Autonomous Algorithmic Collusion

Efficient Learning with Q-Function Approximation

Lindsay Robinson

University of Melbourne

Motivation

Autonomous Algorithmic Collusion

'Autonomous Algorithmic Collusion'

- Firms delegate pricing to reinforcment learning algorithms
- Algorithms learn to tacitly collude through repeated interaction

Reinforcement Learning Algorithms:

- Learn dynamic best response to non-learning competitors
- Strategically rationalizable if:
  1. Learn efficiently enough that losses from suboptimal play during learning period are negligible
  2. Learn to collude with learning competitors

	Non-Learning	**Learning**
Non-Learning	π, π	π, π_{BR}
Learning	π_{BR}, π	**? ?**

Existing Evidence:

- Tabular Q-learning algorithms
  - Calvano et al. (2020 AER; 2023 IJIO)
  - Klein (2021 RJE)
  - Asker, Fershtman & Pakes (2022 AEAPP; 2023 JEMS)
  - Johnson, Rhodes & Wildenbeest (2023 E)
  - Abada & Lambin (2023 MS)
- Deep Q-learning algorithms
  - Hettich (2021)
  - Schlechtinger et al. (2024)
  - Dawid, Harting & Neugart (2024)
  - Deng, Schiffer & Bichler (2025)

This Paper:

- Linear Q-learning algorithms
  - More efficient learning
    $\Longrightarrow$ Strategically rationalizable
    $\Longrightarrow$ Policy relevant

Repeated Pricing Games

- For firm f, at time period $t-1$:
  - State = Price history: $\boldsymbol{P}_{1:t-1}$
  - Action = Own price: p_{ft}
  - Reward = Own profit: π_{ft}

Q-Learning Pricing Algorithms

- Maintain estimate of conditional value function - the ‘Q-function’: $Q\left(p_{ft}|\boldsymbol{P}_{1:t-1}\right)$
  - Recursive Bellman equation:
    $$Q\left(p_{ft}|\boldsymbol{P}_{1:t-1}\right) := \underbrace{\pi_{ft}\left(p_{ft}\right)}_{\text{period profit}} + \delta \underbrace{\max_{p_{ft+1}} Q\left(p_{ft+1}|\boldsymbol{P}_{1:t}(p_{ft})\right)}_{\text{continuation profits}}$$
- Each period, observe $(\boldsymbol{P}_{1:t-1}, p_{ft}, \pi_{ft}, \boldsymbol{P}_{1:t})$ and update estimated $\widehat{Q}$:
  - Gradient step in direction that minimises Bellman error:
    $$\widehat{e}_b = \underbrace{\left(\pi_{ft} + \delta \max_{p_{ft+1}} \widehat{Q}(p_{ft+1}|\boldsymbol{P}_{1:t})\right)}_{\text{realisation}} - \underbrace{\widehat{Q}(p_{ft}|\boldsymbol{P}_{1:t-1})}_{\text{estimate}}$$

Tabular

- Slow, inefficient learning
  - No generalisation from similar experience
- Curse of dimensionality

$$Q(p_{ft}|\boldsymbol{P}_{1:t-1}) = \theta_{p_{ft}\boldsymbol{P}_{1:t-1}} \qquad \boldsymbol{\Theta} = \begin{bmatrix} \theta_{11} & \dots & \theta_{1|\boldsymbol{\mathcal{P}}|} \\ \vdots & \ddots & \vdots \\ \theta_{|\mathcal{P}|1} & \dots & \theta_{|\mathcal{P}||\boldsymbol{\mathcal{P}}|} \end{bmatrix}$$

Deep

- Delayed, outdated learning
  - Backpropogation stability requires experience bank
  - Optimize against old competitor strategies

$$Q(p_{ft}|\boldsymbol{P}_{1:t-1}) = f\left(\boldsymbol{\omega}, f\left(\boldsymbol{\omega}, \dots f\left(\boldsymbol{\omega}, \boldsymbol{\phi}(p_{ft}, \boldsymbol{P}_{1:t-1})\right)\right)\right)$$

Linear

- Efficient learning
  - Generalisation from similar experience
  - Parsimonious parametrization
- Linear update rule
  - Immediate learning, reactivity

$$Q(p_{ft}|\boldsymbol{P}_{1:t-1}) = \boldsymbol{\omega}' \cdot \boldsymbol{\phi}(p_{ft}, \boldsymbol{P}_{1:t-1}) = \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_W \end{bmatrix}' \cdot \begin{bmatrix} \phi_1(p_{ft}, \boldsymbol{P}_{1:t-1}) \\ \vdots \\ \phi_W(p_{ft}, \boldsymbol{P}_{1:t-1}) \end{bmatrix}$$

Motivation

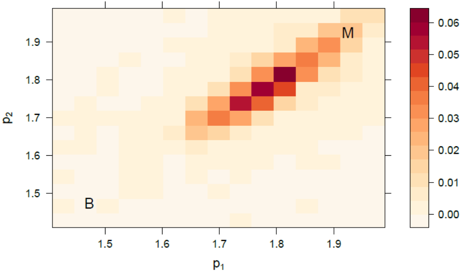
Q-Learning Pricing Algorithms - Q-Function Representations - Practical Significance

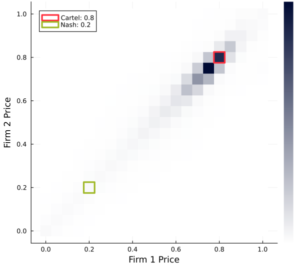
Calvano et al (2020 AER):

- **Q-Function:** Tabular
  - Less efficient learning
- **State:** Most recent prices
  - Myopic conditioning/information set
  - Limited strategy space
- **Initialisation:** Pre-trained offline
  - Requires ex-ante demand, competitor knowledge
- **Periods before Collusion:** 'Millions'
  - Extended algorithm commitment

Current Paper:

- **Q-Function:** Linear
  - More efficient learning
- **State:** Price history
  - Full information set
  - Richer strategy space
- **Initialisation:** Naive
  - No required demand, competitor knowledge
- **Periods before Collusion:** $\sim$3000
  - Shorter algorithm commitment





Setup

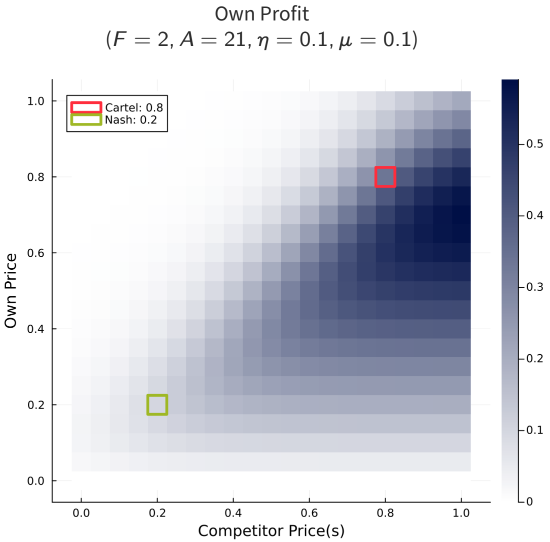
Strategic Environment

Strategic Environment:

- Firms: $f = 1, 2, \ldots, F$
- Time periods: $t = 1, 2, \ldots, T$
- Action space: $p \in \mathcal{P} = \{0, \ldots, 1\}$
  - $|\mathcal{P}| = A$
- Marginal costs: $c_f = 0 \, \forall f$
- Logit demand:

$$\pi_f(p_f, \mathbf{p}_{f^-}) = p_f \cdot \frac{\exp\left(\frac{1-p_f}{\mu}\right)}{\exp\left(\frac{\eta}{\mu}\right) + \sum_{f=1}^{F} \exp\left(\frac{1-p_f}{\mu}\right)}$$

  - μ: Market power
    - Product differentiation
  - η: Aggregate demand elasticity
    - Outside good strength



Setup

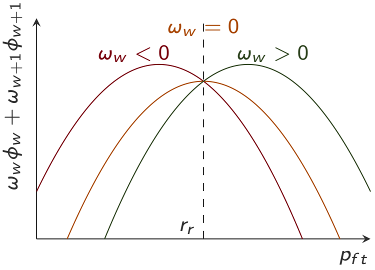
Q-Function Approximation

- Linear Q-function with features:

$$\boldsymbol{\phi}(\boldsymbol{P}_{1:t-1}, p_{ft}) = \begin{bmatrix} \vdots \\ 1[p_{ft} = p_a] \\ \vdots \\ p_{ft} - r_r(\boldsymbol{P}_{1:t-1}) \\ \left(p_{ft} - r_r(\boldsymbol{P}_{1:t-1})\right)^2 \\ \vdots \end{bmatrix} \approx \begin{bmatrix} \vdots \\ \text{Context invariant price indicators} \\ \vdots \\ \text{Quadratic relative pricing rules} \\ \vdots \end{bmatrix}$$

$$\widehat{Q} = \boldsymbol{\omega}' \cdot \boldsymbol{\phi} = \begin{bmatrix} \vdots \\ \omega_w \\ \omega_{w+1} \\ \vdots \end{bmatrix}' \cdot \begin{bmatrix} \vdots \\ p_{ft} - r_r(\boldsymbol{P}_{1:t-1}) \\ \left(p_{ft} - r_r(\boldsymbol{P}_{1:t-1})\right)^2 \\ \vdots \end{bmatrix}$$

