stratEst: Strategy Estimation in R

Fabian Dvorak
Universität Konstanz

Abstract

stratEst is a software package for the estimation of finite mixture models of discrete choice strategies in the statistical computing environment R. The discrete choice strategies are deterministic finite state automata that can be customized by the user to fit the structure of the data. The parameters of the strategy estimation model are the relative frequencies and the choice parameters of the strategies. The model can be extended by adding individual level covariates to explain the selection of strategies by individuals. The estimation function of the package uses expectation maximization and Newton-Raphson methods to find the maximum likelihood estimates of the model parameters. The package contains additional functions for data processing and simulation, strategy generation, parameter tests, model checking, and model selection.

Keywords: decision experiments, discrete choice strategies, mixture models, R.

1. Introduction

stratEst is a software package for strategy estimation in the statistical computing environment R (R Development Core Team, 2008). The goal of strategy estimation is to explain discrete choices of a sample of individuals by a finite mixture of individual choice strategies. Strategy estimation is a form of mixture modeling (McLachlan and Peel 2005), and similar to cluster analysis (Kaufman and Rousseeuw 1990), and latent class analysis (Lazarsfeld 1950). All methods essentially assign observed entities to unobservable classes. In strategy estimation, the entities are individuals observed in a specific choice environment and the unobservable classes are discrete choice strategies.

The **stratEst** package provides a general framework for strategy estimation in R. In principle, the package can be used to fit strategy estimation models to any data set with discrete choices. The main challenge for the generality of the strategy estimation framework is that the candidate strategies must correspond to the choice environment. Strategies that are plausible candidates in one choice environment are often meaningless in other choice environments.

The solution implemented by the package is that candidate strategies are represented as deterministic finite state automata that can be customized by the user. The advantage of the representation as deterministic finite state automata is a reduction of complexity that facilitates the programming of strategies. In the automaton representation, the choice probabilities over the alternatives are a function of a finite set internal states of the automaton and not a function of the larger set of situations that might be encountered by the individual in the choice environment. On the other hand, the concept of finite state automata is flexible enough to represent candidate strategies of many different choice environments.

Strategy estimation was introduced by Dal Bó and Fréchette (2011) to estimate the maximum likelihood frequencies of a set of candidate strategies in the repeated prisoner's dilemma. Since the original publication, strategy estimation was used in several other studies that, almost exclusively, focus on the repeated prisoner's dilemma (e.g. Aoyagi, Bhaskar, and Frechette 2019; Arechar, Dreber, Fudenberg, and Rand 2017; Camera, Casari, and Bigoni 2012; Embrey, Frechette, and Yuksel 2017; Fudenberg, Rand, and Dreber 2012; Frechette and Yuksel 2017).

Embrey, Frechette, and Stacchetti (2013) shift the scope of strategy estimation beyond the prisoner's dilemma and perform strategy estimation in a repeated partnership game with more than two choices. Breitmoser (2015) extends the strategy estimation model of Dal Bó and Fréchette (2011) by adding model parameters for the choice probabilities of the strategies. Dvorak and Fehrler (2018) extend the strategy estimation model further by adding individual level covariates to explain the selection of strategies by individuals.

The parameters of the basic strategy estimation model are the relative frequencies and the choice parameters of the strategies. In the model extension with individual level covariates, the parameters for the relative frequencies of the strategies are replaced by logit coefficients for the effect of the covariates. The estimation function of the package obtains maximum likelihood estimates for the model parameters based on expectation maximization (Dempster, Laird, and Rubin 1977) and Newton-Raphson algorithms.

To speed up the estimation, the package integrates C++ and R with the help of the R packages **Rcpp** (Eddelbuettel and François 2011) and the open source linear algebra library for the C++ language **RppArmadillo** (Sanderson and Curtin 2016). Package development is supported by the packages **devtools** (Wickham, Hester, and Chang 2020b), **testthat** (Wickham 2011), **roxygen2** (Wickham, Danenberg, Csardi, and Eugster 2020a), and **Sweave** (Leisch 2002).

Strategy estimation can also be conducted based on R packages for cluster and latent class analysis like Flexmix (Leisch 2004), poLCA (Linzer and Lewis 2011), and randomLCA (Beath 2011). A potential drawback of using these packages for strategy estimation is that the candidate strategies must have the same structure i.e., the same set of internal states and deterministic state transitions. This often implies that a reasonable set of candidate strategies cannot be constructed for the data at hand.

Throughout the paper, text in `typewriter` font represents R code. The symbol `R>` at the beginning of a new line marks the beginning of a command that should be executed in the R console.

Installation

The most recent CRAN version of **stratEst** is installed by executing the following command in the R console:

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

After the installation, the package is loaded into memory and attached to the search path with the command:

```
R> library(stratEst)
```

Rock-paper-scissors: An introductory example

I illustrate the core features of the package on the basis of the game rock-paper-scissors. In each period of this game, two players simultaneously choose one of three possible actions: rock, paper or scissors. The winner of the period is determined by the following rule: rock crushes scissors, scissors cuts paper, and paper covers rock. If both players choose the same action, this results in a tie. Rock-paper-scissors has a unique Nash equilibrium. The Nash equilibrium suggest that every player uses the same strategy. This strategy plays each of the three actions with probability one-third.

The data set `WXZ2015` contains the data of a rock-paper-scissors experiment conducted by Wang, Xu, and Zhou (2014). The data contains the observations 72 university students playing 300 periods of the rock-paper-scissors game in groups of six participants. In each period, each participant is randomly matched with another participant from the own group. In the experiment, 35.7 percent of all actions are rock (`r`), 32.2 percent are paper (`p`), and 32.1 percent are scissors (`s`) which seems to be fairly inline with the Nash equilibrium prediction.

There are many other choice strategies which can explain the observed distribution of choices. Wang *et al.* (2014) show that a conditional response strategy provides a better explanation for the data than the Nash equilibrium strategy. The conditional response strategy is more complex than Nash play as it takes the outcome of the previous period into account for the choice in current period.

The observed distribution of choices can also be explained by a finite mixture model of several strategies. To give an example, consider a uniform mixture of three types of players, one who always plays rock, one who always plays paper, and one who always plays scissors.

This example illustrates how to use the package in order to fit and compare different strategy estimation models to the data of the rock-paper-scissors experiment. The different strategy estimation models are selected to illustrate the features of the package and lack the theoretical justification provided by (Wang *et al.* 2014) for the conditional response strategy.

Programming strategies

The strategy generation function of the package is `stratEst.strategy()`. The following code creates two strategies: a mixed strategy with unspecified choice probabilities, and the Nash strategy.

```
R> rps = c("r", "p", "s")
R> mixed = stratEst.strategy(choices = rps)
R> nash = stratEst.strategy(choices = rps, prob.choices = rep(1/3, 3))
```

The argument `choices` expects a character vector with the names of the choices. The argument `prob.choice` can be used to define the choice probabilities of the strategy. If printed out in the console, the strategies look like this:

```
R> print(mixed)

  prob.r prob.p prob.s
1     NA     NA     NA

R> print(nash)
```

```
  prob.r prob.p prob.s
1  0.333  0.333  0.333
```

The objects `mixed` and `nash` returned by the strategy function are data frames of class `stratEst.strategy`. Since the choice probabilities of the mixed strategy were not specified, the choice probabilities are `NA`. This indicates to the estimation function that these parameters should be estimated from the data.

The strategies `nash` and `mixed` are simple strategies in the sense that the choice probabilities of these strategies do not change from one situation to the next. The strategies only have one internal state and one associated set of choice probabilities. More complex strategies have several states with different sets of choice probabilities. The complex strategies transition from one state to the other after some input is observed. The input may be a specific situation or history of events in the game.

To illustrate the concept, a strategy is generated which randomizes in the first period and subsequently imitates the choice of the previous period. The following code generates the strategy `imitate`:

```
R> last.choice = c(NA, rps)
R> imitate =  stratEst.strategy(choices = rps, inputs = last.choice,
+                                 num.states = 4,
+                                 prob.choices = c(rep(1/3, 3), 1, 0, 0,
+                                                  0, 1, 0, 0, 0, 1),
+                                 tr.inputs = rep(c(2, 3, 4), 4))
```

What changed compared to the previous function calls is that the argument `inputs` is used to define a set of inputs. This set contains the names of all possible choices in the last period of the game. The value `NA` in the set indicates that the input can be missing. This is the case in the first period when no information from the previous period exists. The argument `num.states` defines the number of states of the strategy. The argument `tr.inputs` defines the deterministic state transitions for all possible inputs in all states. The result is a strategy with four states. Each state is represented by one row of the object `imitate`.

```
R> print(imitate)
```

```
  prob.r prob.p prob.s tremble tr(r) tr(p) tr(s)
1  0.333  0.333  0.333      NA     2     3     4
2  1.000  0.000  0.000      NA     2     3     4
3  0.000  1.000  0.000      NA     2     3     4
4  0.000  0.000  1.000      NA     2     3     4
```

The strategy `imitate` has a column named `tr(x)` for each possible input `x`. An exception is the element `NA` that indicates the missing input in the first period. The values supplied to the argument `tr.inputs` appear in row wise order.

The strategy `imitate` also contains a column with the name `tremble`. The values in this column indicate the probability to choose one of the choices not prescribed by the strategy in the current state. A tremble probability is usually necessary for a pure strategy with choice probabilities of zero and one. The purpose of the tremble probability is to avoid that a single deviation an individual from the choice pattern of the pure strategy results in a likelihood of zero that the individual uses the strategy. The tremble probability can be specified by the argument `trembles`. If the argument is missing, the probability of a tremble is `NA`. This signals to the estimation function that the parameter should be estimated from the data.

The strategy `imitate` transitions from one state to the other by the following deterministic rule.In the first period, the strategy observes the input `NA` since there is no information on the previous choice available. By convention, whenever the input is `NA`, the strategy moves to its start state. The start state of the strategy is the state represented by the first row. The strategy makes a choice according to the choice probabilities in the first row. In period two, the strategy observes the input (either rock, paper or scissors) and moves to the next state defined by the value of `tr(input)` in the current state. The values supplied to `tr.inputs` define the desired behavior of the strategy `imitate`. It randomly makes a choice in the first period, and subsequently plays rock after rock, paper after paper, and scissors after scissors.

