FuzzyStatProb: An R Package for the Estimation of Fuzzy Stationary Probabilities from a Sequence of Observations of an Unknown Markov Chain

Pablo J. Villacorta
University of Granada

José L. Verdegay
University of Granada

Abstract

Markov chains are well-established probabilistic models of a wide variety of real systems that evolve along the time. Countless examples of applications of Markov chains that successfully capture the probabilistic nature of real problems include areas as diverse as biology, medicine, social science, and engineering. One interesting feature of a kind of Markov chains is their stationary distribution, which stands for the global fraction of time the system spends in each state. The computation of the stationary distribution requires precise knowledge of the transition probabilities. When the only information available is a sequence of observations drawn from the system, such probabilities have to be estimated. Here we review an existing method to estimate fuzzy transition probabilities from observations and, with them, obtain the fuzzy stationary distribution of the resulting fuzzy Markov chain. The method also works when the user directly provides fuzzy transition probabilities. We provide an implementation in the R environment that is the first available to the community and serves as a proof of concept. We demonstrate the usefulness of our proposal with computational experiments on a toy problem, namely a time-homogeneous Markov chain that guides the randomized movement of an autonomous robot that patrols a small area.

Keywords: fuzzy probabilities, Markov chain, stationary probabilities, R.

1. Introduction

Assume we observe the state of a system at certain time points (i.e., time is discretized for our observations). An observation X_t obtained at time t represents one state the system can be in. In fact, the collection of observations $X_0, X_1, ..., X_n$ represent an indexed sequence of random variables, each of which can take values within the system *state space* $S = \{1, ..., r\}$, which we will assume finite and countable as well. Such indexed sequence $\{X_t, t = 0, 1, 2, ...\}$ is known as a stochastic process (Medhi 2002) in discrete time and discrete state space S.

A stochastic process $\{X_t, t = 0, 1, 2, ...\}$ with discrete state space and discrete time is a Markov chain if, and only if, it has the Markov property, which can be stated in simple terms as the fact that the state of the system at the next time point depends only on the state at the current time, and it is independent of the succession of states reached in the past. Formally, this independence from the past is expressed as

$$P[X_{t+1} = x_{t+1}|X_t = x_t, X_{t-1} = x_{t-1}, ..., X_0 = x_0] = P[X_{t+1} = x_{t+1}|X_t = x_t] \quad (1)$$

When the probabilities $P[X_{t+1} = x_{t+1}|X_t = x_t]$ do not depend on the time points $t, t+1$ but only on the time difference h between them, the chain is said to be homogeneous (in the preceding example, $h = 1$). In that case, it is usual to collect the transition probabilities in a (stochastic) transition matrix $P^{(h)} = (p_{ij}^{(h)})$ with dimensions $r \times r$ representing the transition probabilities in h steps, $p_{ij}^{(h)} = P[X_{t+h} = j|X_t = i]$. In general, $P^{(h)} = P^h$.

Markov chains are well-established probabilistic models of a wide variety of real systems that evolve along the time. Countless examples of applications of Markov chains to successfully capture the probabilistic nature of real problems include areas as diverse as biology (Abundo and Caramellino 1995), medicine and particularly disease expansion models (Gentleman, Lawless, Lindsey, and Yan 1994; Satten and Longini 1996; Ivanek, Grohn, Ho, and Wiedmann 2007; Gomez, Arenas, Borge-Holthoefer, Meloni, and Moreno 2010; Newton, Mason, Bethel, Bazhenova, Nieva, and Kuhn 2012), economy (Jones 1997) -with emphasis on market modelling (van der Hoek and Elliott 2012)-, social science and many other engineering problems such as wind speed modeling (Sahin and Sen 2001). They are also the basis for Hidden Markov models, which constitute themselves another active research field with important applications in such problems as speech recognition (Juang and Rabiner 1991) and automatic music generation (Schulze and van der Merwe 2011), just to cite a few. A broader survey can be found in Ching and Ng (2006).

One of the features that characterize certain kinds of Markov chains is the stationary distribution, which represents the probability of the chain being at each state after an infinite number of time steps. It can also be thought of as the percentage of time the chain spends at each of its states. The computation of such probabilities requires an accurate knowledge of the transition matrix of the chain. If it is not available for some reason, it has to be estimated in some way, possibly from a set of observations of the chain, with the uncertainty that any estimation procedure involves.

Several methods have been proposed to estimate transition probabilities, see Dent and Ballintine (1971) and more recently Welton and Ades (2005). Other works have addressed the problem of computing stationary probabilities with uncertain data from a mathematical programming point of view (Blanc and Hertog 2008). The approach adopted in the remainder of this work is based on using fuzzy sets (Zadeh 1965, 1975a) to cope with uncertainty, thus considering fuzzy transition probabilities that constitute a fuzzy Markov chain.

Quite a lot of research has been conducted on fuzzy Markov chains. Zadeh himself envisaged their potential importance in fuzzy Markov algorithms in Zadeh (1998). Other recent successful applications of fuzzy Markov chains are Kruse, Buck-Emden, and Cordes (1987) which deals with processor power, Tran and Wagner (1999) for speech recognition, and Feitosa, Costa, Mota, and Feijó (2011); Alves, Mota, Costa, and Feitosa (2012) for multitemporal image classification. Some works consider a fuzzy state space (Kleyle and de Korvin 1997). On the contrary, in our proposal we consider crisp states but fuzzy probabilities. In Kurano, Yasuda, Nakagami, and Yoshida (1992); Avrachenkov and Sanchez (2002) and in most of the works mentioned previously on fuzzy Markov chains, the uncertain transition matrix is modeled as a fuzzy relation between the states. Hence there is no restriction on the values it can take, as far as they are membership grades in the closed interval [0, 1] that in fact represent a perception of a classical Markov chain (Kruse *et al.* 1987).

Another, more restrictive approach presented in a series of more general works on fuzzy probability theory (Buckley 2004, 2005; Buckley and Eslami 2003, 2004, 2008) consists in

defining fuzzy numbers for the transition matrix, subject to the crisp constraint that they must add to 1, no matter how uncertain they are. Operations with the fuzzy transition matrix are carried using a restricted fuzzy matrix multiplication. Little work has been done in this line regarding Markov chains, and the author's ideas still remain as a theoretically feasible procedure whose practical applicability has not been investigated. It may be suitable when the input is a sequence of crisp observations of the chain. For those reasons we have chosen to implement it in the **FuzzyStatProb** package for the R environment (R Core Team 2012). Moreover, to the best of our knowledge, currently there are no implementations available -either free or proprietary- of any of the fuzzy Markov chain approaches, so this aims to be the first one and pave the way for further programming efforts in this field.

The method can be summarized as follows: the fuzzy probabilities of the fuzzy transition matrix will be first decomposed in their α-cuts, which will be used to solve the problem in separate pieces, and finally the results will be aggregated to re-construct the fuzzy stationary probabilities we are searching for. The starting point of the problem is a finite sequence of observations of the system state at each step. This is the input required by the **FuzzyStatProb** package we have developed. The output is provided as a list of fuzzy numbers, one for each state of the input Markov chain, using the **FuzzyNumbers** package described in Gagolewski (2012) that provides plotting functionality and is currently under active development.

The paper is structured as follows. In Section 2, the mathematical background and the method implemented in the package are formally developed. Section 3 describes implementation details and the signature and options of the public function of the package. Section 4 shows use examples. Finally, Section 5 is devoted to conclusions and further work.

2. A method to compute fuzzy stationary probabilities

An homogeneous Markov chain is said to be irreducible if all states form a single group so that every state is accessible from every other state, be it in one or more than one step. Every finite, irreducible Markov chain with transition matrix P has a *stationary distribution* π such that $\pi = \pi P$ (Medhi 2002). The **FuzzyStatProb** package is aimed at computing a vector of fuzzy stationary probabilities $\tilde{\pi} = (\tilde{\pi_1}, ..., \tilde{\pi_r})$ departing from an uncertain (fuzzy) transition matrix $\tilde{P}$. The remainder of this section briefly summarizes the ideas on fuzzy Markov chains presented in Buckley (2005), as they lie at the very core of our package.

2.1. Fuzzy numbers

Let us consider an uncertain transition matrix $\tilde{P} = (\tilde{p}_{ij})$ in which one or more elements are uncertain. We capture the uncertainty regarding some of the elements by substituing them by fuzzy numbers, which are a special case of fuzzy set (Zadeh 1965; Negoita and Ralescu 1975). Roughly speaking, a fuzzy set $\tilde{A}$ over a universe U is a set whose characteristic function is not binary, $f : U \to \{0, 1\}$ as in a classic set, but takes values in the unit interval, $\mu : U \to [0, 1]$, so that $\mu_{\tilde{A}}(x)$ is the degree to which element $x \in U$ belongs to the fuzzy set $\tilde{A}$. A fuzzy number is a fuzzy set over $\mathbb{R}$ representing a number with uncertainty (Dubois and Prade 1978). For instance the uncertain quantity *about 2* may be represented as a fuzzy number $\tilde{A}$ whose membership function reaches its maximum value 1 at $x = 2$, but also assigns non-zero membership degrees to other values close to 2, since a value of, say, 1.8 is also ('belongs also to') *about 2*, not with degree 1 but a bit less, say, 0.6. Then $\mu_{\tilde{A}}(2) = 1$ and $\mu_{\tilde{A}}(1.8) = 0.6$.

We now explain this concept formally as it is the mathematical structure used in our package to cope with uncertainty in the transition and stationary probabilities.

