R でゲーム理論分析
グループ間競争を例として

高知工科大学　経済・マネジメント学群

上條良夫

R 勉強会 @ 高知工科大学

2019年12月5日

Group Size Paradox (GSP)

- The most important point about small groups … is that they may very well be able to provide themselves with a collective good …. In this smaller groups differ from larger ones. The larger a group is, the farther it will fall short of obtaining an optimal supply of any collective good …. In short, larger the group, the less it will further its common interests (Olson 1965, p36)

Group Size Paradox (GSP)

(Two meanings)

Large group may

fail to supply the collective goods（第一の意味）

lose against the small group（第二の意味）

(Two reasons)

because

benefit is (partly) shared by group members (weak incentive for give input)

they strategically free-ride on the effort of their fellows (free-rider problem)

オルソン問題の数理的再検討（GSP の第一の意味）

　1人意思決定問題からナッシュ均衡分析へ

実証分析の整理

　囚人のジレンマなどに関する実験室実験

　社会運動に関する計量社会学的研究

P125

（未解決の）理論的課題

(1) 他の集団との関係を考慮したフォーマライゼーション

… また、その他の集団との競争・競合関係が存在する場合もある。集合行為が成功するかどうかは、この闘争・競争のプロセスにも依存している …

レントシーキングゲーム
コンテストゲーム

ゴードン・タロック

出典: フリー百科事典『ウィキペディア（Wikipedia）』

レントシーキング（英: rent seeking）とは、民間企業などが政府や官僚組織へ働きかけを行い、法制度や政治政策の変更を行うことで、自らに都合よく規制を設定したり、または都合よく規制の緩和をさせるなどして、超過利潤（レント）を得るための活動を指す[1][*要ページ番号*]。またこれらの活動を行う人を**レントシーカー**や**ロビイスト**などと呼ぶ。
これによる支出は生産とは結びつかないため、社会的には資源の浪費とみなされる。

グループコンテストゲーム（集団間競争ゲーム）

研究の目的

グループ間競争の理論的分析、かつ実験室からの検証

木村の指摘

… また、その他の集団との競争・競合関係が存在する場合もある。集合行為が成功するかどうかは、この闘争・競争のプロセスにも依存している …

を踏まえて、幅広い競争形態（パラメータで制御）において、 GSPが起きるのかを検討

In this environment, there is no pure strategy NE (Konrad, 2009)

– According to Sheremeta (2015), “The main reason is that group contests employing the lottery CSF usually have pure strategy equilibria, making it easier for subjects to learn and for researchers to analyze the data, while contests employing the auction CSF have only non-degenerate mixed strategy equilibria (Konrad, 2009)”

Model (experimental task)

- Two competing groups (A and B) have the opportunity to expend scarce resources (x, effort, time, money, or troops) in order to affect the probability of winning prizes (P)
  - Global conflict
  - Rent seeking activity (bands of radio wave for cell phone)
  - Development race among drug firms

- Group A and Group B
- N = 6: the total population
- N_A and N_B : the population of Groups A and B ($N = N_A + N_B$)
- A and B compete for a fixed prize P = 600

- Individual i in Groups A or B chooses an effort level x_i in [0, 100]
- X_A : the sum of efforts of individuals in Group A
- X_B : the sum of efforts of individuals in Group B

- Tullock Contest between groups
  - The probability of Group A obtaining Prize is