Processing data

In order fit the strategies to the rock-paper-scissors data, the data must be in a suitable format. The function `stratEst.data()` can be used to reshape the raw data.

```
R> data.WXZ2014 <- stratEst.data(data = WXZ2014, choice = "choice",
+                                input = c("choice"), input.lag = 1,
+                                id = "id", game = "game",
+                                period = "period")
```

To the first argument of the function, we pass a `data.frame` object with variables in columns. We need to specify the variable in the data which contains the discrete choices using the argument `choice`. The argument `input` allows us to select one or more variable names which serve as input for the strategies in the estimation. If we select more than one variable, the function concatenates the values of these variables to a unique factor level. For this example, we only need one input variable, the choices of the participants in the experiment. As the input should reflect the choice in the previous period we specify a lag of one period. The arguments `id`, `game`, and `period` uniquely identify the participant, and the period within the game.

The function `stratEst.data()` returns an data frame object of class `stratEst.data`. We can inspect this object by printing it to the console with the command `print(data.WXZ2015)`.

Model estimation

The function has two mandatory arguments which are `data` and `strategies`. The object passed to argument `data` must be of class `stratEst.data`. The object passed to argument `strategies` must be a list of `stratEst.strategy` objects. The following code fits four different models to the rock-paper-scissors data:

```
R> model.nash <- stratEst.model(data = data.WXZ2014,
+                               strategies = list("nash" = nash))
R> model.mixed <- stratEst.model(data = data.WXZ2014,
```

```
+                                strategies = list("mixed" = mixed))
R> model.imitate <- stratEst.model(data = data.WXZ2014,
+                                  strategies = list("imitate" = imitate))
R> model.mixture <- stratEst.model(data = data.WXZ2014,
+                                  strategies = list("nash" = nash,
+                                                    "imitate" = imitate))
```

The estimation function `stratEst()` estimates the model parameters and returns a list object of class `stratEst.model`. The elements of this list can be accessed with the syntax `model$object` where object is an object name in `names(model)`. The generic function `summary()` prints a summary of a fitted model to the console.

The function `stratEst.check()` can be used to inspect the global model fit. It summarizes the log likelihood of the model, the number of free model parameters, and the values of three information criteria. The three information criteria are the Akaike information criterion (`aic`), the Bayesian information criterion (`bic`), and Integrated classification likelihood (`icl`).

```
R> models <- list(model.nash, model.mixed, model.imitate, model.mixture)
R> compare <- do.call(rbind, unlist(lapply(models, stratEst.check),
+                                   recursive = F))
R> rownames(compare) <- c("model.nash", "model.mixed", "model.imitate",
+                         "model.mixture")
R> print(compare)

                loglike free.par      aic      bic      icl
model.nash    -23730.03        0 47460.05 47460.05 47460.05
model.mixed   -23704.04        2 47412.09 47416.64 47416.64
model.imitate -23205.91        1 46413.82 46416.10 46416.10
model.mixture -22358.43        2 44720.87 44725.42 44728.34
```

We see that the fit of the model with the mixed strategy is better than the fit of the model with the Nash strategy. The values of the information criteria indicate that this is true even if we take into account that the model with the mixed strategy has more free parameters. The estimated choice probabilities of the mixed strategy can be accessed with the command `model.mixed$probs.par`. The estimated choice probabilities reflect the overall distribution of choices. We can test whether the estimated choice probabilities differ from one-third using the function `stratEst.test()`. With the option `par = "probs"`, the function performs a t test for each estimated choice probability:

```
R> t.probs <- stratEst.test(model = model.mixed, par = "probs", values = 1/3)
R> print(t.probs)

            estimate    diff std.error  t-value df Pr(>|t|)
probs.par.1   0.3223 -0.0111    0.0014  -8.0838 70        0
probs.par.2   0.3566  0.0232    0.0013  17.6404 70        0
probs.par.3   0.3212 -0.0122    0.0012 -10.3417 70        0
```

The model with the strategy `imitate` yields a better fit than the model with the mixed strategy despite having one free parameter less. However, the best global fit is obtained the mixture model of `nash` and `imitate`. The log likelihood of the mixture model is substantially larger than the log likelihood of all other models. The following commands print the estimated shares and strategies of this model the console:

```
R> print(model.mixture$shares, digits = 2)
```

```
      nash imitate
share 0.58    0.42
```

```
R> print(model.mixture$strategies, digits = 3)
```

```
$nash
  prob.r prob.p prob.s tr(p) tr(r) tr(s) tremble
1  0.333  0.333  0.333     1     1     1      NA

$imitate
  prob.r prob.p prob.s tremble tr(r) tr(p) tr(s)
1  0.333  0.333  0.333      NA     2     3     4
2  1.000  0.000  0.000   0.391     2     3     4
3  0.000  1.000  0.000   0.391     2     3     4
4  0.000  0.000  1.000   0.391     2     3     4
```

The estimated shares suggest that each strategy is used by approximately half of the participants in the experiment. The fitted tremble parameter of the strategy `imitate` indicates that a different choice than the one predicted by the strategy is chosen in 39 percent of all observations. In these observations, the strategy suggests that players randomly pick one of the other choices.

2. Terminology and model definition

Suppose we observe discrete choices of N individuals among R choice alternatives. Each choice occurs after a certain history $j \in J$ of observable events. The strategy estimation model assumes that the discrete choices arise from a finite-mixture of K choice strategies. Each individual i ($i = 1, \ldots, N$) follows one of the K strategies. Each strategy k ($k = 1, \ldots, K$) assigns a strategy specific state s_k ($s_k = 1, \ldots, S_k$) to history $j \in J$. The subscript k of the index s_k which indicates that the strategies can have different numbers of states is ignored ignored for better readability. State s determines the probability π_{ksr} that strategy k chooses alternative r after history j.

Let y^i_{ksr} denote the number of times individual i chooses alternative r after all histories for which the state of strategy k is s. The total number of choices observed after these histories is $n^i_{ks} = \sum_{r=1}^{R} y^i_{ksr}$. The central modeling assumption of the strategy estimation model is conditional independence (Bandeen-Roche, Miglioretti, Zeger, and Rathouz 1997). Conditional independence implies that the n^i_{ks} choices are independent conditional on the strategy of individual i. If individual i uses strategy k, the probability to observe the choice vector $Y_{ks}^i = (y_{ks1}^i, \cdots, y_{ksR}^i)$ follows n_{ks}^i independent draws from a multinomial distribution defined by the vector of probabilities $\pi_{ks} = (\pi_{ks1}, \cdots, \pi_{ksR})$ with $\pi_{ksr} \in [0,1]$ and $\sum_{r=1}^R \pi_{ksr} = 1 \; \forall \; s \in S_k$ and $k \in K$.

2.1. The basic strategy estimation model

Let p_k denote the share of individuals in the population which follow strategy k defined by the collection of $R \times S_k$ multinomial parameters π_{ksr}. The estimation function of the package returns the estimates p_k^*, π_{ksr}^* that maximize the log likelihood:

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

The parameter constraints are $p_k \in [0,1]$, $\sum_{k=1}^K p_k = 1$, $\pi_{ksr} \in [0,1]$ and $\sum_{r=1}^R \pi_{ksr} = 1$. The log likelihood defined in Equation (refeq: ln L) neglects the multinomial coefficients of the likelihood which are constant factors and do not affect the location of the optima.

2.2. The model with covariates

The strategy estimation model with covariates has two parts: a measurement and a structural part. The measurement part contains the choice parameters of the strategies and is the same as in the model without covariates. The structural part of the model explains the prior probability p_{ik} that individual i uses strategy k as a function of individual level covariates. The structural part of the model is the same as in latent class regression (Dayton and Macready 1988; Bandeen-Roche *et al.* 1997).

The structural part uses the first strategy as the benchmark. The the log odds of using strategy k compared to the first strategy are modeled by the multinomial logit link function (Agresti 2003). Let x_i denote a row vector that contains the covariates of individual i, then:

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

where p_{ik} is the prior probability that individual i uses strategy k and β_k is a column vector of C coefficients. The K equations above yield:

$$p_{ik} = \frac{e^{x_i \beta_k}}{\sum_{k=1}^K e^{x_i \beta_k}}$$

The log likelihood function of the model with covariates is:

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

The structural part of the model assumes non-differential measurement (Bandeen-Roche *et al.* 1997). Non-differential measurement means that the individual level covariates are not associated with choices if we control for the strategies of the individuals. The measurement part of the model assumes local conditional independence.

When fitting a model with covariates, the parameters in the structural part and the measurement part are estimated simultaneously. This presents an advantage over a two-step estimation. In the two-step estimation, the measurement part of the model is estimated first and individuals are assigned to strategies on the basis of the posterior probability to use each strategy. In the second step, the classification of individuals is used as the dependent variable in a multinomial model with the individual level covariates as independent variables. It can be shown that the two-step approach suffers from downward biased regression coefficients for the effects of covariates if the classification of individuals is noisy (Bolck, Croon, and Hagenaars 2004).

3. Estimation

The estimation function of the package is `stratEst.model()`. The function obtains the maximum likelihood estimates of the model parameters outlined in (refeq: ln L) or (refeq: ln L latent class regression). The estimated model parameters are returned by the estimation function as objects `shares.par`, `probs.par`, `trembles.par`, and `coefficients.par`. The standard errors of the parameter estimates are returned as objects `shares.se`, `probs.se`, `trembles.se`, and `coefficients.se`.

3.1. Algorithm

The estimation function uses expectation maximization (EM, Dempster *et al.* 1977) and Newton-Raphson methods to obtain the maximum likelihood estimates of the model parameters. The expectation maximization algorithm exploits the fact that the maximum likelihood estimates of the strategy parameters could be inferred if the assignments of individuals to strategies were known.

The optimization procedure randomly initializes parameter subject to the parameter constraints. After initialization the EM algorithm iterates between two steps until convergence. 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. For the model without covariates, the posterior probability that individual i uses strategy k is a function of the prior probability p_k and likelihood of the choices given the strategy parameters π_{ksr}.