Definition 1. *The α-cut of a fuzzy set $\tilde{A}$ for any $\alpha \in [0,1]$ is defined as the set of elements that belong to $\tilde{A}$ with degree α or greater:*

$$\tilde{A}(\alpha) = \{x \in U : \mu_{\tilde{A}}(x) \geq \alpha\}$$

Definition 2. *A fuzzy number $\tilde{A}$ is a fuzzy set on $\mathbb{R}$ such that*

(i) $\tilde{A}(\alpha)$ *are nonempty convex sets* $\forall \alpha \in [0,1]$

(ii) $\tilde{A}(\alpha)$ *are compact sets* $\forall \alpha \in [0,1]$

(iii) $\tilde{A}(\alpha)$ *satisfy:*

(a) $\tilde{A}(\alpha) \subseteq \tilde{A}(\beta)$ *for* $\alpha > \beta$

(b) $\tilde{A}(\alpha) = \bigcap_{\beta<\alpha} \tilde{A}(\beta)$ *for* $\alpha \in (0,1]$

Since α-cuts of a fuzzy number are closed intervals of $\mathbb{R}$, they can be written as $\tilde{A}(\alpha) = [\tilde{A}_L(\alpha), \tilde{A}_R(\alpha)]$.

The first two conditions are basically equivalent to the fact that the membership function of a fuzzy number must be continuous and monotone, which are two features of the output fuzzy numbers computed by our method as will be shown later.

2.2. Construction of fuzzy transition probabilities from observations

As stated in Section 1, our data consists of a finite sequence of observations drawn from the system at consecutive time points. Each observation is the state of the system at that time point, represented as an integer belonging to the state space of the chain. In such sequence, an estimation of the transition probability p_{ij} can be computed as the number of times N_{ij} that state i was followed by j in our data, divided by the number of transitions N_i observed from i to another state (including i itself). However, this would be just a point estimate and does not capture the uncertainty associated to it. Interval estimation would be the next step as it also takes into account the sample size and variance of the data, but it calculates an interval only for a significance level that is fixed beforehand.

Our aim is to model each uncertain transition probability as a fuzzy number that properly captures all the uncertainty of the data, regardless external parameters such as the confidence level. For this purpose, the fuzzy number is obtained by the superposition of confidence intervals for the true transition probabilities at different confidence levels, one on top of another, as done in Buckley (2005). Although the significance level and the membership degree are numerically the same in this case, it should be noticed that they are completely different concepts for which the letter α is commonly used. Such intervals are in fact simultaneous confidence intervals for multinomial proportions, since the problem of estimating the transition probabilities from a given state can be reduced to that of estimating the parameters (proportions) of a multinomial distribution. The method applied was introduced by Sison and Glaz (Sison and Glaz 1995; May and Johnson 2000) although many others had been proposed before.

2.3. Fuzzy Markov chains and restricted matrix multiplication

As we have said, fuzzy numbers are used in those entries of the transition matrix on which there is uncertainty. The rest of the elements could be crisp since a crisp number is a special case of fuzzy number. However, the uncertainty is on the probabilities but not on the fact that every row must add to 1. Now assume a Markov chain with r states and fuzzy transition matrix $\tilde{P}$, then for a given $\alpha \in [0,1]$, $\tilde{P}(\alpha) = (\tilde{p}_{ij}(\alpha))$ represents a matrix of intervals. Such matrix can also be thought of as the set of all $r \times r$ matrices M such that their elements fall inside their corresponding interval, $m_{ij} \in \tilde{p}_{ij}(\alpha)$, and every row adds to 1, $\sum_{j=1}^{r} m_{ij} = 1, i = 1, ..., r$. The *domain* of row i for a given α value is defined as the set of r-dimensional vectors that simultaneously fulfill those two constraints for row i, i.e., the set of probability distributions that row i could take in our uncertain transition matrix when we take its α-cuts for that α. If we call $\Delta_r = \{(x_1, ..., x_r) : x_i \geq 0 \text{ and } \sum_{i=1}^{r} x_i = 1\}$, then

$$Dom_i(\alpha) = \left(\times_{j=1}^{r} \tilde{p}_{ij}(\alpha)\right) \bigcap \Delta_r \tag{2}$$

The domain of the matrix for a fixed α, $Dom(\alpha)$ is defined as the Cartesian product of the domain of every row and represents the space of matrices we admit when the uncertainty level is α. $Dom(\alpha)$ is closed and bounded (compact) because it is the cartesian product of compact sets, so any continuous function applied to its elements will have a compact image. In particular, as explained in (Buckley 2005, Chapter 6), the pow to exponent n of the $r \times r$ transition matrix (which is done as successive matrix multiplications) can be thought of as a collection of r^2 independent functions $f_{ij}^{(n)} : \mathbb{R}^{r^2} \to \mathbb{R}$, each of which calculates the value of one entry of the resulting matrix by operating with the entries of P: $p_{ij}^{(n)} = f_{ij}^{(n)}(p_{11}, ..., p_{1r}, p_{21}, ..., p_{2r}, ..., p_{r1}, ..., p_{rr})$. Such functions are continuous and thus, when applied on a compact set like $Dom(\alpha)$ for a concrete α, the image of all of them is another compact set (a closed interval) of $\mathbb{R}$. Such interval can be viewed as an α-cut of the fuzzy number $\tilde{p}_{ij}^{(n)}$, i.e., $\tilde{p}_{ij}^{(n)}(\alpha) = f_{ij}^{(n)}(Dom(\alpha))$. By the representation theorem (Negoita and Ralescu 1975), it is possible to reconstruct the fuzzy number $\tilde{p}_{ij}^{(n)}$ as the union of its α-cuts which, in theory, can be computed using the restricted fuzzy multiplication described before with different values of α. For our problem, we will use the superior of the α values as the union operator.

In particular, the stationary distribution π matches any of the (identical) rows of a matrix Π such that $\Pi = \lim_{n\to\infty} P^n$, which means that theoretically, π can be computed using matrix multiplication. We are thus in the same case explained above which guarantees that the resulting images are compact sets over $\mathbb{R}$, or more precisely the α-cuts of the fuzzy stationary probabilities we aim to calculate.

2.4. User-specified fuzzy transition probabilities

The package also supports user-specified fuzzy transition probabilities. Instead of departing from a sequence of observations of the state of the chain at consecutive time instants, the user provides a fuzzy transition matrix composed of fuzzy numbers that constitute the transition probabilities. As explained before, despite being fuzzy, these numbers must form a probability distribution for each row. According to previous works (Halliwell and Shen 2008), this can be formalized by imposing the constraint that $\tilde{p}_{i1} \oplus \tilde{p}_{i2} \oplus ... \oplus \tilde{p}_{in} \supseteq 1_\chi$ for each row $i = 1, ..., n$. Here, $\tilde{A} \supseteq \tilde{B} \leftrightarrow \mu_{\tilde{A}}(x) \geq \mu_{\tilde{B}}(x) \forall x \in \mathbb{R}$, the symbol $\oplus$ stands for the fuzzy sum, and 1_χ is the

real number 1 represented as a fuzzy number (singleton). Note this is a necessary condition to ensure that $Dom(\alpha)$ is not empty, as proved in Villacorta, Verdegay, and Pelta (2013) and, therefore, our code first checks this condition and stops if it is not fulfilled.

One of the main applications of fuzzy probabilities is related to linguistic probabilities (Halliwell and Shen 2008), as we explain next. In case no data of the chain are available, we may have to elicit the transition probabilities from a human expert or group of experts. However, it can be difficult for them to express their expertise using numeric probabilities which, in addition, must fulfill the requirement of adding exactly 1. Therefore, natural language can help express probabilities in a more natural way. The transition probability becomes a linguistic variable whose possible values are linguistic terms (Zadeh 1975b), such as *Very unlikely, Most likely, It may, Extremely likely* and so on. Each linguistic term has an underlying fuzzy number that captures the vagueness inherent to natural language. The input information would consist of subjective statements like *The robot goes very often to location 6 when it is in location 1, it almost never goes from there to location 2, when it is placed in location 5 it seems to go to state 8 almost surely*, etc.

It is not relevant for our implementation whether the fuzzy transition probabilities have an associated linguistic term or not, but each fuzzy number of the input transition matrix must be referred by a name when calling the function (see the next section).

2.5. Computation of fuzzy stationary probabilities

The approach outlined in Section 2.3 did not make direct use of the membership functions of the fuzzy transition probabilities, but worked only with their α-cuts to establish $Dom(\alpha)$. Thus it is not necessary to fully reconstruct such fuzzy numbers; it suffices to compute their α-cuts for the desired levels of α. Each of those levels provides us with an interval transition matrix that defines a domain, i.e., a space in which we admit our uncertain transition matrix lies.

Now, in order to find the α-cuts of the fuzzy stationary probabilities for a certain α, we just have to find, for each state of the chain, a pair of matrices within $Dom(\alpha)$ which respectively minimize and maximize the stationary probability of that state. This is repeated independently for every state. In other words, assuming a Markov chain with r different states, and focusing on a significance level α, we have to solve r independent minimization problems and r independent maximization problems in order to compute r different α-cuts of the fuzzy stationary probabilities we are looking for. Note that the search space is the same for all the problems, $Dom(\alpha)$, but the objective function is not: for each state s_j whose stationary probability $\tilde{\pi_j}$ is being computed, it is the expression which yields that stationary probability as a function of the entries of the transition matrix. The function is first minimized to find the lower bound of the α-cut $\tilde{\pi_j}(\alpha)$, and then maximized to find the upper bound. It seems clear that a general expression cannot be calculated for any r-state chain to apply conventional optimization techniques, since such function (in which all the matrix coefficients p_{ij} will be symbolic as well) would be too complicated for chains with more than 5 or 6 states. Furthermore, R symbolic capabilities are very poor and make it unsuitable. For both reasons, heuristic search (optimization) algorithms will be employed, as suggested in Buckley (2005).

