Project 2

Technical requirements

- *Deadline:* [2024-01-29 Mon]
- *Format:* A single R file encoded as UTF-8.
- *Content:* The script must contain (a) your name and student's ID number in the first line of a script as a comment, (b) a definition of the function, and (c) two or more examples of the function's applications (you can use the following ones).
- *Packages:* Only standard core packages coming with a typical R installation are allowed (so that no further installation is required).
- *Instructor:* All files must be sent via email from your school account to the instructor's school account.

Project description

What is a game

This project is about game theory. We need to start with some simple definitions. First, we start with a set of players. This set is denoted by N. We work only with finite games, meaning there is a finite number of players. We number the players with integer numbers. Thus, the set of players is $N = \{1, 2, 3, \ldots, N\}$, where we use the symbol N for both the number of players and the set of players for ease of exposition. For example, if the game is a two-person game, we have $N = \{1, 2\}$. For a three-person game, we have $N = \{1, 2, 3\}$, and so on.

Each player has a set of actions (or pure strategies) available. These sets are denoted by A_i for a player i and all are finite. If a player i has k_i actions, the set A_i is $A_i = \{1, 2, \ldots, k_i\}$. For example, if the first player has two actions, the second player has three actions, and the third player has two actions, then $A_1 = \{1, 2\}$, $A_2 = \{1, 2, 3\}$ and $A_3 = \{1, 2\}$. The set of all possible action profiles is a cartesian product of action sets $A = A_1 \times A_2 \times \ldots A_N$. In the above example, the set of profiles has $2 \cdot 3 \cdot 2 = 12$ elements. These elements are listed below, where each row is a single profile.

```r
profiles <- expand.grid(1:2, 1:3, 1:2)
names(profiles) <- c("player 1", "player 2", "player 3")
profiles
```

```
   player 1 player 2 player 3
1         1        1        1
2         2        1        1
3         1        2        1
4         2        2        1
5         1        3        1
6         2        3        1
7         1        1        2
8         2        1        2
9         1        2        2
10        2        2        2
11        1        3        2
12        2        3        2
```

Each player has a utility function $u_i : A \to \mathbb{R}$. This function assigns a payoff to the player i based on the selected actions of all players (a profile of actions). For example, consider the simple, pure coordination game. In this game, there are two players, and each has two actions. Thus, we have $N = \{1, 2\}$ and $A_1 = A_2 = \{1, 2\}$. In this game, players get the payoff 1 if they play the same action (coordinate behavior) and 0 otherwise. Therefore, the payoff function is $u_i(a_1, a_2) = 1$ if $a_1 = a_2$ and $u_i(a_1, a_2) = 0$ if $a_1 \neq a_2$.

You can read about normal-form games in any textbook on game theory. There is plenty of information on the internet as well, e.g., Wikipedia.

How to represent a game in the R language

There are many ways to represent a normal-form game, especially in the R language. The easiest is to represent the payoff function as a multidimensional array. The profile of actions is an index that you can use to extract the payoffs. Here is an example of the simplest, pure coordination game.

```
game <- list(
  "player 1" = array(c(1, 0, 0, 1), dim = c(2, 2)),
  "player 2" = array(c(1, 0, 0, 1), dim = c(2, 2))
)
game
```

```
$`player 1`
     [,1] [,2]
[1,]    1    0
[2,]    0    1

$`player 2`
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

In the above snippet, a list holds two arrays. Using the list, we can extract payoffs for particular action profiles.

```
### Both players use the same actions
game$`player 1`[1, 1]
game$`player 2`[1, 1]
game$`player 1`[2, 2]
game$`player 2`[2, 2]

### Players use different actions
game$`player 1`[1, 2]
game$`player 2`[1, 2]
game$`player 1`[1, 2]
game$`player 2`[1, 2]
```

```
[1] 1
[1] 1
[1] 1
[1] 1
[1] 0
[1] 0
[1] 0
[1] 0
```

Nash equilibrium in pure strategies

What is the solution of a game? The most known solution idea is the concept of Nash equilibrium. Simply put, an action profile is a Nash equilibrium if no player has incentives to deviate from it. Mathematically speaking, an action profile $a = (a_1, \ldots, a_N) \in A$ is a Nash equilibrium if for any player $i \in N$ and any action $\widehat{a}_i \in A_i$ we have $u_i(a_i, a_{-i}) \geq u_i(\widehat{a}_i, a_{-i})$, where a_{-i} is an action profile of all players but the player i.

For example, the simplest, pure coordination game has two pure-strategy Nash equilibria. These are profiles $\{(1,1),(2,2)\}$. The profile $(1,1)$ is a Nash equilibrium because no player has incentives to deviate. The first player gets 1 in the profile $(1,1)$. If the player deviates and plays action 2, then the profile is $(2,1)$ and the payoff for the first player is 0. Thus, the player does not have the incentive to switch the action. The same goes for the second player. Also, the same analysis shows that the profile $(2,2)$ is also a Nash equilibrium.

The project's taks

The project’s task is to write a function that computes and returns a list of all pure-strategy Nash equilibriums. Technically, the goal is to write a function `getAllPureStrategyNE()`, taking a single argument named `game`, which is a list of arrays representing a game (as shown above).

Here is a list of examples of the function applications. We start with a simple coordination game.

