Open Source Software Policy in Industry Equilibrium

Jeff Gortmaker*
Harvard University

January 19, 2025
Latest version

Abstract

Open source software (OSS) is a form of public knowledge widely provided and relied on by the private sector. To study the effects of growing government involvement in this critical public good, I build a new empirical model where high-tech firms choose software inputs and developer labor in competitive equilibrium. For estimation, I create a new dataset of OSS and in-house investment for the global web development industry, where software choices are directly observable. I simulate counterfactuals to assess the global impact of China tightening its recent internet restrictions on cross-border OSS collaboration or increasing its financial support for domestic OSS. I find that stricter restrictions do little to boost domestic OSS investment. Instead, lost spillovers raise web development costs in China by $2 per dollar of disincentive and $7 globally. Heightened subsidies prove more effective at increasing domestic investment and cut global costs by $11 per dollar of subsidy—tripling if the US responds in kind.

*jgortmaker@g.harvard.edu. I am grateful to Myrto Kalouptsidi, Robin Lee, Frank Nagle, and Ariel Pakes for their advice and support. For comments, data, and discussions, I also thank Constanza Abuin, Maxim Alekseev, Luis Armona, Sam Boysel, Kevin Chen, Chris Conlon, Leonardo D'Amico, Pedro Degiovanni, Mert Demirer, Ed Glaeser, Shane Greenstein, Gordon Hanson, Kate Ho, Manuel Hoffmann, Garrett Johnson, Louis Kaplow, Justin Katz, Lev Klarnet, Rem Koning, Josh Lerner, Tianli Li, Andrés de Loera, Katy Ma, Pierfrancesco Mei, Namrata Narain, Nano Ochoa, Raphael Raux, Tomas Reimers, Peleg Samuels, Kunal Sangani, Suproteem Sarkar, Steve Schmeiser, Alexander Staub, Wilbur Townsend, Chris Walker, Andi Wang, Kevin Xu, David Yang, Daniel Yue, Brandon Zborowski, Jieying Zhang, Ruby Zhang, Yanuo Zhou, and Harvard workshop participants.

"Open source software is at the heart of Apple platforms and developer tools. Apple works with developers around the world to create, contribute, and release open source code."

— *Apple's open source website, 2021 redesign, opensource.apple.com.*

"Accelerate the construction of domestic open source communities and create equal access for participants. Develop profit-generating operating mechanisms to guide domestic open source innovation forces to the international open source community."

— *China's 14th Five-Year Plan for National Informatization, 2021, translated by Google.*

1. Introduction

The private sector creates many forms of public knowledge, including basic innovation, open data, and academic research. Open source software (OSS)—computer programs freely available for use and modification—is public knowledge in the form of code and documentation. OSS is widespread: a recent survey found that 96% of commercial codebases use OSS, and 77% of total code is OSS (Synopsys, 2024). Estimates of the cost to recreate all OSS range from billions of dollars in the US (Robbins et al., 2021) to trillions globally (Hoffmann et al., 2024). Even Apple, known for its secrecy, both uses and provides OSS, as quoted above.

While few governments have directly intervened in OSS (Lostri et al., 2023), China has recently expressed its intent to support domestic OSS, quoted above. It has also subsidized its domestic OSS platform and imposed new restrictions on cross-border OSS collaboration, as I show below. More broadly, rising US-China tensions may threaten collaboration on other global public goods, such as basic research (e.g., Flynn et al., 2024). Since US and Chinese firms are global leaders in OSS investment, and OSS is a critical input for firms worldwide, government involvement could have significant consequences.

Two key characteristics of privately provided knowledge are especially relevant for policymaking. First, contributions to public goods can come with *private* benefits (Olson, 1965). For instance, OSS contributors often add features they *themselves* need, as their firms rely on OSS (Nagle et al., 2020). Second, public benefits can be *localized* (Tiebout, 1956). For instance, OSS features documented in Chinese primarily benefit *Chinese* firms. Underprovision due to weak private benefits may justify subsidizing OSS investment (Pigou, 1920), while strongly localized benefits may justify other policies, such as restrictions, that coordinate domestic firms to use and invest in domestic OSS (e.g., Rosenstein-Rodan, 1943).

To quantify policy tradeoffs, I build a new model of private incentives for using and investing in different OSS, tailored to newly-assembled data on the web development industry. Each year, firms choose which software they use to build their websites and how much labor to invest in software codebases, a form of knowledge capital. Firms can use

and invest in both OSS and their own in-house software. These choices affect web development costs through developer wages and website quality through a production function. After firms make choices, consumers choose which websites to visit based on quality, firm country, and content language. Firms then receive profit—revenue from traffic minus labor costs—and consumers receive utility. Firms are myopic, but persistence arises from capital accumulation, switching costs, and autocorrelated unobservables.

The model's central externality is that firms using an OSS benefit from other firms' contributions to that OSS. In addition to this *public* benefit of OSS investment, the model features two *private* benefits: contributions particularly improve a firm's own codebase, and developers may accept lower wages to work on OSS as an amenity.[1] Finally, the model features a form of *localization* particularly relevant to government policy: firms benefit more from contributions within their own country.

In the model, if the public benefits of OSS greatly outweigh the private benefits of OSS investment, OSS will be severely underprovided by the private sector, and subsidies can prove effective at addressing this underprovision. Since public benefits are localized, subsidizing domestic investment or restricting foreign collaboration can help coordinate firms to use and invest in domestic OSS. However, under fierce competition, if OSS greatly damages or improves product quality, business stealing can either strengthen or weaken OSS policies.[2] I estimate these tradeoffs for a specific industry.

I focus on web development: a large but relatively understudied industry,[3] where one can observe a website's software inputs directly in its browser-facing code, offering a rare view "under the hood" of the tech sector. I construct a new panel of firms operating one or more of the world's 10,000 most-visited websites annually from 2014 to 2022. I identify the software firms use to build their websites and measure how they allocate web developer labor to in-house investment and different web OSS. By estimating the hours invested in different software capital stocks, I create new measures of knowledge capital for high-tech firms.

[1]Below, I discuss how much of the OSS literature focuses on why *individuals* contribute. Prominent motivations in the literature, as well as in recent survey evidence (e.g., Nagle et al., 2020), include on-the-job work, career concerns, intrinsic motivation, and reciprocal altruism.

[2]Suppose OSS subsidies increase investment, lowering firms' costs and leading to greater OSS use. If this also improves quality, and competition over quality is intense, firms may internalize that their OSS investment causes business stealing, dampening the policy's impact. Conversely, if OSS use harms website quality but firms still prefer it to cut costs, business stealing could amplify the policy.

[3]To my knowledge, I am the first to estimate an empirical model of the web development industry. Existing literature on web OSS has primarily focused on the security and value of servers (e.g., Greenstein and Nagle, 2014; Murciano-Goroff et al., 2021, 2024; Ackermann and Greenstein, 2024). I focus on backend, JavaScript, and UI frameworks, which have received less attention but more OSS investment in recent years.

My dataset offers several advantages, including rich information on OSS contributions from the global OSS platform (GitHub) and, unusual in the OSS literature, firm-level data on OSS use. I estimate the time spent on each OSS contribution to measure OSS investment in hours. My measure of in-house web development investment is coarser, scaling with the number of web developers employed and their typical workweeks.[4] The core of the dataset is a large-scale match between the global population of web developers—including 223,371 contributors to 27 major web OSS—and the website-owning firms listed on developers' online resumes from LinkedIn and GitHub. This match helps overcome common challenges in identifying innovation spillovers, and allows me to measure spillovers directly: I observe which firms invest in each web OSS and which use these same OSS in production.

Using my dataset, I first document *overall* patterns of OSS investment and use.[5] OSS investment is rare: investment in major web OSS is four orders of magnitude lower than in-house investment in web development. However, OSS use is widespread: nearly half of the firms in my dataset use at least one of these web OSS in production. Large firms, often based in the US, contribute disproportionately. In total, China ranks second. Per developer, however, the US and Europe contribute more intensively than China and India.

Next, I estimate my model of web development costs by matching simulated moments (McFadden, 1989; Pakes and Pollard, 1989) based on *firm-level* patterns of OSS investment and use. Firms tend to invest in and use OSS that is (*i*) highly developed, (*ii*) domestically developed, and (*iii*) self-made. The model translates these patterns into estimated (*i*) public, (*ii*) localized, and (*iii*) private benefits, captured by lower developer wages required to invest in and use OSS with (*i*) more, (*ii*) domestic, and (*iii*) own investment. Since within-firm wage data are scarce, I use firms' revealed preferences to infer these wage reductions, scaled to match typical web developer wages by country. Where possible, I validate these estimates using limited data on wages of web developers working with different web OSS.

Firms minimize web development costs subject to a website quality production function, which I estimate using standard timing assumptions (Olley and Pakes, 1996; Levinsohn and Petrin, 2003; Ackerberg, Caves, and Frazer, 2015). I derive quality from a discrete choice demand model (Berry, Levinsohn, and Pakes, 1995, 2004), where websites are differentiated by quality, firm country, and content language. The model translates overall traffic into

[4] Tambe et al. (2020) also use LinkedIn data on tech employment to measure in-house knowledge capital, focusing on hedonic regressions of US public firms' market value on tech wage bills. My approach is complementary, assigning value to knowledge capital—in-house and open—based on firms' revealed preferences.

[5] I build on earlier work on the geography of OSS contributions (e.g., Gonzalez-Barahona et al., 2008; Wachs et al., 2022) by comparing contributions with firm-level web OSS usage.

quality estimates, and country traffic shares into consumer "home biases." I use firms' quality choices to infer the amount of revenue generated by traffic, and validate my estimates with what limited data are available on website revenue.

These estimates capture the balance of forces in the model, which shape the economic impact of OSS policies. I estimate a wide gap between the private cost reductions from OSS investment and the public cost reductions shared across firms, suggesting subsidies could help close this gap. I also estimate that these cost reductions are moderately, but not overpoweringly localized, affecting how domestic subsidies or restrictions on foreign collaboration can address coordination issues. On the demand side, consumers show strong "home bias," limiting cross-border competition. I also estimate a weak link between OSS use and website quality, further reducing the role of product market competition and business stealing, which could otherwise complicate policy outcomes.

Using the estimated model, I simulate counterfactuals to assess the global impact of China continuing to extend its OSS policies. In the next section, I provide the first empirical evidence that China has recently imposed soft internet restrictions on access to the global OSS platform (GitHub) and that censorship on China's domestic platform (Gitee) led to a large and sustained drop in domestic collaboration. Along with restrictions, China has also provided financial support for Gitee. These policies likely arise from concerns over economic outcomes, information control, and software security—I focus on *economic* consequences.

First, I evaluate China strengthening its restrictions on foreign collaboration, which could, in theory, coordinate Chinese firms to use and invest in domestic OSS. I "tax" Chinese investment in web OSS based on the share of foreign contributions, up to \$100 per hour—six times the typical hourly wage of Chinese web developers. This simulated restriction proves ineffective at promoting investment in domestic web OSS and reduces overall Chinese web OSS investment by 3%. Due to the widespread use of OSS, I find that lost innovation spillovers would raise global web development costs by \$7 per dollar of disincentive and \$2 in China. The model does not guarantee this result; if the estimated public benefits from OSS investment were more localized and larger, I find that restrictions would be more effective.

Second, I evaluate China strengthening its subsidies for domestic OSS investment. Since OSS investment is rare, subsidies appear feasible:[6] \$100 for each of the 40,000 hours invested

[6]A detailed evaluation of specific mechanisms is beyond the scope of this article. Mechanisms include GitHub or Gitee subsidies, tax credits (e.g., New York State Assembly, 2022), or changes to accounting standards (e.g., Financial Accounting Standards Board, 2024). Conti, Gupta, Guzman, and Roche (2023) and Hoffmann, Boysel, Nagle, Peng, and Xu (2024) show that direct rewards and indirect AI assistance for OSS contributors, respectively, increase productivity but also affect the type of contributions.

into my sample's web OSS in 2022 amounts to only $4 million. A Chinese subsidy of $100 per hour invested in domestic web OSS successfully increases Chinese web OSS investment by 20%, especially in domestic OSS. Due to innovation spillovers, the subsidies cut global web development costs by $11 per dollar of subsidy and $5 in China.

Third, in light of recent US policies targeted at China, such as the CHIPS Act (2022), a shift in Chinese OSS policy could prompt a US response. To date, US OSS policies, like those of most countries, focus on procurement and security.[7] However, if the US were to match China's per-hour subsidies, I estimate that US firms would increase web OSS investment by 16%, especially in domestic OSS. Since firms primarily use US-built OSS, global cost reductions per dollar of subsidy would be three times greater than from China's subsidies.

The effects of simulated policies are similar whether I hold web traffic fixed or allow firms to compete on quality for consumers' web traffic, though effects are somewhat larger when firms can optimally adjust their websites' quality in response to OSS policies. The clearest conclusion from my results for OSS policy is its impact on firm costs. Statistically unclear effects on consumer surplus reflect either a truly weak effect of OSS on website quality or noise in my web traffic data. These are empirical findings, not guaranteed by the model. Business stealing could either strengthen or weaken the impact of OSS policies in industries where product quality is greatly damaged or improved by OSS.

Finally, to benchmark the size of US and Chinese subsidies, I simulate a *global* subsidy of $100 per hour invested into *any* of the web OSS in my sample. Cost reductions from global coordination are six times greater than from unilateral and domestic-focused subsidies in the US and China, indicating that the latter can capture a substantial, though incomplete, share of the global benefits from subsidizing OSS. Overall, I find that subsidies are relatively cheap because OSS investment is rare, while their effects are large—especially when targeted at OSS that is used the most—because OSS use is widespread.

For the web development industry, my simulations suggest OSS policy could have large effects. However, the 600,000 hours of OSS investment in my sample are just 0.3% of investment into *all* OSS used in production, a lower bound estimated by Hoffmann et al. (2024). This is not to suggest that I expect OSS policy outcomes to be 300 times larger; the OSS I study are unusually popular and specific to one industry. Rather, to the extent that my approach generalizes across industries, the stakes for broader OSS policy may be high.

My approach emphasizes some incentives and de-emphasizes others. In the private sector,

[7]Some US agencies favor OSS for procurement (e.g., Department of Defence, 2022), and there has been a recent push to increase funding for OSS security (e.g., White House, 2024). The US also funds OSS-related work overlapping with basic research (e.g., National Science Foundation, 2023).

I focus on incentives of *firms*, and in the public sector, on *economic* incentives of governments. Motivations of individual developers (such as career concerns) appear only indirectly through *residual* labor supply curves, and I can speak to other justifications for government intervention (such as national defense) only by bounding justifications within the scope of my model. I focus on firms competing in product markets, conservatively holding fixed the supply of developer labor and OSS contributions not matched to the firms in my data. Future research that incorporates labor markets,[8] upstream firms, or other incentives could be useful for studying OSS policies that seem less justifiable based on my approach.

Related Literature. I primarily contribute to literatures on the economics of OSS, innovation spillovers, and industrial policy.

Starting with Lerner and Tirole (2002), the OSS literature mainly focuses on *individuals*.[9] In the theoretical literature, my work relates more to the model of Llanes and de Elejalde (2013), which highlights why *firms* choose to use and invest in OSS when competing in a product market. By building a richer model with more than two types of firms in different countries, I discipline my assumptions with data and contribute a new toolkit for policy evaluation. To my knowledge, I am the first to estimate an empirical model of firm involvement in OSS.[10] In the empirical literature, studies of OSS contributions are common, but data on OSS use are scarce. My website production estimates relate to those of Nagle (2018, 2019b), who uses survey data on OSS use and Linux contributions to run value-added production function regressions for US firms.[11] By matching contributors to website-owning firms, I create a novel dataset suitable for a new structural approach and policy simulations. In doing so, I introduce a novel method for estimating hours spent on different OSS activities and provide the first analysis of China's domestic OSS platform.

Second, I contribute to the literature on innovation spillovers.[12] At the core of my model are stocks of knowledge capital in the tradition of Griliches (1979). Key challenges in this

[8] Work on how labor competition affects firm participation in OSS includes the theoretical model of Kumar et al. (2011) and the more empirical work of Boysel et al. (2024).

[9] Early reviews include Lerner and Tirole (2005) and Von Krogh and Von Hippel (2006). Survey evidence includes Lakhani and Wolf (2005) and Nagle et al. (2020). Prominent motivations include on-the-job work, career concerns, intrinsic motivation, and reciprocal altruism.

[10] In a systematic literature review of firm participation in OSS, Li et al. (2024) categorize 25 publications as Business & Economics. Most are case studies or reduced-form analyses. Two articles (Economides and Katsamakas, 2006; Llanes and de Elejalde, 2013) present economic models, but neither is empirical.

[11] Hoffmann et al. (2024) use more aggregate data on OSS use and contributions, including for web OSS, to estimate OSS replacement costs. A related literature measures the effects of OSS use (Dushnitsky and Stroube, 2021) and investment (Conti et al., 2021; Wright et al., 2023, 2024) on entrepreneurial productivity.

[12] Bloom et al. (2019) reviews the literature from a policy-making perspective.

literature include defining capital and detecting "the path of the spillovers in the sands of the data" (Griliches, 1992). In my setting, capital refers to a codebase, and I directly measure spillovers as investment in OSS used by other firms.[13] My estimates corroborate and sharpen findings in the literature for the case of web OSS. Similar to Bloom, Schankerman, and Van Reenen (2013), I find that introducing business stealing through product market competition does not significantly dampen the large gains from innovation spillovers through web OSS. Consistent with Grossman and Helpman (1991) and Coe and Helpman (1995), I estimate high domestic returns from foreign investment in web OSS, and large opportunity costs from restricting foreign collaboration.

Third, I contribute to the literature on industrial policy.[14] Key challenges in this literature include the endogeneity of observed policies and the external validity of remaining random variation (Rodrik, 2012). My empirical model for the web development industry aligns with recent work that addresses these challenges using economic theory, granular data, and institutional context. Closest are two pairs of articles that estimate empirical models of China's shipbuilding and automobile industries. Kalouptsidi (2018) and Barwick, Kalouptsidi, and Zahur (2019) find limited benefits from costly shipbuilding subsidies. Barwick, Cao, and Li (2021) report large consumer damages from local protectionism for automobiles, while Bai, Barwick, Cao, and Li (2022) find significant innovation spillovers from foreign joint venture requirements. By comparing restrictions and subsidies, I contribute to a debate dating back at least to Hamilton's (1791) "infant industry" argument: I also find weak support for strategic trade policy in a particular industry, and highlight the potential benefits of industrial policy that promotes innovation spillovers.

The literature on industrial policy for OSS is much smaller. Using a differences-in-differences approach, Nagle (2019a) finds that a change in French procurement policy to favor OSS increased French contributions. Other work, including a recent European Commission report recommending subsidies and the creation of an OSS platform for the European Union (Blind et al., 2021), typically relies on case studies or descriptive panel regressions. In a high-level overview, Lerner and Schankerman (2010) urge caution about unintended costs of government intervention but emphasize the need for more empirical research that is (*i*) tailored to software development, (*ii*) aimed at government policy, and (*iii*) focused on the role of firms. These are precisely the aims of this article.

[13] Starting with Trajtenberg (1990) and Jaffe et al. (1993), the most common approach to directly measuring innovation spillovers uses patent citations. My approach is complementary, providing a more concrete but less general measure for web development.

[14] Lane (2020) and Juhász et al. (2023) compare the recent literature with older empirical work.

2. Background

I begin by providing background on OSS, including recent policies in China, which motivate my simulations in Section 7. My aim is not to argue that these policies have been highly impactful so far, but to highlight the technical and legal tools available to policymakers that allow for their continued expansion. Lastly, I describe the web development industry and why it is a suitable setting for the rest of my empirical analysis.

2.1. Open Source Software

OSS is created by software developers collaborating across geographic and organizational boundaries. The result is a wealth of computer programs that are (*i*) available under non-restrictive licenses and (*ii*) central to most private codebases. A recent survey found that 96% of commercial codebases use OSS, and 77% of total code is OSS (Synopsys, 2024). Other industry and academic reports similarly highlight the widespread use of OSS in private codebases (e.g., Sonatype, 2020; Musseau et al., 2022).

Many developers contribute to OSS as part of their job. A survey by the Linux Foundation and the Laboratory for Innovation Science at Harvard provides recent evidence (Nagle et al., 2020). Among 570 contributors to widely-used OSS projects, 89% are employed full-time, of whom 60% are encouraged to contribute as part of their job; 81% of those encouraged are paid for their time contributing. Out of 10 common motivations, the most frequently mentioned in respondents' top three is the need to contribute features or fixes to OSS they use at work. This is not surprising, as keeping modifications in-house often leads to incompatibilities with OSS updates. However, the next most frequent motivations relate to learning, enjoyment, and reciprocity—developers gain more from contributing than use-value alone.

Compared to the widespread use of OSS, government interventions are often indirect and limited. The Center for Strategic and International Studies (CSIS) publishes a survey of national OSS policies (Lostri et al., 2023). Among 539 policies approved by 108 governments, the CSIS identifies 45% that explicitly prioritize OSS in public procurement, while most others focus on education and studies about OSS. The two US policies approved in 2022 are a Department of Defence (2022) memorandum favoring OSS for procurement and a White House (2022) convention about OSS security. The US also funds OSS-related work overlapping with basic research (e.g., National Science Foundation, 2023). China's OSS policies are broader and more direct. Its last two Five-Year Plans (2016 and 2021) instruct agencies to promote OSS communities directly.[15] The Ministry of Industry and Information

[15] Section 1 quotes the latest Plan from 2021. In 2016, the "Plan for S&T Innovation" and the "Plan for

Technology (MIIT) is particularly involved, providing financial support for China's domestic OSS platform.

Worldwide, nearly all OSS collaboration is on one platform: GitHub.[16] Founded in 2008 and acquired by Microsoft in 2018, GitHub enhances a code versioning system called "Git" with bug tracking, code review, and social networking. GitHub reports over 100 million users (GitHub, 2024), though this likely includes many inactive users (Kalliamvakou et al., 2016) and those using the platform for non-OSS features, such as private projects. In Appendix A, I combine comprehensive public datasets on GitHub activity and identify around 3 million active contributors to public projects per month in 2022. Using self-reported locations, I estimate about 400,000 from the US and 130,000 from China.[17]

Like many of the world's largest websites with social networking features, GitHub has a Chinese counterpart: Gitee. Created in 2013 by Shenzhen Aosi Network Technology, also known as OSChina, Gitee has been officially backed and financially supported by the Chinese state since 2020 (MIIT, 2020). Gitee reports over 12 million users (Gitee, 2024). Unlike GitHub, there is no comprehensive dataset of all public Gitee activity. In Appendix A, I use Gitee's public-facing interface to create such a dataset, identifying around 75,000 active contributors per month in 2022. This amounts to 2.5% of GitHub's OSS community and half the size of GitHub's OSS community in China.

2.2. Restrictions in China

It is unusual for a global website with social networking features to be more popular in China than its domestic counterpart. In fact, it is unusual for a website like GitHub not to be fully blocked by the collection of government policies known as the Great Firewall (GFW).[18]

Instead, I provide empirical evidence that the GFW reduces the quality of internet connections between China and GitHub. I collect data from three sources that regularly test connectivity to popular websites from China, and in Figure 1, plot their success rates for GitHub. For a typical website, the lines are nearly flat at 100%. Fully blocked websites are

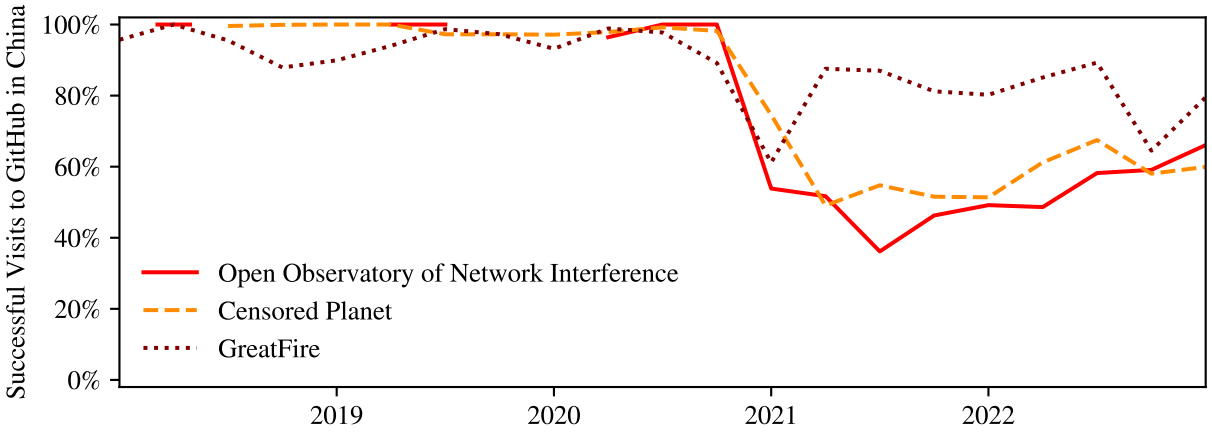
the Development of Strategic Emerging Industries" both recommend supporting OSS communities.

[16]Using comprehensive Ecosyste.ms data (Nesbitt, 2024) on the 2.5 million OSS packages with at least 2 versions from 2014 to 2023 and a repository URL, I find that 95% are on GitHub, 3% are split between Gitee, GitLab, and Bitbucket, and less than 0.1% are on SourceForge. Many of the remaining 2% use project-specific websites. GitLab and Bitbucket mainly compete with GitHub's non-OSS features. Many Gitee repositories, discussed below, also have a GitHub repository, often listed as the primary one.

[17]I discuss geocoding and users who do not self-report in Appendix A. Below, I validate the use of self-reported locations by comparing them with locations based on IP addresses.

[18]Full blocks of GitHub in China are rare and brief—a block in January 2013 (Protalinski, 2013) lasted only a couple of days. Other global social networking sites like Facebook and Twitter are fully blocked.

Figure 1: Success Rate of Visits to GitHub from China



This figure reports the quarterly success rate of visits to GitHub from China, excluding quarters with fewer than 10 attempted visits. The definition of success varies by source. For the Open Observatory of Network Interference (Filasto and Appelbaum, 2012), I consider both HTTP and HTTPS tests, discard failed tests, and define success as no confirmed blocks or anomalous measurements. For Censored Planet (Sundara Raman et al., 2020), I consider only HTTPS probes, and define success as no unexpected outcomes on their data dashboard. For GreatFire (greatfire.org), I discard false positives with empty replies, discard those with unsuccessful paired tests from within the US, and define success as a successful response code.

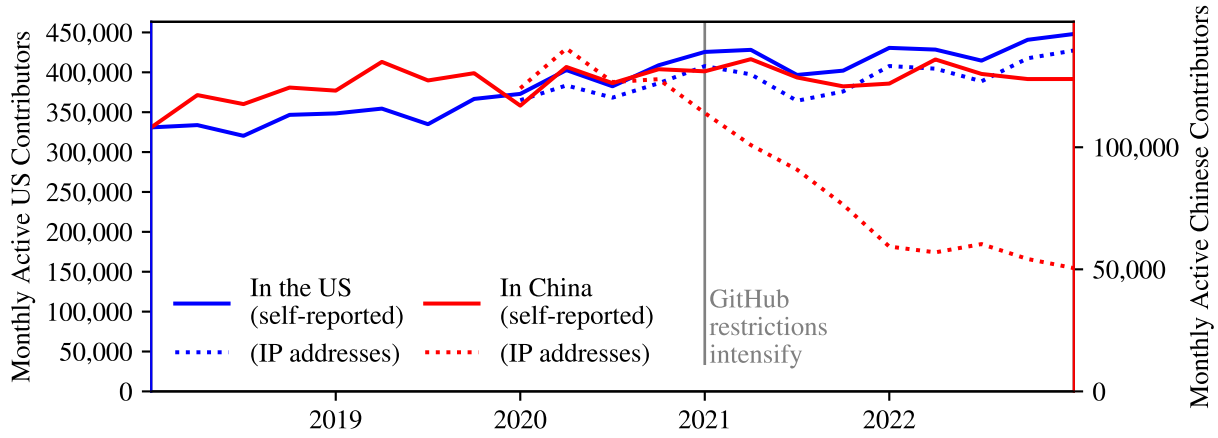
flat at 0%. Across all sources, the success rate for GitHub drops in 2021. Although GFW changes are rarely announced, I do not see a concurrent drop for other popular websites,[19] suggesting a deliberate policy shift to restrict GitHub access from within China.

A natural question is whether Chinese developers use GitHub less, now that direct access is more difficult. In Figure 2, I plot two measures of monthly active GitHub contributors in the US and China: one based on self-reported locations and the other on Internet Protocol (IP) addresses. For the IP-based measures, I use GitHub's Innovation Graph dataset, which provides country-quarter statistics back to 2020. The upward trend in the self-reported line for China does not change in 2021, suggesting that restricting GitHub access did not significantly affect China's contributions.

Reassuringly, the US line based on IP addresses is nearly identical to its self-reported counterpart. The IP-based line for China also tracks its self-reported counterpart through 2020, but begins to decline in 2021, just as GitHub restrictions intensified. By 2023, a 70,000-contributor gap between China's self-reported and IP-based lines likely reflects increased use

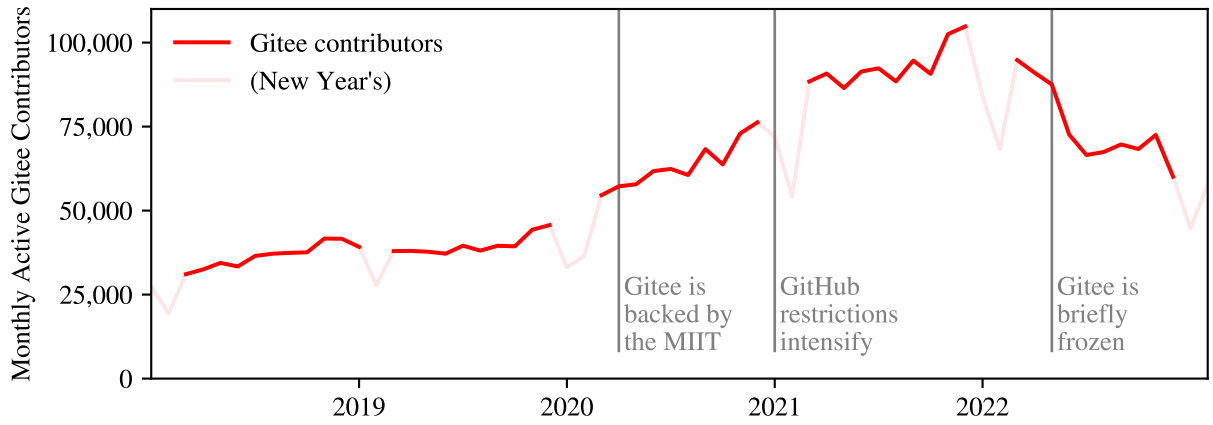
[19] I also find no drop for other countries. For other websites, I collect OONI and GreatFire data for all websites from Section 3 and create the same plot for the 203 with at least one test from both sources. The closest other drop is for signal.org, the website of an encrypted messaging app, which was blocked by the GFW in March 2021 (Associated Press, 2021). I am only aware of one other large change in 2021, when in November the GFW began blocking fully encrypted traffic (Wu et al., 2023).

Figure 2: Monthly Active Contributors on GitHub in the US and China



This figure reports two measures of monthly active GitHub contributors in the US (blue, left) and China (red, right), averaged by quarter. Active contributors are those who author a commit, issue, pull request (PR), review, or comment on a public GitHub repository. I describe my GitHub data and how I use self-reported locations (solid lines) in Appendix A. I scale country shares among self-reporting contributors by the total number of contributors, assuming equal rates of self-reporting across countries. I also compute a measure based on IP addresses using GitHub's Innovation Graph dataset, which is at the country-quarter level, and dates back to the first quarter of 2020. I scale its count of PRs from US and Chinese IP addresses by my own quarterly estimates of PRs per active US and Chinese contributors who do self-report (dashed lines).

Figure 3: Monthly Active Contributors on Gitee



This figure reports the number of monthly active contributors to public Gitee repositories. Active contributors are those who author a commit, issue, pull request, review, or comment on a public Gitee repository. I describe my Gitee data in Appendix A. Januaries and Februaries, which are notably affected by New Year's celebrations in China, are shaded in a lighter red.

of virtual private networks (VPNs), which, when used to bypass the GFW, register as foreign IP addresses.[20] Software developers in China are familiar with VPNs and value GitHub, so it is unsurprising that they are willing and able to circumvent new GFW restrictions.

Another natural question is whether Chinese developers began using Gitee more as it gained state support and GitHub became harder to access. In Figure 3, I plot the number of monthly active Gitee contributors. Growth sped up around Gitee's financial support from the MIIT in April 2020 and heightened GitHub restrictions in January 2021, but the absence of sharp jumps makes interpretation difficult. In May 2022, however, Gitee faced a major censorship event. The company "didn't have a choice" but to temporarily freeze all OSS projects for manual review and require developers to affirm that future OSS projects would not violate Chinese law (Yang, 2022). In my data, this event coincides with a sharp drop of 25,000 contributors and a reversal of Gitee's growth.

Due to VPN use, GFW restrictions seem to have had less impact on OSS collaboration than Gitee censorship. In 2017, the MIIT banned unapproved VPNs (MIIT, 2017), and while there have been several crackdowns, VPN use remains widespread (Chandel et al., 2019), despite recent slowdowns of popular VPNs (The Economist, 2024). However, VPN use can have consequences; in 2023, police confiscated around $140,000 in earnings from a Chinese developer, claiming he used an unauthorized VPN to bypass the GFW (The Guardian, 2023). The developer was paid to contribute to web development OSS on GitHub, managed by a foreign company.

I see China's restrictions on VPN use and OSS collaboration as evidence that its policymakers have the technical and legal infrastructure to extend and strengthen their existing OSS policies. So far, some restrictions appear motivated by political censorship, while subsidizing Gitee aligns more closely with the latest Five-Year Plan's goal of supporting domestic OSS. In this article, I aim to assess the *economic* effects of increased public sector involvement, particularly from China.