$(\omega_{w+1} < 0)$



- Price history reference points $r_1, \ldots, r_R$:
  - Most recent prices
    - Own
    - Competitor mean
    - Competitor minimum
  - Recency weighted average prices
    - Own
    - Competitor mean
    - Competitor minimum

Setup

Algorithm Hyperparameters

Discounting

- δ : Discount rate
  - Governs intertemporal tradeoffs

$$Q(p_{f\,t}|\boldsymbol{P}_{1:t-1}) = \pi_{f\,t} + \delta \max_{p_{f\,t+1}} Q(p_{f\,t+1}|\boldsymbol{P}_{1:t})$$

Exploration

- ε : Exploration rate
  - Determines how often the algorithm chooses a random price

$$p_{f\,t}(\boldsymbol{P}_{1:t-1}) = \begin{cases} \arg\max_{p_{f\,t}} Q(p_{f\,t}|\boldsymbol{P}_{1:t-1}) & \text{if: } \exp(-\frac{t}{\varepsilon}) < U[0,1] \\ p \in_R \mathcal{P} & \text{otherwise} \end{cases}$$

Learning

- λ : Learning rate
  - Determines how quickly old experience is replaced with new experience

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + (\lambda \cdot e_b \cdot \boldsymbol{\phi})$$

Price Memory

- ρ : Recency weighting
  - Determines how much weight older prices receive in state representation

$$\overline{\boldsymbol{p}_t} = \rho\overline{\boldsymbol{p}_{t-1}} + (1-\rho)\boldsymbol{p}_t$$

Setup

Baseline & Robustness

Simulations

- Simulations: $S = \begin{bmatrix} 1 & 1000 \end{bmatrix}$
- Time Periods: $T = \begin{bmatrix} 1000 & 2000 & 3000 & 4000 & 10000 \end{bmatrix}$

Strategic Environment

- Actions: $A = \begin{bmatrix} 21 & 41 \end{bmatrix}$
- Firms: $F = \begin{bmatrix} 2 & 3 & 4 \end{bmatrix}$
- Differentiation: $\mu = \begin{bmatrix} 0.02 & 0.04 & 0.06 & 0.08 & 0.10 \end{bmatrix}$

Algorithm Parameters

- Discount Rate: $\delta = \begin{bmatrix} 0.2 & 0.4 & 0.6 & 0.8 & 0.9 \end{bmatrix}$
- Exploration Intensity: $\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$
- Learning Rate: $\lambda = \begin{bmatrix} 0.08 & 0.12 & 0.16 & 0.20 & 0.24 \end{bmatrix}$
- Price Memory: $\rho = \begin{bmatrix} 0.00 & 0.05 & 0.10 & 0.25 & 0.50 \end{bmatrix}$

Results

Baseline - Single Simulation

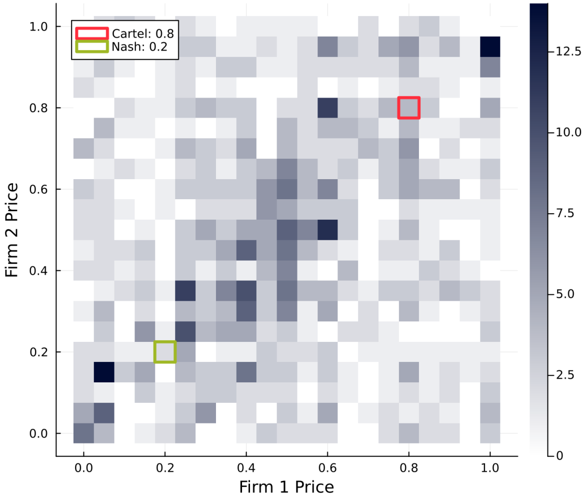
Simulations:

$$S = \begin{bmatrix} 1 & 1000 \end{bmatrix}$$

Time Periods:

$$T = \begin{bmatrix} 1000 & 2000 & 3000 & 4000 & 10000 \end{bmatrix}$$

- Algorithms explore, then price match, then collude



Results

Baseline - Single Simulation

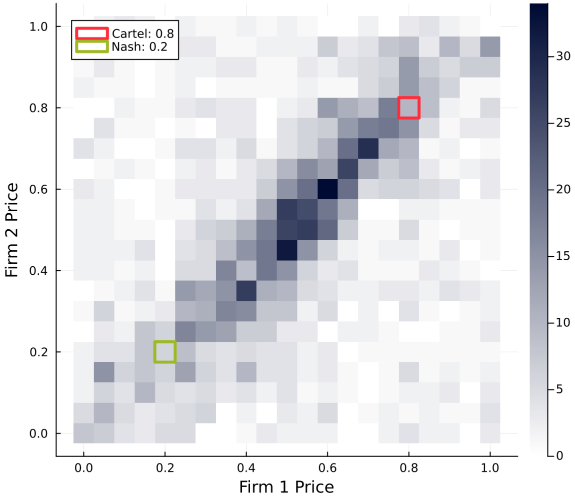
Simulations:

$$S = [1 \quad 1000]$$

Time Periods:

$$T = [1000 \quad 2000 \quad 3000 \quad 4000 \quad 10000]$$

- Algorithms explore, then price match, then collude



Results

Baseline - Single Simulation

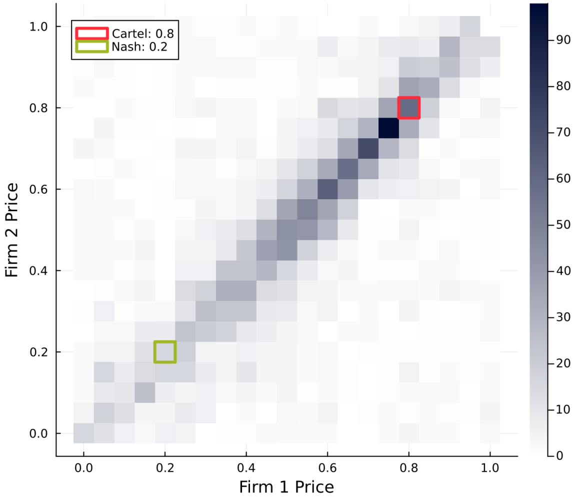
Simulations:

$$S = [1 \quad 1000]$$

Time Periods:

$$T = [1000 \quad 2000 \quad 3000 \quad 4000 \quad 10000]$$

- Algorithms explore, then price match, then collude



Results

Baseline - Single Simulation

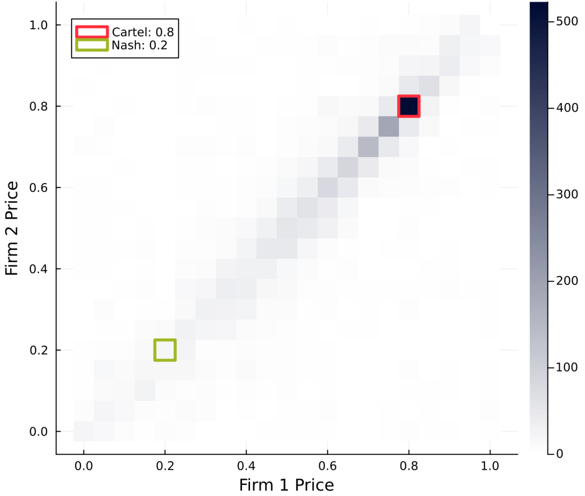
Simulations:

$$S = \begin{bmatrix} 1 & 1000 \end{bmatrix}$$

Time Periods:

$$T = \begin{bmatrix} 1000 & 2000 & 3000 & 4000 & 10000 \end{bmatrix}$$

- Algorithms explore, then price match, then collude



Results

Baseline - Single Simulation

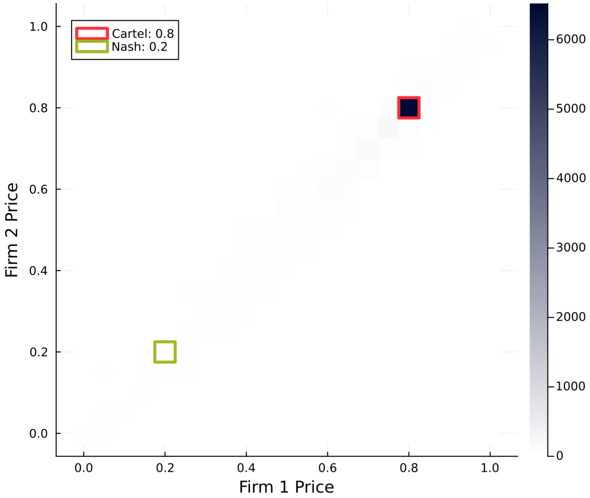
Simulations:

$$S = [1 \quad 1000]$$

Time Periods:

$$T = [1000 \quad 2000 \quad 3000 \quad 4000 \quad 10000]$$

- Algorithms explore, then price match, then collude



Results

Baseline - Empirical Distribution

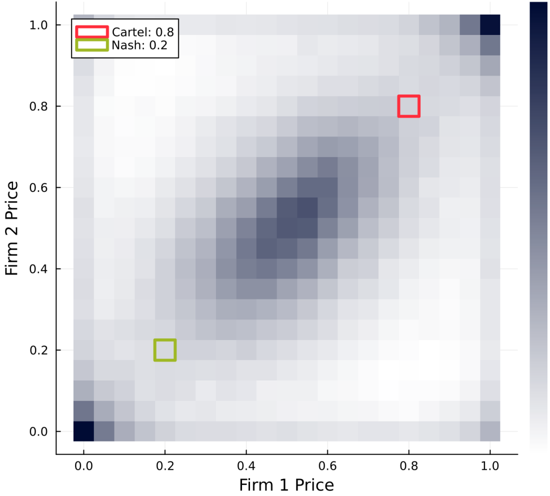
Simulations:

$$S = \begin{bmatrix} 1 & 1000 \end{bmatrix}$$

Time Periods:

$$T = \begin{bmatrix} 1000 & 2000 & 3000 & 4000 & 10000 \end{bmatrix}$$

- Across 1000 simulations, similar action profiles
- Algorithms explore, then price match, then collude



Results

Baseline - Empirical Distribution

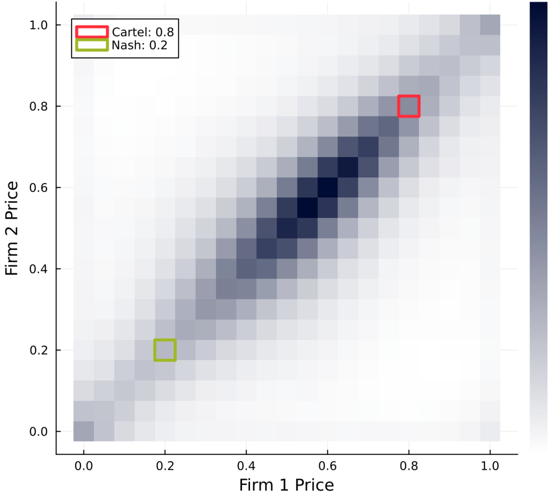
Simulations:

$S = [\color{lightgray}{1} \quad 1000]$

Time Periods:

$T = [\color{lightgray}{1000} \quad 2000 \quad \color{lightgray}{3000} \quad \color{lightgray}{4000} \quad \color{lightgray}{10000}]$

- Across 1000 simulations, similar action profiles
- Algorithms explore, then price match, then collude



Results

Baseline - Empirical Distribution

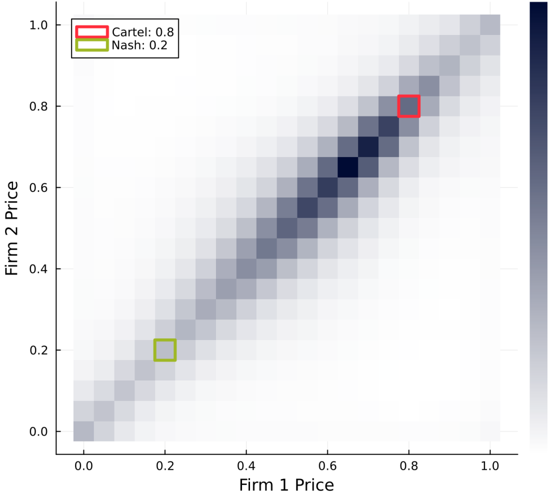
Simulations:

$$S = \begin{bmatrix} 1 & 1000 \end{bmatrix}$$

Time Periods:

$$T = \begin{bmatrix} 1000 & 2000 & 3000 & 4000 & 10000 \end{bmatrix}$$

- Across 1000 simulations, similar action profiles
- Algorithms explore, then price match, then collude



Results

Baseline - Empirical Distribution

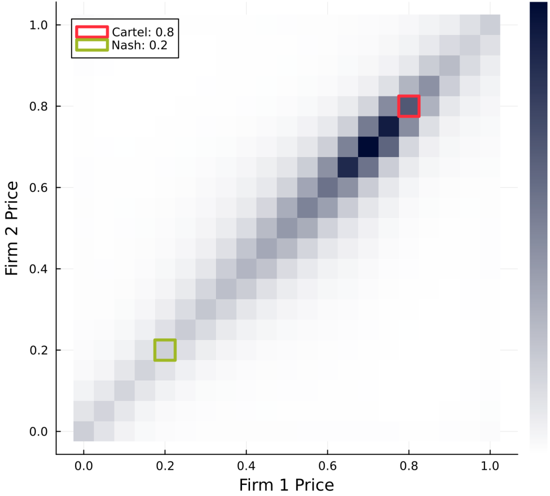
Simulations:

$$S = \begin{bmatrix} 1 & 1000 \end{bmatrix}$$

Time Periods:

$$T = \begin{bmatrix} 1000 & 2000 & 3000 & 4000 & 10000 \end{bmatrix}$$

- Across 1000 simulations, similar action profiles
- Algorithms explore, then price match, then collude



Results

Baseline - Empirical Distribution

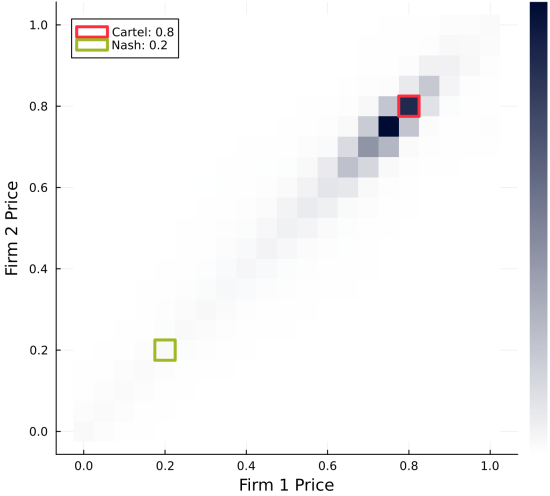
Simulations:

$$S = \begin{bmatrix} 1 & 1000 \end{bmatrix}$$

Time Periods:

$$T = \begin{bmatrix} 1000 & 2000 & 3000 & 4000 & 10000 \end{bmatrix}$$

- Across 1000 simulations, similar action profiles
- Algorithms explore, then price match, then collude



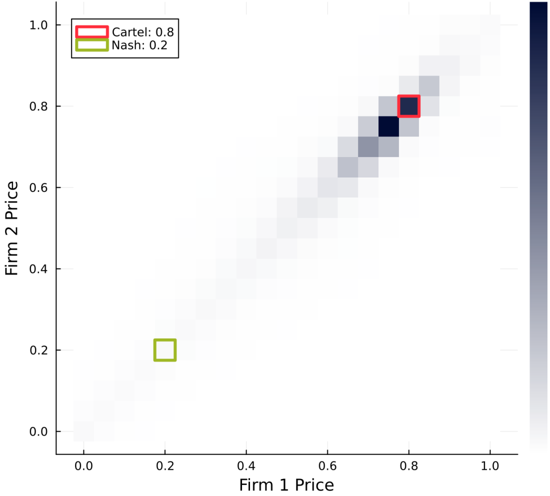
Results

Robustness - Algorithm Parametrization - Discounting

Discount Factor:

$$\delta = \begin{bmatrix} 0.20 & 0.50 & 0.80 & 0.85 & 0.90 \end{bmatrix}$$

- Sufficiently high discount factor required to sustain collusion
  - Consistent with theory on critical discount factors