The α-cuts can be written as follows. Let $g_j : \mathbb{R}^{r^2} \rightarrow \mathbb{R}$ be a function that represents the calculation of the j-th component of the stationary distribution of an r-state Markov chain whose transition matrix is expressed as an r^2-dimensional vector $(p_{11}, ..., p_{rr})$. As explained

in the previous section, such distribution can be obtained by means of matrix multiplication that converges to a matrix Π with identical rows that are indeed the stationary distribution. Using the same notation and assuming we are only interested in the first row of Π, we can set $g_j = \lim_{n\to\infty} f_{1j}^{(n)}$, i.e., g_j is the function that computes the element Π_{1j}. Such function only applies sums and products during the calculation process, so it is continuous and hence $g_j(Dom(\alpha))$ is a compact set on $\mathbb{R}$, i.e., an α-cut. Then

$$\tilde{\pi_j}(\alpha) = [\pi_{j_L}(\alpha), \pi_{j_R}(\alpha)], \quad j = 1, ..., r \tag{3}$$

$$\pi_{j_L}(\alpha) = \min\{w_j | w_j = g_j(p_{11}, ..., p_{rr}), (p_{11}, ..., p_{rr}) \in Dom(\alpha)\} \tag{4}$$

$$\pi_{j_R}(\alpha) = \max\{w_j | w_j = g_j(p_{11}, ..., p_{rr}), (p_{11}, ..., p_{rr}) \in Dom(\alpha)\} \tag{5}$$

Equations 4 and 5 state that, in order to compute the lower and upper bounds of an α-cut of $\tilde{\pi}_j$ (the j-th fuzzy component of the fuzzy stationary distribution vector $\tilde{\boldsymbol{\pi}}$), we must find the minimum and the maximum of the j-th component of all the crisp stationary distributions corresponding to feasible crisp matrices, understanding feasibility as belonging to the space $Dom(\alpha)$ for that α. In practice, the condition $w_j = g_j(p_{11}, ..., p_{rr})$ is implemented as $\mathbf{w} = \mathbf{w}P$.

The final step is to calculate an analytical expression for the membership function of the fuzzy stationary probabilities. If the above α-cuts were exact, i.e., if we could guarantee that the solutions for the optimization problems are global optima, then the adequate way to proceed would be to interpolate the lower and upper bounds for every sampled α to separately obtain expressions for the left and the right sides of the membership function. The number of points depends on how much precision we need for the membership function. If the membership function has to be reconstructed very accurately, we should probably sample α values in [0, 1] in steps of, say, 0.01, which yields 100 points in each side. A larger step size of about 0.05 (20 points) is probably enough for most real problems and is less time consuming.

However, since we have no guarantee that the α-cuts are exact because they have been obtained by heuristic optimization procedures, regression may also be a good choice, and the loss of precision is globally quite small. The form of the regression function depends on what the points look like. In order to allow for the maximum flexibility, it is the user who can choose between spline interpolation or some kind of regression to be applied. More details are provided in Section 3.

Numerical example The first part of the example does not use data from observations but serves to illustrate the mathematical procedure. Let $\tilde{A} = (a, b, c)$, where $a, b, c \in \mathbb{R}, a \leq b \leq c$, be a Triangular Fuzzy Number (TFN), which can be viewed as a particular case of fuzzy number with the following membership function:

$$\mu_{\tilde{A}}(x) = \begin{cases} (x-a)/(b-a) & a \leq x < b \\ (c-x)/(c-b) & b < x \leq c \\ 0 & \text{otherwise} \end{cases}$$

Now consider the fuzzy transition matrix depicted on the left, formed by TFNs which, in this particular case, are symmetric (although they do not have to). Note some of them carry more uncertainty than others. Focusing on a concrete value for α, for instance $\alpha = 0.5$, this matrix yields the interval matrix $\tilde{P}(0.5) = (\tilde{p}_{ij}(0.5))$ on the right, constituted by the 0.5-cuts of the fuzzy transition probabilities of $\tilde{P}$.

$$\tilde{P} = \begin{pmatrix} (0.6,\ 0.7,\ 0.8) & (0.2,\ 0.4,\ 0.6) \\ (0.3,\ 0.4,\ 0.5) & (0.55,\ 0.6,\ 0.65) \end{pmatrix} \qquad \tilde{P}(0.5) = \begin{pmatrix} [0.65,\ 0.75] & [0.3,\ 0.5] \\ [0.35, 0.45] & [0.575,\ 0.625] \end{pmatrix}$$

$$
\begin{aligned}
Dom_1(0.5) &= \{(x,y) \in \mathbb{R}^2 : x+y=1, x \in [0.65, 0.75], y \in [0.3, 0.5]\} \\
Dom_2(0.5) &= \{(x,y) \in \mathbb{R}^2 : x+y=1, x \in [0.35, 0.45], y \in [0.575, 0.625]\} \\
Dom(0.5) &= \{(p_{11}, p_{12}, p_{21}, p_{22}) \in \mathbb{R}^4 : (p_{11}, p_{12}) \in Dom_1(0.5), (p_{21}, p_{22}) \in Dom_2(0.5)\}
\end{aligned}
$$

Let g_1, g_2 be the functions that, for the given $\alpha = 0.5$, compute the elements Π_{11} and Π_{12} of the (crisp) stationary matrix Π corresponding to each of the 2×2 feasible matrices $\{P = (p_{11}, ..., p_{22}) \in Dom(0.5)\}$. Let us rename the crisp stationary distribution (Π_{11}, Π_{12}) of a feasible matrix as $\mathbf{w} = (w_1, w_2)$. We are interested in finding four feasible matrices $P_1, P_2, P_3, P_4 \in Dom(0.5)$ which, respectively, minimize and maximize w_1, and minimize and maximize w_2. The minimum and maximum w_1 act as the lower and upper bounds of the 0.5-cut $\tilde{\pi}_1(0.5)$, and the same applies to w_2 with respect to the 0.5-cut $\tilde{\pi}_2(0.5)$:

$$
\begin{aligned}
\pi_{1L}(0.5) &= \min\{w_1 | w_1 = g_1(P), P \in Dom(0.5)\} = \min\{w_1 | \mathbf{w} = \mathbf{w}P, \quad P \in Dom(0.5)\} \\
\pi_{1R}(0.5) &= \max\{w_1 | w_1 = g_1(P), P \in Dom(0.5)\} = \min\{w_1 | \mathbf{w} = \mathbf{w}P, \quad P \in Dom(0.5)\} \\
\pi_{2L}(0.5) &= \min\{w_2 | w_2 = g_2(P), P \in Dom(0.5)\} = \min\{w_2 | \mathbf{w} = \mathbf{w}P, \quad P \in Dom(0.5)\} \\
\pi_{2R}(0.5) &= \max\{w_2 | w_2 = g_2(P), P \in Dom(0.5)\} = \min\{w_2 | \mathbf{w} = \mathbf{w}P, \quad P \in Dom(0.5)\}
\end{aligned}
$$

This procedure should be repeated by sampling α in [0, 1] with a step size that depends on the precision desired to reconstruct the fuzzy stationary vector from its α-cuts.

Now, suppose we do not depart from a fuzzy transition matrix (provided by a human expert, for instance) as in the example above, but from a collection of observations of the state of the chain at several consecutive time instants, e.g. $\{1, 3, 2, 3, 4, 4, 2, 3...\}$. Then, in order to estimate the α-cuts of the fuzzy transition probabilities that are needed to establish the domains for each α, we resort to simultaneous confidence intervals at level α of multinomial proportions, using the number of times that the chain transitioned from one state to any other as the input. Once the CIs for the transition probabilities have been computed to compose the interval matrices $\tilde{P}(\alpha)$ for each α, we can define the domains and go on with the rest of the process as it remains unchanged. In this case, matrix $\tilde{P}$ is never constructed explicitly because we are only interested in the α-cuts of the fuzzy entries, as they constitute the constraints of the optimization problems.

3. Implementation in the FuzzyStatProb package

The signature of the only public function is

`fuzzyStationaryProb(data,options,step=0.05,...)`

where:

- `data`: This argument can be: (a) an array of either strings or natural numbers representing the observed states of the chain at consecutive time points. The function first coerces the elements to a factor integer. (b) A 2D square matrix of strings representing fuzzy transition probabilities directly given by the user. Each string should be contained in `names(fuzzynumbers)` and refers to the corresponding `FuzzyNumber` object in the `fuzzynumbers` vector (see below). When the transition probability from state i to j is 0 (in the crisp sense), then entry (i, j) must be NA. The matrix should have `colnames` and `rownames` set.

