Autonomous Algorithmic Collusion

Efficient Learning with Q-Function Approximation

Lindsay Robinson

University of Melbourne

Motivation

Autonomous Algorithmic Collusion

'Autonomous Algorithmic Collusion'

- Firms delegate pricing strategies to reinforcement learning algorithms
- Algorithms learn to tacitly collude with one another through repeated interaction

Existing Evidence:

- Less efficient learning algorithms
  - Larger number of parameters
  - Lower dimensional state representation
- Myopic strategic conditioning set

→ Slower collusive coordination

→ Imperfect collusive prices

This Paper:

- More efficient learning algorithms
  - Smaller number of parameters
  - Higher dimensional state representation
- History-aware strategic conditioning set

→ Faster collusive coordination

→ More perfect collusive prices

Q-Learning Pricing Algorithms

- For firm f, at time period $t-1$:
  - State = Price history: $\boldsymbol{P}_{1:t-1}$
  - Action = Own price: p_{ft}
  - Conditional Value Function: $Q(p_{ft}|\boldsymbol{P}_{1:t-1})$

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

This Paper

- Linear Q-learning algorithms

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

- Slow, inefficient learning
  - No generalization from similar experience
- Limited strategic conditioning set
  - Curse of dimensionality in state representation

Q-Learning Pricing Algorithms

- For firm f, at time period $t-1$:
  - State = Price history: $\boldsymbol{P}_{1:t-1}$
  - Action = Own price: p_{ft}
  - Conditional Value Function: $Q(p_{ft}|\boldsymbol{P}_{1:t-1})$

Existing Evidence

- Tabular Q-learning algorithms
- Deep Q-learning algorithms ▶
  - Hettich (2021)
  - Schlechtinger et al. (2024)
  - Dawid, Harting & Neugart (2024)
  - Deng, Schiffer & Bichler (2025)

This Paper

- Linear Q-learning algorithms

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

- Slow learning
  - Many parameters
- Delayed learning
  - Backpropogation stability requires experience bank
  - Optimize against old competitor strategies

Q-Learning Pricing Algorithms

- For firm f, at time period $t-1$:
  - State = Price history: $\boldsymbol{P}_{1:t-1}$
  - Action = Own price: p_{ft}
  - Conditional Value Function: $Q(p_{ft}|\boldsymbol{P}_{1:t-1})$

Existing Evidence

- Tabular Q-learning algorithms
- Deep Q-learning algorithms

This Paper

- Linear Q-learning algorithms ▶

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

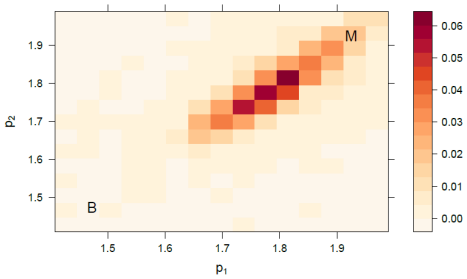
- Efficient learning
  - Parsimonious linear parametrization
- Immediate learning, reactivity
  - Linear update rule

Literature

Q-Function Representations - Practical Significance

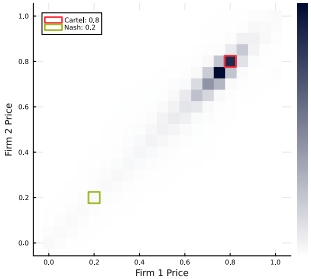
Calvano et al (2020 AER): Tabular Q-Functions

- **State:** Most recent prices
  - Myopic strategic conditioning set
  - Limited strategy space
- **Initialisation:** 'Pre-trained' offline
  - Requires ex-ante demand, competitor knowledge
- **Periods before Collusion:** 400,000-'Millions'
  - Extended algorithm commitment
- **Collusive perfection:**



Current Paper: Linear Q-Functions

- **State:** Price history
  - Long-memory strategic conditioning set
  - Richer strategy space
- **Initialisation:** Naive
  - No required demand, competitor knowledge
- **Periods before Collusion:** ∼3,000
  - Shorter algorithm commitment
- **Collusive perfection:**



Setup

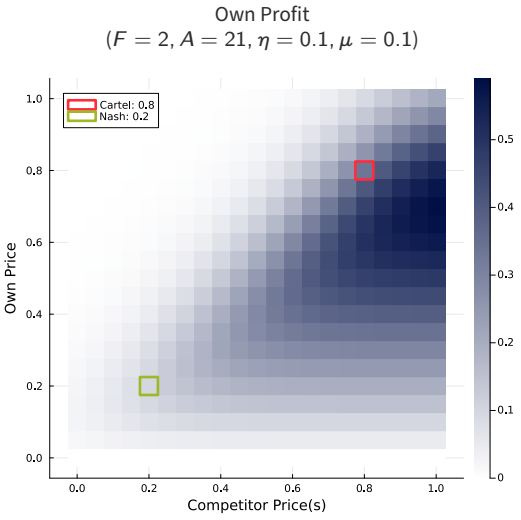
Strategic Environment

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:

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

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

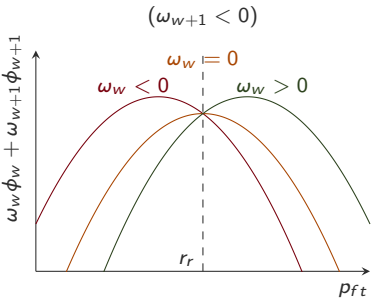


- Linear Q-functions with features:

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

- 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

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



Setup

Algorithm Hyperparameters

Discounting

- δ : Discount factor
  - Governs intertemporal tradeoffs

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

Exploration

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

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

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: $S = [1 \quad 1000]$
- Time Periods: $T = [1000 \quad 2000 \quad 3000 \quad 4000 \quad 10000]$

Strategic Environment

- Actions: $A = [21 \quad 41]$
- Firms: $F = [2 \quad 3 \quad 4]$
- Differentiation: $\mu = [0.02 \quad 0.04 \quad 0.06 \quad 0.08 \quad 0.10]$

Algorithm Parameters

- Discount Factor: $\delta = [0.20 \quad 0.50 \quad 0.80 \quad 0.85 \quad 0.90]$
- Exploration Frequency: $\varepsilon = [200 \quad 300 \quad 400 \quad 500 \quad 600]$
- Learning Rate: $\lambda = [0.12 \quad 0.14 \quad 0.16 \quad 0.20 \quad 0.24]$
- Price Memory: $\rho = [0.00 \quad 0.05 \quad 0.10 \quad 0.25 \quad 0.50]$

Results

Baseline - Single Simulation

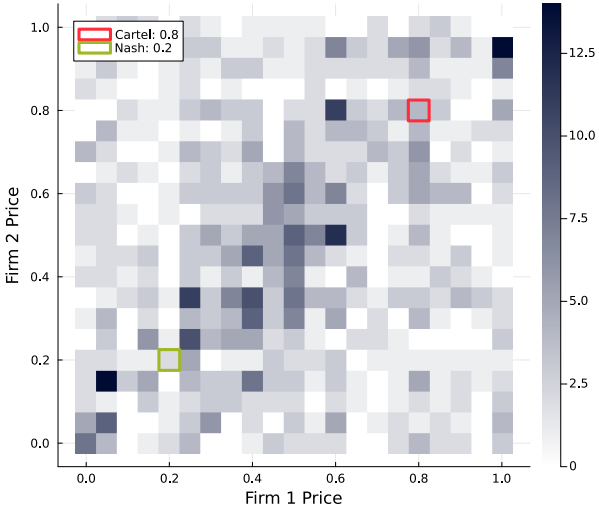
Simulations:

$S = [1 \quad 1000]$

Time Periods:

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

- Algorithms explore, then price match, then collude



Results

Baseline - Single Simulation

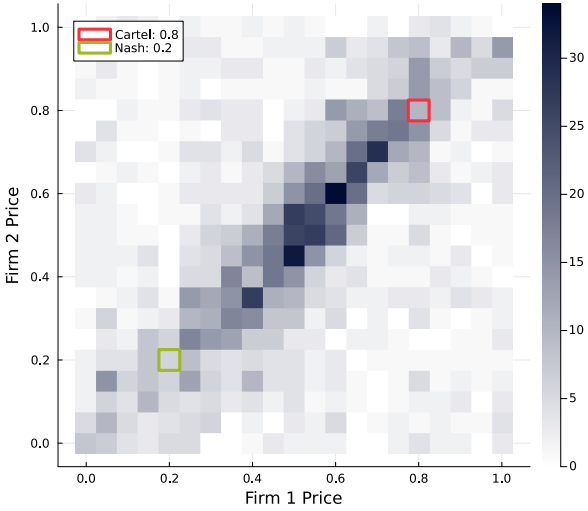
Simulations:

$S = [1 \quad 1000]$

Time Periods:

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

- Algorithms explore, then price match, then collude



Results

Baseline - Single Simulation

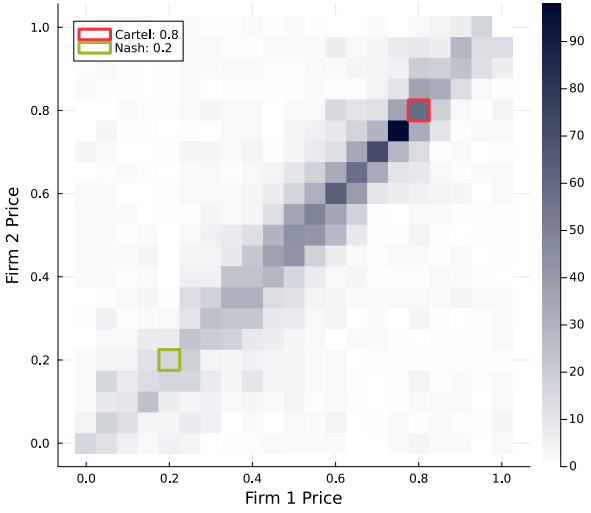
Simulations:

$S = [1 \quad 1000]$

Time Periods:

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

- Algorithms explore, then price match, then collude



Results

Baseline - Single Simulation

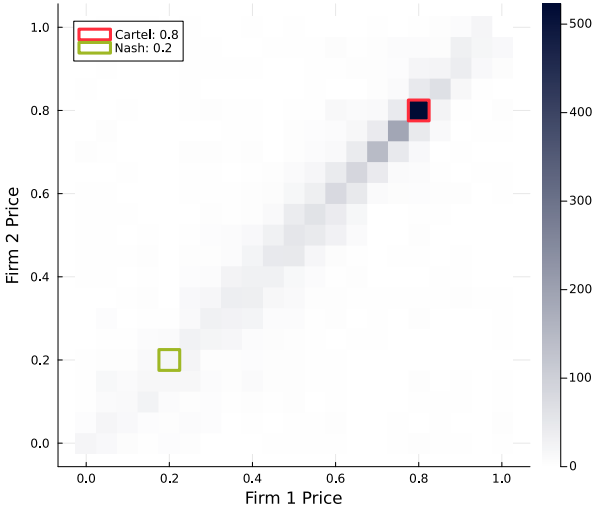
Simulations:

$S = [1 \quad 1000]$

Time Periods:

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

- Algorithms explore, then price match, then collude



Results

Baseline - Single Simulation

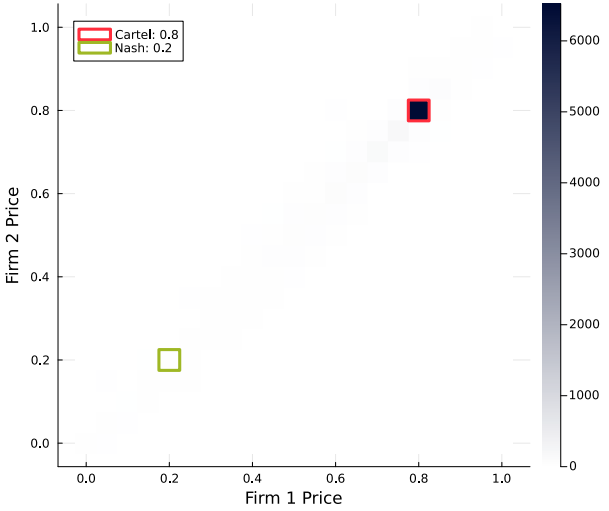
Simulations:

$S = [1 \quad 1000]$

Time Periods:

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

- Algorithms explore, then price match, then collude



Results

Baseline - Empirical Distribution

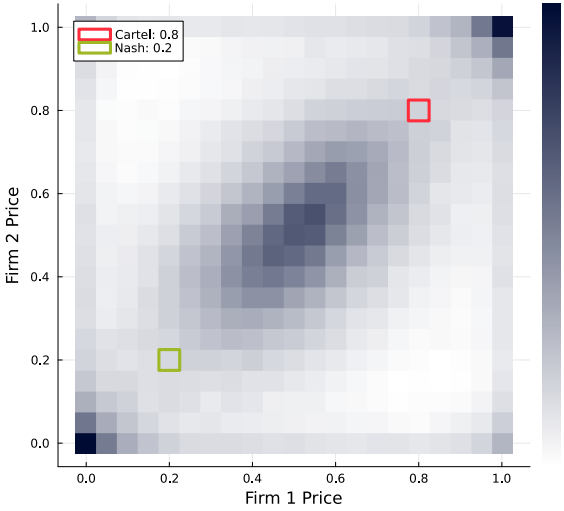
Simulations:

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

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

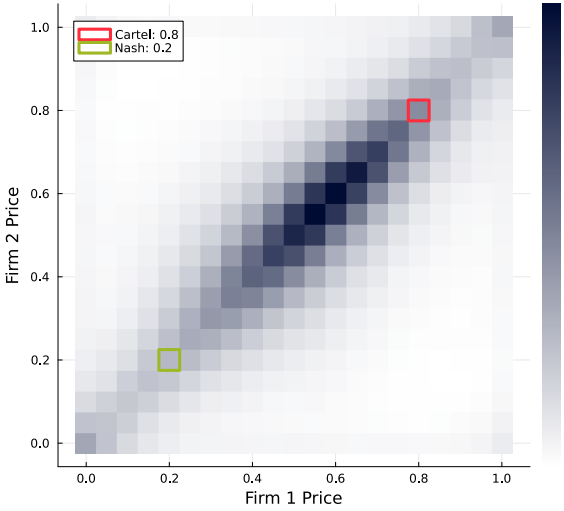
Simulations:

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

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

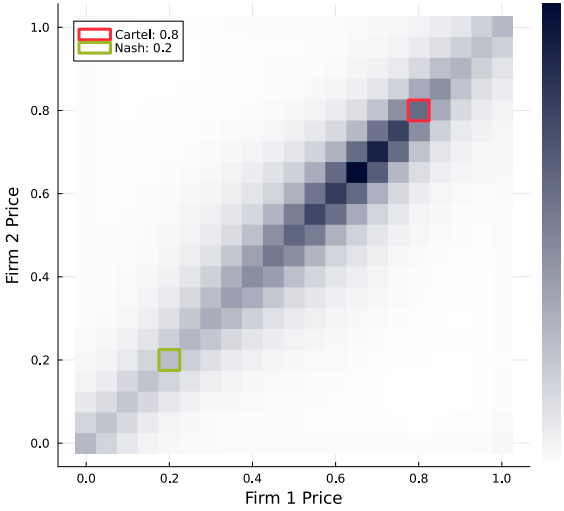
Simulations:

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

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

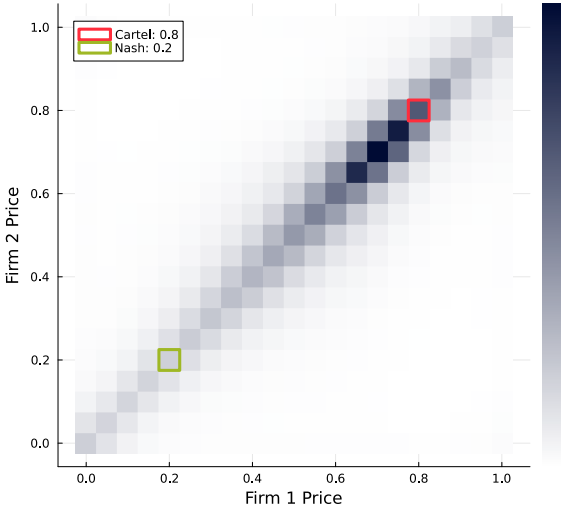
Simulations:

$S = [1 \quad 1000]$

Time Periods:

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

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



Results

Baseline - Empirical Distribution

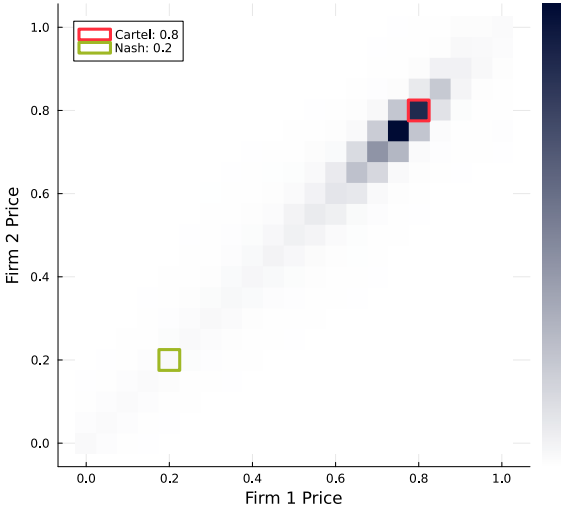
Simulations:

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

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



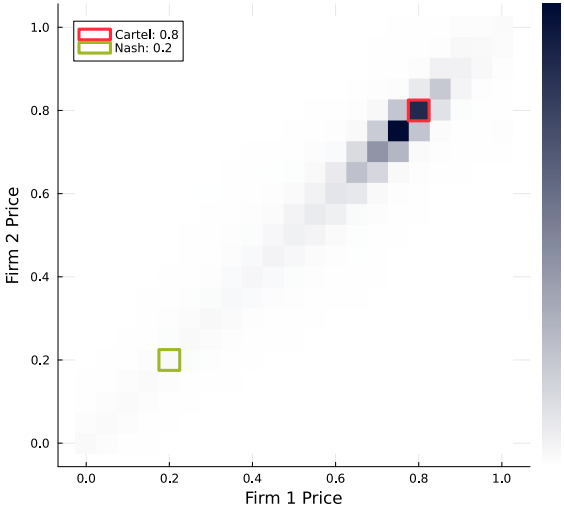
Results

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



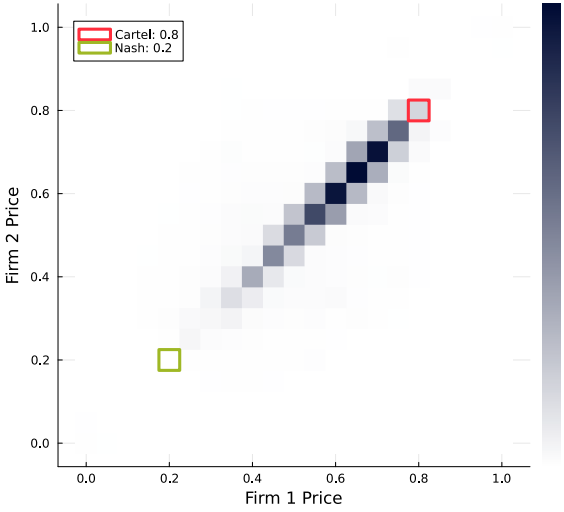
Results

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



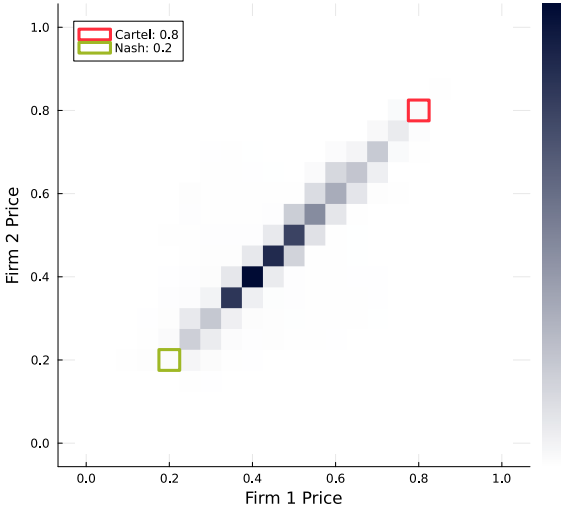
Results

Robustness - Algorithm Parametrization - 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



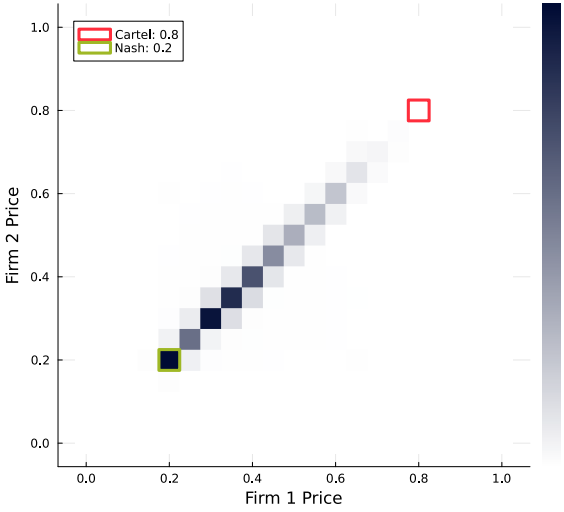
Results

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



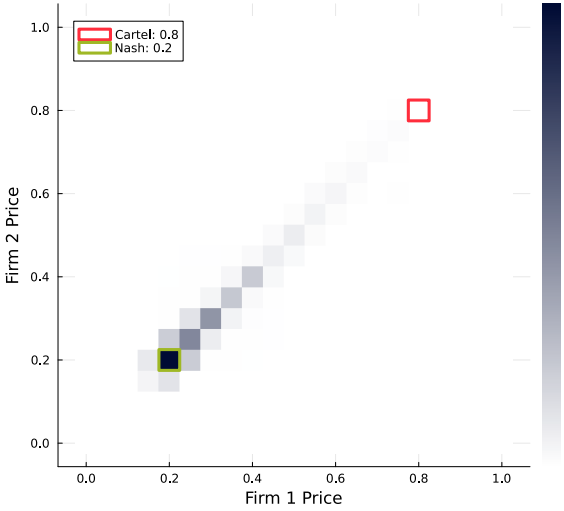
Results

Robustness - Algorithm Parametrization - 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



