# stratEst: Strategy Estimation in R*

Fabian Dvorak
University of Konstanz
fabian.dvorak@uni.kn

This version: December 2018

**Abstract**

stratEst is a statistical software package which can be used to characterize the choices of a sample of individuals as a mixture of individual strategies. Strategies can be estimated from the data or supplied by the user in the form of deterministic finite-state automata. The package uses the EM algorithm (Dempster et al., 1977) and the Newton-Raphson method to obtain maximum-likelihood estimates of the population shares and choice parameters of the strategies. The number and the complexity of strategies can be restricted by the user or selected based on information criteria. The package also features an extension of strategy estimation in the spirit of latent class regression to assess the effects of covariates on strategy use.

**Keywords:** experimental games, individual differences, mixture models

**JEL Classification:** C13, C87, C91, C92

*Please cite this document as: Dvorak, F. (2019). stratEst: Strategy Estimation in R. Working paper version December 2018. http://fabian-dvorak.com/software/stratEst-paper-181201.pdf. I would like to thank Yongping Bao, Yves Breitmoser, Karsten Donnay, Urs Fischbacher, Konstantin Käppner and Susumu Shikano for helpful comments. I am particularly grateful to Sebastian Fehrler for many inspiring conversations. All remaining errors are my own.

## 1 Introduction

stratEst is a software package for the statistical computing environment R (R Development Core Team, 2008). The package implements variants of the strategy estimation method (Dal Bó and Fréchette, 2011). The goal of strategy estimation is to characterize the choices of a sample of individuals as a mixture of individual strategies. Strategy estimation is similar to mixture modeling (McLachlan and Peel, 2005), cluster analysis (Kaufman and Rousseeuw, 1990), and latent class analysis (Lazarsfeld, 1950). All three methods essentially group several entities into several unobservable classes. In the case of strategy estimation, the entities are individuals and the unobservable classes are strategies in the sense of game-theory. A strategy is a complete action plan which prescribe a behavioral response for every situation in a game.[1]

Different variants of strategy estimation exist (Dal Bó and Fréchette, 2011; Breitmoser, 2015; Dvorak and Fehrler, 2018). In one variant, theory indicates a set of reasonable candidate strategies and the researcher is interested to estimate the frequency of the strategies in the population given the sampled data. In another variant, reasonable candidate strategies are unknown and the researcher wants to learn something about the the choice parameters of the strategies while imposing some basic assumptions on the number and the general structure of the strategies. In yet another variant, the researcher is interested in how strategy use is influenced by covariates such as treatment conditions or time.

The main challenge for strategy estimation software is to guarantee a sufficient degree of flexibility across the different variants of the method for large number of games. To address this issue, the stratEst package handles strategies in the form of deterministic finite-state automata.[2] Deterministic finite-state automata map all possible situations of a game into a finite set of strategy-specific states. The behavioral response of an individual following a certain strategy is then a function of the the strategy-specific state and not the situation itself. The handling of strategies as deterministic finite-state automata offers a concise way to customize strategies for many different games and even works for games with infinitely many situations.

The current version of the package is limited to the estimation of strategies which prescribe discrete choices. The central modeling assumption is that the observed actions are independent draws from a multinomial distribution with parameters defined by the current state

[1]Researchers not interested in social interaction may think of a strategy as a behavioral algorithm for a specific decision-making environment instead.

[2]The strategy estimation variants two and three can in principle also be conducted based on R packages for cluster analysis like Flexmix (Leisch, 2004), poLCA (Linzer and Lewis, 2011) and randomLCA (Beath, 2011). The disadvantage of (mis)using these packages for strategy estimation is that all candidate strategies must have the same simple structure and cannot be reasonably adapted to the game or the decision-making environment.

of the strategy used. Maximum-likelihood estimates for the model parameters are obatined based on variants of the Expectation-Maximization algorithm (Dempster et al., 1977) and the Newton-Raphson method. To increase speed the estimation procedures, stratEst uses integration of C++ and R through Rcpp (Eddelbuettel and François, 2011) as well as the open source linear algebra library for the C++ language RppArmadillo (Sanderson and Curtin, 2016). Package development is supported by the packages devtools (Wickham et al., 2018b), testthat (Wickham, 2011), roxygen2 (Wickham et al., 2018a), knitr (Xie, 2018), and R.rsp (Bengtsson, 2018).
The introduction continues with information on how to install the package and two examples. Section 2 of the paper illustrates how strategies are represented as deterministic finite-state automata. Section 3 introduces the strategy estimation method. The general model and algorithm that is used to obtain maximum-likelihood estimates of the model parameters is introduced. Section 4 covers model selection. It explains how the number of model parameters can be restricted by the user or selected based on information criteria. Section 5 introduces the extension of the strategy estimation method in the spirit of latent class regression to assess the role of covariates for strategy use. Section 6 explains the estimation procedures for the standard errors of the model parameters. Section 7 illustrates the validity of the estimation procedures based on a simulation exercise. Section 8 gives an overview over the syntax of the estimation function and its input and output objects.

## Installation

The most recent CRAN version of the stratEst package can be installed by executing the following code in the R console:

```
install.packages("stratEst")
```

You can also install the most recent development version of the package from GitHub using the devtools package:

```
install.packages("devtools")
library(devtools)
install_github("fdvorak/stratEst")
```

After successful installation, the package is loaded into memory and attached to the search path in the usual way by:

```
library(stratEst)
```

Now the package is ready to use.

## An introductory example

In this example, the stratEst package is used to perform strategy estimation based on the fictitious data depicted in Figure 1. It will be useful to assume that the data set contains the data of two individuals playing four periods of a helping game with each other. In each period, both individuals simultaneously decide to help the other individual or not. Helping is costly but receiving help implies a monetary benefit which exceeds the costs of helping. Each row of the data shown in Figure 1 represents the action of one individual in one period of the game. The first four columns identify the individual, the pairwise matching of individuals, the game, and the period within the game. The fifth column contains a dummy variable which is one if the individual helped the other individual in the respective period and zero otherwise.

Figure 1: Fictitious data of a helping game

| *id* | *group* | *game* | *period* | *help* |
|---|---|---|---|---|
| 62 | 13 | 4 | 1 | 1 |
| 62 | 13 | 4 | 2 | 1 |
| 62 | 13 | 4 | 3 | 0 |
| 62 | 13 | 4 | 4 | 1 |
| 87 | 13 | 4 | 1 | 1 |
| 87 | 13 | 4 | 2 | 0 |
| 87 | 13 | 4 | 3 | 1 |
| 87 | 13 | 4 | 4 | 0 |

The goal of strategy estimation is to explain the actions of the two individuals based on strategies that take the behavior of the other individual into account. Two tasks have to be completed by the user before the estimation can be performed.
The first task is to define a common set of inputs and outputs for the candidate strategies. Inputs represent information that can be observed by a player at a certain stage of the game and potentially influence her continuation strategy. Outputs represent possible actions at a certain stage of the game. If both individual can observe their actions, it is natural to define the action profile of the previous period as the new information the strategies react to on a regular basis. In the following, the action profiles $(help, help)$, $(help, no)$, $(no, help)$, $(no, no)$ will be represented by the input values 1, 2, 3, and 4 respectively. The input value 0 is special and must be used at the beginning of a game when no information about past play is available. For the outputs, the two possible actions *no help* and *help* will be represented by the values 1 and 2 in the following. Top complete the first task, the data depicted in Figure 1 must be stored as an R data frame augmented by the variables `input` and `output` (see Subsection 8.1 for information on other data structures). This is achieved by executing

the following statements in the R console:

```
id <- c(62,62,62,62,87,87,87,87)
game <- c(4,4,4,4,4,4,4,4)
period <- c(1,2,3,4,1,2,3,4)
input <- c(0,1,2,3,0,1,3,2)
output <- c(2,2,1,2,2,1,2,1)
data <- as.data.frame(cbind(id,game,period,input,output))
```

The second task is to define which outputs follow after a each possible combination of inputs for each candidate strategy. In the following two candidate strategies are defined. One reciprocal strategy, that randomizes in the first period and subsequently helps if the other participant helped in the previous period, and one alternating strategy, that helps with probability 0.9 in uneven periods and with unknown probability in even periods.[3] Each of these two strategies can be represented by a deterministic finite-state automaton (see Section 2 for more information). The following code creates an R matrix which contains the deterministic finite-state automaton representations of the two strategies:

```
strategies <- matrix(c(1,2,3,1,2,0.5,0,1,0.1,NA,0.5,1,0,0.9,NA,2,2,2,2,1,
                       3,3,3,2,1,2,2,2,2,1,3,3,3,2,1),5,7)
```

Printed out in the console, the matrix looks like this:

```
> strategies
      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
 [1,]    1  0.5  0.5    2    3    2    3
 [2,]    2  0.0  1.0    2    3    2    3
 [3,]    3  1.0  0.0    2    3    2    3
 [4,]    1  0.1  0.9    2    2    2    2
 [5,]    2   NA   NA    1    1    1    1
```

Each row of the matrix represents one state of an automaton. The first three rows of the matrix define the reciprocal strategy. The last two rows of the matrix define the alternating strategy. The first column of the matrix enumerates the states of within each automaton. A row with the value 1 in the first column represents the start state of the automaton. By definition, a strategy is in its start state whenever the current input is zero. The second

[3]Inspection of the data reveals, that the reciprocal strategy perfectly describes the behavior of participant 62 while the alternating strategy provides a better description of the behavior of participant 87. With more data, such inference will of course not be possible and it will be necessary to define reasonable candidate strategies based on theory.

column indicates the probability of the output which has the lowest output value for the respective state. In the example, the output with the lowest value is the one that represents *no help* and the probability to observe this action as output in the start state is one half for the reciprocal strategy. The third column indicates the probability of the output with the next higher value which represents the action *help*. The pattern according to which an automaton moves from one state to the next over the periods of the game is defined in columns four to seven. The value 2 in the first row of column four indicates that the reciprocal strategy moves from state 1 to state 2 if the input is 1. The value 3 in the first row of column five indicates that the reciprocal strategy moves from state 1 to state 3 if the input is 2, and so on.[4] Rows four and five of the matrix `strategies` define the alternating strategy in a similar fashion. The fact that the probability to help in even periods is ex-ante unknown is indicated by inserting NA in column two.[5]
The central function of the package is the function `stratEst()`. It is the estimation function which implements all variants of strategy estimation (see Section 8 for more information). The following code is executed in the console to obtain maximum-likelihood estimates of the population shares and the choice parameters of the two candidate strategies:

```
model <- stratEst(data,strategies)
```

The output objects of the function are stored as anR object of type list and can be called by using the syntax `model$object` where `object` can be one of the output objects (see Subsection 8.2). To display the estimates of the population shares and the final strategy matrix the objects `model$ shares` and `model$strategies` can be used to display the following results:

```
> model$shares      > model$strategies
      [,1]                [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]
 [1,]  0.5           [1,]  1     0.5   0.5   2     3     2     3
 [2,]  0.5           [2,]  2     0.0   1.0   2     3     2     3
                     [3,]  3     1.0   0.0   2     3     2     3
                     [4,]  1     0.1   0.9   2     2     2     2
                     [5,]  2     1.0   0.0   1     1     1     1
```

The first element of the vector `model$shares` indicates that the estimated share of individuals in the sample that use the first strategy defined in `strategies` is 50%. The second

[4]Note that inputs were defined such that inputs 1 and 3 indicate help of the other participant in the previous period. As the reciprocal strategy moves to state 2 after these inputs, and helps with probability of one. The probability to help after observing input 2 or 4 which indicate no help, is zero as the strategy moves to state 3.

