No Free Lunch For Programmers: Digital Supply Chains and the Economics of Software Dependency Management

Sam Boysel

University of Southern California
Department of Economics

October 16, 2022

Outline

Introduction

- Modern software borrows $70 - 90\%$ functionality from external OSS projects (Nagle et al., 2022)
- Downstream *dependents* import functionality from upstream *dependencies*
- Software dependency network resembles digital supply chain for public information good production
  - Benefits: lower development costs, design modularity, specialization
  - Costs: maintenance and risk externalities
- **Question(s):** Can we estimate value created by dependency networks? How does a maintainer balance benefits and risk of dependency usage? What are the equilibrium implications for welfare and network structure?

Background & Motivation

- OSS dependency networks are intricate, widely leveraged public goods infrastructure (Eghbal, 2016)
- Public policy interest in OSS supply chain security (Executive Order 14028, 2021)
- High-profile disruptions and exploits
  - `left-pad`: Abrupt removal of 17 lines of code caused disruptions in 2% of all Node.js packages
  - Equifax breach: 147 million users exposed, $425 million USD in restitution (US FTC, 2022)
  - Other: Heartbleed, SolarWinds, `log4j`
- Average cost of data breach: $4.24 million USD (IBM, 2021)

▸ empirical setting

Contribution

This study

- Empirical study of software dependency network formation and influence.
- Place focus on project maintainer's decisions over (1) internal development and (2) usage of external software.
- **Reduced form:** productivity and quality effects of upstream dependencies on downstream dependents
- **Structural model:** micro-founded strategic network formation in which risk averse maintainer makes decisions under uncertainty.
- **Data:** network panel of 1,263 Node.js projects observed from 2011-2022: (1) network structure and (2) individual project characteristics.
- **Counterfactuals:** (1) modify risk aversion profile, (2) project quality volatility, and (2) remove key projects from the network.

Preview of Results

- Reduced form: upstream dependencies have limited influence on downstream behaviors, on average.
- Stronger complementarity in earlier sample periods
- Quality dependencies are well documented and have “lower bus factors”
- Structural model: individual project characteristics largely drive network formation.
- Increased maintainer risk aversion generates downstream value that offsets costs in aggregate.
- Externalities: removing $\leq 1\%$ of sample’s “core dependencies” reduces aggregate project quality by $\geq 5\%$ for remaining peers.

Literature

- Empirical Software Engineering Kikas et al. (2017), Decan et al. (2019), Decan and Mens (2019)
- Make vs. buy (Coase, 1937; Williamson, 1975, 1985) and vertical integration (Grossman and Hart, 1986)
- Micro-foundations for network formation under risk (Blume et al., 2013; Kovářík and Van der Leij, 2014)
- Supply chain risk and multi-sourcing (Elliott et al., 2022)
- Strategic Network Formation: Ballester et al. (2006), Snijders et al. (2010), Hsieh et al. (2022)

Framework

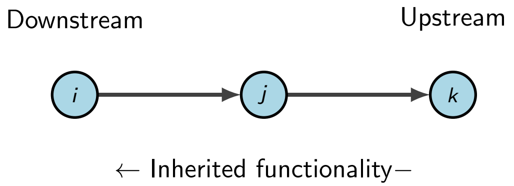


Figure: Project i depends on directly project j and project j depends directly on project k. We say project k is an *indirect dependency* of project i. Additional terminology: Project k is *upstream* of both projects i and j. Project i is *downstream* of both projects j and k. The dependence relationship runs in the opposite direction of the flow of inherited functionality.

