs matrices.

After all of the loops have been carried out, we use package JLD2 to save our results.

3 Basic things

3.1 Compiling LaTeXfiles

The `.tex` file is just a plain text file. It contains the LaTeXformatting codes together with the content of a paper. To get a `.pdf` file you have to compile the `.tex` file using a sequence `pdflatex`, `biblatex`, `pdflatex`, `pdflatex`. This sequence is a default in most editors designed for use with LaTeX.

3.2 Basic formatting for a text

Paragraphs are coded by an empty line. That is is you want to start a new paragraph it is enough to leave an empty line and start typing like that:

```
This is the first paragraph.


This is the next paragraph.
```

Everything about the paragraph is formatted for you including all indents and spacings. Again, you don't have to take care of it manually.

Basic text formatting, e.g. bold face and italic, is achieved with the following commands: `\textbf{}`, `\textit{}`, `\underline{}`, producing **text**, *text*, text. I suggest not overusing those commands!

Alignment is done through environments `center`, `flushleft` and `\flushright` giving the following examples.

This is centered.

This is aligned to the left.

This is aligned to the right.

In other environments it is possible to use `\centering` to center content of that environment (like in `figure` or `table` environments).

3.3 Fonts and fonts' sizes

You do not change fonts and fonts' sizes! Technically it can be done but I will reject this.

4 Mathematics

This is testing footnotes[1].

4.1 Basic mathematics

There are two types of mathematics inside a LaTeX document. The first one is the in-line mathematics and the displayed mathematics. The first one looks like this: $F(x) = \int_{-\infty}^{x} f(\omega) d\omega$ with the code looking like this: `\( F(x) = \int_{-\infty}^{x} f(\omega) d\omega \)`. The displayed mathematics looks like that

$$F(x) = \int_{-\infty}^{x} f(\omega) d\omega$$

with the code

```
\[
F(x) = \int_{-\infty}^{x} f(\omega) d\omega
\]
```

As you can see the same code is formatted differently depending on the type of mathematics.

4.2 Referencing mathematics and other things

To reference mathematics (only displayed formulas) you use the `equation` environment with a `\label{}` within. The reference is done through the `\ref{}` command. The example is

$$F(x) = \int_{-\infty}^{x} f(\omega) d\omega. \tag{1}$$

To reference the equation you use the `\ref{}` command giving (1). The `\label{}` / `\ref{}` pair works for anything that can be referenced.

[1]This is a footnote. We can put some math here $x^2 - f(x) = g(x^2)$ which is not encouraged but sometimes necessary. The other thing we can do is to put here an URL `https://tex.stackexchange.com/questions/249415/set-font-size-for-footnotes`.

4.3 Some more mathematical formulas

Here are slightly more complex formulas. Let A be a matrix

$$A = \left(\begin{bmatrix} 1 & \alpha^2 \\ 2 & \sqrt{\pi} - \log(x - \sin(y)) \end{bmatrix}^2 - \begin{bmatrix} 1 & f(x) \\ 2 & g(y) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \right),$$

where

$$f(x) = \begin{cases} \dfrac{1}{x} & \text{for } x < -\dfrac{1}{2}, \\ \dfrac{1}{1+x^2} & \text{for } x \geq -\dfrac{1}{2} \end{cases}$$

and

$$g(y) = \sin\left(\frac{\mathbf{E}(X)}{\cos(y) + \log(y)}\right), \quad \text{where } X \sim \mathrm{N}(0, \sigma).$$

It is very easy to typeset a normal form game. Below is an example of such a game.

	L	M	H
L	$16, 9$	$3, 13$	$0, 3$
M	$21, 1$	$10, 4$	$-1, 0$
H	$9, 0$	$5, -4$	$-5, -15$

5 Figures and tables

Both figures and tables use the same ideas. To insert a table you use the `table` environment. This is an example of a simple table.

Table 1: This is an example of a table.

Name	property 1	property 2	property 3
Michael	23	34	–
John	34	–	28
Mr. Niceguy	123	231	312

Table 1 is a very simple table and much more is possible.

To insert a figure you need to have a figure. In the catalog there are two figures and the following is an example of the `figure` environment.