[5]Whenever NA is supplied for a model parameter, stratEst will estimate this parameter from the data. See Section 3 for information on the general model and its parameters.

element reveals that the estimated share of the second strategy is also 50%. The matrix `model$strategies` returns the strategies which correspond to the estimated population shares in `model$shares`. The format of the matrix corresponds to the format of the input object `strategies`. The last element in the second column of the estimated strategies indicates that the ex-ante unknown probability to help in even periods when using the alternating strategy is estimated to be zero which coincides with the behavior of the second individual in period four.

## Repeated prisoner's dilemma example

This example illustrates how to perform strategy estimation based on data from an indefinitely repeated prisoner's dilemma.[6] It is shown how to replicate the results of Dal Bó and Fréchette (2011) which was the first paper that performed strategy estimation. The paper reports results on the evolution of cooperation in the indefinitely repeated prisoner's dilemma across six different treatments. The six treatments differ in the stage-game parameters and the continuation probability $\delta$ of the repeated game. The stage-game parameters are depicted in Figure 2 where the parameter $R$ is either 32, 40 or 48. For each value of $R$ two treatments exist with $\delta$ of 1/2 or 3/4 resulting in 2 times three between subject design with six treatments overall. Figure 3 displays the first 8 rows of the data of the experiment

Figure 2: Stage game of Dal Bó and Fréchette (2011)

|   | C | D |
|---|---|---|
| C | R,R | 12,50 |
| D | 50,12 | 25,25 |

conducted by Dal Bó and Fréchette (2011). The first column identifies the treatment. The second column contains an identifier of the participants. Column three enumerates the supergame as each participants plays many repeated games during the experiment. Column four indicates the period within the supergame. Column five and six contain dummy variables which indicate whether the participant and the other player cooperated in the current period of the supergame. The data is available as an $R$ data frame and the first 8 rows can

[6]The indefinitely repeated prisoner's dilemma is special case in two ways: First, data can be submitted in a format specific to this game. Second, the package comes with a set of 22 pre-programmed strategies for the repeated prisoner's dilemma listed in Tables A.1-A.3 of the Appendix.

Figure 3: First ten rows of data from Dal Bó and Fréchette (2011)

| *treatment* | *id* | *supergame* | *period* | *cooperation* | *other_cooperation* |
|---|---|---|---|---|---|
| 1 | 1 | 62 | 1 | 0 | 0 |
| 1 | 1 | 63 | 1 | 0 | 0 |
| 1 | 1 | 63 | 2 | 1 | 0 |
| 1 | 1 | 63 | 3 | 0 | 1 |
| 1 | 1 | 64 | 1 | 0 | 0 |
| 1 | 1 | 64 | 2 | 1 | 0 |
| 1 | 1 | 64 | 3 | 0 | 0 |

be inspected in the console by typing `DF2011[1:8,]`.[7]

Dal Bó and Fréchette (2011) report the results of treatment-wise strategy frequency estimation for six candidate strategies which are: Always Defect (ALLD), Always Cooperate (ALLC), Tit-For-Tat (TFT), Grim-Trigger (GRIM), Win-Stay-Lose-Shift (WSLS), and a trigger strategy with two periods of punishment (T2). The six strategies are included in a set of the pre-programmed repeated prisoner's dilemma listed in Tables A.1-A.3 of the Appendix. The following code can be used to replicate the findings of Dal Bó and Fréchette (2011), where `treatment` $\in \{1, \cdots, 6\}$ specifies the treatment number.

```
data <- DF2011[DF2011$treatment == 1,]
strategies <- rbind(ALLD,ALLC,GRIM,TFT,WSLS,T2)
model <- stratEst(data,strategies)
```

The estimated population shares can be inspected with the command `model$shares` and are identical to those reported in the first column of Table 7 on page 424 of Dal Bó and Fréchette (2011).

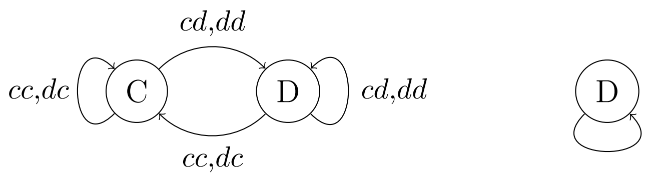
## 2 Strategies as Deterministic Finite-State Automata

Candidate strategies can be customized by the user and handed over to the estimation function in the form of a deterministic finite-state automata (DFA). The DFA representation of a strategy groups all situations where a strategy prescribes an identical behavioral response into one state of the automaton. To give an example, consider the Tit-For-Tat (TFT)

[7]Note that the structure of the data is different to the structure explained in the previous example. Data from a repeated prisoner's dilemma can also be used in the form displayed in Figure 3. The estimation function will assume that data is from a repeated prisoner's dilemma whenever this structure is used. Another possibility is to omit the variable `other_cooperation` and instead use a group identifier with column name `group` which identifies the pairwise matching of participants for a supergame. See Subsection 8.1 more information.

strategy which starts with cooperation and subsequently mimics the behavior of the other player in the previous period. TFT cooperates if the other player cooperated in the previous period and defects if the other player defected in the previous period. If a game continues for several periods, there are many different situations for which TFT prescribes cooperation. For example, in period three, if the action of the other player was defection in period one and cooperation in period two but also if the action of the other player was cooperation in both of the previous periods. Characterizing TFT by specifying the behavioral response in each possible situation is a daunting task if the game has many periods.
The TFT strategy can be characterized by an automaton with only two states $C$ and $D$, and two corresponding multinomial response vectors $\pi_C = \{\pi_C^c = 1, \pi_C^d = 0\}$ and $\pi_D = \{\pi_D^c = 0, \pi_D^d = 1\}$. The elements of the response vectors define the probability for action $a \in \{c, d\}$ conditional on the state $s \in \{C, D\}$ of the TFT automaton. In state $C$ the automaton representation of TFT prescribes to play $c$. In state $D$ it prescribes to play $d$. At the same time, the TFT automaton performs deterministic state transitions conditional on the current state of the automaton and some input. In the case of TFT, the input is the action profile of the previous period. Let $a_i a - j$ denote the action profile of the previous period where $a_i$ indicates the own action and $a_j$ the action of the other player. In state $C$, TFT remains in state $C$ if the input is $cc$ or $dc$ and changes to state $D$ if the input is $dc$ or $dd$. In state $D$, TFT remains in state $D$ if the input is $dc$ or $dd$ and changes to state $C$ if the input is $c$ or $dc$. The input value 0 is used to indicate the empty action profile for period one.

Figure 4: The Tit-For-Tat and Always-Defect automata



The two-state TFT automaton is depicted in the left panel of Figure 4. The two states are represented by the two nodes labeled $C$ and $D$. The deterministic state transitions of the TFT automaton are illustrated by the arrows leaving the nodes. The information next to an arrow indicates the inputs for which the transition takes place.
To give an example of an even simpler automaton, the right panel of Figure 4 depicts the DFA representation of the strategy which always defects (ALLD). ALLD is represented as a DFA with only one state $D$ with the response vector $\pi_D = \{\pi_D^c = 0, \pi_D^d = 1\}$. Since the automaton has only one state, the deterministic state transition function points from state $D$ to itself for every possible input. The fact that this transition occurs for every possible

input is indicated with an unlabeled arrow.

## 2.1 General definition

stratEst uses the following general definition of a deterministic finite-state automaton. For a game with inputs $\omega \in \Omega$ and discrete responses $r \in R$ the DFA $k \in K$ is a 4-tuple $(S_k, s_{k0}, \phi_k, \pi_k)$. $S_k$ is a finite set of strategy-specific states with elements $s$ and start state $s_{k0} \in S$. $\phi_k : S_k \times \omega \to S_k$ is a deterministic transition function which maps every possible combination of states and inputs into the set of states. $\pi_k$ is a collection of multinomial response vectors. The collection contains one vector for each state with $R$ elements and $\sum_{r=1}^{R} \pi_{ksr} = 1, \ \forall \ s \in S_k$ and $k \in K$. The DFA $(S_k, s_{k0}, \phi_k, \pi_k)$ prescribes a probabilistic response pattern for every situation of the game.
To give an example, if automaton $k$ is the TFT automaton: $S_k \in \{C, D\}$, $s_{k0} = C$, $\phi_k(s, c) = C$ and $\phi_k(s, d) = D, \ \forall \ s \in \{C, D\}$, $\pi_{ks} = \{\pi^c = 1, \pi^d = 0\}$ if $s = C$ and $\pi_{ks} = \{\pi^c = 0, \pi^d = 1\}$ if $s = D$. If automaton $k$ is ALLD: $S_k \in \{D\}$, $s_{k0} = D$, $\phi_k(D, \omega) = D \ \forall \ \omega \in \{c, d\}$, and $\pi_{ks} = \{\pi^c = 0, \pi^d = 1\}$ where $s = D$.

## 2.2 Matrix representation

A set of strategies each represented as DFA $(S_k, s_{k0}, \phi_k, \pi_k)$ can be used as a candidate set by the estimation function. The candidate set is submitted to the estimation function in the form of an R matrix. To continue with the examples, the numbers embraced by the brackets in Figure 5 define a candidate set which consists of the strategies TFT and ALLD.

Figure 5: Matrix representation of a candidate set

$$\begin{array}{cccccc} \text{state} & \pi_{ks}^{1} & \phi(s,1) & \phi(s,2) & \phi(s,3) & \phi(s,4) \\ \end{array}$$
$$\left[\begin{array}{cccccc} 1 & 1 & 1 & 2 & 1 & 2 \\ 2 & 0 & 1 & 2 & 1 & 2 \\ 1 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$$

The first two rows of the matrix define the TFT automaton. The third line defines the ALLD automaton. Each row of the matrix corresponds to one state of a strategy, starting with the initial state $s_{k0}$ of an automaton. The labels on top of the matrix illustrate the information in the columns and are not part of the matrix which used in the estimation function. The first column enumerates the states of strategy $k$. Hence, the number one in the first column always indicates the beginning of a new automaton with its start state. Column two contains the element of the multinomial response vectors which predicts the lowest non-zero output value in the data. In the example, this is action $c$, since we will use the value one to indicate cooperation and zero for defection in the data. If there are more output values in the data, more columns have to be added after the second column which contain the response probabilities of the strategies for these output values. The response probability for outputs with the value zero, are always omitted. Hence, the matrix representation of TFT is sufficient if defection is indicated with the value zero in the data. If defection is instead indicated with the value 2, we would need to include another column after the second column with probability values that row-wise sum up to one.