2.3. Web Development

In the rest of my empirical analysis, I focus on OSS for web development to address a key challenge in studying OSS: measuring use. Suppose we wanted to study OSS policy for large language models. It is easy to visit the GitHub page for Alibaba's OSS competitor

[20] As a result, for foreign countries through which Chinese developers route their VPN traffic when bypassing the GFW, we should expect an *increase* in the IP-based line, relative to the self-reported one. This is exactly what we see in similar plots for Hong Kong, South Korea, and Singapore—all desirable VPN server locations for Chinese developers due to their geographic proximity to China.

to ChatGPT, Qwen (github.com/QwenLM). Most of its contributors work, unsurprisingly, at Alibaba. There is plenty of GitHub data on OSS contributions and who makes them. However, there is very little public data on which firms value Qwen enough to use it in production.[21] The lack of direct data on who uses OSS makes it difficult to measure the value of OSS—and hence the economic effects of OSS policy—using revealed preferences.

Web development is an exception. Because of how websites work, it is straightforward to visit a website, download its source code, and, with some technical expertise, identify the software it uses.[22] Web development is a major part of OSS—JavaScript, the most popular language for OSS development (GitHub, 2023b), is primarily used for building websites.

Beyond being a convenient empirical setting for studying OSS, web development is a large and high-tech industry. In the US alone, $20 billion in annual wages are paid to around 186,000 web developers (Bureau of Labor Statistics, 2022) to create and maintain the technical structure of websites. Using online resume data, which I describe in the next section, I estimate that there are 926,000 web developers in China and 8 million worldwide, about 13% of all software developers.

3. Data

I construct a panel of firms that operate the world's most visited websites; technical details are in Appendices A to H. I identify the software these firms use to build their websites and how they allocate web developer labor to OSS and in-house investment. Then, I document patterns of how firms use and invest in OSS at the country level, describe my firm-level sample, and document firm-level patterns that will inform my model.

3.1. Panel Construction

I start with homepages of the 10,000 most visited websites in each month from 2014 to 2022. In Appendix B, I describe my historical data on homepages, which are archived by Common Crawl. Global website traffic ranks are from Alexa.[23] In Appendix C, I convert ranks into traffic by following Chevalier and Goolsbee (2003) and fitting a power law on a shorter panel

[21] Alibaba claims over 90,000 corporate users (Jiang, 2024). Package download statistics provide some insight into downloads by region but are noisy and rarely identify specific firms. Nagle et al. (2022) collect anonymized OSS usage data from software composition analysis companies but also cannot identify firms.

[22] Others have used this feature to value OSS servers (Greenstein and Nagle, 2014; Murciano-Goroff et al., 2021, 2024; Ackermann and Greenstein, 2024) and web OSS in aggregate (Hoffmann et al., 2024).

[23] Alexa, an Amazon subsidiary, published a daily top 1 million list until its discontinuance in 2023. Alexa's traffic data primarily came from its toolbar extension, along with other undisclosed extensions and traffic meters installed by website operators (Pochat et al., 2018; Shiller et al., 2018).

from 2018 to 2020 (from Johnson et al., 2023) with the traffic behind Alexa ranks. I assign websites to 80 markets with Similarweb categories, designed for use by website operators.[24]

Detecting Software. In Appendix D, I identify the software behind each archived homepage by analyzing its source code and other browsing data with Wappalyzer, a tool commonly used by companies to generate business leads.

I focus on three key parts of a website's technical structure: its backend, JavaScript, and user interface (UI) frameworks.[25] Starting with Greenstein and Nagle (2014), most research on web OSS has centered on a fourth part, web servers, which play a critical role in global internet infrastructure. While I also collect data on servers for comparison, they are not my focus because of relatively little OSS activity on popular servers during my sample period and increasing use of externally-managed alternatives with limited data.[26]

I focus on the 27 frameworks in Appendix Table D1 that account for 99% of usage for each of the three website parts. All end up being OSS on GitHub. If no framework is detected for a website part, I assume the website uses a non-OSS alternative, which I label as "in-house." When Common Crawl fails to archive a homepage, I assume its software remains unchanged from the previous month. From my final sample, I discard 8% of websites—typically smaller ones—without a valid archived homepage.

Matching. I manually match each detected OSS framework to its GitHub project page, also known as a repository. None of the 27 frameworks that I focus on have active Gitee repositories. Using the GitHub data described in Appendix A, I collect all contributions to these frameworks, made by a total of 223,371 contributors. For comparison, I estimate that the monthly active contributors in my sample represent 2% of those contributing to GitHub repositories of *all* active OSS packages.[27]

Next, I attempt to match each contributor to a LinkedIn profile, using employment information on their GitHub or Gitee profiles if unsuccessful. My LinkedIn data is a consolidated

[24]Similarweb specializes in web analytics. In my data, the most trafficked markets are "Search Engines," "TV, Movies & Streaming," "Social Networks & Online Communities," "News & Media," and "Marketplace."

[25]Backend frameworks (e.g., Rails) handle server-side logic. JavaScript frameworks (e.g., React) handle client-side interactions. UI frameworks (e.g., Bootstrap) provide styles for web interfaces.

[26]In my sample, Wappalyzer-detected use of managed alternatives (e.g., Azure and Acquia) grew from 3% in 2014 to 17% in 2022, likely an understatement due to server detection challenges. Of the 50% of website-years where Wappalyzer detects an OSS server, 93% use Apache or NGINX. Only a third of commits to Apache and NGINX occurred during my sample, compared to two-thirds for the web frameworks I focus on.

[27]In each month from 2014 to 2022, there are around 4,000 active contributors to the 27 OSS web frameworks that I focus on. My 2% number compares 4,000 with the number of GitHub users active on *any* repository linked in the comprehensive data on OSS packages that I describe in Footnote 16.

collection of 618 million individual profiles and 62 million company profiles, gathered through multiple scrapes of public information from 2016 through 2022. Profiles include full employment histories. When possible, I match firms listed on contributor profiles to information from LinkedIn, Orbis, Pitchbook, Compustat, Lightcast, and WHOIS website records. These sources often provide headquarter locations and, crucially, often allow me to identify which websites each firm operates.

Contributor and firm matching both follow three-step processes, which I design in Appendix E to be practical while accurately matching most contributions, especially those from China, where matching can be more challenging.[28] First, I conservatively match on clean and unique identifiers, such as contributors' emails and LinkedIn-assigned company numbers. Second, I manually fill in missing matches for top contributors and their employers. Third, using features from these matches as training data, I apply regularized gradient boosting (Chen and Guestrin, 2016) to predict the remaining matches for smaller contributors and less prominent employers. In Appendix F, I train similar classifiers to impute missing markets for some websites and missing countries for some firms and contributors.

Measuring Investment. Matching contributors to their website-owning employers allows me to measure the OSS investment from firms in my sample. My LinkedIn data also provides the total number of web developers employed by these firms. A strength of my data is the wealth of information about OSS contributions; the challenge is placing different types of contributions on the same scale. A weakness is the lack of detailed information about in-house investment beyond the total number of employed web developers.

OSS collaboration involves much more than writing code. In my data, developers contribute by authoring commits, issues, pull requests (PRs), reviews, and comments.[29] Much of the OSS literature focuses on individual activities—typically commits—or converts lines of code into hours and dollars using back-of-the-envelope calculations.[30] Instead, in Appendix G, I develop a simple data-driven approach tailored to the software in my data that

[28] I describe how I adjust for China's low LinkedIn penetration rate below and in Appendix H. This adjustment affects my measure of in-house investment. OSS investment comes from contributor matching. To improve the Chinese match rate for contributors, I incorporate Chinese social network identifiers and Gitee information into the matching procedure described in Appendix E.

[29] I describe contribution types in Appendix A. Commits are changes or updates to a codebase. PRs are bundles of commits with a proposal for incorporation. Reviews evaluate code in PRs. Issues are bug reports, feature requests, questions, or other tasks. Comments are follow-ups on PRs or issues.

[30] Nagle (2019a), Robbins et al. (2021), Blind et al. (2021), and Hoffmann et al. (2024) convert lines of code (LOC) into hours using the Constructive Cost Model (COCOMO) II (Boehm, 1984; Boehm et al., 2009), a typical version of which estimates person-hours with $\alpha \times \text{LOC}^{\beta}$ using pre-calibrated values of α and β.

Figure 4: Hours Invested into Web OSS by Contribution Type

This figure reports how much pull requests (PRs), issues, comments, commits, and reviews contribute to estimated hours invested into the web OSS in my sample. I describe contribution types in Appendix A and estimation in Appendix G. Commits are changes or updates to a codebase. PRs are bundles of commits with a proposal for incorporation. Reviews evaluate code in PRs. Issues are bug reports, feature requests, questions, or other tasks. Comments are follow-ups on PRs or issues.

accounts for non-code contributions.

The key requirement for my new approach is an estimate of how many hours per month OSS contributors (similar to those in my sample) spend on OSS work. In a recent survey of top OSS contributors (Nagle et al., 2020), those actively contributing to web OSS report spending an average of 52 hours per month on OSS. At the contributor-month level, I regress this number on counts of each contribution type made by the contributors in my sample to all OSS.[31]

The coefficients in Appendix Table G1 are "hedonic" estimates of hours per contribution type, which I use to place all contributions in my data on the same scale: hours. Figure 4 decomposes the 592,000 estimated hours of OSS investment in my sample. Commits account for only 10% of hours once other contribution types, especially PRs, are included, as PRs bundle multiple proposed commits.[32] Since OSS survey and contributions data are widely available, I am optimistic that my approach can be used to derive more comprehensive and context-specific measures of OSS investment in other settings.

For in-house investment, the challenge is a *lack* of data. The best information I have on how much labor each firm allocates to web development is the total number of employed web developers from my LinkedIn data. In Appendix H, I identify web developers and adjust for country-level, time-varying LinkedIn penetration rates.[33] To convert web developer counts into hours, I scale by country-level measures of average web developer workweeks from recent large-scale surveys (Stack Overflow, 2019, 2020). Finally, I subtract the hours invested into in-sample web OSS to derive my measure of in-house investment.

[31] 52 hours was for contributions to all OSS, not just web OSS. I differentiate between contributors who ever review others' public code and those who do not, which I find to be a particularly predictive variable.

[32] This does *not* imply that empirical analysis of commits alone understates OSS investment by an order of magnitude. Commits are correlated with other activity types, especially PRs. If the same regression were run with only commits, its coefficient would be larger, reflecting the activity correlated with each commit.

[33] Estimating a penetration rate for web developers requires assumptions. I assume its *ratio* to the rate for all workers in a country is the same across countries, and estimate this ratio for the US using BLS data.

Table 1: Country-Level Web OSS Investment and Use, 2014 to 2022

Web OSS Investment				Web OSS Use			
Percent of Total		Per Web Developer-Hour		Percent of Total		Per Potential Use	
1. US	31.3%	1. Netherlands	0.0083%	1. US	40.8%	1. India	16.8%
2. China	12.9%	2. US	0.0055%	2. China	10.7%	2. Canada	14.7%
3. Germany	6.2%	3. Germany	0.0046%	3. India	6.1%	3. UK	13.8%
4. Japan	6.0%	4. Canada	0.0037%	4. UK	5.7%	4. US	13.3%
5. UK	4.9%	5. UK	0.0037%	5. France	4.7%	5. China	11.8%
6. Canada	3.4%	6. France	0.0023%	6. Russia	2.7%	6. Netherlands	11.5%
7. India	3.2%	7. Japan	0.0021%	7. Germany	1.4%	7. France	11.3%
8. France	3.0%	8. Russia	0.0006%	8. Japan	1.4%	8. Germany	10.3%
9. Russia	2.5%	9. China	0.0004%	9. Canada	1.1%	9. Russia	9.0%
10. Netherlands	2.5%	10. India	0.0001%	10. Netherlands	0.7%	10. Japan	6.1%
↪ Combined	75.9%			↪ Combined	75.3%		

This table reports patterns of web OSS investment and use from 2014 to 2022 for the top 10 countries, ranked in the first column by their total investment into web OSS in my sample. The second column re-ranks these countries by their web OSS investment as a percentage of the total hours worked by web developers in each country. The third column ranks the countries by their top websites' share of worldwide web OSS use, while the fourth column ranks them by their web OSS use relative to the number of potential uses, calculated by multiplying the number of top websites owned by firms in a country by three, corresponding to the three considered website parts: backend, JavaScript, and user interface frameworks.

3.2. Country-Level Patterns

On the left of Table 1, I rank the top 10 countries by their investment into the 27 web OSS in my data. The overall pattern aligns with earlier research on the geography of GitHub contributions (e.g., Gonzalez-Barahona et al., 2008; Wachs et al., 2022). The clear leader is the US, followed by China. Relative to total hours worked by each country's web developers, the US and many European countries tend to invest in web OSS more intensively, while China and India tend to invest less.

On the right, I re-rank the same countries by how much their top websites *use* the same web OSS. I assign websites to countries based on the headquarters of their firms. Again, the US is the clear leader, with China in second. However, there is less cross-country variation. Compared to concentrated investment, dispersed use suggests that benefits of OSS investment may largely flow from the US to the rest of the world.

3.3. Firm-Level Sample Restrictions

I construct my firm-level panel at a monthly frequency, as this is the default frequency for most of my data sources. However, website software rarely changes, and some data, such as

a significant portion of LinkedIn employment dates, are reported annually. Because of this, I aggregate to a yearly frequency in the rest of my empirical analysis. I keep each website's earliest software in a given year, and sum monthly traffic and investment to a yearly level.

In my firm panel, I apply three restrictions to discard low-quality and less-relevant data. Since online resume data can conflate employment with education, I exclude firms with any websites in the education and science Similarweb category. To focus on firms that genuinely develop their own websites, I also discard firm-years with no employed web developers on LinkedIn. The restricted sample is an unbalanced panel of 4,135 firms operating 6,392 websites over 9 years, totaling 18,895 firm-years and 28,292 website-years. About 15% of firms and websites—typically small ones—churn in and out of the sample each year.

I do *not* discard any aggregate information about OSS investment from outside the firms in this sample. Doing so would understate the size of OSS codebases. Instead, in the rest of my empirical analysis, I treat OSS investment outside my restricted sample as exogenous.

3.4. Firm-Level Patterns

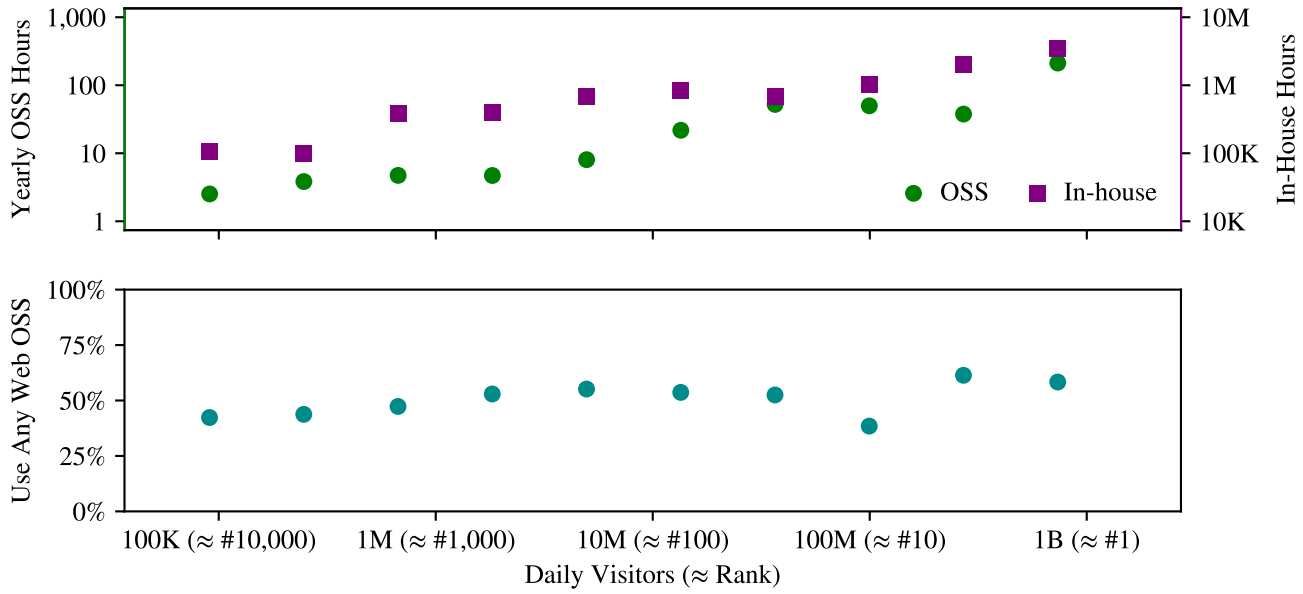
In the top panel of Figure 5, I plot the relationship between a firm's web traffic and its labor allocated to web OSS and in-house investment. The binscatters are on a log scale and slope upward—larger firms invest proportionally more in both OSS and in-house work. However, OSS investment is four orders of magnitude lower than in-house investment into web development. A typical firm in my sample invests only 1-1,000 hours per year into top web OSS, but a modest 50 web developer employees equates to around 100,000 hours of annual in-house investment.

Since larger firms invest more, OSS investment is concentrated. Of the web OSS investment I attribute to in-sample firms, 43% comes from the top 10. However, even among smaller firms, small OSS investments are not uncommon. Across firm-years, 34% invest at least a small amount into any in-sample web OSS. In contrast, OSS usage is widespread and far less concentrated, with 41% of firm-years using in-sample web OSS in production. In the bottom panel of Figure 5, I find that web OSS use does not vary much with firm size.

Firms favor specific OSS. In Figure 6, I show that both investment and use are 1.4 to 1.8 times more likely for highly developed OSS with above-median overall investment and for domestically developed OSS.[34] If a firm already uses or invests in an OSS, it is even more likely—3.3 times more—to invest in or use it concurrently.

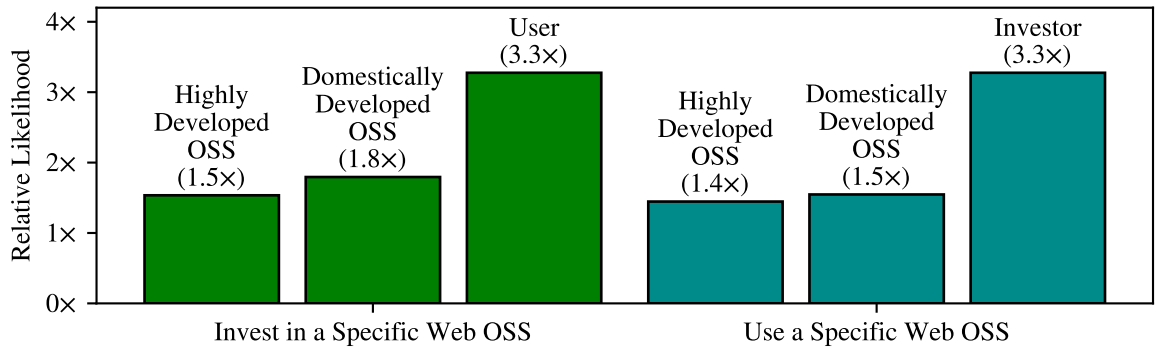
[34] I weight each firm-year-OSS combination by the share of hours invested in the OSS by contributors in same country as the firm. Manually classifying the home country of each OSS project's lead maintainer gives a slightly higher relative likelihood of 1.9 times for investment and about the same for use.

Figure 5: Web Traffic versus Web Development Investment and OSS Use



For each bin of traffic to a firm-year's websites (on a log scale), the top panel reports the hours the firm invests into the web OSS in my sample (green circles, left) and in-house web development (purple squares, right), averaged across firm-years (also on a log scale). The bottom panel reports the percentage of firm-years in each bin that use one of these web OSS in production. The left bin contains the worst web traffic rank in the sample, and the right contains the best: Google.

Figure 6: Relative Likelihoods of Web OSS Investment and Use



This figure reports conditional shares of firm-year-OSS combinations where the firm invests in (left group) or uses (right group) the web OSS in my sample, relative to unconditional shares. Highly developed OSS are those with above-median total investment. The domestically developed OSS bars are weighted by the share of hours invested in the OSS by firms or other contributors in the same country as the firm. Users are firms currently using the OSS in production, and investors are those currently investing in the OSS.

Lastly, investment and use are also persistent: if a firm invested in or used a web OSS last year, it is 14 and 38 times more likely, respectively, to do so again the following year. Firms favoring OSS that is (*i*) highly developed, (*ii*) domestically developed, (*iii*) self-made, and (*iv*) previously invested in or used will feature prominently in my model as reflecting (*i*) public, (*ii*) localized, (*iii*) private, and (*iv*) persistent benefits of OSS investment.

4. Model

I model private incentives for providing and using OSS in the web development industry. Though I focus on firms competing for web traffic, not developer labor, developer motivations still influence each firm's residual labor supply. Rather than strictly defining "firm" versus "worker" decisions, I model their *joint* incentives but use "firm" for simplicity.

Each year t, firms $f \in \mathcal{F}_t \subset \mathcal{F}$ choose which software to use to build their websites $j \in \mathcal{J}_{ft} \subset \mathcal{J}_t$ and how much labor to invest in software capital stocks. Firms can use and invest in different OSS frameworks $s \in \mathcal{OS}_t$ and in-house software $s = f$. Their choices affect web development costs through wages and website quality through a production function. After firms make choices, consumers of types $i \in \mathcal{I}_t$ choose which websites to visit based on quality, firm country, and content language. Firms then receive profit—revenue from traffic minus labor costs—and consumers receive utility. Firms are myopic, but persistence arises from capital accumulation, switching costs, and autocorrelated unobservables.

4.1. Website Supply

Each year t, firm $f \in \mathcal{F}_t$ is endowed websites $j \in \mathcal{J}_{ft}$. Websites have parts $p \in \mathcal{P}$. For each part, I take as given the set of OSS frameworks $\mathcal{OS}_{pt} \subset \mathcal{OS}_t$ that could be used for that part. In my data, there are $|\mathcal{P}| = 3$ parts (backend, JavaScript, and user interface frameworks), and by $t = 2022$, there are $|\mathcal{OS}_t| = 27$ major web OSS frameworks in Appendix Table D1.

For each website and part, the firm makes a discrete choice of software $s_{jpt} \in \mathcal{OS}_{pt} \cup \{f\}$ between using one of the relevant OSS frameworks or in-house software. The firm can also choose to invest web developer labor $\ell_{fst} \geq 0$ into improving each software $s \in \mathcal{OS}_t \cup \{f\}$, including software that it does not use in production.[35] Labor improves software capital $\boldsymbol{K}_t$, which I define below. In total, there are $|\mathcal{OS}_t| + |\mathcal{F}_t|$ different software, but each firm can only use and invest in $|\mathcal{OS}_t| + 1$: all OSS and its own in-house software.

Collect the firm's choices of software across parts in $\boldsymbol{s}_{jt} = \{s_{jpt}\}_{p \in \mathcal{P}}$ and across websites in $\boldsymbol{s}_{ft} = \{\boldsymbol{s}_{jt}\}_{j \in \mathcal{J}_{ft}}$. Collect its choices of labor across software in $\boldsymbol{\ell}_{ft} = \{\ell_{fst}\}_{s \in \mathcal{OS}_t \cup \{f\}}$. In

[35] Developers may contribute to OSS not used in production because they value OSS work as an amenity.

total, the firm makes $|\boldsymbol{s}_{ft}| = |\mathcal{J}_{ft}| \times |\mathcal{P}|$ discrete choices of software and $|\boldsymbol{\ell}_{ft}| = |\mathcal{OS}_t| + 1$ continuous choices of labor to maximize its profit:

$$\max_{\boldsymbol{s}_{ft}, \boldsymbol{\ell}_{ft}} \Bigg\{ \underbrace{\sum_{j \in \mathcal{J}_{ft}} R_{jt} Q_{jt}(\boldsymbol{\xi}_t)}_{\text{Revenue from traffic}} - \underbrace{\sum_{s \in \mathcal{OS}_t \cup \{f\}} W_{fst}(\boldsymbol{s}_{ft}, \boldsymbol{\ell}_{ft}, \boldsymbol{K}_t) \ell_{fst}}_{\text{Cost of labor}} \Bigg\} \quad \text{s.t.} \quad \underbrace{\xi_{jt} = \Xi_{fjt}(\boldsymbol{s}_{jt}, \boldsymbol{K}_t)}_{\text{Quality production}}. \tag{1}$$

Web traffic $Q_{jt}(\boldsymbol{\xi}_t)$ depends on the set of potentially all websites' quality $\boldsymbol{\xi}_t = \{\boldsymbol{\xi}_{jt}\}_{j \in \mathcal{J}_t}$, produced by a production function: $\xi_{jt} = \Xi_{fjt}(\boldsymbol{s}_{jt}, \boldsymbol{K}_t)$. Web traffic generates R_{jt}, marginal revenue net of marginal costs.[36] Profit is revenue minus the cost of web developer labor, determined by a wage function: $W_{fst}(\boldsymbol{s}_{ft}, \boldsymbol{\ell}_{ft}, \boldsymbol{K}_t)$.

Software Capital. For notational simplicity, I allow both the quality and wage functions to depend on the set of potentially all software capital, and suppress its dependence on labor:

$$\boldsymbol{K}_t = \Big\{ \underbrace{\{K_{ft}\}_{f \in \mathcal{F}_t}}_{\substack{\text{Private} \\ \text{capital}}}, \underbrace{\{K_{st}\}_{s \in \mathcal{OS}_t}}_{\substack{\text{OSS} \\ \text{capital}}}, \underbrace{\{K_{cst}\}_{c \in \mathcal{C}, s \in \mathcal{OS}_t}}_{\substack{\text{Country-specific OSS} \\ \text{capital components}}}, \underbrace{\{K_{fst}\}_{f \in \mathcal{F}, s \in \mathcal{OS}_t}}_{\substack{\text{Firm-specific OSS} \\ \text{capital components}}} \Big\}. \tag{2}$$

Not all of these capital stocks affect each quality and wage function. Before making this explicit by parameterizing these functions, I define each capital stock. First, each firm $f \in \mathcal{F}_t$ has a *private* capital stock K_{ft}, created by its total investment $L_{ft} = \ell_{fft} + \sum_{s \in \mathcal{OS}_t} \ell_{fst}$,[37] which depreciates at a rate of $\delta_P \in [0, 1]$ as code becomes obsolete:

$$K_{ft} = \big(1 - \delta_P\big) K_{ft-1} + L_{ft}. \tag{3}$$

Second, each OSS $s \in \mathcal{OS}_t$ has a *public* capital stock K_{st}, created by all firms' investment in the OSS, which depreciates at a potentially different rate of $\delta_O \in [0, 1]$:

$$K_{st} = \big(1 - \delta_O\big) K_{st-1} + \sum_{f \in \mathcal{F}_t} \ell_{fst}. \tag{4}$$

The model's central externality arises from the innovation spillovers in the sum over firms.

[36] Holding R_{jt} fixed is an empirical simplification due to limited data on website revenue, except for public firms. With more data, the model could be extended to let $R_{jt} = R_{jt}(\boldsymbol{s}_{jt}, \boldsymbol{K}_t)$ depend on software and capital, and also on endogenous prices in other industries where product prices are nonzero and observed.

[37] K_{ft} could alternatively be modeled as the discounted sum of only in-house investment ℓ_{fft}. Since ℓ_{fft} is orders of magnitude larger than OSS investment (Section 3.4), K_{ft} will be numerically nearly identical either way. My chosen model of K_{ft} ends up being more convenient when I fix L_{ft} in Section 5.

Public benefits of OSS investment, however, depend on who contributes. To keep track of how OSS investment from a specific country or firms benefits some firms more than others, I rewrite (4) by partitioning it into components K_{cst} contributed by each country $c \in \mathcal{C}$ and sub-components K_{fst} contributed by each firm $f \in \mathcal{F}_c$:

$$K_{st} = \sum_{c \in \mathcal{C}} K_{cst}, \tag{5}$$

$$\hookrightarrow \quad K_{cst} = \sum_{f \in \mathcal{F}_c} K_{fst}, \tag{6}$$

$$\hookrightarrow \quad K_{fst} = \big(1 - \delta_O\big) K_{fst-1} + \ell_{fst}. \tag{7}$$

Developer Wages. To invest labor into different software capital stocks, the firm pays different wages, which I parameterize as follows:

$$\log W_{fst}(\boldsymbol{s}_{ft}, \boldsymbol{\ell}_{ft}, \boldsymbol{K}_t) = \underbrace{\text{Dev}_{fst}(\boldsymbol{\ell}_{ft}, \boldsymbol{K}_t)}_{\substack{\text{“Development”} \\ \text{component}}} + \underbrace{\sum_{j \in \mathcal{J}_{ft}} \frac{Q_{jt}}{Q_{ft}} \sum_{p \in \mathcal{P}} \alpha_p \, \text{Ops}_{fjpt}(s_{jpt}, \boldsymbol{K}_t)}_{\substack{\text{“Operations”} \\ \text{component}}} + \underbrace{\bar{\omega}^W_{ft}}_{\substack{\text{Firm-year} \\ \text{fixed effect}}}. \tag{8}$$

I name the two components after the “DevOps” philosophy in software engineering, which distinguishes between development (i.e., improving software) and operations (i.e., maintaining software in production). I parameterize the development component to rationalize firms' labor choices $\boldsymbol{\ell}_{ft}$. I include a residual labor supply elasticity η, characteristics of the software being improved, and a switching cost κ_L:[38]

$$\text{Dev}_{fst}(\boldsymbol{\ell}_{ft}, \boldsymbol{K}_t) = \tfrac{1}{\eta} \underbrace{\log L_{ft}}_{\substack{\text{Log total} \\ \text{labor}}} + \boldsymbol{\gamma}_L' \underbrace{\boldsymbol{X}^W_{fst}(\boldsymbol{K}_t)}_{\substack{\text{Software} \\ \text{characteristics}}} + \kappa_L \underbrace{\mathbb{1}_{\ell_{fst-1}=0}}_{\substack{\text{Labor} \\ \text{switching}}} + \underbrace{\omega^L_{fst}}_{\substack{\text{Development} \\ \text{unobservable}}}. \tag{9}$$

I parameterize the operations component to rationalize firms' software choices $\boldsymbol{s}_{ft}$. In (8), I weight by traffic and importance parameters α_p when aggregating across each website and part.[39] To rationalize a firm's choice of software s_{jpt} for a specific website and part, I include the same software characteristics and a second switching cost κ_S:

$$\text{Ops}_{fjpt}(s, \boldsymbol{K}_t) = \boldsymbol{\gamma}_S' \underbrace{\boldsymbol{X}^W_{fst}(\boldsymbol{K}_t)}_{\substack{\text{Software} \\ \text{characteristics}}} + \kappa_S \underbrace{\mathbb{1}_{s \neq s_{jpt-1}}}_{\substack{\text{Software} \\ \text{switching}}} + \underbrace{\omega^S_{jst}}_{\substack{\text{Operations} \\ \text{unobservable}}}. \tag{10}$$

[38] To keep notation simple, I suppress the dependence of wages on past choices.

[39] I suppress the dependence of $Q_{jt} = Q_{jt}(\boldsymbol{\xi}_t)$ and $Q_{ft} = \sum_{j \in \mathcal{J}_{ft}} Q_{jt}(\boldsymbol{\xi}_t)$ on $\boldsymbol{\xi}_t$.

I select software characteristics to explain the firm-level patterns of investment and use from Section 3.4. OSS investment scales with total investment, and firms tend to invest in and use highly developed, domestically developed, and self-made OSS:[40]

$$\boldsymbol{X}^W_{fst}(\boldsymbol{K}_t)' = \underbrace{\mathbb{1}_{s\in\mathcal{OS}_t}}_{\substack{\text{OSS}\\ \text{indicator}}}\Bigg[\underbrace{\mathbb{1}}_{\substack{\text{OSS}\\ \text{indicator}}},\ \underbrace{\log K_{ft}}_{\substack{\text{Log private}\\ \text{capital}}},\ \underbrace{\log K_{st}}_{\substack{\text{Log OSS}\\ \text{capital}}},\ \underbrace{K_{c(f)st}/K_{st}}_{\substack{\text{Domestic}\\ \text{share}}},\ \underbrace{\mathbb{1}_{K_{fst}>0}}_{\substack{\text{Firm-specific}\\ \text{extensive margin}}}\Bigg[\underbrace{\mathbb{1}}_{\substack{\text{Firm-specific}\\ \text{extensive margin}}},\ \underbrace{\log K_{fst}}_{\substack{\text{Intensive}\\ \text{margin}}}\Bigg]\Bigg]. \tag{11}$$

Parameters in $\boldsymbol{\gamma}_L$ and $\boldsymbol{\gamma}_S$ capture (*i*) baseline costs of investing in and using OSS versus in-house software, (*ii*) how these costs vary with private capital, (*iii*) *public* benefits of OSS investment, (*iv*) *localization* of these benefits, and (*v*) *private* benefits of OSS investment. Private and localized benefits arise because firms contribute in ways that serve *their* needs and those of *similar* firms. To assess policies more specific than those in Section 7, additional characteristics could be incorporated to better capture specific needs.[41]

Finally, I include firm-year fixed effects $\overline{\omega}^W_{ft}$ in (8) and autocorrelated unobservables ω^L_{fst} and ω^S_{jst} in (8) and (10). These capture components of web developer wages not explained by the model, and, along with switching costs, help rationalize the observed persistence of firms' choices. I discuss their distribution and estimation in Section 5.

Website Quality. I parameterize the production function for website quality ξ_{jt} with the log of private capital and an importance-weighted average of OSS use across website parts:

$$\Xi_{fjt}(\boldsymbol{s}_{jt}, \boldsymbol{K}_t) = \beta_P \underbrace{\log K_{ft}}_{\substack{\text{Log private}\\ \text{capital}}} + \beta_S \underbrace{\sum_{p\in\mathcal{P}} \alpha_p \mathbb{1}_{s_{jpt}\in\mathcal{OS}_t}}_{\substack{\text{Importance-weighted}\\ \text{use of OSS}}} + \underbrace{\overline{\omega}^{\Xi}_{m(j)t}}_{\substack{\text{Market-year}\\ \text{fixed effect}}} + \underbrace{\omega^{\Xi}_{jt} + \varepsilon^{\Xi}_{jt}}_{\substack{\text{Quality}\\ \text{unobservables}}}. \tag{12}$$

The parameter β_P captures how private capital affects website quality, while β_S captures the effect of using OSS on quality. In Section 6, I do not find strong statistical evidence that β_S differs from zero, so I do not include other characteristics in $\boldsymbol{X}^W_{fst}(\boldsymbol{K}_t)$. In industries where OSS has a clearer effect on product quality, a more flexible parameterization would be appropriate to better capture how OSS affects quality and hence business stealing under product market competition.