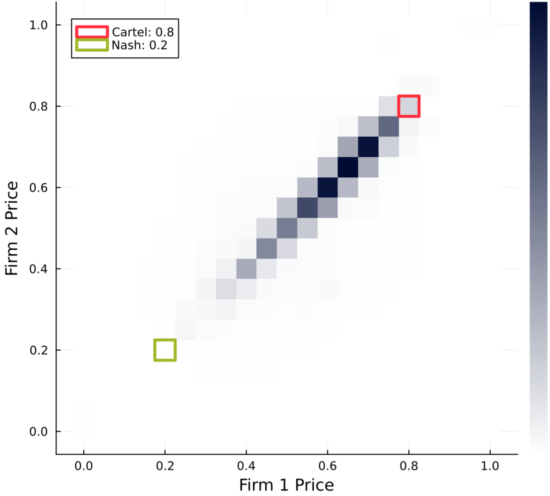
Results

Robustness - Algorithm Parametrization - Discounting

Discount Factor:

$\delta = [0.20 \quad 0.50 \quad 0.80 \quad 0.85 \quad 0.90]$

- Sufficiently high discount factor required to sustain collusion
  - Consistent with theory on critical discount factors



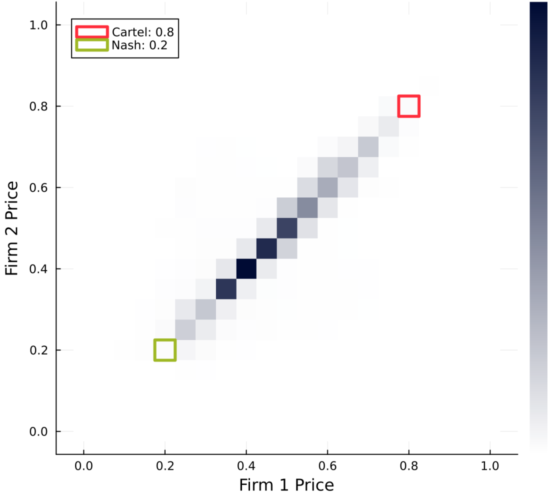
Results

Robustness - Algorithm Parametrization - Discounting

Discount Factor:

$$\delta = \begin{bmatrix} 0.20 & 0.50 & 0.80 & 0.85 & 0.90 \end{bmatrix}$$

- Sufficiently high discount factor required to sustain collusion
  - Consistent with theory on critical discount factors



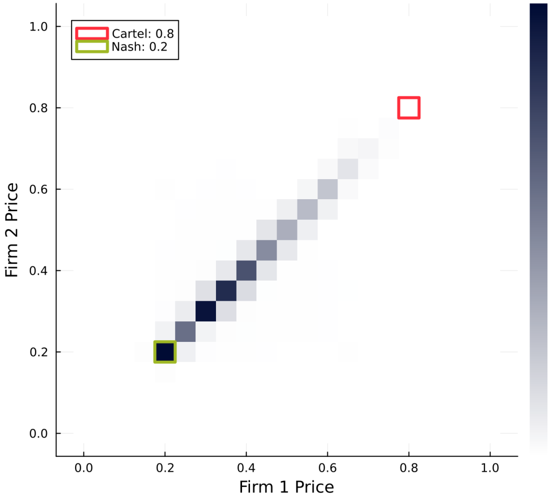
Results

Robustness - Algorithm Parametrization - Discounting

Discount Factor:

$\delta = [0.20 \quad 0.50 \quad 0.80 \quad 0.85 \quad 0.90]$

- Sufficiently high discount factor required to sustain collusion
  - Consistent with theory on critical discount factors



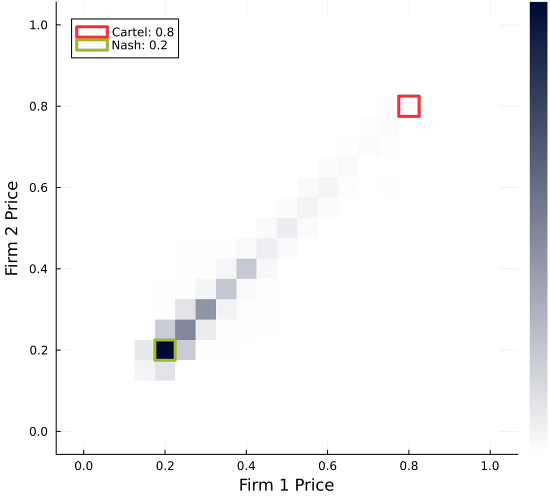
Results

Robustness - Algorithm Parametrization - Discounting

Discount Factor:

$\delta = [0.20 \quad 0.50 \quad 0.80 \quad 0.85 \quad 0.90]$

- Sufficiently high discount factor required to sustain collusion
  - Consistent with theory on critical discount factors



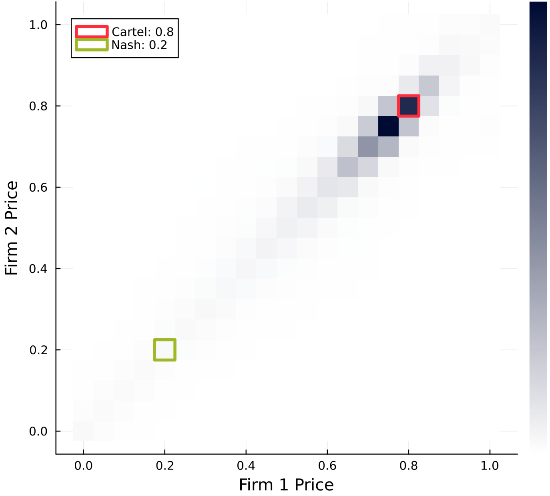
Results

Robustness - Algorithm Parametrization - Price Memory

Price Memory:

$\rho = \begin{bmatrix} 0.00 & 0.05 & 0.10 & 0.25 & 0.50 \end{bmatrix}$

- By design, algorithms intermittently deviate from collusive prices
- Collusive stability requires distinguishing temporary and sustained deviation
    - Requires long-memory state representation



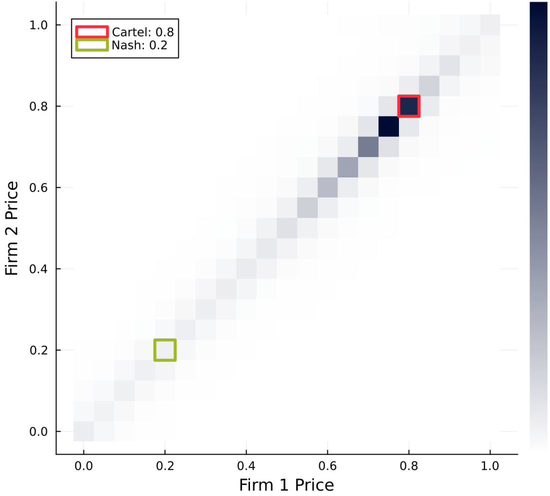
Results

Robustness - Algorithm Parametrization - Price Memory

Price Memory:

$$\rho = \begin{bmatrix} 0.00 & 0.05 & 0.10 & 0.25 & 0.50 \end{bmatrix}$$

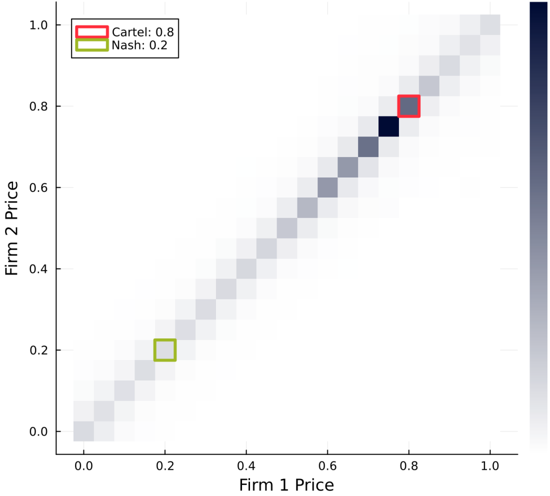
- By design, algorithms intermittently deviate from collusive prices
- Collusive stability requires distinguishing temporary and sustained deviation
  - Requires long-memory state representation



Price Memory:

$$\rho = \begin{bmatrix} 0.00 & 0.05 & 0.10 & 0.25 & 0.50 \end{bmatrix}$$

- By design, algorithms intermittently deviate from collusive prices
- Collusive stability requires distinguishing temporary and sustained deviation
  - Requires long-memory state representation