Columns three to six define the deterministic transitions between states. The numbers in column four indicate the next state if the input is one. In the example, the input is one if the strategy profile of the previous period was $cc$, and the TFT automation moves on to state $C$. The numbers in column four indicate the next state if the input is two. In the example, the input is two if the action profile of the previous period was $cd$, and the TFT automation moves on to state $D$. The interpretation of columns five and six is the same. No column exists for the input 0 as this input always points to the start state unconditional of the current state.

The system of rows and columns illustrated in the example can be used to define candidate sets for many other games. For example, imagine the goal is to analyze data of variant of the prisoner's dilemma with a third action $e$ labeled with the number two in the data. The matrix can be augmented by additional column which indicates the probability of response $e$ for every state. Action $e$ can also be used as an additional input which increases the number of possible inputs from five to ten ($3 \times 3$ action profiles plus input 0).

Generally speaking, the strategies matrix is a matrix where each row corresponds to one state of a strategy, starting with the start state $s_{k0}$ of an automaton. The first column enumerates the states of each strategy in ascending order. A value of one in the first column indicates the begin of a new strategy with its start state. The columns after the first column contain the collection of multinomial response vectors. The number of columns for the multinomial response vectors must correspond to the number of unique non-zero outputs in data. Without a reference output - which is labeled with a zero in the output column of data - the columns specify the complete multinomial response distribution for each unique value in the output column. In this case, the response probabilities in each row must sum to one. With a reference output, the response probability for the response labeled with zero is omitted and the response probabilities in each row must sum to a value smaller or equal to one. The remaining columns of the strategies matrix define the deterministic state transitions. The number of columns must equal the number of unique non-zero inputs in the data.

stratEst contains a set of 22 strategies which have been used to describe behavior in the indefinitely repeated prisoner's dilemma (Dal Bó and Fréchette, 2011; Fudenberg et al., 2012; Breitmoser, 2015). A documentation of the strategies can be found in the Appendix. The strategies documented in the Appendix can also serve as further examples.

## 3 Strategy Estimation

Strategy estimation was first used by Dal Bó and Fréchette (2011) to estimate the population shares of a candidate set of strategies based on a sample of experimental data from the indefinitely repeated prisoner's dilemma. Since the original publication, strategy estimation has been employed in several other studies on the repeated prisoner's dilemma (e.g. Aoyagi and Fréchette, 2009; Aoyagi et al., 2017; Arechar et al., 2017; Camera et al., 2012; Embrey et al., 2013; Fudenberg et al., 2012; Breitmoser, 2015).
Breitmoser (2015) extended the method by simultaneously estimating the relative frequency of strategies and some strategy-specific response parameters from experimental data. The possibility to estimate response parameters from data turns out to be useful when candidate strategies are ex-ante unknown or when some strategy parameters can't be pinned down based on theory (as it is the case for the semi-grim strategies reported in Breitmoser, 2015).
Dvorak and Fehrler (2018) extend strategy estimation in the spirit of latent class regression. In the latent class regression model, the prior probability to use a certain strategy is modeled as a function of individual characteristics which allows to asses the role of covariates for strategy use.
Section 3 proceeds by introducing the general model, the algorithm and the parameter estimates for strategy estimation in Subsections 3.1 - 3.3. The latent class regression model receives separate treatment in Section 5.

### 3.1 Model

This subsection presents the general model for strategy estimation. Consider a collection of categorical responses $r = \{1, \cdots, R\}$ of individuals $i = \{1, \cdots, N\}$ across situations $j = \{1, \cdots, J\}$ of the same game. Each situation is characterized by a unique history of past play leading to situation $j$. The DFA representation of strategy $k$ assigns an internal state $s_k = \{1, \cdots, S_k\}$ to each situation $j$. The state determines the response of the individual in the current situation. The subscript $k$ which indicates that states are strategy-specific will be omitted for better readability.
Let $y_{isr}$ denote the number of times response $r$ is observed in $n_{is}$ observations of individual $i$ when strategy $k$ would be in state $s$. The central assumption of the model is that the probability to observe vector $Y_{is} = \{y_{isr}, \cdots, y_{isR}\}$ follows $n_{is}$ independent draws from

a multinomial distribution with parameters $\pi_{ks} = \{\pi_{ks1}, \cdots, \pi_{ksR}\}$ where $\pi_{ksr} \in [0,1]$ and $\sum_{r=1}^{R} \pi_{ksr} = 1 \; \forall \; s \in S_k$ and $k \in K$. The assumption implies that the behavior of individuals is exclusively determined by the strategy they use. As a result, responses should be conditionally independent when controlling for the underlying strategies.

If $k$ is a pure strategy, the response probabilities $\pi_{ksr}$ are the result of pure underlying response probabilities confounded by trembling hand errors (Selten, 1975). Let $\xi_{ks}$ denote a vector of pure underlying response probabilities with elements $\xi_{ksr} \in \{0,1\}$ and $\gamma_{ks} \in [0,1]$ the probability of a tremble. The response probabilities $\pi_{ksr}$ follow from

$$\pi_{ksr} = \xi_{ksr}(1-\gamma_{ks}) + (1-\xi_{ksr})\frac{\gamma_{ks}}{R-1}. \tag{1}$$

Equation (1) describes a process in which a tremble uniformly implements one of the other responses after a realization has been obtained based on the vector $\xi_{ks}$. The tremble rules out that a single response which is not predicted by a pure strategy results in a likelihood of zero that the individual uses the strategy.

stratEst estimates the maximum-likelihood shares $p_k \in [0,1]$ with $\sum_{k=1}^{K} p_k = 1$ of individuals in the population which follow strategy $k$ with strategy parameters $\pi_{ksr}$ or $\xi_{ksr}$ and $\gamma_{ks}$ in the case of pure strategies. If it was known which individual follows which strategy, $p_k$ would immediately follow and the maximum-likelihood estimates of the strategy parameters could be easily obtained. However, the assignments of individuals to strategies are unknown latent variables which have to be estimated from the data. In the incomplete model, the strategy shares $p_k$ indicate the prior probability that individual $i$ uses strategy $k$. The observed likelihood of the incomplete model is:

$$L = \prod_{i=1}^{N} \sum_{k=1}^{K} p_k \prod_{s=1}^{S_k} \binom{n_{is}}{y_{is1}, \cdots, y_{isR}} \prod_{r=1}^{R} (\pi_{ksr})^{y_{isr}}$$

Since the multinomial coefficients are constant factors of the likelihood function, stratEst maximizes the log-likelihood function

$$\ln L = \sum_{i=1}^{N} \ln \left( \sum_{k=1}^{K} p_k \prod_{s=1}^{S_k} \prod_{r=1}^{R} (\pi_{ksr})^{y_{isr}} \right). \tag{2}$$

stratEst reports the parameters $p_k^*$, $\pi_{ksr}^*$ which maximize (2) under the parameter constraints $\pi_{ksr} \in [0,1]$ and $\sum_{r=1}^{R} \pi_{ksr} = 1$, and $p_k \in [0,1]$ and $\sum_{k=1}^{K} p_k = 1$.

## 3.2 Algorithm

stratEst uses the Expectation-Maximization algorithm (EM, Dempster et al., 1977) to obtain the maximum-likelihood estimates $p_k^*$, $\pi_{ksr}^*$ of the incomplete data problem outlined in (2). The algorithm exploits the fact that the ML estimates of the strategy parameters can be inferred if the assignments of individuals to strategies are known. At the same time, for known strategy parameters, the computation of the posterior probability estimates of the assignments of individuals to strategies is straightforward. After constrained random initialization of the model parameters, the EM algorithm iterates between the two steps until convergence of the log-likelihood defined in (2). In the expectation step, the posterior probability that individual $i$ uses strategy $k$ is calculated based on the current values of the parameters $p_k$ and $\pi_{ksr}$ according to:

$$\theta_{ik} = \frac{p_k \prod_{s=1}^{S_k} \prod_{r=1}^{R} (\pi_{ksr})^{y_{isr}}}{\sum_{k=1}^{K} p_k \prod_{s=1}^{S_k} \prod_{r=1}^{R} (\pi_{ksr})^{y_{isr}}}. \tag{3}$$

In the maximization step, the posterior probability assignments are used to update the population shares and strategy parameters. The population shares $p_k$ are updated to the expected values of the posterior probability assignments. The strategy parameters $\pi_{ks}$ are updated based on $K$ weighted data sets. To obtain the weighted data for strategy $k$, the responses $Y_{is}$ of individual $i$ are considered proportional to the posterior probability that $i$ uses $k$. The two steps are subsequently repeated until the log-likelihood converges.

Depending on the starting values used, the EM algorithm may converge to local optima. To avoid that local optima are returned by the estimation function, stratEst executes several runs of the EM algorithm from different starting points. stratEst uses the procedure proposed by Biernacki et al. (2003) to avoid local optima in an efficient way. During an outer run of the solver, several short inner runs of the EM algorithm are performed from different starting points and only the best solution obtained from the short runs is followed until convergence. stratEst reports the best solution found in several outer runs of the solver.

## 3.3 Parameter estimation

To find the maximum-likelihood estimates $p_k^*$, $\pi_{ksr}^*$, the stratEst solver starts by randomly initializing parameter values participant to the parameter constraints. In the expectation step of each iteration, the posterior probability that individual $i$ uses strategy $k$ is updated based on the current values of the model parameters according to (3). In the maximization step of each iteration, the model parameters are updated in order to maximize (2) conditional on the updated posterior probability assignments. For the strategy shares $p_k$, this requires optimization with respect to a sum-to-one constraint which is achieved based on the Lagrange multiplier function

$$\Lambda(p_k, \lambda) = \ln L + \lambda \left( \sum_{k=1}^{K} p_k - 1 \right).$$

Setting the partial derivatives $\partial\Lambda/\partial p_k$ and $\partial\Lambda/\partial\lambda$ to zero and solving for $p_k$ and $\lambda$ yields the conditions

$$p_k = -\sum_{i=1}^{N} \frac{\theta_{ik}}{\lambda} \quad \text{and} \quad \sum_{k=1}^{K} p_k = 1$$

which together yield $\lambda = -N$. Substitution into the first condition shows that the updated values of the shares follow from the updated posterior probability assignments calculated in the expectation step since

$$p_k^{new} = \frac{\sum_{i=1}^{N} \theta_{ik}}{N}. \tag{4}$$

If some shares $p_{k'}$ with $k' \in K$ have fixed values specified by the user, the remaining shares are updated according to (4) and subsequently rescaled by $1 - \sum_{k' \in K} p_{k'}$ to fulfill the sum-to-one constraint. The response probabilities $\pi_{ksr}$ are also updated participant to the sum-to-one constraint. Using Lagrange multipliers the updated values follow from

$$\pi_{ksr^{new}} = \sum_{i=1}^{N} \frac{\theta_{ik} y_{isr}}{\sum_{i=1}^{N} \theta_{ik} n_{is}}. \tag{5}$$

Again, if parameters $\pi_{ksr\prime}$ with $r' \in R$ are fixed, the remaining updated parameters are rescaled by $1 - \sum_{r' \in R} \pi_{ksr'}$ to fulfill the sum-to-one constraint.

