Package: stratEst (via r-universe)

September 30, 2024

Type Package

Title Strategy Estimation

Version 1.1.6

Author Fabian Dvorak

Maintainer Fabian Dvorak <`fabian.dvorak@uni.kn`>

Date 2022-11-29

Description Variants of strategy estimation (Dal Bo & Frechette, 2011, <doi:10.1257/aer.101.1.411>), including the model with parameters for the choice probabilities of the strategies (Breitmoser, 2015, <doi:10.1257/aer.20130675>), and the model with individual level covariates for the selection of strategies by individuals (Dvorak & Fehrler, 2018, <doi:10.2139/ssrn.2986445>).

Copyright See the file COPYRIGHTS for copyright, authorship and license details

License GPL-3

Encoding UTF-8

LazyData true

LinkingTo Rcpp, RcppArmadillo (>= 0.9.900.0.0)

Imports Rcpp (>= 0.12.18), stats, graphics

RoxygenNote 7.1.1

Suggests DiagrammeR (>= 1.0.6.1), DiagrammeRsvg (>= 0.1), spelling, testthat

Depends R (>= 3.5)

URL `https://github.com/fdvorak/stratEst`

BugReports `https://github.com/fdvorak/stratEst/issues`

NeedsCompilation yes

Language en-US

Repository CRAN

Date/Publication 2022-11-29 21:00:02 UTC

Contents

`data.DF2011` *Data of Dal Bo and Frechette (2011)*

Description

The dataset of Dal Bo and Frechette (DF2011, 2011) as stratEst.data frame.

Usage

```
data(data.DF2011)
```

Format

A data frame with 7358 rows and 7 variables:

treatment A treatment identifier of the experiment.

id Variable which identifies a participant.

game The supergame number.

period The period of the supergame.

choice A factor with two levels which is indicates if the participant cooperates (c) or defects (d) in the current period.

other.choice A factor with two levels which indicates if the other participant cooperates (c) or defects (d) in the current period.

input A factor with four levels which is indicates the action profile in the previous round. The first letter indicates the action of the participant, the second letter the action of the partner in the previous round. In the first round of a game the input is NA.

Source

`https://www.aeaweb.org/articles?id=10.1257/aer.101.1.411`

References

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

`data.DFS2020` *Data of Dvorak, Fischbacher and Schmelz (2020)*

Description

A stratEst.data object with observations from an experiment on social influence.

Usage

```
data(data.DF2011)
```

Format

A data frame with 569 rows and 8 variables:

id A vector of integers which identifies the participant.

game A vector of integers which identifies the game.

period A vector of integers which identifies the period.

others.choices A factor with two levels which indicates if the choices of the other two group members are in line (in line) or not in line (not in line) with the preference of the participant.

choice A factor with two levels which is indicates if the participant deviates (deviate) from or follows (follow) the own preference in the current period.

intercept An intercept. One for every observation.

conformity.score The conformity score of the participant in a post-experimental conformity questionnaire (Mehrabian and Stefl, 1995).

input A factor which indicates the input for the strategies. The variable choice with two levels which is indicates if the participant deviates (deviate) from or follows (follow) the own preference in the current period.

References

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

Mehrabian A, Stefl CA (1995). "Basic Temperament Components of Loneliness, Shyness, and Conformity." *Social Behavior and Personality*, 23(3), 253–263.

`data.FRD2012` *Data of Fudenberg, Rand, and Dreber (2012)*

Description

The dataset of Fudenberg, Rand, and Dreber (2012) as stratEst.data frame.

Usage

```
data(data.FRD2012)
```

Format

A data frame with 13126 rows and 10 variables:

treatment A factor with six levels which identifies the treatments of the experiment.

id A vector of integers which identifies the participant.

game A vector of integers which identifies the supergame.

period A vector of integers which identifies the period of the supergame.

choice A factor with two levels which is indicates if the participant cooperates (c) or defects (d) in the current period.

last.choice A factor with two levels which indicates if the participant cooperated (c) or defected (d) in the previous period.

last.other A factor with two levels which indicates if the other participant cooperated (c) or defected (d) in the previous period.

bc A factor which indicates the benefit to cost ratio of the treatment.

error A factor which indicates the noise level of the treatment.

input A factor with four levels which is indicates the action profile in the previous round. The first letter indicates the action of the participant, the second letter the action of the partner in the previous round. In the first round of a game the input is NA.

Source

https://www.aeaweb.org/articles?id=10.1257/aer.102.2.720

References

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.

`data.WXZ2014` *Data of the rock-paper-scissors game from Wang, Xu, and Zhou (2014)*

Description

A stratEst.data object that contains the data of Wang, Xu, and Zhou (2014).

Usage

```
data(data.WXZ2014)
```

Format

A stratEst.data object with 21.600 rows and 7 variables:

id Variable which identifies the participant.

game The identifier of the game.

period The period within the game.

choice A factor with three levels which indicates if the player chooses rock, paper or scissors.

other_choice A factor with three levels which indicates if the other player chooses rock, paper or scissors.

result A factor with three levels which indicates if the result for the player.

input A factor with three levels which is indicates the action in the previous round. In the first period of a game the input is NA.

References

Wang Z, Xu B, Zhou HJ (2014). "Social Cycling and Conditional Responses in the Rock-Paper-Scissors Game." *Scientific Reports*, 4(1), 2045-2322.

`DF2011` *Data of Dal Bo and Frechette (2011)*

Description

A dataset with observations from the repeated prisoner's dilemma experiment of Dal Bo and Frechette (2011).