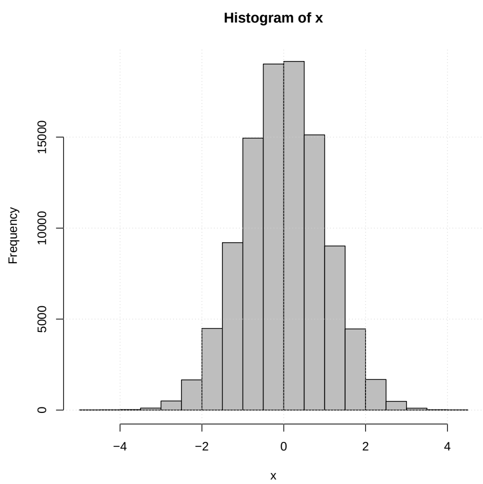
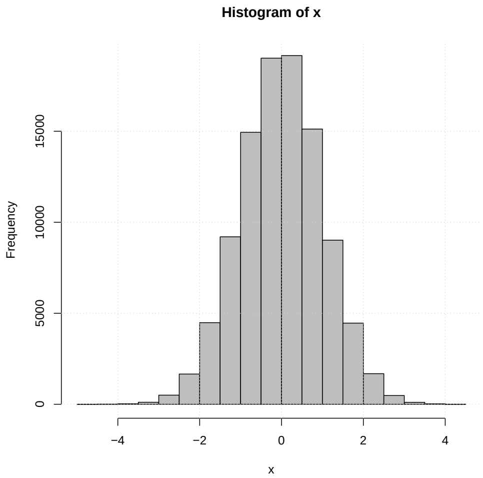
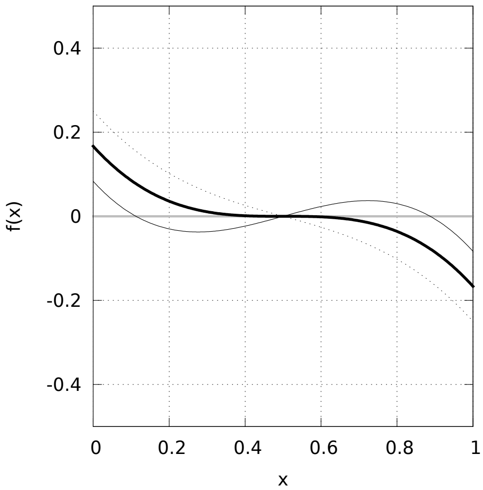


Figure 1: This is just an example. *Source:* own calculations.

Figure 2 is a slightly more complex than just a simple figure but it is useful to have such template. It is possible to refrence subfigures as 2a and 2b.



(a) This is a caption for the first figure. This caption is wrapped at the right width and the hight is being compensated.



(b) This is another caption.

Figure 2: This is the main caption and it is below the figures. *Source:* own calculations

6 Bibliography

The content for the bibliography is in a different file named `refs.bib`. You can change the name but then you have to change the information in this file from `\bibliography{refs}` to `\bibliography{new-name}` where `new-name` is the name of your file. The file `refs.bib` contains some examples for books and papers.

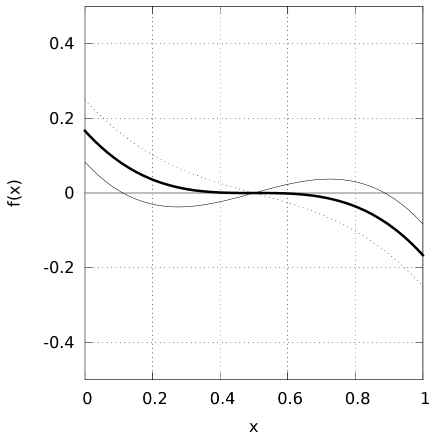


Figure 3: This is how one can wrap a text around a figure. *Source:* own calculations

The process of citation is simple. The command `\cite{garland2010}` gives this **?** and puts all information into the bibliography section at the end. Everything is sorted and formatted for you so that you don't have to worry about this. An example of a paper with many authors is **?** or **?**.

Table 2: Binary variables used in the VAR model

t	year	elections	crises	tax cuts
1	1961	0	0	0
2	1962	0	0	0
3	1963	0	0	0
4	1964	1	0	0
5	1965	0	0	1
6	1966	0	0	0
7	1967	0	0	0
8	1968	1	0	0
9	1969	0	0	0
10	1970	0	0	0
11	1971	0	0	0
12	1972	1	0	0

Continued on next page

Table 2 – *Continued from previous page*

t	year	elections	crises	tax cuts
13	1973	0	0	0
14	1974	0	1	0
15	1975	0	1	0
16	1976	1	0	0
17	1977	0	0	0
18	1978	0	0	0
19	1979	0	0	0
20	1980	1	0	0
21	1981	0	0	0
22	1982	0	1	1
23	1983	0	0	0
24	1984	1	0	0
25	1985	0	0	0
26	1986	0	0	1
27	1987	0	0	0
28	1988	1	0	0
29	1989	0	0	0
30	1990	0	0	0
31	1991	0	1	0
32	1992	1	0	0
33	1993	0	0	0
34	1994	0	0	0
35	1995	0	0	0
36	1996	1	0	0
37	1997	0	0	0
38	1998	0	0	0
39	1999	0	0	0
40	2000	1	0	0

Continued on next page

Table 2 – *Continued from previous page*

t	year	elections	crises	tax cuts
41	2001	0	1	1
42	2002	0	0	1
43	2003	0	0	1
44	2004	1	0	0
45	2005	0	0	0
46	2006	0	0	0
47	2007	0	0	0
48	2008	1	1	0
49	2009	0	1	1
50	2010	0	0	1
51	2011	0	0	0
52	2012	1	0	0
53	2013	0	0	0
54	2014	0	0	0
55	2015	0	0	0

A Appendix: Some important stuff

This appendix contains all the necessary important stuff, blah, blah, blah ...