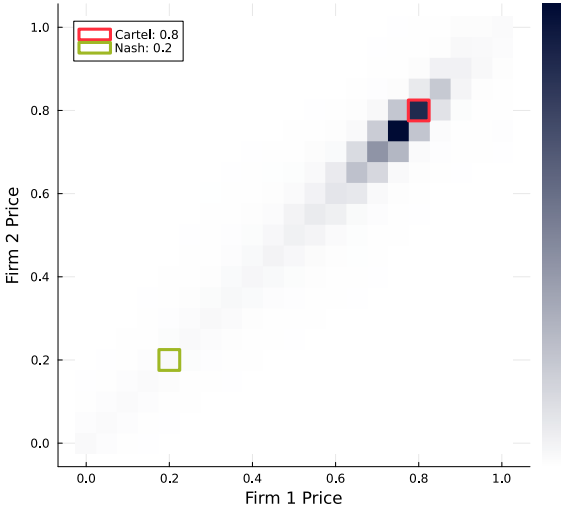
Results

Robustness - Algorithm Parametrization - Price Memory

Price Memory:

$\rho = [0.00 \quad 0.05 \quad 0.10 \quad 0.25 \quad 0.50]$

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



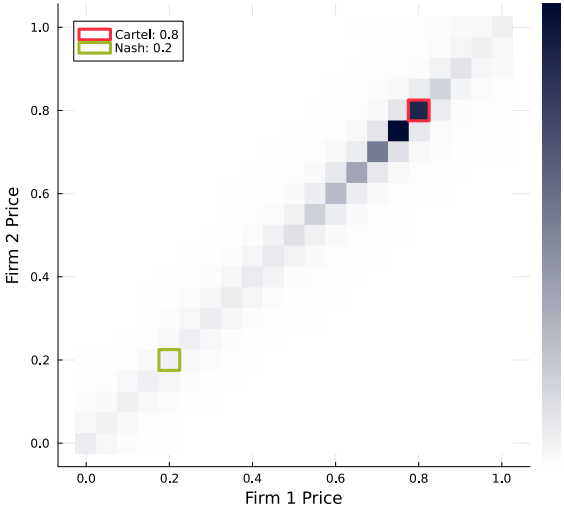
Results

Robustness - Algorithm Parametrization - Price Memory

Price Memory:

$\rho = [0.00 \quad 0.05 \quad 0.10 \quad 0.25 \quad 0.50]$

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



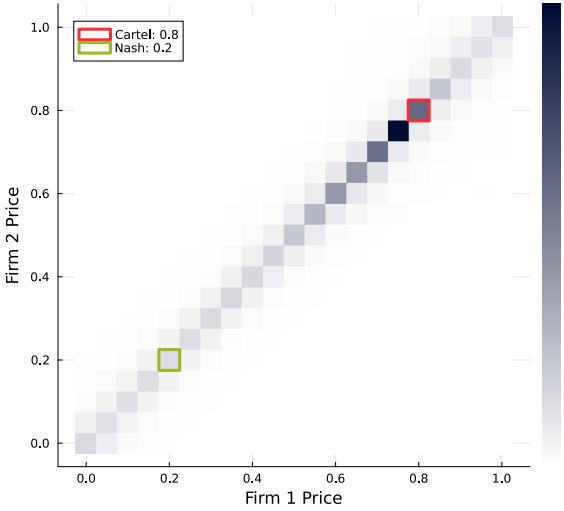
Results

Robustness - Algorithm Parametrization - Price Memory

Price Memory:

$\rho = [0.00 \quad 0.05 \quad 0.10 \quad 0.25 \quad 0.50]$

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



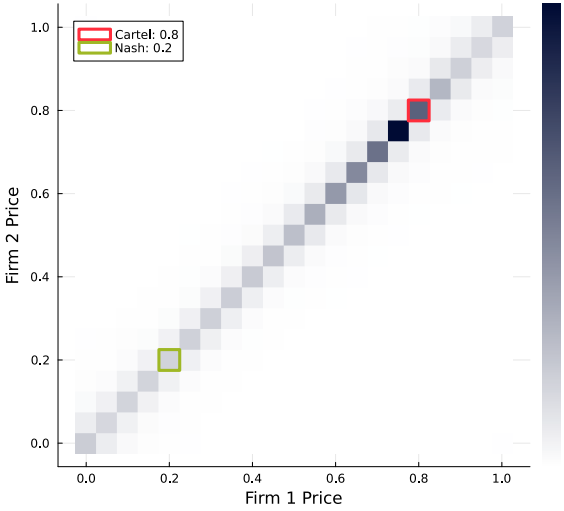
Results

Robustness - Algorithm Parametrization - Price Memory

Price Memory:

$\rho = [0.00 \quad 0.05 \quad 0.10 \quad 0.25 \quad 0.50]$

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



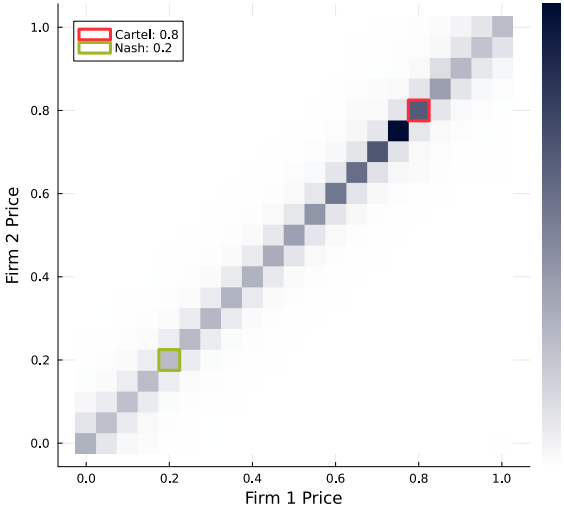
Results

Robustness - Algorithm Parametrization - Price Memory

Price Memory:

$\rho = [0.00 \quad 0.05 \quad 0.10 \quad 0.25 \quad 0.50]$

- 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

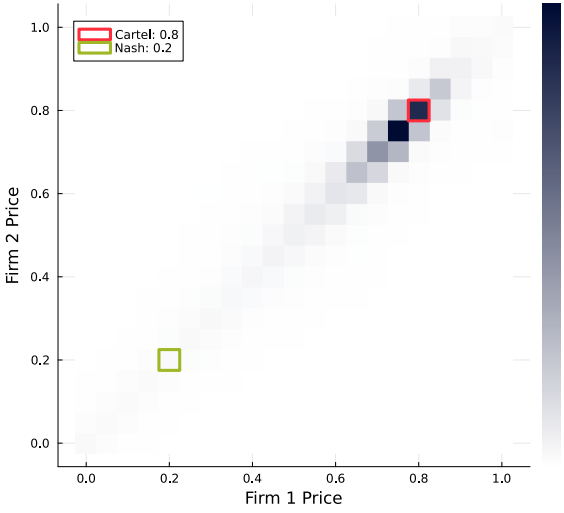
Exploration Frequency:

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

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

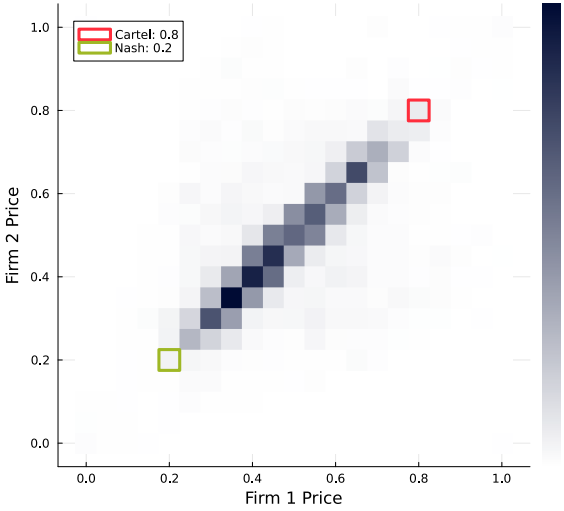
Exploration Frequency:

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

Learning Rate:

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

- 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

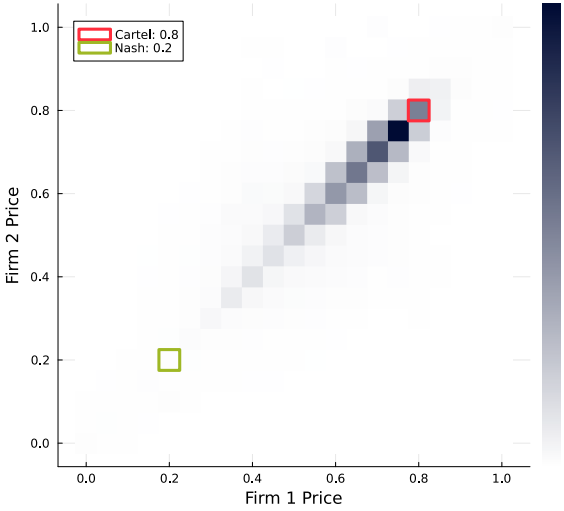
Exploration Frequency:

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

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

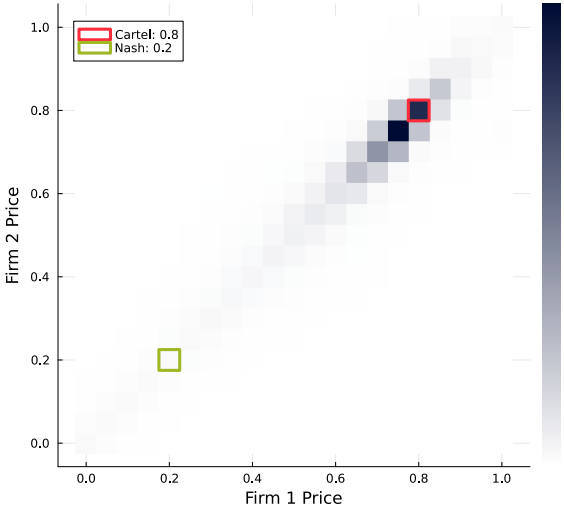
Exploration Frequency:

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

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

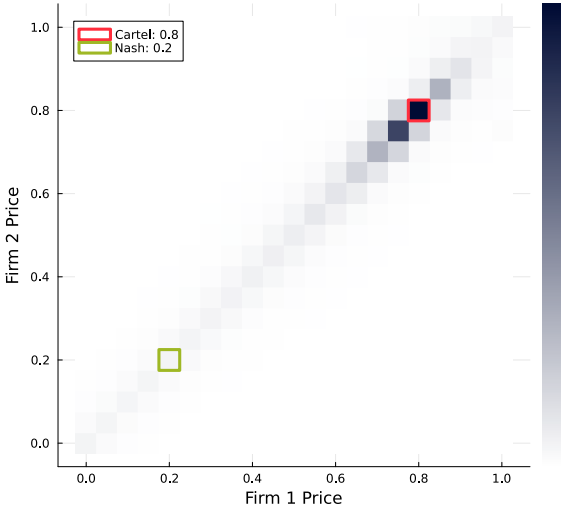
Exploration Frequency:

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

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

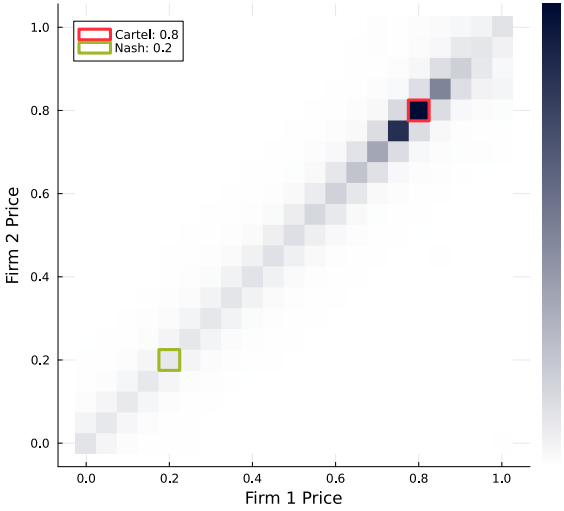
Exploration Frequency:

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

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

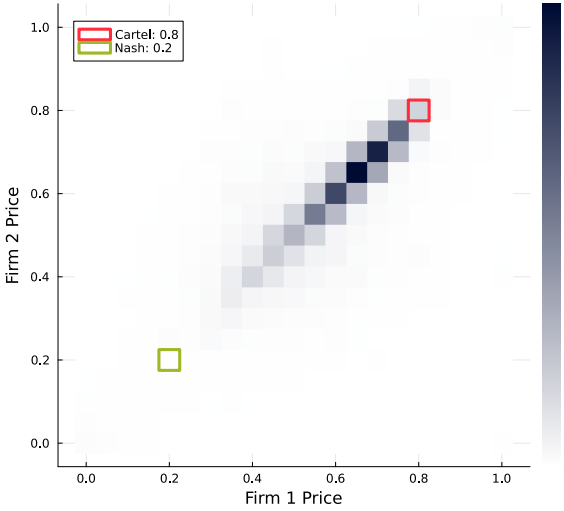
Exploration Frequency:

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

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

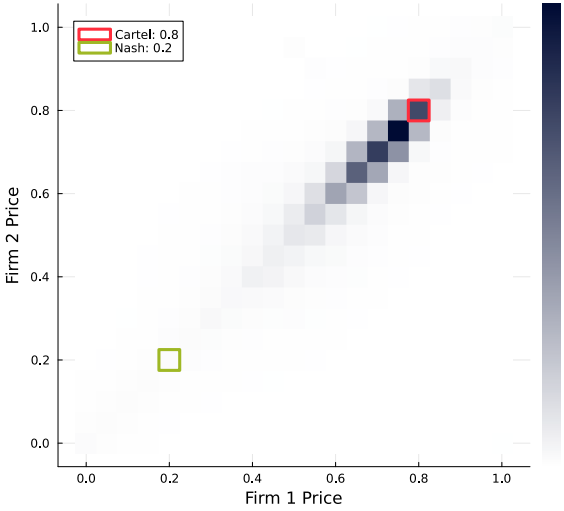
Exploration Frequency:

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

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

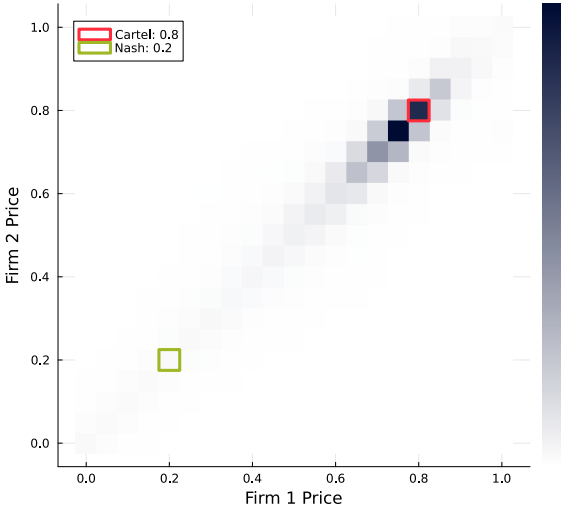
Exploration Frequency:

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

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

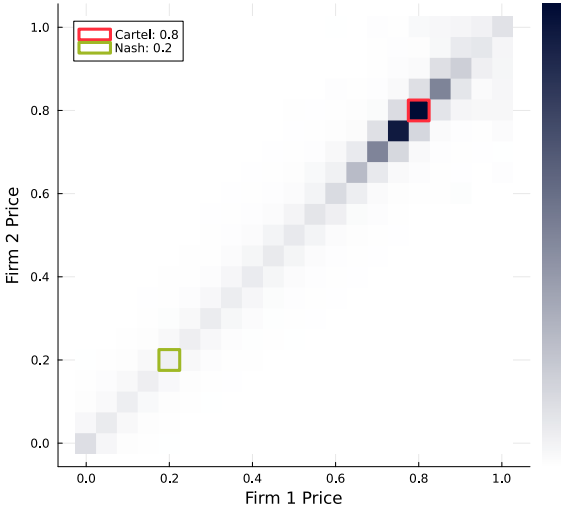
Exploration Frequency:

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

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

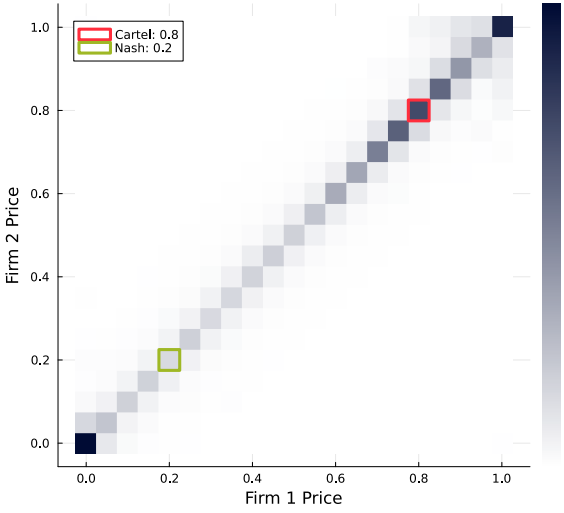
Exploration Frequency:

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

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

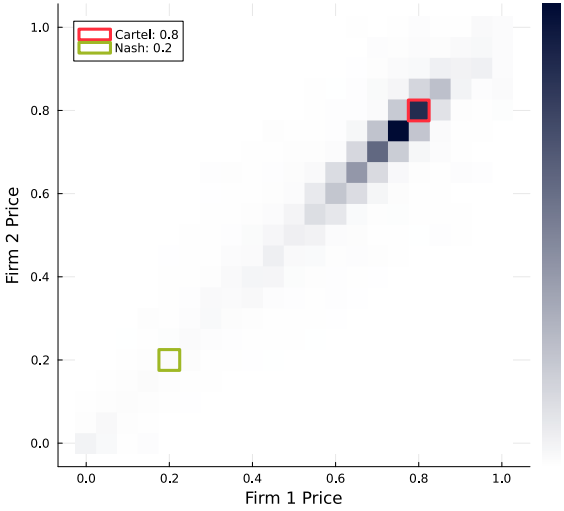
Exploration Frequency:

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

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

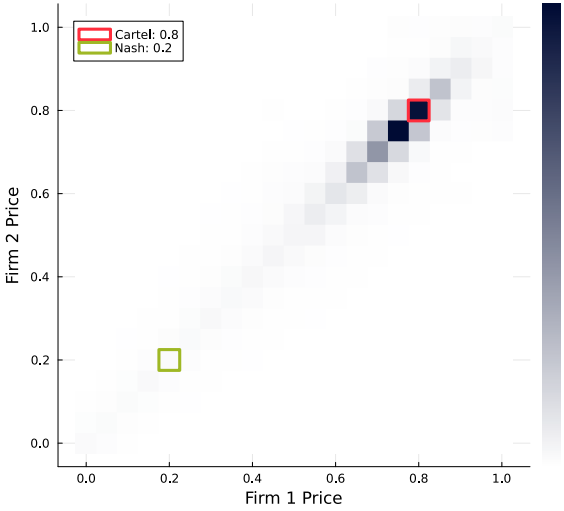
Exploration Frequency:

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

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

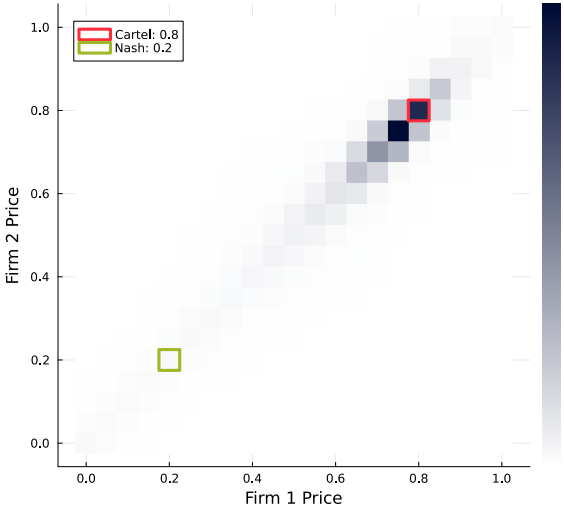
Exploration Frequency:

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

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

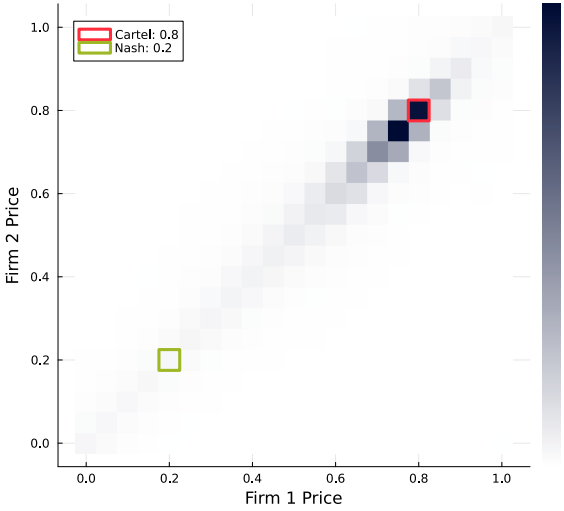
Exploration Frequency:

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

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

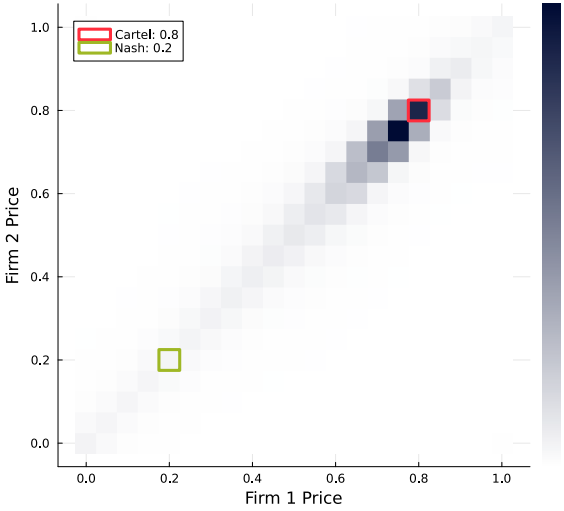
Exploration Frequency:

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

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

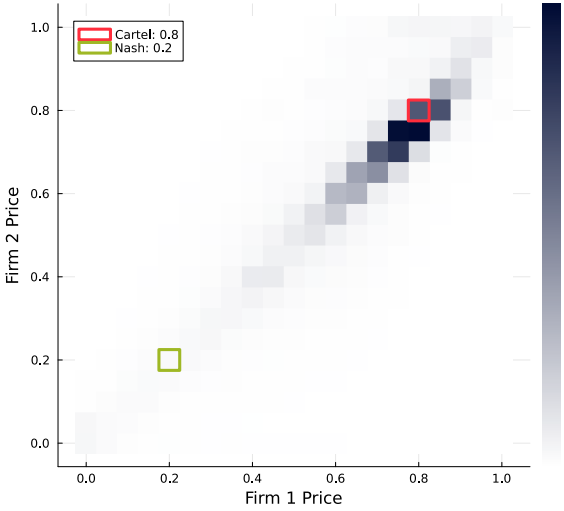
Exploration Frequency:

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

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

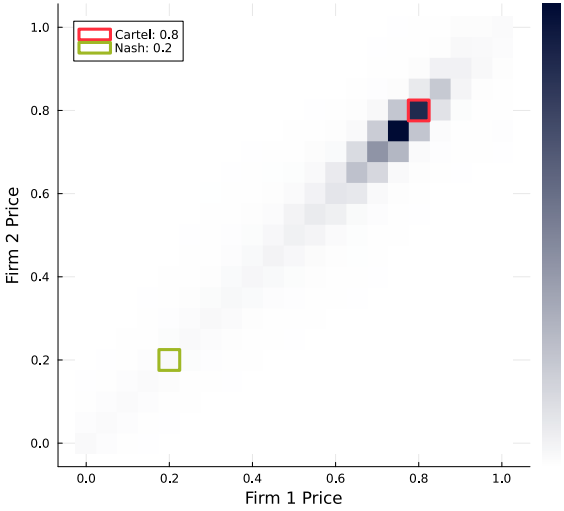
Exploration Frequency:

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

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

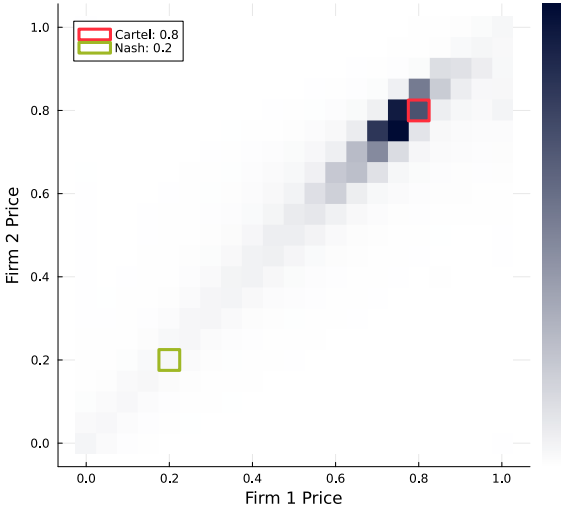
Exploration Frequency:

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

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



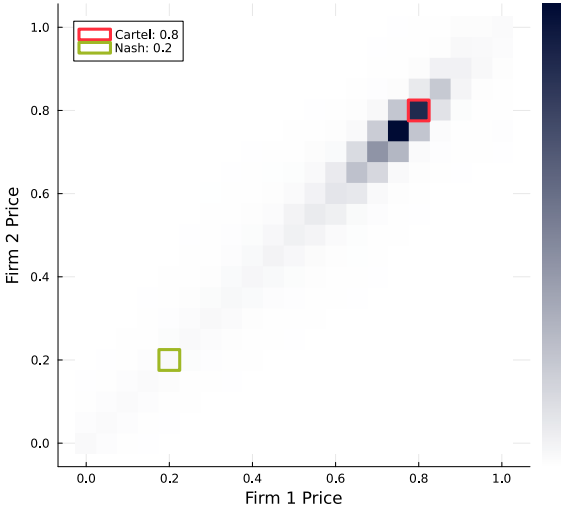
Results

Robustness - Strategic Environment - Action Space

Action Space:

$A = [21 \quad 41]$

- Collusive outcomes robust to granularity of action space



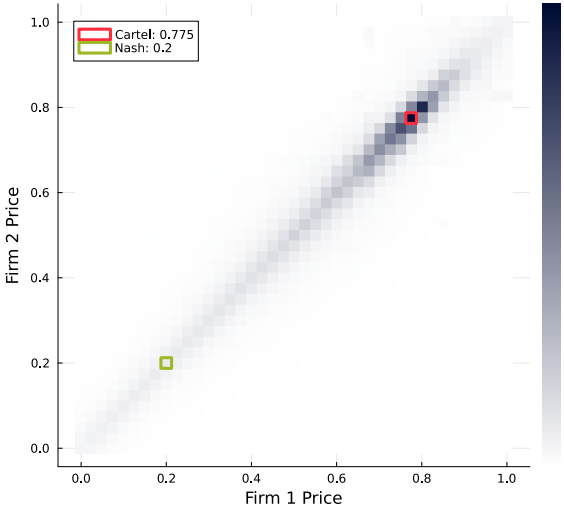
Results

Robustness - Strategic Environment - Action Space

Action Space:

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

- Collusive outcomes robust to granularity of action space



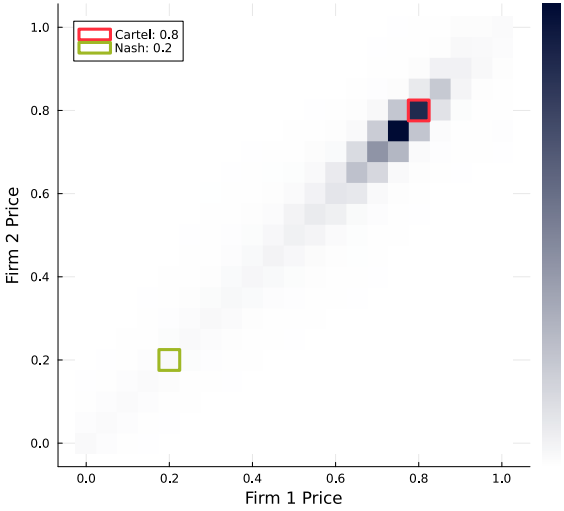
Results

Robustness - Strategic Environment - Product Differentiation

Product Differentiation:

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

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



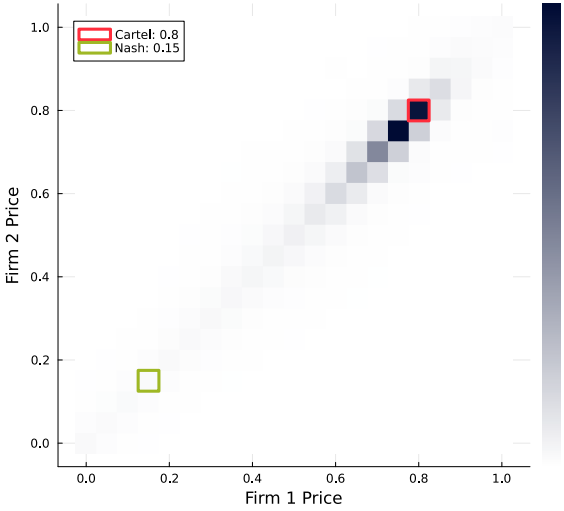
Results

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



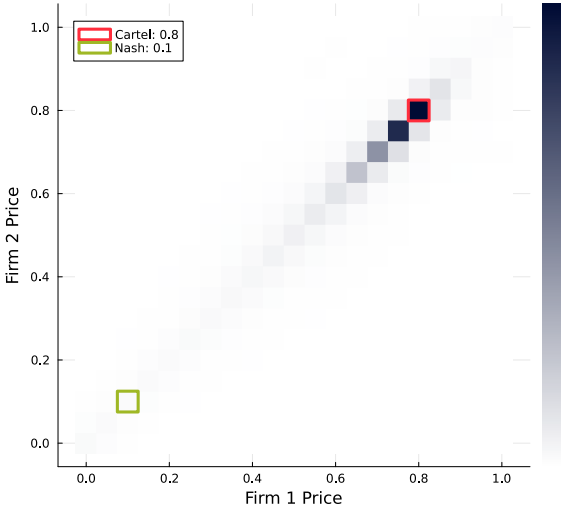
Results

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



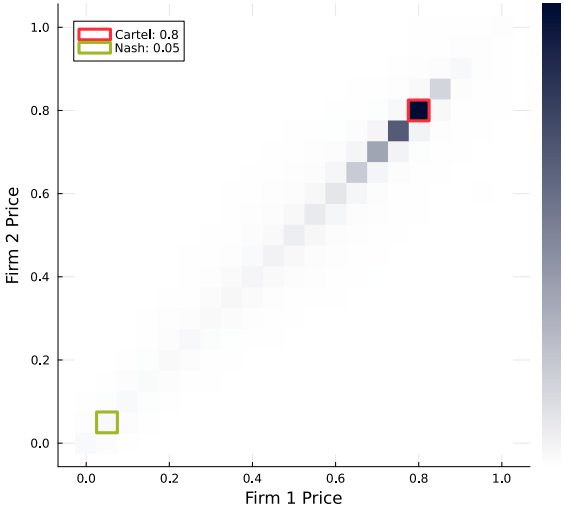
Results

Robustness - Strategic Environment - Product Differentiation

Product Differentiation:

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

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



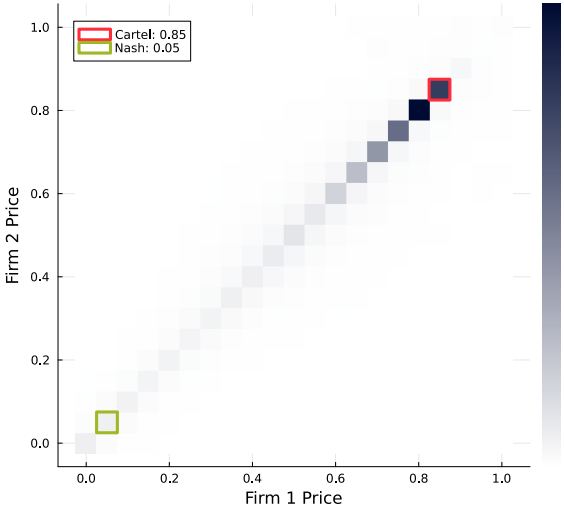
Results

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



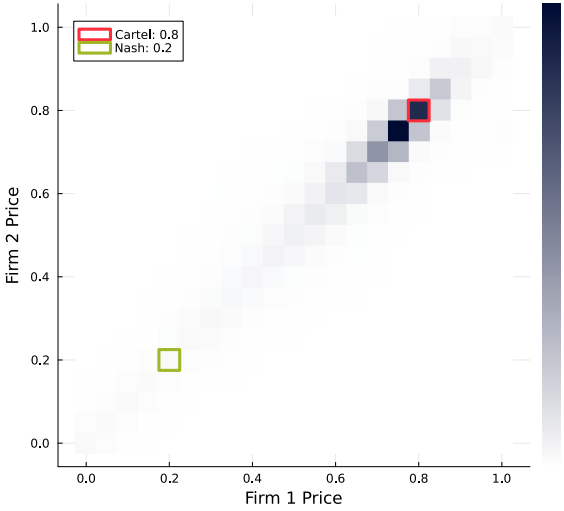
Results

Robustness - Strategic Environment - Firms

Firms:

$F = [2 \quad 3 \quad 4]$

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



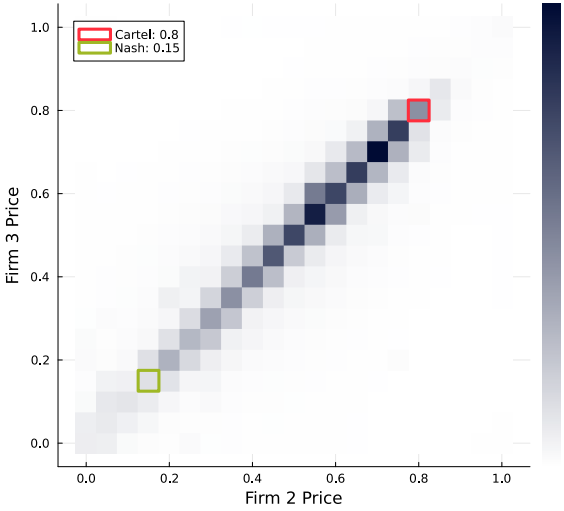
Results

Robustness - Strategic Environment - Firms

Firms:

$F = [2 \quad 3 \quad 4]$

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



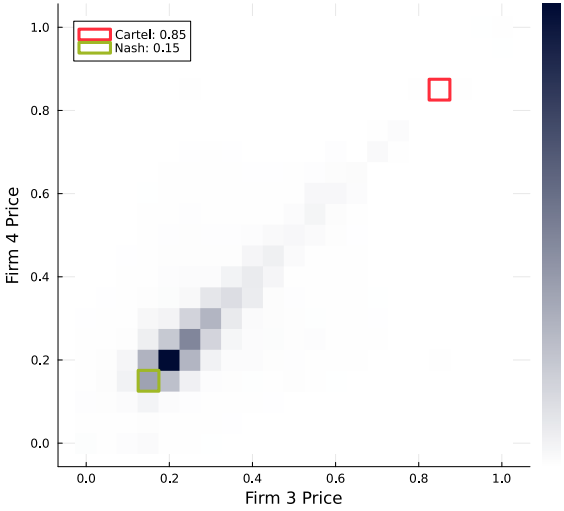
Results

Robustness - Strategic Environment - Firms

Firms:

$F = [2 \quad 3 \quad 4]$

- 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
  - Faster collusive coordination
    - Stable collusion after $\sim$3000 time periods
  - More perfect collusive prices
- Collusion most sensitive to number of firms
  - Difficult to coordinate with 3+ firms intermittently deviating
- Sensible algorithm hyperparameters required for collusion
  - Sufficient discount factor
  - Sufficient price memory
  - Efficient learning & exploration