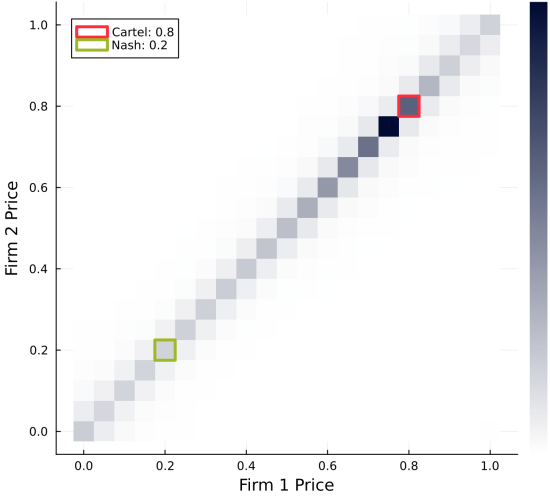
Results

Robustness - Algorithm Parametrization - Price Memory

Price Memory:

$$\rho = \begin{bmatrix} 0.00 & 0.05 & 0.10 & 0.25 & 0.50 \end{bmatrix}$$

- By design, algorithms intermittently deviate from collusive prices
- Collusive stability requires distinguishing temporary and sustained deviation
  - Requires long-memory state representation



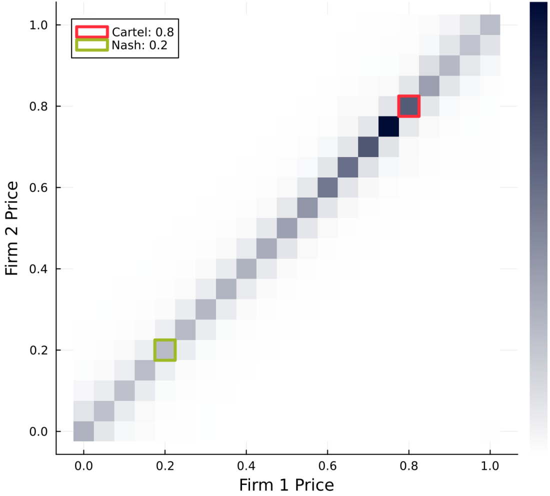
Results

Robustness - Algorithm Parametrization - Price Memory

Price Memory:

$$\rho = \begin{bmatrix} 0.00 & 0.05 & 0.10 & 0.25 & 0.50 \end{bmatrix}$$

- By design, algorithms intermittently deviate from collusive prices
- Collusive stability requires distinguishing temporary and sustained deviation
  - Requires long-memory state representation



Results

Robustness - Algorithm Parametrization - Exploration & Learning

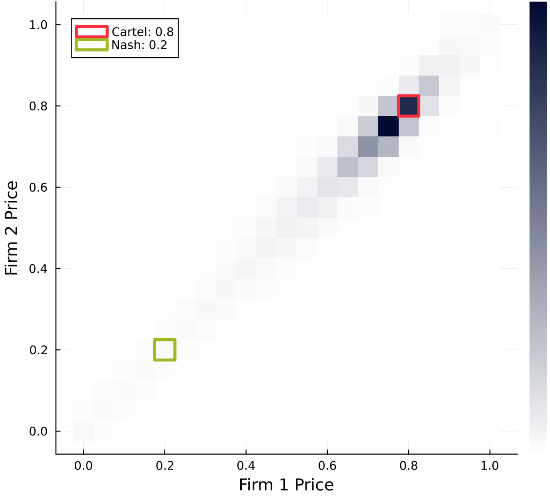
Exploration:

$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.18 & 0.20 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ
- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

Exploration:

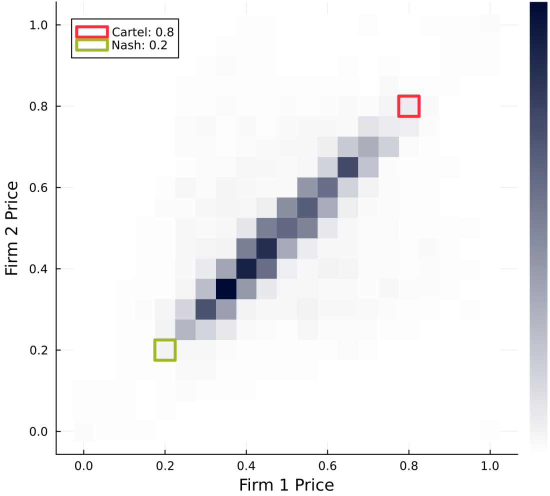
$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.18 & 0.20 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ

- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

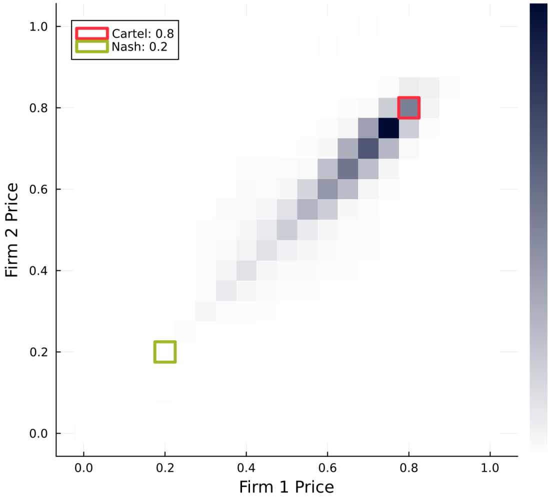
Exploration:

$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.18 & 0.20 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ
- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

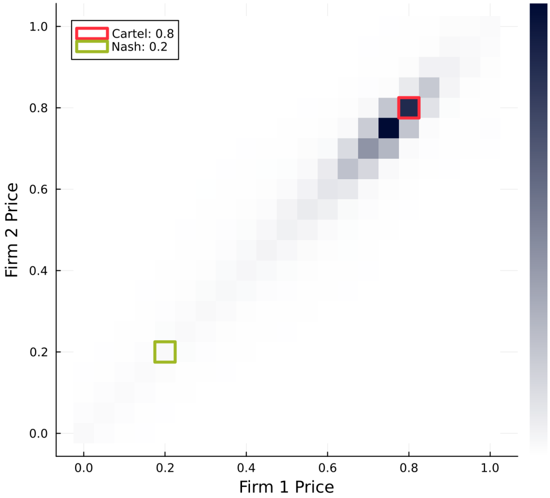
Exploration:

$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.18 & 0.20 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ
- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

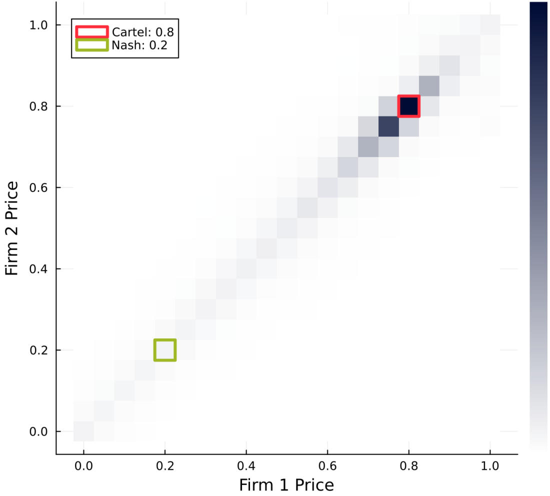
Exploration:

$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.18 & 0.20 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ
- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

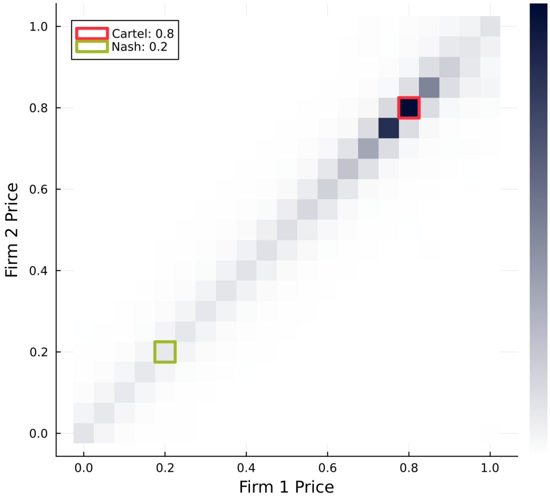
Exploration:

$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.18 & 0.20 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ
- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

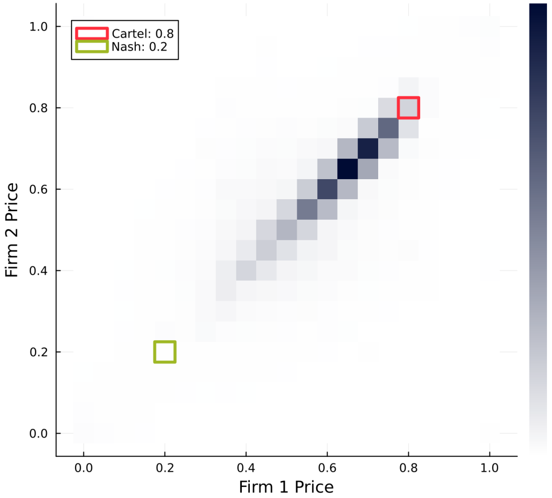
Exploration:

$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.20 & 0.24 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ
- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

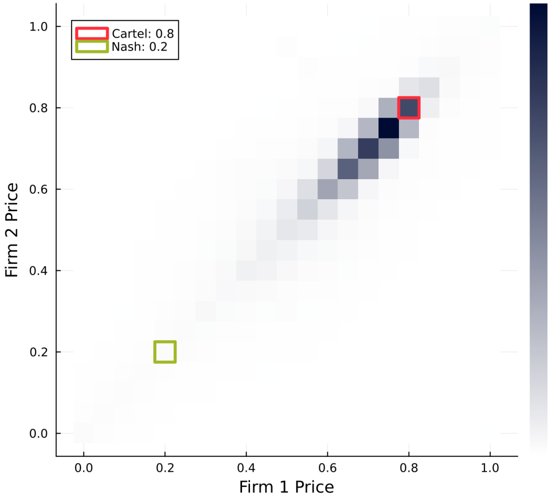
Exploration:

$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.20 & 0.24 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ
- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

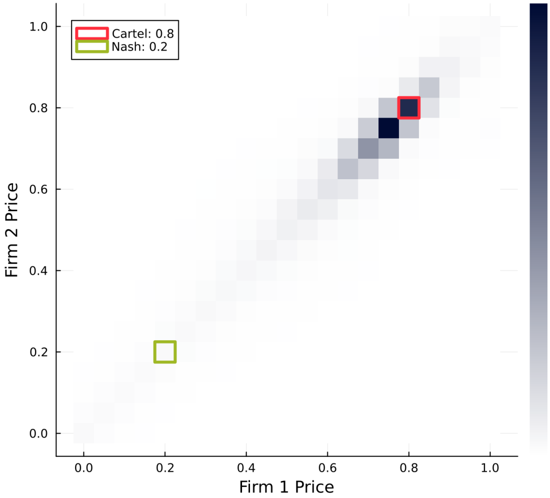
Exploration:

$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.20 & 0.24 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ
- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

Exploration:

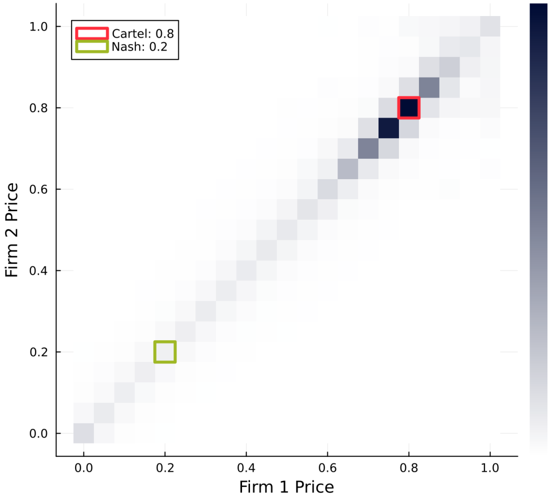
$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.20 & 0.24 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ

- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

Exploration:

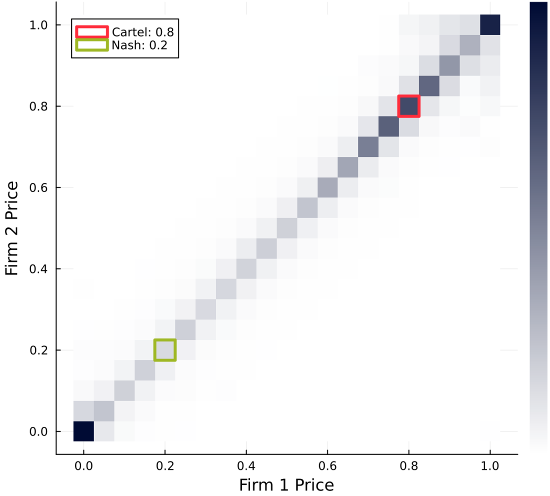
$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.20 & 0.24 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ

- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

Exploration:

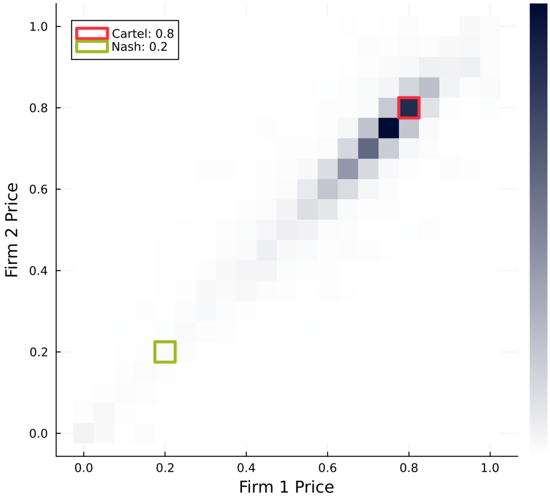
$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.20 & 0.24 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ

- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

Exploration:

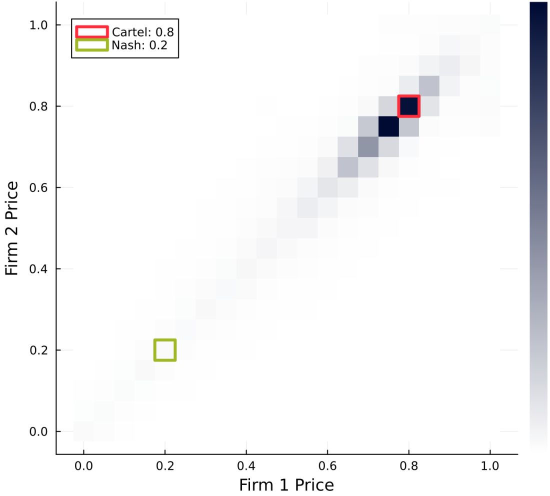
$$\varepsilon = [200 \quad 300 \quad 400 \quad 500 \quad 600]$$

Learning Rate:

$$\lambda = [0.12 \quad 0.14 \quad 0.16 \quad 0.20 \quad 0.24]$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ

- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

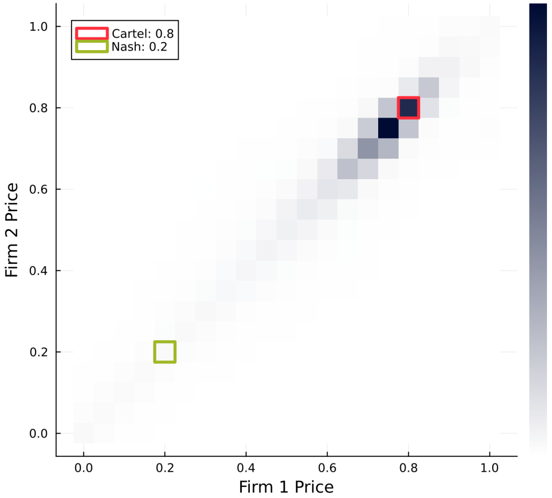
Exploration:

$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.20 & 0.24 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ
- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

Exploration:

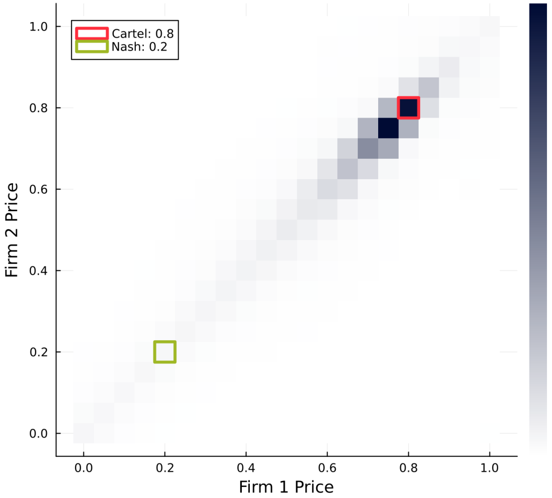
$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.20 & 0.24 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ

- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning

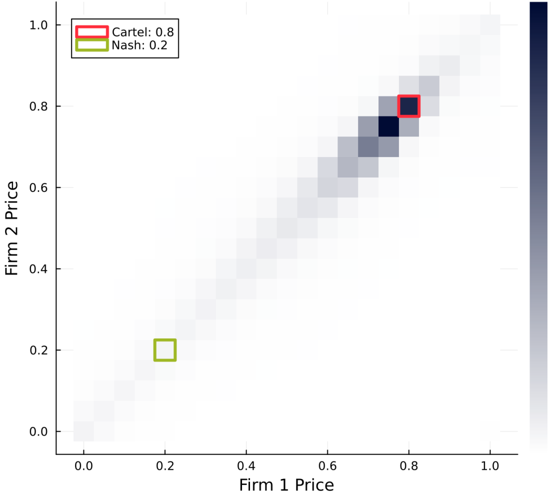
Exploration:

$$\varepsilon = \begin{bmatrix} 200 & 300 & 400 & 500 & 600 \end{bmatrix}$$

Learning Rate:

$$\lambda = \begin{bmatrix} 0.12 & 0.14 & 0.16 & 0.20 & 0.24 \end{bmatrix}$$

- Learning efficiency determinants:
  - Exploration frequency: ε
  - Learning rate: λ
- Efficient learning required to sustain collusion:
  - High λ, low ε
  - Low λ, high ε