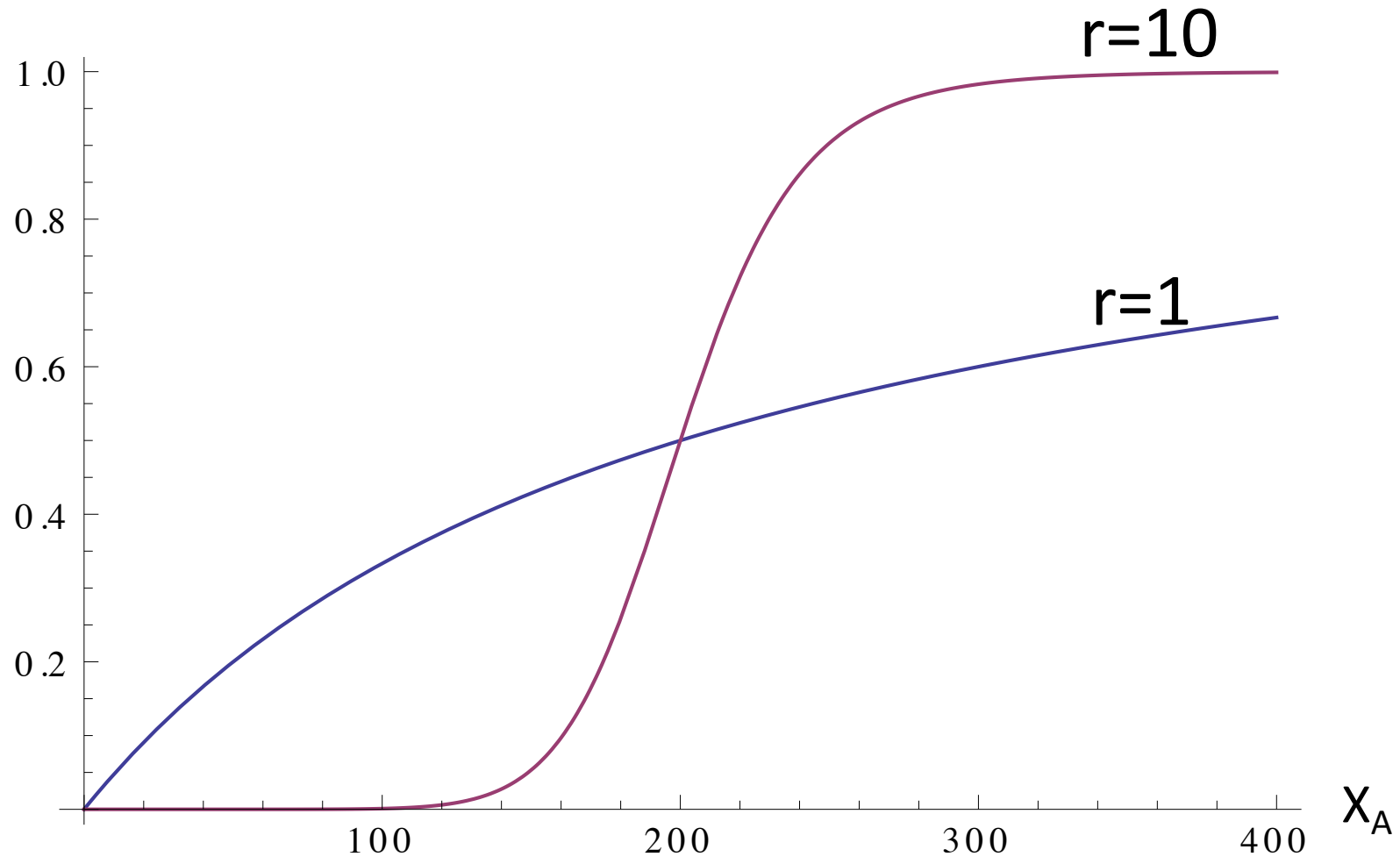
$$\frac{X_A^r}{X_A^r + X_B^r}$$

- r (>0) is a parameter that measures the sensitivity of the probability of winning to the ratio of group efforts

- Small r (e.g., r = 1) case is referred to a Lottery Contest
- Large r (e.g., r = 10) case is referred to a Auction-Like Contest

$$\frac{100^1}{100^1 + 200^1} \approx 0.33 \qquad \frac{100^{10}}{100^{10} + 200^{10}} \approx 0.00$$

Winning Prob. of Group A, given $X_B = 200$



- P = 600 : the size of prize.
  - Equal division among members in winning group

- Individuals are risk-neutral and have a linear cost technology

- The expected payoff of individual i in Group A is

$$\frac{X_A^r}{X_A^r + X_B^r} \times \frac{P}{N_A} - x_i$$

Probability of Group A winning | i's share of prize P | Cost of effort

- Individuals choose their effort at the same time
  - Usually, Nash Equilibrium is used to analyze this situation

直感的な予想

勝利確率がグループの努力量に鋭敏に反応する競争形態（r=10）では、鈍感な競争形態（r=1）よりも、GSP は起きにくいのではなかろうか

Theoretical challenge to analyze auction-like contests

- It is shown that in a Tullock Contest, the unique pure strategy NE is guaranteed only when $r < \text{or} = 2$.
  - This holds even for two-player contest
  - Thus, our group contest game does not have a pure strategy NE if contest is auction-like ($r = 10$)

- The existence of mixed strategy NE is always guaranteed by Nash (1950), but
  - Strong doubt to the presumption that actual people follow the mixed strategy and their profile converges to an equilibrium

- そこで、この研究では、ナッシュ均衡の合理性の仮定を弱めた均衡概念である Quantal Response Equilibrium を解く

Quantal Response Equilibrium

Description of a game

- Player set: $N = \{1, 2, ..., n\}$
- Strategy space: S_i for $i \in N$
- Strategy profile: $s \in S = S_1 \times ... \times S_n$
- i's payoff function: $u_i(s)$ for $s \in S$

Quantal response equilibrium (QRE)

- Probability choosing s_i: $p_i(s_i)$
- i's expected utility when i choose s_i and the other $n-1$ players follow ps.

$$\pi_i(s_i, p_{-i}) = \sum_{s_1 \in S_1} ... \sum_{s_{i-1} \in S_{i-1}} \sum_{s_{i+1} \in S_{i+1}} ... \sum_{s_n \in S_n}$$

$$p_1(s_1) \times ... \times p_{i-1}(s_{i-1}) \times p_{i+1}(s_{i+1}) \times ... \times p_n(s_n) u(s_1, ..., s_i, ..., s_n)$$

- i's Rationality parameter $\lambda_i > 0$
- $p = (p_1, ..., p_n)$ is QRE iff

$$p_i(\hat{s}_i) = \frac{e^{\lambda \pi_i(\hat{s}_i, p_{-i})}}{\sum_{s_i \in S_i} e^{\lambda \pi_i(s_i, p_{-i})}} ...(1)$$

for all $\hat{s}_i \in S_i$ and for all $i \in N$.

- $p = (p_1, ..., p_n)$ is QRE iff

$$p_i(\hat{s}_i) = \frac{e^{\lambda \pi_i(\hat{s}_i, p_{-i})}}{\sum_{s_i \in S_i} e^{\lambda \pi_i(s_i, p_{-i})}} ...(1)$$

for all $\hat{s}_i \in S_i$ and for all $i \in N$.