[40] I include an extensive margin for the firm's own OSS contributions because these are often zero. For empirical analysis of investment, I only consider firm-years with employed web developers (Section 3.3), implying $K_{ft} > 0$. Since active $s \in \mathcal{OS}_t$ have at least one contribution, $K_{st} > 0$ as well.

[41] For instance, the domestic *language* share could inform policies that subsidize documentation translation.

Lastly, I include market-year fixed effects $\overline{\omega}^{\Xi}_{mt}$ and unobservables ω^{Ξ}_{jt} and ε^{Ξ}_{jt}. These capture components of website quality not explained by the model. I discuss their distributions and estimation in Section 5.

4.2. Website Demand

After firms make choices, consumers generate web traffic by making discrete choices.[42] I assign each website $j \in \mathcal{J}_t$ to a market $m(j) \in \mathcal{M}$. In my data, there are $|\mathcal{M}| = 80$ markets. In each, consumers of types $i \in \mathcal{I}_t$ choose to either visit a website $j \in \mathcal{J}_{mt} \subset \mathcal{J}_t$ or the outside option $j = 0$ of not visiting a website to maximize their indirect utility:

$$\max_{j \in \mathcal{J}_{mt} \cup \{0\}} \left\{ V_{ijt}(\xi_{jt}) + \varepsilon^{V}_{ijt} \right\} \quad \text{where} \quad V_{ijt}(\xi_{jt}) = \begin{cases} 0 & \text{if} \quad j = 0, \\ \underbrace{\boldsymbol{\beta}'_V \boldsymbol{X}^V_{ijt}}_{\substack{\text{Horizontal} \\ \text{“home bias”}}} + \underbrace{\xi_{jt}}_{\substack{\text{Vertical} \\ \text{quality}}} & \text{if} \quad j \in \mathcal{J}_{mt}. \end{cases} \tag{13}$$

I normalize the systematic utility of consumers' outside option $j = 0$ to zero. Websites $j \in \mathcal{J}_{mt}$ are vertically differentiated by quality ξ_{jt}. Beyond market segmentation, websites are horizontally differentiated according to "home bias" indicators in $\boldsymbol{X}^V_{ijt}$ for whether the website is operated by a firm in consumer type i's country and whether its content is in the consumer's primary language. Consumer types $i \in \mathcal{I}_t$ are country-language pairs.

I parameterize utility with firm country and content language because these variables are relevant to my focus on industrial policy—strong "home bias" will limit the amount of international competition in my policy simulations. Since I focus on the US and China, I include separate indicators in $\boldsymbol{X}^V_{ijt}$ for the US, China, English, and Chinese.

Website Traffic. Consumer choices generate traffic $Q_{jt}(\boldsymbol{\xi}_t)$. To derive an expression for traffic $Q_{ijt}(\boldsymbol{\xi}_t)$ from consumers of type $i \in \mathcal{I}_t$, I assume the non-systematic portion of utility, ε^V_{ijt}, is i.i.d. type I extreme value. Assuming a maximum potential traffic Q_{imt},

$$Q_{jt}(\boldsymbol{\xi}_t) = \sum_{i \in \mathcal{I}_t} Q_{ijt}(\boldsymbol{\xi}_t) \quad \text{where} \quad Q_{ijt}(\boldsymbol{\xi}_t) = \frac{\exp V_{ijt}(\xi_{jt})}{\sum_{k \in \mathcal{J}_{mt} \cup \{0\}} \exp V_{ikt}(\xi_{kt})} \cdot Q_{imt}. \tag{14}$$

After consumers make their choices of which websites to visit, firms then receive profit: net marginal revenue R_{jt} from traffic $Q_{jt}(\boldsymbol{\xi}_t)$ minus labor costs.

[42] Based on Comscore browsing micro data from the US in 2020, a discrete choice assumption seems reasonable. Among consumers visiting a top website $j \in \mathcal{J}_{mt}$, 75% visit no other $k \in \mathcal{J}_{mt} \setminus \{j\}$ on the same day, and 96% of the time spent on $\mathcal{J}_{mt}$ is on a single website.

4.3. Industry Equilibrium

I consider pure strategy Nash equilibria. An equilibrium in each year t is software and labor choices $\{\boldsymbol{s}_{ft}, \boldsymbol{\ell}_{ft}\}_{f \in \mathcal{F}_t}$ where firms maximize their profit in (1) and consumers maximize their utility in (13). Firms are myopic, but their choices affect future payoffs through capital accumulation and switching costs.

With multi-product firms and flexible preference heterogeneity, it is difficult to establish the existence and uniqueness of equilibria.[43] My model introduces positive externalities, further widening the scope for multiple equilibria. I focus on numerically computing equilibria for my policy simulations and assume an equilibrium exists when estimating the model.

5. Estimation

I estimate the model, report results in Section 6, and provide details in Appendices I to L. Firms' choices reveal depreciation rates, developers' wage function, and net marginal revenue from web traffic. Consumers' choices reveal their "home biases" and website quality. I incorporate statistical structure by making distributional assumptions about the model's econometric unobservables.

In Section 5.1, I match moments based on web traffic to estimate consumers' systematic utility function V_{ijt} (Berry, Levinsohn, and Pakes, 1995, 2004). Given website quality, in Section 5.2, I use standard timing assumptions to estimate its production function Ξ_{fjt} (Olley and Pakes, 1996; Levinsohn and Petrin, 2003; Ackerberg, Caves, and Frazer, 2015). Subject to the production function, in Section 5.3, I simulate firms' cost minimizing-choices and match simulated moments to estimate rates of capital depreciation δ_O and δ_P and the majority of developers' wage function W_{fst} (McFadden, 1989; Pakes and Pollard, 1989). In Section 5.4, I introduce stronger assumptions about profit maximization to estimate net marginal revenue R_{jt} from quality optimality conditions. In Section 5.5, I estimate the remainder of W_{fst} not identified by revealed preferences from cost minimization alone.

I summarize my estimation procedure in Table 2. For each of the five steps, I list the parameters estimated and, in words, how I identify them.

[43] Morrow and Skerlos (2010) review assumptions for the existence and uniqueness of Bertrand-Nash pricing equilibria under mixed logit demand, a case similar to equilibria in qualities here.

Table 2: Summary of Parameters and Estimation

Section 5.1: Demand Estimation			
ξ_{jt}	Website quality	$\Longleftarrow$	Traffic by website and year
$\boldsymbol{\beta}_V$	Consumer "home biases"	$\Longleftarrow$	Traffic by website and country
Section 5.2: Quality Production Function Estimation			
β_P	Effect of private capital on quality	$\Longleftarrow$	Traffic & total labor covariation
β_S	Effect of using OSS on quality	$\Longleftarrow$	Traffic & OSS use covariation
$\bar{\omega}^{\Xi}_{mt}$	Market-year quality fixed effect	$\Longleftarrow$	Traffic by market and year
Section 5.3: Revealed Preferences from Cost Minimization			
δ_O	OSS capital depreciation rate	$\Longleftarrow$	Firm choices & current vs. past OSS labor covariation
δ_P	Private capital depreciation rate	$\Longleftarrow$	Firm choices & current vs. past total labor covariation
$\boldsymbol{\alpha}$	Website part importance weights	$\Longleftarrow$	Software & characteristics covariation by part
$\boldsymbol{\gamma}_L$	Dev wage shifters	$\Longleftarrow$	Labor & characteristics covariation
$\boldsymbol{\gamma}_{S3:6}$	Majority of Ops wage shifters	$\Longleftarrow$	Software & majority of characteristics covariation
κ_L	Dev switching cost	$\Longleftarrow$	Labor autocorrelation
κ_S	Ops switching cost	$\Longleftarrow$	Software autocorrelation
ρ_L	Dev unobservable autocorrelation	$\Longleftarrow$	Initial labor & characteristics covariation
ρ_S	Ops unobservable autocorrelation	$\Longleftarrow$	Initial software & characteristics covariation
σ_L	Dev unobservable heterogeneity	$\Longleftarrow$	Labor variation
σ_S	Ops unobservable heterogeneity	$\Longleftarrow$	Software variation
$\bar{\omega}^{W}_{ft}$	Firm-year wage fixed effect	$\Longleftarrow$	Median web developer wages by country
Section 5.4: Net Marginal Revenue Estimation			
R_{jt}	Marginal revenue net of costs	$\Longleftarrow$	Quality optimality conditions
η	Residual labor supply elasticity	$\Longleftarrow$	From Roussille and Scuderi (2024)
Section 5.5: Revealed Preferences from Profit Maximization			
$\boldsymbol{\gamma}_{S1:2}$	Remaining Ops wage shifters	$\Longleftarrow$	Software & remaining characteristics covariation

This table summarizes the five steps of my estimation procedure, detailed in Sections 5.1 to 5.5. For each step, I list the estimated parameters and describe what identifies them. Since the parameters are estimated jointly in a parametric model, this table provides only high-level intuition. "Firm choices" are software $\boldsymbol{s}_{ft}$ and labor $\boldsymbol{\ell}_{ft}$. "Characteristics" refer to the software characteristics $\boldsymbol{X}_{fst}(\boldsymbol{K}_t)$ in (11). "Initial" refers to the years firms and websites enter the sample, where I assume no initial switching costs to distinguish state dependence from unobserved preference heterogeneity (Heckman, 1978, 1981).

5.1. Demand Estimation

I first estimate consumer "home biases" captured by $\boldsymbol{\beta}_V$ and website quality ξ_{jt} in (13) using PyBLP, my own OSS for demand estimation (Conlon and Gortmaker, 2020, 2025). I match one observed statistic per indicator in $\boldsymbol{X}^V_{ijt}$ with its simulated counterpart. Using Similarweb data on traffic shares by country, I match the share of global traffic to domestic firms (separately for the US and China) and to websites in each country's primary language (separately for English and Chinese).[44] Details are in Appendix I.

Estimation depends on the potential traffic Q_{imt} in (14) from consumers of type i to websites in market m. I assume global traffic $Q_t = \sum_{i,m} Q_{imt}$ is the number of internet users from the International Telecommunication Union, scaled by 20 websites per day.[45] To obtain Q_{imt}, in Appendix I, I scale Q_t by (i) a country-specific Similarweb-based adjustment to the global 20, (ii) each country-year's share of internet users, (iii) each country-language's Ethnologue population share (Lewis et al., 2016), and (iv) each market-year's within-sample traffic share.

5.2. Quality Production Function Estimation

Given website quality $\hat{\xi}_{jt}$ from demand estimation, I estimate the parameters β_P and β_S in its production function. In (12), market-year fixed effects $\bar{\omega}^{\Xi}_{mt}$ help account for bias from my imperfect measure of potential traffic (Zhang, 2024), while ω^{Ξ}_{jt} and ε^{Ξ}_{jt} are quality shocks that firms do and do not condition on, respectively, when making their production decisions.

To account for any simultaneity bias,[46] I use a fairly standard proxy variable approach (Olley and Pakes, 1996; Levinsohn and Petrin, 2003; Ackerberg, Caves, and Frazer, 2015). After removing fixed effects, I assume ω^{Ξ}_{jt} is a flexible function of private capital, weighted OSS use, and a proxy: total employees from my LinkedIn data. This is a proxy for ω^{Ξ}_{jt} in that I assume ω^{Ξ}_{jt} increases with the number of workers employed by the firm. A proxy helps isolate ω^{Ξ}_{jt} from ε^{Ξ}_{jt}, but it is not strictly necessary. In Appendix J, I compare with other approaches from the dynamic panel literature (e.g., Blundell and Bond, 1998, 2000), which relax some assumptions but impose others. Results are similar—I discuss them when presenting estimates in Section 6.

[44] I use FastText's (Joulin et al., 2016) lid.176.bin model to identify the most likely language of each website homepage's text content. Details on country, language, and market classification are in Appendix F.

[45] 20 is around the 99th percentile of daily in-sample visits in 2020 Comscore browsing data for the US. Since such market size assumptions are difficult to justify, I ensure my policy simulations in Section 7 are not driven by outside substitution, which is sensitive to such assumptions (Zhang, 2024).

[46] Omitting ω^{Ξ}_{jt} from a regression of quality on observables would likely introduce bias because firms condition on it when choosing labor and software.

More impactful are timing assumptions. I assume ω_{jt}^{Ξ} follows a first-order Markov process. While the firm can know the distribution of this process, I assume it cannot accurately predict next year's innovation in unobserved website quality:

$$\omega_{jt}^{\Xi} = \mathbb{E}\left[\omega_{jt}^{\Xi} \,\middle|\, \omega_{jt-1}^{\Xi}\right] + \nu_{jt}^{\Xi} \quad \text{where} \quad \mathbb{E}\left[\nu_{jt+1}^{\Xi} \,\middle|\, \boldsymbol{Z}_{fjt}^{\Xi}\right] = 0. \tag{15}$$

Given my assumptions about ω_{jt}^{Ξ}, I estimate β_P and β_S using a conventional two-step procedure described in Appendix J. In the first step, I net out ε_{jt}^{Ξ} by regressing estimated quality on a flexible function of private capital, weighted OSS use, total employees, and market-year fixed effects. In the second step, I form efficiently weighted moment conditions from innovations ν_{jt}^{Ξ}, recovered from a flexible regression of ω_{jt}^{Ξ} on ω_{jt-1}^{Ξ}.

To target β_P and β_S, I instrument with two lags of log total labor and weighted OSS use.[47] I do not include instruments in $\boldsymbol{Z}_{fjt}^{\Xi}$ to target depreciation of private capital δ_P or website part weights $\boldsymbol{\alpha}$. These parameters, which enter nonlinearly in (12), could theoretically be identified from variation in web traffic, as consumers might respond more to recent labor or changes in specific website parts. However, in practice, I find that firms' choices are far more informative about these parameters than consumer demand. As a result, I jointly estimate the quality production function with other parameters that inform firms' choices.

5.3. Revealed Preferences from Cost Minimization

Subject to the production function, the firm's profit maximization problem in (1) becomes cost minimization. The firm minimizes costs to achieve its quantity $Q_{jt}(\boldsymbol{\xi}_t)$ of web traffic:

$$\min_{\boldsymbol{s}_{ft}, \boldsymbol{\ell}_{ft}} \left\{ \sum_{s \in \mathcal{OS}_t \cup \{f\}} W_{fst}(\boldsymbol{s}_{ft}, \boldsymbol{\ell}_{ft}, \boldsymbol{K}_t) \ell_{fst} \right\} \quad \text{s.t.} \quad \hat{\xi}_{jt} = \Xi_{fjt}(\boldsymbol{s}_{jt}, \boldsymbol{K}_t). \tag{16}$$

I use firms' revealed preferences from their cost-minimizing choices to estimate the wage function and the rates of software capital depreciation. Estimates are informed by firms' choices about how they *allocate* their labor and *which* OSS they use. Recall that quality in (12) is produced by private capital and OSS use. Since my model lacks a labor market, I simply fix total labor L_{ft} to its observed value, and consider firms' labor allocation. This also fixes private capital K_{ft}, meaning the quality constraint limits whether each part of a website uses in-house software or OSS.

[47] I include two lags because, when just-identified, I sometimes find multiple estimates that satisfy the sample moment conditions, pointing to the under-identification issue documented by Ackerberg, Frazer, il Kim, Luo, and Su (2023). In my case, over-identification seems to resolve this issue.

Cost-minimizing choices are unaffected by positive affine transformations of the cost function, so revealed preferences from cost minimization do not identify the level or scale of costs. I assume firms' software and labor choices do not affect fixed costs (i.e., the level) and report only cost differences in my policy simulations. In Section 6.2, I scale each firm-year's costs by adjusting the firm-year fixed effects $\bar{\omega}^W_{ft}$ in (8) to match country-level wage data.

Since I fix total labor L_{ft} to its observed value, revealed preferences from cost minimization also do not identify the firm's residual labor supply elasticity η in (9).[48] I return to η below when estimating net marginal revenue.

I statistically rationalize firms' cost minimizing choices of labor and software with the wage unobservables ω^L_{fst} and ω^S_{jst} in (9) and (10). Similar to the quality unobservable ω^Ξ_{jt}, I assume each follows an AR(1) to capture persistence not explained by the model. However, unlike for ω^Ξ_{jt}, it is difficult to invert costs and form moment conditions from ω^L_{fst} and ω^S_{jst}.[49] Instead, I assume their innovations are Gaussian:

$$\omega^L_{fst} = \rho_L\, \omega^L_{fst-1} + \nu^L_{fst}, \qquad \nu^L_{fst} \sim N\big(0, \sigma^2_L\big), \qquad \omega^L_{fs0} \sim N\big(0, \sigma^2_L/(1-\rho^2_L)\big), \tag{17}$$

$$\omega^S_{jst} = \rho_S\, \omega^S_{jst-1} + \nu^S_{jst}, \qquad \underbrace{\nu^S_{jst} \sim N\big(0, \sigma^2_S\big)}_{\text{Gaussian innovations}}, \qquad \underbrace{\omega^S_{js0} \sim N\big(0, \sigma^2_S/(1-\rho^2_S)\big)}_{\text{Initial conditions from stationary distributions}}. \tag{18}$$

I use a full solution approach for estimation. For each guess of the free parameters in $\boldsymbol{\theta} = [\delta_O, \delta_P, \boldsymbol{\alpha}', \boldsymbol{\gamma}'_L, \boldsymbol{\gamma}'_S, \kappa_L, \kappa_S, \rho_L, \rho_S, \sigma_L, \sigma_S]'$,[50] I draw 10 histories of wage unobservables.[51] For each, I solve the constrained cost minimization problem in (16) for all firm-years, selecting the observed equilibrium.[52] To make this step feasible, I concentrate out software choices, use Hessians to optimize labor, and smooth discrete jumps. Appendix L provides details. To

[48] In principle, it is possible to relax this constraint and identify η by matching the average L_{ft}. However, without a model of the developer labor market or a different approach, I expect correlation between L_{ft} and wage unobservables $\bar{\omega}^W_{ft}$ to bias any such estimate of how wages respond to increases in total labor. My approach more credibly addresses how wages respond to changes in the *allocation* of labor.

[49] Invertibility of my demand system and production function allows for weak assumptions about unobserved quality. Labor nonnegativity and software discreteness complicate cost inversion. For example, any value of ω^S_{jst} within a small range would rationalize the same software choice. Partial identification could help but would pose computational challenges for policy simulations.

[50] These are all but a α_p, since $\sum_p \alpha_p = 1$, and $\boldsymbol{\gamma}_{S1:2}$, which are not identified by revealed preferences from cost minimization alone because fixing ξ_{jt} and L_{ft} fixes $\mathbb{1}_{s_{jpt} \in \mathcal{OS}_t}$ and K_{ft}. I return to $\boldsymbol{\gamma}_{S1:2}$ in Section 5.5.

[51] Only a few draws are needed because simulation error averages out as the number of firm-years increases: Ackerberg (2009) notes that for the method of simulated moments, using just 10 simulation draws increases the asymptotic variance by only 10% (McFadden, 1989; Pakes and Pollard, 1989).

[52] With externalities, multiple equilibria are possible. I select the one observed in the data by fixing the channels through which these externalities operate—K_{st} and K_{cst} in (5) and (6)—to their observed values after solving for all firms' choices. I fully solve for equilibria in the policy simulations in Section 7.

target each parameter in $\boldsymbol{\theta}$, I simulate statistics, average them over firm-years and simulation draws, and form a minimum distance objective function based on approximately efficiently-weighted deviations from observed counterparts.

I list all targeted moments in Appendix Table K1. To target $\boldsymbol{\gamma}_L$, I use log-transformed labor-weighted averages of the software characteristics in (11), replacing capital with sums over different horizons to target δ_O and δ_P. For $\boldsymbol{\gamma}_S$, I weight by traffic and software use, separately for each website part to target $\boldsymbol{\alpha}$. In both weighted averages, I include switching indicators to target κ_L and κ_S. To separate state dependence from persistence of unobserved preference heterogeneity (Heckman, 1978, 1981), captured by ρ_L and ρ_S, I assume no initial switching costs and interact averages for new firms and websites with initial year indicators.[53] Finally, I target the amount of this heterogeneity, captured by σ_L and σ_S, with standard deviations of labor and traffic shares across software.

5.4. Net Marginal Revenue Estimation

Fixing quality allows me to estimate web development costs without assuming that quality choices are profit-maximizing. This is desirable because much of website quality is determined by factors outside the scope of my model, such as website content.

However, fixing demand prevents me from conducting policy simulations involving competition for web traffic. For these simulations, I assume firms choose quality optimally to maximize profit while continuing to hold total labor L_{ft} fixed. Denoting the minimum cost from (16) as $C_{ft}(\hat{\boldsymbol{\xi}}_t, L_{ft})$, we can rewrite the firm's profit maximization problem in (1) as a choice over quality $\boldsymbol{\xi}_{ft} = \{\xi_{jt}\}_{j \in \mathcal{J}_{ft}}$:

$$\max_{\boldsymbol{\xi}_{ft}} \left\{ \sum_{j \in \mathcal{J}_{ft}} R_{jt} Q_{jt}(\boldsymbol{\xi}_t) - C_{ft}(\boldsymbol{\xi}_t, L_{ft}) \right\}. \tag{19}$$

Each choice of quality implies (i) a utility-maximizing choice of traffic, with parameters estimated from consumers' revealed preferences, and (ii) a cost-minimizing choice of software and labor, with parameters estimated from firms' revealed preferences from cost minimization. Balancing traffic against costs in profit maximization simulations requires estimating R_{jt}, marginal revenue net of marginal costs.

Data on website revenue is limited. Instead, I estimate R_{jt} using revealed preferences

[53] There is a remaining initial conditions problem (Heckman, 1981) for 44% of firms with an initial year before 2015. Choices depend on past choices, which are unobserved, as they depend on unobservables. I approximate 2014 choices by simulating them without switching costs.

implied by quality optimality. Consider the quality production function in (12). Using OSS instead of in-house software causes a discrete change in quality, but on the margin, firms increase the quality of all their websites by increasing private capital K_{ft} through additional labor L_{ft}. This implies a single quality optimality condition for each firm-year in my sample, which I use to estimate a common R_{ft} for its websites. Letting $\partial \boldsymbol{\xi}_{ft}$ represent a marginal change in all qualities at once and $L_{ft}(\boldsymbol{\xi}_{ft})$ the labor required to achieve them,

$$R_{ft} = \left(\frac{\partial Q_{ft}(\boldsymbol{\xi}_t)}{\partial \boldsymbol{\xi}_{ft}} \right)^{-1} \left(\frac{\partial C_{ft}(\boldsymbol{\xi}_t, L_{ft})}{\partial \boldsymbol{\xi}_{ft}} + \frac{\partial C_{ft}(\boldsymbol{\xi}_t, L_{ft})}{\partial L_{ft}} \frac{\partial L_{ft}(\boldsymbol{\xi}_{ft})}{\partial \boldsymbol{\xi}_{ft}} \right). \tag{20}$$

My estimator $\hat{R}_{ft}$ is an average over simulation draws. In Appendix K, I construct nearly all the above terms from my estimates, except for the derivative of cost with respect to total labor. This derivative is the average wage scaled by $1 + 1/\eta$, where η is the firm's residual labor supply elasticity in (8). Recall that because I fix total labor, revealed preferences from cost minimization do not identify η. Instead, I take η from Roussille and Scuderi (2024), who estimate a range of firm-level elasticities for US software developers using detailed data from Hired.com. I use their average, $\eta = 4.7$.[54]

5.5. Revealed Preferences from Profit Maximization

Lastly, I address the first two coefficients in $\boldsymbol{\gamma}_S$ on OSS use, $\mathbb{1}_{s \in \mathcal{OS}_t}$, and its interaction with the log of private capital, K_{ft}. Revealed preferences from cost minimization identify neither of these coefficients because fixing quality determines the choice between in-house software and OSS, and fixing total labor determines private capital. As a result, moments based on use-weighted averages of $\mathbb{1}_{s \in \mathcal{OS}_t}$ and $\log K_{ft}$ are perfectly matched during cost minimization.

Given net marginal revenue $\hat{R}_{ft}$, I estimate both coefficients by slightly modifying my estimation procedure in Section 5.3. I (i) allow firms to optimize profit and (ii) use only the previously perfectly-matched moments to form the minimum distance objective function. Computationally, profit maximization requires minimizing costs for each choice between in-house and OSS for all website parts. To make this feasible, I restrict firm-years with multiple websites to make this choice once per part and use only one set of wage unobservables.[55] Appendices K and L provide technical details.

[54] An $\eta = 4.7$ is not unusual among firm-level elasticities found in the literature across occupations (e.g., Naidu et al., 2018; Sokolova and Sorensen, 2021). It is close to the median of 4.8 estimated by Azar et al. (2022) using a similar approach with CareerBuilder.com but for more occupations.

[55] Using only one set increases the variance of my estimator, which I account for by re-drawing wage unobservables in my bootstrap procedure. I expect the restriction on multi-website firms to have minimal impact: across firm-year-parts with multiple websites, 87% either only use in-house software or only OSS.

6. Estimates

After presenting estimates, I incorporate additional data and assumptions to express them in dollars, allowing me to quantify the dollar effects of the next section's policy simulations. When possible, I use what limited data are available on web developer wages and website revenue to provide reassurance about my estimates.

To quantify uncertainty, I form bootstrapped confidence intervals by estimating the model 80 times, once for each set of bootstrapped traffic and OSS investment from Appendices C and G. I do not attempt to account for uncertainty from predictive matching and classification in Section 3.1.[56] When bootstrapping my estimation procedure, I re-sample firms and re-draw cost unobservables to account for both sampling and simulation error.

6.1. Demand Estimates

Demand estimation relies on two inputs: global traffic data from 2014 to 2022 and country-specific traffic shares for 2021 from Similarweb. Since my "home bias" estimates in $\hat{\boldsymbol{\beta}}_V$ match country-specific shares, they are based solely on 2021 data. However, I derive website quality $\hat{\xi}_{jt}$ from global traffic data spanning 2014 to 2022. Raw demand estimates $\hat{\boldsymbol{\beta}}_V$ can be difficult to interpret, so I relegate them to Appendix Table I1.

In Table 3, I report traffic-weighted likelihoods of consumers visiting a domestic or same-language website, compared to if it were foreign or in another language. From left to right, I consider preferences for firm country, content language, and both. Country and language explain similar variation, so their individual effects are comparable. Combined, their effects are dampened but still show strong "home bias." Consumers are 21 times more likely to visit a domestic website and 10 times more likely to visit one in their primary language. The country effect is largely driven by Chinese consumers—a large portion of global internet users—while the language effect is driven by non-English and non-Chinese speakers.

I use the rightmost specification. A key result is that consumers show a strong home bias for websites operated by domestic firms, especially in China. This is unsurprising—China's internet restrictions have decoupled the Chinese internet. My revealed preference approach translates decoupling into a strong home "preference" for Chinese consumers.[57] In the policy simulations, home bias will limit cross-border concerns about business stealing.

[56] Recent work on inference with flexibly-predicted data has focused on simpler models with clearer divides between observed and predicted observations (e.g., Angelopoulos et al., 2023; Zrnic and Candès, 2024). Explicitly modeling matching and classification errors could help but would pose computational challenges.

[57] Unlike other countries, where home bias likely reflects genuine consumer preferences, China's home bias reflects the *joint* preferences of its consumers and government. The latter prefers to restrict the former.

Table 3: Website Demand Estimates

		Preference Heterogeneity		
		Country	Language	Both
$\hat{\mathbb{P}}$(Visit | Same country) ÷ $\hat{\mathbb{P}}$(Visit | Different country)	All consumers	40.3 [32.0, 58.3]		21.3 [16.6, 33.1]
	↪ US-based	1.8 [1.7, 1.8]		1.3 [1.3, 1.4]
	↪ China-based	141.6 [106.9, 213.6]		75.7 [55.8, 123.8]
	↪ Other countries	4.2 [3.7, 5.0]		3.0 [2.7, 3.4]
$\hat{\mathbb{P}}$(Visit | Same language) ÷ $\hat{\mathbb{P}}$(Visit | Different language)	All consumers		39.7 [37.4, 43.9]	9.5 [9.5, 9.8]
	↪ English-speaking		1.5 [1.5, 1.5]	1.5 [1.5, 1.5]
	↪ Chinese-speaking		112.7 [97.3, 135.2]	5.1 [4.7, 5.7]
	↪ Other languages		19.1 [16.4, 22.4]	16.8 [16.3, 17.7]
Same country traffic share	All websites	31.6%		31.6%
	↪ US-based owner	28.6%		28.6%
	↪ China-based owner	91.2%		91.2%
Same main language traffic share	All websites		43.4%	43.4%
	↪ English content		32.7%	32.7%
	↪ Chinese content		94.7%	94.7%
Observation counts	Countries	138		138
	Languages		97	97
	Markets	80	80	80
	Website-years	152,622	152,622	152,622
	↪ Websites	58,106	58,106	58,106
	↪ Total years	9	9	9
	↪ Micro years	1	1	1

This table reports demand estimates for three specifications: country indicators in $\boldsymbol{X}_{ijt}^{V}$ on the left, language indicators in the middle, and both on the right. Appendix Table I1 provides the underlying parameter estimates. The top panel of this table reports more interpretable relative likelihoods of a consumer visiting a website if it is owned by a firm in the same country or has content in the consumer's primary language. These ratios are computed for each website-consumer type pair, with traffic-weighted averages reported separately by country and language. The middle panel reports the matched Similarweb statistics from 2021. In brackets, 95% confidence intervals are from 80 bootstrap samples that account for sampling error from traffic estimation in Appendix Table C1.

Dollar-Denominated Demand. In policy simulations where I allow firms to choose quality, I express changes in consumer welfare in dollars. I base this on a back-of-the-envelope calculation using results from a large-scale incentivized choice experiment conducted by Brynjolfsson et al. (2023) in 2022. The experiment measured Facebook users' willingness to accept (WTA) compensation for losing access to Facebook for a month.[58] It included nearly 40,000 participants from 13 countries, but for simplicity, I use the pooled WTA of $31.

In Appendix I, I simulate losing Facebook access for users in the same year and countries. I divide $31 by their average monthly welfare change and use the resulting conversion factor to scale $\hat{\boldsymbol{\beta}}_V$ in Appendix Table I1. I estimate that consumers are willing to pay between $2 in the US and $33 in China to spend their daily time on a domestic website rather than a similar foreign one. I trust this conversion more for US consumers than for those in China, who were not surveyed and whose traffic also reflects government preferences.

Sample Restrictions. The sample sizes at the bottom of Table 3 are larger than those of the restricted sample discussed in Section 3.3. Since consumer choices depend on the *full* set of alternatives in each market, I do not exclude any websites when estimating demand. I use quality estimates from this unrestricted sample when estimating the supply side, where it is important to discard low-quality and less-relevant investment data. For supply-side estimation and the policy simulations, I focus on endogenizing the restricted sample and treat everything else it depends on as exogenous.

6.2. Supply Estimates

I report supply estimates in Table 4. At the top, the estimated depreciation rate for OSS capital, $\hat{\delta}_O$, is much lower than for private capital, $\hat{\delta}_P$. Both are identified by how strongly firm choices correlate with lagged investment. My estimates suggest that web OSS investment retains value for years, but given my coarse measure of in-house investment, it is unsurprising that generic developer hours rapidly lose value. Below the depreciation estimates, I find sizable importance weights $\hat{\boldsymbol{\alpha}}$ for all three website parts.

In the quality column, I divide $\hat{\beta}_P$ and $\hat{\beta}_S$ by $\hat{\beta}_{V1}$, the domestic preference for consumers outside the US and China. I estimate that a 1% increase in a firm's private capital is comparable to its website being 0.2% closer to domestic for these consumers. This estimate is driven by the strong link between in-house investment and traffic, as shown in Figure 5 from Section 3.4. Consistent with the weaker link between OSS use and traffic in the same

[58] Researchers worked with Facebook to ensure compliance for a random sample of participants selected for payment. A similar estimate was obtained in an earlier US-only experiment (Brynjolfsson et al., 2019).

Table 4: Website Supply Estimates

		OSS Capital	Private Capital	
$\hat{\delta}_O$, $\hat{\delta}_P$	Yearly rate of capital depreciation	0.09 [0.05, 0.19]	0.72 [0.70, 0.75]	
		Backend	JavaScript	User Interface
$\hat{\boldsymbol{\alpha}}$	Website part importance weights	0.27 [0.24, 0.34]	0.38 [0.31, 0.41]	0.35 [0.29, 0.41]
		Quality	Development	Operations
$\hat{\beta}_P/\hat{\beta}_{V1}$	Log of private capital	+0.17 [+0.14, +0.20]		
$\hat{\beta}_S/\hat{\beta}_{V1}$, $\hat{\boldsymbol{\gamma}}_L$, $\hat{\boldsymbol{\gamma}}_S$	OSS indicator	−0.13 [−0.33, +0.07]	+2.97 [+2.82, +3.07]	+11.67† [+1.44, +15.06]
	× Log of private capital		+0.50 [+0.46, +0.62]	−0.34† [−0.45, +0.67]
	× Log of OSS capital		−0.48 [−0.51, −0.40]	−0.87 [−0.95, −0.80]
	× Domestic share of OSS capital		−0.55 [−0.60, −0.49]	−0.44 [−0.58, −0.34]
	× Any firm-specific OSS capital			−0.64 [−0.84, −0.52]
	↪ × Log of firm-specific OSS capital		−2.05 [−2.17, −1.77]	−0.21 [−0.37, −0.09]
$\hat{\kappa}_L$, $\hat{\kappa}_S$	Switching costs		3.96 [3.48, 4.10]	4.12 [3.71, 4.32]
$\hat{\rho}_L$, $\hat{\rho}_S$	Unobservable autocorrelation		0.59 [0.47, 0.63]	0.44 [0.25, 0.52]
$\hat{\sigma}_L$, $\hat{\sigma}_S$	Unobservable heterogeneity		0.48 [0.45, 0.55]	0.18 [0.11, 0.33]
	Firm-years	10,555	17,059	
	↪ Firms	2,481	3,980	
	↪ Years	7	8	

This table reports supply estimates. Those in $\hat{\boldsymbol{\gamma}}_S$ with † superscripts come from profit maximization. I divide $\hat{\beta}_P$ and $\hat{\beta}_S$ by $\hat{\beta}_{V1}$ from Appendix Table I1 to express their units as preferences for domestic websites for consumers outside the US and China; Appendix Table J1 provides unscaled estimates and those in dollars per daily visit. In brackets, 95% confidence intervals are from 80 bootstrap samples that account for sampling error from traffic estimation in Appendix Table C1, OSS investment estimation in Appendix Table G1, and demand estimation in Table 3. For each bootstrap sample, I re-sample firms and re-draw unobservables. Targeted moments are listed in Appendix Table K1, with details in Appendices J to L.

figure, my estimate of $\hat{\beta}_S$ is small, with a wide confidence interval that includes zero. Overall, I fail to find strong evidence that OSS use significantly impacts website quality. This could be due to weak internet user responses to OSS use, noise in my web traffic data, or both.

