Package: RSGSolve (via r-universe)

August 27, 2024

Type Package

Title Solve Stochastic Games in R

Version 0.1.0

Author Sebastian Kranz

Maintainer Sebastian Kranz <sebastian.kranz@uni-ulm.de>

Description Simple R Interface to Ben Brook's SGSolve library. See http://www.benjaminbrooks.net/software.shtml

License GPL >= 2.0

LazyData TRUE

Imports Rcpp (>= 0.12.7)

Depends jsonlite, restorepoint

SystemRequirements C++11

LinkingTo Rcpp

RoxygenNote 5.0.0

Repository https://skranz.r-universe.dev

RemoteUrl https://github.com/skranz/RSGSolve

RemoteRef master

RemoteSha c784c114fec6857f737fcb4ae8043e71758d13af

Contents

`loadJsonSG` *Loads a json file rsg game specification and returns the rsg object as R list You then can call solveSG on the result to get the approximations to the equilibrium payoff sets*

Description

Loads a json file rsg game specification and returns the rsg object as R list You then can call solveSG on the result to get the approximations to the equilibrium payoff sets

Usage

```
loadJsonSG(file)
```

Arguments

`file`	the file name of the json file

`rcpp_hello` *Hello, Rcpp!*

Description

Returns an R `list` containing the character vector `c("foo", "bar")` and the numeric vector `c(0, 1)`.

Usage

```
rcpp_hello()
```

Examples

```
rcpp_hello()
```

solveSG *Solve a two-player stochastic game of perfect monitoring*

Description

Uses Ben Brooks implementation SGSolve of the pencil sharpening algorithm of Abreu, Brooks and Sanikov (2016) See http://www.benjaminbrooks.net/sgsolvedoc/html/index.html

Usage

```
solveSG(delta = rsg$delta, states = rsg$states, rsg = NULL,
  duplicate.first.point = TRUE, all.iter = TRUE, tol = 1e-12,
  normtol = tol, directiontol = tol, leveltol = tol, improvetol = tol)
```

Arguments

delta	the discount factor between 0 and 1
all.iter	shall return value contain the field ipoints that contains the pivots of all iterations?
stages	a list of stages. Each stage is a list with the following elements: - numActions: a size 2 vector that contains the number of actions for player 1 and 2. Its product is numActionProfiles. - payoffs: a matrix of dimension numActionProfiles x 2. The payoffs as function of the action profile. The actions are mapped to action profiles as described in the documentation of Brooks C++ library: A pair (a1,a2) is mapped into an action profile index using the formula a=a1+a2*numActions[s][a1]. - transition: a matrix of dimension numActionProfiles x numStates. The transition probabilities. Each row has to sum up to 1. @param duplicate.first.point if TRUE (default) the first row o the point matrices will be added again to the end. This facilitates plotting of the payoff set using the lines command.

Value

a list with the following elements solved: TRUE if the game could be solved

points: a list with a matrix for every state that contains the the extreme points of the final payoff set approximation. Using the terminology of ABS (2016): the pivots from the last revolution.

ipoints: a list with a matrix for every state, containing the pivots (extreme points) from every iteration.

revolution: a vector that denotes the revolution of each point in the ipoints matrices.

Index