▸ risk and network structure ▸ a maintainer's choice ▸ fragility

▸ project quality and contribution costs ▸ preferences and UMP

Data

- **Network panel:** 1263 Node.js (JavaScript) packages observed from October 2010 through September 2022
    1. Dependency declarations
    2. Socio-technical features: number of contributors, number of commits, age, size, documentation, modularity, churn, cyclomatic complexity, "bus factor"
    3. Derive a measure of quality = log(complexity) + log(# contributors)
- Upstream sample: start with 10 most depended upon packages, traverse (upstream) dependency tree recursively
- Use of software version control to "rewind" software source code

▸ Snapshot of dependency network sample ▸ Sampling procedure ▸ Descriptive Statistics

Reduced Form: Contribution

Relation between upstream and downstream contribution (x_{it})

$$\underbrace{x_{it}}_{\text{Downstream}} = a_i + \alpha \underbrace{\sum_{j \neq i} G_{ijt} x_{jt}}_{\text{Upstream}} + \delta' W_{it} + \epsilon_{it} \qquad (1)$$

- $\alpha > 0$ (< 0) implies upstream and downstream contribution are complements (substitutes)
- **Identification:** (1) dependencies formed unilaterally, (2) scope for coordination possible but limited, (3) use project FEs for OVB concerns
- **Findings:** α small and positive on average, virtually 0 when controlling for project FEs
- **Robustness:** α stronger in earlier sub-sample periods. Small, centered around 0 at project-level.

estimates

Reduced Form: Quality

Relation between upstream and downstream project quality (y_{it})

$$y_{it} = b_{0i} + b_{1i} x_i + \beta \sum_{j \neq i} G_{ijt} y_{jt} + \delta' W_{it} + \epsilon_{it} \tag{2}$$

- $\beta > 0$ (< 0) implies quality in upstream dependencies improves (diminishes) quality of downstream dependents
- **Findings:** β small and positive on average, virtually 0 when controlling for project FEs
- **Robustness:** β similar in all sub-sample periods. Small, centered around 0 at project-level.

estimates

Reduced Form

Number of upstream dependencies

$$d_{ijt}^{\text{out}} = \delta' W_{it} + \epsilon_{ijt} \tag{3}$$

Number of downstream dependents

$$d_{ijt}^{\text{in}} = \delta' W_{it} + \epsilon_{ijt} \tag{4}$$

Findings:

- Higher quality packages declare more dependencies
- Core dependencies well documented, fewer upstream dependencies themselves.
- Virtually no effect of project size (commits or SLOC), number of contributors, modularity, age
- Some evidence that core dependencies have higher “bus factor”: reliant on small group of contributors

estimates

Structural Approach

- Maintainer's problem: choose (1) level of development effort and (2) set of dependency relationships to minimize development costs and maintain project quality.
- Key feature: maintainers averse to uncertainty over dependency project quality
- Equilibrium: sequential game with myopic maintainers
- Data generating process: Characterizes co-evolution of software dependency network relationships and development levels

setup equilibrium comparative statics estimation

Counterfactual Analysis

1. Modify maintainer risk aversion
2. Modify project quality uncertainty
3. Remove key packages

▸ estimates

Discussion

- **Recap:**
  - Software dependency networks embed complex economic tradeoffs
  - Externalities are present but nuanced
  - Upstream dependencies create value via complementarity
  - Disruptions in core packages cause disruptions downstream at scale
- **Policy:** transparency, automation, public sponsorship of critical projects are ongoing efforts
- **Future directions:** consider alternative measures of quality, shift focus onto fixed costs of development

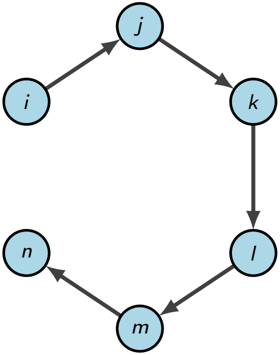
Details

Background: Empirical Setting

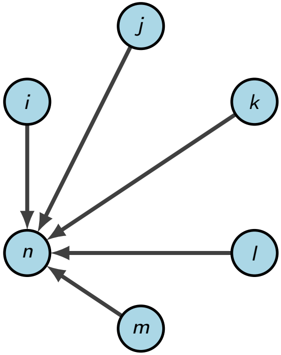
- `npm` (Node.js Package Manager) ecosystem.
- 2.6 million packages (Katz, 2020)
- Example: `babel-loader`
  - Registry Listing: `https://www.npmjs.com/package/babel-loader`
  - Source Code Repository: `https://github.com/babel/babel-loader`

back

Framework: Risk and Network Structure



(a) Indirect Network



(b) Hub Network

Figure: The network in Panel 2a is subject to more indirect risk than the network in Panel 2b. When considering the extent of both direct and indirect dependence, project *n* is the most critical project in both networks.