- Soft max 型の反応
  - 期待利得の高い純戦略を高い確率で選ぶ
- 均衡
  - 互いに、相手の混合戦略に対して 、(1) で計算される混合戦略を取り合う状況
- ラムダの意味
  - ラムダが 0 のとき完全にランダム
  - ラムダがプラス無限大のときナッシュ均衡（混合戦略）

Group Contest Game の QRE を計算したい

Group Contest Game of our Expriment

- Player set: $N = \{1, 2, 3, 4, 5, 6\}$
- Strategy space: $S_i = \{0, 1, 2, ..., 100\}$ for $i \in N$
- Strategy profile: $s \in S = S_1 \times ... \times S_n$
- N is partitioned into A and B.
- i's payoff function: for $i \in A$,

$$u_i(s) = \frac{(\sum_{j \in A} s_j)^r}{(\sum_{j \in A} s_j)^r + (\sum_{k \in B} s_k)^r} \times \frac{600}{|A|} - s_i,$$

  where $r > 0$ represents the sensitivity of the winning function.

- i's payoff function: for $i \in B$,

$$u_i(s) = \frac{(\sum_{k \in B} s_j)^r}{(\sum_{j \in A} s_j)^r + (\sum_{k \in B} s_k)^r} \times \frac{600}{|B|} - s_i,$$

R による数値計算に挑戦

数値計算の手順

(1) 混合戦略プロファイル (p) を入力すると、pに対する soft max 型の反応 (p*) を計算し、p* と p の二乗差を返す関数を準備（これを Loss 関数とよぶ）

(2) Loss 関数を制約条件付きで最小化する（solnp）

Loss 関数 = 0 (つまり、p = p*)が QRE の必要十分条件

上記手順で QRE を計算できる

普通に計算すると膨大な時間がかかる

純戦略 s_i を取った際の期待値

$$\pi_i(s_i, p_{-i}) = \sum_{s_1 \in S_1} ... \sum_{s_{i-1} \in S_{i-1}} \sum_{s_{i+1} \in S_{i+1}} ... \sum_{s_n \in S_n}$$

$$p_1(s_1) \times ... \times p_{i-1}(s_{i-1}) \times p_{i+1}(s_{i+1}) \times ... \times p_n(s_n) u(s_1, ..., s_i, ..., s_n)$$

普通に計算すると膨大な時間がかかる
純戦略 s_i を取った際の期待値

$$\pi_i(s_i, p_{-i}) = \sum_{s_1 \in S_1} \dots \sum_{s_{i-1} \in S_{i-1}} \sum_{s_{i+1} \in S_{i+1}} \dots \sum_{s_n \in S_n}$$

$$p_1(s_1) \times \dots \times p_{i-1}(s_{i-1}) \times p_{i+1}(s_{i+1}) \times \dots \times p_n(s_n) u(s_1, \dots, s_i, \dots, s_n)$$

```
for (i1 in 1:100) {
    for (i2 in 1:100){
        for (i3 in 1:100){
            for (i4 in 1:100){
                for (i5 in 1:100){

EU = EU + u(s, i1, i2, i3, i4, i5) * p1(i1) * p2(i2) * p3(i3) * p4(i4) * p5(i5)

                }
            }
        }
    }
}
```

これを普通に計算すると => 100^5 = 10^10 = 100億回のループ
仮に1ループの結果を保存してそれが1バイトだとしても、10GBになる

戦略空間を {5, 15, 25, …, 95} に限定して計算する
つまり、戦略の数が十分の一

```
for (i1 in 1:10) {
      for (i2 in 1:10){
            for (i3 in 1:10){
                  for (i4 in 1:10){
                        for (i5 in 1:10){

EU = EU + u(s, i1, i2, i3, i4, i5) * p1(i1) * p2(i2) * p3(i3) * p4(i4) * p5(i5)

                        }
                  }
            }
      }
}
```