stratEst uses the following procedure to update the pure underlying response parameters and the corresponding trembles. The pure response parameters are updated by transforming the updated values of the corresponding mixed parameters $\pi_{ksr}$ according to

$$\xi_{ksr}^{new} = \begin{cases} 1 & \text{if } \pi_{ksr}^{new} > \pi_{ksr'}^{new} \ \forall \ r' \neq r \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Equation 6 assigns density of one to the maximum of the updated response vector $\pi_{ks}^{new}$. This assures that the corresponding tremble parameters $\gamma_{ks}$ are as small as possible. If the maximum is not unique, the first parameter is set to one and all others values to zero. The updated values of the trembles can be found based on the substitution of (1) into (5). For the update of the tremble all response probabilities affected by the tremble are taken into account which yields

$$\gamma_{ks}^{new} = \sum_{i=1}^{N} \frac{\theta_{ik} \sum_{r=1}^{R} (y_{isr} - n_{is}\xi_{ksr}^{new}) \left(\frac{R-1}{1-R\cdot\xi_{ksr}^{new}}\right)}{\sum_{i=1}^{N} \theta_{ik} \cdot R \cdot n_{is}}. \tag{7}$$

Whenever parameters specified by the user are pure (i.e. zero or one), stratEst will automatically estimate a tremble parameter for these parameters. For mixed parameters, no tremble is estimated. Generally, for all response parameters which are estimated from the data, the restriction applies that all estimated parameters have to be of the same type either pure or mixed. Strategy parameters specified by the user are not affected by this restriction and can be pure or mixed independent of the type of the estimated parameters.
After all model parameters have been updated based on (4) and (5), the log-likelihood of the updated model is determined based on (2) and compared to the value from the last iteration. The algorithm continues with the next iteration as long as the log-likelihood increased in the current iteration.

## 4 Model Selection

The number of free parameters of a completely unspecified model with mixed responses equals $(K-1) + (R-1) \cdot \sum_{k=1}^{K} S_k$. Depending on the number of strategies, states and responses the number of free model parameters can be quite large. Four different approaches can be used to reduce the number of model parameters.

1. The conventional approach fixes parameters to specific values based on theoretical considerations. Fixed parameters are not estimated and reduce the number of free model parameters on a one to one basis.

2. Restrictions can be imposed on the strategy parameters. The restrictions imply that all model parameters of the same family ($\pi$, $\gamma$) which are affected by the restriction are reduced to a single vector of parameters. Three variants exist. Either each parameter of the same family is replaced by a single parameter vector for each strategy, for each state or overall.

3. The number of parameters of the same family can be selected based on information criteria. Three variants exist. The optimal number of parameter vectors of the same family is selected for each strategy, for each state or overall.

4. The number of strategies used to describe the data can be selected based on information criteria. For a candidate set strategies, nested models with fewer strategies are

estimated and the best model is selected.

The first approach is achieved by fixing the elements of inputs objects at specific values and illustrated in Section 8. The second, third , and fourth approaches are discussed in the following Subsections 4.1, 4.2, and 4.3.

## 4.1 Restrictions on strategy parameters

By default strategy parameters are assumed to be strategy-state specific, i.e. a response vector $\pi_{ks}$ is estimated for every state of every strategy in the case of mixed responses and a response vector $\xi_{ks}$ plus a tremble $\gamma_{ks}$ is estimated in the case of pure responses. One possibility to reduce the number of free parameters is to impose restrictions that some of the estimated strategy parameters from the same family are reduced to a single parameter vector. Restrictions can be imposed independently on the parameter vectors $\pi_{ks}$ and the trembles $\gamma_{ks}$. In both families of parameters, stratEst offers three variants of restrictions. Parameters can either be strategy-specific, state-specific or global. The first variant estimates a single parameter vector $\pi_k$ or a single parameter vector $\xi_k$ and a single tremble $\gamma_k$ per strategy which applies in all states of the strategy. The second variant estimates a single parameter vector $\pi_s$ or a single parameter vector $\xi_s$ and a single tremble $\gamma_s$ which applies in all states with the same number in the first column of the strategy matrix which enumerates the states. The third variant estimates a single parameter vector $\pi$ or a single parameter vector $\xi$ and a single tremble $\gamma$ which apply globally across all states and strategies. It is up to the user to decide if any of the restrictions can be justified based on theory. Please note that the second variant does not take into account whether states have the same deterministic state transitions across strategies.

If restrictions to the strategy parameters apply, the maximization step in the parameter estimation is adapted accordingly. Let $Z_t$ denote the set of all states $s_k$ of strategy $k$ where the corresponding strategy parameters are restricted to have the same underlying parameter vector $\zeta_t$, where $t \in \{1, \cdots, T\}$ is an index for the restrictions. The individual score contributions to $\zeta_t$ take all parameters affected by restriction $t$ into account, i.e.

$$\pi_{tr}^{new} = \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{s \in Z_t} \frac{\theta_{ik} y_{isr}}{\sum_{i=1}^{N} \sum_{s \in Z_t} \theta_{ik} n_{is_k}} \tag{8}$$

if $\zeta_t$ is a vector of response parameters. If the responses are pure, the updated response parameters are processed as described in Subsection and the trembles $\zeta_t$ are updated according to

$$\gamma_t^{new} = \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{s \in Z_t} \frac{\theta_{ik} \sum_{r=1}^{R} (y_{isr} - n_{is}\xi_{ksr}^{new}) \left( \frac{R-1}{1-R \cdot \xi_{ksr}^{new}} \right)}{\sum_{i=1}^{N} \sum_{s \in Z_t} \theta_{ik} \cdot R \cdot n_{is}}. \tag{9}$$

## 4.2 Selection of the number of strategy parameters

stratEst executes the following procedure to select the number of strategy parameters $\pi$, $\xi$ and $\gamma$ based on one of the three information criteria outlined in Subsection 4.4. First, an unrestricted model is estimated with a different parameter vectors for every strategy-state combination. A strategy-state-specific response vector $\pi_{ks}$ is estimated if the response is mixed, and a strategy-state-specific response vector $\xi_{ks}$ and a tremble $\gamma_{ks}$ are estimated if the response is pure. Next, a set of candidate parameter vectors of the same family is identified. The candidate sets of parameter vectors which are participant the selection can be restricted based on the three variants. The selection procedure for strategy parameters can be restricted to select the parameter vectors globally, for states across all strategies, or within strategies. For every pairwise combination of the elements of the set of candidate parameter vectors, a model is estimated where two candidate parameter vectors are reduced to a single parameter vector. The procedure continues as long as the fusion of any combination of two candidate parameter vectors improves the information criterion specified by the user.

## 4.3 Selection of the number of strategies

The number of strategies can be reduced based on a procedure introduced by Breitmoser (2015). The procedure starts by estimating the complete model with $K$ strategies. Next, $K$ nested models which result from deleting one strategy from the set of $K$ strategies are estimated. The $K$ nested models are ranked according to their log-likelihood. The procedure starts with the nested model with the lowest log-likelihood and compares the value of the information criterion with the value of the complete model with $K$ strategies. If eliminating the strategy is recommended based on the information criterion, the procedure is repeated starting with the reduced model with $K-1$ strategies. Otherwise, the nested model with the second lowest log-likelihood is considered. The strategy selection procedure stops if no strategy can be eliminated.

## 4.4 Information criteria

Three different penalized-likelihood criteria can be used to select the number of strategy parameters and strategies. The criteria are the Akaike Information Criterion (AIC, Akaike, 1973), the Bayesian Information Criterion (BIC, Schwarz, 1978), and the Integrated Classification Likelihood (ICL, Biernacki et al., 2000). The formulas for the three model selection criteria are

$$\text{AIC} = -\text{ln}L + df$$

$$\text{BIC} = -\text{ln}L + \frac{df}{2}log(N)$$

$$\text{ICL} = \text{BIC} - \sum_{i=1}^{N}\sum_{K=1}^{K}\theta_{ik}log(\theta_{ik}),$$

In all three formulas, *df* represents the number of free parameters of the model. Different assumptions and asymptotic approximations are needed to derive the formulas above. From the practitioner's point of view, the main difference between the three criteria is the size of the penalty for model complexity. ICL penalizes complexity more than BIC, and BIC more than AIC. In practice, all three model selection criteria will often deliver the same results. It is recommended to use AIC when it is more important to avoid underfitting of the data, and BIC when it is more important to avoid overfitting of the data. As $ICL$ includes an extra penalty for the entropy of the posterior probability assignments, it is recommended to use $ICL$ whenever precise predictions of individual strategy use are important.

## 5 Latent Class Regression

stratEst features latent class regression to assess the role of covariates for strategy use. Using the posterior probability assignments of individuals to strategies as dependent variables in a multinomial model leads to downward biased coefficients for the effects of covariates (Bolck et al., 2004). Latent class regression (Dayton and Macready, 1988; Bandeen-Roche et al., 1997) is a method which models the prior probability that individual $i$ uses strategy $k$ as a function of covariates. The simultaneous estimation of model parameters and coefficients generates unbiased estimates for the effect of the covariates.

### 5.1 The multinomial latent class model

stratEst uses the generalized multinomial logit link function to model the effects of covariates on the prior probability that individual $i$ uses strategy $k$ (see Agresti, 2003, for an introduction). The model takes the probability to use the first strategy as the benchmark. Then the the log-odds of using strategy $k$ compared to the first strategy are modeled as a linear function of covariates. stratEst will automatically add an intercept which will result in a covariate matrix $X$ with $N$ rows and $C$ columns if $C - 1$ variables are supplied. Let $X_i$

denote the the $i^{th}$ row of the covariate matrix $X$, then:

$$\ln(p_{ik}/p_{i1}) = X_i\beta_k \;\forall\; k \in K$$

where $p_{ik}$ represent the prior probability that individual $i$ uses strategy $k$ and $\beta_k$ is a column vector of coefficients with $C$ elements. Algebraic manipulations of the $K$ equations above yield

$$p_{ik} = \frac{e^{X_i\beta_k}}{\sum_{k=1}^{K} e^{X_i\beta_k}} \tag{10}$$

and the posterior probability that individual $i$ uses strategy $k$ is now

$$\theta_{ik} = \frac{p_{ik}\prod_{s=1}^{S_k}\prod_{r=1}^{R}(\pi_{ksr})^{y_{isr}}}{\sum_{k=1}^{K} p_{ik}\prod_{s=1}^{S_k}\prod_{r=1}^{R}(\pi_{ksr})^{y_{isr}}}. \tag{11}$$

The log-likelihood function of the latent class regression model is:

$$\ln L = \sum_{i=1}^{N} \ln\left(\sum_{k=1}^{K} p_{ik}\prod_{s=1}^{S_k}\prod_{r=1}^{R}(\pi_{ksr})^{y_{isr}}\right). \tag{12}$$

## 5.2 Parameter estimation

To goal of latent class regression is to identify the maximum-likelihood parameters $\beta^*$ and $\pi^*_{ksr}$ which maximize (12). This is achieved based a variant of the EM algorithm which uses a Newton-Raphson step to update the coefficients during the maximization step of the EM algorithm (Bandeen-Roche et al., 1997). After initialization, the expectation step consists of calculating the posterior probabilities $\theta_{ik}$ according to (11). In the maximization step, the column vector of coefficient $\beta$ is updated using

$$\beta^{new} = \beta - H_\beta^{-1}\nabla_\beta \tag{13}$$

where $\nabla_\beta$ is the score of the coefficient vector with elements:

$$\frac{\partial \ln L}{\partial \beta_{qk}} = x_{iq}(\theta_{ik} - p_{ik}) \tag{14}$$

and $H_\beta$ is the Hessian of (12) for the coefficients with elements

$$\frac{\partial^2 \ln L}{\partial \beta_{bl}\partial \beta_{ck}} = \sum_{i=1}^{N} x_{ib}x_{ic}(\theta_{il}(\delta_{lk} - \theta_{ik}) - p_{il}(\delta_{lk} - p_{ik})) \tag{15}$$

where $l, k \in \{1, \cdots, K\}$ and $b, c \in \{1, \cdots, C\}$ and $\delta_{lk} = 1$ if $l = k$ and $\delta_{lk} = 0$ otherwise. Note that in order to calculate $p_{ik}$, for individual $i$, the row vector $X_i$ must be complete and missing values are not allowed in the covariate matrix.