Usage

```
data(DF2011)
```

Format

A data frame with 7358 rows and 6 variables:

treatment A factor with six levels which identifies the treatments of the experiment.

id A vector of integers which identifies the participant.

game A vector of integers which identifies the supergame.

period A vector of integers which identifies the period of the supergame.

choice A factor with two levels which is indicates if the participant cooperates (c) or defects (d) in the current period.

other.choice A factor with two levels which indicates if the other participant cooperates (c) or defects (d) in the current period.

Source

`https://www.aeaweb.org/articles?id=10.1257/aer.101.1.411`

References

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

`DFS2020` *Data of Dvorak, Fischbacher and Schmelz (2020)*

Description

A dataset with observations from an experiment on social influence.

Usage

```
data(DF2011)
```

Format

A data frame with 569 rows and 7 variables:

id A vector of integers which identifies the participant.

game A vector of integers which identifies the game.

period A vector of integers which identifies the period.

others.choices A factor with two levels which indicates if the choices of the other two group members are in line (in line) or not in line (not in line) with the preference of the participant.

choice A factor with two levels which is indicates if the participant deviates (deviate) from or follows (follow) the own preference in the current period.

intercept An intercept. One for every observation.

conformity.score The conformity score of the participant in a post-experimental conformity questionnaire (Mehrabian and Stefl, 1995).

References

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

Mehrabian A, Stefl CA (1995). "Basic Temperament Components of Loneliness, Shyness, and Conformity." *Social Behavior and Personality*, 23(3), 253–263.

`FRD2012` *Data of Fudenberg, Rand, and Dreber (2012)*

Description

A dataset with observations from the repeated prisoner's dilemma experiment of Fudenberg, Rand, and Dreber (2012).

Usage

```
data(FRD2012)
```

Format

A data frame with 13126 rows and 9 variables:

treatment A factor with six levels which identifies the treatments of the experiment.

id A vector of integers which identifies the participant.

game A vector of integers which identifies the supergame.

period A vector of integers which identifies the period of the supergame.

choice A factor with two levels which is indicates if the participant cooperates (c) or defects (d) in the current period.

last.choice A factor with two levels which indicates if the participant cooperated (c) or defected (d) in the previous period.

last.other A factor with two levels which indicates if the other participant cooperated (c) or defected (d) in the previous period.

bc A factor which indicates the benefit to cost ratio of the treatment.

error A factor which indicates the noise level of the treatment.

Source

`https://www.aeaweb.org/articles?id=10.1257/aer.102.2.720`

References

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.

`is.stratEst.check` *Class stratEst.check*

Description

Checks if an object is of class `stratEst.check`.

Usage

```
is.stratEst.check(x)
```

Arguments

`x`	object to be tested.

Details

Objects of class `stratEst.check` are returned by the function `stratEst.check()` of package `stratEst`.

Value

`is.stratEst.check` returns `TRUE` if its argument is a `stratEst.check` object (that is, has "stratEst.check" amongst its classes) and `FALSE` otherwise.

`is.stratEst.data` *Class stratEst.data*

Description

Checks if an object is of class `stratEst.data`.

Usage

```
is.stratEst.data(x)
```

Arguments

`x`	object to be tested.

Details

Objects of class `stratEst.data` are returned by the functions `stratEst.data()` and `stratEst.simulate()` of package `stratEst`.

Value

`is.stratEst.data` returns `TRUE` if its argument is a `stratEst.data` object (that is, has "strat-Est.data" amongst its classes) and `FALSE` otherwise.

`is.stratEst.model` *Class stratEst.model*

Description

Checks if an object is of class `stratEst.model`.

Usage

```
is.stratEst.model(x)
```

Arguments

`x`	object to be tested.

Details

Objects of class `stratEst.model` are returned by the estimation function `stratEst.model()` of package `stratEst`.

Value

`is.stratEst.model` returns `TRUE` if its argument is a `stratEst.model` object (that is, has "strat-Est.model" amongst its classes) and `FALSE` otherwise.

`is.stratEst.strategy` *Class stratEst.strategy*

Description

Checks if an object is of class `stratEst.strategy`.

Usage

```
is.stratEst.strategy(x)
```

Arguments

`x`	object to be tested.

Details

Objects of class `stratEst.strategy` is returned by the function `stratEst.strategy()` of package `stratEst`.

Value

`is.stratEst.strategy` returns `TRUE` if its argument is a `stratEst.strategy` object (that is, has "stratEst.strategy" amongst its classes) and `FALSE` otherwise.

`plot.stratEst.strategy` *Plot Method for stratEst.strategy*

Description

Plot Method for stratEst.strategy

Usage

```
## S3 method for class 'stratEst.strategy'
plot(
  x,
  y,
  ...,
  title = NULL,
  show.legend = TRUE,
  show.title = TRUE,
  node.fontsize = 25,
  main.fontsize = 40,
  arrow.fontsize = 20,
  legend.fontsize = 20,
  legend.width = 1,
  node.width = 1,
  arrowsize = 1,
  node.penwidth = 1,
  arrow.penwidth = 1,
  fillcolor = NULL,
  ranksep = 0,
  file = NA
)
```

Arguments

`x`	An object of class `stratEst.strategy`.
`y`	Argument two of the generic function.
`...`	Further arguments passed to or from other methods.
`title`	String. The title of the plot.
`show.legend`	Logical. Hide plot legend if FALSE. Default is TRUE.
`show.title`	Logical. Hide plot title if FALSE. Default is TRUE.
`node.fontsize`	Font-size of the plot labels.
`main.fontsize`	Font-size of the plot title.
`arrow.fontsize`	Font-size of the arrow labels.
`legend.fontsize`	Font-size of the legend.
`legend.width`	Width of the legend items.
`node.width`	Width of the nodes.
`arrowsize`	Size of the arrowhead.
`node.penwidth`	Width of the nodes.
`arrow.penwidth`	Width of the nodes.
`fillcolor`	Vector of hex-color codes of the choices.
`ranksep`	Separation of nodes with the same rank.
`file`	String. A valid path followed by a file name. Should end with .svg. Default is NA and no file is written.

