Package 'FuzzyStatProb'

January 20, 2025

License LGPL (>= 3)

Author Pablo J. Villacorta <pjvi@decsai.ugr.es>

Maintainer Pablo J. Villacorta <pjvi@decsai.ugr.es>

URL http://decsai.ugr.es/~pjvi/r-packages.html

LazyData true

Title Fuzzy Stationary Probabilities from a Sequence of Observations of an Unknown Markov Chain

Type Package

Description An implementation of a method for computing fuzzy numbers representing stationary probabilities of an unknown Markov chain, from which a sequence of observations along time has been obtained. The algorithm is based on the proposal presented by James Buckley in his book on Fuzzy probabilities (Springer, 2005), chapter 6. Package 'FuzzyNumbers' is used to represent the output probabilities.

Version 2.0.4

Date 2019-2-9

Imports MultinomialCI, parallel, FuzzyNumbers, DEoptim

Suggests markovchain, R.rsp

VignetteBuilder R.rsp

NeedsCompilation no

Repository CRAN

Date/Publication 2019-02-09 17:33:13 UTC

RoxygenNote 6.0.1

Contents

`fuzzyStationaryProb` *Fuzzy stationary probabilities of Markov chains from observations*

Description

Computation of LR fuzzy numbers representing fuzzy stationary probabilities of an unknown Markov chain from which a sequence of observations has been drawn. The fuzzy Markov chain considered during the processing follows the approach proposed by J. Buckley (see the reference section).

Usage

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

Arguments

`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. When the transition probability from state i to j is 0 (in the crisp sense), then entry (i,j) must be NA. The `colnames` and `rownames` of the `data` matrix should have been set before calling this function.

`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.
- `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 interpolation). In all cases (including the gaussian), a different curve is fitted for each side of tue fuzzy number. The `gaussian` option fits curves of the form $\mu(x) = \exp(-1/2|(x-c)/s|^m)$. 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.
- `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.

- `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 `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 $alpha$ that is always present equals 0.001, and the rest of values are calculated as $\alpha = k$ `step` for $k \geq 1$. The greatest sampled value that is always present as well is $\alpha = 0.999$. Defaults to 0.05 when 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.

Details

Given a sequence of consecutive observations of the state of the chain, a fuzzy transition matrix is constructed according to the approach proposed in J. Buckley's *Fuzzy Probabilities* book. Fuzzy transition probabilities are constructed as the superposition of intervals (α-cuts), which in this case represent simultaneous confidence intervals for multinomial proportions, and are computed using the input sequence of observations drawn from the chain. For each value of α, the α-cuts of such fuzzy transition probabilities define a matrix space where we seek for the the matrices producing respectively the minimum and maximum possible stationary probability for each state of the chain, using heuristic optimization tools (Differential Evolution). Both points define a closed real interval that is indeed an α cut of the output fuzzy number representing the fuzzy stationary probability for that state. Solving these problems for different α allows to reconstruct the fuzzy stationary probabilities from their α-cuts, applying the decomposition theorem. Regression is applied at the final stage to compute the membership functions of the stationary probabilities.

Value

An object of the new S3 class `FuzzyStatObj`, which is a tagged list with the following components:

`fuzzyStatProb` A list of `FuzzyNumber` objects. The length of the list equals that of the `states` tag of the `options` argument. The object at a given position `i` corresponds to the fuzzy stationary probability of the state indicated at position i of the `states` vector. If any of the states indicated in the `states` option is not found in the `data` input vector, the corresponding position in `fuzzyStatProb` will be NA. If the function was called with `acutsonly` set to TRUE, then the returned object will not have a `fuzzyStatProb` tag.

`acuts` A list of data frame objects containing the α-cuts of every fuzzy stationary probability, represented as bidimensional points (lowerBound,α) and (upperBound,α) where $\tilde{\pi}(\alpha) = [lowerBound, upperBound]$ is an α-cut of the fuzzy number $\tilde{\pi}$. The length of the list also equals that of the `states` tag of the `options` argument. Again, object at position `i` corresponds to α-cuts of the state indicated at

position `i` of the `states` vector of the option list. If any of the states indicated in the `states` option is not found in the `data` input vector, the corresponding position in `acuts` will be NA.

References

Buckley, J.J. Fuzzy Probabilities: New Approach and Applications, 2nd edition, volume 115 of Studies in Fuzziness and Soft Computing. Springer, 2005.

Glaz, J. and C.P. Sison. Simultaneous confidence intervals for multinomial proportions. Journal of Statistical Planning and Inference 82:251-262 (1999).

May, W.L. and W.D. Johnson. Constructing two-sided simultaneous confidence intervals for multinomial proportions for small counts in a large number of cells. Journal of Statistical Software 5(6) (2000). Paper and code available at http://www.jstatsoft.org/v05/i06.

Gagolewski M. FuzzyNumbers Package: Tools to deal with fuzzy numbers in R (2012). Tutorial available at http://www.ibspan.waw.pl/~gagolews/FuzzyNumbers/doc/FuzzyNumbers-Tutorial.pdf

Amigoni, F., Basilico, N., Gatti, N. Finding the Optimal Strategies for Robotic Patrolling with Adversaries in Topologically-Represented Eenvironments. In Proc. of ICRA 2009, pp. 819-824.