## 6 Standard Errors

By default analytic standard errors are reported for all estimated model parameters. The estimation strategy rests on the assumption that the individuals' strategy use is the result of independent realizations of the same stochastic process. This assumption might be violated for instance due to matching group or session effects in experiments. Unfortunately, using cluster robust standard errors is not a solution to this problem. Parameter estimates will be biased in cases where normal and cluster robust standard errors diverge due to the non-linearity of the model (see King and Roberts, 2015). One way to deal with such data is to use the same individual identifier for all observations which belong to the same cluster. The reported posterior probability assignments can then be interpreted as the shares of the strategies in each cluster. For the following calculations is assumed that the strategy use of the entities of interest $i \in \{1, \cdots, N\}$ follow independent draws from the prior distribution of strategies.

### 6.1 Analytic standard errors

stratEst estimates analytic standard errors based on the *empirical observed* information matrix (Meilijson, 1989) (see Linzer and Lewis, 2011, for an earlier application of the estimation procedure). The *empirical observed* information matrix is defined as

$$I_e(Y, \hat{\Psi}) = \sum_{i=1}^{N} s(Y_i, \hat{\Psi}) s^T(Y_i, \hat{\Psi}), \tag{16}$$

where $s(Y_i, \hat{\Psi})$ is the score contribution of individual $i$ with respect to parameter vector $\Psi$, evaluated at the ML estimate $\hat{\Psi}$. The reported standard errors are the square roots of the main diagonal of the inverse of $I_e(Y, \hat{\Psi})$. To calculate the standard error of the parameter $\eta_r$ with $\sum_{r=1}^{R} \eta_r = 1$, the score function $s(Y_i, \hat{\eta_r})$ is reparameterized in terms of log-ratios $\mu_r = ln(\eta_r/\eta_1)$ and the variance-covariance matrix VAR($\eta$) is calculated based on (16). The variance-covariance matrix VAR($\eta$) of the parameters is approximated using the delta method by

$$\text{VAR}(f(\hat{\mu})) = f'(\mu) I_e(Y, \hat{\mu})^{-1} f'(\mu)^T, \tag{17}$$

where $f'(\mu)$ is the Jacobian of the function $f(\mu_r) = \eta_r = e^{mu_r} / \sum_{r=1}^{R} \mu_r$ which converts the values back to the original units.

stratEst uses the following score vectors to calculate the empirical observed information matrix defined in (16). The shares are reparameterized in terms of log-rations as $p_k^* = ln(p_k/p_1)$ and the score contribution $\partial \text{ln} L / \partial p_k^*$ of individual $i$ is

$$s(Y_i, p_k^*) = \theta_{ik} - p_k. \tag{18}$$

Let $f(p_k^*) = p_k = e^{p_k^*} / \sum_{l=1}^{K} e^{p_l^*}$ denote the inverse of the reparameterization, then the Jacobian $f'(p^*)$ has elements

$$\frac{\partial f(p_k^*)}{\partial p_l^*} = \begin{cases} -p_k p_l & \text{if } l \neq k \\ p_k(1-p_l) & \text{if } l = k \end{cases} \tag{19}$$

and the variance-covariance matrix of the shares is approximated by (17) using the inverse of (16) based on the score contributions defined in (18).

If $\pi_{ksr}$ are mixed parameters standard errors are calculated based on the reparameterization $\pi_{ksr}^* = ln(\pi_{ksr}/\pi_{ks1})$ and the score contribution $\partial \text{ln} L / \partial \pi_{ksr}^*$ of individual $i$ is

$$s(Y_i, \pi_{ksr}^*) = \theta_{ik} \left( y_{isr} - n_{is} \pi_{ksr} \right). \tag{20}$$

Let $g(\pi_{ksr}^*) = \pi_{ksr} = e^{\pi_{ksr}^*} / \sum_{r=1}^{R} \pi_{ksr}^*$ denote the inverse of the reparameterization, then the Jacobian $g'(\pi^*)$ has elements

$$\frac{\partial g(\pi_{ksr}^*)}{\partial \pi_{ltq}^*} = \begin{cases} -\pi_{ksr}\pi_{ltq} & \text{if } k = l \text{ and } s = t \text{ and } r \neq q \\ \pi_{ksr}(1-\pi_{ltq}) & \text{if } k = l \text{ and } s = t \text{ and } r = q \\ 0 & otherwise \end{cases} \tag{21}$$

and the variance-covariance matrix of the shares is approximated by (17) using the inverse of (16) based on the score contributions defined in (20).

No reparameterization is needed to obtain the standard errors of the tremble parameters since for any value of the tremble $\gamma_{ks}$ the sum-to-one constraint is always fulfilled for all affected response probabilities. The score contribution $\partial \text{ln} L / \partial \gamma_{ks}$ of individual $i$ is

$$s(Y_i, \gamma_{ks}^*) = \theta_{ik} \sum_{r=1}^{R} \frac{y_{isr}}{\pi_{ksr}} \left( \frac{1 - \xi_{ksr}}{R-1} - \xi_{ksr} \right) \tag{22}$$

and the variance-covariance matrix is approximated by the inverse of (16) using the score of

the coefficients outlined in (22).
The reported standard errors of the latent class regression coefficients $\beta_{ck}$ are the square roots of the main diagonal of (16) using the score of the coefficients outlined in (14).

## 6.2 Analytic standard errors with parameter restrictions

If restrictions apply, the score vectors change as before where the score contribution of individual $i$ is the summation over all states $s_k \in Z_t$ where parameters are affected by restriction $t \in \{1, \cdots, T\}$. For mixed response probabilities, the score contribution with respect to the underlying common parameter $\partial \ln L / \partial \pi^*_{tr}$ of individual $i$ is

$$s(Y_i, \pi^*_{tr}) = \sum_{k=1}^{K} \theta_{ik} \sum_{s_k \in Z_t} (y_{isr} - n_{is}\pi_{ksr}) \, . \tag{23}$$

and the Jacobian $g'(\pi^*)$ has elements

$$\frac{\partial g(\pi^*_{tr})}{\partial \pi^*_{uq}} = \begin{cases} -\pi_{tr}\pi_{uq} & \text{if } t = u \text{ and } r \neq q \\ \pi_t(1-\pi_u) & \text{if } t = u \text{ and } r = q \\ 0 & \textit{otherwise.} \end{cases} \tag{24}$$

With restrictions, the score contribution of a tremble parameter $\partial \ln L / \partial \gamma_t$ of individual $i$ is

$$s(Y_i, \gamma_t) = \sum_{k=1}^{K} \sum_{s_k \in Z_t} \theta_{ik} \sum_{r=1}^{R} \frac{y_{isr}}{\pi_{ksr}} \left( \frac{1 - \xi_{ksr}}{R-1} - \xi_{ksr} \right) \tag{25}$$

All other model parameters are not affected by the restrictions and calculated as outlined before.

## 6.3 Bootstrapped standard errors

stratEst obtains bootstrapped standard errors for the population shares and strategy parameters by resampling individuals with replacement. In each bootstrap sample $m \in M$, parameter estimates are obtained based on the observations of $N$ individuals sampled in iteration $m$. Estimates for the strategy parameters are generated by fixing the value of all remaining strategy parameters of the model at the original maximum-likelihood estimate for these parameters. Fixation of the other model parameters is crucial to maintain the original structure of the model across the bootstrap estimates.
When performing latent class regression, analytic standard errors are reported for the aver-

age priors $p_k$ and the coefficient vector $\beta$. The reason is that it is likely to produce samples which suffer from quasi-complete separation during the bootstrap procedure. Under quasi-complete separation, ML-estimate for the latent class regression coefficients may not exist in some bootstrap replications. In these replications, estimates of the latent class regression coefficients take very large values and bias the bootstrapped standard errors for these estimates.

## 7 Simulation

This section illustrates the validity of the estimation procedures based on simulated data. $M = 1000$ samples are generated. Each sample $m \in \{1, \cdots, M\}$ consists of the binary responses of $N = 200$ individuals following one of the following two strategies:

$$s_1 = \begin{bmatrix} 1 & 1-\gamma_m & 1 & 2 \\ 2 & \gamma_m & 1 & 2 \end{bmatrix} ; \; s_2 = \begin{bmatrix} 1 & \pi_m & 1 & 2 \\ 2 & \pi_m & 1 & 2 \end{bmatrix}$$

The parameters $\pi_m$ and $\gamma_m$ are independent draws from a normal distribution with mean 0.25 and variance 0.1. The assignment of individuals to strategies in sample $m$ is influenced by covariate vector $x_i$ which contains an intercept and a dummy variable for individual $i$. The dummy variable is one for half of the individuals and zero for the other half. The probability of using $s_i$ for individual $i$ is given by:

$$Pr(s = s_1|x_i) = \frac{1}{1 + e^{x_i \beta_m}}$$

The elements of the coefficient vector $\beta_m$ are the intercept $\beta_{0m}$ and the coefficient of the dummy variable $\beta_{1m}$. For each sample $m$, the coefficients $\beta_{0m}$ and $\beta_{1m}$ are independent draws from a normal distribution with mean zero and variance of one. Individuals are randomly assigned to $s_1$ with $Pr(s_i = s_1|x_i)$. In every sample $m$, 20 binary responses are generated for each individual in each of the two states. The binary responses of an individual are independent realizations of a Bernoulli process with success probability equal to the state specific response probability of the strategy the individual has been assigned.
The following matrix is used to submit the strategies to the solver:

$$\texttt{strategies =} \begin{bmatrix} 1 & 1 & 1 & 2 \\ 2 & 0 & 1 & 2 \\ 1 & \text{NA} & 1 & 2 \\ 2 & \text{NA} & 1 & 2 \end{bmatrix} \tag{26}$$

The first two rows of the matrix correspond to $s_1$ and the second and third row to $s_2$. stratEst will automatically estimate tremble parameters for both states of the first strategy if the submitted response probabilities are one and zero. Setting the response parameters of the second strategy to NA implies that these parameters should be estimated from the data. To estimate the correct model specification, the estimation is restricted to only one response parameter $\pi$ and one tremble parameter $\gamma$ probability. To include covariates in the estimation, a matrix *covar* is generated with one column and as many rows as data. Each row of *covar* which contains an observation of individual $i$ is set to the dummy variable of individual $i$. For each sample $m$, the following syntax is used to estimate the correct model:

```
stratEst(data,strategies,covar,r.responses ="global",r.trembles ="global")
```

Omitting the restrictions `r.responses ="global"` and `r.trembles ="global"` will estimate two response parameters - one for each occurrence of `NA` in strategies - and two tremble parameters - one for each occurrence of one or zero in strategies - respectively.