- `options`: a tagged list containing the following parameters:
  - `verbose`: boolean, set to `TRUE` if progress information should be printed during the process. It is set to `FALSE` if this option is not specified.
  - `states`: an array of strings indicating the states for which the stationary distribution should be computed. The values should match those specified in the `data` argument. If this option is not specified, the fuzzy stationary probabilities are computed for every state of the chain.
  - `acutsonly`: boolean, set to `TRUE` if no regression should be done after computing the α-cuts. This option is set to `FALSE` if not specified.
  - `regression`: a string with the type of the regression to be applied at the end of the algorithm for fitting the membership functions of the fuzzy stationary probabilities. Possible values are 'linear', 'quadratic', 'cubic', 'gaussian', 'spline' and 'piecewise' (piecewise linear). In all cases (including the gaussian), a different curve is fitted for each side of the fuzzy number. The `gaussian` option fits curves of the form $\mu(x) = \exp\left(-\frac{1}{2}\left|\frac{x-c}{s}\right|^m\right)$. The `spline` option performs interpolation by a monotone cubic spline according to the Hyman method (see `splinefun` documentation) while `piecewise` computes a piecewise linear membership function by connecting consecutive points of the α-cuts with straight lines, using the built-in `PiecewiseLinearFuzzyNumber` subclass of the **FuzzyNumbers** package. If this option is not specified, quadratic regression is carried out by default. If `acutsonly` is set to true, this option is ignored.
  - `ncores`: positive integer representing the maximum number of cores that can be used when running in parallel. If set to more than 1, then each processor takes care of all the computations involving one of the values of α that have to be sampled, via `parLapply` function of the **parallel** package. Defaults to 1 (sequential) if not specified. If `ncores` is greater than the actual number of cores in the computer, all available cores are used.
  - `fuzzynumbers`: a tagged list with all the different `FuzzyNumber` objects that appear in `data` when `data` is a matrix of labels; ignored otherwise. Every element of the list must have a name, referenced in at least one entry of `data`.
- `step`: step size for sampling α when computing the α-cuts. The smallest α is always present and equals 0.001, and the rest of values are calculated as $\alpha = k \cdot$ `step` for $k \geq 1$. The greatest sampled value that is always present as well is $\alpha = 0.999$. It is set to 0.05 when this option is not specified.
- `...` Further arguments to be passed to `DEoptim.control` to customize the algorithm that finds the lower and upper bounds of the α-cuts by solving a minimization and a maximization problem.

The value returned by the function is a tagged list that belongs to a new S3 class called `FuzzyStatObj`. This class has only two specific methods, `print` and `summary`, and both print exactly the same: a brief summary of the processing that has been done, including the number of states, number of observations, regression type, step size and time elapsed, and information about the names of the two tags that should be queried to retrieve the results, namely `$fuzzyStatProb` and `$acuts`. The former is a tagged list of `FuzzyNumber` objects

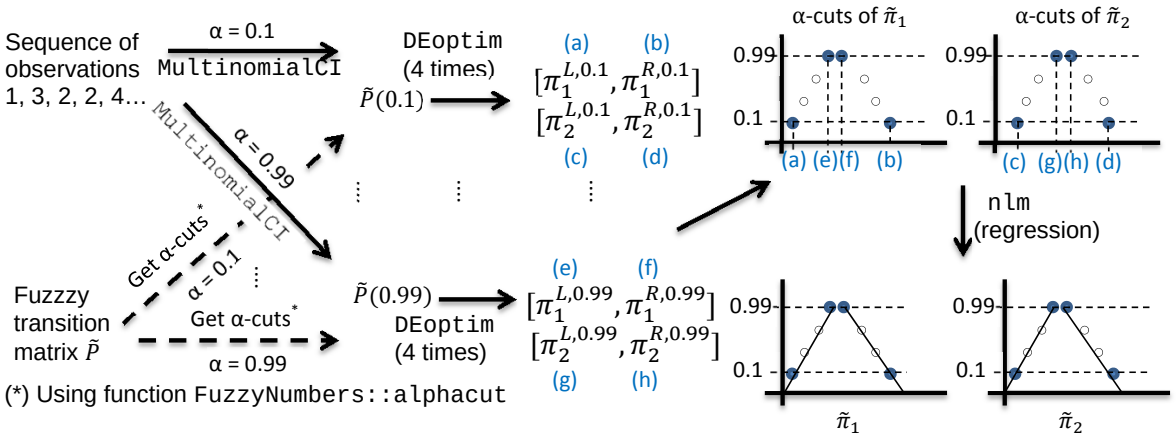


Figure 1: Diagram of the function workflow and the R packages used in each phase.

(see the **FuzzyNumbers** package cited before) whose length equals `length(options$states)` and whose names match those in `options$nstates`. The latter is another tagged list with the same length and names, consisting of data frame objects that contain the α-cuts of every output stationary probability, represented as pairs $(\tilde{\pi}_{jL}(\alpha), \alpha)$ and $(\tilde{\pi}_{jR}(\alpha), \alpha)$. The object at each position represents the fuzzy stationary probabilities and α-cuts of the element of the `options$states` list that is in the same position. Note that in case the user specifies within `options$state` a state that is not found among the elements of the `data` argument, then the corresponding position of `$fuzzyStatProb` and `$acuts` will be `NA`. When setting `acutsonly = TRUE`, the returned object does not have a `$fuzzyStatProb` tag but only `$acuts`. Section 4 contains a fragment of code and the R output showing how our function should be called.

3.1. Implementation issues

This section describes how each step of the mathematical process explained before has been implemented in R and the external packages used. A general scheme of the package is depicted in Figure 1, which has two possible starting points depending on whether the user provides a sequence of observations or a fuzzy transition matrix.

Confidence intervals The first step is to build the fuzzy numbers of the fuzzy transition matrix, as the superposition of confidence intervals. The procedure samples several α from 0 to 1; the step size of the sampling is specified by the user in the `step` argument of the function. For each α and for each state s_i of the chain, the method builds as many simultaneous confidence intervals as transitions to other states s_j have been observed from s_i. The procedure makes use of the **MultinomialCI** package created by the first author (Villacorta 2012) which implements the Sison and Glaz method (Sison and Glaz 1995) to calculate simultaneous confidence intervals for multinomial proportions. It is a direct R translation of the SAS code published in May and Johnson (2000). The **FuzzyStatProb** loads this package during the execution.

If the user has provided directly the fuzzy transition probabilities, this step is omitted and the α-cuts are taken directly from the fuzzy numbers passed to the function.

Computation of α-cuts of stationary probabilities The second (and main) step of the method is to solve the optimization problems of Equations 4 and 5, using some heuristic optimization technique. As explained in Section 2.5, the confidence intervals which are the α-cuts of the fuzzy entries of the transition matrix are used in this step as box-constraints for the optimization that searches for the minimum and maximum value of each stationary probability within $Dom(\alpha)$. The **DEoptim** package (Mullen, Ardia, Gil, Windover, and Cline 2011) was employed for this purpose. It is an implementation of the Differential Evolution (DE) algorithm (Storn and Price 1997; Price, Storn, and Lampinen 2005) that has exhibited excellent performance in a wide variety of hard continuous optimization problems. The package provides built-in capabilities to specify box-constraints for every variable but cannot handle equality constraints (in this case, the probabilities of every row of the matrix being evaluated must add to 1), so they have to be controlled directly in the objective function.

Computation of membership functions via regression The third and last step is to reconstruct the membership function of the fuzzy stationary probabilities whose α-cuts were computed in the preceding step. Focusing on regression, the user can choose among linear, quadratic, cubic and gaussian regression. When fitting a quadratic or cubic membership function, care should be put to avoid non-monotonous functions, not allowed in this context for fuzzy numbers. To be precise:

- Since the membership function has to be continuous, it should intersect with the horizontal axis in at least one point between 0 and the central (core) point of the fuzzy number for the left-side function, and between the central point and 1 for the right-side function. If the function cuts the X axis in more than one point, only the greatest of those between 0 and the central point is considered, as well as the smallest of those between the central point and 1.
- The function only needs to be monotonic in those intervals so local minima cannot exist within them.
- The left and right functions (before normalizing to [0, 1]) must satisfy $f(\text{core}) = g(\text{core}) = 1$, so the degrees of freedom decrease in one because it is possible to write one of the curve parameters as a function of the others.

It is clear that the problem can be viewed as a constrained minimization of the squared error function with respect to the cloud of points representing the α-cuts. First, the `nls` function for Non-linear least squares regression is called. The diminished degrees of freedom are expressed in the formula passed to `nls`, in which one of the parameters of the curve is expressed as a function of the others. This function yields a solution satisfying $f(\text{core}) = 1$, but the rest of the conditions are not guaranteed by `nls` so they have to be checked in the obtained solution. If they are fulfilled, the solution is accepted and returned.

If they are not fulfilled, and taking into account that regression accuracy at this stage is not as fundamental as the satisfaction of all the constraints, again Differential Evolution is used to minimize the total quadratic regression error function for gaussian, quadratic and cubic regression. Constraints were implemented in the objective function of DE. For quadratic regression, the vertex must not fall between the horizontal cut-points and the core, and such cut-points cannot be negative nor greater than 1. For cubic regression, a turning point must

not exist between the cut-points and the core. Linear and gaussian regression are themselves monotonic so the only constraint to be considered is the intersection with the horizontal axis. By definition, the gaussian function does not intersect the horizontal axis at any point but for practical purposes, it can be rounded to 0 when the membership degree is smaller than, say, 0.01.