ここまでやって、ようやく1時間程度で、ある条件・あるラムダにおける QRE が計算できた

ある条件のあるラムダに関するQREを計算するのに１時間だと、様々なバリエーションを検討するのが大変

さらなる計算上の工夫が必要

（１）並列計算（snowパッケージ）

（２）Loss 関数を　C++ で記述（Rcpp パッケージ）

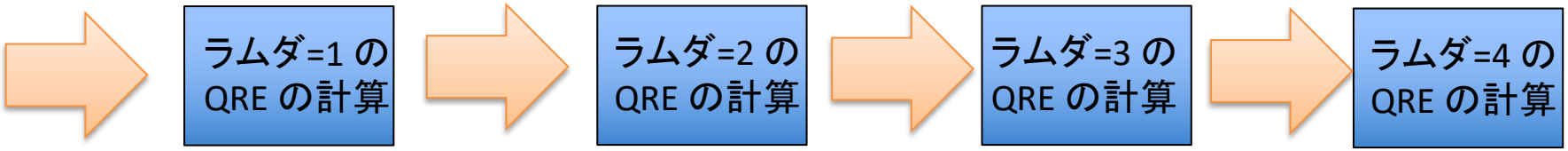
（３）Loop の畳み込み

ここまでやって、QREを数秒で計算できるようになった

並列化

全体の作業＝ラムダ 1, 2, 3, 4 の QRE の計算

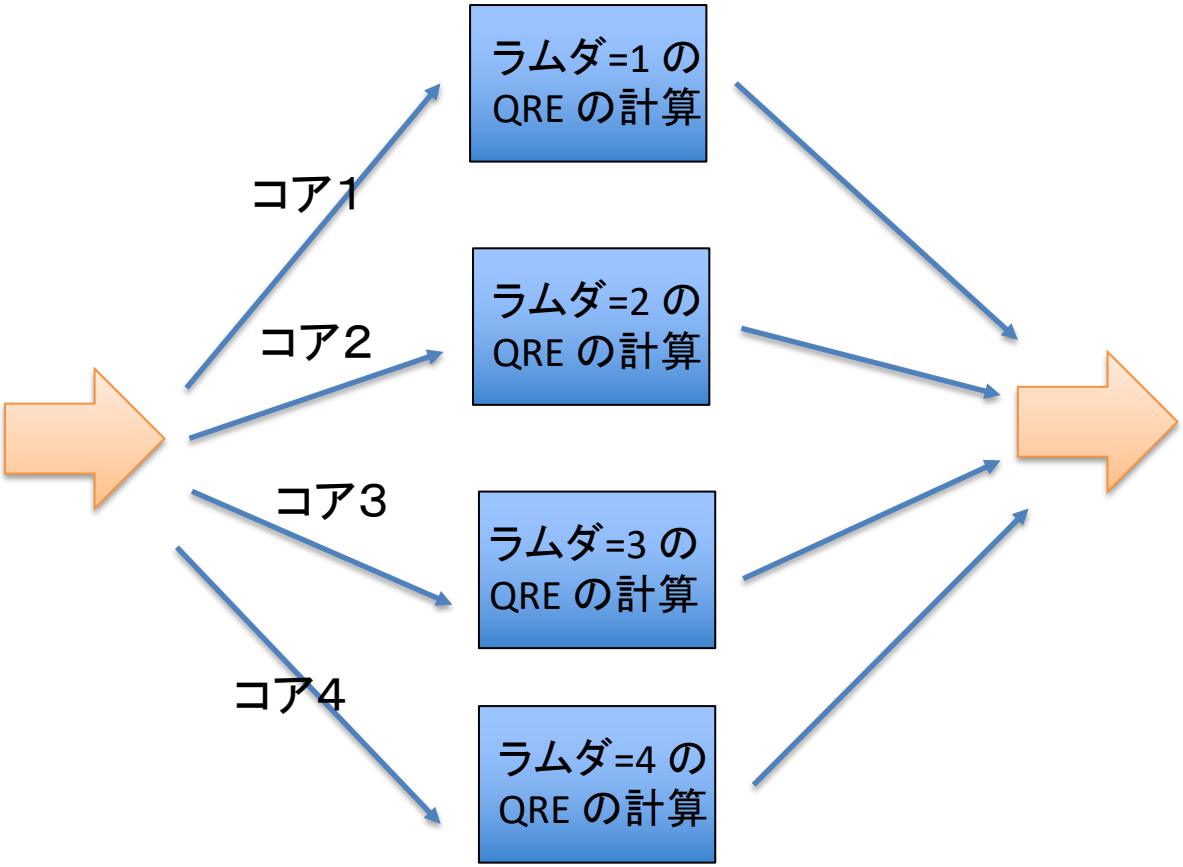
普通の方法



並列化

snowパッケージ

コアごとに計算を分業



計算時間は 1/2 ~ 1/3 に

Loss 関数を　C++ で記述

Rcpp パッケージ

別の C++ ファイルで書いた関数を souceCPP で読み込む

あとは通常の関数として使用可能

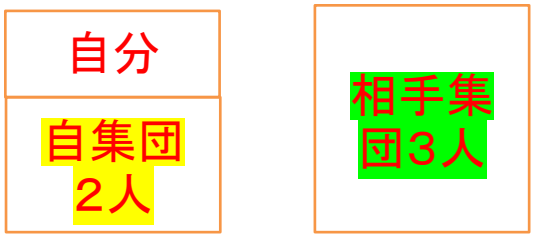
```
WinGroup1 = 0 # Group 1 expected winning probability when one player in group 1
WinGroup2 = 0 # Group 2 expected winning probability when one player in group 2

for (i1 in 1:10){
  for (i2 in 1:10){
    for (j1 in 1:10){
      for (j2 in 1:10){
        for (j3 in 1:10){
          OurGroupEffort = myChoice + 10 * (i1-1) + 5 + 10 * (i2-1) + 5
          YourGroupEffort = 10 * (j1-1) + 5 + 10 * (j2-1) + 5 + 10 * (j3-1) + 5
          WinGroup1 = WinGroup1 + OurGroupEffort^r/(OurGroupEffort^r + YourGrou
            OppProb1[i1] * OppProb1[i2] * OppProb2[j1] * OppProb2[j2] * OppProb

          OurGroupEffort = myChoice + 10 * (i1-1) + 5 + 10 * (i2-1) + 5
          YourGroupEffort = 10 * (j1-1) + 5 + 10 * (j2-1) + 5 + 10 * (j3-1) + 5
          WinGroup2 = WinGroup2 + OurGroupEffort^r/(OurGroupEffort^r + YourGrou
            OppProb2[i1] * OppProb2[i2] * OppProb1[j1] * OppProb1[j2] * OppProb
        }
      }
    }
  }
}
```

```
WinGroup1 = 0;
WinGroup2 = 0;

for (int i1 = 0; i1 < 20; i1++ ){
  for (int i2 = 0; i2 < 20; i2++ ){
    for (int j1 = 0; j1 < 20; j1++ ){
      for (int j2 = 0; j2 < 20; j2++ ){
        for (int j3 = 0; j3 < 20; j3++ ){

          OurGroupEffort = myChoice + 5 * i1 + 5 + 5 * i2 + 5;
          YourGroupEffort = 5 * j1 + 5 + 5 * j2 + 5 + 5 * j3 + 5;

          WinGroup1 = WinGroup1 + ( pow(OurGroupEffort, var2) /( pow(O

          OurGroupEffort = myChoice + 5 * i1 + 5 + 5 * i2 + 5;
          YourGroupEffort = 5 * j1 + 5 + 5 * j2 + 5 + 5 * j3 + 5;

          WinGroup2 = WinGroup2 + ( pow(OurGroupEffort, var2) / ( pow(

        }
      }
    }
  }
}
```