◂ back

Framework: The Maintainer's Dilemma

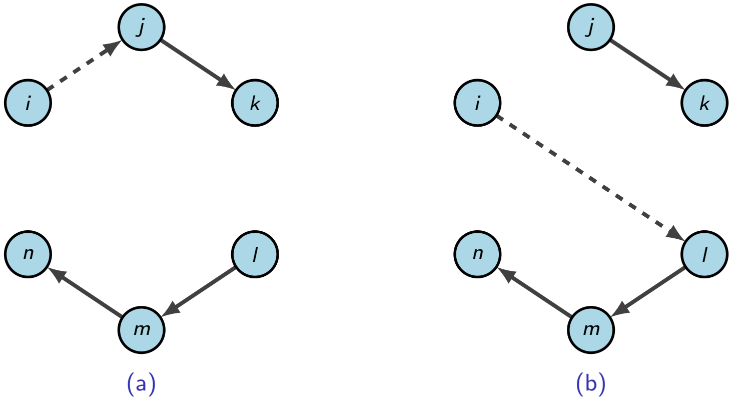
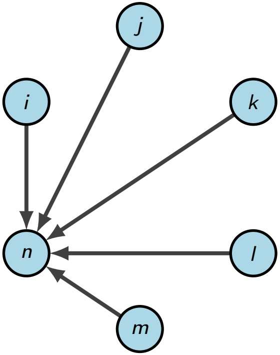


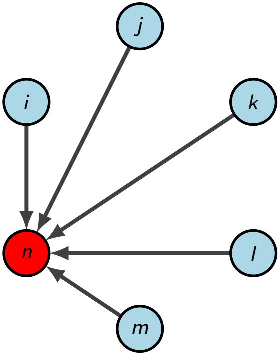
Figure: In Panel 3a, maintainer *i* prefers depending on project *k* over project *l* and avoids a greater level of indirect risk embedded in project *l*. In Panel 3b, maintainer *i* prefers *l* to *j* despite a greater level of indirect risk embedded by project *l*.

back

Framework: Network Fragility (1/2)



(a) Central project is low risk

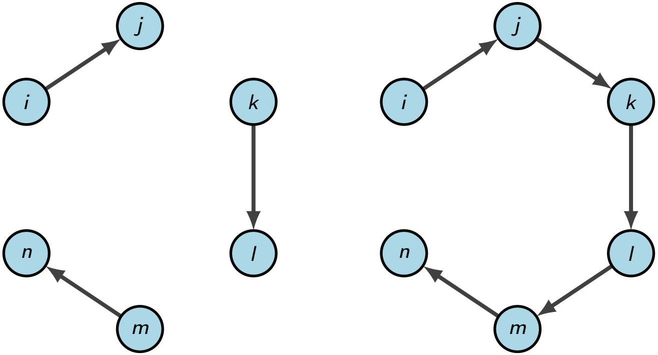


(b) Central project is high risk

Figure: Different project characteristics can influence system-wide fragility for networks with identical structure.

back

Framework: Network Fragility (2/2)



(a) Structure isolates risk (b) Structure amplifies risk

Figure: Different network structures can influence overall network fragility when projects characteristics are held constant.