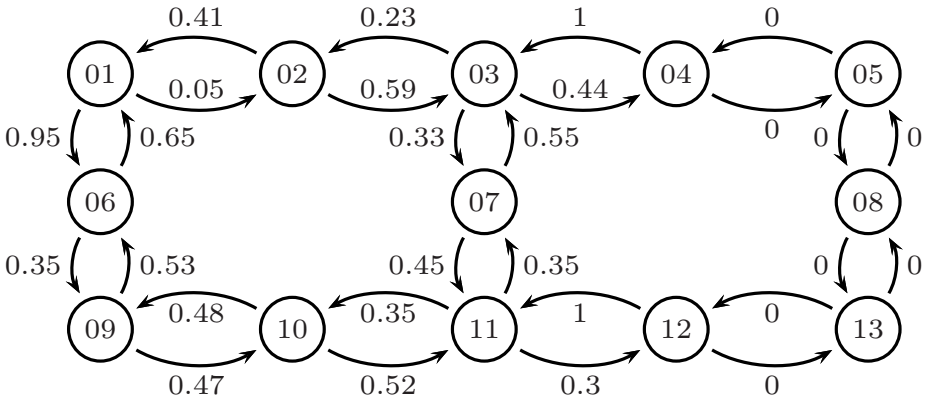
Computation of membership functions via interpolation This option always yields good and fast results, although the expression of the solution is a bit more complicated. Interpolation with cubic spline functions can be applied to the α-cuts to obtain a membership functions with a soft shape, yet continuous. Monotonicity is guaranteed by the use of the *hyman* option of the `splinefun` function of the **stats** package; see the associated *vignette* for details. An alternative interpolation method supported is linear piecewise interpolation, provided by the **FuzzyNumbers** package in the `PiecewiseLinearFuzzyNumber` subclass. The membership function is built by connecting a monotone[1] sequence of points $(x_i, \mu(x_i))$ with straight lines.

Parallelism The most computationally demanding task is the calculation of the α-cuts of fuzzy stationary probabilities. It requires running DE twice per each α value, and it is expected that any user samples at least ten α values, which yields 20 full executions of the optimization process for each state whose stationary probability has to be calculated. In our implementation, when α has been set, the α-cuts are computed for all the chain states indicated by the user, which means that lengthy computations have to be done before moving to another α value. For this reason, and since the computations with each α are independent of the rest, we have introduced parallelism at this level by using the built-in `parLapply` function of the **parallel** package over a vector containing all the α values to be sampled. Each physical core thus takes care of all the computations for a given α, i.e., runs a minimization and a maximization for every state of the chain. With this approach, possibly all the cores available will be engaged in heavy computations since the number of sampled α's will most likely be greater than the number of processors. Such coarse grain parallelization is enough, although the `DEoptim` function provides a built-in option to run in parallel as well. Parallelism is disabled initially in our package (`ncores = 1` by defult) to prevent a sudden slowdown of the computer when getting all the available processors engaged in our computations, but can be enabled by setting `ncores` to a higher number, which is strongly recommended to keep computation times within affordable limits.

4. The package in use

Now we will show how the package can be used with a toy example. We will generate a realization of a 10-state Markov chain which controls the randomized movement of an autonomous robot that patrols an area, modeled as a two-dimensional grid (Figure 2). Each cell of the grid is a state of the chain, and the robot can only move to an adjacent cell at a turn (a discrete time instant). The movement is based on a probability distribution over the adjacent cells. Such distribution depends only on the current cell, and not on the trajectory followed by the robot in the past to reach that cell, so clearly it is a Markovian movement. It

[1] $x_i < x_{i+1} \Leftrightarrow \mu(x_i) \leq \mu(x_{i+1})$ for the left side function and $x_i < x_{i+1} \Leftrightarrow \mu(x_i) \geq \mu(x_{i+1})$ for the right side



(a)

	01	02	03	04	06	07	09	10	11	12
01	0	0.05	0	0	0.95	0	0	0	0	0
02	0.41	0	0.59	0	0	0	0	0	0	0
03	0	0.23	0	0.44	0	0.33	0	0	0	0
04	0	0	1	0	0	0	0	0	0	0
06	0.65	0	0	0	0	0	0.35	0	0	0
07	0	0	0.55	0	0	0	0	0	0.45	0
09	0	0	0	0	0.53	0	0	0.47	0	0
10	0	0	0	0	0	0	0.48	0	0.52	0
11	0	0	0	0	0	0.35	0	0.35	0	0.3
12	0	0	0	0	0	0	0	0	1	0

(b)

Figure 2: (a) Optimal Markovian patrolling strategy according to game-theoretic techniques for a map with 13 cells. Reproduced from Amigoni *et al.* (2009). Non-reachable states 5, 8 and 13 were not considered for the Markov chain. (b) The 10x10 transition matrix.

is time-homogeneous as well, since the probabilities used by the robot do not change along the time. Such models are very common in the field of autonomous robotic patrolling (Agmon, Kraus, and Kaminka 2008; Amigoni *et al.* 2009). There are two main reasons that justify the randomness of this kind of strategies:

- The area is large enough to prevent full coverage with a deterministic patrolling scheme, since some locations would remain uncovered in the sense that the time between two consecutive visits is larger than the time needed by the intruder to successfully penetrate that location.

- The robotic movement should be unpredictable when facing a full-knowledge opponent, i.e., one that has perfectly learnt the robot's patrolling scheme. This situation arises when the opponent has been observing the robot's movement for a long time before choosing the cell to attack. A randomized movement leads to maximizing the robot payoff even in this case, since the only knowledge the intruder may acquire is the exact probability distribution employed by the robot, not the actual trajectory he will follow

at each walk. Note that considering the strongest adversary is the common assumption in game theory as it is the worst case one may face.

Although the problem is usually solved with game-theoretic techniques[2], it represents indeed a good example of a Markov chain with very large state space (actually, proportional to the extension of the area under consideration). The concept of stationary distribution becomes relevant since it gives an idea of the locations in which the robot spends less time or, in other words, the places that have less coverage and are thus the most promising to be attacked[3]. If the assumption of perfect adversarial knowledge is eliminated, i.e., if we consider a more realistic situation in which the adversary only has a sequence of observations of the robot's movement, then what the opponent does is actually an estimation of stationary probabilities based on the observations from an unknown Markov chain that guides the robot's movement. A fuzzy estimation captures more information about the observations than a point estimate.

4.1. Departing from a sequence of observations

The Markov chain of Figure 2 depicts the game-theoretic solution (Amigoni *et al.* 2009) against a full knowledge opponent in a map with 13 cells, three of which are not reachable and thus discarded for our Markov chain. Assume the adversary observes a realization of 200 time instants of this chain. The R code that generates such sequence of observations and then computes the fuzzy stationary probabilities is shown below. `robotStates = c("01", "02", ..., "12")` is a 10-component string vector with the names of the states, and `transRobot` is the transition matrix of Figure 2. Both variables are defined in an `.Rda` file attached to the package. The `states` argument is not specified in the call to `fuzzyStationaryProb`, which means the fuzzy stationary probabilities of all the states should be computed.

```
R> library(markovchain) # for simulating from a known crisp Markov chain
R> mcPatrol <- new("markovchain", states = robotStates, byrow = TRUE,
+ transitionMatrix = transRobot, name = "Patrolling") # Markov chain object
R> set.seed(666);
R> simulatedData <- rmarkovchain(n = 200, object = mcPatrol,
+ t0 = sample(robotStates, 1)) # 200 obs starting in a random state
R> mcfit = markovchainFit(simulatedData) # Fit with markovchain package
R> vsteady = steadyStates(mcfit$estimate) # 1 x n matrix of stat. probs
R> quadratic = fuzzyStationaryProb(simulatedData, list(verbose = TRUE,
+        regression = "quadratic", ncores = 4), step = 0.1);

Parallel computation of a-cuts in progress...finished successfully
(elapsed:  194.04 s.)
Applying quadratic regression to obtain the membership functions of fuzzy
stationary probabilities...
Fitting memb.func. for state: 01  02  03  04  06  07  09  10  11  12
```

Now we demonstrate different types of regression to fit the membership function of the fuzzy stationary probabilities.

[2]Most often the equilibrium strategy in security games is computed using bi-level non-linear programming.

[3]This deserves further discussion since a poorly covered location may not be worth attacking if the benefits of a successful attack are low for the attacker, but intuitively coverage is modeled by the stationary distribution.

```
R> linear = fuzzyStationaryProb(simulatedData, list(verbose = FALSE,
+        regression="linear", ncores = 4), step=0.1)
R> cubic = fuzzyStationaryProb(simulatedData, list(verbose = FALSE,
+        regression = "cubic", ncores = 4), step = 0.1)
R> gaussian = fuzzyStationaryProb(simulatedData, list(verbose = FALSE,
+        regression = "gaussian", ncores = 4), step = 0.1)
R> splines = fuzzyStationaryProb(simulatedData, list(verbose = FALSE,
+        regression = "spline", ncores = 4), step = 0.1)
R> pwlinear = fuzzyStationaryProb(simulatedData, list(verbose = FALSE,
+        regression = "piecewise", ncores = 4), step = 0.1)
```