```
### Example (coordination game for two players)
game <- list(
  "player1" = array(c(1, 0, 0, 1), dim = c(2, 2)),
  "player2" = array(c(1, 0, 0, 1), dim = c(2, 2))
)

getAllPureStrategyNE(game)
```

```
$`11`
[1] 1 1

$`22`
[1] 2 2
```

As discussed earlier, there are two Nash equilibria. The following example is the widely known prisoner’s dilemma game.

```
### Example (prisoners' dilemma for two players)
game <- list(
  "player1" = array(c(5, 10, 1, 2), dim = c(2, 2)),
  "player2" = array(c(5, 1, 10, 2), dim = c(2, 2))
)

getAllPureStrategyNE(game)
```

```
$`22`
[1] 2 2
```

The only Nash equilibrium in the prisoner’s dilemma is the profile $(2,2)$, which is not Pareto optimal. The next game is another popular game, named stag hunt.

```
### Example (stag-hare game for two players)
game <- list(
  "player1" = array(c(3, 2, 0, 2), dim = c(2, 2)),
  "player2" = array(c(3, 0, 2, 2), dim = c(2, 2))
)

getAllPureStrategyNE(game)
```

```
$`11`
[1] 1 1

$`22`
[1] 2 2
```

There are two Nash equilibria, one where both players hunt for a stag (profile $(1, 1)$) and the other where each hunts for a hare (profile $(2, 2)$). Let us investigate the anti-coordination game. This is a game where players get nothing if they play the same action and 1 otherwise. This is two-person example.

```
### Example (anti-coordination game for two players)
game <- list(
  "player1" = array(c(0, 1, 1, 0), dim = c(2, 2)),
  "player2" = array(c(0, 1, 1, 0), dim = c(2, 2))
)

getAllPureStrategyNE(game)
```

```
$`12`
[1] 1 2

$`21`
[1] 2 1
```

There are two pure-strategy Nash equilibria that are asymmetric. So far, we have dealt only with two-person games. Let's investigate a three-person version of the anti-coordination game.

```
### Example (anti-coordination game for  three players)
game <- list(
  "player1" = array(c(0, 1, 1, 1, 1, 1, 1, 0), dim = c(2, 2, 2)),
  "player2" = array(c(0, 1, 1, 1, 1, 1, 1, 0), dim = c(2, 2, 2)),
  "player3" = array(c(0, 1, 1, 1, 1, 1, 1, 0), dim = c(2, 2, 2))
)

getAllPureStrategyNE(game)
```

```
$`112`
[1] 1 1 2

$`121`
[1] 1 2 1

$`122`
[1] 1 2 2

$`211`
[1] 2 1 1

$`212`
[1] 2 1 2

$`221`
[1] 2 2 1
```

There are a lot of pure-strategy Nash equilibria. Intuitively, all these equilibria are such that players do not use the same action. Here is an example of a three-person coordination game.

```
### Example (coordination game for three players)
game <- list(
  "player1" = array(c(1, 0, 0, 0, 0, 0, 0, 1), dim = c(2, 2, 2)),
  "player2" = array(c(1, 0, 0, 0, 0, 0, 0, 1), dim = c(2, 2, 2)),
  "player3" = array(c(1, 0, 0, 0, 0, 0, 0, 1), dim = c(2, 2, 2))
)
```

```
getAllPureStrategyNE(game)
```

```
$`111`
[1] 1 1 1

$`222`
[1] 2 2 2
```

Not surprisingly, there are only two pure-strategy Nash equilibria at which players coordinate their behavior. However, let’s investigate the same coordination game for four players.

```
### Example (coordination game for four players)
game <- list(
  "player1" = array(c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1), dim = c(2, 2, 2,
  "player2" = array(c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1), dim = c(2, 2, 2,
  "player3" = array(c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1), dim = c(2, 2, 2,
  "player4" = array(c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1), dim = c(2, 2, 2,
)

getAllPureStrategyNE(game)
```

```
$`1111`
[1] 1 1 1 1

$`1122`
[1] 1 1 2 2

$`1212`
[1] 1 2 1 2

$`1221`
[1] 1 2 2 1

$`2112`
[1] 2 1 1 2

$`2121`
[1] 2 1 2 1

$`2211`
[1] 2 2 1 1

$`2222`
[1] 2 2 2 2
```

There are a lot more pure-strategy Nash equilibria for four players. However, not all equilibria are equal. The ones comprising diverse actions (not to mistake with mixed strategies) give zero payoff. Nonetheless, all of these are equilibria.

This project is not too difficult. The main problem is to understand what is the task. Other than that, it’s just an exercise in loops.

Date: 2023-10-25 Wed 00:00
Author: Michał Ramsza
Created: 2023-10-29 Sun 12:17
Validate