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

For the model with covariates the prior probability is replaced by the probability p_{ik} which is a function of the covariates.

In the *maximization step* of each iteration, the model parameters are updated to the values that maximize (1) or (2) conditional on the calculated posterior probability assignments of individuals to strategies.

After all model parameters have been updated, the log likelihood of the updated model is determined based on (1) or (2) and compared to the log likelihood calculated in the previous iteration. The algorithm continues with the next iteration as long as the increase in the log likelihood exceeds a certain threshold.

To avoid that local optima are returned by the estimation function, the optimization procedure performs a series of short 'inner' runs of the EM algorithm from different starting points. The best solution obtained in the inner runs is used as the starting point of an 'outer' run of EM.

The estimation function of the package returns the best solution obtained in the outer runs. Biernacki, Celeux, and Govaert (2003) show that this method can be used to efficiently locate the maximum likelihood parameters of mixture models.

3.2. Parameter maximization

In the maximization step of each iteration, the model parameters are updated according to the following rules.

Strategy shares

In the model without covariates, the population shares p_k are updated to the expected values of the posterior probability assignments of individuals to strategies. The optimization of strategy shares p_k, with respect to a sum-to-one constraint is performed 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 λ 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 yields

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

If user defined values are supplied for some strategy shares the remaining strategy shares are scaled by one minus the sum of these values to fulfill the sum-to-one constraint.

Choice probabilities

The strategy parameters π_{ks} are updated based on K weighted data sets. To obtain the weighted data for strategy k, the choices of individual i are considered proportional to the posterior probability θ_{ik} that individual i uses strategy k. Using Lagrange multipliers, the updated choice probabilities π_{ksr} follow from

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

If k is a pure strategy, it is assumed that the choice probabilities π_{ksr} are the result of trembling hand errors (Selten 1975). Let ξ_{ks} denote a vector of pure choice parameters with elements $\xi_{ksr} \in \{0, 1\}$ and $\gamma_{ks} \in [0, 1]$ the probability of a tremble. The choice probabilities π_{ksr} follow from

$$\pi_{ksr} = \xi_{ksr}(1 - \gamma_{ks}) + (1 - \xi_{ksr}) \frac{\gamma_{ks}}{R - 1}. \tag{6}$$

Equation 6 describes a process in which a tremble uniformly implements one of the choices not predicted by the strategy. The tremble rules out that a single choice which is not predicted by a pure strategy results in a likelihood of zero that the individual uses the strategy.

The pure choice parameters ξ_{ksr} are updated in the maximization step by a transformation of the updated choice probabilities π_{ksr} according to

$$\xi_{ksr}^{next} = \begin{cases} 1 & \text{if } \pi_{ksr}^{next} > \pi_{ksr'}^{next} \; \forall \; r' \neq r \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Equation 7 assigns density of one to the maximum of the updated vector π_{ks}^{next}. This assures that the corresponding tremble parameters γ_{ks} remain as small as possible. If there is more than one parameter with the maximum probability, the first parameter is set to one and the others to zero.

The updated values of the trembles follow from the substitution of (6) into (5). For the update of the tremble all choice probabilities affected by the tremble are taken into account which yields

$$\gamma_{ks}^{next} = \sum_{i=1}^{N} \frac{\theta_{ik} \sum_{r=1}^{R} (y_{ksr}^{i} - n_{ks}^{i} \xi_{ksr}^{next}) \left(\frac{R-1}{1-R \cdot \xi_{ksr}^{next}} \right)}{\sum_{i=1}^{N} \theta_{ik} \cdot R \cdot n_{ks}^{i}}. \tag{8}$$

Whenever parameters specified by the user are pure (i.e. zero or one), stratEst will automatically estimate a tremble parameter.

Regression coefficients

The regression coefficients of the model with covariates are updated based on a a Newton-Raphson step (Bandeen-Roche *et al.* 1997). The updated column vector of coefficients β is

$$\beta^{next} = \beta - H_{\beta}^{-1} \nabla_{\beta} \tag{9}$$

where ∇_{β} is the score of the coefficient vector with elements

$$\frac{\partial \ln L}{\partial \beta_{qk}} = \sum_{i=1}^{N} x_{iq} (\theta_{ik} - p_{ik}) \tag{10}$$

in columns and H_{β} is the Hessian of (2) for the coefficients with elements

$$\frac{\partial^2 \ln L}{\partial \beta_{bl} \partial \beta_{ck}} = \sum_{i=1}^{N} x_{ib} x_{ic} (\theta_{il} (\delta_{lk} - \theta_{ik}) - p_{il} (\delta_{lk} - p_{ik})) \tag{11}$$

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. In order to calculate p_{ik}, for individual i, the row vector x_i which contains the covariates of individual i cannot contain missing values.

Firth (1993) proposes to use the penalized log likelihood function $\ln L^p = \ln L + \frac{1}{2} log(|H_\beta|)$ to account for the fact that the maximum likelihood estimates of the coefficients are biased in finite samples. In some instances the maximum likelihood coefficients for the sample can be infinite even though the true coefficients are not.

The Firth correction produces reasonable estimates and standard errors in such situations by penalizing the likelihood function with the Jeffreys' invariant prior. Using the penalized likelihood $L^p = L|H_\beta|^{\frac{1}{2}}$ effectively shrinks coefficients towards zero which guarantees that maximum likelihood estimates do exist.

Taking the derivative of the penalized log likelihood with respect to the coefficients yields the penalized score vector $\hat{\nabla}_{\beta_{ah}}$. Bull, Mak, and Greenwood (2002) show that the small sample bias can effectively be reduced by using the penalized score $\hat{\nabla}_{\beta_{ah}}$ to update the coefficients of the multinomial logistic regression model. The penalized score vector of the regression coefficients in the model with covariates is:

$$\hat{\nabla}_{\beta_{ah}} = \frac{\partial \hat{\ln L}^p}{\partial \beta_{ah}} = \sum_{i=1}^{N} (x_{ia}(\theta_{ih} - p_{ih})) + \frac{1}{2} \operatorname{tr} \left(H_\beta^{-1} \frac{\partial H_\beta}{\partial \beta_{ah}} \right) \tag{12}$$

where $h \in \{1, \cdots, K\}$ and $a \in \{1, \cdots, C\}$ and $\operatorname{tr}(\cdot)$ is the trace of the matrix. The derivative of the element in row l with covariate b and column k with covariate c of the hessian matrix H_β with respect to β_{ah} is:

$$\begin{aligned} \frac{\partial H_\beta}{\partial \beta_{ah}} = \sum_{i=1}^{N} x_{ia} x_{ib} x_{ic} (\theta_{il}(\delta_{lk} - \theta_{ik})(\delta_{lh} - \theta_{ih}) - \theta_{il}\theta_{ik}(\delta_{kh} - \theta_{ih}) + \\ p_{il}(\delta_{lk} - p_{ik})(\delta_{lh} - p_{ih}) - p_{il}p_{ik}(\delta_{kh} - p_{ih})) \end{aligned}$$

where $h, l, k \in \{1, \cdots, K\}$ and $a, b, c \in \{1, \cdots, C\}$ and $\delta_{ab} = 1$ if $a = b$ and 0 otherwise.

3.3. Standard errors

Standard errors of the model parameters are estimated by methods employed by Linzer and Lewis (2011) using the *empirical observed* information matrix (Meilijson 1989). The *empirical observed* information matrix is

$$I_e(Y, \hat{\Psi}) = \sum_{i=1}^{N} s(Y_i, \hat{\Psi}) s^T(Y_i, \hat{\Psi}), \tag{13}$$

where $s(Y_i, \hat{\Psi})$ is the score contribution of individual i with respect to parameter vector Ψ, evaluated at the maximum likelihood 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 η_r with $\sum_{r=1}^{R} \eta_r = 1$, the score function $s(Y_i, \hat{\eta_r})$ transformed into log-ratios $\mu_r = ln(\eta_r/\eta_1)$ and the variance-covariance matrix $\mathrm{VAR}(\eta)$ is calculated based on (13). The variance-covariance matrix $\mathrm{VAR}(\mu)$ of the parameters is approximated using the delta method

$$\mathrm{VAR}(f(\hat{\mu})) = f'(\mu) I_e(Y, \hat{\mu})^{-1} f'(\mu)^T, \tag{14}$$

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.

The following score contributions are used to calculate the empirical observed information matrix defined in (13).

Strategy shares

The shares are transformed into log-rations as $p_k^* = ln(p_k/p_1)$ and the score contribution $\partial \ln L / \partial p_k^*$ of individual i is

$$s(Y_i, p_k^*) = \theta_{ik} - p_k. \tag{15}$$

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

$$\frac{\partial f(p_k^*)}{\partial p_l^*} = \begin{cases} -p_k p_l & \text{if } l \neq k \\ p_k(1-p_l) & \text{if } l = k \end{cases} \tag{16}$$

and the variance-covariance matrix of the shares is estimated by (14) using the inverse of (13) based on the score contributions of the shares defined in (15).

Choice probabilities

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

$$s(Y_i, \pi_{ksr}^*) = \theta_{ik}\left(y_{ksr}^i - n_{ks}^i \pi_{ksr}\right). \tag{17}$$

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

$$\frac{\partial g(\pi_{ksr}^*)}{\partial \pi_{ltq}^*} = \begin{cases} -\pi_{ksr}\pi_{ltq} & \text{if } k = l \text{ and } s = t \text{ and } r \neq q \\ \pi_{ksr}(1-\pi_{ltq}) & \text{if } k = l \text{ and } s = t \text{ and } r = q \\ 0 & \textit{otherwise} \end{cases} \tag{18}$$

and the variance-covariance matrix of the choice probabilities is estimated by (14) using the inverse of (13) based on the score contributions defined in (17).

For a pure strategy, the score contribution $\partial \ln L/\partial \gamma_{ks}$ of individual i is

$$s(Y_i, \gamma_{ks}^*) = \theta_{ik} \sum_{r=1}^{R} \frac{y_{ksr}^i}{\pi_{ksr}} \left(\frac{1-\xi_{ksr}}{R-1} - \xi_{ksr}\right) \tag{19}$$

the reported estimates of the variance-covariance of the tremble probabilities is the inverse of (13) using the score contributions outlined in (19).