Table 1: Simulation exercise with two strategies

| | | | | coverage probability | | selection |
|---|---|---|---|---|---|---|
| $\theta_m$ | $E(\hat{\theta}_m)$ | $E(\theta_m - \hat{\theta}_m)$ | $E(\lvert\theta_m - \hat{\theta}_m\rvert)$ | analytic | bootstrap | correct |
| $p_1$ | 0.498 | 0.000 | 0.026 | 0.957 | 0.956 | 0.999 |
| $p_2$ | 0.502 | -0.000 | 0.026 | 0.957 | 0.956 | - |
| $\pi$ | 0.250 | 0.000 | 0.006 | 0.938 | 0.937 | 0.198 |
| $\gamma$ | 0.252 | -0.000 | 0.006 | 0.946 | 0.944 | 0.384 |
| $\beta_0$ | -0.023 | 0.007 | 0.194 | 0.956 | 0.959 | - |
| $\beta_1$ | 0.078 | -0.010 | 0.366 | 0.935 | 0.947 | - |

*Notes:* The Table depicts the results of $M = 1000$ Monte Carlo samples of data consisting of the responses of 200 individuals observed 20 times in each of two states. $\theta_m$ represents the true parameter in sample $m$ and $\hat{\theta}_m$ the corresponding estimate based on sample $m$. Bootstrapped standard errors based on 1000 samples.

Table 1 summarizes the results of the simulation exercise. Columns two to four depict the averages of the estimated parameters, the difference between the estimated to the true parameters, and the absolute difference between the estimated and the true parameters over the 1000 samples. The averages of the parameter estimates in the first column are close to the actual means of the distributions where the parameters are drawn from. The averages of the differences between the estimated and the drawn parameters in the second column show no systematic biases of the estimates. The averages of the absolute differences between the estimated and the true parameters are small. Note that the absolute differences are larger for the latent class regression coefficients due to the different scale of these parameters.

Columns five and six of Table 1 report the coverage probabilities of analytic and bootstrapped of 95% confidence intervals for the model parameters. The coverage probabilities are close to the expected probability of 0.95. For the coverage probabilities displayed in rows one to four, the 95 percent confidence interval based on 1000 Monte Carlo samples can be calculated which spans from 0.936 to 0.964.

Next, the simulated data is used to illustrate the selection of the number of strategies, response probabilities, and trembles. Two additional strategies are added to the strategy matrix as red herrings which are similar to the two true underlying strategies. The augmented candidate set of four strategies offers several different possibilities to over-fit the data. To include the two additional strategies, `strategies` is augmented by the following rows:

$$\begin{bmatrix} 1 & 1 & 1 & 2 \\ 2 & \text{NA} & 1 & 2 \\ 1 & \text{NA} & 1 & 2 \\ 2 & 0 & 1 & 2 \end{bmatrix} \tag{27}$$

In each of the $M$ samples, a selection of the number of strategies, responses and trembles based on the ICL criterion is executed with the command:

```
stratEst(data,strategies,covar,select ="all",crit = "ICL")
```

The last column of Table 1 depicts the results of the parameter selection. The numbers displayed in the last column indicate the frequency of selecting the correct number of shares, response probabilities and trembles across the $M$ samples. The first row indicates that the correct number of strategies selected in 999 out of 1000 samples. Rows three and four show that the probability to select the correct number of response parameters and trembles is substantially lower. This indicates that the selection of the number of responses and trembles frequently produces results generally over-estimate the number of responses and trembles.

## 8 Using the Package

The central estimation function of the package is the function `stratEst()`. The function expects input objects in the following order:

```
stratEst(data,strategies,shares,covariates,response,r.responses,r.trembles,
            r.select,crit,se,outer.runs,outer.tol,outer.max,inner.runs,
            inner.tol,inner.max,lcr.runs,lcr.tol,lcr.max)
```

Subsections 8.1 and 8.2 explain the input and the output objects of the function.

## 8.1 Input objects

`data`: Mandatory input object which contains the data for the estimation in the long format. Each row in data represents one observation of one individual. The object data must be an R data frame object with variables in columns. Three columns are mandatory: A column named id which identifies the observations of the same individual across the rows of the data frame. A column named input which indicates the type of information observed by the individual before giving a response. A column named output which contains the behavioral response of the individual after observing the input. If an individual plays the same game for more than one period with the same partner, `data` must contain a variable period which identifies the period within the game. If an individual plays the same game more than once with different partners, `data` must contain a variable game (or `supergame`) which identifies data from different games. For data from prisoner's dilemma experiments, two more data formats are possible. Instead of using the variables input and output, the data frame may also contain the variables cooperation and other_cooperation, or alternatively, the variables cooperation and group. The variable cooperation should be a dummy which indicates if the participant cooperated in the current period. The variable other_cooperation should be a dummy which indicates if the other player cooperated in the current period. The variable group should be an identifier variable with a unique value for each unique match of two individuals.

`strategies`: Mandatory input object. Can be either a positive integer or a matrix. If an integer is used, the estimation function will generate the respective number of memory-one strategies with as many states as there are unique input values in data. A matrix can be used to supply a set of customized strategies. In the matrix, each row corresponds to one state of a strategy, starting with the start state of an automaton. The first column enumerates the states of each strategy in ascending order. A value of one in the first column indicates the begin of a new strategy with its start state. The columns after the first column contain the collection of multinomial response vectors. The number of columns for the multinomial response vectors must correspond to the number of unique non-zero outputs in data. Without a reference output - which is labeled with a zero in the output column of data - the columns specify the complete multinomial response distribution for each unique value in the output column. In this case, the response probabilities in each row must sum to one. With a reference output, the response probability for the response labeled with zero is omitted and the response probabilities in each row must sum to a value smaller or equal to one. The remaining columns of the strategies matrix define the deterministic state transitions.

The number of columns must equal the number of unique non-zero inputs in the data. The numbers in the first column indicate the next state of the automaton if the input is one. The numbers in the second column indicate the next state if the input is two and so on.

`shares`: A column vector of strategy shares. The number of elements must correspond to the number of strategies defined in the strategies matrix. Elements which are NA are estimated from the data. If the object is not supplied, stratEst estimates a share for every strategy defined in the strategies matrix.

`covariates`: A matrix where each row corresponds to same row in data. Hence, the covariate matrix must have as many rows as the data matrix. Observations which have the same ID in data must also have the same vector of covariates. Missing value are not allowed. If covariates are supplied, stratEst estimates the latent class regression model introduced in Section 5.

`response`: String which can be set to *pure* or *mixed.* If set to *pure* all response probabilities estimated from the data are pure responses. If set to *mixed* all response probabilities estimated from the data are mixed responses. The default is *mixed.*

`r.responses`: A string which can be set to *no*, *strategies*, *states* or *global.* If set to *strategies*, the estimation function estimates strategies with one strategy specific vector of responses in every state of the strategy. If set to *states*, one state specific vector of responses is estimated for each state. If set to *global*, a single vector of responses is estimated which applies in every state of each strategy. Default is *no.*

`r.trembles`: String which can be set to *no*, *strategies*, *states* or *global.* If set to *strategies*, the estimation unction estimates strategies with one strategy specific tremble probability. If set to *states*, one state specific tremble probability is estimated for each state. If set to *global*, a single tremble is estimated which applies in every state of each strategy. Default is *no.*

`select`: String which can be set to *no*, *strategies*, *responses*, *trembles*, *both*, and *all.* If set to *strategies*, *responses*, *trembles*, the number of strategies, responses, trembles respectively are selected based on the selection criterion specified in option *crit.* If set to *both*, the number of responses and trembles are selected. If set to *all*, the number of strategies, responses, and trembles are selected. Default is *no.*

`crit`: String which can be set to *BIC*, *AIC* or *ICL.* If set to *BIC*, model selection based on the Bayesian Information criterion is performed. If set to AIC, the Akaike Information criterion is used. If set to *ICL* the Integrated Classification Likelihood criterion is used. Default is *BIC.*

`se`: String which can be set to *no*, *yes* or *bs.* If set to *no*, standard errors are not reported. If set to *yes*, analytic standard errors are reported. If set to *bs*, bootstrapped standard errors are reported for responses and trembles. Default is *yes*.

`outer.runs`: Positive integer which stets the number of outer runs of the solver. Default is 10.

`outer.tol`: Positive number which stets the tolerance of the continuation condition of the outer runs. The iterative algorithm stops after iteration $j$ if $1 - LL_j/LL_{j-1} <$ outer.tol. Default is 0.

`outer.max`: Positive integer which stets the maximum number of iterations of the outer runs of the solver. The iterative algorithm stops after iteration $j$ if $j =$ outer.max. Default is 1000.

`inner.runs`: Positive integer which stets the number of inner runs of the solver. Default is 100.

`inner.tol`: Positive number which stets the tolerance of the continuation condition of the inner EM runs. The iterative algorithm stops after iteration $j$ if $1 - LL_j/LL_{j-1} <$ inner.tol. Default is 0.

`inner.max`: Positive integer which stets the maximum number of iterations of the inner EM runs. The iterative algorithm stops after iteration $j$ if $j =$ inner.max. Default is 100.

`lcr.runs`: Positive integer which stets the number of estimation runs for latent class regression. Default is 100.

`lcr.tol`: Positive number which stets the tolerance of the continuation condition of the latent class regression runs. The iterative algorithm stops after iteration $j$ if $1 - LL_j/LL_{j-1} <$ LCR.tol. Default is 0.

`lcr.max`: Positive integer which stets the maximum number of iterations of the latent class regression EM runs. The iterative algorithm stops after iteration $j$ if $j =$ inner.max. Default is 1000.

## 8.2 Output objects

`shares`: Column vector which contains the estimates of population shares for the strategies. The first element corresponds to the first strategy defined in the strategy matrix, the second element to corresponds to the second strategy and to on. Can be used as input object of the estimation function.

`strategies`: Matrix which contains the strategies of the model. Can be used as input object of the of the estimation function.

`responses`: Column vector which contains the response probabilities of the strategies. The value -1 indicates that the corresponding response could not be estimated since data does not contain observations the model assigns to the corresponding state.

`trembles:` Column vector which contains the tremble probabilities of the strategies. The value -1 indicates that the corresponding response could not be estimated since data does not contain observations the model assigns to the corresponding state.