Value

No return value, called to create a plot.

print.stratEst.check *Print Method for stratEst.check*

Description

Print Method for stratEst.check

Usage

```
## S3 method for class 'stratEst.check'
print(x, ...)
```

Arguments

x	An object of class `stratEst.check`.
...	Further arguments passed to or from other methods.

Value

Prints a `matrix` that contains the log-likelihood, the number of free model parameters, and the values of three information criteria in columns.

print.stratEst.data *Print Method for stratEst.data*

Description

Print Method for stratEst.data

Usage

```
## S3 method for class 'stratEst.data'
print(x, ...)
```

Arguments

x	An object of class `stratEst.data`.
...	Further arguments passed to or from other methods.

Value

Prints a `data.frame` object that contains the data.

`print.stratEst.model` *Print Method for stratEst.model*

Description

Print Method for stratEst.model

Usage

```
## S3 method for class 'stratEst.model'
print(x, ...)
```

Arguments

`x`	An object of class `stratEst.model`.
`...`	Further arguments passed to or from other methods.

Value

No return value, prints a summary of the model to the console.

`print.stratEst.strategy` *Print Method for stratEst.strategy*

Description

Print Method for stratEst.strategy

Usage

```
## S3 method for class 'stratEst.strategy'
print(x, ...)
```

Arguments

`x`	An object of class `stratEst.strategy`.
`...`	Further arguments passed to or from other methods.

Value

No return value, prints a summary of the strategy to the console.

`round.stratEst.strategy` *Round Method for stratEst.strategy*

Description

Round Method for stratEst.strategy

Usage

```
## S3 method for class 'stratEst.strategy'
round(x, digits = 0)
```

Arguments

`x`	An object of class `stratEst.strategy`.
`digits`	Further arguments passed to or from other methods.

Value

A `stratEst.strategy` object with rounded variable values. 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`.

`strategies.DF2011` *strategies.DF2011*

Description

List of six prisoner's dilemma strategies (Dal Bo and Frechette 2011).

Usage

```
data(strategies.DF2011)
```

Format

Each strategy is encoded as a data.frame object. The rows of the data frame represent the states of the automaton. The first row is the start state of the automaton. Each data.frame object contains the following variables:

`prob.d` Probability to defect.

`prob.c` Probability to cooperate.

`tremble` Probability of a tremble.

`tr(cc)` State transition for the input cc.

`tr(cd)` State transition for the input cd.

`tr(dc)` State transition for the input dc.

`tr(dd)` State transition for the input dd.

Details

The prisoner’s dilemma strategies are:

Strategy which always defects.

ALLD ALLC Strategy which always cooperates.

GRIM Strategy which cooperates until one player defects, then GRIM defects forever.

TFT Strategy which cooperates unless the partner defected in the last round.

WSLS Strategy which cooperates if both players chose the same action last round, otherwise WSLS defects.Also known as PTFT.

T2 Strategy which cooperates until either player defects, then it defects twice and returns to cooperation (regardless of the actions during the punishment phase).

References

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

Examples

```
strategies <- strategies.DF2011[c("ALLD","ALLC","TFT","GRIM")]
```

`strategies.DFS2020` *strategies.DFS2020*

Description

The conformist and anticonformist strategy identifies by Dvorak, Fischbacher, and Schmelz (2020).

Usage

```
data(strategies.DFS2020)
```

Format

Each strategy is encoded as a stratEst.strategy object. The rows of the data frame represent the states of the strategy. The first row is the start state of the strategy. Each stratEst.strategy object contains the following variables:

`prob.follow` Probability to follow own preference.

`prob.deviate` Probability to deviate from the own preference.

`tr(not in line)` State transition for the input the choices of the others are not in line with the own preference.

`tr(in line)` State transition for the input the choices of the others are in line with the own preference.

Details

The strategies are:

Strategy that generally follows the own preference if the choices of the other group members are in line with the own preference and deviates from the own preference the choices of the other group members are not in line with the own preference.

conformist **anticonformist** Strategy that frequently deviates from the own preference the choices of the other group members are in line with the own preference and follows the own preference if the choices of the other group members are not in line with the own preference.

References

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

Examples

```
strategies <- strategies.DFS2020[c("conformist","anticonformist")]
```

`strategies.FRD2012` *strategies.FRD2012*

Description

List of eleven prisoner's dilemma strategies (Fudenberg, Rand, and Dreber 2012).

Usage

```
data(strategies.FRD2012)
```

Format

Each strategy is encoded as a data.frame object. The rows of the data frame represent the states of the automaton. The first row is the start state of the automaton. Each data.frame object contains the following variables:

`prob.d` Probability to defect.

`prob.c` Probability to cooperate.

`tremble` Probability of a tremble.

`tr(cc)` State transition for the input cc.

`tr(cd)` State transition for the input cd.

`tr(dc)` State transition for the input dc.