Regression coefficients

The reported estimates of the variance-covariance is the inverse of (13) using the score of the regression coefficients outlined in (10) or (12) if the Firth penalty is used.

Bootstrapped standard errors

Standard errors of the model parameters can also be obtained by a parametric bootstrap procedure (Efron and Tibshirani 1993). In each bootstrap sample m (m=,1,...,M), parameter estimates are obtained based on the observations of N individuals sampled with replacement. 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 to maintain the original structure of the model across the bootstrap estimates.

For the model with covariates, the Firth penalty is used to obtain the estimates of the regression coefficients for each sample. The reason is that it is likely that some samples suffer from quasi complete separation. In a sample with quasi complete separation, maximum likelihood estimates of the regression coefficients do not exist. The penalized estimation prevents that extreme parameter values are obtained in these samples which would bias the estimated standard errors of the regression coefficients.

4. Model fit

The model checking function of the package is `stratEst.check()`. The function returns the log likelihood of the model, the number of free model parameters, and the values of three information criteria. The function can also be used to assess the global and local model fit based on the Pearson chi square goodness of fit statistic.

4.1. Information criteria

Three different penalized-likelihood criteria can be used to assess the global model fit. The criteria are the Akaike Information Criterion (AIC, Akaike 1973), the Bayesian Information Criterion (BIC, Schwarz 1978), and the Integrated Classification Likelihood (ICL, Biernacki, Celeux, and Govaert 2000). The formulas for the three model selection criteria are

$$\text{AIC} = -2\ln L + 2df$$

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

$$\text{ICL} = \text{BIC} + 2\sum_{i=1}^{N}\sum_{k=1}^{K}\theta_{ik}log(\theta_{ik}),$$

In all three formulas, df represents the number of free parameters of the model returned as the object `model$free.par`. The three information criteria differ in the size of the penalty for model complexity. AIC penalizes the log likelihood with two times the number estimated parameters. BIC penalizes the log likelihood with the number of estimated parameters times the natural logarithm of the number of observations (`model$num.obs`). ICL uses the BIC penalty plus an extra penalty term for the entropy of the posterior probability assignments of individuals to strategies.

4.2. χ^2 test of global fit

The Pearson χ^2 goodness of fit test can be used to assess the global model fit of latent class models (van Kollenburg, Mulder, and Vermunt 2015). Pearson χ^2 goodness of fit test statistic is:

$$\chi^2 = \sum_{k=1}^{K}\sum_{s=1}^{S_k}\sum_{r=1}^{R}\frac{(o_{ksr} - e_{ksr})^2}{e_{ksr}} \tag{20}$$

where $o_{ksr} = \sum_{i=1}^{N}\theta_{ik}y^{i}_{ksr}$ and $e_{ksr} = \sum_{i=1}^{N}p_k\pi_{ksr}n^{i}_{ks}$ represent the observed and the expected number of choices of alternative r by strategy k in state s. As the assignments of individuals to strategies are unknown, the statistic is calculated using the posterior probability assignment θ_{ik} of individual i to strategy k.

The distribution of the test statistic is estimated by a parametric bootstrap. The bootstrap procedure simulates M samples of data for the fitted model. In each sample, individuals are randomly assigned to the strategies with probabilities equal to the estimated shares. For each individual, choices are simulated conditional on the input observed by the individual and the fitted choice parameters of the fitted strategy. The distribution of the test statistic is approximated by calculating the statistic defined in (20) in each of the M samples.

4.3. χ^2 test of local fit

To local fit of each strategy is assessed by assigning individuals to strategies based on the maximum values of the posterior probability assignments (Bandeen-Roche *et al.* 1997). Let N_k denote the set of all individuals with a posterior probability maximum for strategy k. The Pearson χ^2 statistic for strategy k is:

$$\chi_k^2 = \sum_{i \in N_k} \sum_{s=1}^{S_k} \sum_{r=1}^{R} \frac{(o_{ksr} - e_{ksr})^2}{e_{ksr}} \tag{21}$$

with $o_{ksr} = y_{ksr}^i$ and $e_{ksr} = \pi_{ksr} n_{ks}^i$ as the observed and the expected number of choices of alternative r by strategy k in state s.

The distributions of the K local fit statistics are estimated by a parametric bootstrap. The bootstrap simulates M samples of data for the fitted model. In each sample, individuals are randomly assigned to the strategies with probabilities equal to the estimated shares. For each individual, choices are simulated conditional on the input observed by the individual and the fitted choice parameters of the fitted strategy. For each sample individuals are assigned to strategies based on the maximum values of the posterior probability. The distribution of the test statistic is approximated by calculating the statistic defined in (20) in each of the M samples.

5. Model selection

The number of free model parameters equals $(K-1)+(R-1)\cdot\sum_{k=1}^{K} S_k$ for the model without covariates and $C(K-1) + (R-1) \cdot \sum_{k=1}^{K} S_k$ for the model with covariates. Four different methods can be used to reduce the number of free model parameters.

5.1. Parameter fixation

The first method is to fix model parameters at user defined values. This option exists for all classes of model parameters. The fixation of model parameters can often be justified on the basis of theory. It is generally possible to fixate only a subset of parameters of the same class (For example two out of four strategy shares). An exception is the class of regression coefficients. For this class, either all or no parameter can be fixed. Fixed parameters are not estimated and reduce the number of free model parameters accordingly.

The fixation of parameters which are subject to a sum-to-one constraint affects all other parameters affected by the constraint. If parameters are fixed at a certain value, the remaining parameters are updated and subsequently scaled by the one minus the sum of fixed parameters.

5.2. Parameter restrictions

The argument `r.probs` of the estimation function `stratEst.model()` can be used to restrict the number of estimated choice probabilities π. Three options can be used which are `"strategies"`, `"states"`, and `"global"`.

The option `"strategies"` estimates one parameter vector $\pi_k = (\pi_{k1}, \ldots, \pi_{kR})$ for each of the K strategies. The vector π_k determines the probability of choices in all states $s \in \{1, \ldots, S_k\}$ of strategy k, reducing the number of free model parameters by $(R-1) \cdot \sum_{k=1}^{K} S_k - 1$.

The option `"states"` estimates one parameter vector $\pi_s = (\pi_{s1}, \ldots, \pi_{sR})$ for each state $s \in \{1, \ldots, max(S_k)\}$. The vector π_s determines the probability of choices in state s for all strategies, reducing the number of free model parameters by $(R-1) \cdot \sum_{k=1}^{K} S_k - 1$.

The option `"global"` estimates a single parameter vector $\pi = (\pi_1, \ldots, \pi_R)$ that determines the probability of choices in all states of each strategy, reducing the number of free model parameters by $(R-1) \cdot \sum_{k=1}^{K} S_k - 1$.

For pure strategies, the argument `r.trembles` works equivalently. The option `"strategies"` estimates one tremble probability γ_k per strategy. The option `"states"` estimates one tremble probability γ_s per state. The option `"global"` estimates a single tremble probability γ_k which applies globally.

Restricted parameter estimation

If restrictions to the strategy parameters apply, the maximization step in the parameter estimation needs to be adapted accordingly. Let Z_t denote the set of all states s of strategy k where the corresponding strategy parameters are restricted to have the same underlying parameter vector ζ_t, where $t(t = 1, \ldots, T)$ is the index of the restrictions. The individual score contributions to ζ_t take all parameters affected by restriction t into account, i.e.

$$\pi_{tr}^{next} = \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{s \in Z_t} \frac{\theta_{ik} y_{ksr}^{i}}{\sum_{i=1}^{N} \sum_{s \in Z_t} \theta_{ik} n_{ks}^{i}} \tag{22}$$

if ζ_t is a vector of choice probabilities. The tremble probabilities ζ_t are updated according to

$$\gamma_t^{next} = \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{s \in Z_t} \frac{\theta_{ik} \sum_{r=1}^{R} (y_{ksr}^{i} - n_{ks}^{i} \xi_{ksr}^{next}) \left(\frac{R-1}{1-R \cdot \xi_{ksr}^{next}} \right)}{\sum_{i=1}^{N} \sum_{s \in Z_t} \theta_{ik} \cdot R \cdot n_{ks}^{i}}. \tag{23}$$

Restricted standard errors

The score vectors change accordingly. The score contribution of individual i is the sum over all states $s \in Z_t$ whit parameters are affected by restriction t. The contribution of individual i to the score of the restricted choice probability $\partial \ln L / \partial \pi_{tr}^{*}$ is

$$s(Y_i, \pi_{tr}^{*}) = \sum_{k=1}^{K} \theta_{ik} \sum_{s \in Z_t} \left(y_{ksr}^{i} - n_{ks}^{i} \pi_{ksr} \right). \tag{24}$$

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

$$\frac{\partial g(\pi_{tr}^{*})}{\partial \pi_{uq}^{*}} = \begin{cases} -\pi_{tr}\pi_{uq} & \text{if } t = u \text{ and } r \neq q \\ \pi_t(1-\pi_u) & \text{if } t = u \text{ and } r = q \\ 0 & \textit{otherwise.} \end{cases} \tag{25}$$

The contribution of individual i to the score of the restricted tremble probability $\partial \ln L / \partial \gamma_t$ of individual i is

$$s(Y_i, \gamma_t) = \sum_{k=1}^{K} \sum_{s \in Z_t} \theta_{ik} \sum_{r=1}^{R} \frac{y_{ksr}^i}{\pi_{ksr}} \left(\frac{1 - \xi_{ksr}}{R - 1} - \xi_{ksr} \right) \tag{26}$$

5.3. Parameter selection

The number of choice parameters π and γ can be selected with the argument `select` of the estimation function `stratEst.model()`. The options `"probs"` and `"trembles"` select the number of choice parameters π, and γ respectively. The selection is performed based on one of the three information criteria outlined in Section ??. The argument which identifies the information criterion is `crit`. Options are `"aic"`, `"bic"` or `"icl"`.

The arguments `r.probs` and `r.trembles` control which combinations of parameter vectors can be reduced to a single parameter vector. The option `"strategies"` defines that the parameter vectors within each strategy are selected. The option `"states"` defines that the parameter vectors within each state across strategies are selected. The option `"global"` defines that all parameter vectors are selected.