Finally we are going to depict in a figure the α-cuts of the fuzzy stationary probabilities computed by our program, and the functions fitted to them. The code relies on the plot function for `FuzzyNumber` objects that were created for each state as a result of the regression. In order to compare to the output given by the `steadyStates` function of the **markovchain** package, dashed lines have been drawn at the stationary probabilities calculated by that function, using function `abline`.

```
R> m <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ,11, 11), nrow = 4, ncol = 3,
+        byrow = TRUE) # layout for multiple plots in one figure
R> layout(mat = m, heights = c(0.25, 0.25, 0.25, 0.25))
R> for (i in robotStates){  # i is a string with the state name
+ par(mar = c(4, 4, 2, 1)) # set margin of this cell
+ plot(linear$fuzzyStatProb[[i]], col = "blue", main = paste("State ", i),
+        cex.lab = 1.1, lwd = 2);   # linear regression line in blue
+ plot(quadratic$fuzzyStatProb[[i]], col = "red", cex.lab = 1.1, lwd = 2,
+        add = TRUE);  # quadratic regression curve in red
+ plot(cubic$fuzzyStatProb[[i]], col = "springgreen4", cex.lab = 1.1, lwd = 2,
+        add = TRUE);  # cubic regression curve in green
+ plot(gaussian$fuzzyStatProb[[i]], col = "black", cex.lab = 1.1, lwd = 2,
+        add = TRUE); # gaussian regression curve in black
+ plot(splines$fuzzyStatProb[[i]], col = "orange", cex.lab = 1.1, lwd = 1,
+        add = TRUE);  # cubic spline regression curve in orange
+ abline(v = vsteady[1,i], lty = "dashed"); # vertical dashed line
+ points(linear$acuts[[i]]); # alpha-cuts of this state stat.probability
+ } # Finally, we add a legend box to the figure:
R> plot(1, type = "n", axes = FALSE, xlab = "", ylab = "")
R> plot_colors <- c("blue", "red", "springgreen4", "black", "orange")
R> legend(x = "top", inset = 0,
+ legend = c("Linear", "Quadratic", "Cubic", "gaussian", "Spline"),
+ col = plot_colors, lwd = 2, cex = 1, bty = "n", horiz = FALSE)
R> summary(quadratic)

 . Fuzzy stationary probabilities of a Markov chain with 10 states
 . Probabilities have been computed for states: 01
 . Number of input observations: 200
 . Parameters:
```

```
        Step size: 0.1
        Execution was done in parallel ( 4 cores used )
        Regression curve for output membership functions: quadratic .
        To retrieve the results use $fuzzyStatProb and $acuts with this
        object
 . Computation of alpha-cuts took 194.04 seconds
 . Membership functions regression took 34.71 seconds
```

Let us retrieve the `FuzzyNumber` object corresponding to the stationary probability of state 01, and the α-cuts from which such number was built. The α-cuts are returned as a `data.frame` object in which the first column represents the probability and the second, the membership degree α. Note there are always two rows (rows 1 and 10, 2 and 11, etc) with the same y (membership) value, one for the lower bound and the other for the upper bound of that α-cut. This way, the limits of the α-cuts can be plotted as if they were 2D points.

```
R> quadratic$fuzzyStatProb[["01"]] # FuzzyNumber object of state 01

Fuzzy number with:
   support=[0, 0.294939],
      core=[0.15054, 0.15054].

R> quadratic$acuts[["01"]]

              x     y
1  0.0001480569 0.001 # a-cuts of the fuzzy stationary probability of
2  0.0336437600 0.100 # state named "01"
3  0.0510534166 0.200
4  0.0725788506 0.300
5  0.0936139309 0.400
6  0.1111090020 0.500
7  0.1300688992 0.600
8  0.1387885412 0.700
9  0.1505397380 0.999
10 0.3822915287 0.001
11 0.3046761228 0.100
12 0.2549764457 0.200
13 0.2434278879 0.300
14 0.2104871378 0.400
15 0.1949950486 0.500
16 0.1763232720 0.600
17 0.1625293319 0.700
18 0.1505397380 0.999
```

An important remark should be made here. There are no output α-cuts for $\alpha = 0.8$ and $\alpha = 0.9$. This is due to the fact that the intervals of the input interval matrices were very small, and thus the minimization and maximization problems for each of these values have all

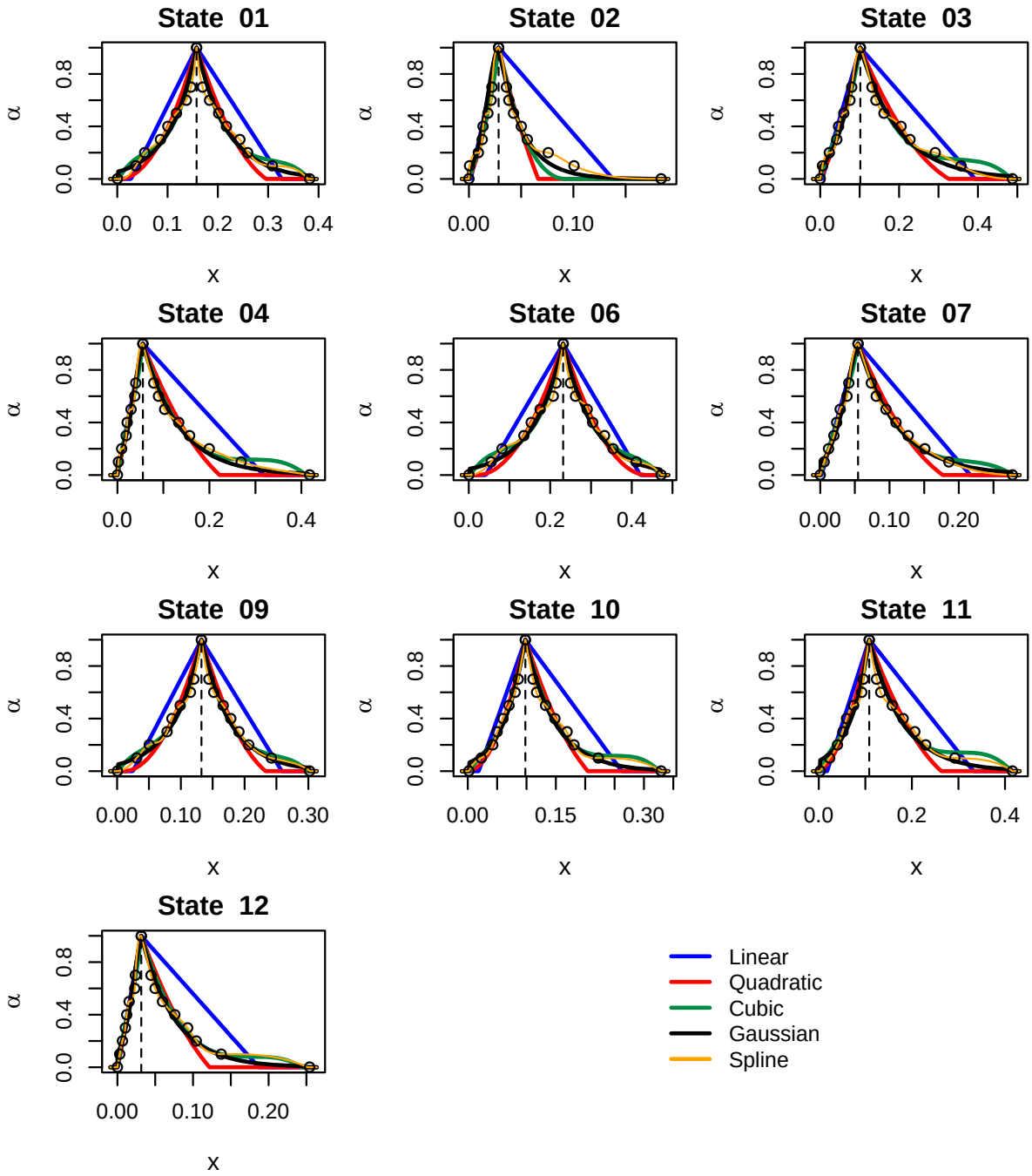


Figure 3: Fuzzy stationary distribution of the Markov chain of Figure 2, computed using 200 observations (several fitting curves available are shown). Dashed lines correspond to stationary probabilities returned by the function `steadyStates` of **markovchain**.

the same solution. It is because the multinomial confidence intervals are very small when the required confidence level is also very low. This happens for $\alpha = 0.8, 0.9$ and 0.999. In practice, the corresponding output α-cuts are points rather than intervals, and actually it is the same point for those three values of α. By the representation theorem (Negoita and Ralescu 1975), the membership degree of all the elements within such very small output α-cuts is actually the greatest of them, i.e., 0.999, so the others can be discarded and the 0.999 is the only one retained. In our implementation, before carrying the regression we always discard those α-cuts such that the distance from the lower or upper bounds of the α-cut to the fuzzy number vertex (which is the point estimate of the stationary probability) is less than 0.005.

The graphical output of the above code is the plot of Figure 3. A number of details can be observed. The gaussian curve usually provides the best fit, and this is actually the reason for which it was included as an option in our package. However, it has the most complicated expression. Cubic fitting also fits quite well. Regarding the shape of the figures, it is clear that the fuzzy numbers are in general not symmetric around the point estimate as the vertex is sometimes very close to 0 or 1 so the points cannot spread much in that side.

Moreover, fuzzy stationary probabilities provide more information than a point estimate based on the transition proportions from the sequence of observations. A comparison between stationary probabilities of states 6 and 8 serves to illustrate how beneficial it is to have fuzzy estimations. According to point estimates (which are the central point of the output fuzzy numbers), the stationary probability of being at state 6 is smaller than that of state 8. However, the α-cuts of fuzzy stationary probability for state 6 are wider and the fuzzy number is more right-skewed than state 8, indicating there is more uncertainty on this value and suggesting that it could be actually greater. In fact, the true stationary probabilities are $\pi_6 = 0.1946, \pi_8 = 0.1154$, and therefore $\pi_6 > \pi_8$. Note, additionally, that the output of the function `steadyStates` from **markovchain** consists of punctual estimates of stationary probabilities. The package is able to compute bootstrap confidence intervals for the transition probabilities but not for the stationary probabilities. Therefore fuzzy transition probabilities are more informative. As could be expected, the point estimates of stationary probabilities given by `steadyStates` match the 1-cuts of our fuzzy stationary probabilities.