Table 3: Tutaj jest tytu tablicy

Nazwa atrybutu	Wartoci	Opis
chk_acct	-	stan rodków na rachunku biecym (jakociowa)
	A11	... <0 Marek Niemieckich
	A12	0 <... <200 Marek Niemieckich
	A13	... >200 Marek Niemieckich
	A14	brak rachunku biecego
duration	-	czas trwania kredytu w miesicach (numeryczna)
history	-	przeszo kredytowa (jakociowa)
	A30	brak kredytów w historii/wszystkie kredyty poprawnie spacone
	A31	wszystkie kredyty poprawnie spacone (zacignite w tym banku)
	A32	kredyty poprawnie spacane po dzie dzisiejszy
	A33	opónienia w poprzednich spatach kredytu
	A34	konto krytyczne/zacignite kredyty w innych bankach
purpose	-	cel (jakociowa)
	A40	nowy samochód
	A41	uywany samochód
	A42	meble
	A43	telewizor
	A44	urzdzenia gospodarstwa domowego
	A45	remont
	A46	edukacja
	A47	wakacje
	A48	przekwalifikowanie
	A49	biznes
	A410	inne

kontynuowane na nastpnej stronie

Table 3 – *kontynuacja z poprzedniej strony*

Nazwa atrybutu	Wartoci	Opis
amount	-	kwota kredytu (numeryczna)
say_acct	-	saldo na rachunku oszczdnociowym/warto posiadanych obligacji (jakociowa)
	A61	... <100 Marek Niemieckich
	A62	100 <= ... <500 Marek Niemieckich
	A63	500 <= ... <1000 Marek Niemieckich
	A64	... >= 1000 Marek Niemieckich
	A65	nieznane/ brak oszczdnoci
employment	-	czas zatrudnienia w obecnej pracy (jakociowa)
	A71	brak zatrudnienia
	A72	... <1 rok
	A73	1 <= ... <4 lata
	A74	4 <= ... <7 lat
	A75	... >= 7 lat
install_rate	-	wielko raty jako procent rozporzdzalnego przychodu (liczbowa)
pstatus	-	pe i stan cywilny (jakociowa)
	A91	mczyzna; rozwodnik/w separacji
	A92	kobieta; rozwiedziona/ w separacji/ matka
	A93	mczyzna ; wolny
	A94	mczyzna ; onaty/ wdowiec
	A95	kobieta ; wolna
other_debtor	-	inni dunicy/ porczyciele (jakociowa)
	A101	brak
	A102	wspókredytobiorca
	A103	porczyciel
property	-	wasno/ mienie (jakociowa)

kontynuowane na nastpnej stronie

Table 3 – *kontynuacja z poprzedniej strony*

Nazwa atrybutu	Wartoci	Opis
	A121	nieruchomo
	A122	(jeli nie A121) umowa oszczdnociowa/ ubezpieczenie na ycie
	A123	(jeli nie A121/A122) samochód lub inne
	A124	nieznane
timer_resid	-	czas zamieszkania w aktualnym miejscu zamieszkania (liczbowa)
age	-	wiek w latach (liczbowa)
other_install	-	inne zobowizania ratalne (jakociowa)
	A141	bank
	A142	sklepy
	A143	brak
housing	-	warunki mieszkaniowe (jakociowa)
	A151	wynajem
	A152	wasno
	A153	zamieszkanie bez ponoszenia kosztów
other_credits	-	liczba aktualnych kredytów w tym banku (liczbowa)
job	-	praca (jakociowa)
	A171	bezrobotny/niewykwalifikowany; cudzoziemiec
	A172	niewykwalifikowany; rezydent
	A173	wykwalifikowany pracownik/urzdnik
	A174	menader/ samozatrudniony/ wysocewykwalifikowany/ wyszy urzdnik
num_depend	-	liczba osób na utrzymaniu (liczbowa)
telephone	-	telefon (jakociowa)
	A191	brak
	A192	tak, zarejestrowany pod nazwiskiem klienta
foreign	-	pracownik zagraniczny (jakociowa)

kontynuowane na nastpnej stronie

Table 3 – *kontynuacja z poprzedniej strony*

Nazwa atrybutu	Wartoci	Opis
	A201	tak
	A202	nie
response	-	decyzja kredytowa
	1	tak
	2	nie

List of Tables

List of Figures

Streszczenie

Tutaj zamieszczaj Pastwo streszczenie pracy. Streszczenie powinno by dugoci okoo pó strony.