The selection procedure starts by estimating the unrestricted model. For every pairwise combination of parameters vectors of the same parameter class, a model is estimated where two vectors of parameters are reduced to a single vector. The lowest value of the information criterion of these models is compared to the value of the information criterion of the unrestricted model. If the model with the reduced number of parameters has a better fit according to the selection criterion, it is the new best model. The procedure continues as long as the reduction of any feasible combination of two parameters vectors improves the fit of the model.

5.4. Strategy selection

The number of strategies K is selected with the option `select = "strategies"`. The selection is performed based on one of the three information criteria outlined in Section ??. The argument that identifies the information criterion is `crit`. Options are `"aic"`, `"bic"` or `"icl"`.

The selection procedure starts by estimating the complete model with K strategies. Next, the K nested models with $K - 1$ strategies are estimated. The K nested models are obtained by excluding one strategy from the set of candidate strategies. The best value of the information criterion of the K nested models is compared to the value of the complete model with K strategies. If the value of the nested model is lower, this is the new best model. The selection procedure is repeated as long as the the exclusion of one strategy improves the fit of the model.

6. Simulated data

The simulation function of the package is `stratEst.simulate()`. The function can be used to generate data on the basis of a fully specified model. A fully specified model can be obtained defining each parameter of the model by hand or by fitting the model to some data.

The simulation function can be used to validate the parameter estimates and standard errors returned by the estimation function. Consider a model with two strategies for the choices left and right:

```
R> set.seed(1)
R> lr <- c("left","right")
R> mixed <- stratEst.strategy(choices = lr, inputs = lr, num.states = 1)
R> pure <- stratEst.strategy(choices = lr, inputs = lr,
+                            prob.choices = c(1,0,0,1),
+                            tr.inputs = c(1,2,1,2))
R> strategies <- list("mixed" = mixed, "pure" = pure)
```

Strategy `mixed` plays left with a mixed probability π drawn from $U(0,1)$. Strategy `pure` plays left if the input is left, and right if the input is right with tremble probability γ from $U(0,0.25)$. The strategy shares are the result of regression coefficient β drawn from $N(0,1)$.

```
R> pi <- runif(1)
R> gamma <- runif(1)/4
R> beta <- rnorm(1)
```

The value of β defines the share of the mixed strategy p. The parameters π and γ are inserted into the strategies.

```
R> p <- 1/sum(1 + exp(beta))
R> sim.shares <- c(p, 1-p)
R> mixed$prob.left <- pi
R> mixed$prob.right <- 1 - pi
R> pure$tremble <- gamma
R> sim.strategies <- list("mixed" = mixed, "pure" = pure)
```

Now, the model is fully specified and can be used to simulate a data set.

```
R> sim.data <- stratEst.simulate(strategies = sim.strategies,
+                                shares = sim.shares, num.ids = 100,
+                                num.games = 10, num.periods = rep(5,10))
```

The function call creates a `stratEst.data` object which contains the observations of 100 individuals. Each individual is assigned to one of the strategies with probabilities `sim.shares`. Each individual plays ten games. The data contains the observations of five periods of each individual per game. In each period, the individual observes an input randomly drawn from the set of inputs considered by the stratgies. The input triggers the state transition of the strategy and the indvidual makes a choice in line with the probabilities defined by the new state.

Two models are estimated. One model without covariates, and one model with an intercept as covariate.

```
R> model <- stratEst.model(data = sim.data, strategies = strategies)
R> sim.data$intercept <- rep(1,nrow(sim.data))
R> model.lcr <- stratEst.model(data = sim.data, strategies = strategies,
+                              covariates = "intercept")
```

The estimated parameters differ from the true parameters because of sampling error. The function `stratEst.test()` can be used to test if the estimated parameters differ from the true parameters.

```
R> pars <- c(p, 1-p, pi, 1-pi, gamma)
R> test.pars <- stratEst.test(model, values = pars)
R> print(test.pars)
```

```
               estimate    diff std.error t-value df Pr(>|t|)
shares.par.1     0.4300 -0.0242    0.0495 -0.4892 97   0.6258
shares.par.2     0.5700  0.0242    0.0495  0.4892 97   0.6258
probs.par.1      0.2716  0.0061    0.0094  0.6535 97   0.5150
probs.par.2      0.7284 -0.0061    0.0094 -0.6535 97   0.5150
trembles.par.1   0.0923 -0.0008    0.0057 -0.1327 97   0.8947
```

The simulation function generates a variable `strategy` that contains the result of the probabilistic assignment of individuals to strategies. This variable can be used to verify that the estimated model parameters returned by the estimation function are the maximum likelihood parameters of the sample.

```
R> strategy <- sim.data$strategy
R> choice <- sim.data$choice
R> input <- sim.data$input
R> p.ml <- mean(strategy == "mixed")
R> pi.ml <- mean(choice[strategy == "mixed"] == "left")
R> gamma.ml <- mean( choice[strategy == "pure"] != input[strategy == "pure"])
R> print(round(c(p.ml, 1 - p.ml, pi.ml, 1 - pi.ml, gamma.ml), 4))
```

```
[1] 0.4300 0.5700 0.2716 0.7284 0.0923
```

Table 1 summarizes the estimation results obtained by repeating the simulation example 10.000 times. The first three rows depict the results for the parameters of the model without the covariate. The last three rows depict the results for the parameters of the model with the covariate. The columns show the means of the estimated parameters, the difference and absolute difference of the estimated and the maximum likelihood parameters, and the rejection probability of a t test for the model parameters.

The first column shows that the means of the estimated parameters are close to the means of the distributions the parameters are sampled from. Columns two and three show that the estimation algorithm generally coverges to the maximum likelihood parameters of the sample. Columns four and five show that the rejection rate of t tests of the model parameters is close to the 5 percent alpha level for both analytic and bootstrapped standard errors.

				$\mathsf{P}(>\lvert t\rvert) < 0.05$	
θ	$\hat{\theta}_m$	$\theta^*_m - \hat{\theta}_m$	$\lvert\theta^*_m - \hat{\theta}_m\rvert$	analytic	bootstrap
model without covariates					
p	0.4966	6e-05	0.0012	0.0537	0.0551
π	0.5002	2e-06	0.0002	0.0439	0.0558
γ	0.1265	-3e-05	0.0004	0.0497	0.0613
model with covariates					
β	-0.0012	-4e-02	1.6639	0.0442	0.0468
π	0.5002	2e-06	0.0002	0.0439	0.0553
γ	0.1265	-3e-05	0.0004	0.0497	0.0612

Table 1: Estimates and rejection probability for simulated data. Average estimates and rejection probability across 10.000 Monte Carlo samples of simulated data. Each sample contains the choices of 100 individuals in 10 games with 5 periods. θ_m is the maximum likelihood estimate of the parameter in sample m. $\hat{\theta}_m$ is the parameter estimate returned by the estimation function for sample m. Bootstrapped standard errors are based on 100 samples.

7. Examples

7.1. Dal Bo and Frechette, 2011

This example illustrates how to replicate the strategy estimation results of the seminal strategy estimation study by Dal Bó and Fréchette (2011). The study 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 δ of the repeated game.

The stage-game parameters are depicted in Figure 1 where the parameter R is either 32, 40 or 48. For each value of R two treatments exist with δ of 1/2 or 3/4 resulting in 2 times three between subject design with six treatments overall. Dal Bó and Fréchette (2011) report the

	C	D
C	R,R	12,50
D	50,12	25,25

Figure 1: Stage game of Dal Bó and Fréchette (2011)

results of treatment-wise strategy frequency estimation for six candidate strategies: 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 the elemenets of the list `strategies.DF2011`. The Tit-For-Tat strategy looks like this:

```
R> print(strategies.DF2011$TFT)

  prob.d prob.c tremble tr(cc) tr(cd) tr(dc) tr(dd)
1      0      1      NA      1      2      1      2
2      1      0      NA      1      2      1      2
```

The strategy TFT chooses between the alternatives defect (`d`) and cooperate (`c`). State transitions are triggered by four different inputs: The four inputs reflect the combination of actions in the last period. The first letter represents the own action in the last period, and the second letter the action of the other player. All strategies in the list `strategies.DF2011` have the same structure of choices and inputs.

The data frame `DF2011` contains the experimental data collected by Dal Bó and Fréchette (2011). The data can be inspected in the console with the command `print(DF2011)`.

The following code creates a `stratEst.data` frame which fits the structure of strategies:

```
R> data.DF2011 <- stratEst.data(data = DF2011, choice = "choice",
+                               input = c("choice","other.choice"),
+                               input.lag = 1)
```

The options `input = c("choice","other.choice")` and `input.lag = 1` create the input variable by concatenating the own and the other player's choice of the previous period. The following model estimation commmand can be used to replicate the findings of Dal Bó and Fréchette (2011):

```
R> model.DF2011 <- stratEst.model(data = data.DF2011,
+                                 strategies = strategies.DF2011,
+                                 sample.id = "treatment" )
```

The command estimates one vector of shares and one tremble parameter for each treatment. The estimated shares are the strategy shares reported in the first column of Table 7 on page 424 of Dal Bó and Fréchette (2011).

```
R> print(round(do.call(rbind, model.DF2011$shares), 2))

                 ALLD ALLC GRIM  TFT WSLS   T2
treatment.D5R32  0.92 0.00 0.00 0.08 0.00 0.00
treatment.D5R40  0.78 0.08 0.04 0.10 0.00 0.00
treatment.D5R48  0.53 0.07 0.00 0.38 0.02 0.00
treatment.D75R32 0.65 0.00 0.00 0.35 0.00 0.00
treatment.D75R40 0.11 0.30 0.27 0.33 0.00 0.00
treatment.D75R48 0.00 0.08 0.12 0.56 0.00 0.24
```

7.2. Fudenberg, Rand and Dreber, 2011

Fudenberg *et al.* (2012) conduct a prisoner's dilemma experiment in which intended choices are implemented with noise. The stage-game payoffs are such that cooperation means paying a cost c to give a benefit b to the other player. The authors run four between subjects treatments. The cost c is fixed at 2 points experimental currency in every treatment. The benefit to cost ratio b/c varies across treatments and took the values 1.5, 2, 2.5, and 4.