`tr(dd)` State transition for the input dd.

Details

c("ALLC","TFT","TF2T","TF3T","T2FT","T2F2T","GRIM","GRIM2","GRIM3","ALLD","DTFT")
The prisoner's dilemma strategies are:

ALLC Strategy which always cooperates.

TFT Strategy which cooperates unless the partner defected in the last round.

TF2T Strategy which cooperates unless the partner defected in the last two rounds.

TF3T Strategy which cooperates unless the partner defected in the last three rounds.

T2FT Strategy which cooperates unless the partner defected in either of the last two rounds.

T2F2T Strategy which cooperates unless the partner defected for two consecutive rounds of the last three rounds.

GRIM Strategy which cooperates until one player defects, then GRIM defects forever.

GRIM2 Strategy which cooperates until two consecutive rounds occur in which one player defected, then GRIM2 defects forever.

GRIM3 Strategy which cooperates until three consecutive rounds occur in which one player defected, then GRIM3 defects forever.

ALLD Strategy which always defects.

DTFT Strategy which starts with defection, then plays according to TFT.

References

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.

Examples

```
strategies <- strategies.FRD2012[c("ALLC","ALLD","TFT","GRIM","PTFT")]
```

`strategies.PD` *strategies.PD*

Description

List of 24 prisoner's dilemma strategies (Dal Bo and Frechette 2011; Fudenberg, Rand, and Dreber 2012; Breitmoser 2015).

Usage

```
data(strategies.PD)
```

Format

Each strategy is encoded as a data.frame object. The rows of the data frame represent the states of the automaton. The first row is the start state of the automaton. Each data.frame object contains the following variables:

`prob.d` Probability to defect.

`prob.c` Probability to cooperate.

`tremble` Probability of a tremble.

`tr(cc)` State transition for the input cc.

`tr(cd)` State transition for the input cd.

`tr(dc)` State transition for the input dc.

`tr(dd)` State transition for the input dd.

Details

The prisoner's dilemma strategies are:

Strategy which always cooperates.

AIAIICLD Strategy which always defects.

DC Strategy which starts with defection and then alternates between cooperation and defection.

DGRIM2 Strategy which starts with defection, then plays according to GRIM2.

DGRIM3 Strategy which starts with defection, then plays according to GRIM3.

DTF2T Strategy which starts with defection, then plays according to TF2T.

DTF3T Strategy which starts with defection, then plays according to TF3T.

DTFT Strategy which starts with defection, then plays according to TFT.

FC Strategy which starts with cooperation, then defects forever.

GRIM Strategy which cooperates until one player defects, then GRIM defects forever.

GRIM2 Strategy which cooperates until two consecutive rounds occur in which one player defected, then GRIM2 defects forever.

GRIM3 Strategy which cooperates until three consecutive rounds occur in which one player defected, then GRIM3 defects forever.

M1BF Strategy which cooperates if both players cooperated, and defects if both players defected in the last round. If the own action was cooperation and the other player defected, cooperate with some probability. If the own action was defection and the other player cooperated, cooperate with some (potentially different) probability.

PT2FT Strategy which cooperates if both players cooperated in the last two rounds, both players defected in the last two rounds, or both players defected two rounds ago and cooperated in the last round. Otherwise PT2FT defect.

PTFT Strategy which cooperates if both players chose the same action last round, otherwise PTFT defects.Also known as WSLS.

RAND Strategy which uniformly randomizes between cooperation and defection.

SGRIM Semi grim strategy (Breitmoser, 2015). The strategy cooperates if both players cooperated, and defects if both players defected in the last round. If one player defected and the other cooperated, cooperate with some probability.

T2 Strategy which cooperates until either player defects, then it defects twice and returns to cooperation (regardless of the actions during the punishment phase).

T2F2T Strategy which cooperates unless the partner defected for two consecutive rounds of the last three rounds.

T2FT Strategy which cooperates unless the partner defected in either of the last two rounds.

TF2T Strategy which cooperates unless the partner defected in the last two rounds.

TF3T Strategy which cooperates unless the partner defected in the last three rounds.

TFT Strategy which cooperates unless the partner defected in the last round.

WSLS Strategy which cooperates if both players chose the same action last round, otherwise WSLS defects.Also known as PTFT.

References

Breitmoser Y (2015). "Cooperation, but no Reciprocity: Individual Strategies in the Repeated Prisoner's Dilemma." *American Economic Review*, 105(9), 2882-2910.

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

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.

Examples

```
strategies <- strategies.PD[c("ALLC","ALLD","TFT","GRIM","PTFT")]
```

`strategies.RPS` *strategies.RPS*

Description

Six rock-paper-scissors strategies.

Usage

```
data(strategies.RPS)
```

Format

Each strategy is encoded as a stratEst.strategy object. The rows of the data frame represent the states of the strategy. The first row is the start state of the strategy. Each stratEst.strategy object contains the following variables:

`prob.r` Probability to play rock.

`prob.p` Probability to play paper.

`prob.s` Probability to play scissors.

`tremble` Probability of a tremble.

`tr(r)` State transition for the input last choice was rock.

`tr(p)` State transition for the input last choice was paper.

`tr(s)` State transition for the input last choice was scissors.

Details

The rock-paper-scissors strategies are:

Strategy which plays rock.

rock **paper** Strategy which plays paper.

scissors Strategy which plays scissors.

nash Nash equilibrium strategy which plays every action with probability one-third.

mixed Strategy which plays every action with a certain probability.

imitate Strategy which plays a random action in the first round and subsequently imitates the last choice.