`coefficients:` Column vector which contains the latent class regression coefficients for strategies 2 to $k$.

`response.mat:` Matrix which contains the estimates of the response probabilities for the columns of the strategy matrix which represent the response probabilities.

`tremble.mat:` Matrix which contains the estimates of the tremble probabilities for the columns of the strategy matrix which represent the response probabilities.

`coefficient.mat:` Matrix which contains the latent class regression coefficients of strategies 2 to $K$ in columns.

`ll.val:` The log-Likelihood value corresponding to the reported estimates. Bigger values indicate a better fit of the model to the data.

`crit.val:` The value of the selection criterion defined with option *crit*. Bigger values indicate a better fit of the model.

`eval:` Number of iterations of the solver. The reported number is the sum of iterations performed in the inner and the outer run which led to the reported estimates.

`tol.val:` The tolerance value in the last iteration.

`assignments:` Matrix which contains the posterior probability assignments $\theta_{ik}$ of individuals to strategies. The matrix has $N$ rows which correspond to the ID sorted in ascending order beginning with the individual with the lowest ID. The matrix has $K$ columns which correspond to the strategies, starting with the first strategy defined in the strategy matrix in column one.

`priors:` Matrix which contains the individual prior probabilities $p_{ik}$ of individuals as predicted by the covariate vectors of the individuals. The matrix has $N$ rows which correspond to the ID sorted in ascending order beginning with the individual with the lowest ID. The matrix has $K$ columns which correspond to the strategies, starting with the first strategy defined in the strategy matrix.

`shares.se:` Column vector which contains the standard errors of the estimated shares. The elements correspond to the vector of estimates.

`responses.se:` Column vector which contains the standard errors of the reported responses. The elements correspond to the vector of estimates.

`trembles.se:` Column vector which contains the standard errors of the reported trembles.

The elements correspond to the vector of estimates.

`coefficients.se`: Column vector which contains the standard errors of the reported coefficients. The elements correspond to the vector of estimates.

`convergence`: Row vector which reports the maximum value of the score vector of the shares as the first element, responses as the second element, trembles as the third element, and LCR coefficients as the forth element. Small values indicate convergence of the algorithm to a (local) maximum.

## References

Agresti, A. (2003): *Logit Models for Multinomial Responses*, Wiley-Blackwell, chap. 7, 267–313.

Akaike, H. (1973): *Second International Symposium on Information Theory*, Budapest, Hungary: Akademiai Kiado, chap. Information Theory and an Extension of the Maximum Likelihood Principle, 267–281.

Aoyagi, M., V. Bhaskar, and G. R. Fréchette (2017): "The Impact of Monitoring in Infinitely Repeated Games: Perfect, Public, and Private," *Mimeo.*

Aoyagi, M. and G. R. Fréchette (2009): "Collusion as public monitoring becomes noisy: Experimental evidence," *Journal of Economic Theory*, 144, 1135–1165.

Arechar, A. A., A. Dreber, D. Fudenberg, and D. G. Rand (2017): "I'm just a soul whose intentions are good?: The role of communication in noisy repeated games," *Games and Economic Behavior*, 104, 726–743.

Bandeen-Roche, K., D. L. Miglioretti, S. L. Zeger, and P. J. Rathouz (1997): "Latent Variable Regression for Multiple Discrete Outcomes," *Journal of the American Statistical Association*, 92, 1375–1386.

Beath, K. (2011): "randomLCA: Random Effects Latent Class Analysis," Tech. rep., R package version 0.7, URL http://CRAN.R-project.org/package=randomLCA.

Bengtsson, H. (2018): *R.rsp: Dynamic Generation of Scientific Reports*, r package version 0.43.0.

Biernacki, C., G. Celeux, and G. Govaert (2000): "Assessing a mixture model for clustering with the integrated completed likelihood," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 719–725.

——— (2003): "Choosing starting values for the {EM} algorithm for getting the highest likelihood in multivariate Gaussian mixture models," *Computational Statistics & Data Analysis*, 41, 561–575.

Bolck, A., M. Croon, and J. Hagenaars (2004): "Estimating Latent Structure Models with Categorical Variables: One-Step Versus Three-Step Estimators," *Political Analysis*, 12, 3–27.

Breitmoser, Y. (2015): "Cooperation, but no reciprocity: Individual strategies in the repeated prisoner's dilemma," *American Economic Review*, 105, 2882–2910.

Camera, G., M. Casari, and M. Bigoni (2012): "Cooperative strategies in anonymous economies: An experiment," *Games and Economic Behavior*, 75, 570–586.

Dal Bó, P. and G. R. Fréchette (2011): "The evolution of cooperation in infinitely repeated games: Experimental evidence," *American Economic Review*, 101, 411–429.

Dayton, C. M. and G. B. Macready (1988): "Concomitant-Variable Latent-Class Models," *Journal of the American Statistical Association*, 83, 173–178.

Dempster, A., N. Laird, and D. B. Rubin (1977): "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society Series B*, 39, 1–38.

Dvorak, F. and S. Fehrler (2018): "Negotiating Cooperation Under Uncertainty: Communication in Noisy, Indefinitely Repeated Interactions," Tech. rep., IZA Discussion Paper No. 11897.

Eddelbuettel, D. and R. François (2011): "Rcpp: Seamless R and C++ Integration," *Journal of Statistical Software*, 40, 1–18.

Embrey, M., G. Fréchette, and E. Stacchetti (2013): "An Experimental Study of Imperfect Public Monitoring: Efficiency Versus Renegotiation-Proofness," *Mimeo.*

Fudenberg, D., D. G. Rand, and A. Dreber (2012): "Slow to Anger and Fast to Forgive: Cooperation in an Uncertain World," *American Economic Review*, 102, 720–749.

Kaufman, L. and P. J. Rousseeuw (1990): *Finding groups in data. an introduction to cluster analysis*, Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics, New York: John Wiley & Sons, Inc.

King, G. and M. E. Roberts (2015): "How Robust Standard Errors Expose Methodological Problems They Do Not Fix, and What to Do About It," *Political Analysis*, 23, 159–179.

Lazarsfeld, P. F. (1950): *Measurement and Prediction*, New York: John Wiley & Sons, Inc., chap. The Logical and Mathematical Foundations of Latent Structure Analysis, 362–412.

Leisch, F. (2004): "FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R," *Journal of Statistical Software*, 11.

Linzer, D. A. and J. B. Lewis (2011): "poLCA: An R Package for Polytomous Variable Latent Class Analysis," *Journal of Statistical Software*, 42.

McLachlan, G. and D. Peel (2005): *Finite Mixture Models*, New York: John Wiley & Sons, Inc.

Meilijson, I. (1989): "A Fast Improvement to the EM Algorithm on its Own Terms," 51, 127–138.

R Development Core Team (2008): "R: A language and environment for statistical computing," Tech. rep., R Foundation for Statistical Computing.

Sanderson, C. and R. Curtin (2016): "Armadillo: a template-based C++ library for linear algebra." *Journal of Open Source Software*, 1, 26.

Schwarz, G. (1978): "Estimating the Dimension of a Model," *Ann. Statist.*, 6, 461–464.

Selten, R. (1975): "Reexamination of the perfectness concept for equilibrium points in extensive games," *International Journal of Game Theory*, 4, 25–55.

Wickham, H. (2011): "testthat: Get Started with Testing," *The R Journal*, 3, 5–10.

Wickham, H., P. Danenberg, and M. Eugster (2018a): *roxygen2: In-Line Documentation for R*, r package version 6.1.1.

Wickham, H., J. Hester, and W. Chang (2018b): *devtools: Tools to Make Developing R Packages Easier*, r package version 2.0.1.

Xie, Y. (2018): "knitr: A General-Purpose Package for Dynamic Report Generation in R," Tech. rep., R package version 1.20.

# Appendix

Tables A.1, A.2, and A.3 depict 22 strategies for the repeated prissonne's dilemma which can be used after the package is loaded and attached to the search path with the command `library(stratEst)`. Strategies 1-20 and their descriptions are taken from Fudenberg et al. (2012). Strategy 21 is the semi-grim structure discovered by Breitmoser (2015). In the automata representations in column three, circles represent strategy states and arrows transitions between strategy states. The start state $s_{k0}$ of the strategies is always the first circle from the left. Capital letters in the circles indicates the action the automata prescribes in the state. Variables indicate an ex-ante unspecified probability to cooperate. Transitions between states occur conditional on the action profile of the current period. Letters next to transition arrows indicate that the transition occurs conditional on observing this profile. The first letter of action profiles indicates the own action and the second letter the action of the other player in the current period. To give an example, if the action profile next to the arrow is *cd*, the transition arrow is applies if the own action i *c* and the action of the other player i *d* in the current period. If no action profile is depicted next to an arrow, the transition arrow applies unconditionally, for all possible action profiles which can be observed.

The strategies depicted in Tables A.1, A.2, and A.3, can be used for data which has the following format: The output in column five must be 1 if the action of the player was cooperation and 0 if the action was defection. Zeros have to be used as inputs in period one. In all other periods the values 1, 2, 3 and 4 are used to indicate the strategy profiles *cc*, *cd*, *dc* and *dd* of the current period respectively. The matrix representation of strategies follows Section 2. Each row represents one state of a strategy and the first column indicates the state number. The second column the probability to play $C$ in every state. Columns 3-6 indicate the deterministic state transitions after observing the action profiles *cc*, *cd*, *dc*, and *dd* respectively.

Table A.1: Pre-programmed prisoner's dilemma strategies (1-10)