◂ back

Framework: Project Quality and Contribution Costs

Project Quality

$$y_i(x, y_{-i}, G) = b_i x_i + \beta \sum_{j \neq i} G_{ij} y_j + \xi_i \tag{5}$$

Contribution Costs

$$c_i(x, G) = \frac{1}{2} x_i^2 - \left(a_i + \alpha \sum_{j \neq i} G_{ij} x_j \right) x_i \tag{6}$$

◂ back

Framework: Maintainer Preferences

von Neumann-Morgenstern utility: maintainer has a stronger preference for software stability as $r_i \to \infty$:

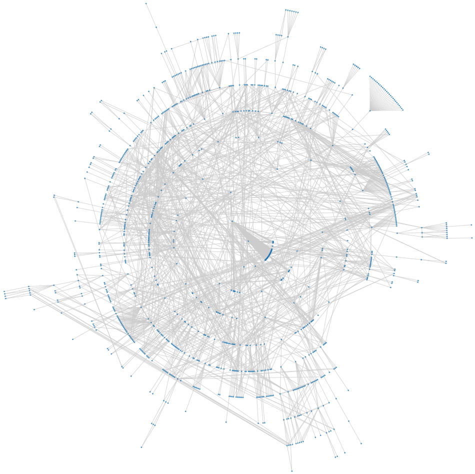
$$u_i(x, y, G) = \mathbb{E}\left[-e^{-r_i y_i}\right] \tag{7}$$

Cost Minimization: Maintainer i chooses $(y_i^\star, x_i^\star, \{G_{ij}^\star\}_{j\neq i})$ such that

$$\begin{aligned}(y_i^\star, x_i^\star, \{G_{ij}^\star\}_{j\neq i}) = \underset{y_i, x_i > 0, \{G_{ij}\}_{j\neq i}}{\arg\min} \quad & c_i(x, G) \\ \text{s.t.} \quad & u_i(x, y, G) \geq \underline{u}_i\end{aligned} \tag{8}$$

◂ back

Data: Sample Dependency Network (September 2022)



back

Data: "Upstream" Sampling Procedure

1. Let $k = 0$. Begin with a set of the 10 most depended-upon packages in the Node.js ecosystem, $\mathcal{N}_k$.
2. For each $i \in \mathcal{N}_k$, get a list of timestamps t for the published minor versions of package i, $\mathcal{T}_i$
   - For each $t \in \mathcal{T}_i$, add the set of packages i declares as runtime dependencies to the running set of sampled packages:
     $$\{j \in \mathcal{N} \mid G_{ijt} = 1\} \cup \mathcal{N}_k \to \mathcal{N}_{k+1}$$
   - Return to Step 2 using $\mathcal{N}_{k+1}$
3. Limit the search depth to 5th degree dependencies ($k \leq 5$).

Note: result is a set critical core dependencies in the Node.js ecosystem. Lower bound for estimates of network externalities.

back

Data: Descriptive statistics

Notation	Measure	Obs.	Mean	SD	Min	Median	Max
x_{it}	**Project Contribution**	206,598	2,703	6,573	1	195	81,641
$\sum_{j \neq i} G_{ijt} x_{jt}$	**Dependency Contribution**	206,598	1,451	19,427	0	0	1,098,604
y_{it}	**Project Quality**	206,598	0.363	0.199	0	0.334	1
$\sum_{j \neq i} G_{ijt} y_{jt}$	**Dependency Quality**	206,598	0.225	1.236	0	0	55.2
ω_{it}	**Time Allocation**	206,598	2,909	7,571	2	202	105,504
W_{it}	**Contributors**	206,598	233	983	1	20	17,195
W_{it}	**Core Contributors**	206,598	20	233	1	1	4,421
W_{it}	**Bus Factor**	206,598	0.209	0.260	0.003	0.121	1
W_{it}	**Files**	206,598	2,692	7,109	1	26	65,724
W_{it}	**SLOC**	206,598	129,556	326,412	2	2,717	2,974,829
W_{it}	**Modularity**	206,598	222	2,528	1.87	50.5	74,349
W_{it}	**Documentation**	206,598	0.110	0.172	0	0.046	3.18
W_{it}	**Number of languages**	206,598	7.87	3.34	1	7	29
W_{it}	**Project Age**	206,598	1,240	987	0	1,019	4,278
d_{it}^{out}	**Upstream Dependencies**	161,211	2	3.76	0	1	74
d_{it}^{in}	**Downstream Dependents**	161,211	5	7.94	0	2	66