Examples

```
strategies <- strategies.RPS[c("nash","mixed","imitate")]
```

`stratEst` *Strategy Estimation Function*

Description

Performs variants of the strategy estimation method.

Usage

```
stratEst(
  data,
  strategies,
  shares,
  coefficients,
  covariates,
  sample.id,
  response = "mixed",
  sample.specific = c("shares", "probs", "trembles"),
  r.probs = "no",
  r.trembles = "global",
  select = NULL,
  min.strategies = 1,
  crit = "bic",
  se = "analytic",
  outer.runs = 1,
  outer.tol = 1e-10,
  outer.max = 1000,
  inner.runs = 10,
  inner.tol = 1e-05,
  inner.max = 10,
  lcr.runs = 100,
  lcr.tol = 1e-10,
  lcr.max = 1000,
  bs.samples = 1000,
  quantiles = c(0.01, 0.05, 0.5, 0.95, 0.99),
  stepsize = 1,
  penalty = FALSE,
  verbose = TRUE
)
```

Arguments

`data`	A `stratEst.data` object or `data.frame`. Must contain the variables `choice`, `input`, `id`, `game`, `period`. The variable `id` identifies observations of the same individual across games and periods. The factor `input` indicates the discrete information observed by the individual before making a choice. The factor `choice` indicates the choice of the individual.

`strategies`	A list of strategies. Each strategy is a data.frame of class `stratEst.strategy`. Each row of the data.frame represents one state of the strategy. The first row defines the initial state which is entered if the variable input is NA. Column names which start with the string 'output.' indicate the columns which contain the multinomial choice probabilities of the strategy. For example, a column labeled 'output.x' contains the probability to observe the output 'x'. The column 'tremble' contains a tremble probability for pure strategies. Column names which start with the string 'input.' indicate the columns which contain the deterministic state transition of the strategy. For example, a column with name 'input.x' indicates the state transition after observing input 'x'.
`shares`	A vector of strategy shares. The elements to the order of strategies in the list `strategies`. Shares which are `NA` are estimated from the data. With more than one sample and sample specific shares, a list of column vectors is required.
`coefficients`	Column vector which contains the latent class regression coefficients. The elements correspond to the vector of estimates.
`covariates`	A character vector indicating the names of the variables in data that are the covariates of the latent class regression model. Rows with the same id must have the values of covariates. Missing value are not allowed.
`sample.id`	A character indicating the name of the variable which identifies the samples. Individual observations must be nested in samples. The same must be true for clusters if specified. If more than one sample exists, shares are estimated for each sample. All other parameters are estimated for the data of all samples. If the object is not supplied, it is assumed that the data contains only one sample.
`response`	A string which can be set to `"pure"` or `"mixed"`. If set to `"pure"` all estimated choice probabilities are pure, i.e. either zero or one. If set to `"mixed"` all estimated choice probabilities are mixed. The default is `"mixed"`.
`sample.specific`	A character vector defining which model parameters are sample specific. If the vector contains the character `"shares"` (`"probs"`, `"trembles"`), the estimation function estimates a set of shares (choice probabilities, trembles) for each sample in the data. If the vector does not contains the character `"shares"` (`"probs"`, `"trembles"`) one set of shares (choice probabilities, trembles) is estimated for the pooled data of all samples. Default is `c("shares","probs","trembles")`.
`r.probs`	A string which can be set to `"no"`, `"strategies"`, `"states"` or `"global"`. If set to `"strategies"`, the estimation function estimates strategies with one strategy specific vector of choice probabilities in every state of the strategy. If set to `"states"`, one state specific vector of choice probabilities is estimated for each state. If set to `"global"`, a single vector of probabilities is estimated which applies in every state of each strategy. Default is `"no"`.
`r.trembles`	A string which can be set to `"no"`, `"strategies"`, `"states"` or `"global"`. If set to `"strategies"`, the estimation unction estimates strategies with one strategy specific tremble probability. If set to `"states"`, one state specific tremble probability is estimated for each state. If set to `"global"`, a single tremble probability is estimated which globally. Default is `"global"`.
`select`	A character vector indicating which model parameters are selected. If the vector contains the character `"strategies"` (`"probs"`, `"trembles"`), the number