Because of the noisy implementation of choices, Fudenberg and colleagues add several lenient and forgiving strategies to the original set of candidate strategies used by Dal Bó and Fréchette (2011). The augmented set of strategies is available as list object `strategies.FRD2012`. The choices of the strategies are `d` (defect), and `c` (cooperate). The four inputs reflect the four different combinations of the own choice, and the choice of the other player in the previous period.

The data frame `FRD2012` contains the raw data of the experiment. It contains two variables that indicate own choice and the choice of the other player in the last period. These two variables are passed to the argument `inputs` of the data generation function:

```
R> data.FRD2012 <- stratEst.data(data = FRD2012, choice = "choice",
+                                  input =c("last.choice","last.other"))
```

The following code replicates the strategy shares reported by Fudenberg *et al.* (2012) in Table 3 on page 733 of the paper.

```
R> model.FRD2012 <- stratEst.model(data = data.FRD2012,
+                                    strategies = strategies.FRD2012,
+                                    sample.id = "bc")
R> print(round(do.call(rbind, model.FRD2012$shares), 2))
```

```
       ALLC  TFT TF2T TF3T T2FT T2F2T GRIM GRIM2 GRIM3 ALLD DTFT
bc.1.5 0.00 0.19 0.05 0.01 0.06  0.00 0.14  0.06  0.06 0.29 0.14
bc.2   0.03 0.06 0.00 0.03 0.07  0.11 0.07  0.18  0.28 0.17 0.00
bc.2.5 0.00 0.09 0.17 0.05 0.02  0.11 0.11  0.02  0.24 0.14 0.05
bc.4   0.07 0.09 0.18 0.13 0.05  0.09 0.06  0.07  0.10 0.14 0.03
```

For the data the treatment $b/c = 4$, the estimation function finds a better solution with a larger log likelihood than the solution reported by Fudenberg *et al.* (2012).

7.3. Dvorak, Fischbacher and Schmelz, 2020

Dvorak, Fischbacher, and Schmelz (2020) study conformity and anticonformity in a binary choice experiment. Participants are matched in groups of three and compete for a monetary reward with the other two group members. In some choices, one group member is informed about the choices of two other group members before making the own choice. For these choices, the experimental design allows to predict the prefered alternative of the participant.

Dvorak *et al.* (2020) find that two-thirds of the participants follow a conformist strategy. The conformist strategy generally follows the own preference if the choices of the other group members are in line with the own preference. It frequently deviates from the own preference and chooses the other alternative if the choices of the other group members are not in line with the own preference.

The remaining one-third of the participants follows an anticonformist strategy. The anticonformist strategy generally follows the own preference if the choices of the other group members are not in line with the own preference. It frequently deviates from the own preference the choices of the other group members are in line with the own preference.

The fitted choice parameters of the strategies are:

```
R> print(strategies.DFS2020)

$anticonformist
  prob.follow prob.deviate tr(not in line) tr(in line)
1       0.823        0.177               1           2
2       0.404        0.596               1           2

$conformist
  prob.follow prob.deviate tr(not in line) tr(in line)
1       0.425        0.575               1           2
2       0.860        0.140               1           2
```

The data set `DFS2020` contains the experimental data of Dvorak *et al.* (2020). The variables `"choice"` indicates if the choice of the participant follows the own preference or deviates from the own preference. The variable `"others.choices"` indicates if the choices of the two other two group members are in line with the preference of the participant or not.

The data set additionally contains two the variables which are used as covariates of the strategy estimation model by Dvorak *et al.* (2020). The first is an intercept, which is one for every observation. The second is the score of the participant in a post-experimental conformity questionnaire (Mehrabian and Stefl 1995). The mean conformity score is -0.078 with a standard deviation of 1.02.

The following command creates a `stratEst.data` object with the variable `others.choices` as input:

```
R> data.DFS2020 <- stratEst.data(data = DFS2020,
+                               input = c("others.choices"))
```

The model with covariates is estimated with the command:

```
R> model.DFS2020 <- stratEst.model(data = data.DFS2020 ,
+                                 strategies = strategies.DFS2020,
+                                 covariates = c("intercept",
+                                                "conformity.score"))
```

The estimated coefficients are:

```
R> print(model.DFS2020$coefficients)

                 anticonformist conformist
intercept                     0  0.8273618
conformity.score              0  0.8697285
```

The first strategy is the reference category of the structural model. The coefficients for the reference category are always zero. The second column contains the estimated coefficients for the conformist strategy. The estimated coefficients indicate that prior probability to use the conformist strategy increases with the score in the conformity questionnaire. The individual prior probabilities of the participants are returned as object `model.DFS2020$prior.assignment`.

The estimated coefficient of the intercept can be used to calculate the estimated prior probability to use the conformist for a participant with a conformity score of zero. The prior probability is $exp(0.83)/(1 + exp(0.83)) = 0.69$. A participant who scores on standard deviation higher than average in the conformity questionnaire has a prior probability of $exp(0.83 + 0.87)/(1 + exp(0.83 + 0.87)) = 0.85$ to use the conformist strategy.

The function `stratEst.test()` can be used to test whether the estimated coefficients differ from zero.

```
R> test.coefficients <- stratEst.test(model.DFS2020, par = "coefficients")
R> print(test.coefficients)
```

```
                   estimate std.error t-value  df Pr(>|t|)
coefficients.par.1   0.8274    0.3065  2.6997 103   0.0081
coefficients.par.2   0.8697    0.3184  2.7319 103   0.0074
```

The function `stratEst.check()` can be used to assess the global and local model fit based on the Pearson χ^2 test statistic.

```
R> check.DFS2020 <- stratEst.check(model.DFS2020, chi.tests = TRUE,
+                                  bs.samples = 100)
R> print(check.DFS2020$chi.global)
```

```
                   chi^2        min     mean      max p.value
model.DFS2020 0.08554165 0.07108578 2.623929 8.065177    0.99
```

```
R> print(check.DFS2020$chi.local)
```

```
                   chi^2      min      mean       max p.value
anticonformist  52.29308 17.60894  47.81437  79.03572    0.38
conformist     117.70654 58.92359 109.50872 155.70476    0.31
```

The distribution of the test statistics is approximated by drawing 100 bootstrap samples to limit computation time. The p value of the global test indicates the probability of the data given that the estimated model is the true model. The p value of the local test for the anticonformist strategy indicates the probability of the data of the subset of participants classified as anticonformist given that the fitted strategy is the true strategy. The p value of the test for the conformist strategy can be interpreted in the same way. Hence, both tests address the null hypothesis that the model is the true data generating model.

8. Function documentation

8.1. stratEst.strategy()

The strategy generation function of the package. The syntax of the function call is:

```
R> stratEst.strategy(choices, inputs = NULL, prob.choices = NULL,
+                     tr.inputs = NULL, trembles = NULL, num.states = 1)
```

Inputs

`choices:`	a character vector. The levels of the factor `choice` in the data.
`inputs:`	a character vector. The levels of the factor `input` in the data.
`prob.choices:`	a numeric vector. The choice probabilities of the strategy in row wise order.
`tr.inputs:`	a vector of integers. The deterministic state transitions of the strategy in row wise order.
`trembles:`	a numeric vector. The tremble probabilities of the strategy.
`num.states:`	an integer. The number states of the strategy.

Outputs

A `stratEst.strategy` object. A data frame with the following variables:

`prob.x`	the probability of choice `x`.
`tremble:`	the probability to observe a tremble.
`tr(x):`	the deterministic state transitions of the strategy for input `x`.

8.2. stratEst.data()

The data generation function of the package. The syntax of the function call is:

```
R> stratEst.data(data, choice = "choice", id = "id", input, input.lag = 0,
+                input.sep = "", game = "game", period = "period",
+                add = NULL, drop = NULL)
```

Input

`data:` a `data.frame` in the long format.

`choice:` a character string. The variable in `data` which contains the discrete choices. Default is `"choice"`.

`input:` a character string. The names of the input generating variables in `data`. At least one input generating variable has to be specified. Default is `c("input")`.

`input.lag:` a numeric vector. The time lag in periods of the input generating variables. The vector must have as many elements as variables specified in the object `input`. Default is zero.

`input.sep:` a character string. Separates the input generating variables. Default is `""`.

`id:` a character string. The name of the variable in `data` that identifies observations of the same individual. Default is `"id"`.

`game:` a character string. The name of the variable in `data` that identifies observations of the same game. Default is `"game"`.

`period:` a character string. The name of the variable in `data` that identifies the periods of a game. Default is `"period"`.

`add:` a character vector. The names of variables in the global environment that should be added to the `stratEst.data` object. Default is `NULL`.

`drop:` a character vector. The names of variables in `data` that should be dropped. Default is `NULL`.

Output

A `stratEst.data` object. A data frame in the long format with the following variables:

`id:` the variable that identifies observations of the same individual.

`game:` the variable that identifies observations of the same game.

`period:` the period of the game.

`choice:` the discrete choices.

`input:` the inputs.

8.3. stratEst.model()

The estimation function of the package. The syntax of the function call is:

```
R> stratEst.model(data, strategies, shares = NULL, coefficients = NULL,
+                 covariates = NULL, sample.id = NULL, response = "mixed",
+                 sample.specific = c("shares","probs","trembles"),
+                 r.probs = "no", r.trembles = "global", select = NULL,
+                 outer.tol = 1e-10, outer.max = 1000, inner.runs = 10,
+                 inner.tol = 1e-5, inner.max = 10, lcr.runs = 100,
+                 lcr.tol = 1e-10, lcr.max = 1000 , bs.samples = 1000,
+                 quantiles = c(0.025,0.5,0.975), stepsize = 1,
+                 penalty = FALSE, verbose = TRUE)
```

Input

`data:` a `stratEst.data` object or `data.frame`.

`strategies:` a list of strategies. Each element if the list must be an object of class `stratEst.strategy`.

`shares:` a numeric vector of strategy shares. The order of the elements corresponds to the order in `strategies`. Elements which are `NA` are estimated from the data. Use a list of numeric vectors if data has more than one sample and shares are sample specific.