back

Reduced Form: Contribution

	Project Commits							
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Constant	2,653***	-68.19***						
	(14.51)	(9.744)						
Dependency Commits	0.034***	0.000***	0.003***	0.007***	0.002	0.000*	0.000	-0.000
	(0.002)	(0.000)	(0.001)	(0.001)	(0.001)	(0.000)	(0.000)	(0.000)
Project Quality		156.2***				959.5	126.9***	790.1***
		(22.26)				(518.5)	(19.57)	(322.0)
# Contributors		0.101***				0.546*	0.211***	1.542*
		(0.017)				(0.276)	(0.034)	(0.742)
Bus Factor		18.96***				68.50*	7.401*	6.286
		(3.851)				(32.56)	(2.928)	(21.80)
SLOC		0.000***				0.000	0.000***	0.000
		(0.000)				(0.000)	(0.000)	(0.000)
Documentation		-6.203				108.5	9.413***	136.5*
		(3.201)				(67.99)	(3.578)	(61.60)
Modularity		-0.001***				-0.001	-0.001***	-0.002
		(0.000)				(0.001)	(0.000)	(0.001)
# Languages		6.423***				7.306	5.836***	2.513
		(0.996)				(11.41)	(0.973)	(11.57)
Age		-0.012***				-0.002	-0.017***	-3.617
		(0.002)				(0.026)	(0.004)	(2.415)
Controls		✓				✓	✓	✓
Project FE			✓		✓	✓		✓
Time FE				✓	✓		✓	✓
R^2	0.010	0.999	0.964	0.674	0.988	0.999	0.999	0.999
Observations	206,598	202,905	206,575	201,336	201,308	202,869	196,930	196,894

Note: This table contains coefficient estimates from ordinary least squares (OLS) and fixed effect (FE) estimates for the reduced form relationship between project contribution and upstream dependency contribution in Equation (1). Specification variations are reported in columns. Standard errors, heteroskedasticity-robust for OLS and clustered by project for FE models, are reported in parentheses below each coefficient. Additional covariate controls not reported in the table include 3 lags each of project commits and dependency commits and the square of project age. All terms rounded to four significant figures. Statistical significance indicators: *** $\Rightarrow p \leq 0.001$, *** $\Rightarrow p \leq 0.01$, and * $\Rightarrow p \leq 0.1$ where p is the p-value for the coefficient estimate.

back

Reduced Form: Quality

	Project Quality							
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Constant	0.302***	0.002***						
	(0.001)	(0.049)						
Dependency Quality	0.017***	0.000	0.004***	0.009***	0.002	0.001	0.000**	0.000
	(0.001)	(0.000)	(0.002)	(0.001)	(0.001)	(0.001)	(0.000)	(0.001)
Project Commits	0.000***	0.000	0.000*	0.000***	0.000***	0.000	0.000	0.000
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.063)	(0.047)	(0.052)
# Contributors		0.000				-0.000	0.000	0.000
		(0.000)				(0.000)	(0.000)	(0.002)
Bus Factor		-0.003***				-0.007***	-0.003***	-0.007***
		(0.000)				(0.001)	(0.000)	(0.001)
SLOC		-0.000				0.000	-0.000	-0.000
		(0.000)				(0.000)	(0.000)	(0.001)
Documentation		0.001***				0.002***	0.001***	0.003
		(0.000)				(0.000)	(0.000)	(0.003)
Modularity		0.000				0.000	0.000	0.000
		(0.000)				(0.000)	(0.000)	(0.000)
# Languages		0.000***				0.002***	0.000***	0.002
		(0.000)				(0.000)	(0.000)	(0.002)
Age		0.000				0.000	0.000	0.000
		(0.000)				(0.000)	(0.000)	(0.000)
Controls		✓				✓	✓	✓
Project FE			✓		✓	✓		✓
Time FE				✓	✓		✓	✓
R^2	0.514	0.999	0.966	0.764	0.987	0.999	0.999	0.999
Observations	206,598	202,905	206,575	201,336	201,308	202,869	196,930	196,894