Loss 関数を　C++ で記述

計算時間は 1/10 以下に！

畳み込み（Convolution）
3 vs 3 を例に

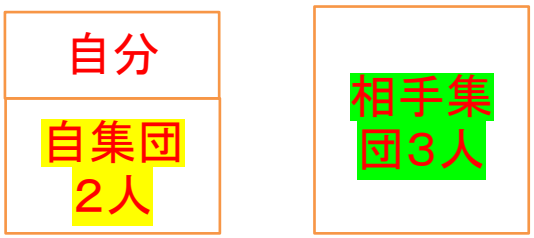


自分が戦略 s をとり、周りが混合戦略をとっていえるときの、自分の期待利得は？

畳み込み（Convolution）
3 vs 3 を例に

```
for (i1 in 1:10) {
    for (i2 in 1:10){
        for (i3 in 1:10){
            for (i4 in 1:10){
                for (i5 in 1:10){

    EU = EU + u(s, i1, i2, i3, i4, i5) * p1(i1) * p2(i2) * p3(i3) * p4(i4) * p5(i5)
                }
            }
        }
    }
}
```

u(s, i1 + i2, i3 + i4 + i5)



畳み込みとは
3 vs 3 を例に

```
for (i1 in 1:10) {
    for (i2 in 1:10){
        for (i3 in 1:10){
            for (i4 in 1:10){
                for (i5 in 1:10){

    EU = EU + u(s, i1, i2, i3, i4, i5) * p1(i1) * p2(i2) * p3(i3) * p4(i4) * p5(i5)

                u(s, i1 + i2, i3 + i4 + i5)

                }
            }
        }
    }
}
```

畳み込みとは
3 vs 3 を例に

```
for (i1 in 1:10) {
    for (i2 in 1:10){
        for (i3 in 1:10){
            for (i4 in 1:10){
                for (i5 in 1:10){

    EU = EU + u(s, i1, i2, i3, i4, i5) * p1(i1) * p2(i2) * p3(i3) * p4(i4) * p5(i5)
```

u(s, i1 + i2, i3 + i4 + i5)　　p12(k)　　p345(h)

k　　h

```
                }
            }
        }
    }
}
```

畳み込みとは
3 vs 3 を例に

```
for (i1 in 1:10) {
    for (i2 in 1:10){
        for (i3 in 1:10){
            for (i4 in 1:10){
                for (i5 in 1:10){

    EU = EU + u(s, i1, i2, i3, i4, i5) * p1(i1) * p2(i2) * p3(i3) * p4(i4) * p5(i5)

                }
            }
        }
    }
}
```

for (k in 2:20) {

for (h in 3:30) {

u(s, i1 + i2, i3 + i4 + i5)

k h

p12(k)

p345(h)

```
for (k in 2:20) {
    for (h in 3:30) {
        EU = EU + u(s, k, h) p12(k) p345(h)
    }
}
```

```
for (i1 in 1:10) {
        for (i2 in 1:10){
                k = i1 + i2
                p12(k) = p12(k)  + p1(i1) * p2(i2)
        }
}
```

```
for (k in 2:20) {
        for (h in 3:30) {
                EU = EU +   u(s,  k, h)    p12(k)    p345(h)
        }
}
```

```
for (i1 in 1:10) {
        for (i2 in 1:10){
                k = i1 + i2
                p12(k) = p12(k)  + p1(i1) * p2(i2)
        }
}
for (i3 in 1:10) {
        for (i4 in 1:10){
                for (i5 in 1:10){
                        h = i3 + i4 + i5
                        p345(h) = p345(h)  + p3(i3) * p3(i4) * p5(i5)
                }
        }
}
for (k in 2:20) {
        for (h in 3:30) {
                EU = EU +   u(s,  k, h)      p12(k)      p345(h)
        }
}
```

```
for (i1 in 1:10) {
        for (i2 in 1:10){
                k = i1 + i2
                p12(k) = p12(k)  + p1(i1) * p2(i2)
        }
}
for (i3 in 1:10) {
        for (i4 in 1:10){
                for (i5 in 1:10){
                        h = i3 + i4 + i5
                        p345(h) = p345(h)  + p3(i3) * p3(i4) * p5(i5)
                }
        }
}
  for (k in 2:20) {
      for (h in 3:30) {
          EU = EU +   u(s,  k, h)      p12(k)    p345(h)
      }
}
```

100回

1000回

約600回

10^5 = 100000回から約1700回まで削減

畳み込み（Convolution）
3 vs 3 を例に

計算時間は1/10以下に！！！

Group Contest Game の QRE を計算したい

Group Contest Game of our Expriment

- Player set: $N = \{1, 2, 3, 4, 5, 6\}$
- Strategy space: $S_i = \{0, 1, 2, ..., 100\}$ for $i \in N$
- Strategy profile: $s \in S = S_1 \times ... \times S_n$
- N is partitioned into A and B.
- i's payoff function: for $i \in A$,

$$u_i(s) = \frac{(\sum_{j \in A} s_j)^r}{(\sum_{j \in A} s_j)^r + (\sum_{k \in B} s_k)^r} \times \frac{600}{|A|} - s_i,$$