`coefficients:` a matrix of latent class regression coefficients.

`covariates:` a character vector with the names of the covariates of the latent class regression model in the data. The covariates cannot have missing values.

`sample.id:` a character string indicating the name of the variable which identifies the samples in data. Individual observations must be nested in samples.

`response:` a character string which is either `"pure"` or `"mixed"`. If `"pure"` the estimated choice probabilities are either zero or one. If `"mixed"` the estimated choice probabilities are mixed parameters. The default is `"mixed"`.

`sample.specific:` a character vector, Defines the model parameters that are sample specific. Can contain the character strings `"shares"` (`"probs"`, `"trembles"`. If the vector contains `"shares"` (`"probs"`, `"trembles"`), the estimation function estimates a set of shares (choice probabilities, trembles) for each sample in the data.

`r.probs:` a character string. Options are `"no"`, `"strategies"`, `"states"` or `"global"`. Option `"no"` yields one vector of choice probabilities per strategy and state. Option `"strategies"` yields one vector of choice probabilities per strategy. Option `"states"` yields one vector of choice probabilities per state. Option `"global"` yields a single vector of choice probabilities. Default is `"no"`.

`r.trembles:` a character string. Options are `"no"`, `"strategies"`, `"states"` or `"global"`. Option `"no"` yields one tremble probability per strategy and state. Option `"strategies"` yields one tremble probability per strategy. Option `"states"` yields one tremble probability per state. Option `"global"` yields a single tremble probability. Default is `"no"`.

`select:` a character vector. Indicates the classes of model parameters that are selected. Can contain the strings `"strategies"`, `"probs"`, and `"trembles"`. If the vector contains`"strategies"` (`"probs"`, `"trembles"`), the number of strategies (choice probabilities, trembles) is selected based on the selection criterion specified in `"crit"`. The selection can be restricted with the arguments `r.probs` and `r.trembles`. Default is `NULL`.

`min.strategies:` an integer. The minimum number of strategies in case of strategy selection. The strategy selection procedure stops if the minimum is reached.

`crit:` a character string. Defines the information criterion used for model selection. Options are `"bic"` (Bayesian information criterion), `"aic"` (Akaike information criterion) or `"icl"` (Integrated Classification Likelihood). Default is `"bic"`.

`se:` a string. Defines how standard errors are obtained. Options are `"analytic"` or `"bootstrap"`. Default is `"analytic"`.

`outer.runs:` an integer. The number of outer runs of the solver. Default is 1.

`outer.tol:` a number close to zero. The tolerance of the stopping condition of the outer runs. The iterative algorithm stops if the relative decrease of the log likelihood is smaller than this number. Default is 1e-10.

`outer.max:` an integer. The maximum number of iterations of the outer runs of the solver. The iterative algorithm stops after `"outer.max"` iterations if it does not converge. Default is 1000.

`inner.runs:` an integer. The number of inner runs of the solver. Default is 10.

`inner.tol:` a number close to zero. The tolerance of the stopping condition of the inner runs. The iterative algorithm stops if the relative decrease of the log likelihood is smaller than this number. Default is 1e-5.

`inner.max:` an integer. The maximum number of iterations of the outer runs of the solver. The iterative algorithm stops after `"inner.max"` iterations if it does not converge. Default is 10.

`lcr.runs`: an integer. The number of latent class regression runs of the solver. Default is 100.

`lcr.tol`: a number close to zero. The tolerance of the stopping condition of the latent class regression runs. The iterative algorithm stops if the relative decrease of the log likelihood is smaller than this number. Default is 1e-10.

`lcr.max`: an integer. The maximum number of iterations of the latent class regression runs of the solver. The iterative algorithm stops after `"lcr.max"` iterations if it does not converge. Default is 1000.

`bs.samples`: an integer. The number of bootstrap samples.

`quantiles`: a numeric vector. The quantiles of the sampling distribution of the estimated parameters. Depending on the option of `se`, the quantiles are either estimated based on a t-distribution with `res.degrees` of freedom and the analytic standard errors or based the bootstrap.

`step.size`: a number between zero and one. The step size of the Fisher scoring step which updates the coefficients. Values smaller than one slow down the convergence of the algorithm and prevent overshooting. Default is one.

`penalty`: a logical. If `TRUE` the Firth penalty is used to estimate the coefficients of the latent class regression model. Default is `FALSE`.

`verbose`: a logical. If `TRUE` information about the estimation process are printed to the console. Default is `FALSE`.

Output

An object of class `stratEst`. A list with the following elements.

`strategies`: the fitted strategies.

`shares`: the strategy shares.

`probs`: the choice probabilities of the strategies.

`trembles`: the tremble probabilities of the strategies.

`gammas`: the gamma parameters of the strategies.

`coefficients`: the coefficients of the covariates.

`shares.par`: the estimated strategy share parameters.

`probs.par`: the estimated choice probability parameters.

`trembles.par:`	the estimated tremble parameters.
`gammas.par:`	the estimated gamma parameters.
`coefficients.par:`	the estimated coefficient parameters of the covariates.
`shares.indices:`	the parameter indices of the strategy shares.
`probs.indices:`	the parameter indices of the choice probabilities.
`trembles.indices:`	the parameter indices of the tremble probabilities.
`coefficients.indices:`	the parameter indices of the coefficients.
`loglike:`	the log likelihood of the model.
`num.ids:`	the number of individuals.
`num.obs:`	the number of observations.
`num.par:`	the total number of model parameters.
`free.par:`	the number of free model parameters.
`res.degrees:`	the residual degrees of freedom.
`aic:`	the Akaike information criterion.
`bic:`	the Bayesian information criterion.
`icl:`	The integrated classification likelihood.
`crit.val:`	the value of the selection criterion defined by the argument `crit`.
`eval:`	the total number of iterations of the solver.
`tol.val:`	the relative decrease of the log likelihood in the last iteration of the algorithm.
`convergence:`	the maximum of the absolute scores of the estimated model parameters.
`entropy.model:`	the entropy of the model.
`entropy.assignments:`	the entropy of the posterior probability assignments of individuals to strategies.
`chi.global:`	the chi square statistic for global model fit.
`chi.local:`	the chi square statistics for local model fit.
`state.obs:`	the weighted observations for each strategy state.
`post.assignments:`	the posterior probability assignments of individuals to strategies.

`prior.assignments`: the prior probability of each individual to use a strategy as predicted by the individual covariates.

`shares.se`: the standard errors of the estimated share parameters.

`probs.se`: the standard errors of the estimated choice probability parameters.

`trembles.se`: the standard errors of the estimated tremble probability parameters.

`gammas.se`: the standard errors of the estimated gamma parameters.

`coefficients.se`: the standard errors of the estimated coefficients.

`shares.quantiles`: the quantiles of the estimated population shares.

`probs.quantiles`: the quantiles of the estimated choice probabilities.

`trembles.quantiles`: the quantiles of the estimated trembles.

`coefficients.quantiles`: the quantiles of the estimated coefficients.

`shares.score`: the scores of the estimated share parameters.

`probs.score`: the score of the estimated choice probabilities.

`trembles.score`: the score of the estimated tremble probabilities.

`coefficients.score`: the score of the estimated coefficients.

`shares.fisher`: the Fisher information matrix of the estimated shares.

`probs.fisher`: the Fisher information matrix of the estimated choice probabilities.

`trembles.fisher`: the Fisher information matrix of the estimated trembles.

`coefficients.fisher`: the fisher information matrix of the estimated coefficients.

`fit.args`: the input objects of the function call.

8.4. stratEst.simulate()

The simulation function of the package. The syntax of the function call is:

```
R> stratEst.simulate( data = NULL, strategies, shares = NULL,
+                     coefficients = NULL, covariates = NULL,
+                     num.ids = 100, num.games = 5, num.periods = NULL,
+                     fixed.assignment = FALSE, input.na = FALSE)
```

Input

`data:` a `stratEst.data` object. Alternatively, the arguments `num.ids`, `num.games`, and `num.periods` can be used if no data is available.

`strategies:` a list of strategies. Each element if the list must be an object of class `stratEst.strategy`.

`shares:` a numeric vector of strategy shares. The order of the elements corresponds to the order in `strategies`. `NA` values are not allowed. Use a list of numeric vectors if data has more than one sample and shares are sample specific.

`coefficients:` a matrix of regression coefficients. Column names correspond to the names of the strategies, row names to the names of the covariates.

`covariate.mat:` a matrix with the covariates in columns. The column names of the matrix indicate the names of the covariates. The matrix must have as many rows as there are individuals.

`num.ids:` an integer. The number of individuals. Default is 100.

`num.games:` an integer. The number of games. Default is 5.

`num.periods:` a vector of integers with as many elements `num.games`. The elements specify the number of periods in each game. Default (`NULL`) means 5 periods in each game.

`fixed.assignment:` a logical value. If `FALSE` individuals use potentially different strategies in each each game. If `TRUE`, individuals use the same strategy in each game. Default is `FALSE`.

`input.na:` a logical value. If `FALSE` an input value is randomly selected for the first period. Default is `FALSE`.

`sample.id:` a character string indicating the name of the variable which identifies the samples in data. Individual observations must be nested in samples. Default is `NULL`.

Output

A `stratEst.data` object. A data frame in the long format with the following variables:

`id:` the variable that identifies observations of the same individual.

`game:` the variable that identifies observations of the same game.

`period:` the period of the game.

`choice:` the discrete choices.

`input:` the inputs.

`sample:` the sample of the individual.

`strategy:` the strategy of the individual.

8.5. stratEst.test()

The test function of the package. The syntax of the function call is:

```
R> stratEst.test(model, par = c("shares", "probs", "trembles",
+                                "coefficients"), values = 0,
+                 alternative = "two.sided", digits = 4)
```

Input

`model:` a fitted model of class `stratEst.model`.

`par:` a character vector. The class of model parameters to be tested. The default is to test all classes of model parameters.

`values:` a numeric vector. The values the parameter estimates are compared to. Default is zero.

`alternative:` a character string. The alternative hypothesis. Options are `"two.sided"`, `"greater"` or `"less"`. Default is `"two.sided"`.