Note: This table contains coefficient estimates from ordinary least squares (OLS) and fixed effect (FE) estimates for the reduced form relationship between project quality and upstream dependency quality in Equation (2). Specification variations are reported in columns. Standard errors, heteroskedasticity-robust for OLS and clustered by project for FE models, are reported in parentheses below each coefficient. Additional covariate controls not reported in the table include 3 lags each of project quality and dependency quality and the square of project age. All terms rounded to four significant figures. Statistical significance indicators: *** $\Rightarrow p \leq 0.001$, *** $\Rightarrow p \leq 0.01$, and * $\Rightarrow p \leq 0.1$ where p is the p-value for the coefficient estimate.

back

Reduced Form: Dependency Formation

	# of Upstream Dependencies				# of Downstream Dependents			
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Constant	0.720***				4.720***			
	(0.121)				(0.147)			
Project Commits	-0.000	-0.000	0.000	-0.000	0.001	0.000	0.001	0.001
	(0.027)	(2.493)	(0.126)	(0.000)	(0.038)	(0.001)	(0.210)	(0.001)
Upstream Commits	-0.000	-0.000	-0.000	-0.000	0.000	0.000**	0.000	0.000**
	(0.001)	(0.205)	(0.028)	(0.000)	(0.017)	(0.000)	(0.038)	(0.000)
Project Quality	3.073***	4.898***	5.180***	4.306*	3.935***	7.444	5.850***	5.350
	(0.029)	(0.255)	(0.002)	(1.986)	(0.098)	(4.481)	(0.044)	(3.444)
Upstream Quality	3.000***	1.227***	2.818***	1.176***	-1.683***	-0.313***	-1.585***	-0.327***
	(0.005)	(0.037)	(0.030)	(0.222)	(0.012)	(0.089)	(0.035)	(0.080)
# of Contributors	-0.000	0.000	-0.001	0.002	-0.009***	-0.002	-0.008	-0.004
	(0.001)	(0.019)	(0.000)	(0.001)	(0.003)	(0.003)	(0.005)	(0.004)
Bus Factor	-0.181***	0.078***	-0.227***	0.215	-1.098***	0.595	-0.787***	1.040*
	(0.010)	(0.003)	(0.000)	(0.243)	(0.018)	(0.667)	(0.001)	(0.526)
SLOC	-0.000	-0.000	-0.000	0.000	-0.000	-0.000	-0.000	-0.000*
	(0.004)	(0.037)	(0.005)	(0.000)	(0.016)	(0.000)	(0.001)	(0.000)
Documentation	-0.941***	-0.979***	-1.170***	-0.873	4.325***	2.607*	4.640***	2.513***
	(0.058)	(0.042)	(0.001)	(0.544)	(0.124)	(1.047)	(0.005)	(0.697)
Modularity	-0.000	-0.000	-0.000	-0.000	0.000	0.000	-0.000	0.000
	(0.005)	(0.010)	(0.002)	(0.000)	(0.012)	(0.000)	(0.009)	(0.000)
# of Languages	-0.055***	-0.036	-0.061	0.002	-0.575***	-0.272	-0.427**	-0.280
	(0.002)	(0.693)	(0.063)	(0.001)	(0.005)	(0.217)	(0.156)	(0.166)
Age	-0.000	-0.000	-0.000	-0.000	0.002	0.002***	0.004	-0.056
	(0.002)	(0.004)	(0.001)	(0.000)	(0.004)	(0.000)	(0.003)	(0.067)
Project FE		✓		✓		✓		✓
Time FE			✓	✓			✓	✓
R^2	0.535	0.875	0.622	0.899	0.143	0.946	0.384	0.965
Observations	161,211	161,183	161,211	161,175	161,211	161,183	161,211	161,275