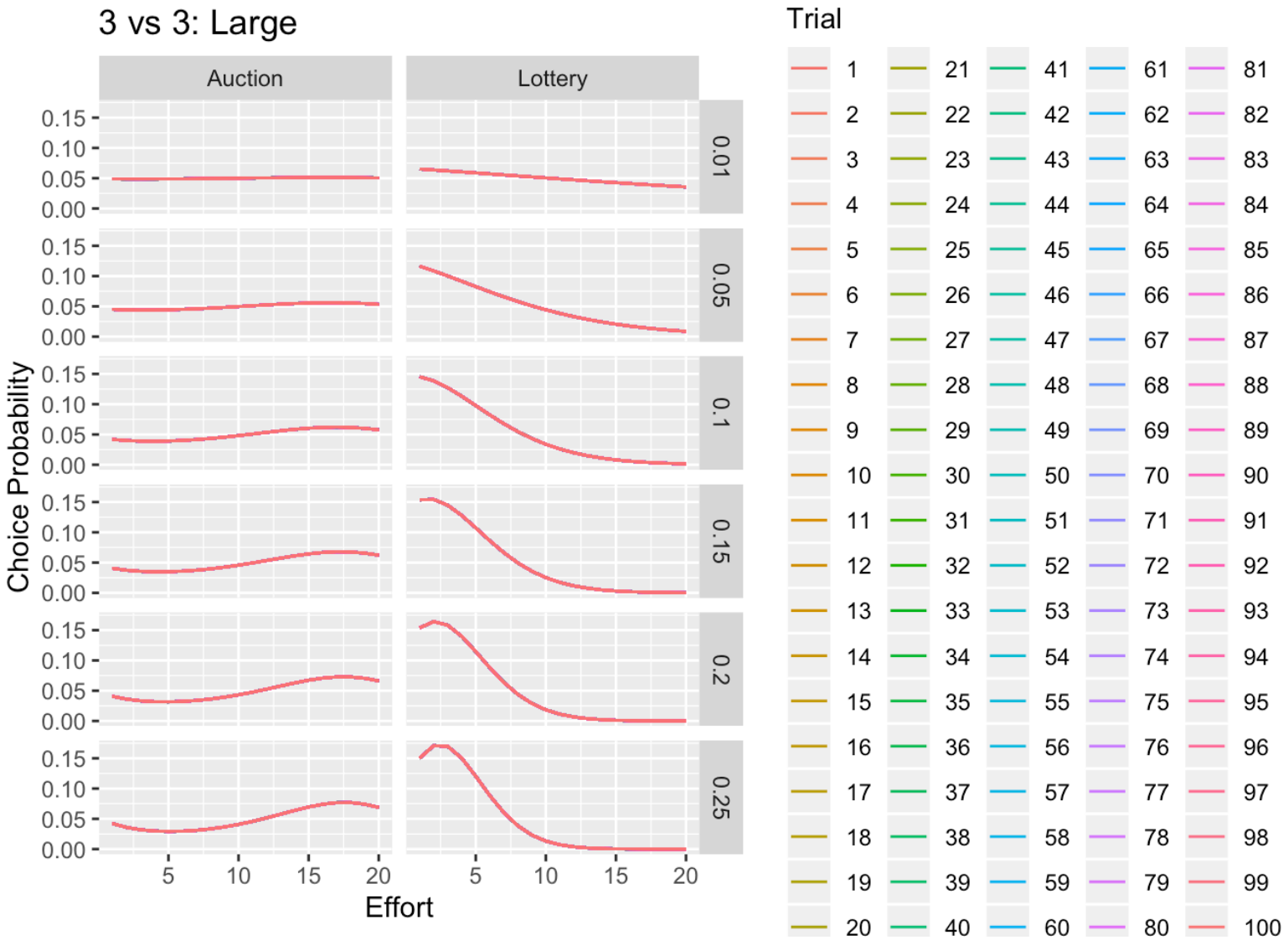
  where $r > 0$ represents the sensitivity of the winning function.

- i's payoff function: for $i \in B$,

$$u_i(s) = \frac{(\sum_{k \in B} s_j)^r}{(\sum_{j \in A} s_j)^r + (\sum_{k \in B} s_k)^r} \times \frac{600}{|B|} - s_i,$$