Results

Robustness - Algorithm Parametrization - Exploration & Learning - Asymmetric Firms

Exploration:

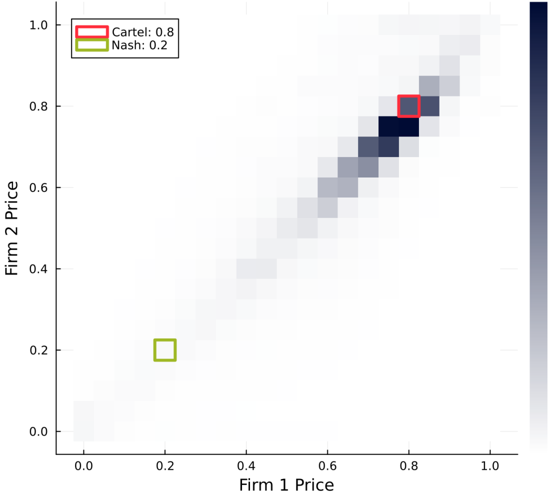
$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix} = \begin{bmatrix} 200 & 400 & 600 \\ 200 & 400 & 600 \end{bmatrix}$$

Learning Rate:

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 0.12 & 0.16 & 0.24 \\ 0.12 & 0.16 & 0.24 \end{bmatrix}$$

- Ex-ante strategic parametrization:
  - Faster learning algorithm exploits slower learning algorithm

- Asymmetric collusive outcomes still persist



Results

Robustness - Algorithm Parametrization - Exploration & Learning - Asymmetric Firms

Exploration:

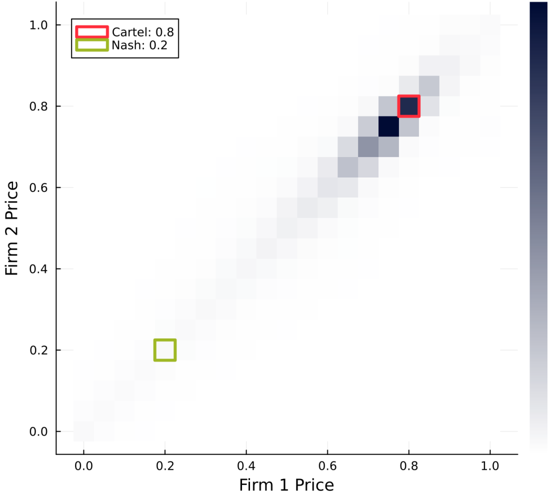
$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix} = \begin{bmatrix} 200 & 400 & 600 \\ 200 & 400 & 600 \end{bmatrix}$$

Learning Rate:

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 0.12 & 0.16 & 0.24 \\ 0.12 & 0.16 & 0.24 \end{bmatrix}$$

- Ex-ante strategic parametrization:
  - Faster learning algorithm exploits slower learning algorithm

- Asymmetric collusive outcomes still persist



Results

Robustness - Algorithm Parametrization - Exploration & Learning - Asymmetric Firms

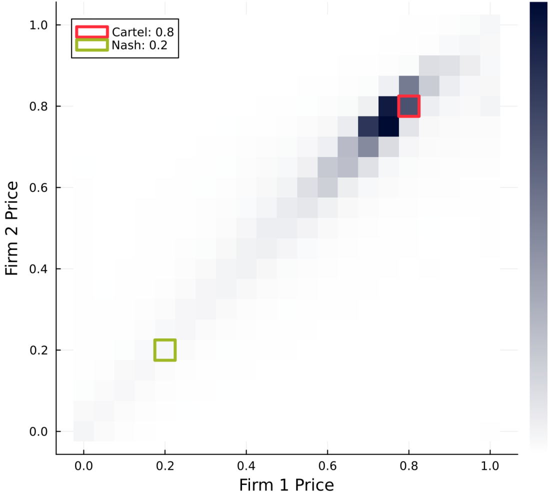
Exploration:

$$\begin{bmatrix}\varepsilon_1 \\ \varepsilon_2\end{bmatrix} = \begin{bmatrix}200 & 400 & 600 \\ 200 & 400 & 600\end{bmatrix}$$

Learning Rate:

$$\begin{bmatrix}\lambda_1 \\ \lambda_2\end{bmatrix} = \begin{bmatrix}0.12 & 0.16 & 0.24 \\ 0.12 & 0.16 & 0.24\end{bmatrix}$$

- Ex-ante strategic parametrization:
  - Faster learning algorithm exploits slower learning algorithm
- Asymmetric collusive outcomes still persist



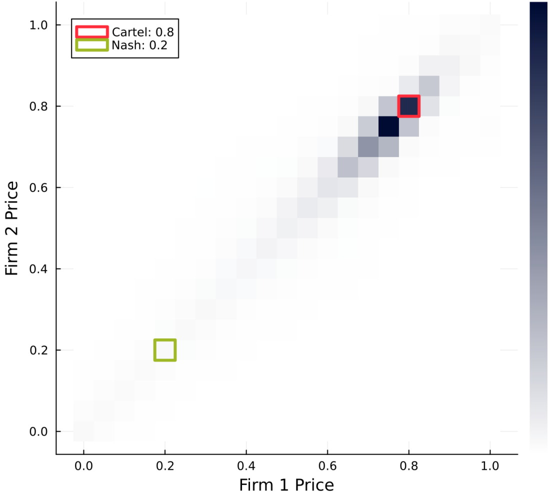
Results

Robustness - Strategic Environment - Action Space

Action Space:

$$A = \begin{bmatrix} 21 & 41 \end{bmatrix}$$

- Collusive outcomes robust to granularity of action space



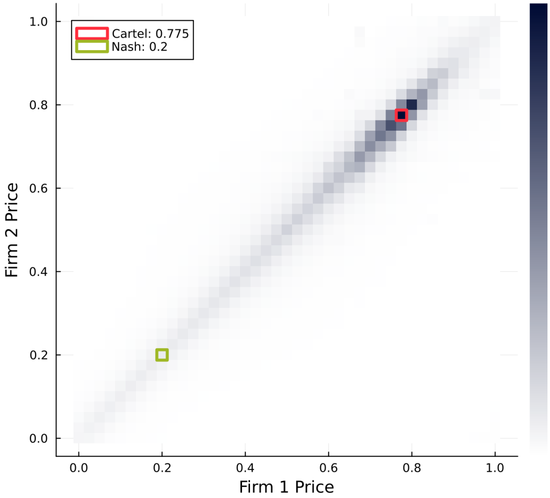
Results

Robustness - Strategic Environment - Action Space

Action Space:

$$A = \begin{bmatrix} 21 & 41 \end{bmatrix}$$

- Collusive outcomes robust to granularity of action space



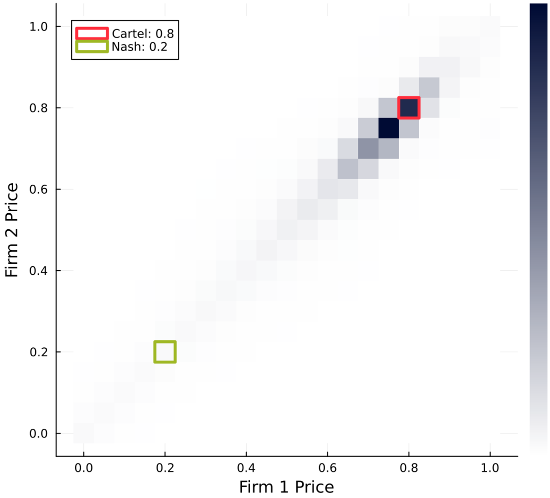
Results

Robustness - Strategic Environment - Product Differentiation

Product Differentiation:

$$\mu = \begin{bmatrix} 0.02 & 0.04 & 0.06 & 0.08 & 0.10 \end{bmatrix}$$

- Collusive outcomes robust to degree of product differentiation
- Product homogeneity increases price symmetry
  - Profits more sensitive to being undercut