4.2. Departing from user-specified fuzzy transition probabilities

Now we depart from a matrix of fuzzy numbers that constitute the fuzzy transition probabilities and are given directly by the user. In this case we assume they are linguistic probabilities as each of them is represented by a (meaningful) linguistic term. However, we could have chosen the names to be meaningless labels like "L_1", "L_2", ..., "L_l" where l stands for the number of different fuzzy numbers employed in the matrix. The linguistic transition matrix created for our problem (Figure 4(a)) is similar to that in Figure 2(a). We have used Trapezoidal Fuzzy Numbers (TrFNs) in accordance to previous studies about the probability ranges that human people associate with each linguistic expression (Bonissone and Decker 1985) (Figure 4(b)). Every row of the fuzzy matrix fulfills the condition of being a well-formed probability distribution in the sense stated in Section 2.4.

The code to compute the fuzzy stationary probabilities from this matrix is the following. Variable `linguisticTransitions` is a matrix of labels (strings) defined in an `.Rdata` file of the package. Its entries should match the names of `allnumbers`, which is a list of `FuzzyNumbers`.

```
R> EU = TrapezoidalFuzzyNumber(0,0,0.02,0.07); # Extremely unlikely
```

	1	2	3	4	5	6	7	8	9	10
1	-	VLC	-	-	ML	-	-	-	-	-
2	IM	-	IM	-	-	-	-	-	-	-
3	-	SC	-	SC	-	IM	-	-	-	-
4	-	-	EL	-	-	-	-	-	-	-
5	MC	-	-	-	-	-	SC	-	-	-
6	-	-	IM	-	-	-	-	-	IM	-
7	-	-	-	-	IM	-	-	IM	-	-
8	-	-	-	-	-	-	IM	-	IM	-
9	-	-	-	-	-	SC	-	IM	-	SC
10	-	-	-	-	-	-	-	-	EL	-

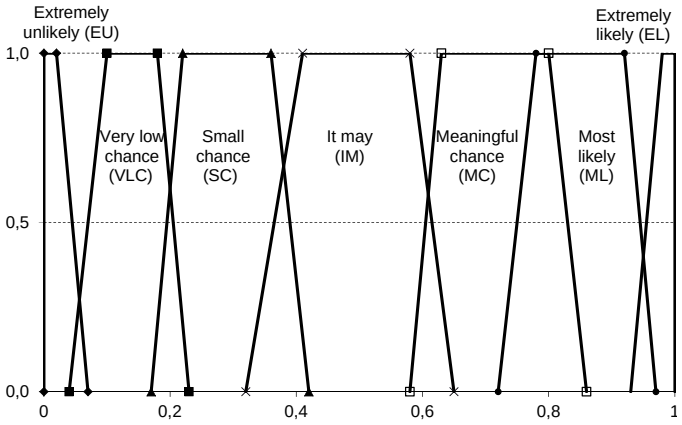


Figure 4: (a) Linguistic transition matrix specified by the user. (b) Associated TrFNs.

```
R> VLC = TrapezoidalFuzzyNumber(0.04,0.1,0.18,0.23); # Very low chance
R> SC = TrapezoidalFuzzyNumber(0.17,0.22,0.36,0.42); # Small chance
R> IM = TrapezoidalFuzzyNumber(0.32,0.41,0.58,0.65); # It may
R> MC = TrapezoidalFuzzyNumber(0.58,0.63,0.8,0.86); # Meaningful chance
R> ML = TrapezoidalFuzzyNumber(0.72,0.78,0.92,0.97); # Most likely
R> EL = TrapezoidalFuzzyNumber(0.93,0.98,1,1); # Extremely likely
R> allnumbers = c(EU,VLC,SC,IM,MC,ML,EL);
+    # The names should match the entries of the linguistic matrix
R> names(allnumbers) = c("EU","VLC","SC","IM","MC","ML","EL");
R> rownames(linguisticTransitions) = robotStates;
R> colnames(linguisticTransitions) = robotStates;
R> linear = fuzzyStationaryProb(linguisticTransitions, list(verbose=TRUE,
+    regression="linear", ncores = 4, fuzzynumbers = allnumbers),step=0.1)
```

The code that plots the results (Figure 5) is similar to the previous section so we will not reproduce it again. Notice the perfectly trapezoidal shape of the output fuzzy sets defined by its α-cuts, mirroring the shape of the input probabilities. In this case, package **markovchain** cannot deal with uncertain transition probabilities directly, in absence of data.

As a concluding remark concerning the uncertainty represented by a fuzzy number, it should be pointed out that, in some cases, fuzzy numbers are asymmetric as they tend to indicate where the true value could be located. This can be observed in some of the plots of Figure 3, specially in states 2 and 4. In both cases, the α-cuts have very large upper bounds at the base, indicating that values greater than the center are more likely than those smaller than it, i.e., the *possibility* that values located in the right side are the true stationary probability is greater. This is the kind of information that crisp numbers cannot provide.

5. Conclusions and further work

We have addressed the problem of estimating stationary probabilities of an unknown Markov chain from a sequence of observations. We first decided to estimate the transition probabilities by using fuzzy numbers, and carry all the computations with them, to obtain fuzzy stationary

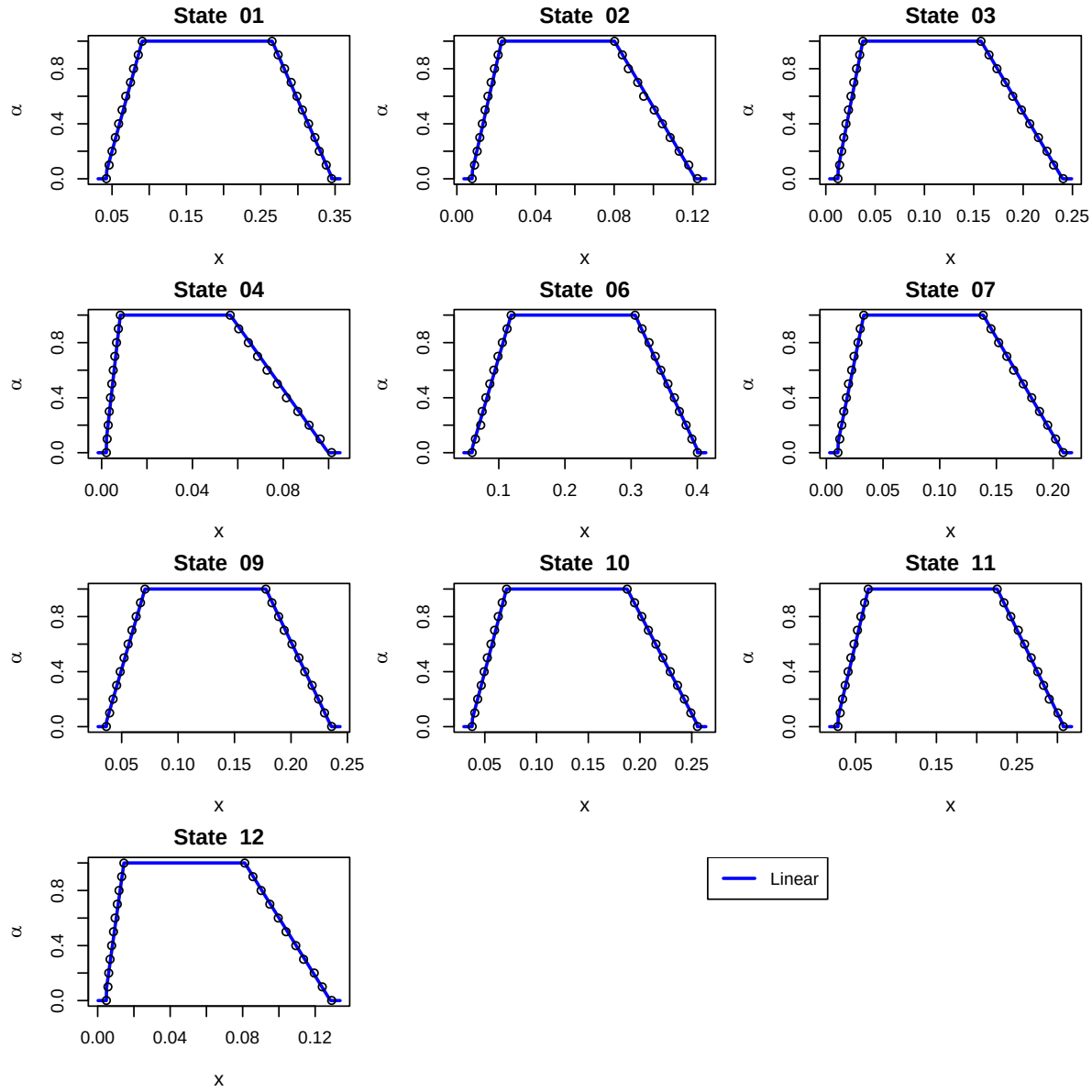


Figure 5: Fuzzy stationary distribution of the Markov chain of Figure 2, computed from user-specified fuzzy transition probabilities.

probabilities. Our work serves as a proof of concept of the method proposed in Buckley (2005) and, at the same time, provides the first freely available, ready-to-use implementation of a fuzzy Markov chain estimation procedure in a widely extended programming environment, namely the R language.

The R package developed for this purpose, **FuzzyStatProb**, has a number of advantages. The main of them is the ease of use: it just requires to have a sequence of observations represented as integers. In addition, it provides great flexibility as it allows the user to specify the kind of regression to be used for the output membership functions, and it can run in parallel, thus exploiting the computational facilities of modern multi-core architectures of conventional computers without any additional software requirement for parallelism. The results rely on the **FuzzyNumbers** package that has been recently published, is under active development and will most likely be adopted by the fuzzy community working on the R implementation of other fuzzy tools and methods.