In Appendix J, I compare my proxy variable approach to others from the dynamic panel literature (e.g., Blundell and Bond, 1998, 2000), which rely on different assumptions. After accounting for simultaneity bias, my estimate $\hat{\beta}_P$ remains fairly stable, though noisier when differencing out website fixed effects. The story for $\hat{\beta}_S$ is similar as well—I find no strong evidence that using OSS improves or harms website quality. I expect OSS to have a greater impact on product quality in other industries, but for web development, my results suggest OSS plays a clearer role in firm costs.

For development, the first coefficient in $\hat{\boldsymbol{\gamma}}_L$ on the OSS indicator scales the amount of OSS investment to match its observed level. The coefficient on the log of private capital is more interpretable: it indicates that a 1% increase in a firm's private capital makes OSS work 0.5% more expensive, compared to in-house work. Negative coefficients on the remaining characteristics reflect value developers derive from contributing to OSS. The model translates higher likelihoods in Figure 6 from Section 3.4 of investing in highly developed, domestically developed, and self-made OSS into wage reductions from higher OSS capital K_{st}, its domestic share $K_{c(f)st}/K_{st}$, and the intensive margin of its firm-specific component K_{fst}.[59]

For operations, the first coefficient in $\hat{\boldsymbol{\gamma}}_S$ on the OSS indicator is large and noisy because it offsets the other coefficients in $\hat{\boldsymbol{\gamma}}_S$ and the noisily-estimated $\hat{\beta}_S$ to match the observed level of OSS use. The zero-including confidence interval on the log of private capital indicates that the operational cost of using web OSS does not clearly vary with the firm's private capital. Similar to the development column, the negative coefficients on the remaining characteristics reflect the higher likelihoods in Figure 6 of using highly developed, domestically developed, and self-made OSS. If a firm were to use OSS for all of its websites' parts, my estimates indicate that a 1% increase in their overall OSS capital, domestic shares, or firm-specific components would reduce required wages by 0.9%, 0.4%, and 0.2%, respectively.

My estimates indicate a wide gap between the private cost reductions from investing in OSS and the resulting public cost reductions enjoyed across firms, which could be addressed with subsidies. For a given amount of OSS investment, private and public benefits are of a similar order of magnitude for a *single* firm. However, public benefits are multiplied by the widespread use of that OSS across *many* firms. I estimate that these large public benefits are moderately, but not overpoweringly localized, which influences how much domestic subsidies

[59]There is no coefficient in $\hat{\boldsymbol{\gamma}}_L$ on its extensive margin because $K_{fst} > 0$ whenever $L_{fst} > 0$.

or restrictions on foreign collaboration can address coordination issues.

Finally, sizable switching costs $\hat{\kappa}_L$ and $\hat{\kappa}_S$, along with high unobservable autocorrelations $\hat{\rho}_L$ and $\hat{\rho}_S$, help explain the persistence of labor and software choices documented in Section 3.4. Unobservable scales $\hat{\sigma}_L$ and $\hat{\sigma}_S$ capture the variation in firms' choices left unexplained by the rest of the model.

Dollar-Denominated Supply. The scale of costs, determined by firm-year fixed effects $\overline{\omega}_{ft}^W$ in (8), does not affect cost minimization estimates but impacts policy simulations. Ideally, I would scale costs to match firm-level wage data, but such data are scarce. Instead, I match each firm-year's average hourly cost to the median wage of full-time web developers in the firm's country using large-scale developer surveys (Stack Overflow, 2019–2023). The matched wages for the US and China are \$57 and \$17, respectively, in 2017 dollars.[60]

These surveys also allow respondents to indicate if they have done extensive work with a subset of the OSS frameworks in my sample—8 to 10, depending on the year. For 225,339 response-OSS pairs, a regression of log wages on the (i) highly and (ii) domestically developed variables from Section 3.4 shows that working with web OSS that has above-median investment is associated with a 1% lower wage, and domestic web OSS with a 3% lower wage, after adjusting for cross-country differences with country-year fixed effects.[61] Although I do not expect these magnitudes to be comparable with my estimates, and the anonymous surveys do not report employer identities,[62] it is reassuring that both the survey data and my model indicate that it is cheaper to work on both highly developed and domestic OSS.

Scaling costs also scales net marginal revenue $\hat{R}_{ft}$, estimated from the quality optimality conditions in (20). I estimate a median of \$0.10 per daily visitor, with an interquartile range of \$0.02 to \$0.57. While website revenue data are limited, public firm data provide some reassurance about this order of magnitude. For 1,184 firm-years matched to Compustat segment data, I estimate \$0.06 from a regression of annual web-related revenue on traffic, adjusting for firm and year fixed effects.[63] Case studies offer further reassurance: Lambrecht

[60] I deflate using the PCE. \$57, which includes bonuses and perks, is slightly higher than \$45 from the BLS (2022). Across surveys, there are 43,729 responses about compensation from employed developers working on PHP, CSS/HTML, or "web frameworks" in the US, 784 in China, and 139,292 in other countries.

[61] Clustering by 116,575 responses, standard errors are 0.4% and 2.3%, respectively.

[62] An alternative would be to compare OSS mentioned in job postings to wages. I do not pursue this, as it would require adjusting for firms' choice to post based on their willingness to pay, likely needing a model of this choice. More generally, posted wages are known to provide little information (Batra et al., 2023).

[63] I use my matching results from Appendix E to obtain annual sales data for public firms in North America. I restrict to NAICS 4541, 5112, 5182, 5191, 5192, and 5415—business segments potentially linked to web traffic. After winsorizing at 5 and 95% levels, I regress annual segment revenue on web traffic and adjust for firm and year fixed effects. Clustering by firm, the standard error is \$0.03.

and Misra (2017) estimate $0.05 for ESPN from advertising alone, and Ortiz-Cordova and Jansen (2012) report about $0.002 for the smaller music website BuenaMusica.

7. Simulations

Using my estimates, I simulate counterfactuals to assess the global impact of China continuing to extend its OSS policies discussed in Section 2.2, along with the possibility of a US response. After discussing rationales for restrictions and subsidies, I report two sets of simulations. First, I hold web traffic fixed, and conservatively report how restrictions and subsidies affect OSS investment and firm costs through cost minimization alone. Second, I allow quality and demand to adjust, and report how competitive profit maximization impacts firm profit and consumer surplus. Allowing quality and demand to adjust requires stronger assumptions, but enables me to address product market competition and allows firms to more fully respond to OSS policies.

As in estimation, I conservatively continue holding each firm's total labor fixed, and focus on how firms allocate this labor.[64] I compare each counterfactual simulation to a baseline without new policies. For each simulation, I fully solve for an equilibrium by iterating on best responses until the industry converges. By starting from the observed data, this selects the equilibrium that is, loosely speaking, closest to the status quo. Computational details are in Appendix L. I quantify uncertainty by extending my bootstrap procedure. For each bootstrap sample, I re-draw demand, supply, traffic, and OSS investment estimates from Tables 3 and 4 and Appendices C and G.

7.1. Rationales for Government Intervention

In Appendix M, I illustrate the impact of government involvement in the private provision of OSS using simple model parameterizations and a series of payoff matrices; here, I focus on the underlying intuition. I do not take a firm stance on the objective function of policymakers, who may value outcomes beyond domestic firm profits and consumer surplus, such as national security. Instead, I emphasize economic inefficiencies that motivate familiar interventions.

First, positive externalities can lead to free-riding and underprovision of OSS. If the private benefits of investing in OSS are much weaker than its public benefits, OSS will be

[64] I see this restriction on firms' optimal production decisions as providing conservative estimates of OSS policy outcomes. In principle, one could unfix L_{ft} by finding a credible estimate for η in (8). In practice, however, given the small scale of OSS investment in my simulations, it is challenging to separate meaningful changes in total employment from numerical noise when solving the firm's problem. If total employment changes were larger, accounting for *aggregate* labor supply elasticities would also be necessary.

severely underprovided by the private sector. To increase the private provision of this public good, governments could use subsidies (Pigou, 1920). Subsidies naturally enter the model through cost reductions in proportion to labor ℓ_{fst} invested in $s \in \mathcal{OS}_t$.

Second, localized benefits may lead to domestic coordination problems. Consider Chinese firms that typically use $s \in \mathcal{OS}_t$ with capital stocks K_{st} primarily created by US firms (i.e., K_{cst} is high for $c =$ the US). From China's perspective, the private sector wants more investment in domestic OSS (i.e., with high K_{cst} for $c =$ China), but private incentives to improve it are weak because it is rarely used. A "big push" (Rosenstein-Rodan, 1943) could coordinate domestic investment by subsidizing investment in OSS with high domestic capital shares $K_{c(f)st}/K_{st}$, or by penalizing investment in OSS with more foreign capital.

7.2. Heightened Restrictions in China

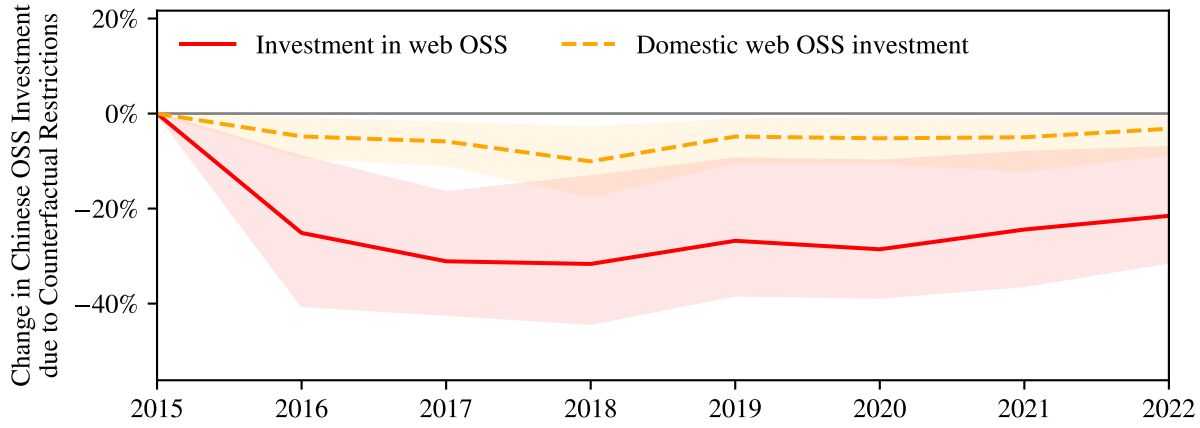
I consider penalties first by designing a counterfactual scenario where we imagine that China's 13th Five-Year Plan from 2016 instructed agencies to *restrict* investment in foreign OSS, instead of merely promoting OSS. China's restrictions on VPN use and OSS collaboration documented in Section 2.2 indicate that China has the technical and legal infrastructure to continue extending its restrictions on foreign OSS collaboration. Tightened restrictions could be economically justified if, for example, they coordinated China's private sector to use and invest in domestic OSS, resulting in domestic cost savings. They could also be justified on other grounds, such as domestic control over information or software security.

To begin, I hold web traffic fixed and conservatively focus on equilibrium changes from cost minimization alone. Among the 27 web OSS in my sample, I identify the five with lead maintainers from China.[65] Starting in $t = 2016$, I assume that for every hour Chinese firms invest in $s \in \mathcal{OS}_t$ outside of these five, they incur an additional cost of \$100 multiplied by $1 - K_{c(f)st}/K_{st}$, the foreign share of OSS capital. A \$100 disincentive for investing in fully foreign OSS is around six times the median hourly wage of web developers in China.

In Figure 7, I plot the percent change in Chinese OSS investment relative to a baseline without tightened restrictions, reporting medians across wage unobservables. The counterfactual and baseline are identical before 2016, then diverge as restrictions are introduced. Chinese investment in web OSS declines by 30% before somewhat recovering. I also plot the percent change in Chinese investment to web OSS scaled by their Chinese capital shares.

[65] I manually identify lead maintainers based on project descriptions. Those OSS with lead maintainers from China also have unusually high Chinese capital shares. They are two backend frameworks (OpenResty and ThinkPHP), one JavaScript framework (Vue), and two CSS frameworks (Ant Design and Element UI). A full list of the web OSS in my sample is in Appendix Table D1.

Figure 7: Dynamics of Counterfactual Restrictions in China



This figure reports the percent change in OSS investment for the cost minimization counterfactual with restrictions in China, relative to a baseline simulation without restrictions. Percentages are medians across 10 simulated histories of wage unobservables. The dashed line scales investment ℓ_{fst} in $s \in \mathcal{OS}_t$ by $K_{c(f)st}/K_{st}$, the domestic share of OSS capital. For the counterfactual, in $t \geq 2016$ I add $\$100 \times \ell_{fst} \times (1 - K_{c(f)st}/K_{st})$ to the costs of Chinese firms for investing in each $s \in \mathcal{OS}_t$, except for the five OSS listed in Footnote 65 with lead maintainers from China. Shaded areas are 95% confidence intervals based on 80 bootstrap samples that account for sampling error from traffic estimation in Appendix Table C1, OSS investment estimation in Appendix Table G1, demand estimation in Table 3, and supply estimation in Table 4.

Domestic OSS investment declines less but does not significantly increase by 2022. Overall, these short-run effects appear at odds with China's goal quoted in Section 1 to "accelerate the construction of domestic open source communities."

Long-Term Effects. With switching costs, the effects of policies may take many years to fully develop. I approximate a long-term horizon by fixing the sample as of 2022 and solving for an equilibrium without switching costs.[66] I compare long-term equilibrium outcomes with those from a long-term baseline without any new policies, continuing to hold web traffic fixed. I winsorize costs to mitigate the effects of a few spurious outliers from optimization issues.[67]

In column (i) of Table 5, I report effects of the tightened restrictions on the world, the US, and China. In the long run, Chinese investment into web OSS does not fully recover, declining 3% overall. Domestic OSS investment increases in some bootstrapped simulations, but the tightened restrictions do not significantly increase domestic investment by more than

[66] I set $\kappa_L = \kappa_S = 0$, allowing me to remain neutral on whether κ_L and κ_S reflect actual operational costs, are conflated with hurdle rates (e.g., Wollmann, 2018), or capture behavioral or agency frictions.

[67] Numerical error from rare optimization issues in (16)—amplified by my log wage parameterization in (8)—can create spurious cost outliers. I winsorize the ratio of each firm's average cost to its country's matched wage at the 5 and 95% levels for the US, China, and elsewhere combined. This is the same as winsorizing average costs for the US and China. Pooling other countries handles countries with few firms.

Table 5: Long Term Annual Effects of Counterfactual OSS Policies

Restrict investment in *foreign* web OSS Subsidize *domestic* web OSS investment Subsidize investment in *all* web OSS	(*i*) China	(*ii*) China	(*iii*) China & US	(*iv*) Global
Investment in web OSS	−0.2% [−0.5, −0.0]	+1.6% [+1.3, +1.9]	+9.1% [+8.0, +10.0]	+57.9% [+49.9, +63.9]
↪ US	+0.0% [−0.1, +0.1]	−0.0% [−0.1, +0.0]	+16.2% [+13.9, +17.1]	+39.8% [+35.7, +41.8]
↪ China	−3.1% [−5.1, −1.4]	+21.8% [+17.8, +25.0]	+21.4% [+19.2, +25.3]	+57.4% [+42.6, +63.6]
Domestic web OSS investment	−0.0% [−0.1, +0.1]	+5.0% [+4.5, +5.4]	+21.2% [+18.6, +22.5]	+44.8% [+40.1, +46.6]
↪ US	+0.0% [−0.0, +0.1]	+0.0% [−0.1, +0.1]	+19.6% [+17.0, +20.9]	+38.6% [+34.4, +40.2]
↪ China	−0.4% [−1.0, +0.5]	+47.2% [+41.8, +49.9]	+46.7% [+41.9, +50.1]	+65.2% [+60.1, +72.9]
Investment incentives, millions				+$6.2 [+5.1, +7.1]
↪ US			+$0.8 [+0.6, +0.9]	+$2.6 [+2.1, +2.8]
↪ China	−$0.1 [−0.1, −0.1]	+$0.1 [+0.1, +0.1]	+$0.1 [+0.1, +0.1]	+$0.5 [+0.4, +0.6]
Firm costs, per dollar of incentive	+$7.2 [+1.6, +16.4]	−$11.2 [−16.4, −7.8]	−$30.4 [−36.6, −26.9]	−$26.1 [−28.3, −25.1]
↪ US	+$4.2 [+0.3, +11.5]	−$4.5 [−8.3, −0.6]	−$25.5 [−30.4, −22.7]	−$20.9 [−22.4, −19.8]
↪ China	+$1.8 [+1.3, +2.5]	−$5.1 [−6.0, −4.4]	−$1.7 [−1.8, −1.5]	−$1.4 [−1.5, −1.3]

This table reports the equilibrium effects of cost minimization counterfactuals relative to a baseline with no new government policies. To approximate a long-term horizon, I fix the sample as of 2022 and solve for an equilibrium with no switching costs: $\kappa_L = \kappa_S = 0$. Estimates are medians over 10 sets of wage unobservables drawn from their stationary distributions. Costs are winsorized as described in Footnote 67. In column (*i*), I add $\$100 \times \ell_{fst} \times (1 - K_{c(f)st}/K_{st})$ to the costs of Chinese firms for investing in each $s \in \mathcal{OS}_t$, except for the five OSS projects listed in Footnote 65 with lead maintainers from China; in (*ii*), I instead subtract $\$100 \times \ell_{fst} \times K_{c(f)st}/K_{st}$ for each $s \in \mathcal{OS}_t$; in (*iii*), I additionally subtract the same for US firms; and in (*iv*), I instead subtract $\$100 \times \ell_{fst}$ from the costs of all firms for investing in any $s \in \mathcal{OS}_t$. Reported costs do not include these investment incentives. In brackets, 95% confidence intervals are from 80 bootstrap samples that account for sampling error from traffic estimation in Appendix Table C1, OSS investment estimation in Appendix Table G1, demand estimation in Table 3, and supply estimation in Table 4.

1% in the long run. Investment from other countries does not significantly change.

To benchmark dollar magnitudes, I estimate that if the tightened restrictions were implemented as a tax, they would generate about $100,000 per year in tax revenue from the firms in my sample.[68] For each dollar of disincentive, the tightened restrictions do $2 of damage to the Chinese web development firms in my sample. Since foreign firms would have benefited from lost Chinese investment into foreign OSS, global damages are $7 per dollar of disincentive.

Firm costs in Table 5 do not account for any government transfers. If accounted for, the effective costs incurred by Chinese firms would be slightly higher. If the tightened restrictions were instead implemented using non-tax mechanisms—such as the VPN restrictions or GitHub traffic throttling discussed in Section 2.2—these "transfers" would generate no revenue and instead contribute to deadweight loss. I do not attempt to combine costs and transfers into an overall welfare effect.[69]

The model in Section 4 does not guarantee that restrictions will be ineffective. Their impact depends on the relative sizes of the forces in the model, and particularly on the scale and localization of public benefits from OSS investment, which result in the coordination problem. For example, if the estimated operations coefficients in $\hat{\gamma}_S$ on OSS capital and its domestic share were five times larger, I find in Appendix Table N1 that restrictions are more likely to increase investment into Chinese OSS and even slightly reduce domestic costs.

Ultimately, restrictions on foreign collaboration can only do so much. Without stronger measures, like requiring firms to invest a set amount in OSS—far more extreme than China's current policies discussed in Section 2.2—firms may simply choose not to invest in OSS.

7.3. Heightened Subsidies

The support for domestic OSS in China's last two Five-Year Plans aligns more closely with subsidies. By providing state support to Gitee, policymakers in China have already shown they are willing to financially support domestic OSS investment. Funneling subsidies through an OSS platform is just one possible mechanism. Others include tax credits (e.g., New York State Assembly, 2022) or changes to accounting standards (e.g., Financial Accounting Standards Board, 2024). A detailed evaluation of the incidental costs associated with different mechanisms is beyond the scope of this paper. Instead, I simulate incentives and ignore

[68] If my results generalize beyond this sample of a single industry that covers only a fraction of OSS, absolute magnitudes such as the amount of tax revenue would be much larger.

[69] This would require making strong assumptions about the form of restrictions, the operational efficiency of OSS policy, constraints on fiscal capacity, and the opportunity cost of foregone policies in other industries.

implementation costs.

In column (*ii*) of Table 5, I assume that for every hour Chinese firms invest in $s \in \mathcal{OS}_t$, they receive \$100 multiplied by $K_{c(f)st}/K_{st}$, the domestic share of OSS capital. I find that the heightened subsidies are fairly cheap and successfully promote Chinese OSS investment, especially in domestic OSS, which increases by nearly 50% compared to the baseline. Per dollar of subsidy, costs fall by \$5 in China and by \$11 globally. Beyond cost reductions, if China derives national security benefits from more OSS built in China, the domestic advantage of heightened subsidies would be even greater.

What if there were a US response? Recent US policies including the CHIPS Act (2022) have been direct responses to Chinese industrial policy. As tensions rise, for instance, over OSS large language models,[70] the US government may prefer to maintain its OSS dominance for economic or geopolitical reasons. In column (*iii*), I retain China's subsidies and match them in the US. US subsidies are eight times larger because the firms in my sample are disproportionately based in the US. Global cost reductions amount to \$30 per dollar of subsidy. US firms benefit the most, and investment into US-built OSS increases.

Large innovation spillovers could justify even broader subsidies. While difficulties with international coordination make a global subsidy unlikely, simulating one helps benchmark potential gains from more realistic, unilateral subsidies in columns (*ii*) and (*iii*). In the final column (*iv*), I replace unilateral policies with a \$100 subsidy per hour invested by *all* firms in *any* web OSS. Though cost reductions per dollar of subsidy are slightly lower than under unilateral policies, global subsidies are seven times larger, leading to about six times the global cost reductions. Overall, I find that unilateral and US-focused policies can capture a substantial, though incomplete, share of the global benefits from subsidizing OSS.[71]

Again, the model does not guarantee that subsidies will be this effective. If the estimated private benefits of OSS investment were much stronger relative to its public benefits, the private sector would better-provide OSS, making subsidies less effective. For example, if the estimated development coefficient in $\hat{\gamma}_L$ on a firm's own OSS contributions were five times larger, I find in Appendix Table N1 that with stronger private benefits, global subsidies would reduce costs by four times less per dollar of subsidy.

The effectiveness of subsidies also depends on *which* OSS are targeted. Domestic-focused

[70] Alan Estevez, who leads the US Bureau of Industry and Security (BIS), stated in December 2023 (Center for Security and Emerging Technology, 2023) that the BIS was considering options for regulating exports of OSS large language models (LLMs). The US and China produce many of the world's leading LLMs, including Meta's Llama (github.com/meta-llama) and Alibaba's Qwen (github.com/QwenLM).

[71] I do not attempt to solve for "optimal" policies. This would require taking a strong stance on the full objective function of policymakers. Much larger OSS subsidies would also require less-credible extrapolation.

US subsidies are especially cost effective because firms primarily use and benefit from US-built OSS. If domestic and global subsidies were instead weighted by the share of websites using each OSS, I find in Appendix Table N2 that cost reductions per dollar of subsidy are 30% to 80% higher. By focusing on web development, I can identify the most-used web OSS, but effectively targeting subsidies in other industries will require more data collection, such as the OSS Census project of Nagle et al. (2022) with the Linux Foundation, or the US Census's recent inclusion of questions about use of automation technology in its Annual Business Survey (McElheran et al., 2024; Acemoglu et al., 2024).

Lastly, cost-effectiveness varies with scale. In Appendix Tables N3 and N4, I simulate per-hour subsidies that are 50% lower and higher than the baseline of $100 per hour of investment. Cost-effectiveness declines somewhat as subsidy size increases, suggesting that the OSS most affected by smaller subsidies tend to generate the greatest public benefits. In other words, private and public benefits of investing in web OSS are generally aligned across OSS, and subsidies primarily help narrow the gap between the two.

7.4. Product Market Competition

So far, I have held web traffic fixed and conservatively focused on changes from cost minimization alone. Allowing quality and demand to adjust in equilibrium requires stronger assumptions, but enables me to address product market competition, and allows firms in my model to more fully respond to OSS policies.

In theory, business stealing can either strengthen or weaken OSS policies. For example, suppose OSS subsidies increase investment, lowering firms' costs and leading to greater OSS use. If this also improves website quality, and competition over quality is intense, firms may internalize that their OSS investment causes business stealing, dampening the policy's impact. Conversely, if OSS use harms website quality but firms still prefer it to cut costs, business stealing could amplify the effects of OSS policies. I provide more examples in Appendix M, along with simple model parameterizations and a series of payoff matrices.

In Table 6, I re-simulate the policies from Table 5, but allow firms to choose quality. As with my other simulations, I keep each firm's total labor fixed and winsorize costs. Since spurious outliers from optimization issues can also affect revenue, I winsorize marginal revenue as well.[72] To prevent results from being driven by different baseline use of in-house software, which was previously fixed during cost minimization, I draw in-house operations unobservables from truncated distributions consistent with observed usage.[73] To prevent demand-side

[72] As in Footnote 67, I winsorize $\hat{R}_{ft}$ at the 5 and 95% levels for the US, China, and elsewhere combined.
[73] This ensures that OSS use in both profit maximization and cost minimization simulations are similar.

Table 6: Long Term Annual Effects with Product Market Competition

Restrict investment in *foreign* web OSS Subsidize *domestic* web OSS investment Subsidize investment in *all* web OSS	(*i*) China	(*ii*) China	(*iii*) China & US	(*iv*) Global
Investment in web OSS	−1.7% [−2.0, −1.3]	+0.8% [+0.6, +1.6]	+6.0% [+5.0, +7.3]	+85.7% [+68.5, +107.2]
↪ US	−0.0% [−0.2, +0.1]	−0.0% [−0.2, +0.0]	+15.1% [+11.5, +17.8]	+43.0% [+35.5, +51.5]
↪ China	−13.7% [−15.1, −11.4]	+9.3% [+6.7, +16.3]	+13.3% [+9.1, +17.7]	+69.7% [+51.7, +85.5]
Domestic web OSS investment	−0.4% [−0.7, −0.2]	+3.3% [+2.0, +5.0]	+18.3% [+14.1, +22.0]	+44.0% [+33.1, +53.2]
↪ US	+0.4% [+0.3, +0.8]	−0.1% [−0.4, +0.0]	+19.0% [+13.9, +22.3]	+24.5% [+18.7, +28.4]
↪ China	−7.4% [−8.9, −5.6]	+36.3% [+19.0, +41.5]	+32.3% [+23.6, +39.4]	+48.4% [+39.5, +60.6]
Investment incentives, millions				+$47.0 [+34.1, +56.3]
↪ US			+$3.0 [+2.5, +3.2]	+$10.8 [+8.8, +12.3]
↪ China	−$1.7 [−1.8, −1.5]	+$0.5 [+0.4, +0.6]	+$0.6 [+0.5, +0.7]	+$5.0 [+3.6, +6.0]
Firm costs, per dollar of incentive	+$12.4 [+7.1, +18.1]	−$14.3 [−17.1, −11.9]	−$6.5 [−10.7, −4.1]	−$9.7 [−11.0, −7.9]
↪ US	+$10.2 [+5.7, +14.0]	−$5.6 [−8.3, −1.0]	−$5.2 [−9.1, −1.6]	−$6.5 [−7.4, −5.0]
↪ China	+$2.4 [+1.3, +4.7]	−$6.3 [−7.8, −2.9]	−$0.6 [−1.5, −0.1]	−$1.2 [−1.5, −1.0]
Firm profit, per dollar of incentive	−$14.7 [−29.4, −0.8]	+$17.7 [+1.3, +56.0]	+$13.2 [−0.1, +20.9]	+$13.8 [+10.2, +20.0]
↪ US	−$13.5 [−19.2, −7.2]	+$10.7 [+3.4, +29.9]	+$9.4 [+1.9, +17.6]	+$8.7 [+6.9, +11.7]
↪ China	−$2.2 [−5.1, +3.2]	+$7.3 [+3.4, +10.1]	+$1.1 [−1.0, +5.2]	+$1.5 [+0.9, +1.8]
Consumer surplus, per capita	+$2.3 [−2.6, +11.9]	−$0.4 [−1.8, +0.7]	−$2.4 [−13.4, +4.5]	−$19.2 [−80.3, −0.1]
↪ US	+$1.3 [−7.6, +10.2]	−$0.2 [−0.9, +4.4]	−$7.8 [−35.9, +6.0]	−$80.1 [−249.6, −9.1]
↪ China	+$8.0 [−7.2, +40.3]	−$0.8 [−9.4, +1.5]	−$2.3 [−28.0, +28.8]	−$21.0 [−120.7, +10.0]

This table reports results from the same simulations as Table 5, but with firms optimizing website quality. Costs and marginal revenue are winsorized as described in Footnotes 67 and 72. Profit is the maximum from (19). Per capita consumer surplus for $\mathcal{I}_t' \subseteq \mathcal{I}_t$ is $\hat{\tau} \times \left(\sum_{i \in \mathcal{I}_t'} \sum_{m \in \mathcal{M}} Q_{imt} \log \sum_{j \in \mathcal{J}_{mt}} \exp \hat{V}_{ijt} \right) \div U_t'$ where $\hat{\tau}$ is the dollar conversion factor from Appendix I, $Q_{imt} = \sum_{j \in \mathcal{J}_{mt}} Q_{ijt}$, and U_t' is the number of internet users with types in $\mathcal{I}_t'$. In brackets, 95% confidence intervals are from 80 bootstrap samples that account for the same sources of uncertainty as those in Table 5.

results from being driven by market size assumptions (Zhang, 2024), I fix the total in-sample traffic by adjusting the utility each consumer receives from the outside option.[74]

In the top two panels, policy impacts on investment and costs remain qualitatively similar. Tightened restrictions in China prove ineffective and raise global costs, while heightened subsidies are more effective in promoting domestic OSS investment and generate large innovation spillovers that reduce costs. Quantitatively, firms in the new baseline simulation tend to invest more in OSS,[75] leading to larger investment incentives and hence differently-scaled dollar impacts. Chinese policies tend to have larger effects, while US and global subsidies have a more muted impact per dollar of subsidy.

In the bottom panel, I report changes in firm profit per dollar of incentive. Cost increases from tightened restrictions translate almost directly into profit reductions. However, firms are able to slightly compound cost reductions from OSS subsidies into greater profit increases by adjusting their profit-maximizing choices between using in-house software and OSS.

Lastly, I report changes in per capita consumer surplus. The point estimates are negative under heightened subsidies because the estimated effect of OSS on quality, $\hat{\beta}_S < 0$, is negative, and subsidies encourage firms to use more OSS. However, since the confidence interval for $\hat{\beta}_S$ includes zero, the confidence intervals for changes in consumer surplus generally include zero as well. This uncertainty may reflect a combination of weak internet user responses to OSS use and noise in my web traffic data.

The clearest conclusion from my results for OSS policy is its impact on firm costs. Again, this is an empirical finding, not guaranteed by the model. For example, if $\hat{\beta}_S$ were strongly positive, I find in Appendix Table N5 that business stealing would offset cost reductions and nearly eliminate profit gains from global subsidies. International competition also plays a role—in the same table, I find that replacing consumer "home bias" with equally strong "foreign bias" increases consumers' ability to avoid lower-quality websites, reducing consumer surplus losses, and weakens firms' ability to compound lower costs into higher profit.

For each part p with $s_{jpt} = f$, ω^S_{jft} does not affect cost minimization because the choice of in-house versus OSS is fixed. For profit maximization, I truncate the distribution of ω^S_{jft} in (18) so the firm makes the same choice of in-house versus OSS for p, given the other software characteristics in the cost minimization baseline and the net marginal revenue estimated from quality optimality conditions.

[74] This ensures that in each market, firms in my simulations compete for a fixed amount of web traffic. For estimation, I normalize $V_{i0t}(\xi_t) = 0$ for the outside option $j = 0$ in each market m. For simulations, I set $V_{i0t}(\xi_t)$ so that $Q_{i0t}(\xi_t)$ in (14) remains fixed—if the quality of all in-sample websites increases, so does the quality of the outside alternative, including out-of-sample websites.

[75] The model does not perfectly rationalize the data, so even with the adjustment in Footnote 73, firms' equilibrium choices differ when they are allowed to maximize profit.

8. Conclusion

OSS is privately provided knowledge at the core of most codebases. As OSS policies in China and elsewhere become more common, understanding their economic effects grows increasingly important. My model indicates that these effects depend empirically on how private benefits compare to the strength and localization of public benefits from OSS. In the web development industry alone, my simulations indicate that restrictions on OSS investment can greatly raise firm costs, while subsidies can generate large innovation spillovers. Interventions can be cheap because OSS investment is rare, but valuable because OSS use is widespread.

Since web development represents only a fraction of overall OSS investment, my findings suggest that the stakes may be high for OSS policy on a broader scale. I focus on web development because it is a large industry where, unusually, software use is directly observable. Effects of OSS policies are likely to differ in other high-tech industries, especially those where product quality responds strongly to the use of OSS.

Data on OSS use is crucial for better understanding its value for firms and consumers, and for efficiently targeting subsidies. As more data becomes available, my model-based empirical strategy could be adapted to study OSS more broadly. It could also serve as a starting point for richer models of privately provided knowledge—including OSS, open data, and other forms of basic innovation—that more directly incorporate the incentives of workers in competitive labor markets, upstream producers, or other motivations of governments, such as national security.