Note: Columns (1) through (4) report coefficient estimates for the specification in Equation (3), a regression of the number of upstream dependencies for a given project (i.e. out-degree) on observable project characteristics. Columns (5) through (8) report coefficient estimates for the specification in Equation (4), a regression of the number of downstream dependents for a given project (i.e. in-degree) on observable project characteristics.

back

Structural: Setup (1/2)

Project Quality

$$y_i(x, y_{-i}, G) = b_i x_i + \beta \sum_{j \neq i} G_{ij} y_j + \xi_i \tag{9}$$

Information Sets

$$\xi_i \sim N(0, \sigma_i^2) \tag{10}$$

Commonly known only in distribution to all maintainers.

back

Structural: Setup (2/2)

Contribution Costs

$$c_i(x, G) = \frac{1}{2}x_i^2 - \left(a_i + \alpha \sum_{j \neq i} G_{ij} x_j\right) x_i \tag{11}$$

Maintainer Preferences:

$$u_i(x, y, G) = \mathbb{E}\left[-e^{-r_i y_i}\right] \tag{12}$$

◂ back

Structural: Equilibrium (1/5)

Maintainer *i*'s cost-minimization problem under uncertainty

$$\begin{aligned}
\min_{y_i, x_i > 0, \{G_{ij}\}_{j \neq i}} \quad & \frac{1}{2} x_i^2 - \left(a_i + \alpha \sum_{j \neq i} G_{ij} x_j \right) x_i && \text{(Cost)} \\
\text{s.t.} \quad & \mathbb{E}\left[-e^{-r_i y_i} \right] \geq \underline{u}_i && \text{(Utility)} \\
& y_i = b_i x_i + \beta \sum_{j \neq i} G_{ij} y_j + \xi_i && \text{(Quality)} \\
& \xi_i \sim N(0, \sigma_i^2) && \text{(Uncertainty)}
\end{aligned} \tag{13}$$

◂ back

Structural: Equilibrium (2/5)

Timing: at each sample moment $t \in \mathcal{T}$ is presented with an opportunity to re-evaluate her dependency relationships

1. Maintainer i chooses set of optimal dependencies, $\{G^{\star}_{ijt}\}_{j \neq i}$
2. All maintainers $i, j \in \mathcal{N}$ determine optimal contribution levels for their projects, $x^{\star}_{ijt} > 0$.

◂ back

Structural: Equilibrium (3/5)

Optimal level of project development

$$\begin{aligned} x_i^\star &= a_i + \alpha \sum_{j \neq i} G_{ij} x_j^\star \\ &= \sum_j A_{ij} a_j \end{aligned} \tag{14}$$

Equilibrium Project Quality

$$\begin{aligned} y_i^\star &= b_i \left(a_i + \alpha \sum_{j \neq i} G_{ij} x_j^\star \right) + \beta \sum_{j \neq i} G_{ij} y_j^\star + \xi_i \\ &= \sum_j B_{ij} \left(b_j \left(\sum_k A_{jk} a_k \right) + \xi_j \right) \end{aligned} \tag{15}$$

where $A = (I - \alpha G)^{-1}$ and $B = (I - \beta G)^{-1}$.

◂ back

Structural: Equilibrium (4/5)

Maintainer's expected utility over expected project quality

$$\begin{aligned} u_i(x^\star, y^\star, G) &= -\exp\left(-r_i \sum_j B_{ij}\left(b_j x_j^\star - \frac{r_i}{2} B_{ij}\sigma_j^2\right)\right) \\ &= -\exp\left(-r_i \sum_j B_{ij}\left(b_j\left(\sum_k A_{jk} a_k\right) - \frac{r_i}{2} B_{ij}\sigma_j^2\right)\right) \end{aligned} \tag{16}$$

◂ back

Structural: Equilibrium (5/5)