The implementation has demonstrated the usefulness of the method, and the advantages of having fuzzy estimations for stationary probabilities. Fuzzy numbers are able to better capture uncertainty on the output, and this is indicated by the asymmetric, wider α-cuts. Defuzzification can better approximate the true crisp stationary probabilities. Furthermore, the method paves the way to the computation of linguistic stationary probabilities, which are more intuitive than crisp values and may be in turn the only type of output suitable when the input information about the chain is given in a linguistic manner, which is inherently uncertain.

Further work in this line may include the implementation of other fuzzy Markov chain approaches, mainly those based on fuzzy relations for the transition matrix, and establishing a comparison in the computational effort, usefulness and meaningfulness of the results.

References

Abundo M, Caramellino L (1995). "Some Remarks on a Markov Chain Modelling Cooperative Biological Systems." *Open Systems & Information Dynamics*, **3**, 325–343.

Agmon N, Kraus S, Kaminka G (2008). "Multi-Robot Perimeter Patrol in Adversarial Settings." In *Proc. of the IEEE Conf. on Robotics and Automation*, pp. 2339–2345.

Alves AO, Mota GLA, Costa GAOP, Feitosa RQ (2012). "Estimation of Transition Possibilities for Fuzzy Markov Chains Applied to the Analysis of Multitemporal Image Sequences." In *Proc. of the 4th Int. Conf. on Geographic Object-Based Image Analysis (GEOBIA)*, pp. 367–371.

Amigoni F, Basilico N, Gatti N (2009). "Finding the Optimal Strategies for Robotic Patrolling with Adversaries in Topologically-Represented Eenvironments." In *Proc. of the IEEE Conf. on Robotics and Automation*, pp. 819–824.

Avrachenkov KE, Sanchez E (2002). "Fuzzy Markov Chains and Decision-Making." *Fuzzy Optimization and Decision Making*, **1**, 143–159.

Blanc J, Hertog Dd (2008). "On Markov Chains with Uncertain Data." *Discussion Paper 2008-50*, Tilburg University, Center for Economic Research.

Bonissone P, Decker K (1985). "Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-Off Precision and Complexity." In *Proc. of the Conf. Annual Conference on Uncertainty in Artificial Intelligence (UAI-85)*, pp. 57 – 66.

Buckley JJ (2004). "Uncertain Probabilities III: the Continuous Case." *Soft Computing*, **8**, 200–206.

Buckley JJ (2005). *Fuzzy Probabilities: New Approach and Applications, 2nd Edition*, volume 115 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag.

Buckley JJ, Eslami E (2003). "Uncertain Probabilities I: the Discrete Case." *Soft Computing*, **7**, 500–505.

Buckley JJ, Eslami E (2004). "Uncertain Probabilities II: the Continuous Case." *Soft Computing*, **8**, 193–199.

Buckley JJ, Eslami E (2008). "Fuzzy Markov Chains: Uncertain Probabilities." *Mathware & Soft Computing*, **9**(1).

Ching WK, Ng MK (2006). *Markov Chains: Models, Algorithms and Applications*. Springer-Verlag, New York.

Dent W, Ballintine R (1971). "A Review of the Estimation of Transition Probabilities in Markov Chains." *Australian Journal of Agricultural Economics*, **15**(2), 69–81.

Dubois D, Prade H (1978). "Operations on Fuzzy Numbers." *International Journal of Systems Science*, **9**(6), 613–626.

Feitosa RQ, Costa GAOP, Mota GLA, Feijó B (2011). "Modeling Alternatives for Fuzzy Markov Chain-Based Classification of Multitemporal Remote Sensing Data." *Pattern Recognition Letters*, (32), 927 – 940.

Gagolewski M (2012). ***FuzzyNumbers** Package: Tools to Deal with Fuzzy Numbers in R*. URL http://www.ibspan.waw.pl/~gagolews/FuzzyNumbers/.

Gentleman RC, Lawless JF, Lindsey JC, Yan P (1994). "Multi-State Markov Models for Analyzing Incomplete Disease History Data with Illustrations for HIV Disease." *Statistics in Medicine*, **13**(8), 805–821.

Gomez S, Arenas A, Borge-Holthoefer J, Meloni S, Moreno Y (2010). "Discrete-Time Markov Chain Approach to Contact-Based Disease Spreading in Complex Networks." *Euro Physics Letters (EPL)*, **89**(38009), 6.

Halliwell J, Shen Q (2008). "Linguistic Probabilities: Theory and Application." *Soft Computing*, **13**(2), 169–183.

Ivanek R, Grohn YT, Ho AJJ, Wiedmann M (2007). "Markov Chain Approach to Analyze the Dynamics of Pathogen Fecal Shedding - Example of Listeria Monocytogenes Shedding in a Herd of Dairy Cattle." *Journal of Theoretical Biology*, **245**(1), 44–58.

Jones CI (1997). "On the Evolution of the World Income Distribution." *Journal of Economic Perspectives*, **11**(3), 19–36.

Juang BH, Rabiner LR (1991). "Hidden Markov Models for Speech Recognition." *Technometrics*, **33**(3), 251–272.

Kleyle R, de Korvin A (1997). "Transition Probabilities for Markov Chains Having Fuzzy States." *Stochastic Analysis and Applications*, **15**(4), 527–546.

Kruse R, Buck-Emden R, Cordes R (1987). "Processor Power Considerations - An Application of Fuzzy Markov Chains." *Fuzzy Sets and Systems*, **21**(3), 289–299.

Kurano M, Yasuda M, Nakagami J, Yoshida Y (1992). "A Limit Theorem in some Dynamic Fuzzy Systems." *Fuzzy Sets and Systems*, **51**(1), 83 – 88.

May WL, Johnson WD (2000). "Constructing Two-Sided Simultaneous Confidence Intervals for Multinomial Proportions for Small Counts in a Large Number of Cells." *Journal of Statistical Software*, **5**(6), 1–24.

Medhi J (2002). *Stochastic Models in Queueing Theory, 2nd Edition*. Academic Press, Boston.

Mullen KM, Ardia D, Gil DL, Windover D, Cline J (2011). "DEoptim: An R Package for Global Optimization by Differential Evolution." *Journal of Statistical Software*, **40**(6), 1–26.

Negoita C, Ralescu D (1975). *Applications of Fuzzy Sets to Systems Analysis*. John Wiley & Sons.

Newton PK, Mason J, Bethel K, Bazhenova LA, Nieva J, Kuhn P (2012). "A Stochastic Markov Chain Model to Describe Lung Cancer Growth and Metastasis." *PLoS ONE*, **7**(4), e34637.

Price K, Storn R, Lampinen J (2005). *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag - Natural Computing Series.

R Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. URL http://www.R-project.org/.

Sahin AD, Sen Z (2001). "First-Order Markov Chain Approach to Wind Speed modelling." *Journal of Wind Engineering and Industrial Aerodynamics*, **89**(3-4), 263–269.

Satten GA, Longini IM (1996). "Markov Chains With Measurement Error: Estimating the 'True' Course of a Marker of the Progression of Human Immunodeficiency Virus Disease." *Journal of the Royal Statistical Society C*, **45**(3), 275–309.

Schulze W, van der Merwe B (2011). "Music Generation with Markov Models." *IEEE Multimedia*, **18**, 78–85.

Sison CP, Glaz J (1995). "Simultaneous Confidence Intervals and Sample Size Determination for Multinomial Proportions." *Journal of the American Statistical Association*, **90**(429), 366–369.

Storn R, Price K (1997). "Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces." *Journal of Global Optimization*, **11**(10), 341–359.

Tran D, Wagner M (1999). "Fuzzy Hidden Markov Models for Speech and Speaker Recognition." In *Proc. of 18th Int. Conf. of the North American Fuzzy Information Processing Society (NAFIPS)*, pp. 426–430.

van der Hoek J, Elliott RJ (2012). "American Option Prices in a Markov Chain Market Model." *Applied Stochastic Models in Business and Industry*, **28**(1), 35–59.

Villacorta PJ (2012). *Package **MultinomialCI**: Simultaneous Confidence Intervals for Multinomial Proportions According to the Method by Sison and Glaz*. URL http://cran.r-project.org/web/packages/MultinomialCI/MultinomialCI.pdf.

Villacorta PJ, Verdegay JL, Pelta DA (2013). "Towards Fuzzy Linguistic Markov Chains." In *Proc. of the 8th Conf. of the European Society for Fuzzy Logic and Technology (EUSFLAT)*, pp. 707–713.

Welton NJ, Ades AE (2005). "Estimation of Markov Chain Transition Probabilities and Rates from Fully and Partially Observed Data: Uncertainty Propagation, Evidence Synthesis, and Model Calibration." *Medical Decision Making*, **25**(6), 633–645.

Zadeh LA (1965). "Fuzzy Sets." *Information and Control*, **8**(3), 338 – 353. ISSN 0019-9958.

Zadeh LA (1975a). "The Concept of a Linguistic Variable and its Application to Approximate Reasoning - I." *Information Sciences*, **8**(3), 199 – 249.

Zadeh LA (1975b). "The Concept of a Linguistic Variable and its Application to Approximate Reasoning - III." *Information Sciences*, **9**(1), 43–80.

Zadeh LA (1998). "Maximizing Sets and Fuzzy Markoff Algorithms." *IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, (28), 9 – 15.

Affiliation:

Pablo J. Villacorta, José L. Verdegay
Models of Decision and Optimization (MODO) Research Group
CITIC-UGR, Department of Computer Science and Artificial Intelligence
University of Granada
18071 Granada, Spain
E-mail: pjvi@decsai.ugr.es, verdegay@decsai.ugr.es
URL: http://decsai.ugr.es/~pjvi, http://decsai.ugr.es/~verdegay,
http://modo.ugr.es