References

Acemoglu, D., G. Anderson, D. Beede, C. Buffington, E. Childress, E. Dinlersoz, L. Foster, N. Goldschlag, J. Haltiwanger, Z. Kroff, P. Restrepo, and N. Zolas (2024). Automation and the workforce: A firm-level view from the 2019 Annual Business Survey.

Ackerberg, D. A. (2009). A new use of importance sampling to reduce computational burden in simulation estimation. *Quantitative Marketing and Economics 7*(4), 343–376.

Ackerberg, D. A. (2023). Timing assumptions and efficiency: Empirical evidence in a production function context. *Journal of Industrial Economics 71*(3), 644–674.

Ackerberg, D. A., K. Caves, and G. Frazer (2015). Identification properties of recent production function estimators. *Econometrica 83*(6), 2411–2451.

Ackerberg, D. A., G. Frazer, K. il Kim, Y. Luo, and Y. Su (2023). Under-identification of structural models based on timing and information set assumptions. *Journal of Econometrics 237*(1), 105463.

Ackermann, K. and S. Greenstein (2024). The state of web server software: Determinants of worldwide dispersion in use.

Angelopoulos, A. N., S. Bates, C. Fannjiang, M. I. Jordan, and T. Zrnic (2023). Prediction-powered inference. *Science 382*(6671), 669–674.

Arellano, M. and O. Bover (1995). Another look at the instrumental variable estimation of error-components models. *Journal of Econometrics 68*(1), 29–51.

Associated Press (2021). Encrypted messaging app Signal blocked in China. NBC News.

Azar, J. A., S. T. Berry, and I. Marinescu (2022). Estimating labor market power.

Bai, J., P. J. Barwick, S. Cao, and S. Li (2022). Quid pro quo, knowledge spillover, and industrial quality upgrading: Evidence from the Chinese auto industry.

Bakshy, E., L. Dworkin, B. Karrer, K. Kashin, B. Letham, A. Murthy, and S. Singh (2018). AE: A domain-agnostic platform for adaptive experimentation. In *Conference on Neural Information Processing Systems*, pp. 1–8.

Balandat, M., B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy (2019). BoTorch: Programmable Bayesian optimization in PyTorch.

Barwick, P. J., S. Cao, and S. Li (2021). Local protectionism, market structure, and social welfare: China's automobile market. *American Economic Journal: Economic Policy 13*(4), 112–151.

Barwick, P. J., M. Kalouptsidi, and N. B. Zahur (2019). China's industrial policy: An empirical evaluation.

Batra, H., A. Michaud, and S. Mongey (2023). Online job posts contain very little wage information.

Berry, S., J. Levinsohn, and A. Pakes (1995). Automobile prices in market equilibrium. *Econometrica 63*(4), 841–890.

Berry, S., J. Levinsohn, and A. Pakes (2004). Differentiated products demand systems from a combination of micro and macro data: The new car market. *Journal of Political Economy 112*(1), 68–105.

Blind, K., S. Pätsch, S. Muto, M. Böhm, T. Schubert, P. Grzegorzewska, and A. Katz (2021). *The impact of open source software and hardware on technological independence, competitiveness and innovation in the EU economy*. Publications Office of the European Union. European Commission, Directorate-General for Communications Networks, Content and Technology.

Blondel, M., Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert (2022). Efficient and modular implicit differentiation. *Advances in neural information processing systems 35*, 5230–5242.

Bloom, N., M. Schankerman, and J. Van Reenen (2013). Identifying technology spillovers and product market rivalry. *Econometrica 81*(4), 1347–1393.

Bloom, N., J. Van Reenen, and H. Williams (2019). A toolkit of policies to promote innovation. *Journal of Economic Perspectives 33*(3), 163–184.

Blundell, R. and S. Bond (1998). Initial conditions and moment restrictions in dynamic panel data models. *Journal of Econometrics 87*(1), 115–143.

Blundell, R. and S. Bond (2000). GMM estimation with persistent panel data: An application to production functions. *Econometric Reviews 19*(3), 321–340.

Boehm, B. W. (1984). Software engineering economics. *IEEE transactions on Software Engineering SE-10*(1), 4–21.

Boehm, B. W., C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. J. Reifer, and B. Steece (2009). *Software cost estimation with COCOMO II*. Prentice Hall Press.

Boysel, S., M. Hoffmann, and F. Nagle (2024). Labor competition and open innovation.

Bradbury, J., R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang (2018). JAX: Composable transformations of Python+NumPy programs. http://github.com/google/jax.

Brynjolfsson, E., A. Collis, and F. Eggers (2019). Using massive online choice experiments to measure changes in well-being. *Proceedings of the National Academy of Sciences 116*(15), 7250–7255.

Brynjolfsson, E., A. Collis, A. Liaqat, D. Kutzman, H. Garro, D. Deisenroth, N. Wernerfelt, and J. J. Lee

(2023). The digital welfare of nations: New measures of welfare gains and inequality.

Bureau of Labor Statistics (2022). May 2022 national occupational employment and wage estimates. https://www.bls.gov/oes/2022/may/oes_nat.htm.

Center for Security and Emerging Technology (2023). Fireside chat with Under Secretary of Commerce Alan Estevez. https://cset.georgetown.edu/event/cset-to-host-under-secretary-of-commerce-alan-estevez.

Chandel, S., Z. Jingji, Y. Yunnan, S. Jingyao, and Z. Zhipeng (2019). The Golden Shield Project of China: A decade later—An in-depth study of the Great Firewall. In *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 111–119.

Chen, T. and C. Guestrin (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794.

Chevalier, J. and A. Goolsbee (2003). Measuring prices and price competition online: Amazon.com and BarnesandNoble.com. *Quantitative Marketing and Economics 1*, 203–222.

Coe, D. T. and E. Helpman (1995). International R&D spillovers. *European Economic Review 39*(5), 859–887.

Conlon, C. and J. Gortmaker (2020). Best practices for differentiated products demand estimation with PyBLP. *RAND Journal of Economics 51*(4), 1108–1161.

Conlon, C. and J. Gortmaker (2025). Incorporating micro data into differentiated products demand estimation with PyBLP. *Journal of Econometrics*.

Conneau, A., K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov (2019). Unsupervised cross-lingual representation learning at scale.

Conti, A., V. Gupta, J. Guzman, and M. P. Roche (2023). Incentivizing innovation in open source: Evidence from the GitHub Sponsors program.

Conti, A., C. Peukert, and M. P. Roche (2021). Beefing it up for your investor? Open sourcing and startup funding: Evidence from GitHub.

Daulton, S., S. Ament, D. Eriksson, M. Balandat, and E. Bakshy (2023). Unexpected improvements to expected improvement for Bayesian optimization. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pp. 20577–20612.

Department of Defence (2022). Software development and open source software. https://dodcio.defense.gov/Portals/0/Documents/Library/SoftwareDev-OpenSource.pdf.

Dushnitsky, G. and B. K. Stroube (2021). Low-code entrepreneurship: Shopify and the alternative path to growth. *Journal of Business Venturing Insights 16*, e00251.

Economides, N. and E. Katsamakas (2006). Two-sided competition of proprietary vs. open source technology platforms and the implications for the software industry. *Management Science 52*(7), 1057–1071.

Filasto, A. and J. Appelbaum (2012). OONI: Open observatory of network interference. In *2nd USENIX Workshop on Free and Open Communications on the Internet*. USENIX Association.

Financial Accounting Standards Board (2024). Accounting for and disclosure of software costs: Project update. https://fasb.org/projects/current-projects/accounting-for-and-disclosure-of-software-costs-401660.

Flynn, R., B. Glennon, R. Murciano-Goroff, and J. Xiao (2024). Building a wall around science: The effect of US-China tensions on international scientific research.

Frazier, P. I., W. B. Powell, and S. Dayanik (2008). A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization 47*(5), 2410–2439.

Gabaix, X. and R. Ibragimov (2011). Rank − 1/2: A simple way to improve the OLS estimation of tail exponents. *Journal of Business & Economic Statistics 29*(1), 24–39.

Gablonsky, J. M. and C. T. Kelley (2001). A locally-biased form of the DIRECT algorithm. *Journal of Global Optimization 21*, 27–37.

Gitee (2024). About us. https://gitee.com/about-us.

GitHub (2023a). Innovation graph. https://innovationgraph.github.com.

GitHub (2023b). Octoverse: The state of open source and rise of AI in 2023. https://github.blog/2023-11-08-the-state-of-open-source-and-ai.

GitHub (2024). About GitHub. https://github.com/about.

Gonzalez-Barahona, J. M., G. Robles, R. Andradas-Izquierdo, and R. A. Ghosh (2008). Geographic origin of libre software developers. *Information Economics and Policy 20*(4), 356–363.

Gortmaker, J., J. Jeffers, and M. Lee (2023). Labor reactions to credit deterioration: Evidence from LinkedIn activity.

Gousios, G. and D. Spinellis (2012). GHTorrent: GitHub's data from a firehose. In *9th IEEE Working Conference on Mining Software Repositories*, pp. 12–21.

Greenstein, S. and F. Nagle (2014). Digital dark matter and the economic contribution of Apache. *Research Policy 43*(4), 623–631.

Grigorik, I. (2024). GH Archive. https://www.gharchive.org.

Griliches, Z. (1979). Issues in assessing the contribution of research and development to productivity growth. *Bell Journal of Economics 10*(1), 92–116.

Griliches, Z. (1992). The search for R&D spillovers. *Scandinavian Journal of Economics 94*, S29–S47.

Grossman, G. M. and E. Helpman (1991). Trade, knowledge spillovers, and growth. *European Economic Review 35*(2-3), 517–526.

Haklay, M. and P. Weber (2008). OpenStreetMap: User-generated street maps. *IEEE Pervasive Computing 7*(4), 12–18.

Hamilton, A. (1791). Report on manufactures. In H. C. Syrett (Ed.), *The papers of Alexander Hamilton*, Volume 10, pp. 302–304. Columbia University Press.

Heckman, J. (1978). Simple statistical models for discrete panel data developed and applied to test the hypothesis of true state dependence against the hypothesis of spurious state dependence. *Annales de l'inséé 30/31*, 227–269.

Heckman, J. (1981). Statistical models for discrete panel data. In C. F. Manski and D. McFadden (Eds.), *Structural analysis of discrete data with econometric applications*, pp. 114–178. MIT Press.

Hoffmann, M., S. Boysel, F. Nagle, S. Peng, and K. Xu (2024). Generative AI and distributed work: Evidence from open source software.

Hoffmann, M., F. Nagle, and Y. Zhou (2024). The value of open source software.

Jaffe, A. B., M. Trajtenberg, and R. Henderson (1993). Geographic localization of knowledge spillovers as evidenced by patent citations. *Quarterly Journal of Economics 108*(3), 577–598.

Jiang, B. (2024). Alibaba says its Tongyi Qianwen AI models are used by over 90,000 corporate clients in China. South China Morning Post.

Johnson, G. A., S. K. Shriver, and S. G. Goldberg (2023). Privacy and market concentration: Intended and unintended consequences of the GDPR. *Management Science 69*(10), 5695–5721.

Jones, D. R., C. D. Perttunen, and B. E. Stuckman (1993). Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications 79*, 157–181.

Joulin, A., E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov (2016). FastText.zip: Compressing text classification models.

Juhász, R., N. Lane, and D. Rodrik (2023). The new economics of industrial policy. *Annual Review of*

Economics 16, 213–242.

Kalliamvakou, E., G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian (2016). An in-depth study of the promises and perils of mining GitHub. *Empirical Software Engineering 21*, 2035–2071.

Kalouptsidi, M. (2018). Detection and impact of industrial subsidies: The case of Chinese shipbuilding. *Review of Economic Studies 85*(2), 1111–1158.

Kumar, V., B. R. Gordon, and K. Srinivasan (2011). Competitive strategy for open source software. *Marketing Science 30*(6), 1066–1078.

Lakhani, K. R. and R. G. Wolf (2005). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. In *Perspectives on free and open source software*. MIT Press.

Lambrecht, A. and K. Misra (2017). Fee or free: When should firms charge for online content? *Management Science 63*(4), 1150–1165.

Lane, N. (2020). The new empirics of industrial policy. *Journal of Industry, Competition and Trade 20*(2), 209–234.

Lerner, J. and M. Schankerman (2010). Assessing government policies toward software. In *The comingled code: Open source and economic development*, pp. 206–208. MIT Press.

Lerner, J. and J. Tirole (2002). Some simple economics of open source. *Journal of Industrial Economics 50*(2), 197–234.

Lerner, J. and J. Tirole (2005). The economics of technology sharing: Open source and beyond. *Journal of Economic Perspectives 19*(2), 99–120.

Levinsohn, J. and A. Petrin (2003). Estimating production functions using inputs to control for unobservables. *Review of Economic Studies 70*(2), 317–341.

Lewis, M. P., G. F. Simons, and C. D. Fennig (2016). *Ethnologue: Languages of the world* (19 ed.). SIL International.

Li, X., Y. Zhang, C. Osborne, M. Zhou, Z. Jin, and H. Liu (2024). Systematic literature review of commercial participation in open source software. ACM Transactions on Software Engineering and Methodology.

Llanes, G. and R. de Elejalde (2013). Industry equilibrium with open-source and proprietary firms. *International Journal of Industrial Organization 31*(1), 36–49.

Lostri, E., G. Wood, and M. Jain (2023). Government open source software policies. Center for Strategic and International Studies.

McElheran, K., J. F. Li, E. Brynjolfsson, Z. Kroff, E. Dinlersoz, L. Foster, and N. Zolas (2024). AI adoption in America: Who, what, and where. *Journal of Economics & Management Strategy 33*(2), 375–415.

McFadden, D. (1989). A method of simulated moments for estimation of discrete response models without numerical integration. *Econometrica 57*(5), 995–1026.

Ministry of Industry and Information Technology (2017). Notice of the Ministry of Industry and Information Technology on cleaning up and regulating the internet network access service market. https://www.miit.gov.cn/jgsj/xgj/gzdt/art/2020/art_6dd0e345bc3947b2a7c88509c4951cd0.html. Translated by Google.

Ministry of Industry and Information Technology (2020). 2020 open source hosting platform project results announcement. https://www.cec-ec.com.cn/cms/channel/1xmgg3/648.htm. Translated by Google.

Morrow, W. R. and S. Skerlos (2010). On the existence of Bertrand-Nash equilibrium prices under logit demand.

Murciano-Goroff, R., R. Zhuo, and S. Greenstein (2021). Hidden software and veiled value creation: Illustrations from server software usage. *Research Policy 50*(9), 104333.

Murciano-Goroff, R., R. Zhuo, and S. Greenstein (2024). Navigating software vulnerabilities: Eighteen years of evidence from medium and large US organizations.

Musseau, J., J. S. Meyers, G. P. Sieniawski, C. A. Thompson, and D. German (2022). Is open source eating the world's software? Measuring the proportion of open source in proprietary software using Java binaries. In *Proceedings of the 19th International Conference on Mining Software Repositories*, pp. 561–565.

Nagle, F. (2018). Learning by contributing: Gaining competitive advantage through contribution to crowdsourced public goods. *Organization Science 29*(4), 569–587.

Nagle, F. (2019a). Government technology policy, social value, and national competitiveness.

Nagle, F. (2019b). Open source software and firm productivity. *Management Science 65*(3), 1191–1215.

Nagle, F., J. Dana, J. Hoffman, S. Randazzo, and Y. Zhou (2022). Census II of free and open source software - Application libraries. Linux Foundation, Harvard Laboratory for Innovation Science, and the Open Source Security Foundation.

Nagle, F., D. A. Wheeler, H. Lifshitz-Assaf, H. Ham, and J. Hoffman (2020). Report on the 2020 FOSS contributor survey. Linux Foundation's Core Infrastructure Initiative and the Laboratory for Innovation Science at Harvard.

Naidu, S., E. A. Posner, and G. Weyl (2018). Antitrust remedies for labor market power. *Harvard Law Review 132*(2), 536–601.

National Science Foundation (2023). NSF invests over $26 million in open-source projects. https://new.nsf.gov/tip/updates/nsf-invests-over-26m-open-source-projects.

Nesbitt, A. (2024). Ecosyste.ms: Roadmap. https://ecosyste.ms.

New York State Assembly (2021–2022). Bill A94. https://www.nysenate.gov/legislation/bills/2021/A94.

Nocedal, J. and S. J. Wright (2006). *Numerical optimization* (2 ed.). Springer.

Olley, G. S. and A. Pakes (1996). The dynamics of productivity in the telecommunications equipment industry. *Econometrica 64*(6), 1263–1297.

Olson, M. (1965). *The logic of collective action: Public goods and the theory of groups*. Harvard University Press.

Ortiz-Cordova, A. and B. J. Jansen (2012). Classifying web search queries to identify high revenue generating customers. *Journal of the American Society for Information Science and Technology 63*(7), 1426–1441.

Pakes, A. and D. Pollard (1989). Simulation and the asymptotics of optimization estimators. *Econometrica 57*(5), 1027–1057.

Pigou, A. C. (1920). *The economics of welfare*. Macmillan.

Pochat, V. L., T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen (2018). Tranco: A research-oriented top sites ranking hardened against manipulation.

Protalinski, E. (2013). The Chinese government appears to be blocking GitHub via DNS. The Next Web.

Robbins, C., G. Korkmaz, L. Guci, J. B. S. Calderón, and B. Kramer (2021). A first look at open-source software investment in the United States and in other countries, 2009-2019.

Rodrik, D. (2012). Why we learn nothing from regressing economic growth on policies. *Seoul Journal of Economics 25*(2), 137–151.

Rosenstein-Rodan, P. N. (1943). Problems of industrialisation of eastern and south-eastern Europe. *Economic Journal 53*(210-211), 202–211.

Roussille, N. and B. Scuderi (2024). Bidding for talent: A test of conduct in a high-wage labor market.

Schmeiser, S. (2015). The size distribution of websites. *Economics Letters 128*, 62–68.

Shiller, B., J. Waldfogel, and J. Ryan (2018). The effect of ad blocking on website traffic and quality. *RAND Journal of Economics 49*(1), 43–63.

Sokolova, A. and T. Sorensen (2021). Monopsony in labor markets: A meta-analysis. *ILR Review 74*(1),

27–55.

Sonatype (2020). State of the software supply chain. https://www.sonatype.com/resources/white-paper-state-of-the-software-supply-chain-2020.

Stack Overflow (2019–2023). Annual developer survey. https://survey.stackoverflow.co.

Sundara Raman, R., P. Shenoy, K. Kohls, and R. Ensafi (2020). Censored Planet: An internet-wide, longitudinal censorship observatory. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 49–66.

Synopsys (2024). Open source security and risk analysis report. https://www.synopsys.com/blogs/software-security/open-source-trends-ossra-report.html.

Tambe, P., L. Hitt, D. Rock, and E. Brynjolfsson (2020). Digital capital and superstar firms.

The Economist (2024). Why are VPNs getting slower in China? https://www.economist.com/china/2024/08/22/why-are-vpns-getting-slower-in-china.

The Guardian (2023). Chinese programmer ordered to pay 1m yuan for using virtual private network. https://www.theguardian.com/world/2023/oct/09/chinese-programmer-ordered-to-pay-1m-yuan-for-using-virtual-private-network.

Tiebout, C. M. (1956). A pure theory of local expenditures. *Journal of Political Economy 64*(5), 416–424.

Trajtenberg, M. (1990). A penny for your quotes: Patent citations and the value of innovations. *RAND Journal of Economics 21*(1), 172–187.

US Congress (2022). H.R. 7178 - 117th Congress (2021–2022): CHIPS and Science Act. https://www.congress.gov/bill/117th-congress/house-bill/4346.

Varadhan, R. and C. Roland (2008). Simple and globally convergent methods for accelerating the convergence of any EM algorithm. *Scandinavian Journal of Statistics 35*(2), 335–353.

Von Krogh, G. and E. Von Hippel (2006). The promise of research on open source software. *Management Science 52*(7), 975–983.

Wachs, J., M. Nitecki, W. Schueller, and A. Polleres (2022). The geography of open source software: Evidence from GitHub. *Technological Forecasting and Social Change 176*, 121478.

White House (2022). Readout of White House meeting on software security. https://www.whitehouse.gov/briefing-room/statements-releases/2022/01/13/readout-of-white-house-meeting-on-software-security.

White House (2024). Fact sheet: Biden-Harris administration releases summary report of 2023 RFI on open source software security initiative. https://www.whitehouse.gov/oncd/briefing-room/2024/08/09/fact-sheet-biden-harris-administration-releases-end-of-year-report-on-open-source-software-security-initiative-2.

Wollmann, T. G. (2018). Trucks without bailouts: Equilibrium product characteristics for commercial vehicles. *American Economic Review 108*(6), 1364–1406.

Wright, N. L., F. Nagle, and S. Greenstein (2023). Open source software and global entrepreneurship. *Research Policy 52*(9), 104846.

Wright, N. L., F. Nagle, and S. Greenstein (2024). Contributing to growth? The role of open source software for global startups.

Wu, M., J. Sippe, D. Sivakumar, J. Burg, P. Anderson, X. Wang, K. Bock, A. Houmansadr, D. Levin, and E. Wustrow (2023). How the Great Firewall of China detects and blocks fully encrypted traffic. In *32nd USENIX Security Symposium*, pp. 2653–2670.

Yang, Z. (2022). How censoring China's open-source coders might backfire. MIT Technology Review.

Zhang, L. (2024). Identification and estimation of market size in discrete choice demand models.

Zrnic, T. and E. J. Candès (2024). Cross-prediction-powered inference. *Proceedings of the National Academy*

of Sciences 121(15), e2322083121.

Appendices

A. GitHub and Gitee Data

In this appendix, I explain how I create my data on public GitHub and Gitee activity, used for aggregate statistics in Section 2 and the main dataset in Section 3. I describe the types of OSS activity and how I collect them from GitHub and Gitee, focusing on web frameworks. I also detail how I handle bot activity and measure user locations.

Activity Types. I collect data on six types of public activity on GitHub and Gitee, representing the main ways OSS collaboration occurs. Users engage with repositories (repos), also known as projects.

A "commit" is a change or update to a codebase, involving the addition, modification, or deletion of code. A "pull request" (PR) is a collection of commits and a proposal to merge these changes from one repo—usually temporary—into another. A "review" evaluates code changes in a PR. An "issue" is a bug report, feature request, question, or task needing attention. A "comment" follows up on a PR or issue. A "push" uploads one or more commits from a local machine to a remote repo on GitHub or Gitee.[76]

Recently, GitHub's "discussion" feature has begun to replace some of the functions previously handled by issues. Introduced near the end of my sample in late 2020, it has been adopted slowly, so I do not collect data on discussions in this article. However, future work may find it important to include them.

GHTorrent and GH Archive. My primary GitHub data come from GHTorrent (Gousios and Spinellis, 2012) and GH Archive (Grigorik, 2024). Both datasets were built by querying GitHub's application programming interface (API) over several years. GH Archive only archives data from GitHub's public timeline, while GHTorrent, when active, also captured additional information, including public user activity pages. The main downsides of GHTorrent are a data gap from July 2019 to January 2020 and its discontinuation in 2021. In contrast, GH Archive has continuously archived GitHub data since 2011 with few gaps. I combine data from both sources to create a unified dataset of GitHub activity.

From GHTorrent's MongoDB files, I create separate datasets for users, organizations, commits, issues, PRs, reviews, and comments. From GH Archive's daily files, I create similar datasets for commits, issues, PRs, reviews, comments, and pushes. Unlike GHTorrent, GH Archive lacks detailed user or organization data but includes pushes. For commits,

[76] Since pushes are just bundles of commits, I only use them when comparing with GitHub's Innovation Graph data, discussed below.

issues, PRs, reviews, and comments, I keep the most recently archived version from either GHTorrent or GH Archive. I de-duplicate using hashes for commits and GitHub IDs for other activities.

Each activity observation includes a timestamp, the user who authored it, the repo where it occurred, and other metadata. The only other metadata I use in this article is the text content. Commits have messages; issues and PRs have titles and text bodies; reviews and comments only have text bodies. I use this text data to create features for prediction in Appendix F.

Gitee. There is no comprehensive dataset like GHTorrent or GH Archive for Gitee activity, so I construct one by repeatedly querying Gitee's public API. Unlike the GitHub API, the Gitee API does not seem to strongly rate-limit requests, allowing me to build a Gitee dataset with just one account.[77]

Since Gitee's API is similar to GitHub's, I can create close analogues to my GitHub datasets. Specifically, I create separate datasets for users, commits, issues, PRs, reviews, and comments. I de-duplicate using hashes for commits and Gitee IDs for other activities.

Activity for Web Frameworks. When combined, GHTorrent and GH Archive provide good coverage of public GitHub activity since at least 2011, though coverage of older activity is less comprehensive. This is not an issue for the aggregate statistics from 2018 to 2022 in Section 2, but it does affect some web frameworks focused on in the rest of the article. Some frameworks have activity going back many years.[78] Capturing this early activity is crucial for constructing stocks of intangible capital.

Compared to the millions of projects on GitHub, there are only a few web frameworks that I focus on starting in Section 3. I list these frameworks, how I detect them on websites, and how I manually match them to GitHub repos in Appendix D. No framework with enough usage to be in my final sample has an active Gitee repo.

Although these frameworks' repos are large and very active, there are few enough that I can query GitHub's API directly instead of relying on GHTorrent and GH Archive. Unlike Gitee, GitHub has tight API rate limits. I query GitHub's API for complete histories of commits, issues, PRs, reviews, and comments on these web frameworks' repos, and I also gather up-to-date information on the authors' user pages.

[77] This process involves running 40 parallel queries for nearly a month.

[78] For example, the first commit to Ruby on Rails, a popular backend framework, was authored in 2004.

Bot Detection. Some activity on GitHub and Gitee is authored by bot accounts rather than human users. Some of this bot activity aligns with how this article defines OSS investment—the joint choice of a firm and its workers to improve a codebase—while some is just noise. For example, actual OSS investment includes old commits automatically moved from an older repo to a primary one. Noise includes a pet project that automatically commits every second to update a clock. While only a small fraction of GitHub and Gitee accounts generate automated activity, it can still influence aggregate statistics.

The web framework repos I focus on maintain high-quality standards for activity. I find that the maintainers of these frameworks self-police bot activity, so there is little noise in my primary sample. However, some users exhibit unusually high activity in short periods. I manually review these cases and generally find automated movements of code or issues, like the example above. I reclassify timestamps and authors as needed and discard clearly noisy activity, such as code coverage reports.

Manually checking unusually high activity rates across all GitHub and Gitee repos is infeasible, so I use a different approach for the aggregate statistics in Section 2. I focus on monthly active contributors because this measure is largely unaffected by bot activity—a single bot might artificially inflate global commit counts by 10%, but it still counts as just one among millions of contributors.

The one statistic significantly impacted by how I handle bots is the numerator in my country and quarter-level measure of pushes per monthly active user, which I use to re-scale data from GitHub's Innovation Graph, discussed below. When calculating pushes per country-quarter, I follow GitHub's method for the Innovation Graph, discarding (*i*) any user-quarter with more pushes than a reasonable human threshold,[79] and (*ii*) users flagged as bots in GHTorrent's archived API queries.

Filling in Missing Users. Since GH Archive does not collect user page information, geocoding a significant portion of GitHub activity after GHTorrent's discontinuation in 2021 is challenging. For instance, if the author of a 2022 commit never had their user page archived by GHTorrent, it is difficult to assign a country to that commit based on the author's self-reported location. To address this, I query GitHub's API for the public user pages of roughly 16 million GitHub users in my combined GHTorrent-GH Archive dataset through mid-2023 whose pages were not archived by GHTorrent.[80]

[79] I use GitHub's cutoff but cannot report the exact number to prevent "gaming" of the Innovation Graph.
[80] Since GitHub's API is rate-limited, this required several volunteers and weeks of repeated queries.

Geocoding User Locations. Many GitHub users list their location on their public profiles, especially those who are very active, whose profiles are usually well-detailed. However, these locations are in plain text and need to be geocoded to systematically assign activity to different countries.

For most users, I rely on GHTorrent, which geocodes locations using OpenStreetMap (Haklay and Weber, 2008) for all archived GitHub profiles in its MySQL dataset. For missing profiles that I retrieve from GitHub's API, I use the GHTorrent-geocoded location if the text matches an archived user's location. Otherwise, I query OpenStreetMap myself.

While highly active GitHub users often self-report their locations, many leave the location field blank, and some locations cannot be geocoded by OpenStreetMap. For the aggregate statistics in Section 2 on active user counts by country, I assume no systematic bias in self-reporting and scale the country shares of contributors who self-report locations by the total number of contributors. For web framework contributors in my primary sample, I use a more detailed classification process described in Appendix F.

Innovation Graph Comparisons. A concern with using self-reported locations is that self-reporting rates may systematically differ across countries and user characteristics. For example, if active contributors in China are less likely to self-report their location than those in the US, my estimate of GitHub users in China in Section 2 would be biased downward.

To address this, I incorporate data from GitHub's Innovation Graph (GitHub, 2023a). Released in 2023, the Innovation Graph reports aggregated GitHub statistics at the country-quarter level, dating back to 2020. Its main advantages over my combined GHTorrent and GH Archive data are ease of use and, crucially, the geocoding of users based on Internet Protocol (IP) addresses rather than self-reported locations.

The Innovation Graph does not report monthly active contributors, my preferred bot-resistant activity measure, but it does report the number of pushes per country-quarter, filtered using the fixed push cutoff mentioned earlier. Using self-reported locations and the same cutoff, I compute the average number of pushes per monthly active contributor at the country-quarter level and use these ratios to convert GitHub's IP-based push data into an IP-based measure of monthly active contributors.

In Figure 2, the close alignment of the self-reported and IP-based lines for the US and China before 2021 suggests self-reporting does not differ systematically between the two countries. As discussed in Section 2, the 2021 divergence of the China line likely reflects increased use of virtual private networks (VPNs).

B. Website Data

In this appendix, I describe my data on website homepages from Common Crawl, a nonprofit that regularly archives much of the internet. I ensure proper handling of redirects to improve coverage rates.

Archives. My primary sample begins in 2014 when Common Crawl started archiving the web on a roughly monthly basis. For each monthly crawl, I download archived homepages for *all* websites that *ever* appear on Alexa's top 10,000 list at any point during my sample period. I describe Alexa in Appendix C. I discard archives without HTTP response codes of 200 or text/HTML content types. If multiple homepages exist for the same website in a given crawl, I use the earliest one archived.

Redirects. For most websites, requesting their top-level domain's (TLD) archive from Common Crawl is sufficient.[81] However, some TLDs redirect to another, non-top-level address, which is the actual homepage. I follow these redirects to improve coverage.

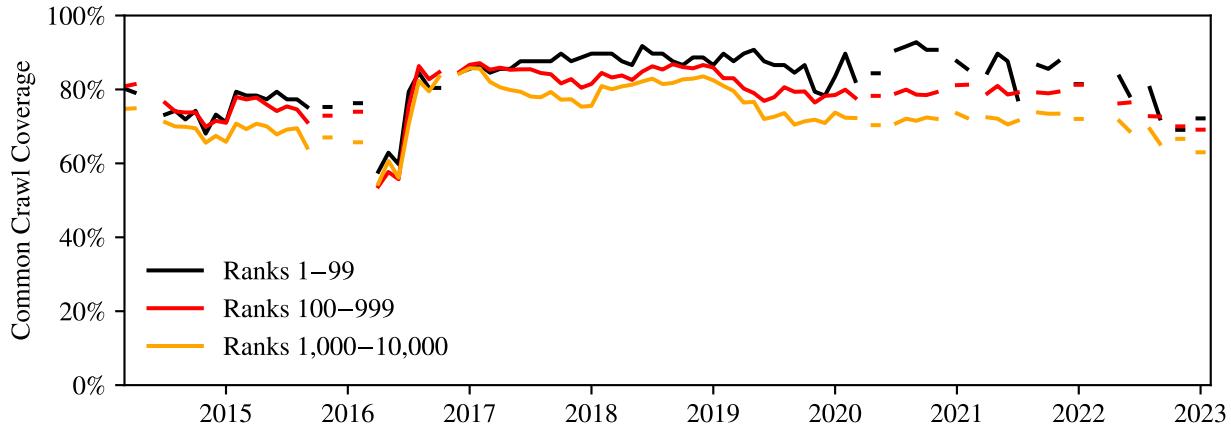
I handle two types of redirects: server-side and client-side. Server-side redirects are indicated by HTTP response codes starting with 3. When a TLD archive has such a code, I identify the redirected URL and use its archive if available in the same crawl. Client-side redirects occur within the user's browser when loading the website. I search the archived HTML for meta refreshes and JavaScript-based redirects, identify the target URL, and use its archive when available.

Coverage. I report monthly Common Crawl coverage rates in Appendix Figure B1, separately for websites ranked 1–99 in that month, 100–999, and 1,000–10,000. Coverage averages around 80%, and is slightly higher for higher-traffic websites.

When Common Crawl fails to archive a homepage, I use the previous month's information. Ultimately, I work with yearly data, using the earliest homepage available for each year. This approach mitigates issues from the few months without crawls. Websites change little month-to-month, so when in the year to take a snapshot matters little. From my final sample, I discard 8% of websites without a valid archived homepage.

[81] For example, downloading the homepage for Google.com is straightforward by using the first record in a crawl with that TLD.

Appendix Figure B1: Common Crawl Coverage Rates



This figure reports the monthly percentage of Alexa top 10,000 websites for which I successfully obtain an archived homepage from Common Crawl, broken down by Alexa rank group. Gaps in the lines represent months when Common Crawl did not perform any crawls.

C. Traffic Data

In this appendix, I describe my data on website traffic from Alexa, which I use to identify the world's most visited websites and, following Chevalier and Goolsbee (2003), to measure their traffic over time. Acquired by Amazon in 1999,[82] Alexa Internet offered various web traffic analysis services until it was discontinued in 2022.

Global Traffic Ranks. Alexa is best known for its publicly available global traffic ranks. It maintained a running list of the top 1 million websites globally, ranked by its estimate of daily traffic. These traffic estimates were primarily derived from Alexa's toolbar extension, with additional data from other undisclosed extensions and traffic meters used by website operators (Pochat et al., 2018; Shiller et al., 2018). Archived data from toplists.net.in.tum.de covers 2009 to early 2023, but since my Common Crawl data starts in 2014, I only use Alexa ranks from 2014 onward.