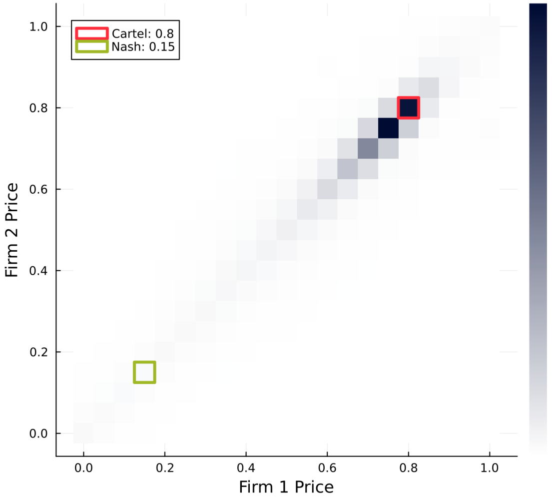
Results

Robustness - Strategic Environment - Product Differentiation

Product Differentiation:

$$\mu = [0.02 \quad 0.04 \quad 0.06 \quad 0.08 \quad 0.10]$$

- Collusive outcomes robust to degree of product differentiation
- Product homogeneity increases price symmetry
  - Profits more sensitive to being undercut



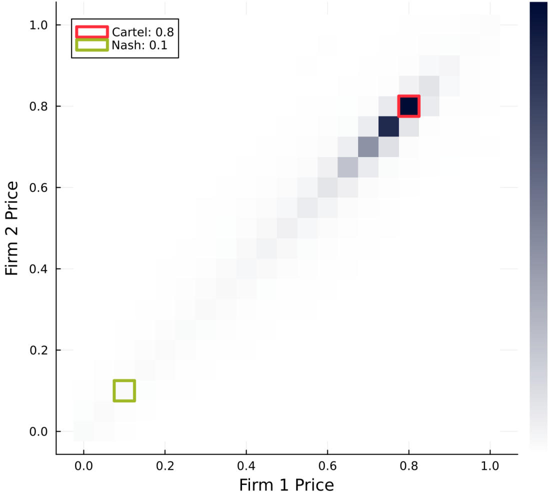
Results

Robustness - Strategic Environment - Product Differentiation

Product Differentiation:

$\mu = [0.02 \quad 0.04 \quad 0.06 \quad 0.08 \quad 0.10]$

- Collusive outcomes robust to degree of product differentiation
- Product homogeneity increases price symmetry
  - Profits more sensitive to being undercut



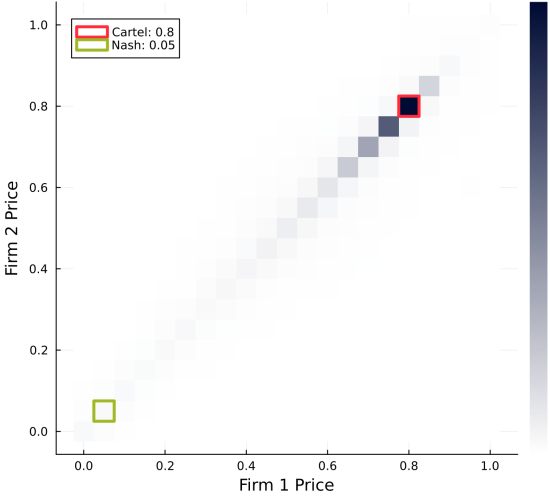
Results

Robustness - Strategic Environment - Product Differentiation

Product Differentiation:

$$\mu = [0.02 \quad 0.04 \quad 0.06 \quad 0.08 \quad 0.10]$$

- Collusive outcomes robust to degree of product differentiation
- Product homogeneity increases price symmetry
  - Profits more sensitive to being undercut



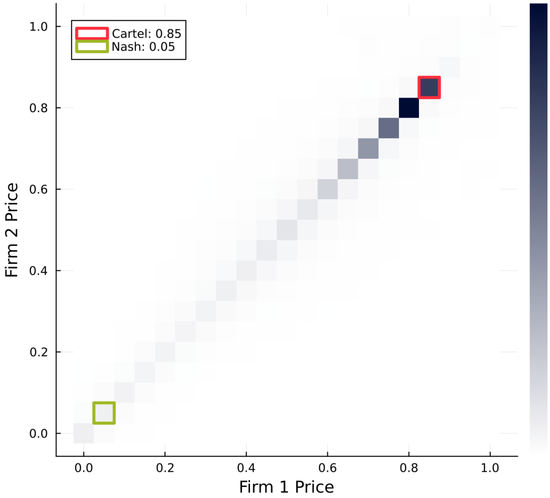
Results

Robustness - Strategic Environment - Product Differentiation

Product Differentiation:

$$\mu = \begin{bmatrix} 0.02 & 0.04 & 0.06 & 0.08 & 0.10 \end{bmatrix}$$

- Collusive outcomes robust to degree of product differentiation
- Product homogeneity increases price symmetry
  - Profits more sensitive to being undercut



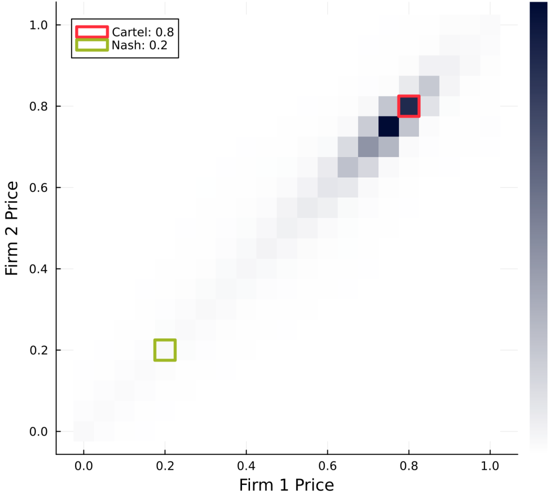
Results

Robustness - Strategic Environment - Firms

Firms:

$$F = \begin{bmatrix} 2 & 3 & 4 \end{bmatrix}$$

- Difficult to sustain collusion with 3+ firms intermittently deviating
- Consistent with:
  - Theoretical comparative statics
  - Antitrust coordinated effects rule of thumb



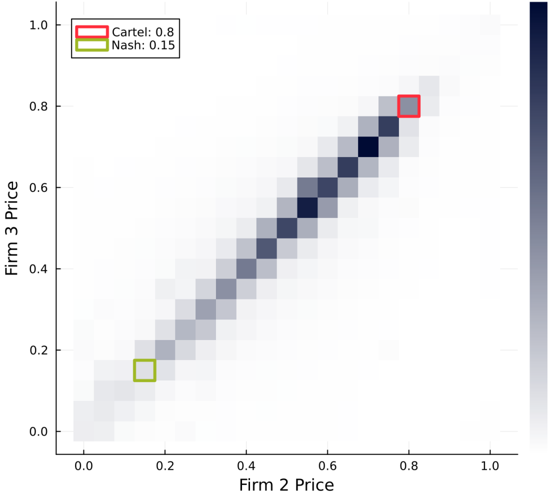
Results

Robustness - Strategic Environment - Firms

Firms:

$$F = \begin{bmatrix} 2 & 3 & 4 \end{bmatrix}$$

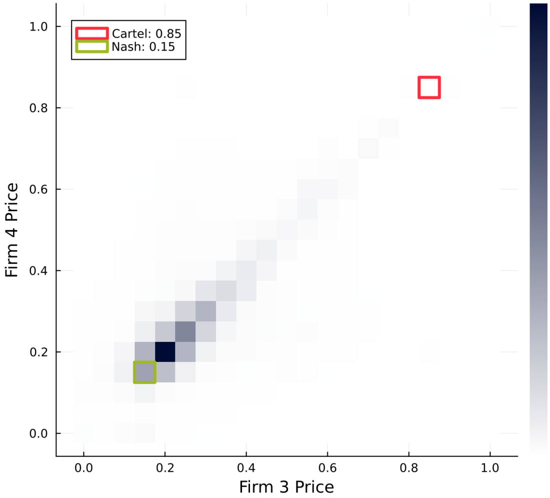
- Difficult to sustain collusion with 3+ firms intermittently deviating
- Consistent with:
  - Theoretical comparative statics
  - Antitrust coordinated effects rule of thumb



Firms:

$$F = \begin{bmatrix} 2 & 3 & 4 \end{bmatrix}$$

- Difficult to sustain collusion with 3+ firms intermittently deviating
- Consistent with:
  - Theoretical comparative statics
  - Antitrust coordinated effects rule of thumb



Conclusion

Summary

- Q-function approximation $\rightarrow$ efficient learning
  - More perfect collusion
  - Stable collusion after $\sim$3000 time periods
- Collusion most sensitive to number of firms
  - Difficult to coordinate with 3+ firms intermittently deviating
- Sensible algorithm parameters required for collusion
  - Sufficient discount factor
  - Sufficient price memory
  - Efficient learning & exploration