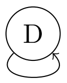
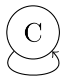
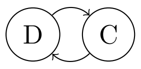
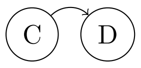
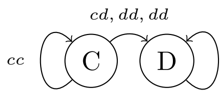
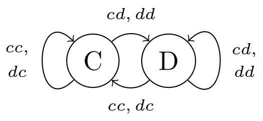
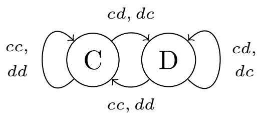
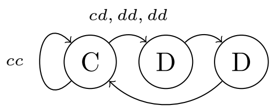
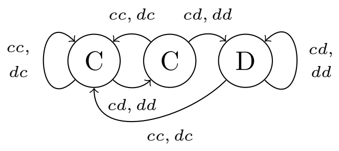
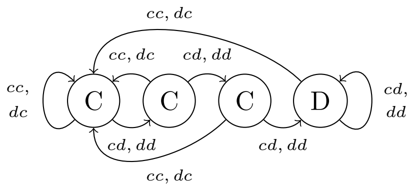
| Acronym | Description | Automaton | Matrix $s$ | $\pi^c$ | $cc$ | $cd$ | $dc$ | $dd$ |
|---|---|---|---|---|---|---|---|---|
| ALLD | Always play D |  | 1 | 0 | 1 | 1 | 1 | 1 |
| ALLC | Always play C |  | 1 | 1 | 1 | 1 | 1 | 1 |
| DC | Start with D, then alternate between C and D |  | 1<br>2 | 0<br>1 | 2<br>1 | 2<br>1 | 2<br>1 | 2<br>1 |
| FC | Play C in the first period, then D forever |  | 1<br>2 | 1<br>0 | 2<br>2 | 2<br>2 | 2<br>2 | 2<br>2 |
| Grim | Play C until either player plays D, then play D forever |  | 1<br>2 | 1<br>0 | 1<br>2 | 2<br>2 | 2<br>2 | 2<br>2 |
| TFT | Play C unless partner played D in previous period |  | 1<br>2 | 1<br>0 | 1<br>1 | 2<br>2 | 1<br>1 | 2<br>2 |
| PTFT (WSLS) | Play C if both players chose the same move in the previous period, otherwise play D |  | 1<br>2 | 1<br>0 | 1<br>1 | 2<br>2 | 2<br>2 | 1<br>1 |
| T2 | Play C until either player plays D, then play D twice and return to C (regardless of all actions during the punishment periods) |  | 1<br>2<br>3 | 1<br>0<br>0 | 1<br>3<br>1 | 2<br>3<br>1 | 2<br>3<br>1 | 2<br>3<br>1 |
| TF2T | Play C unless partner played D in both of the last 2 periods |  | 1<br>2<br>3 | 1<br>1<br>0 | 1<br>1<br>1 | 2<br>3<br>3 | 1<br>1<br>1 | 2<br>3<br>3 |
| TF3T | Play C unless partner played D in all of the last 3 periods |   cc, dc; cc, dc; cd, dd; cc, dc; C; C; C; D; cd, dd; cd, dd; cd, dd; cc, dc | 1<br>2<br>3<br>4 | 1<br>1<br>1<br>0 | 1<br>1<br>1<br>1 | 2<br>3<br>4<br>4 | 1<br>1<br>1<br>1 | 2<br>3<br>4<br>4 |

Table A.2: Pre-programmed prisoner's dilemma strategies (11-18)

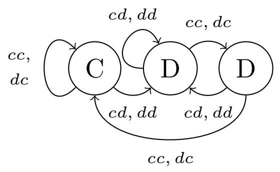
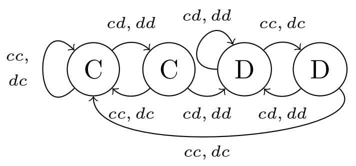
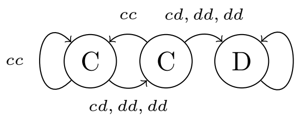
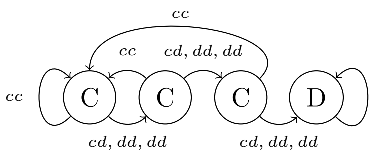
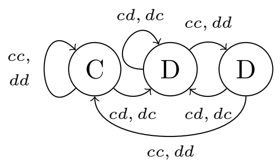
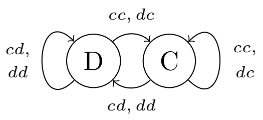
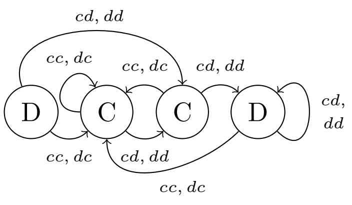
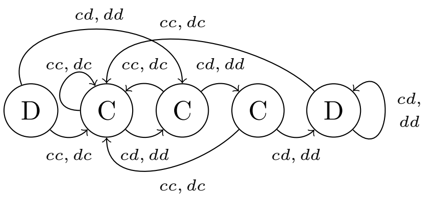
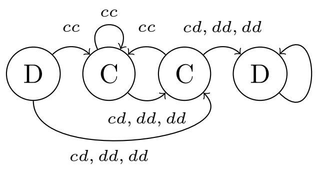
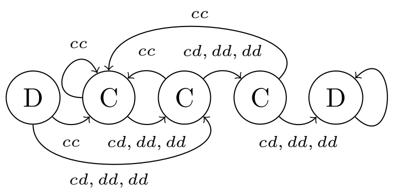
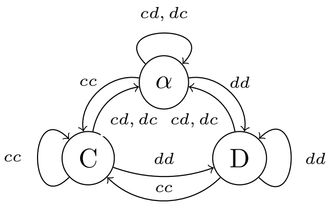
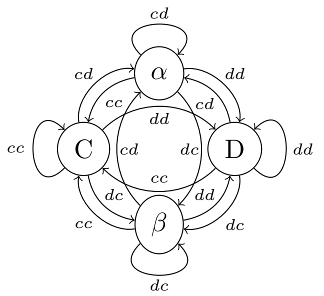
| Acronym | Description | Automaton | Matrix | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | $s$ | $\pi^c$ | $cc$ | $cd$ | $dc$ | $dd$ |
| T2FT | Play C unless partner played D in either of the last 2 periods (2 periods of punishment if partner plays D) |  $cd, dd$<br>$cc, dc$<br>$cc,$ $dc$<br>C<br>D<br>D<br>$cd, dd$<br>$cd, dd$<br>$cc, dc$ | 1 | 1 | 1 | 2 | 1 | 2 |
| | | | 2 | 0 | 3 | 2 | 3 | 2 |
| | | | 3 | 0 | 1 | 2 | 1 | 2 |
| T2F2T | Play C unless partner played 2 consecutive Ds in the last 3 periods (2 periods of punishment if partner plays D twice in a row) |  $cd, dd$<br>$cd, dd$<br>$cc, dc$<br>$cc,$ $dc$<br>C<br>C<br>D<br>D<br>$cc, dc$<br>$cd, dd$<br>$cd, dd$<br>$cc, dc$ | 1 | 1 | 1 | 2 | 1 | 2 |
| | | | 2 | 1 | 1 | 3 | 1 | 3 |
| | | | 3 | 0 | 4 | 3 | 4 | 3 |
| | | | 4 | 0 | 1 | 3 | 1 | 3 |
| Grim2 | Play C until 2 consecutive periods occur in which either player played D, then play D forever |  | 1 | 1 | 1 | 2 | 2 | 2 |
| | | | 2 | 1 | 1 | 3 | 3 | 3 |
| | | | 3 | 0 | 3 | 3 | 3 | 3 |
| Grim3 | Play C until 3 consecutive periods occur in which either player played D, then play D forever |  $cc$<br>$cc$<br>$cd, dd, dd$<br>$cc$<br>C<br>C<br>C<br>D<br>$cd, dd, dd$<br>$cd, dd, dd$ | 1 | 1 | 1 | 2 | 2 | 2 |
| | | | 2 | 1 | 1 | 3 | 3 | 3 |
| | | | 3 | 1 | 1 | 4 | 4 | 4 |
| | | | 4 | 0 | 4 | 4 | 4 | 4 |
| PT2FT | Play C if both players played C in the last 2 periods, both players played D in the last 2 periods, or both players played D 2 periods ago and C in the previous period. Otherwise play D |  $cd, dc$<br>$cc, dd$<br>$cc,$ $dd$<br>C<br>D<br>D<br>$cd, dc$<br>$cd, dc$<br>$cc, dd$ | 1 | 1 | 1 | 2 | 2 | 1 |
| | | | 2 | 0 | 3 | 2 | 2 | 3 |
| | | | 3 | 0 | 1 | 2 | 2 | 1 |
| DTFT | Play D in the first period, then play TFT |  | 1 | 0 | 2 | 1 | 2 | 1 |
| | | | 2 | 1 | 2 | 1 | 2 | 1 |
| DTF2T | Play D in the first period, then play TF2T |  $cd, dd$<br>$cc, dc$<br>$cc, dc$<br>$cd, dd$<br>D<br>C<br>C<br>D<br>$cd,$ $dd$<br>$cc, dc$<br>$cd, dd$<br>$cc, dc$ | 1 | 0 | 2 | 3 | 2 | 3 |
| | | | 2 | 1 | 2 | 3 | 2 | 3 |
| | | | 3 | 1 | 2 | 4 | 2 | 4 |
| | | | 4 | 0 | 2 | 4 | 2 | 4 |
| DTF3T | Play D in the first period, then play TF3T |  $cd, dd$<br>$cc, dc$<br>$cc, dc$<br>$cc, dc$<br>$cd, dd$<br>D<br>C<br>C<br>C<br>D<br>$cd,$ $dd$<br>$cc, dc$<br>$cd, dd$<br>$cd, dd$<br>$cc, dc$ | 1 | 0 | 2 | 3 | 2 | 3 |
| | | | 2 | 1 | 2 | 3 | 2 | 3 |
| | | | 3 | 1 | 2 | 4 | 2 | 4 |
| | | | 4 | 1 | 2 | 5 | 2 | 5 |
| | | | 5 | 0 | 2 | 5 | 2 | 5 |

Table A.3: Pre-programmed prisoner's dilemma strategies (19-22)

| Acronym | Description | Automaton | Matrix | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | $s$ | $\pi^c$ | $cc$ | $cd$ | $dc$ | $dd$ |
| DGrim2 | Play D in the first period, then play Grim2 |  D<br>C<br>C<br>D<br>cc<br>cc<br>cc<br>cd, dd, dd<br>cd, dd, dd<br>cd, dd, dd | 1 | 0 | 2 | 3 | 3 | 3 |
| | | | 2 | 1 | 2 | 3 | 3 | 3 |
| | | | 3 | 1 | 2 | 4 | 4 | 4 |
| | | | 4 | 0 | 4 | 4 | 4 | 4 |
| DGrim3 | Play D in the first periodperiod, then play Grim3 |  D<br>C<br>C<br>C<br>D<br>cc<br>cc<br>cc<br>cd, dd, dd<br>cc<br>cd, dd, dd<br>cd, dd, dd<br>cd, dd, dd | 1 | 0 | 2 | 3 | 3 | 3 |
| | | | 2 | 1 | 2 | 3 | 3 | 3 |
| | | | 3 | 1 | 2 | 4 | 4 | 4 |
| | | | 4 | 1 | 2 | 5 | 5 | 5 |
| | | | 5 | 0 | 5 | 5 | 5 | 5 |
| SGrim | Play C if both players played C, and D if both players played D. If one player played D and the other C, play C with probability $\alpha$. |  cd, dc<br>$\alpha$<br>cc<br>dd<br>cd, dc<br>cd, dc<br>cc<br>C<br>dd<br>D<br>dd<br>cc | 1 | 1 | 1 | 2 | 2 | 3 |
| | | | 2 | NA | 1 | 2 | 2 | 3 |
| | | | 3 | 0 | 1 | 2 | 2 | 3 |
| M1BF | Play C if both players played C, and D if both players played D. If the own action was C and the other player played D, play C with probability $\alpha$. If the own action was D and the other player played C, play C with probability $\beta$. |  cd<br>$\alpha$<br>cd<br>dd<br>cc<br>cd<br>dd<br>cc<br>C<br>cd<br>dc<br>D<br>dd<br>cc<br>dc<br>dd<br>cc<br>$\beta$<br>dc<br>dc | 1 | 1 | 1 | 2 | 3 | 4 |
| | | | 2 | NA | 1 | 2 | 3 | 4 |
| | | | 3 | NA | 1 | 2 | 3 | 4 |
| | | | 4 | 0 | 1 | 2 | 3 | 4 |