Games.jl での type 設計の実践例

尾山 大輔

github.com/oyamad/presentations

JuliaTokyo #6

2016 年 9 月 3 日

自己紹介・いきさつ

- 尾山大輔
  東京大学経済学研究科
- 専門：経済学理論 (より詳しくはゲーム理論)
  (研究はいまのところ紙と鉛筆のみ)
- プログラミング
  - 2014 年度から “ゼミ” で学生といっしょに Python を学びはじめる
  - 今年度のゼミでは Julia をやっている (quant-econ.net の Julia 版)

    (「ゼミの課題」https://github.com/OyamaZemi/exercises2016)
- QuantEcon.py/jl プロジェクトの Lead Developers の一員
  (あまりよくわかっていないけど)

前回 (JuliaTokyo #5) からの進展

- 前回の課題
  - (進化) ゲーム理論シミュレーションの Python コードを仕上げて Julia に移植する

    `github.com/oyamad/game_theory_models`
- 進展
  - ゲーム理論の基礎概念の部分を Python で書いた (QuantEcon.py)
  - それを Julia に移植した (Games.jl)
  - (進化ゲーム・シミュレーションの方はあまり進展なし)

今回の内容—Games.jl での経験談

- ゲーム理論の基礎概念をタイプで定義し，メソッドをいろいろ定義したが，そのときの経験をお話しする
- 初心者を脱したくらいの人の参考にでもなれば

ゲーム理論とは

- ゲーム理論は経済学の基礎理論のひとつ
  - 社会の構成主体 (人とか企業とか) の行動は互いに影響しあっている
  - 主体たちの誘因 (インセンティブ) がかみ合っている状態 (“均衡”) として社会の現象をとらえる
- 標準形ゲーム (normal form game)
  - プレイヤーの集合 $I = \{1, \ldots, N\}$
  - 各プレイヤー $i \in I$ の行動の集合 $A_i = \{1, \ldots, n_i\}$
  - 各プレイヤー $i \in I$ の利得関数 $u_i \colon A_i \times \cdots \times A_{i+N-1} \to \mathbb{R}$
- ナッシュ均衡 (Nash equilibrium)

  互いに最適反応になっているような行動の組のこと

例 1—協調ゲーム

- プレイヤーの集合 $I = \{1, 2\}$
- 行動空間 $A_1 = A_2 = \{1, 2\}$

  (たとえば 1: 言語 J, 2: 言語 P)
- 利得表

	J	P
J	$4, 4$	$0, 2$
P	$2, 0$	$3, 3$

- (純粋行動) ナッシュ均衡は (J, J) と (P, P)

例 2—じゃんけん

- プレイヤーの集合 $I = \{1, 2\}$
- 行動空間 $A_1 = A_2 = \{1, 2, 3\}$

  (1: Rock, 2: Paper, 3: Scissors)
- 利得表

	R	P	S
R	$0, 0$	$-1, 1$	$1, -1$
P	$1, -1$	$0, 0$	$-1, 1$
S	$-1, 1$	$1, -1$	$0, 0$

- 純粋行動ナッシュ均衡は存在しない
- 混合行動ナッシュ均衡は $((1/3, 1/3, 1/3), (1/3, 1/3, 1/3))$
- 例 1 のゲームも $((3/5, 2/5), (3/5, 2/5))$ という混合行動ナッシュ均衡をもつ

デモ

`github.com/oyamad/presentations/tree/master/JuliaTokyo06`

- `Player` タイプ，`NormalFormGame` タイプを実装する
- 与えられた行動の組がナッシュ均衡であるかどうかを判定する関数を実装する

今後の課題

- 利得配列をプリントする関数を作る (Issue #7)
- 型安定性の問題を解決する (Issue #2)
- 有名なゲームを返す関数を作る (Issue #3)
- 利得配列ではなく利得関数を field としてもたせるのはどうか (Issue #6)
- ナッシュ均衡を計算するアルゴリズムを実装する
  - Lemke-Howson (1964) (PR #8, #10)
  - Lemke (1965)
  - D. Avis, G. Rosenberg, R. Savani, and B. von Stengel (2010)
  - ...
- (進化) ゲーム理論シミュレーションの Python コードを仕上げて Julia に移植する (`github.com/oyamad/game_theory_models`)