	of strategies (choice probabilities, trembles) is selected based on the selection criterion specified in `"crit"`. The selection of choice probabilities and trembles occurs obeying the restriction specified in `r.probs` and `r.trembles`. (E.g. if `r.probs` is set to `"strategies"`, `select = "probs"` will select the sets of choice probabilities within each strategy). Default is `NULL`.
`min.strategies`	An integer which specifies the minimum number of strategies in case of strategy selection. The strategy selection procedure stops if the minimum is reached.
`crit`	A string which can be set to `"bic"`, `"aic"` or `"icl"`. If set to `"bic"`, model selection based on the Bayesian Information criterion is performed. If set to `"aic"`, the Akaike Information criterion is used. If set to `"icl"` the Integrated Classification Likelihood criterion is used. Default is `"bic"`.
`se`	A string which can be set to `"analytic"` or `"bootstrap"`. If set to `"bootstrap"`, bootstrapped standard errors are reported. Default is `"analytic"`.
`outer.runs`	A positive integer which stets the number of outer runs of the solver. Default is 1.
`outer.tol`	A positive number which stets the tolerance of the continuation condition of the outer runs. The iterative algorithm stops if the relative decrease of the log-likelihood is smaller than `outer.tol`. Default is 0.
`outer.max`	A positive integer which stets the maximum number of iterations of the outer runs of the solver. The iterative algorithm stops if it did not converge after `"outer.max"` iterations. Default is 1000.
`inner.runs`	A positive integer which stets the number of inner runs of the solver. Default is 10.
`inner.tol`	A positive number which stets the tolerance of the continuation condition of the inner EM runs. The iterative algorithm stops if the relative decrease of the log-likelihood is smaller than `inner.tol`. Default is 0.
`inner.max`	A positive integer which stets the maximum number of iterations of the inner EM runs. The iterative algorithm stops if it did not converge after `inner.max` iterations. Default is 10.
`lcr.runs`	A positive integer which stets the number of estimation runs for latent class regression. Default is 100.
`lcr.tol`	A positive number which stets the tolerance of the continuation condition of the Latent Class Regression runs. The iterative algorithm stops if the relative decrease of the log-likelihood is smaller than `lcr.tol`. Default is 0.
`lcr.max`	A positive integer which stets the maximum number of iterations of the Latent Class Regression EM runs. The iterative algorithm stops if it did not converge after `lcr.max` iterations. Default is 1000.
`bs.samples`	A positive integer which sets the number of bootstrap samples drawn with replacement.
`quantiles`	A numeric vector indicating the quantiles of the sampling distribution of the estimated parameters. The quantiles are identified based on the standard error or based on bootstrapping the sampling distribution of the parameter.
`stepsize`	A positive number which sets the stepsize of the Fisher scoring algorithm used to estimate the coefficients of the latent class regression model. Default is one. Values smaller than one slow down the convergence of the algorithm.

`penalty`	A logical indicating if the Firth penalty is used to estimate the coefficients of the latent class regression model. Default is `FALSE`. Irrespective of the value specified here, the penalty is used in the case of a bootstrap of the standard errors of latent class regression coefficients.
`verbose`	A logical, if `TRUE` messages of the estimation process and a summary of the estimated model is printed to the console. Default is `TRUE`.

Details

The estimation function `stratEst()` returns maximum-likelihood estimates for the population shares and choice probabilities of a set of candidate strategies given some data from an economic experiment. Candidate strategies can be supplied by the user in the form of deterministic finite-state automata. The number and the complexity of strategies can be restricted by the user or selected based on information criteria. stratEst also features latent class regression to assess the influence of covariates on strategy choice.

Value

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

`strategies`	A list of fitted strategies.
`shares`	Matrix of strategy shares. The order of rows corresponds to the order of strategies defined in the input object `strategies`.
`probs`	Matrix of choice probabilities. The value `NA` indicates that the probability could not be estimated since data does not contain observations the model assigns to the corresponding state.
`trembles`	Matrix of tremble probabilities of the strategies. The value `NA` indicates that the corresponding probability could not be estimated since data does not contain observations the model assigns to the corresponding state.
`coefficients`	Matrix of latent class regression coefficients for strategies.
`shares.par`	Estimated strategy shares.
`probs.par`	Estimated choice probabilities.
`trembles.par`	Estimated tremble probabilities.
`coefficients.par`	Estimated latent class regression coefficients.
`shares.indices`	Indices of strategy shares.
`probs.indices`	Indices of choice probabilities.
`trembles.indices`	Indices of tremble probabilities.
`coefficients.indices`	Indices of latent class regression coefficients.
`loglike`	The log-likelihood of the model. Larger values indicate a better fit of the model to the data.
`crit.val`	The value of the selection criterion defined under `crit`. Larger values indicate a better fit of the model.

`eval`	Number of iterations of the solver. The reported number is the sum of iterations performed in the inner and the outer run which produced the reported estimates.
`tol.val`	The relative decrease of the log-likelihood in the last iteration of the algorithm.
`convergence`	Maximum absolute score of the model parameters. Small values indicate convergence of the algorithm to a (local) maximum of the negative log likelihood.
`entropy`	Entropy of the posterior probability assignments of individuals to strategies.
`state.obs`	A column vector with the number of weighted observations for each strategy state corresponding to the rows of `strategies`.
`posterior.assignments`	Posterior probability of each individual to use a strategy.
`prior.assignments`	Prior probability of each individual to use a strategy as predicted by the individual covariates.
`shares.se`	Standard errors of the estimated shares.
`probs.se`	Standard errors of the estimated choice probabilities.
`trembles.se`	Standard errors of the estimated trembles.
`coefficients.se`	Standard errors of the estimated coefficients.
`shares.score`	Score of the estimated shares.
`probs.score`	Score of the reported choice probabilities.
`trembles.score`	Score of the reported trembles.
`coefficients.score`	Score of the reported coefficients.
`shares.fisher`	Fisher information of the estimated shares.
`probs.fisher`	Fisher information of the reported choice probabilities.
`trembles.fisher`	Fisher information of the reported trembles.
`coefficients.fisher`	Fisher information of the reported coefficients.
`num.obs`	Number of observations.
`num.ids`	Number of individuals.
`num.par`	Total number of model parameters.
`free.par`	Total number of free model parameters.
`res.degrees`	Residual degrees of freedom (num.ids - free.par).
`shares.quantiles`	Quantiles of the estimated shares.
`probs.quantiles`	Quantiles of the estimated choice probabilities.
`trembles.quantiles`	Quantiles of the estimated tremble probabilities.
`coefficients.quantiles`	Quantiles of the estimated latent class regression coefficients.