`digits:` an integer. The number of digits of the result.

Output

A `data.frame` with one row for each tested parameter and 6 variables:

`estimate:` the parameter estimate.

`diff:` the difference between the estimated parameter and the numeric value (if supplied).

`std.error:` the standard error of the estimated parameter.

`t.value:` the t statistic.

`res.degrees:` the residual degrees of freedom of the model.

`p.value:` the p value of the t statistic.

8.6. stratEst.check()

The function for model checking of the package. The syntax of the function call is:

```
R> stratEst.check( model, chi.tests = F, bs.samples = 100, verbose = FALSE )
```

Input

`model:`	a fitted model of class `stratEst.model`.
`chi.tests:`	a logical. If `TRUE` chi square tests of global and local model fit are performed. Default is `FALSE`.
`bs.samples:`	an integer. The number of parametric bootstrap samples for the chi square tests. Default is 100.
`verbose:`	a logical, if `TRUE` messages of the checking process are printed to the console. Default is `FALSE`.

Output

A list of check results with the following elements:

`fit:`	a matrix. Contains the log likelihood, the number of free model parameters, and the value of the three information criteria.
`chi.global:`	a matrix. The results of the chi square test for global model fit.
`chi.local:`	a matrix. The results of the chi square test for local model fit.

Computational details

The results in this paper were obtained using R 4.0.0 with the **stratEst** 1.0.0 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/`.

Acknowledgments

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.

References

Agresti A (2003). *Logit Models for Multinomial Responses*, chapter 7, pp. 267–313. Wiley-Blackwell. ISBN 9780471249689. `doi:10.1002/0471249688.ch7`. `https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471249688.ch7`, URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/0471249688.ch7`.

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

Aoyagi M, Bhaskar V, Frechette GR (2019). "The Impact of Monitoring in Infinitely Repeated Games: Perfect, Public, and Private." *American Economic Journal: Microeconomics*, **11**(1), 1–43. doi:10.1257/mic.20160304. URL https://www.aeaweb.org/articles?id=10.1257/mic.20160304.

Arechar AA, Dreber A, Fudenberg D, Rand DG (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. ISSN 10902473. doi:10.1016/j.geb.2017.06.013. URL http://dx.doi.org/10.1016/j.geb.2017.06.013.

Bandeen-Roche K, Miglioretti DL, Zeger SL, Rathouz PJ (1997). "Latent Variable Regression for Multiple Discrete Outcomes." *Journal of the American Statistical Association*, **92**(440), 1375–1386. doi:10.1080/01621459.1997.10473658. https://doi.org/10.1080/01621459.1997.10473658, URL https://doi.org/10.1080/01621459.1997.10473658.

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

Biernacki C, Celeux G, Govaert G (2000). "Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**, 719–725.

Biernacki C, Celeux G, Govaert G (2003). "Choosing Starting Values for the {EM} Algorithm for Getting the Highest Likelihood in Multivariate Gaussian Mixture Models." *Computational Statistics & Data Analysis*, **41**(3âĂŞ4), 561–575. ISSN 0167-9473. doi:http://dx.doi.org/10.1016/S0167-9473(02)00163-9. URL http://www.sciencedirect.com/science/article/pii/S0167947302001639.

Bolck A, Croon M, Hagenaars J (2004). "Estimating Latent Structure Models with Categorical Variables: One-Step Versus Three-Step Estimators." *Political Analysis*, **12**(1), 3–27. doi:10.1093/pan/mph001.

Breitmoser Y (2015). "Cooperation, but no Reciprocity: Individual Strategies in the Repeated Prisoner's Dilemma." *American Economic Review*, **105**(9), 2882–2910. ISSN 00028282. doi:10.1257/aer.20130675.

Bull SB, Mak C, Greenwood CM (2002). "A Modified Score Function Estimator for Multinomial Logistic Regression in Small Samples." *Computational Statistics & Data Analysis*, **39**(1), 57 – 74. ISSN 0167-9473. doi:https://doi.org/10.1016/S0167-9473(01)00048-2. URL http://www.sciencedirect.com/science/article/pii/S0167947301000482.

Camera G, Casari M, Bigoni M (2012). "Cooperative Strategies in Anonymous Economies: An Experiment." *Games and Economic Behavior*, **75**(2), 570–586. ISSN 08998256. doi:10.1016/j.geb.2012.02.009.

Dal Bó P, Fréchette GR (2011). "The Evolution of Cooperation in Infinitely Repeated Games: Experimental Evidence." *American Economic Review*, **101**(1), 411–429.

Dayton CM, Macready GB (1988). "Concomitant-Variable Latent-Class Models." *Journal of the American Statistical Association*, **83**(401), 173–178.

Dempster A, Laird N, Rubin DB (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society Series B*, **39**(1), 1–38. ISSN 00359246. doi:http://dx.doi.org/10.2307/2984875. 0710.5696v2, URL http://www.jstor.org/stable/10.2307/2984875.

Dvorak F, Fehrler S (2018). "Negotiating Cooperation under Uncertainty: Communication in Noisy, Indefinitely Repeated Interactions." *IZA Working Paper No. 11897*.

Dvorak F, Fischbacher U, Schmelz K (2020). "Incentives for Conformity and Anticonformity." *Technical report*, TWI Working Paper Series.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08. URL http://www.jstatsoft.org/v40/i08/.

Efron B, Tibshirani RJ (1993). *An Introduction to the Bootstrap*. Chapman & Hall/CRC. Boca Raton, FL.

Embrey M, Frechette GR, Stacchetti E (2013). "An Experimental Study of Imperfect Public Monitoring: Efficiency Versus Renegotiation-Proofness." *SSRN Working Paper*. doi:10.2139/ssrn.2346751. URL https://ssrn.com/abstract=2346751.

Embrey M, Frechette GR, Yuksel S (2017). "Cooperation in the Finitely Repeated PrisonerŠs Dilemma." *The Quarterly Journal of Economics*, **133**(1), 509–551. ISSN 0033-5533. doi:10.1093/qje/qjx033. https://academic.oup.com/qje/article-pdf/133/1/509/30636531/qjx033.pdf, URL https://doi.org/10.1093/qje/qjx033.

Firth D (1993). "Bias Reduction of Maximum Likelihood Estimates." *Biometrika*, **80**(1), 27–38. ISSN 00063444. URL http://www.jstor.org/stable/2336755.

Frechette GR, Yuksel S (2017). "Infinitely repeated games in the laboratory: four perspectives on discounting and random termination." *Exp Econ*, **20**(2), 279 – 308. doi:10.1007/s10683-016-9494-z. URL https://doi.org/10.1007/s10683-016-9494-z.

Fudenberg D, Rand DG, Dreber A (2012). "Slow to Anger and Fast to Forgive: Cooperation in an Uncertain World." *American Economic Review*, **102**(2), 720–749. ISSN 0002-8282. doi:10.1257/aer.102.2.720.

Kaufman L, Rousseeuw PJ (1990). *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics. John Wiley & Sons, Inc., New York.

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

Leisch F (2002). *Compstat.*, chapter **Sweave**: Dynamic Generation of Statistical Reports Using Literate Data Analysis. Physica, Heidelberg.

Leisch F (2004). "**FlexMix**: A General Framework for Finite Mixture Models and Latent Class Regression in R." *Journal of Statistical Software*, **11**(8).

Linzer DA, Lewis JB (2011). "**poLCA**: An R Package for Polytomous Variable Latent Class Analysis." *Journal of Statistical Software*, **42**(10).

McLachlan G, Peel D (2005). *Finite Mixture Models.* John Wiley & Sons, Inc., New York. doi:10.1002/0471721182.

Mehrabian A, Stefl CA (1995). "Basic Temperament Components of Loneliness, Shyness, and Conformity." *Social Behavior and Personality*, **23**(3), 253–263. ISSN 0301-2212. doi:doi:10.2224/sbp.1995.23.3.253.

Meilijson I (1989). "A Fast Improvement to the EM Algorithm on its Own Terms." *Journal of the Royal Statistical Society B*, **51**(1), 127–138. ISSN 00359246. URL http://www.jstor.org/stable/2345847.

R Development Core Team (2008). "R: A Language and Environment for Statistical Computing." *Technical report*, R Foundation for Statistical Computing. URL http://www.R-project.org.

Sanderson C, Curtin R (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**(2), 461–464. doi:10.1214/aos/1176344136. URL https://doi.org/10.1214/aos/1176344136.

Selten R (1975). "Reexamination of the Perfectness Concept for Equilibrium Points in Extensive Games." *International Journal of Game Theory*, **4**(1), 25–55. ISSN 1432-1270. doi:10.1007/BF01766400. URL https://doi.org/10.1007/BF01766400.

van Kollenburg GH, Mulder J, Vermunt JK (2015). "Assessing Model Fit in Latent Class Analysis When Asymptotics Do Not Hold." *Methodology*, **11**(2), 65–79. doi:10.1027/1614-2241/a000093. https://doi.org/10.1027/1614-2241/a000093, URL https://doi.org/10.1027/1614-2241/a000093.

Wang Z, Xu B, Zhou HJ (2014). "Social Cycling and Conditional Responses in the Rock-Paper-Scissors Game." *Scientific Reports*, **4**(1), 5830 2045–2322. doi:10.1038/srep05830. URL https://doi.org/10.1038/srep05830.

Wickham H (2011). "**testthat**: Get Started with Testing." *The R Journal*, **3**, 5–10. URL https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.

Wickham H, Danenberg P, Csardi G, Eugster M (2020a). ***roxygen2**: In-Line Documentation for R.* R package version 7.1.0, URL https://CRAN.R-project.org/package=roxygen2.

Wickham H, Hester J, Chang W (2020b). ***devtools**: Tools to Make Developing R Packages Easier.* R package version 2.3.0, URL https://CRAN.R-project.org/package=devtools.

Affiliation:

Fabian Dvorak
Centre for the Advanced Study of Collective Behaviour
and
Chair of Applied Research in Economics
Universität Konstanz
Box 146
D-78457, Konstanz
Germany
E-mail: fabian.dvorak@uni-konstanz.de
URL: http://fabian-dvorak.com/