計算したらこんな感じになった（20分割）

3 vs 3



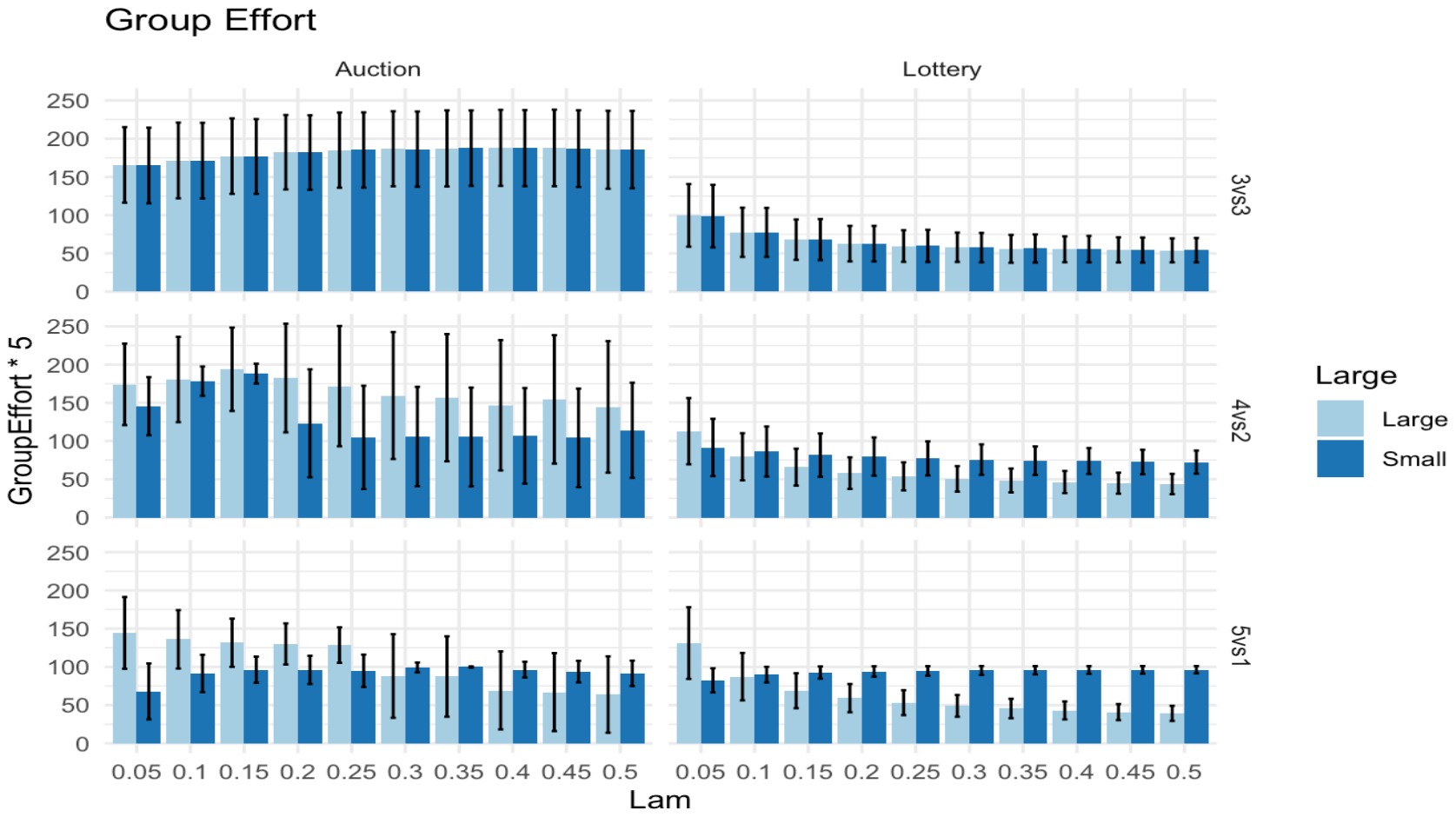
Predictions from QRE

- To investigate the relation between the competition structures (r=1 or r=10) and the GSP, we adopt the following 2 * 3 design

	Auction (r = 10)	**Lottery (r = 1)**
3 vs 3		
2 vs 4		
1 vs 5		

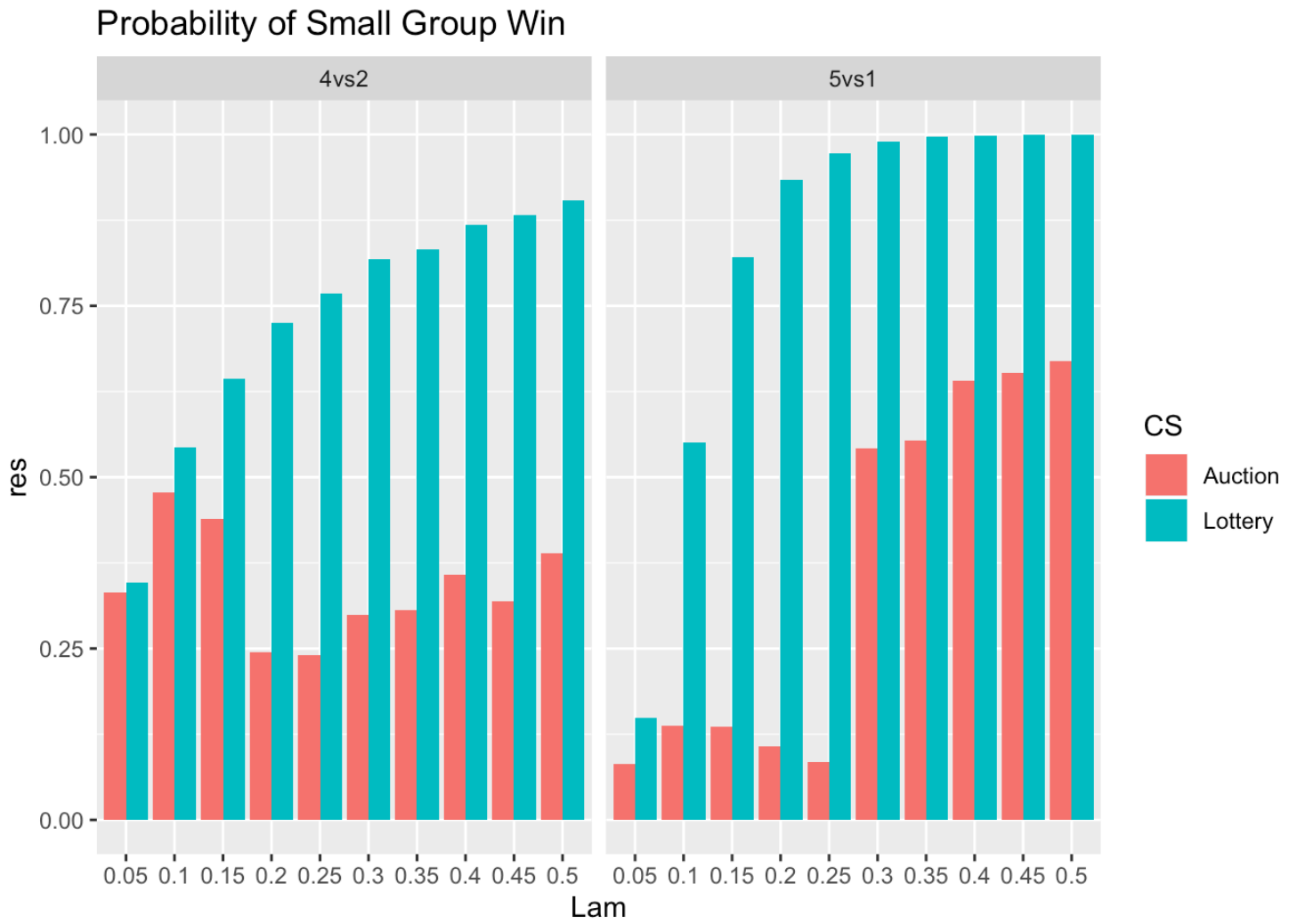
Experimental Design and Predictions from QRE