Optimal Dependency Formation Decision

$$G_{ij} = 1 \iff u_i(x^\star, y^\star, G + ij) \geq u_i(x^\star, y^\star, G - ij) \tag{17}$$

Likelihood of dependency formation, $\Pr(G_{ij} = 1)$, becomes

$$F_\epsilon\left(r_i \underbrace{\left(\sum_j \Delta B_{ij} b_j \left(\sum_k \Delta A_{jk} a_k \right) \right)}_{Z_{0ij}} - \frac{1}{2} r_i^2 \underbrace{\left(\sum_j \Delta B_{ij}^2 \sigma_j^2 \right)}_{Z_{1ij}} ; \theta_\epsilon \right) \tag{18}$$

under multiplicative linking disturbances ϵ (Fosgerau and Bierlaire, 2009)

◂ back

Structural: Comparative Statics

1. **Risk Aversion (r_i):**

$$\frac{\partial \Pr(G_{ij}=1)}{\partial r_i} \begin{cases} < 0 & \text{if } Z_{ij} < r_i \\ \geq 0 & \text{if } Z_{ij} \geq r_i \end{cases}$$

   since $\frac{\partial F_\epsilon}{\partial z} > 0$ and $r_i > 0$ and $Z_{ij} \equiv \frac{Z_{0ij}}{Z_{1ij}}$, net benefit threshold.

2. **Project Quality Variance (σ_k^2):**

$$\frac{\partial \Pr(G_{ij}=1)}{\partial \sigma_k^2} \leq 0$$

◂ back

Structural: Estimation

Observed data is $\mathcal{D} = (x_t, y_t, G_t, W_t)_{t\in\mathcal{T}}$. Procedure to estimate structural parameters $\theta = (a, \alpha, b, \beta, \Sigma, r, \theta_\epsilon)$:

1. Estimate (b, β) given $\mathcal{D}$ using the project quality specification Equation (9) via OLS
2. Estimate (a, α) given $\mathcal{D}$ and estimates for b using equilibrium contribution Equation (14) via OLS. Use residuals to estimate Σ.
3. Estimate (r, θ_ϵ) via MLE, maximizing a likelihood function based on equilibrium link formation described in Equation (18).

$$
\begin{aligned}
L(\theta \mid \mathcal{D}) = \prod_{t\in\mathcal{T}} \Pr(\mathcal{D}_t | \mathcal{D}_{t-1}, \theta) &= \prod_t \prod_{j\neq i} \Pr(G_{ij} = 1 \mid \mathcal{D}_{t-1}, \theta) \\
&= \prod_t \prod_{j\neq i} F_\epsilon\left(u_{ijt}\right)^{G_{ijt}} \left(1 - F_\epsilon(u_{ijt})\right)^{1-G_{ijt}}
\end{aligned}
$$

Note that $\{\Pr(\mathcal{D}_t \mid \mathcal{D}_{t-1})\}_{t\in\mathcal{T}}$ forms a Markov chain.

back

Counterfactual

Counterfactual	Welfare	Project Quality	Contribution	Costs
Minimize Risk Aversion	0.22	0.40	0.00	0.00
Increase risk aversion ($+1\sigma$)	-0.07	-0.09	0.00	0.00
Maximize Risk Aversion	-0.04	2.11	0.00	0.00
Reduce quality volatility (-50%)	0.00	0.19	0.00	0.00
Increase quality volatility (+100%)	0.00	1.51	0.00	0.00
Increase quality volatility (+300%)	0.05	1.14	0.00	0.00
Remove top package (Katz-Bonacich)	0.00	0.03	0.00	0.00
Remove top 10 packages (Katz-Bonacich)	-0.07	-5.73	-1.30	-0.87
Remove top 10 packages (Betweenness)	-0.08	0.57	-0.07	0.00

Note: The elements of the four right-most columns indicate the percentage change in each aggregate measure under the counterfactual scenario relative to the baseline equilibrium observed in the data (September 2022).

back