See Also

markovchainFit

Examples

```
# ----------------- CREATE DATA ----------
# Simulate 200 observations of a 10-state Markov chain,
# and compute fuzzy stationary probability of state 1
if(require("markovchain")){ # for simulating from a known crisp Markov chain
# Transition matrix taken from Fig. 1 of Amigoni et al. (see references)
mcPatrol <- new("markovchain", states = robotStates, byrow = TRUE,
transitionMatrix = transRobot, name = "Patrolling")
set.seed(666)
simulatedData <- rmarkovchain(n = 200, object = mcPatrol, t0 =
  sample(robotStates, 1))
mcfit = markovchainFit(simulatedData) # Fit with markovchain package
vsteady = steadyStates(mcfit$estimate) # 1 x n matrix of stat. probs
# ----------------------------------------
# Simplest case: compute only alpha-cuts for alpha=0.001 and alpha=0.999
# Set itermax to 30 (too few) just for a fast example (not good results)
linear = fuzzyStationaryProb(simulatedData,list(verbose=TRUE, states="01",
   regression="piecewise"), step=1, itermax = 30)
summary(linear)
linear$fuzzyStatProb[["01"]]
plot(linear$fuzzyStatProb[["01"]])
points(linear$acuts[["01"]])
}
## Not run:
# A more accurate approximation, with steps of 0.1 (takes much longer!)
# Run the previous code to create mcPatrol, vsteady and simlatedData
quadratic = fuzzyStationaryProb(data = simulatedData,list(verbose=TRUE,
```

```
  ncores = 2, regression="quadratic"), step=0.1)
m <- matrix(c(1,2,3,4,5,6,7,8,9,10,11,11),nrow = 4,ncol = 3,byrow = TRUE)
layout(mat = m,heights = c(0.25,0.25,0.25,0.25))
for (i in robotStates){
par(mar = c(4,4,2,1))
    plot(quadratic$fuzzyStatProb[[i]],col="red",main=paste("State ",i),
      cex.lab = 1.1,lwd=2);
    points(quadratic$acuts[[i]]);
    abline(v = vsteady[1,i], lty = "dashed");
}
plot(1, type = "n", axes=FALSE, xlab="", ylab="")
plot_colors <- c("red")
legend(x = "top",inset = 0, legend = c("Quadratic"), col=plot_colors,
  bty = "n", lwd=2, cex=1, horiz = FALSE)

# Now departing from user-specified fuzzy transition probabilities
library(FuzzyNumbers)
EU = TrapezoidalFuzzyNumber(0,0,0.02,0.07); # Extremely unlikely
VLC = TrapezoidalFuzzyNumber(0.04,0.1,0.18,0.23); # Very low chance
SC = TrapezoidalFuzzyNumber(0.17,0.22,0.36,0.42); # Small chance
IM = TrapezoidalFuzzyNumber(0.32,0.41,0.58,0.65); # It may
MC = TrapezoidalFuzzyNumber(0.58,0.63,0.8,0.86); # Meaningful chance
ML = TrapezoidalFuzzyNumber(0.72,0.78,0.92,0.97); # Most likely
EL = TrapezoidalFuzzyNumber(0.93,0.98,1,1); # Extremely likely
allnumbers = c(EU,VLC,SC,IM,MC,ML,EL);
names(allnumbers) = c("EU","VLC","SC","IM","MC","ML","EL");
rownames(linguisticTransitions) = robotStates; # see the package data
colnames(linguisticTransitions) = robotStates;

# Simplest case: compute only alpha-cuts for alpha=0.001 and alpha=0.999
# linguisticTransitions is a matrix of strings defined in the package data
linear = fuzzyStationaryProb(linguisticTransitions,list(verbose=TRUE,
  regression="linear", ncores = 4, fuzzynumbers = allnumbers),step=0.2)
summary(linear)

## End(Not run)
```

`robotMatrix` *Transition matrix of a Markov chain that guides the movement of an autonomous patrolling robot*

Description

Transition matrix of a Markov chain that guides the movement of an autonomous patrolling robot. The chain has been computed according to game-theoretic techniques as the equilibrium solution of a leader-follower game between a potential intruder and a patroller. See the references section.

Usage

```
robotStates
```

transRobot
linguisticTransitions

Format

transRobot is a 10x10 2D matrix, linguisticTransitions is another 10x10 matrix of strings, and robotStates is a vector of state names of length 10.

Details

In the game-theoretic patrolling model proposed in Amigoni et al., the equilibrium solution of the leader-follower game is a Markov chain that can be computed by solving a set of independent linear programming problems. The transition probabilites are described in Fig. 1 of Amigoni et al. linguisticTransitions is a matrix of labels whose names should match the tags of the fuzzynumbers list argument in the call to fuzzyStationaryProb when linguisticTransitions is passed as first argument.

Author(s)

Pablo J. Villacorta Iglesias, Department of Computer Science and Artificial Intelligence, University of Granada (Spain).

<pjvi@decsai.ugr.es> - http://decsai.ugr.es/~pjvi/r-packages.html

References

Amigoni, F., Basilico, N., Gatti, N. Finding the Optimal Strategies for Robotic Patrolling with Adversaries in Topologically-Represented Eenvironments. In Proc. of ICRA 2009, pp. 819-824.

Index