`gammas`	Gamma parameter of the model.
`gammas.par`	Estimated gamma parameters.
`gammas.se`	Standard errors of the gamma parameters.
`#`	
`aic`	Akaike information criterion.
`bic`	Bayesian information criterion.
`icl`	Integrated classification likelihood information criteria.

Note

The strategy estimation method was introduced by (Dal Bo & Frechette 2011) to estimate the relative frequency of a fixed set of pure strategies in the indefinitely repeated prisoner's dilemma. Breitmoser (2015) extended the method to the estimation of behavior strategies. The **stratEst** package uses the EM algorithm (Dempster, Laird & Rubin 1977) and the Newton-Raphson method to obtain maximum-likelihood estimates for the population shares and choice probabilities of a set of candidate strategies. The package builds on other software contributions of the R community. To increase speed the estimation procedures, the package uses integration of C++ and R achieved by the Rcpp package (Eddelbuettel & Francois 2011) and the open source linear algebra library for the C++ language RppArmadillo (Sanderson & Curtin 2016).

References

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

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

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

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

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

`stratEst.check` *Check model assumptions*

Description

Check model assumptions

Usage

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

Arguments

`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`.

Details

The function for model checking of the package.

Value

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.

References

Wang Z, Xu B, Zhou HJ (2014). "Social Cycling and Conditional Responses in the Rock-Paper-Scissors Game." *Scientific Reports*, 4(1), 2045-2322.

Examples

```
## Fit and check a mixture model for the rock-paper-scissors data of Wang, Xu, and Zhou (2014).
strategies.mixture = strategies.RPS[c("nash","imitate")]
model.mixture <- stratEst.model(data.WXZ2014,strategies.mixture)
model.mixture.check <- stratEst.check( model.mixture )
print(model.mixture.check$fit)
```

`stratEst.data` *Creates a stratEst.data object.*

Description

Creates a stratEst.data object.

Usage

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

Arguments

`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. An integer or a vector of integers with 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`.

Details

The data generation function of the package.

Value

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.

Examples

```
## Transform the prisoner's dilemma data of Dal Bo and Frechette (2011).
data.DF2011 <- stratEst.data(DF2011, choice = "choice",
                             input=c("choice", "other.choice"), input.lag = 1)

## Transform the prisoner's dilemma data of Fudenberg, Rand, and Dreber (2012).
data.FRD2012 <- stratEst.data(data = FRD2012, choice = "choice",
                             input = c("last.choice", "last.other"))
```

`stratEst.model` *Strategy Estimation Function*

Description

The estimation function of the package.

Usage

```
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,
  min.strategies = 1,
  crit = "bic",
  se = "analytic",
  outer.runs = 1,
  outer.tol = 1e-10,
  outer.max = 1000,
  inner.runs = 10,
  inner.tol = 1e-05,
  inner.max = 10,
  lcr.runs = 100,
  lcr.tol = 1e-10,
  lcr.max = 1000,
  bs.samples = 1000,
  quantiles = c(0.05, 0.5, 0.95),
  step.size = 1,
  penalty = FALSE,
  verbose = FALSE
)
```

Arguments

`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 character 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`.

Details

The estimation function of the package obtains maximum likelihood estimates for the model parameters based on expectation maximization and Newton-Raphson algorithms.

Value

An object of class `stratEst.model`. 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 coefficient.
`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.

Note

Strategy estimation was introduced by Dal Bo and Frechette (2011) to estimate the maximum likelihood frequencies of a set of candidate strategies in the repeated prisoner's dilemma. Breitmoser (2015) introduces model parameters for the choice probabilities of individual strategies to the strategy estimation model. Dvorak and Fehrler (2018) extend the basic strategy estimation model by individual level covariates to explain the selection of strategies by individuals. 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 decrease the computation time, the package integrates C++ and R with the help of the R packages **Rcpp** (Eddelbuettel and Francois, 2011) and the open source linear algebra library for the C++ language **RppArmadillo** (Sanderson and Curtin, 2016).

References

Breitmoser Y (2015). "Cooperation, but no Reciprocity: Individual Strategies in the Repeated Prisoner's Dilemma." *American Economic Review*, 105(9), 2882-2910.

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

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.

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." *TWI Working Paper Series*.

Eddelbuettel D, Francois R (2011). "Rcpp: Seamless R and C++ Integration." *Journal of Statistical Software*, 40(8), 1-18.

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.

Sanderson C, Curtin R (2016). "Armadillo: A Template-Based C++ Library for Linear Algebra." *Journal of Open Source Software*, 1, 26.

Wang Z, Xu B, Zhou HJ (2014). "Social Cycling and Conditional Responses in the Rock-Paper-Scissors Game." *Scientific Reports*, 4(1), 2045-2322.

Examples

```
## Strategy model for rock-paper-scissors data of Wang, Xu, and Zhou (2014).
## Fit a mixture of the Nash strategy and a strategy that imitates the last choice.
strategies.mixture = list("nash" = strategies.RPS$nash, "imitate" = strategies.RPS$imitate)
model.mixture <- stratEst.model(data.WXZ2014,strategies.mixture)
```

`stratEst.simulate` *Simulation function for strategy estimation.*

Description

The simulation function of the package.

Usage

```
stratEst.simulate(
  data = NULL,
  strategies,
  shares = NULL,
  coefficients = NULL,
  covariate.mat = NULL,
  num.ids = 100,
  num.games = 5,
  num.periods = NULL,
  fixed.assignment = TRUE,
  input.na = FALSE,
  sample.id = NULL
)
```

Arguments

`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`.

Value

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.