I follow Pochat et al. (2018) to aggregate daily traffic ranks into monthly ranks. For each daily archive of Alexa's top 1 million list, I assign Dowdall points based on the inverse of ranks: rank 1 receives 1 point, rank 2 gets 0.5 points, and so on. I then sum the points for each website within a month and re-rank them based on their monthly Dowdall points. This process produces a more stable monthly ranking of websites.

[82] Alexa Internet is unrelated to Amazon's virtual assistant, also called Alexa.

Appendix Table C1: Traffic Estimates

	log(Traffic rank − 1/2)
−log(Visitor-days)	0.75
	(0.01)
R^2	0.64
Website-months	210,592
↪ Websites	15,637
↪ Months	24

This table reports results from the regression estimating a power law relationship between monthly traffic ranks and traffic in visitor-days. The regression includes a fixed effect for each month. The standard error (in parentheses) is $(2/n)^{1/2}\hat{\zeta}$ from Gabaix and Ibragimov (2011), where n is conservatively taken as the number of websites, not website-months.

Converting Ranks into Traffic. My estimation approach in Section 5 requires traffic estimates for each website. While global Alexa traffic ranks were public, the underlying traffic data was not, and historical access is now limited. To convert ranks into traffic estimates, I follow Chevalier and Goolsbee (2003) and fit a power law relationship on a shorter panel from 2018 to 2020 (used by Johnson et al., 2023), which provides monthly traffic corresponding to those years' ranks. Specifically, the data gives the share of the global internet population visiting each site daily, which I scale using global internet population data from the International Telecommunication Union.

Like many size distributions, web traffic follows a power law (Schmeiser, 2015). Denote the traffic to website j by Q_{jt}, measured in daily visits. Here, t denotes months. Also denote the website's global traffic rank $\#_{jt}$. I restrict the sample to $1 \leq \#_{jt} \leq 10{,}000$. In Appendix Table C1, I report results from the following regression, where I subtract one-half to accurately estimate the Pareto coefficient (Gabaix and Ibragimov, 2011):

$$\log(\#_{jt} - 1/2) = \bar{\omega}_t^{\#} - \zeta \log Q_{jt} + \varepsilon_{jt}^{\#}. \tag{C1}$$

Monthly fixed effects $\bar{\omega}_t^{\#}$ account for time-varying changes in the global number of internet users and the share of daily visits to the top 10,000 websites in my sample. The total number of observations is slightly less than $24 \times 10{,}000$ because some websites lacked archived traffic

data; I found no clear pattern in the missing websites. I estimate a Pareto exponent of $\hat{\zeta} = 0.75$, which is somewhat lower than Schmeiser's (2015) estimate of around one.[83]

Only global traffic ranks $\#_{jt}$ are consistently available from 2014 to 2022. I estimate monthly traffic using $\hat{Q}_{jt} \propto \exp(-\log(\#_{jt} - 1/2)/\hat{\zeta})$.[84] I choose a monthly proportionality constant such that $\sum_j \hat{Q}_{jt} = U_t \times D_t \times \overline{S}$, where the sum is over the top 10,000 websites in month t, U_t is the global number of internet users in that year (from the International Telecommunication Union), D_t is the number of days in the month, and $\overline{S}$ is the average share of daily internet users visiting the top 10,000 websites over the 24 months of available data. In my bootstrap procedure for all downstream estimation, I draw from the estimated asymptotic distribution of $\hat{\zeta}$ and re-compute my traffic estimates for each bootstrap sample.

D. Detecting Software

In this appendix, I explain how I identify the software underlying each homepage archived by Common Crawl. I also describe and list all the web frameworks that are the focus of the main article.

Wappalyzer. I use Wappalyzer, a formerly open source application popular in industry for lead generation.[85] I prefer Wappalyzer over alternatives like BuiltWith because it is more transparent and offers finer control.

Wappalyzer's detection rules are clearly defined, using patterns that match source code and browser metadata to software. Unlike other services (including Wappalyzer's paid API), which provide pre-built historical datasets of URLs and detected software, Wappalyzer can be directly applied to Common Crawl archive files. This avoids concerns about detection changes over time, which could lead to spurious changes in software use in a pre-built dataset constructed over many years.

Three Approaches. I detect software with three approaches: (*i*) "static" detection, which analyzes raw archived information, (*ii*) "dynamic" detection, which loads the archive into a browser to apply additional matching patterns, and (*iii*) "online" detection, which allows the browser to load external pages. I use the union of detections from (*i*) and (*ii*) to construct

[83] Schmeiser's (2015) is based on 2014 data with three-month averages, rather than my one-month averages.

[84] I discard residuals $\hat{\epsilon}^{\#}_{jt}$. I also discard fixed effects $\hat{\bar{\omega}}^{\#}_{t}$, which are only available for the shorter panel.

[85] Wappalyzer was open source on GitHub until August 2023, when it was made private. According to discussions with its creator, this decision was motivated by free-riding from competitors. I use a community continuation of the Wappalyzer project, which remains OSS (github.com/enthec/webappanalyzer).

my panel in Section 3.1. I only use (*iii*) to confirm that not loading external pages does not significantly impact my ability to detect web frameworks.

I begin with a "static" approach, passing the archived URL, HTML, and headers to a simpler version of Wappalyzer (github.com/kikobeats/simple-wappalyzer), which only uses matching patterns that do not require a fully-loaded page in a browser. This method is reliable but does not catch software that only becomes detectable after JavaScript is executed.

Next, I use a "dynamic" approach, loading the archive into a browser while blocking outgoing network requests. I configure Wappalyzer to use a proxy that responds with the archived HTML and headers but blocks all other network requests. Occasionally, this method encounters unrecoverable browser errors;[86] when this happens, I fall back to the static approach. The dynamic approach improves detection, reducing false negatives. For the 78 manually identified web frameworks discussed below, 26% of detections on website-months ranked in the top 10,000 by Alexa would not have been caught by the static method alone.

Blocking network requests helps avoid false positives.[87] However, this could increase false negatives if a framework is only detectable by loading an external script. To check this, I use an "online" approach, which loads the archive into a browser but allows outgoing network requests. For the 78 web frameworks, only 14% of online detections on top 10,000-ranked website-months are missed by the combined static and dynamic methods. Since some of this 14% likely includes false positives, blocking network requests does not seem to cause many false negatives.

Web Frameworks. Wappalyzer is a large application capable of detecting thousands of website features, many of which are niche. With the exception of Footnote 26, for which I collect data on use of OSS and non-OSS servers, I focus on three key parts of websites: backend, JavaScript, and user interface (UI) frameworks. These roughly correspond to Wappalyzer categories of the same name.[88]

These Wappalyzer categories include some niche features, like JavaScript widgets and extensions, which are not the focus of this article. I manually evaluate each feature detected on at least one archived homepage in my sample and identify 78 "actual" web frameworks—those with broad standalone functionality, not just extensions of larger frameworks. Of these, 27 account for 99% of detected usage. For nearly all website-years, only one framework is

[86] Wappalyzer uses the Chromium browser to handle HTML and headers.

[87] For instance, a homepage might load a script from an external URL that did not contain a web framework when archived, but does today.

[88] The exception is that Wappalyzer labels "backend frameworks" as "web frameworks," a term I use to refer to all three parts collectively.

detected per part; in cases of multiple detections, I take the one with more overall usage.

Throughout the article, I focus on the 27 OSS frameworks with significant usage, listed in Appendix Table D1. The remaining 78 frameworks with lower usage are listed in Appendix Table D2. While many of Wappalyzer's detected features are not OSS, the frameworks I focus on are almost exclusively OSS. All 78 frameworks have fairly permissive licenses, which are also listed in the tables.

On the right side of each table, I list the OSS repositories matched to each framework. Most are on GitHub, though a few in Appendix Table D2 are on Gitee. For most frameworks, the majority of activity takes place in a single repository, which I always include. If there is a separate repository for documentation, I include that too, as it is an important component of OSS projects. Some frameworks, like OpenResty, are spread across multiple repositories. I do not match repositories that are unrelated to core functionality, such as plugins or extensions.

Appendix Table D1: Web Frameworks with Significant Use in the Main Sample

Part	Software	License	Matched Repositories
Backend	ASP.NET	MIT	dotnet/aspnetcore
	CakePHP	MIT	cakephp/cakephp
			cakephp/docs
	CodeIgniter	MIT	bcit-ci/codeigniter
			codeigniter4/codeigniter4
	Django	BSD 3 Clause	django/django
	Express.js	MIT	expressjs/express
	Laravel	MIT	laravel/framework
			laravel/laravel
	OpenResty	BSD 2 Clause	calio/form-input-nginx-module
			calio/iconv-nginx-module
			frickle/ngx_postgres
			openresty/array-var-nginx-module
			openresty/echo-nginx-module
			openresty/encrypted-session-nginx-module
			openresty/headers-more-nginx-module
			openresty/lua-cjson
			openresty/lua-nginx-module
			openresty/lua-rds-parser
			openresty/lua-redis-parser

Continued on the next page.

Continued from the previous page.

Part	Software	License	Matched Repositories
			openresty/lua-resty-core
			openresty/lua-resty-dns
			openresty/lua-resty-limit-traffic
			openresty/lua-resty-lock
			openresty/lua-resty-lrucache
			openresty/lua-resty-memcached
			openresty/lua-resty-mysql
			openresty/lua-resty-redis
			openresty/lua-resty-shell
			openresty/lua-resty-signal
			openresty/lua-resty-string
			openresty/lua-resty-upload
			openresty/lua-resty-upstream-healthcheck
			openresty/lua-resty-websocket
			openresty/lua-tablepool
			openresty/lua-upstream-nginx-module
			openresty/memc-nginx-module
			openresty/openresty
			openresty/opm
			openresty/rds-csv-nginx-module
			openresty/rds-json-nginx-module
			openresty/redis2-nginx-module
			openresty/set-misc-nginx-module
			openresty/srcache-nginx-module
			openresty/stream-lua-nginx-module
			openresty/xss-nginx-module
			vision5/ngx_devel_kit
	Ruby on Rails	MIT	rails/rails
	Spring Framework	Apache 2.0	spring-projects/spring-framework
	ThinkPHP	Apache 2.0	top-think/framework
			top-think/think
			top-think/thinkphp
	Yii	BSD 3 Clause	yiisoft/yii
			yiisoft/yii2
JavaScript	Angular	MIT	angular/angular
	Angular.js	MIT	angular/angular.js

Continued on the next page.

Continued from the previous page.

Part	Software	License	Matched Repositories
	GWT	Apache 2.0	gwtproject/gwt
	Marko	MIT	marko-js/marko
	React	MIT	facebook/react
	Stimulus	MIT	hotwired/stimulus
			hotwired/stimulus-rails
	Vue	MIT	vuejs/core
			vuejs/vue
User Interface	Ant Design	MIT	ant-design/ant-design
	Bootstrap	MIT	twbs/bootstrap
			twbs/bootstrap-rubygem
			twbs/bootstrap-sass
			twbs/icons
	Element UI	MIT	elemefe/element
	Emotion.js	MIT	emotion-js/emotion
	Foundation Sites	MIT	foundation/foundation-sites
	MDL	Apache 2.0	google/material-design-lite
	Pure CSS	BSD 3 Clause	pure-css/pure
	Styled Components	MIT	styled-components/jest-styled-components
			styled-components/styled-components
			styled-components/vue-styled-components
			styled-components/xstyled
	UIkit	MIT	uikit/uikit

This table lists the manually-classified web frameworks with significant use detected by Wappalyzer in the main sample of top websites. All repositories are on GitHub.

Appendix Table D2: Web Frameworks Below the Use Cutoff

Part	Software	License	Matched Repositories
Backend	Adonis	MIT	adonisjs/core
			adonisjs/legacy-docs
	Akka HTTP	Apache 2.0	akka/akka-http
	Apache Wicket	Apache 2.0	apache/wicket
	CherryPy	BSD 3 Clause	cherrypy/cherrypy
			cherrypy/cherrypy-obsolete
	F3	GPL 3.0	bcosca/fatfree
	Flask	BSD 3 Clause	pallets/flask
	Frappe	MIT	frappe/frappe
	Koa	MIT	koajs/koa
	Lift Framework	Apache 2.0	lift/framework
			lift/lift
			lift/modules
	Macaron	Apache 2.0	go-macaron/authz
			go-macaron/bindata
			go-macaron/binding
			go-macaron/cache
			go-macaron/captcha
			go-macaron/csrf
			go-macaron/gzip
			go-macaron/i18n
			go-macaron/macaron
			go-macaron/method
			go-macaron/oauth2
			go-macaron/pongo2
			go-macaron/renders
			go-macaron/session
			go-macaron/sockets
			go-macaron/switcher
			go-macaron/toolbox
			rossmeier/piwik-middleware
			xyproto/permissions2
	Meteor	MIT	meteor/meteor
	Mojolicious	Artistic 2.0 Perl	mojolicious/mojo
	Mono	MIT	mono/mono
	Neos Flow	GPL 3.0	neos/flow-development-collection

Continued on the next page.

Continued from the previous page.

Part	Software	License	Matched Repositories
	Nette	BSD 3 Clause	nette/application
		GPL 2.0	nette/bootstrap
		GPL 3.0	nette/caching
			nette/component-model
			nette/database
			nette/di
			nette/forms
			nette/http
			nette/latte
			nette/mail
			nette/neon
			nette/nette
			nette/php-generator
			nette/robot-loader
			nette/routing
			nette/safe-stream
			nette/schema
			nette/security
			nette/tester
			nette/tracy
			nette/utils
	Perl Dancer	Artistic 1.0 Perl	perldancer/dancer
		GPL 1.0 or later	perldancer/dancer2
	Phoenix Framework	MIT	phoenixframework/phoenix
	Play Framework	Apache 2.0	playframework/play1
			playframework/playframework
	Revel	MIT	revel/revel
	Sails.js	MIT	balderdashy/sails
			balderdashy/sails-docs
	Symfony	MIT	symfony/symfony
			symfony/symfony-docs
	Tornado	Apache 2.0	tornadoweb/tornado
	ZK	LGPL 2.1	zkoss/zk
JavaScript	Alpine	MIT	alpinejs/alpine
			alpinejs/alpine-next
	Aurelia	MIT	aurelia/framework

Continued on the next page.

Continued from the previous page.

Part	Software	License	Matched Repositories
	Ember.js	MIT	emberjs/ember-rails
			emberjs/ember.js
			emberjs/guides
	Ext.js	GPL 3.0	tremez/extjs-gpl
	Inferno.js	MIT	infernojs/inferno
	Knockout	MIT	knockout/knockout
			knockout/tko
			knockout/tko-policy
			knockout/tko.provider
			knockout/tko.utils
	Livewire	MIT	livewire/livewire
	Moon.js	MIT	kbrsh/moon
	Right.js	MIT	rightjs/rightjs-core
	Riot	MIT	riot/examples
			riot/riot
	Solid.js	MIT	solidjs/solid
	Svelte	MIT	sveltejs/svelte
	Wink Toolkit	BSD 3 Clause	winktoolkit/wink
	ef.js	MIT	theneuronproject/ef-core
			theneuronproject/ef.js
			theneuronproject/eft-parser
User Interface	Bulma	MIT	jgthms/bulma
	Chakra UI	MIT	chakra-ui/chakra-ui
	Layui	MIT	layui/layui†
			layui/layui-vue†
			layui/layui
			layui/layui-vue
	MDUI	MIT	zdhxiong/mdui†
			zdhxiong/mdui
	MUI	MIT	mui/material-ui
			mui/material-ui-docs
	Materialize	MIT	materializecss/materialize
			materializecss/materialize-docs
	Milligram	MIT	milligram/milligram
	Ripple	Apache 2.0	dpc-sdp/ripple
	TDesign	MIT	tencent/tdesign

Continued on the next page.

Continued from the previous page.

Part	Software	License	Matched Repositories
	Tachyons	MIT	tachyons-css/tachyons
	Tailwind CSS	MIT	tailwindlabs/tailwindcss
	USWDS	CC BY 1.0	uswds/uswds
	UnoCSS	MIT	unocss/unocss
	W3.CSS	MIT	janirefsnes/w3css

This table lists manually-classified web frameworks detected by Wappalyzer in the bottom 1% of use in the main sample of top websites. Repositories with † superscripts are on Gitee rather than GitHub.

E. Matching Contributors and Firms

In this appendix, I explain how I match GitHub contributors to their other online profiles and how I link firms listed on these profiles to standard sources of firm and website information. Both contributor and firm matching involve three-step processes, designed to ensure accurate matches for most contributions while remaining practical. I match contributors first, then use those results to guide the firm matching process.

Matching Data. I match GitHub, Gitee, and LinkedIn profiles of contributors. My GitHub user profile data comes from GHTorrent and GitHub's API, as discussed in Appendix A. Gitee data is directly from Gitee's API, also covered in Appendix A. LinkedIn profile data comes from a consolidated collection of 618 million individual profiles, gathered through multiple scrapes of public information from 2016 through 2022. LinkedIn profiles are especially valuable for providing cleaner data on contributor work histories compared to GitHub and Gitee profiles.

I match firms listed on these profiles to information from LinkedIn, Orbis, Pitchbook, Compustat, Lightcast, and WHOIS. The LinkedIn firm data comes from the consolidated collection of scrapes, which includes 62 million company profiles. My data from Orbis, Pitchbook, and Compustat is standard. Additionally, I use the Whoxy service to obtain historical WHOIS domain registration records for all top-level domains with a monthly Alexa rank of 10,000 or better at any point in my sample.[89]

[89] When fields are not censored for privacy, WHOIS records provide valuable information (on top of websites linked in other sources) about the websites operated by the firms in my sample.

Step 1: Conservative Matching. First, I conservatively match using clean and unique identifiers. Contributor identifiers include email addresses and usernames from GitHub, Gitee, LinkedIn, Twitter, Facebook, Instagram, Weibo, and QQ. I treat usernames as platform-specific because the same username on different platforms can belong to different people. However, I also use a platform-agnostic username-name identifier, which matches profiles with the same username and normalized name across platforms. I normalize text by converting it to lowercase and removing excess whitespace.

Firm identifiers include firm names, LinkedIn text IDs, LinkedIn integer IDs, Orbis BvD IDs, Pitchbook IDs, Compustat GVKEYs, Lightcast IDs, and top-level domains (TLDs). LinkedIn text IDs appear in LinkedIn company page URLs, while their integer IDs consolidate some pages representing the same company. Like contributor identifiers, I normalize firm names and TLDs by converting them to lowercase and removing extra whitespace.

These firm identifiers are less conservative than the identifiers I use to match contributors.[90] To be cautious, I discard some firm identifier observations: names that map to multiple organizations in my GitHub data; names that map to multiple LinkedIn text IDs in my LinkedIn data; and names, LinkedIn text IDs, and TLDs that map to multiple BvD IDs in Orbis, Pitchbook IDs, GVKEYs in Compustat, or Lightcast IDs. Similarly, for LinkedIn profiles, I discard emails that map to multiple LinkedIn usernames to avoid false positives

For contributors, I create a dataset of identifiers by stacking observations from GitHub/Gitee commits and GitHub/Gitee/LinkedIn user profiles. Commits link GitHub/Gitee author usernames with the emails used to author the commit. GitHub user profiles include those archived by GHTorrent and additional profiles I request via GitHub's API (described in Appendix A). Besides social media profiles and emails, GitHub/Gitee profiles can include unstructured URLs and biographies, which I programmatically search for emails and usernames. For all GitHub users who contributed to one of the web frameworks in Appendix D, I also download the HTML from any personal websites linked to their GitHub/Gitee profiles and search the HTML for emails and usernames.

For firms, I create a similar dataset by stacking observations of top-ranked TLDs, companies listed on GitHub/Gitee user profiles, GitHub organization profiles, LinkedIn/Orbis/Pitchbook/Compustat/Lightcast company profiles, alternative company names from these sources, lists of company social media usernames from Orbis/Pitchbook, LinkedIn work experiences of matched contributors, and WHOIS registrants. I also include a manually-constructed crosswalk from Gortmaker, Jeffers, and Lee (2023) linking Compustat firms to

[90] For example, two different firms could have the same generic name.

their LinkedIn company pages. GitHub user profiles can list a company name or its GitHub organization, and some users provide their company's URL, from which I extract the TLD. Organization and company profiles supply TLDs and primary names, while LinkedIn work experiences provide alternative company names for LinkedIn IDs. From WHOIS data, I use the company name when available, and otherwise the full contact name, removing question marks indicating censored information.

Each stacked dataset of identifiers forms an undirected graph, with each observation linking one or more identifiers. The connected components of the graph represent contributors or firms. To compute these connected components, I use a straightforward but inefficient procedure. I begin by treating each observation as a separate contributor or firm, then iteratively group observations by each identifier, assigning them to the same contributor or firm, and repeat until assignments converge. This results in two conservative mappings: one linking contributors' online profiles and the other linking firms to their information sources.

Step 2: Manual Matching. Second, I manually fill in missing matches for top contributors and their employers. The goal is to ensure that most OSS contributions in my sample are accurately matched to firms. Manually matching all contributions is impractical, so I select reasonable activity cutoffs.

For contributor matching, I create a list of all GitHub users with at least 1,000 contributions to the web frameworks I focus on. Contributions include commits, issues, pull requests (PRs), comments, and reviews. If a user is already conservatively matched to other online profiles, I manually verify the accuracy of the match, which is almost always correct—my conservative matching procedure is designed to minimize false positives for top contributors.[91] If a user with more than 1,000 contributions is not conservatively matched to a LinkedIn account, I manually match them if they have one.

For firm matching, I use the results of conservative/manual contributor matching and conservative firm matching to create an employment history for each contributor, detailed in Appendix G. I then create a list of firm identifiers with at least 1,000 contributions to the web frameworks that I focus on, or with at least 10 employed contributors to these frameworks. I manually verify the accuracy of these matches, which, again, are almost always correct given the conservative matching process.[92] If a firm identifier above either cutoff is not conservatively matched to both a TLD and LinkedIn page, I manually match it if possible.

[91] Aside from bots (discussed in Appendix A) I discard a few non-real identifiers, such as "test@test.com."

[92] I do find more non-real firm identifiers than contributor ones, such as "frontend developer" in the GitHub company field or "whoisprivacy limited" in censored WHOIS records, which I discard.

Step 3: Predictive Matching. Finally, I predict remaining matches for smaller contributors and their employers. For contributors, the goal is to match GitHub users who contributed to the web frameworks in my primary sample to their LinkedIn profiles, provided they have LinkedIn accounts. For firms, the goal is to match the company names or GitHub organizations listed on these contributors' GitHub profiles to their LinkedIn company profiles, which have already been linked to other sources of firm information through conservative and manual matching.

For contributors, I create a large dataset of GitHub user and LinkedIn profile pairs. The GitHub users are those who contributed to the web frameworks in my primary sample. The LinkedIn profiles are those listing employment as a software developer at a LinkedIn company matched to a TLD with an Alexa rank of 10,000 or better during my sample period.[93] This dataset contains all possible GitHub-LinkedIn matches relevant to my panel of firms operating the world's most visited websites. I describe below how I eliminate implausible matches using similarity measures.

For firms, I create a large dataset of pairs linking companies listed on GitHub profiles to LinkedIn company pages. The GitHub companies are those listed on the profiles of contributors to the web frameworks in my primary sample, or the GitHub organizations listed in their place. The LinkedIn company pages are those matched to a TLD with an Alexa rank of 10,000 or better during my sample period. As with contributor pairs, this dataset contains all possible GitHub company-LinkedIn company matches relevant to my panel. I explain below how I filter out implausible matches based on similarity measures.

In each dataset of pairs, I use the results from conservative and manual matching to fill in match indicators for GitHub users already matched to a single LinkedIn profile and for companies listed on GitHub profiles already matched to a single LinkedIn company page. These indicators—ones for matched pairs and zeros for others—serve as training data to classify the remaining pairs. I train classifiers using features derived from my GitHub and LinkedIn data. There are two types of features: pair features, which relate to both the GitHub and LinkedIn profiles or companies, and standalone features, which are linked to just one side of the pair.

For contributors, standalone features for GitHub and LinkedIn profiles include an indicator for whether the username matches the lowercase listed name (substituted with the username when the name is unavailable), their respective lengths, and text embeddings. For firms, standalone features for companies listed on GitHub profiles and LinkedIn com-

[93] I classify job titles as software developers in Appendix H.

pany profiles include the length of the company name and its text embedding. I embed usernames and names using FastText's (Joulin et al., 2016) multilingual lid.176.bin model, which generates 16 features per embedding.

For firms, the pair features include the normalized Levenshtein distance between lowercase company names and the cosine similarity between their embeddings. I discard pairs with a normalized Levenshtein distance above 0.7. For contributors, I include Levenshtein distances between GitHub and LinkedIn usernames and names, cosine similarities between embeddings, and discard pairs with both normalized Levenshtein distances above 0.4.[94] Since I have more data on contributors than firms, I also include an indicator for whether GitHub and LinkedIn profiles list the same country and information on overlapping work histories. If both profiles have geocodable but non-overlapping countries, I discard the pair.

For contributors, I add pair features based on overlapping work histories. My GitHub and LinkedIn data contain work history information; I infer employment histories from GitHub data in Appendix G, while LinkedIn histories are taken directly from my data, which is cleaner. For each pair of GitHub and LinkedIn profiles, I list company name pairs with overlapping start and end months, keeping up to 10 pairs with the most overlapping months. For each, I add Levenshtein distances between lowercase company names, their lengths, and their number of overlapping months as pair features.

I use regularized gradient boosting (XGBoost, Chen and Guestrin, 2016) to train binary classifiers on the matched pairs and their features, using XGBoost's default hyperparameters. Each trained classifier predicts matches for the remaining pairs. If a GitHub user is matched to multiple LinkedIn accounts, or a company listed on a GitHub profile is matched to multiple LinkedIn company pages, I select the highest probability match.

F. Classifying Countries, Languages, and Markets

In this appendix, I explain how I assign contributors and firms to countries, and websites to languages and markets. Contributor countries are used to geocode investments not linked to firms in my primary sample. Firm countries are based on headquarter locations. I use website languages and markets for demand estimation, and compare my preferred Similarweb-based markets with alternatives. I rely on direct information when available and train classifiers to impute missing values when necessary.

[94] I set the cutoffs as high as possible while ensuring that training does not exceed 250GB of memory.

Contributor Countries. Contributor matching in Appendix E often provides multiple sources of information about a contributor's self-reported location. If a contributor is matched to a LinkedIn account with a self-reported country, I use the most recently scraped country. Otherwise, I rely on the latest self-reported country that can be successfully geocoded from the contributor's GitHub profile. The geocoding process for GitHub user locations is described in Appendix A.

This procedure fails to identify the country for some contributors to the web frameworks in my sample who lack LinkedIn profiles or self-reported locations. For these contributors, I use regularized gradient boosting (XGBoost, Chen and Guestrin, 2016) to impute missing countries, with default XGBoost hyperparameters. For each GitHub user in my full dataset, I construct features from their latest profile information and activity, train a classifier on those with successfully geocoded self-reported countries, and predict countries for the remaining users.

I construct features informative about GitHub user locations. From each user's latest profile information, I identify the account creation timestamp, whether their lowercase username matches their self-reported name, the lengths of both, and their text embeddings. I generate these embeddings using FastText's (Joulin et al., 2016) multilingual lid.176.bin model, which produces 16 features per embedding.

Most of my features, however, come from users' GitHub activity. I create separate feature sets for each type of activity described in Appendix A: commits, pull requests (PRs), issues, reviews, and comments. For each, I calculate the activity count, the share completed in each hour of the day, the share done on federal holidays for various countries, the share on repositories owned by self-reported residents of each country, and the confidence-weighted share of text content in each of the 176 languages identified by FastText (Joulin et al., 2016), separated into messages, titles, and text bodies. I discard very sparse holiday and owner shares for countries with fewer than 0.1% of self-reported GitHub user locations and remove these rare countries from my list of classes. Similarly, I discard language shares with fewer than 0.1% nonzero values. For each message, title, and text body, I keep only the top two languages, assigning zero confidence to others.

Firm Countries. Firm matching in Appendix E similarly results in multiple sources of information about a firm's headquarters location. I use countries from LinkedIn company pages when available. Otherwise, I rely on information from Compustat, Pitchbook, or Orbis, in that order. If the preferred source lists multiple countries, I choose the one with

the highest employee count.[95]

This process fails to identify the country for some smaller firms, typically websites ranked above the Alexa top 10,000 cutoff used throughout this article, but with little or no firm information matched in Appendix E. For these firms, I train an XGBoost classifier to predict the country for websites ever ranked in Alexa's monthly top 10,000 during my sample period, using default XGBoost hyperparameters. For each top-level domain (TLD), I create features from homepage information and train a classifier on those directly linked to LinkedIn, Compustat, Pitchbook, or Orbis company data.[96] I predict countries for the remaining TLDs, excluding rare countries representing fewer than 0.1% of already-matched TLDs.

I create features informative about the firm operating each website. First, I create an indicator for each TLD extension that represents more than 0.1% of TLDs in my sample, including country-specific extensions like .us and .uk. Second, I use WHOIS domain name registrations data described in Appendix E to link TLDs with contact countries when available. For each TLD, I use the most recent contact country, prioritizing the registrant, then the administrative contact, and finally the technical contact. I create a separate indicator for each country representing more than 0.1% of observations in the WHOIS data.

Additionally, I include rich embeddings of each website's text content. For each Common Crawl archive of the website's homepage (described in Appendix B), I embed the pre-extension TLD text, title, description, keywords, and page text separately. To manage the page text volume, I only use the text from the first 100 visible page elements. Given the large amount of information in the page text, I embed each using the multilingual XLM-R model (Conneau et al., 2019), which generates 768 features per embedding.[97] I average each embedding across all homepage crawls in my data, limiting this to crawls when the website was in Alexa's top 10,000, if available.

Website Languages. Demand estimation in Section 5.1 requires classifying the natural language of each website's text content. I start with each Common Crawl archive of the website's homepage, concatenating the pre-extension TLD text, title, description, keywords, and page text, while limiting it to the first 100 visible page elements. I use FastText's (Joulin et al., 2016) lid.176.bin model to identify the top two most likely languages of this

[95] For example, a firm might be matched to multiple LinkedIn company pages or Compustat GVKEYs. For LinkedIn, I use the most recent employee count. For Compustat, Pitchbook, and Orbis, I use the average employee count from 2014 to 2022.

[96] I treat each source-company as a separate observation for prediction, filtering for TLDs in Alexa's monthly top 10,000 during my sample period, and discarding those without a valid country.

[97] I do not explicitly include language features, as these should be captured within the XLM-R embeddings.

concatenated text, along with their percent confidences, and set the confidence for all other languages to zero. Across archives, I assign the confidence-weighted most likely language.

Website Markets. Demand estimation also requires assigning a market to each website. There are several ways to classify websites, and I evaluate four options. First, I use the 80 categories provided by Similarweb, a web analytics company. Second, I consider using scraped website categories from Curlie (formerly DMOZ), a grassroots, Wikipedia-like project for categorizing the internet. Third, I explore using NAICS and LinkedIn industries from matched firm information.

Overall, I prefer Similarweb categories, which I use throughout the article for market definition. These categories are designed for Similarweb clients, often operators of websites like those in my sample. From evaluating random samples, I find that market definitions based on firm industries are disconnected from the likelihood of substitution between websites, which is what markets aim to capture for the purposes of demand estimation. Industry classifications are also too broad, as websites make up only a small part of the economy. Curlie categories, on the other hand, tend to be too specific.[98]

Another advantage of Similarweb is that its categories are rarely missing. Coverage for other market definitions is much lower and would require more imputation. For the few websites that Similarweb does not classify, I use the TLD extension and text embedding features mentioned earlier to predict website categories. Using these features and Similarweb's category classifications, I train an XGBoost classifier to predict the Similarweb category for websites ever ranked in Alexa's top 10,000 during my sample period, with default XGBoost hyperparameters. I use this classifier to impute the few missing markets in my sample.

G. Measuring OSS Investment

In this appendix, I explain how I measure the hours of investment into web OSS from the firms in my sample. First, I construct employment histories using the contributor and firm matches from Appendix E. These employment histories allow me to attribute OSS activity to firms in my sample, but there are many types of OSS activity, described in Appendix A. To standardize this activity in terms of hours, I develop a simple data-driven approach tailored to the software in my dataset.

[98] While higher levels of Curlie category aggregation are possible, they seem somewhat arbitrary.

Employment Histories. I first construct employment histories for contributors using their matched online profiles and various sources of firm information. For each contributor-month, I prioritize LinkedIn employment data, as my scraped LinkedIn data provides relatively clean employment histories. If a contributor is not matched to a LinkedIn profile or their profile lacks employment data for a specific month, I fall back on information from their GitHub or matched Gitee profiles, in that order.

When needed, I infer employment histories from historical GitHub user page archives, described in Appendix A. For the lowercase company name listed on a GitHub profile's first archive, I set its start month to the profile's creation month. For the name listed on the final archive, I set its end month to the end of my sample. For others, I set the start month to when the company name first appears and set the previous company's end month to this same month. I use the matching results from Appendix E to link these company names to other firm information. In rare cases where a contributor has no LinkedIn or GitHub employment data but has a Gitee account, I assume they are employed at the company listed in my one archive of their Gitee page, if available, throughout the sample.

This process results in a list of firms where each contributor is employed during each month. Typically, only one firm is listed: the contributor's employer. If there are no firms listed, I include the contributor's OSS activity in aggregate and country-level investment measures (using the contributor's country) but do not attribute their activity to any firms in my sample. If multiple firms are listed, I attribute equal shares of their OSS activity to each firm.

Hours Estimation. In Appendix A, I describe the five main types of OSS collaboration on the web frameworks in my sample: commits, issues, pull requests (PRs), reviews, and comments. I develop a simple procedure to convert all of this activity into hours, which I sum into the OSS investment measures used throughout the article. I am optimistic that this approach can be applied in other research to derive context-specific measures of OSS investment, reflecting the many ways OSS collaboration occurs.