- To investigate the relation between the competition structures (r=1 or r=10) and the GSP, we adopt the following 2 * 3 design



Prediction 3

Small Group Win は Auction よりも Lottery においてより観察される



直感的な予想

勝利確率がグループの努力量に鋭敏に反応する競争形態（r=10）では、鈍感な競争形態（r=1）よりも、GSP は起きにくいのではなかろうか

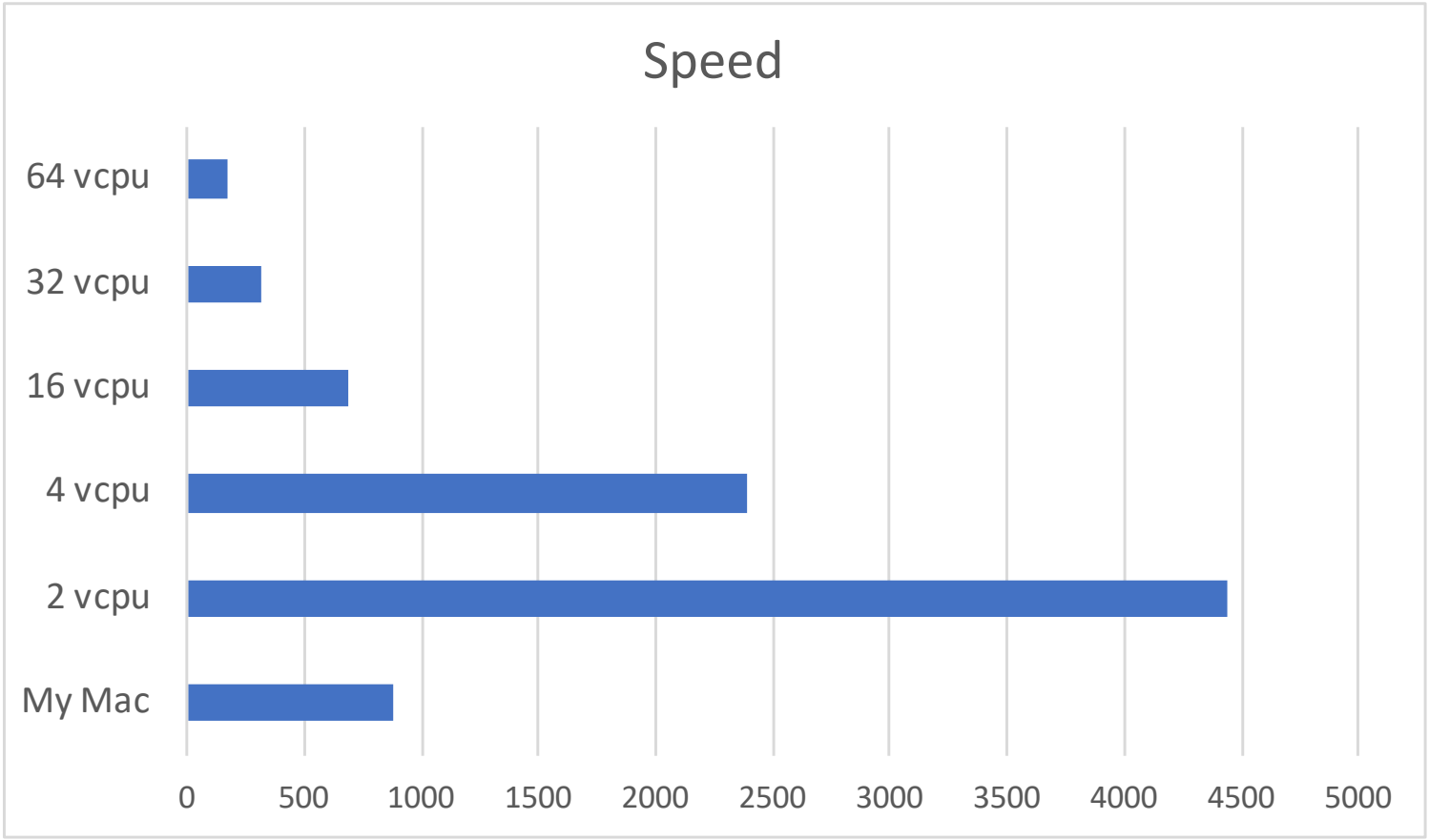
結論

ゲーム理論的な裏付けが得られた

さらなる速さを目指して

100戦略のケースを計算したい

- Microsoft Azure を試す（Thanks to Hayashi）



さらなる速さを目指して

100戦略のケースを計算したい

スパコンで計算する？

そもそもすべてを C やフォートランで計算した方が早い？