Examples

```
## Simulate data of two strategies for choices "left" and "right".
lr <- c("left","right")
pi <- runif(1)
pr <- c(1,0,0,1)
tr <- c(1,2,1,2)
mixed <- stratEst.strategy(choices = lr, inputs = lr, prob.choices = c(pi, 1 - pi))
pure <- stratEst.strategy(choices = lr, inputs = lr, prob.choices = pr, tr.inputs = tr)
gamma <- runif(1)/4
pure$tremble <- gamma
beta <- rnorm(1)
p <- 1/sum(1 + exp(beta))
sim.shares <- c(p, 1-p)
sim.strategies <- list("mixed" = mixed, "pure" = pure)
sim.data <- stratEst.simulate(strategies = sim.strategies, shares = sim.shares)
```

`stratEst.strategy` *Creates a stratEst.strategy object.*

Description

Creates a stratEst.strategy object.

Usage

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

Arguments

`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 columnwise order.
`tr.inputs`	a vector of integers. The deterministic state transitions of the strategy in columnwise order.
`trembles`	a numeric vector. The tremble probabilities of the strategy.
`num.states`	an integer. The number states of the strategy.

Details

The strategy generation function of the package.

Value

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`.

Examples

```
## Nash equilibrium strategy of rock-paper-scissors
ins = c(NA,"rock","paper","scissors")
rps = c("rock","paper","scissors")
mixed = stratEst.strategy(choices = rps)
nash = stratEst.strategy(choices = rps, prob.choices = rep(1/3,3))
rock = stratEst.strategy(choices = rps, prob.choices = c(1,0,0))
```

`stratEst.test` *Runs t-tests if model parameters differ from user defined values*

Description

Runs t-tests if model parameters differ from user defined values

Usage

```
stratEst.test(
  model,
  par = c("shares", "probs", "trembles", "coefficients"),
  values = NA,
  alternative = "two.sided",
  coverage = 0.95,
  digits = 4,
  plot.tests = TRUE
)
```

Arguments

`model`	a fitted model of class `stratEst.model`.
`par`	a character vector. The class of model parameters to be tested. Default is `c("shares","probs","trembles", "coefficients")`.
`values`	a numeric vector. The values the parameter estimates are compared to. Default is NA which means zero.

`alternative`	a character string. The alternative hypothesis. Options are `"two.sided"`, `"greater"` or `"less"`. Default is `"two.sided"`.
`coverage`	a probability. The coverage of the plotted confidence intervals. Default is 0.95.
`digits`	an integer. The number of digits of the result.
`plot.tests`	a logical. Plots tests if `TRUE`.

Details

The test function of the package.

Value

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.
`std.error`	the standard error of the estimated parameter.
`t.value`	the TRUE statistic.
`res.degrees`	the residual degrees of freedom of the model.
`p.value`	the p value of the TRUE statistic.

References

Wang Z, Xu B, Zhou HJ (2014). "Social Cycling and Conditional Responses in the Rock-Paper-Scissors Game." *Scientific Reports*, 4(1), 2045-2322.

Examples

```
## Test if the choice probabilities of a mixed strategy for rock-paper-scissors.
## The rock-paper-scissors data is from Wang, Xu, and Zhou (2014).
model.mixed <- stratEst.model(data = data.WXZ2014, strategies = strategies.RPS["mixed"])
t.probs <- stratEst.test(model = model.mixed, par = "probs", values = 1/3)
print(t.probs)
```

`summary.stratEst.check` *Method dispatch for Generic Function Summary*

Description

Method dispatch for Generic Function Summary

Usage

```
## S3 method for class 'stratEst.check'
summary(object, ...)
```

Arguments

`object`	An object returned by the function`stratEst.check()`. An object of class `stratEst.check`.
`...`	additional arguments affecting the summary produced.

Value

No return value, prints a summary of the model checks to the console.

`summary.stratEst.data` *Method dispatch for Generic Function summary*

Description

Method dispatch for Generic Function summary

Usage

```
## S3 method for class 'stratEst.data'
summary(object, ...)
```

Arguments

`object`	An object to be summarized.
`...`	additional arguments affecting the result.

Value

No return value, prints a summary of the datas to the console.

`summary.stratEst.model` *Method dispatch for Generic Function Summary*

Description

Method dispatch for Generic Function Summary

Usage

```
## S3 method for class 'stratEst.model'
summary(object, ..., plot.shares = TRUE)
```

Arguments

`object`	An object returned by the estimation function`stratEst.model()`. An object of class `stratEst.model`.
`...`	additional arguments affecting the summary produced.
`plot.shares`	Logical. If TRUE a barchart of the shares is plotted.

Value

No return value, prints a summary of the model to the console.

`WXZ2014` *Data of the rock-paper-scissors game from Wang, Xu, and Zhou (2014)*

Description

Experimental data of 72 participants playing 300 periods of the rock-paper-scissors game in matching groups of six.

Usage

```
data(WXZ2014)
```

Format

A data.frame with 21.600 rows and 6 variables:

id Variable which identifies the participant.

game The identifier of the game.

period The period within the game.

choice A factor with three levels which indicates if the player chooses rock, paper or scissors.

other_choice A factor with three levels which indicates if the other player chooses rock, paper or scissors.

result A factor with three levels which indicates if the result for the player.

References

Wang Z, Xu B, Zhou HJ (2014). "Social Cycling and Conditional Responses in the Rock-Paper-Scissors Game." *Scientific Reports*, 4(1), 2045-2322.

Index