The key requirement for my approach is a measure of the average hours per month spent contributing by those in my sample. I base this on a recent survey (Nagle et al., 2020) of top OSS contributors. Among 511 respondents who currently contribute to OSS, the average weekly contributions is 14.8 hours. When I restrict this to the 85 respondents who work on web OSS,[99] their average weekly contribution is slightly lower at 13.1 hours. Since the

[99]These are the contributors whose OSS work is, according to them, best categorized as on web frameworks, HTTP modules, CSS, or content management systems

contributors to web OSS in my sample closely match this group, I use the lower estimate.

This average, scaled to a monthly frequency, forms the left-hand side of a "hedonic" regression, where observations are contributor-months during my primary sample period of 2014 to 2022. I discard contributor-months with no commits to the web OSS repositories in Appendix Tables D1 and D2 to focus on estimates relevant to the software I study, and because surveyed contributors were told that "active" meant committing.

The goal of the "hedonic" regression is to decompose the total hours spent on OSS contributions into hours from each category of OSS activity. To do this, I include a separate regressor for each activity type. Since the left-hand side reflects contributions to all OSS, not just the primary web OSS in my sample, it's important to count each contributor's total OSS contributions for a given month on the right-hand side. My comprehensive GitHub (and Gitee, for applicable contributors) activity data allows me to construct these total counts for each contributor and month.

This approach also makes it possible to account for differences between contributors. For instance, one contributor might take more time than another to make a typical commit. Various contributor characteristics can be included on the right-hand side of the regression. In practice, I find that whether a contributor reviews code on an OSS project is a particularly predictive factor—these reviewers generally require fewer hours to complete similar activities. I include separate regressors for their activity.

Specifically, let $\mathbb{E}[L_{it}]$ denote the survey-based average hours contributing to OSS, where t denotes months and i represents contributors. Let X^L_{ict} denote the number of contributions made by i in month t for activity $a \in \mathcal{A}$. If i is a reviewer, non-reviewer activity categories are zero, and vice versa. I estimate the following non-standard regression with a constant left-hand side and without an intercept on the right-hand side:

$$\mathbb{E}[L_{it}] = \sum_{a \in \mathcal{A}} \lambda_a \, X^L_{ait} + \varepsilon^L_{it}. \tag{G1}$$

The residual ε^L_{it} captures how contributors' actual hours spent contributing in a given month differ from the survey-based average. Each coefficient λ_a measures the hours required for activity a. Since the counts X^L_{ait} can include outliers, I winsorize nonzero counts at the 99% level within each month.

I report estimates in Appendix Table G1. All point estimates are positive, though they can be negative in bootstrap samples, where I resample contributors. Since downstream estimation requires nonnegative investment, I use nonnegative least squares for bootstrap

samples. I report 95% confidence intervals in brackets.

I find that for the web OSS in my sample, a typical PR takes between one and two hours. Conditional on the number of PRs, commits and attached reviews take less time. Importantly, this does *not* mean that analyzing commits alone would significantly understate the overall OSS investment. If this regression were run using only commits, their coefficient would be larger, capturing the activity correlated with each commit.

When constructing my OSS investment measures, I convert activity counts into hours using the estimated "hedonic" coefficients $\hat{\boldsymbol{\lambda}}$. Since my focus isn't on whether, for example, commits generate more value than issues, or whether reviewers contribute more than others, I simply sum the hours across different activity categories. However, future research focused on the specifics of OSS collaboration could keep these estimates separate.

Appendix Table G1: Hours Estimates

	Mean Hours Contributed	
	Reviewers	Others
Commits	0.10	0.19
	[0.08, 0.11]	[0.17, 0.21]
Issues	0.50	0.62
	[0.34, 0.68]	[0.39, 0.85]
Pull requests	0.77	1.92
	[0.66, 0.87]	[1.76, 2.05]
Comments	0.07	0.09
	[0.05, 0.10]	[0.04, 0.13]
Reviews	0.03	
	[0.00, 0.07]	
R^2	0.34	
Contributor-months	94,867	
↪ Contributors	36,104	
↪ Months	108	

This table reports results from the nonnegative least squares regression of the survey measure for mean hours per month spent contributing on the number of counts for each activity category. Nonzero counts are winsorized from above at the 99% level within each month. In brackets, 95% confidence intervals are from 80 bootstrap samples, for which I resample contributors with replacement.

My estimates $\hat{\boldsymbol{\lambda}}$ reflect the typical hours spent on web OSS contributions in my sample. I do not expect $\boldsymbol{\lambda}$ to be the same in other contexts. For instance, contributors to other types of OSS might write larger PRs or commits that take more time. Adapting my approach to other settings would require re-estimating the regression with left- and right-hand sides tailored to those contexts.

H. Measuring In-House Investment

In this appendix, I explain how I measure hours of in-house web development investment for the firms in my sample. I begin with matches from Appendix E between firms and their LinkedIn pages. Using my scraped LinkedIn data, I estimate how many web developers each firm employs and scale this by survey-based estimates of their average workweeks. Finally, I subtract the small amount of OSS investment attributed to firms from Appendix G.

My LinkedIn data also allows me to estimate the total number of workers employed by each firm, which I use as a proxy for production function estimation in Appendix J. By breaking this down into web developers, other software developers, and non-developers, I estimate the population of web developers and other software developers by country and globally in Section 2.3.

Identifying Developers. My scraped LinkedIn data includes only raw job titles in work experiences. To identify different types of web developers, I use data from Lightcast (formerly Burning Glass), which maps many LinkedIn job titles on US profiles to SOC codes.

Across different SOC versions, web developer codes are 5-1134, 15-1257, 15-1254, and 15-1255. More broadly, software developer codes are 15-113 and 15-125. Using these, I create three classes of work experiences: web developers, other developers, and non-developers. For lowercase job titles that Lightcast matches to only one class,[100] I assign that class. For the rest, I use Lightcast matches as training data, where each observation is a job title-class pair.

I use FastText's (Joulin et al., 2016) multilingual lid.176.bin model to create 16 features for each lowercase job title. Then, I train a classifier using regularized gradient boosting (XGBoost, Chen and Guestrin, 2016), with these features as inputs and my three classes as the output. I use default XGBoost hyperparameters. Once trained, I use the classifier to predict the classes for job titles that cannot be uniquely matched to a class using the Lightcast data.

[100] Lightcast uses a two-step procedure to match job titles to SOC codes: pattern matching followed by a machine learning model that incorporates job title and profile experience. Multiple SOC codes may be matched if there is no direct pattern match, such as when the job title is vague.

Adjusting for LinkedIn Penetration. My LinkedIn data is a consolidated collection of 618 million individual profiles and 62 million company profiles, gathered through multiple scrapes of public information from 2016 through 2022. Individual profiles include work experiences, which are typically linked to company profiles. For each LinkedIn company and month, I count the number of LinkedIn users with a work experience at the company that overlaps with the month and has a web developer job title. However, this only captures the number of web developer employees who list their employment on LinkedIn.

Not all web developers use LinkedIn, and LinkedIn penetration rates vary across occupations and countries. If unaddressed, both factors could bias my estimates of in-house investment. To estimate LinkedIn penetration rates, I make an assumption. I assume the penetration rate for an occupation category *relative* to the unconditional penetration rate is the same across countries:

$$\frac{\mathbb{P}(\text{LinkedIn} \mid \text{country, occupation})}{\mathbb{P}(\text{LinkedIn} \mid \text{country})} = \frac{\mathbb{P}(\text{LinkedIn} \mid \text{other country, occupation})}{\mathbb{P}(\text{LinkedIn} \mid \text{other country})}. \tag{H1}$$

I estimate the denominator $\mathbb{P}(\text{LinkedIn} \mid \text{country})$ by dividing the yearly number of users in my LinkedIn data for each country by the country's workforce data from the World Bank. For $\mathbb{P}(\text{LinkedIn} \mid \text{US, occupation})$, I divide the yearly number of US web developers and other developers in my LinkedIn data by the US Bureau of Labor Statistics (BLS) estimate of their total numbers, using the same SOC codes as before.

With the US estimates for the right-hand side and the denominator for each country, I compute the LinkedIn penetration rate $\mathbb{P}(\text{LinkedIn} \mid \text{country, occupation})$ for each country and occupation category. When World Bank data are missing for a country in a given year, I linearly interpolate its penetration rate. If World Bank data are always missing for a country, I use employee-weighted global penetration rates.

Developer penetration rates generally range from 5% to 50%. For example, the web developer penetration rate for China—where LinkedIn is less commonly used—is around 5%, while it is about 2% for other developers. In some cases, penetration rates exceed 100%. In the US, for instance, web developer penetration rates range from 100% to 200%. This suggests my classification of web developers based on LinkedIn job titles is broader than the BLS classification; adjusting for penetration above 100% helps align my estimates with the BLS definition of web developers.

Converting to In-House Hours. For each firm and year in my sample, I compute the number of web developers employed on LinkedIn and divide this by the LinkedIn penetration

rate for the firm's country and year. This provides an estimate of the firm's investment in web development, expressed in terms of developers. To convert this into hours, I scale these counts using country-specific average workweeks from survey data.

Specifically, I calculate country-level averages for web developer workweeks based on two recent large-scale surveys (Stack Overflow, 2019, 2020).[101] For countries with fewer than 100 responses, I use the global average. Averages range from 37 to 45 hours per week, with 40 hours being the most common. I use these averages—scaled by 52 weeks per year—to convert web developer counts into total yearly investment in web development.

Finally, I adjust for hours of OSS investment from Appendix G. This adjustment is minor because web OSS investment is rare compared to total web development investment. To adjust carefully, however, I subtract my estimate of yearly hours invested into the web OSS in my sample, scaled by an average of 48% of hours spent on OSS during work hours, which I take from the same sample of survey respondents that I use to compute the average weekly hours spent contributing to OSS in Appendix G.

I. Demand Estimation Details

In this appendix, I provide details about website demand estimation from Section 5.1. Estimation follows standard micro BLP (Berry, Levinsohn, and Pakes, 1995, 2004), so I use PyBLP, my own OSS for demand estimation (Conlon and Gortmaker, 2020, 2025). I describe my data, how I parameterize the demand system, define and adjust for market size mismeasurement, and estimate the demand system. Finally, I explain how I express consumer preferences in dollars using a back-of-the-envelope calculation.

Demand Data. The two main inputs for demand estimation are my traffic estimates from Appendix C and additional data from Similarweb on country-specific traffic shares. While the traffic estimates cover the full sample, my Similarweb data on country-specific shares is limited to 2021. For that year, it includes most websites in my sample. I use Similarweb shares to scale my traffic estimates, generating country-specific traffic estimates for 2021.

Demand Parameterization. In each year $t = 2014$ to 2022, I estimate demand for websites $j \in \mathcal{J}_t$ ranked 10,000 or better according to my monthly Alexa traffic data from

[101] These surveys include 20,944 responses on weekly hours worked by employed developers with any work on PHP, CSS/HTML, or "web frameworks" in the US, 397 in China, and 65,397 from other countries. I clip a few outliers above 80 hours per week and average the results within each country.

Appendix C. I segment websites into markets $m \in \mathcal{M}$ using the 80 Similarweb categories discussed in Appendix F. Let $\mathcal{J}_{mt} \subset \mathcal{J}_t$ denote the websites in market m.

Consumers are differentiated by their types $i \in \mathcal{I}_t$, which are country-language pairs. Let $c(i)$ and $l(i)$ denote the country and language of type i, and let $c(j) = c(f(j))$ and $l(j)$ denote the country and language of website j. Using this notation, the "home bias" indicators in my parameterization of systematic utility in (13) are

$$\boldsymbol{X}_{ijt}^{V\prime} = \left[\mathbb{1}_{c(i)=c(j)}\left[\mathbb{1},\ \mathbb{1}_{c(i)=\text{US}},\ \mathbb{1}_{c(i)=\text{China}}\right],\ \mathbb{1}_{l(i)=l(j)}\left[\mathbb{1},\ \mathbb{1}_{l(i)=\text{English}},\ \mathbb{1}_{l(i)=\text{Chinese}}\right]\right]. \tag{I1}$$

Potential Demand. Estimation requires taking a stance on the potential traffic Q_{imt} in (14) from consumers of type i to websites in market m. Let $Q_t = \sum_{i \in \mathcal{I}_t} \sum_{m \in \mathcal{M}} Q_{imt}$ denote the overall potential traffic in year t. I simply assume

$$Q_t = U_t \times 20, \tag{I2}$$

in which U_t is the number of internet users from the International Telecommunication Union, and 20 websites per day is roughly the 99th percentile of daily in-sample visits in 2020 Comscore browsing data for the US. This scaling reflects an untestable assumption common in most market size definitions; below, I discuss how I ensure that my estimates and simulations are not significantly affected by this assumption. I decompose global potential traffic as

$$Q_{imt} = Q_t \times A_{c(i)} \times \frac{U_{c(i)t}}{U_t} \times S_{c(i)l(i)} \times \frac{\sum_{j \in \mathcal{J}_{mt}} Q_{jt}}{\sum_{j \in \mathcal{J}_t} Q_{jt}}, \tag{I3}$$

where A_c is a Similarweb-based adjustment to the global 20 websites per day for country c, U_{ct}/U_t is the share of internet users in country c, S_{cl} is the Ethnologue (Lewis et al., 2016) population share of country c with primary language l, and the rightmost term is the within-sample traffic share of market m.

Each term introduces additional assumptions. By distributing potential traffic to different markets based on within-sample traffic shares, I assume that traffic to websites outside Alexa's top 10,000—captured by each market's outside alternative—follows a similar pattern across markets. By distributing a country's internet users according to primary language shares based on overall populations, I assume these language shares are the same for the country's internet users.

The Similarweb-based adjustment A_c accounts for differences in internet user activity

across countries. Intuitively, it functions similarly to including country-specific fixed effects in my parameterization of systematic utility.[102] For the 12 months in 2021 for which I have country-specific traffic from Similarweb, I compute the amount of in-sample traffic from each country and scale this by the global share of traffic covered by Similarweb to estimate each country-month's potential traffic. I divide this by the number of internet users and the days in each month, then average across 2021 to get a country-specific estimate of websites visited per day. The adjustment A_c is this value divided by the baseline of 20 websites per day.

Market Size Mismeasurement. Like most measures of potential demand, mine is imperfect and relies on several assumptions that are either difficult to verify or untestable. Fortunately, my policy simulations in Section 7 do not focus on substitution between inside alternatives and the outside option, which is particularly sensitive to such assumptions (Zhang, 2024). To ensure my estimates and simulations are not overly affected by how I define market sizes, I make two choices, discussed in Sections 5.1 and 7.4, which I describe in more detail here.

First, when estimating my quality production function, a concern is that misspecified market sizes could bias my estimates. In a simple logit model with no random coefficients, $\boldsymbol{\beta}_V = \mathbf{0}$, biased market sizes appear as market-year fixed effects $\bar{\omega}^{\Xi}_{mt}$, so I always include these when estimating the production function. These fixed effects also help reduce bias in models with random coefficients (Zhang, 2024).

Even if my estimators of $\boldsymbol{\beta}_V$, β_P, and β_S were unbiased, a remaining concern is that misspecified market sizes could bias my profit maximization simulations in Section 7.4.[103] To prevent resulting unrealistic outside substitution from driving my results, I fix this margin by adjusting the systematic utility each consumer receives from the outside option. Specifically, in my profit maximization simulations, I replace the normalization of $V_{i0t}(\boldsymbol{\xi}_t) = 0$ during estimation with the $V_{i0t}(\boldsymbol{\xi}_t)$ required to fix $Q_{i0t}(\boldsymbol{\xi}_t)$ to its estimated value. This assumption seems reasonable since the outside option includes all out-of-sample websites in a market, which are likely produced similarly to those in my sample and would experience similar quality changes.

[102] I opt for adjusting each country's potential demand to avoid introducing many new nonlinear parameters.

[103] Large potential demand implies large outside shares and, consequently, a large logit quality for the outside option, potentially leading to significant substitution from inside alternatives to the outside option in counterfactual simulations.

Micro Moments. I estimate the coefficients $\boldsymbol{\beta}_V$ on the "home bias" indicators $\boldsymbol{X}_{ijt}^V$ in (I1) by matching one observed statistic per indicator with its model-predicted counterpart. These statistics use my Similarweb data from 2021, so I only compute model-predicted counterparts in 2021. Using the notation-abusing shorthand from Conlon and Gortmaker (2025), I match

$$\begin{aligned}
&\text{“}\mathbb{P}(\mathbb{1}_{c(i)=c(j)} \mid j \neq 0), \quad \mathbb{P}(\mathbb{1}_{c(j)=c(j)} \mid c(j) = \text{US}), \quad \mathbb{P}(\mathbb{1}_{c(j)=c(j)} \mid c(j) = \text{China}),\text{”} \\
&\text{“}\mathbb{P}(\mathbb{1}_{l(c(i))=l(j)} \mid j \neq 0), \quad \mathbb{P}(\mathbb{1}_{l(c(j))=l(j)} \mid l(j) = \text{English}), \quad \mathbb{P}(\mathbb{1}_{l(c(j))=l(j)} \mid l(j) = \text{Chinese}),\text{”}
\end{aligned}$$

where $l(c)$ denotes the language of country c with the highest population share. Moments based on $l(i)$ would be more informative about coefficients in $\boldsymbol{\beta}_V$ on language indicators, but my Similarweb data only contains country-specific traffic shares and does not report separate shares by language of internet users.

Since the discrete choices in my model of web traffic demand are units of traffic, these micro moments are traffic shares. I report their observed values in Table 3. As the model is just-identified, with exactly one micro moment per parameter, my estimation procedure matches each statistic exactly, up to numerical error.

Estimation Procedure. My minimum distance estimator for $\boldsymbol{\beta}_V$ is simpler than the full estimator in Conlon and Gortmaker (2025) because I only use micro moments. Since the model is just identified, I can use the identity weighting matrix without any loss of efficiency:

$$\hat{\boldsymbol{\beta}}_V = \underset{\boldsymbol{\beta}_V}{\operatorname{argmin}}\, \boldsymbol{g}_V(\boldsymbol{\beta}_V)' \boldsymbol{g}_V(\boldsymbol{\beta}_V), \tag{I4}$$

in which $\boldsymbol{g}_V(\boldsymbol{\beta}_V)$ consists of differences between observed traffic shares and their model counterparts.

To compute $\hat{\boldsymbol{\beta}}_V$, I follow the recommendations in Conlon and Gortmaker (2020, 2025). I optimize $\boldsymbol{\beta}_V$ using SciPy's trust-region algorithm and consistently achieve an objective value numerically indistinguishable from zero. For each guess of $\boldsymbol{\beta}_V$, I first use the SQUAREM algorithm from Varadhan and Roland (2008) to accelerate the Berry, Levinsohn, and Pakes (1995) contraction mapping used to recover website qualities $\hat{\xi}_{jt}(\boldsymbol{\beta}_V)$. I terminate with a strict absolute tolerance of 10^{-14}. Second, I compute the model counterpart for each observed micro statistic. For example,

$$\text{“}\mathbb{P}(\mathbb{1}_{c(i)=c(j)} \mid j \neq 0)\text{”} = \frac{\sum_{i \in \mathcal{I}_t} \sum_{m \in \mathcal{M}} \sum_{j \in \mathcal{J}_{mt}} Q_{ijt}(\boldsymbol{\beta}_V) \cdot \mathbb{1}_{c(i)=c(j)}}{\sum_{i \in \mathcal{I}_t} \sum_{m \in \mathcal{M}} \sum_{j \in \mathcal{J}_{mt}} Q_{ijt}(\boldsymbol{\beta}_V)}, \tag{I5}$$

in which $t = 2021$. I use Bayes' Rule to compute the model counterparts conditioned on the US, China, English, and Chinese.

Since my micro moments are only for 2021, computing $\hat{\boldsymbol{\beta}}_V$ requires calculating website qualities $\hat{\xi}_{jt}(\boldsymbol{\beta}_V)$ only for 2021. However, after estimation, I compute the estimated website qualities $\hat{\xi}_{jt}$ for all years in my sample, which are inputs for quality production function estimation. I also calculate the estimated consumer type-level traffic $\hat{Q}ijt$, which are inputs for my simulation of losing access to Facebook.

Facebook Simulation. To express $\hat{\boldsymbol{\beta}}_V$ in dollar terms, I use a back-of-the-envelope calculation based on simulating the experiment conducted in 2022 by Brynjolfsson et al. (2023), which measured Facebook users' willingness to accept (WTA) compensation for losing access to the website for a month. I use their pooled WTA estimate of \$31, scaled by 30 days per month.

Specifically, let k be Facebook, let $m = m(k)$ be its market, let t be 2022, let $\mathcal{I}'_t \subset \mathcal{I}_t$ be consumers in the 13 surveyed countries, and denote the inclusive value of websites $\mathcal{J}$ by

$$\hat{V}_{it}(\mathcal{J}) = \log \sum_{j \in \mathcal{J} \cup \{0\}} \exp\left(\hat{\boldsymbol{\beta}}'_V \boldsymbol{X}^V_{ijt} + \hat{\xi}_{jt}\right). \tag{I6}$$

Using my estimated model of demand, I compute a utility-to-dollars conversion factor

$$\hat{\tau} = \frac{\$31}{30} \div \frac{\sum_{i \in \mathcal{I}'_t} \hat{Q}_{ikt} \cdot \left(\hat{V}_{it}(\mathcal{J}_{mt}) - \hat{V}_{it}(\mathcal{J}_{mt} \setminus \{k\})\right)}{\sum_{i \in \mathcal{I}'_t} \hat{Q}_{ikt}}, \tag{I7}$$

in which weighting by $\hat{Q}_{ikt}$ selects Facebook users.

In Appendix Table I1, I report the unscaled estimates of $\hat{\boldsymbol{\beta}}_V$ underlying Table 3. For the preferred specification with both types of preference heterogeneity, I also report $\hat{\tau} \times \hat{\boldsymbol{\beta}}_V$, expressed in dollars per daily visit. I report 95% confidence intervals in brackets. The only source of uncertainty in my demand estimates comes from traffic estimation in Appendix C.[104]

[104] Since my traffic data and Similarweb country shares are based on an unknown—but presumably large—number of underlying observations, I do not account for sampling error in my micro moments.

Appendix Table I1: Demand Parameter Estimates

		Preference Heterogeneity			
		Country	Language	Both	Dollars
Same country	All consumers	+2.1 [+1.9, +2.3]		+1.5 [+1.4, +1.7]	+9.8 [+8.8, +10.8]
	× US-based	−1.4 [−1.6, −1.2]		−1.1 [−1.3, −1.0]	−7.4 [−8.2, −6.6]
	× China-based	+3.7 [+3.2, +4.4]		+3.6 [+3.1, +4.3]	+23.3 [+20.3, +27.5]
Same main language	All consumers		+4.5 [+4.4, +4.6]	+4.3 [+4.3, +4.4]	+27.9 [+27.8, +28.1]
	× English-speaking		−4.0 [−4.1, −3.9]	−3.8 [−3.9, −3.8]	−24.7 [−25.1, −24.6]
	× Chinese-speaking		+1.2 [+0.9, +1.6]	−2.2 [−2.3, −2.1]	−14.3 [−14.9, −13.5]

This table reports unscaled parameter estimates underlying Table 3. The estimates $\hat{\boldsymbol{\beta}}_V$ are coefficients on indicators in $\boldsymbol{X}^V_{ijt}$ for consumer type i and website j being from the same country and using the same main language, with separate interactions for the US, China, English, and Chinese. The rightmost column scales $\hat{\boldsymbol{\beta}}_V$ by $\hat{\tau}$, converting the values to dollars per daily visit. In brackets, 95% confidence intervals are from 80 bootstrap samples that account for sampling error from traffic estimation in Appendix C.

J. Production Function Estimation Details

In this appendix, I provide details about estimating the production function for website quality in Section 5.2. Since the quality production function relies on parameters I estimate using revealed preferences from cost minimization in Appendix K, I perform these two steps jointly. However, conditional on depreciation δ_P and weights $\boldsymbol{\alpha}$, production function estimation follows a standard two-step procedure.

Specifically, I follow the literature that uses proxy variables and timing assumptions (Olley and Pakes, 1996; Levinsohn and Petrin, 2003; Ackerberg, Caves, and Frazer, 2015). I also compare my proxy variable estimates with those from alternative approaches (e.g., Blundell and Bond, 1998, 2000), which relax some assumptions but impose others. My discussion in this appendix mirrors some of the discussion in Ackerberg (2023).

Proxy Variable Approach. My production function in (12) has two primary regressors. Let $\boldsymbol{\beta}_\Xi = [\beta_P, \beta_S]'$ be the coefficients on the log of private capital and weighted OSS use:

$$\boldsymbol{X}^\Xi_{fjt} = \left[\log K_{ft},\ \sum_{p\in\mathcal{P}} \alpha_p\, \mathbb{1}_{s_{jpt}\in\mathcal{OS}_t}\right]'. \tag{J1}$$

Given my assumption that ω^Ξ_{jt} in (12) is a flexible function of these two regressors and the proxy variable, total employees E_{ft}, the first step in the standard two-step procedure is to run the following fixed effects regression:

$$\xi_{jt} = \boldsymbol{\beta}'_\Xi \boldsymbol{X}^\Xi_{fjt} + \overline{\omega}^\Xi_{m(j)t} + \omega^\Xi_{jt} + \varepsilon^\Xi_{jt} = \Phi^\Xi_{fjt}(\boldsymbol{X}^\Xi_{fjt}, E_{ft}) + \overline{\omega}^\Xi_{m(j)t} + \varepsilon^\Xi_{jt}, \tag{J2}$$

where I replace quality with $\hat{\xi}_{jt}$ from demand estimation in Appendix I, and I replace Φ^Ξ_{fjt} with a quadratic polynomial of the regressors in $\boldsymbol{X}^\Xi_{fjt}$ and $\log E_{ft}$.

This first step provides an estimator $\hat{\Phi}^\Xi_{fjt}$ that I use in the second step. Given a guess of $\boldsymbol{\beta}_\Xi$, I compute $\hat{\omega}^\Xi_{jt}(\boldsymbol{\beta}_\Xi) = \hat{\Phi}^\Xi_{fjt} - \boldsymbol{\beta}'_\Xi \boldsymbol{X}^\Xi_{fjt}$. Given my assumption in (15) that ω^Ξ_{jt} follows a first-order Markov process, I then run the following regression:

$$\omega^\Xi_{jt} = g(\omega^\Xi_{jt-1}) + \nu^\Xi_{jt}, \tag{J3}$$

where I replace each ω^Ξ_{jt} with $\hat{\omega}^\Xi_{jt}(\boldsymbol{\beta}_\Xi)$ and I replace g with a quadratic polynomial. This provides an estimator $\hat{\nu}^\Xi_{jt}(\boldsymbol{\beta}_\Xi)$, which I use to form sample moments:

$$\hat{\boldsymbol{\beta}}_\Xi = \underset{\boldsymbol{\beta}_\Xi}{\operatorname{argmin}}\, \boldsymbol{g}_\Xi(\boldsymbol{\beta}_\Xi)'\, \boldsymbol{W}_\Xi\, \boldsymbol{g}_\Xi(\boldsymbol{\beta}_\Xi), \quad \boldsymbol{g}_\Xi(\boldsymbol{\beta}_\Xi) = \hat{\mathbb{E}}\big[\nu^\Xi_{jt}(\boldsymbol{\beta}_\Xi)\cdot \boldsymbol{Z}^\Xi_{fjt-1}\big], \tag{J4}$$

where $\hat{\mathbb{E}}$ denotes the average over website-years with at least two lags, starting in 2016.

Instruments $\boldsymbol{Z}^\Xi_{fjt-1} = [\boldsymbol{X}^{\Xi\prime}_{fjt-1}, \boldsymbol{X}^{\Xi\prime}_{fjt-2}]'$ are two lags of the regressors. As discussed in Footnote 47, I include two lags to help with the under-identification issue documented by Ackerberg, Frazer, il Kim, Luo, and Su (2023). I use two-step GMM with initial weighting matrix $\boldsymbol{W}_\Xi = \hat{\mathbb{C}}(\boldsymbol{Z}^\Xi_{fjt})^{-1}$.

Appendix Table J1: Quality Parameter Estimates

		Unscaled	Dollars
$\hat{\beta}_P$	Log of private capital	+0.25 [+0.22, +0.29]	+1.62 [+1.40, +1.85]
$\hat{\beta}_S$	OSS indicator	−0.20 [−0.49, +0.12]	−1.30 [−3.12, +0.77]

This table reports the unscaled $\hat{\beta}_P$ and $\hat{\beta}_S$ estimates underlying Table 4, and in the right column, scales these estimates by the conversion factor from Appendix I, expressing the parameters in dollars per daily visit. In brackets, 95% confidence intervals are from 80 bootstrap samples that account for sampling error from traffic estimation in Appendix C, OSS investment estimation in Appendix G, demand estimation in Table 3, and supply estimation in Table 4.

Estimation Results. I report $\hat{\boldsymbol{\beta}}_{\Xi} = [\hat{\beta}_P, \hat{\beta}_S]'$ in the unscaled column of Appendix Table J1. In the second column, I scale by the conversion factor $\hat{\tau}$ from Appendix I, which puts $\hat{\beta}_P$ and $\hat{\beta}_S$ in dollars per daily visit. I report 95% confidence intervals in brackets. Uncertainty comes from estimation of traffic in Appendix C, OSS investment in Appendix G, demand in Appendix I, and δ_P and $\boldsymbol{\alpha}$ in Appendix K. For each bootstrap sample, I re-sample firms before re-estimating the quality production function.

Alternative Approaches. In Appendix Table J2, I compare estimates of $\hat{\beta}_P$ and $\hat{\beta}_S$ from the above proxy variable approach with other methods. The middle column is the same as the unscaled column from Appendix Table J1. To its left, I report results from an OLS regression of $\hat{\xi}_{jt}$ on $\boldsymbol{X}^{\Xi}_{fjt}$ and market-year fixed effects. I then add website fixed effects (FE). When simultaneity bias is not addressed, $\hat{\beta}_P$ is biased downward. Adding website fixed effects helps, but does not fully account for the endogeneity of residual unobserved quality after removing fixed effects.

Appendix Table J2: Alternative Approaches to Production Function Estimation

	OLS	FE	Proxy	DP	BB
$\hat{\beta}_P$	+0.10 [+0.04, +0.15]	+0.17 [+0.08, +0.24]	+0.25 [+0.22, +0.29]	+0.31 [+0.14, +0.62]	+0.63 [+0.17, +1.23]
$\hat{\beta}_S$	+0.13 [−0.12, +0.41]	−0.15 [−0.31, −0.02]	−0.20 [−0.49, +0.12]	−0.43 [−0.79, −0.08]	−4.42 [−22.24, +39.88]
$\hat{\rho}_\Xi$				+0.97 [+0.96, +0.98]	+0.77 [+0.71, +0.87]
Firm-years	10,555	10,555	10,555	10,555	10,555
↪ Firms	2,481	2,481	2,481	2,481	2,481
↪ Years	7	7	7	7	7

This table reports estimates from different approaches to production function estimation, all accounting for market-year fixed effects. From left to right, the ordinary least squares (OLS) approach runs a simple regression, the fixed effect (FE) approach adds a website fixed effect, the proxy variable approach (Olley and Pakes, 1996; Levinsohn and Petrin, 2003; Ackerberg et al., 2015) is the baseline specification used for Table 4 and Appendix Table J1, the "dynamic panel" (DP) approach replaces the proxy variable with a parametric assumption about unobserved quality transitions, and the Blundell and Bond (1998, 2000) approach differences out a website fixed effect. All approaches use the same sample of website-years with two valid lags. In brackets, 95% confidence intervals are from 80 bootstrap samples that account for sampling error from traffic estimation in Appendix C, OSS investment estimation in Appendix G, demand estimation in Table 3, and supply estimation in Table 4.

To the right of the proxy column, I report results using approaches from the dynamic panel literature. I discuss the results before describing each estimation procedure. I begin with a simple "dynamic panel" (DP) approach that drops the proxy variable assumption but imposes a stronger assumption that ω_{jt}^{Ξ} follows an AR(1). The results for $\hat{\beta}_P$ are similar, and although $\hat{\beta}_S$ is more negative, it still has a wide confidence interval. The estimated autocorrelation $\hat{\rho}_\Xi$ of ω_{jt}^{Ξ} is very high, indicating that unobserved website quality is highly persistent.

Finally, I compare with the approach of Blundell and Bond (1998, 2000), which supplements the DP approach with "double differencing" to allow for website fixed effects. This method is more demanding on my data, so I follow convention and impose additional stationarity restrictions on the firms' operating environments.[105] Although $\hat{\beta}_P$ and $\hat{\beta}_S$ are much noisier, the general conclusion remains the same: $\hat{\beta}_P$ is fairly consistent across specifications that account for simultaneity bias, while $\hat{\beta}_S$ remains noisier.

[105] In my setting, Arellano and Bover's (1995) stationarity assumptions require that inputs for persistently high-quality websites, on average, after accounting for market-year fixed effects, change neither faster nor slower than those for lower-quality websites (Ackerberg, 2023).

"Dynamic Panel" Approach. For the DP column, I assume ω_{jt}^{Ξ} follows an AR(1) with innovations that satisfy the same moment conditions as in (15):

$$\omega_{jt}^{\Xi} = \rho_{\Xi}\, \omega_{jt-1}^{\Xi} + \nu_{jt}^{\Xi}. \tag{J5}$$

Let Δ_{ρ_Ξ} denote the quasi-first difference operator: $\Delta_{\rho_\Xi} \boldsymbol{X}_{fjt}^{\Xi} = \boldsymbol{X}_{fjt}^{\Xi} - \rho_{\Xi} \boldsymbol{X}_{fjt-1}^{\Xi}$. Differencing the production function in (12) gives

$$\Delta_{\rho_\Xi} \xi_{jt} = \boldsymbol{\beta}_{\Xi}' \Delta_{\rho_\Xi} \boldsymbol{X}_{fjt}^{\Xi} + \Delta_{\rho_\Xi} \overline{\omega}_{m(j)t}^{\Xi} + \nu_{jt}^{\Xi} + \Delta_{\rho_\Xi} \varepsilon_{jt}^{\Xi}. \tag{J6}$$

For estimation, I concentrate out $\boldsymbol{\beta}_{\Xi}$ and market-year fixed effects, optimizing a GMM objective over only ρ_{Ξ}:

$$\hat{\rho}_{\Xi 2} = \underset{\rho_\Xi}{\operatorname{argmin}}\, \boldsymbol{g}_{\Xi 2}(\rho_{\Xi})'\, \boldsymbol{W}_{\Xi 2}\, \boldsymbol{g}_{\Xi 2}(\rho_{\Xi}), \quad \boldsymbol{g}_{\Xi 2}(\rho_{\Xi}) = \hat{\mathbb{E}}\big[\hat{\nu}_{jt}^{\Xi 2}(\rho_{\Xi}) \cdot \boldsymbol{Z}_{fjt}^{\Xi 2}\big], \tag{J7}$$

I use two-step GMM with initial weighting matrix $\boldsymbol{W}_{\Xi 2} = \hat{\mathbb{C}}(\boldsymbol{Z}_{fjt}^{\Xi 2})^{-1}$. Instruments target $\boldsymbol{\beta}_{\Xi}$, ρ_{Ξ}, and each market-year fixed effect $\overline{\omega}_{mt}^{\Xi}$, respectively:

$$\boldsymbol{Z}_{fjt}^{\Xi 2} = \left[\boldsymbol{X}_{fjt-1}^{\Xi}, \boldsymbol{X}_{fjt-2}^{\Xi}, \hat{\xi}_{jt-1}, \hat{\xi}_{jt-2}, \left[\mathbb{1}_{m(j)=m, t=u} \right]_{m \in \mathcal{M}, u=2016,\ldots,2022} \right]'. \tag{J8}$$

For each guess of ρ_{Ξ}, I estimate the regression in (J6) with linear IV-GMM, using the same instruments and weighting matrix in (J7).[106] This regression delivers $\hat{\beta}_{\Xi 2}(\rho_{\Xi})$, which, given a ρ_{Ξ}, minimizes the GMM objective in (J7), along with a residual $\hat{\nu}_{jt}^{\Xi 2}(\rho_{\Xi})$ used to compute the objective.

"System GMM" Approach. For the Blundell and Bond (1998, 2000) column, I add a website fixed effect $\overline{\omega}_{j}^{\Xi}$ to my quality parameterization in (12). I maintain the same AR(1) assumption for ω_{jt}^{Ξ}, but impose Arellano and Bover's (1995) stationarity assumptions:

$$\mathbb{E}\big[\overline{\omega}_{j}^{\Xi} \,\big|\, \Delta \boldsymbol{Z}_{fjt}^{\Xi 2}\big] = 0, \tag{J9}$$

[106] Computationally, instead of including market-year indicators, I de-mean both sides of the regression and the instruments within market-year.

where Δ denotes the non-quasi difference operator: $\Delta \boldsymbol{Z}^{\Xi 2}_{fjt} = \boldsymbol{Z}^{\Xi 2}_{fjt} - \boldsymbol{Z}^{\Xi 2}_{fjt-1}$. Differencing the production function in (12) gives the following system:

$$\begin{bmatrix} \Delta\Delta_{\rho_\Xi}\xi_{jt} \\ \Delta_{\rho_\Xi}\xi_{jt} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\beta}'_\Xi \Delta\Delta_{\rho_\Xi} \boldsymbol{X}^\Xi_{fjt} \\ \boldsymbol{\beta}'_\Xi \Delta_{\rho_\Xi} \boldsymbol{X}^\Xi_{fjt} \end{bmatrix} + \begin{bmatrix} \Delta\Delta_{\rho_\Xi}\overline{\omega}^\Xi_{m(j)t} \\ \Delta_{\rho_\Xi}\overline{\omega}^\Xi_{m(j)t} \end{bmatrix} + \begin{bmatrix} \Delta\eta^\Xi_{jt} \\ \eta^\Xi_{jt} \end{bmatrix}, \tag{J10}$$

where $\eta^\Xi_{jt} = \Delta_{\rho_\Xi}\overline{\omega}^\Xi_j + \nu^\Xi_{jt} + \Delta\rho_\Xi \varepsilon^\Xi_{jt}$ is a composite error. The "system GMM" approach is similar to the case without website fixed effects:

$$\hat{\rho}_{\Xi 3} = \underset{\rho_\Xi}{\operatorname{argmin}}\, \boldsymbol{g}_{\Xi 3}(\rho_\Xi)' \boldsymbol{W}_{\Xi 3}\, \boldsymbol{g}_{\Xi 3}(\rho_\Xi), \quad \boldsymbol{g}_{\Xi 3}(\rho_\Xi) = \hat{\mathbb{E}} \begin{bmatrix} \Delta\hat{\eta}^\Xi_{jt}(\rho_\Xi) \cdot \boldsymbol{Z}^{\Xi 3}_{fjt} \\ \hat{\eta}^\Xi_{jt}(\rho_\Xi) \cdot \Delta\boldsymbol{Z}^{\Xi 3}_{fjt} \end{bmatrix}. \tag{J11}$$

I use two-step GMM with initial weighting matrix $\boldsymbol{W}_{\Xi 3} = \operatorname{diag}(\hat{\mathbb{C}}(\boldsymbol{Z}^{\Xi 3}_{fjt}), \hat{\mathbb{C}}(\Delta\boldsymbol{Z}^{\Xi 3}_{fjt}))^{-1}$. Second lags appear in differences of the instruments, which are

$$\boldsymbol{Z}^{\Xi 3}_{fjt} = \left[\boldsymbol{X}^\Xi_{fjt-1},\ \hat{\xi}_{jt-1},\ \left[\mathbb{1}_{m(j)=m, t=u} \right]_{m\in\mathcal{M}, u=2016,\ldots,2022} \right]'. \tag{J12}$$

For each guess of ρ_Ξ, I estimate the stacked regression in (J10) with linear IV-GMM, using the same instruments and weighting matrix.

K. Supply Estimation Details

In this appendix, I provide details on estimating the supply side of the model from Sections 5.3 to 5.5. I use the method of simulated moments (McFadden, 1989; Pakes and Pollard, 1989) and a full solution approach, simulating all firm-years' choices for each guess of the model's parameters. Computational details on solving firms' cost minimization and profit maximization problems are in Appendix L.

Here, I focus on my two-stage estimation procedure. In the first stage, I simulate firms' cost-minimizing choices and match moments from the observed data that target the model's parameters. This stage allows me to estimate all but two parameters, with moments perfectly matched during cost minimization. In the second stage, I estimate net marginal revenue from quality optimality conditions, simulate firms' profit-maximizing choices, and use the previously perfectly-matched moments to target the remaining parameters.

Targeted Moments. In Appendix Table K1, I report the moments targeted in both stages. Each moment is the difference between a statistic computed from the observed

data and its simulated counterpart. Observed statistics are averages over firm-years. The first year used to compute these statistics is 2015—one year after the start of the primary sample—because some statistics condition on lagged choices.

To ensure that outliers in my data do not drive the results, I log-transform the statistics before averaging across firms and years. After adding one, nearly all statistics are nonnegative, except those involving the logs of estimated hours of OSS investment. For these, I scale by $\underline{\ell}$, the smallest increment of OSS investment estimated in Appendix G, which I also use to bound quantities in Appendix L.

I report bootstrapped standard errors for the observed statistics in parentheses on the right. The sources of uncertainty include traffic estimation from Appendix C and OSS investment estimation from Appendix G. For each bootstrap sample, I re-sample firms before re-computing bootstrapped statistics.

Appendix Table K1: Targeted Moments for Supply-Side Estimation

$\hat{\mathbb{E}}[\cdots \mid f \in \mathcal{F}_t,\, t = 2015, \ldots, 2022]$	Average over firm-years	Observed Statistic	Simulated Statistic	
$\log\left(1 + \sum_{s \in \mathcal{OS}_t \cup \{f\}} \frac{\ell_{fst}}{L_{ft}} \cdots\right)$	Labor-weighted			
$\mathbb{1}_{s \in \mathcal{OS}_t}$	OSS indicator	0.00030	0.00031	(0.00009)
$\times \mathbb{1}_{f \notin \mathcal{F}_{t-1}}$	$\times$ Initial year indicator	0.000057	0.000042	(0.000015)
$\times \log L_{ft}$	$\times$ Log total labor	0.0019	0.0027	(0.0004)
$\hookrightarrow \times \mathbb{1}_{f \notin \mathcal{F}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.00033	0.00031	(0.00008)
$\times \log \sum_{\tau=t}^{t-1} L_{f\tau}$	$\times$ Log total labor, plus 1 year	0.0020	0.0028	(0.0004)
$\times \log \sum_{\tau=t}^{t-2} L_{f\tau}$	$\times$ Log total labor, plus 2 years	0.0021	0.0029	(0.0004)
$\times \log \sum_{\tau=1}^{t} L_{s\tau}/\underline{\ell}$	$\times$ Log OSS labor	0.0029	0.0042	(0.0006)
$\hookrightarrow \times \mathbb{1}_{f \notin \mathcal{F}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.00060	0.00056	(0.00014)
$\times \log \sum_{\tau=1}^{t-1} L_{s\tau}/\underline{\ell}$	$\times$ Log OSS labor, less 1 year	0.0029	0.0041	(0.0005)
$\hookrightarrow \times \mathbb{1}_{f \notin \mathcal{F}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.00058	0.00056	(0.00013)
$\times \log \sum_{\tau=1}^{t-2} L_{s\tau}/\underline{\ell}$	$\times$ Log OSS labor, less 2 years	0.0027	0.0041	(0.0005)
$\hookrightarrow \times \mathbb{1}_{f \notin \mathcal{F}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.00050	0.00055	(0.00012)
$\times \sum_{\tau=1}^{t} L_{c(f)s\tau} / \sum_{\tau=1}^{t} L_{s\tau}$	$\times$ Domestic share	0.000089	0.000079	(0.000030)
$\hookrightarrow \times \mathbb{1}_{f \notin \mathcal{F}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0000145	0.0000107	(0.0000056)
$\times \sum_{\tau=1}^{t-1} L_{c(f)s\tau} / \sum_{\tau=1}^{t-1} L_{s\tau}$	$\times$ Log OSS labor, less 1 year	0.000088	0.000080	(0.000029)
$\hookrightarrow \times \mathbb{1}_{f \notin \mathcal{F}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0000145	0.0000108	(0.0000058)
$\times \sum_{\tau=1}^{t-2} L_{c(f)s\tau} / \sum_{\tau=1}^{t-2} L_{s\tau}$	$\times$ Log OSS labor, less 2 years	0.000083	0.000081	(0.000027)
$\hookrightarrow \times \mathbb{1}_{f \notin \mathcal{F}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0000146	0.0000110	(0.0000058)
$\times \log \sum_{\tau=1}^{t} \ell_{fs\tau}/\underline{\ell}$	$\times$ Log own labor	0.00182	0.00236	(0.00048)

Continued on the next page.

Continued from the previous page.

		Observed	Simulated	
$\hookrightarrow \times \mathbb{1}_{f \notin \mathcal{F}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.000338	0.000260	(0.000105)
$\times \mathbb{1}_{\sum_{\tau=1}^{t-1} \ell_{fs\tau} > 0}$	$\times$ Any own labor, less 1 year	0.00146	0.00219	(0.00044)
$\hookrightarrow \times \log \sum_{\tau=1}^{t-1} \ell_{fs\tau} / \underline{\ell}$	$\hookrightarrow \times$ Log own labor, less 1 year	0.00146	0.00219	(0.00044)
$\times \mathbb{1}_{\sum_{\tau=1}^{t-2} \ell_{fs\tau} > 0}$	$\times$ Any own labor, less 2 years	0.00123	0.00197	(0.00041)
$\hookrightarrow \times \log \sum_{\tau=1}^{t-2} \ell_{fs\tau} / \underline{\ell}$	$\hookrightarrow \times$ Log own labor, less 2 years	0.00123	0.00197	(0.00041)
$\mathbb{1}_{\ell_{fst-1}=0}$	Switching indicator	0.000049	0.000008	(0.000005)
$\log\left(1 + \hat{\mathbb{S}}(\cdots \mid s \in \mathcal{OS}_t \cup \{f\})\right)$	Standard deviation over software			
ℓ_{fst} / L_{ft}	Labor share	0.000054	0.000046	(0.000018)
$\log\left(1 + \sum_{j \in \mathcal{J}_{ft}, s=s_{pjt}} \frac{Q_{jt}}{Q_{ft}} \cdots\right)$	Traffic-weighted, p = backend			
$\mathbb{1}_{s \in \mathcal{OS}_t}$	OSS indicator	0.087	0.109†	(0.004)
$\times \log L_{ft}$	$\times$ Log total labor	0.307	0.389†	(0.013)
$\times \log \sum_{\tau=1}^{t} L_{s\tau} / \underline{\ell}$	$\times$ Log OSS labor	0.343	0.356	(0.016)
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0464	0.0482	(0.0033)
$\times \log \sum_{\tau=1}^{t-1} L_{s\tau} / \underline{\ell}$	$\times$ Log OSS labor, less 1 year	0.339	0.355	(0.016)
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0459	0.0480	(0.0033)
$\times \log \sum_{\tau=1}^{t-2} L_{s\tau} / \underline{\ell}$	$\times$ Log OSS labor, less 2 years	0.323	0.352	(0.017)
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0439	0.0477	(0.0032)
$\times \sum_{\tau=1}^{t} L_{c(f)s\tau} / \sum_{\tau=1}^{t} L_{s\tau}$	$\times$ Domestic share	0.019	0.014	(0.001)
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0020	0.0014	(0.0002)
$\times \sum_{\tau=1}^{t-1} L_{c(f)s\tau} / \sum_{\tau=1}^{t-1} L_{s\tau}$	$\times$ Domestic share, less 1 year	0.020	0.014	(0.001)
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0020	0.0014	(0.0002)
$\times \sum_{\tau=1}^{t-2} L_{c(f)s\tau} / \sum_{\tau=1}^{t-2} L_{s\tau}$	$\times$ Domestic share, less 2 years	0.020	0.015	(0.002)
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0020	0.0014	(0.0002)
$\times \mathbb{1}_{\sum_{\tau=1}^{t} \ell_{fs\tau} > 0}$	$\times$ Any own labor	0.014	0.015	(0.001)
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0010	0.0015	(0.0002)
$\hookrightarrow \times \log \sum_{\tau=1}^{t} \ell_{fs\tau} / \underline{\ell}$	$\hookrightarrow \times$ Log own labor	0.031	0.048	(0.005)
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0023	0.0046	(0.0006)
$\times \mathbb{1}_{\sum_{\tau=1}^{t-1} \ell_{fs\tau} > 0}$	$\times$ Any own labor, less 1 year	0.012	0.015	(0.001)
$\hookrightarrow \times \log \sum_{\tau=1}^{t-1} \ell_{fs\tau} / \underline{\ell}$	$\hookrightarrow \times$ Log own labor, less 1 year	0.026	0.047	(0.004)
$\times \mathbb{1}_{\sum_{\tau=1}^{t-2} \ell_{fs\tau} > 0}$	$\times$ Any own labor, less 2 years	0.010	0.015	(0.001)
$\hookrightarrow \times \log \sum_{\tau=1}^{t-2} \ell_{fs\tau} / \underline{\ell}$	$\hookrightarrow \times$ Log own labor, less 2 years	0.021	0.043	(0.004)
$\mathbb{1}_{s \notin S_{jt-1}}$	Switching indicator	0.028	0.039	(0.001)
$\log\left(1 + \sum_{j \in \mathcal{J}_{ft}, s=s_{pjt}} \frac{Q_{jt}}{Q_{ft}} \cdots\right)$	Traffic-weighted, p = JavaScript			

Continued on the next page.

Continued from the previous page.

		Observed	Simulated	
$\mathbb{1}_{s\in\mathcal{OS}_t}$	OSS indicator	0.074	0.035†	(0.003)
$\times \log L_{ft}$	× Log total labor	0.259	0.124†	(0.009)
$\times \log \sum_{\tau=1}^{t} L_{s\tau}/\underline{\ell}$	× Log OSS labor	0.294	0.299	(0.013)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0410	0.0417	(0.0027)
$\times \log \sum_{\tau=1}^{t-1} L_{s\tau}/\underline{\ell}$	× Log OSS labor, less 1 year	0.292	0.298	(0.013)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0407	0.0415	(0.0027)
$\times \log \sum_{\tau=1}^{t-2} L_{s\tau}/\underline{\ell}$	× Log OSS labor, less 2 years	0.289	0.296	(0.012)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0403	0.0412	(0.0026)
$\times \sum_{\tau=1}^{t} L_{c(f)s\tau}/\sum_{\tau=1}^{t} L_{s\tau}$	× Domestic share	0.015	0.011	(0.001)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0017	0.0013	(0.0002)
$\times \sum_{\tau=1}^{t-1} L_{c(f)s\tau}/\sum_{\tau=1}^{t-1} L_{s\tau}$	× Domestic share, less 1 year	0.015	0.011	(0.001)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0017	0.0013	(0.0002)
$\times \sum_{\tau=1}^{t-2} L_{c(f)s\tau}/\sum_{\tau=1}^{t-2} L_{s\tau}$	× Domestic share, less 2 years	0.016	0.011	(0.001)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0017	0.0013	(0.0002)
$\times \mathbb{1}_{\sum_{\tau=1}^{t} \ell_{fs\tau}>0}$	× Any own labor	0.030	0.014	(0.002)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0023	0.0022	(0.0003)
$\hookrightarrow \times \log \sum_{\tau=1}^{t} \ell_{fs\tau}/\underline{\ell}$	↪ × Log own labor	0.064	0.038	(0.011)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0049	0.0061	(0.0012)
$\times \mathbb{1}_{\sum_{\tau=1}^{t-1} \ell_{fs\tau}>0}$	× Any own labor, less 1 year	0.028	0.014	(0.002)
$\hookrightarrow \times \log \sum_{\tau=1}^{t-1} \ell_{fs\tau}/\underline{\ell}$	↪ × Log own labor, less 1 year	0.060	0.036	(0.011)
$\times \mathbb{1}_{\sum_{\tau=1}^{t-2} \ell_{fs\tau}>0}$	× Any own labor, less 2 years	0.025	0.013	(0.002)
$\hookrightarrow \times \log \sum_{\tau=1}^{t-2} \ell_{fs\tau}/\underline{\ell}$	↪ × Log own labor, less 2 years	0.053	0.034	(0.009)
$\mathbb{1}_{s\notin S_{jt-1}}$	Switching indicator	0.026	0.031	(0.001)
$\log\left(1+\sum_{j\in\mathcal{J}_{ft},s=s_{pjt}} \frac{Q_{jt}}{Q_{ft}}\cdots\right)$	Traffic-weighted, p = user interface			
$\mathbb{1}_{s\in\mathcal{OS}_t}$	OSS indicator	0.091	0.073†	(0.003)
$\times \log L_{ft}$	× Log total labor	0.318	0.261†	(0.011)
$\times \log \sum_{\tau=1}^{t} L_{s\tau}/\underline{\ell}$	× Log OSS labor	0.365	0.368	(0.016)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0567	0.0569	(0.0035)
$\times \log \sum_{\tau=1}^{t-1} L_{s\tau}/\underline{\ell}$	× Log OSS labor, less 1 year	0.364	0.367	(0.016)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0565	0.0567	(0.0035)
$\times \log \sum_{\tau=1}^{t-2} L_{s\tau}/\underline{\ell}$	× Log OSS labor, less 2 years	0.363	0.365	(0.016)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0563	0.0563	(0.0035)
$\times \sum_{\tau=1}^{t} L_{c(f)s\tau}/\sum_{\tau=1}^{t} L_{s\tau}$	× Domestic share	0.017	0.016	(0.001)
$\hookrightarrow \times \mathbb{1}_{j\notin\mathcal{J}_{t-1}}$	↪ × Initial year indicator	0.0019	0.0018	(0.0002)
$\times \sum_{\tau=1}^{t-1} L_{c(f)s\tau}/\sum_{\tau=1}^{t-1} L_{s\tau}$	× Domestic share, less 1 year	0.017	0.016	(0.001)

Continued on the next page.

Continued from the previous page.

		Observed	Simulated	
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0020	0.0018	(0.0002)
$\times \sum_{\tau=1}^{t-2} L_{c(f)s\tau} / \sum_{\tau=1}^{t-2} L_{s\tau}$	$\times$ Domestic share, less 2 years	0.017	0.016	(0.001)
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0020	0.0018	(0.0002)
$\times \mathbb{1}_{\sum_{\tau=1}^{t} \ell_{fs\tau} > 0}$	$\times$ Any own labor	0.024	0.019	(0.002)
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0024	0.0026	(0.0003)
$\hookrightarrow \times \log \sum_{\tau=1}^{t} \ell_{fs\tau} / \underline{\ell}$	$\hookrightarrow \times$ Log own labor	0.052	0.053	(0.008)
$\hookrightarrow \times \mathbb{1}_{j \notin \mathcal{J}_{t-1}}$	$\hookrightarrow \times$ Initial year indicator	0.0050	0.0070	(0.0011)
$\times \mathbb{1}_{\sum_{\tau=1}^{t-1} \ell_{fs\tau} > 0}$	$\times$ Any own labor, less 1 year	0.022	0.019	(0.002)
$\hookrightarrow \times \log \sum_{\tau=1}^{t-1} \ell_{fs\tau} / \underline{\ell}$	$\hookrightarrow \times$ Log own labor, less 1 year	0.049	0.051	(0.008)
$\times \mathbb{1}_{\sum_{\tau=1}^{t-2} \ell_{fs\tau} > 0}$	$\times$ Any own labor, less 2 years	0.020	0.019	(0.002)
$\hookrightarrow \times \log \sum_{\tau=1}^{t-2} \ell_{fs\tau} / \underline{\ell}$	$\hookrightarrow \times$ Log own labor, less 2 years	0.044	0.047	(0.007)
$\mathbb{1}_{s \notin S_{jt-1}}$	Switching indicator	0.032	0.039	(0.001)
$\hat{\mathbb{S}}(\cdots \mid s \in \mathcal{OS}_t \cup \{f\}, \cdots)$	Standard deviation over software			
$\sum_{j \in \mathcal{J}_{ft}, s = s_{jpt}} \frac{Q_{jt}}{Q_{ft}}$	Traffic share, $p =$ backend	0.2748	0.2743	(0.0002)
$\sum_{j \in \mathcal{J}_{ft}, s = s_{jpt}} \frac{Q_{jt}}{Q_{ft}}$	Traffic share, $p =$ JavaScript	0.3372	0.3366	(0.0002)
$\sum_{j \in \mathcal{J}_{ft}, s = s_{jpt}} \frac{Q_{jt}}{Q_{ft}}$	Traffic share, $p =$ user interface	0.3056	0.3050	(0.0002)

This table reports the moments targeted in supply-side estimation. Observed statistics are averages over firm-years, while simulated statistics are also averaged over unobservable draws. Moments without a † superscript are targeted during cost minimization estimation, while those with superscripts are targeted during profit maximization estimation. Due to the quality and total investment constraints, the latter are matched with zero error during cost minimization. The total labor of firm f is $L_{ft} = \ell_{fft} + \sum_{s \in \mathcal{OS}_t} \ell_{fst}$ and the labor of firms $f \in \mathcal{F}_c$ in country c on OSS $s \in \mathcal{OS}_t$ is $L_{cst} = \sum_{f \in \mathcal{F}_c} \ell_{fst}$. In parentheses, standard errors are from the same bootstrap procedure used to report uncertainty in Table 4.

Estimation Procedure. Let $\boldsymbol{\theta} = [\delta_O, \delta_P, \boldsymbol{\alpha}', \boldsymbol{\gamma}_L', \boldsymbol{\gamma}_S', \kappa_L, \kappa_S, \rho_L, \rho_S, \sigma_L, \sigma_S]'$. In the first stage (cost minimization) I optimize over the 20 parameters in $\boldsymbol{\theta}$ without † superscripts in Table 4, subject to the constraint that $\boldsymbol{\alpha}$ sums to one. Denote these parameters by $\boldsymbol{\theta}_1$. In the second stage (profit maximization) I optimize over the two parameters in $\boldsymbol{\theta}$ with † superscripts, denoted by $\boldsymbol{\theta}_2$. In each stage, I minimize the log of a minimum distance objective function:[107]

$$\hat{\boldsymbol{\theta}}_1 = \underset{\boldsymbol{\theta}_1}{\operatorname{argmin}} \log \Big(\boldsymbol{g}_1(\boldsymbol{\theta}_1)' \, \boldsymbol{W}_1 \, \boldsymbol{g}_1(\boldsymbol{\theta}_1) \Big) \quad \text{and} \quad \hat{\boldsymbol{\theta}}_2 = \underset{\boldsymbol{\theta}_2}{\operatorname{argmin}} \log \Big(\boldsymbol{g}_2(\boldsymbol{\theta}_2)' \, \boldsymbol{W}_2 \, \boldsymbol{g}_2(\boldsymbol{\theta}_2) \Big). \tag{K1}$$

[107] I find that minimizing in log space alleviates some of the numerical challenges discussed below. Since each estimator is overidentified, the argument of each logarithm is always positive.

Sample moments in $\boldsymbol{g}_1(\hat{\boldsymbol{\theta}}_1)$ and $\boldsymbol{g}_2(\hat{\boldsymbol{\theta}}_2)$ are differences between observed and simulated averages in Appendix Table K1 without and with † superscripts, respectively. Weighting matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are approximations of the optimal weighting matrices. The exception is when $\boldsymbol{\theta}_1$ leads to a very low average share of investment in OSS, as optimization (discussed below) can get stuck in a local optimum with near-zero OSS investment. To avoid this, I smoothly penalize the objective for $\boldsymbol{\theta}_1$ values that result in an average OSS investment share more than half a standard deviation below its observed value. This penalty is not binding at the global optimum.

Since there are far more elements in $\boldsymbol{W}_1$ than my 80 bootstrap samples, I do not attempt to estimate optimal weighting matrices by directly inverting bootstrapped covariance matrices. Instead, I use an approximation that accounts for the primary source of uncertainty—firm sampling—while also considering traffic and OSS investment uncertainty in how these scale into covariances. Specifically, at the point estimates of traffic and OSS investment, I compute the correlation matrix of the statistics in Appendix Table K1, clustering by firm. For each stage, I extract the relevant sub-matrix of correlations, invert it, and scale it into covariances using the bootstrapped standard deviations from Appendix Table K1.

Observed statistics are averages over firm-years, while simulated statistics are also averaged over 10 histories of wage unobservables ω^L_{fst} and ω^S_{jst} drawn according to (17) and (18). For each set of wage unobservables and firm-year in my final sample, I solve the cost minimization problem in (16) or, for the second stage, the profit maximization problem in (19).

Because of spillovers, multiple equilibria are possible. During estimation, I select the observed equilibrium by conditioning on the spillovers observed in the data. Specifically, after solving for firms' choices, instead of updating overall OSS capital K_{st} and country-specific contributions K_{cst} based on the new choices, I fix both to their observed values, conditional on the current guess of the OSS capital depreciation rate δ_O. Similarly, in the second stage, I also fix consumers' inclusive values to their observed values, conditional on the estimated demand system in Appendix I. I relax both of these restrictions for the policy simulations in Section 7.

Optimization. A challenge with optimizing the objectives in (K1) is that they involve thousands of nested optimization problems, one for each set of unobservables and firm-year. In Appendix L, I describe how I make solving these problems computationally feasible. Despite using log-transformed matched statistics, tight termination tolerances, and slight smoothing of discrete jumps, the large number of nested problems still introduces noise that

can stall traditional optimizers. When far from the optimum, optimization failure rates can reach up to 15%, but when closer to the optimum, failure rates drop to below 0.1%. Instead of starting with a classical optimizer, which would struggle with noise from relatively high failure rates, I use a method from the machine learning literature, where optimizing noisy loss functions is well-researched.

I begin with Bayesian optimization, a derivative-free global approach. I optimize the first-stage objective function in (K1) using Meta's OSS framework Ax (Bakshy et al., 2018), which relies on BoTorch (Balandat et al., 2019) for Bayesian optimization. Ax constructs a surrogate Gaussian Process model from the evaluated objectives and uses an acquisition function to select the next best parameter values, balancing exploration and exploitation.[108]

The first stage (cost minimization) is challenging, as it requires optimizing 19 parameters. I begin with 10 sets of 50 random initial evaluations within wide bounds, followed by 150 Bayesian iterations for each set. After global optimization, I fine-tune with a round of local optimization starting from the parameters with the lowest objective across all Bayesian evaluations. Near the global optimum, a maximum of 50 iterations using SciPy's gradient-based trust-region algorithm efficiently fine-tunes the objective.[109]

The second stage (profit maximization) is much simpler, involving just two parameters. Instead of Bayesian optimization, which requires local fine-tuning, I use more iterations of a deterministic global optimization algorithm. Specifically, I find that 100 objective evaluations using the locally-biased DIRECT algorithm for global optimization works well (Jones, Perttunen, and Stuckman, 1993; Gablonsky and Kelley, 2001).

Both objectives must also be optimized for each bootstrap sample. For each bootstrapped first stage of estimation, I begin local optimization from the point estimates. Since profit maximization involves discrete quality choices, which are difficult to smooth, gradient-based optimization is less effective. For each bootstrapped second stage, I use the same Bayesian optimization configuration that I used to obtain the point estimates.

Net Marginal Revenue. After cost minimization and before profit maximization, I estimate marginal revenue net of any marginal costs. Since on the margin, each firm-year in my sample affects the quality of its websites through a single continuous choice of total labor

[108] In my case, I find that the Log Noisy Expected Improvement (LogNEI) acquisition function (Daulton et al., 2023) performs better and is considerably cheaper to compute than the Knowledge Gradient (KG) acquisition function (Frazier et al., 2008), a popular alternative.

[109] I compute automatic gradients for the log of the cost minimization objective function in (K1) with Google's OSS framework JAX (Bradbury et al., 2018) and use the implicit function theorem to differentiate through nested optimization (Blondel et al., 2022).

L_{ft}, this implies a single optimality condition that I use in (20) to estimate a common net marginal revenue R_{ft} for each firm-year's websites. Specifically, I evaluate (20) for each of the 10 sets of cost unobservables used during estimation, and my estimator $\hat{R}_{ft}$ is the average across these sets.

There are four derivatives in (20). The first is $\partial Q_{ft}/\partial \boldsymbol{\xi}_{ft}$, where $Q_{ft} = \sum_{j \in \mathcal{J}_{ft}} Q_{jt}$ is the firm's overall traffic, and $\partial \boldsymbol{\xi}_{ft}$ denotes a marginal change in all of its websites' qualities at once.[110] I compute this first derivative by automatically differentiating the logit demand system in (14).

The second two are $\partial C_{ft}/\partial \boldsymbol{\xi}_{ft}$ and $\partial C_{ft}/\partial L_{ft}$, where C_{ft} is the minimum cost in (16). I compute the former using the implicit function theorem to differentiate through the cost minimization problem (Blondel et al., 2022). With L_{ft} fixed, quality affects costs only through the traffic shares in (10), making this term relatively unimportant. More important is the latter derivative, which, as discussed in the text, equals the average wage C_{ft}/L_{ft} multiplied by $1 + 1/\eta$, where I use an averaged residual labor supply elasticity of $\eta = 4.7$ from Roussille and Scuderi (2024).

The final derivative is $\partial L_{ft}/\partial \boldsymbol{\xi}_{ft}$, the marginal change in total labor required to achieve a marginal increase in the firm's website qualities. I compute it by implicitly differentiating the quality production function in (12): $\partial L_{ft}/\partial \boldsymbol{\xi}_{ft} = K_{ft}/\hat{\beta}_P$ because $\partial K_{ft}/\partial L_{ft} = 1$.

L. Computational Details

In this appendix, I provide computational details on how I simulate data from the model. Since I use a full solution approach for estimation, the simulation procedure is nearly identical for both estimation in Sections 5 and 6 and the policy simulations in Section 7. The key difference is that I select the observed equilibrium during estimation but fully solve for equilibria during policy simulations. I automate differentiation, compilation, and parallelization of the simulation procedure using Google's OSS framework JAX (Bradbury et al., 2018).

Simulation Procedure. The simulation procedure is a nested loop over (i) histories of simulated cost unobservables, (ii) years, (iii) equilibrium iterations, and (iv) firms. For the long-run simulations in Section 7, there is only one year, while for estimation in Sections 5 and 6, there is no equilibrium iteration. The computationally demanding part of this process is (iv), as it involves solving one or more nonlinear optimization problems and must be

[110] For single-website firms, $\partial \boldsymbol{\xi}_{ft} = \partial \xi_{jt}$ denotes a simple partial derivative. For firms with multiple websites, it is the derivative with respect to x where each ξ_{jt} in $\boldsymbol{\xi}_{ft}$ is replaced by $x_{jt} + x$, evaluated at $x = 0$.

repeated many thousands of times.

The first year of simulated data is either 2014, or, for long-run simulations, 2022. In this year t, I draw unobservables ω^L_{fst} and ω^S_{jst} from their stationary distributions in (17) and (18). In each subsequent year t, I update unobservables with innovations ν^L_{fst} and ν^S_{jst} drawn from their respective distributions. I also update private capital K_{ft}, OSS capital K_{st}, country-specific components K_{cst}, and firm-specific components K_{fst} according to (3) to (7). Since my ability to adjust for LinkedIn penetration rates improves in 2012, two years before my sample starts, I only accumulate private capital up to a limited horizon.[111]

For the policy simulations, I fully solve for each year's equilibrium by iterating on best responses, starting with those in the data. Given data for year t, I compute firms' choices conditional on others' and update the simulated data with their new choices. I update capital K_{ft}, K_{st}, K_{cst}, and K_{fst} according to (3) to (7), and, when choices include quality, traffic Q_{jt} according to (14). In practice, I find that because of switching costs, three iterations are sufficient for the dynamic simulations in Figure 7, and without switching costs, five iterations are sufficient for the long-run simulations in Tables 5 and 6.

During estimation, I compute firms' best responses to the observed data but perform a partial update to select the observed equilibrium. Specifically, after computing firms' choices conditional on the observed choices of others, I do not update total OSS capital K_{st} or its country-specific components K_{cst}. I hold these fixed at their observed values, which are a function of observed investment and the OSS capital depreciation rate δ_O.

Firm f's choices include software $\boldsymbol{s}_{ft}$, labor $\boldsymbol{\ell}_{ft}$, and, when maximizing profit, quality $\boldsymbol{\xi}_{ft}$. I first describe how I solve the firm's cost minimization problem, followed by how I solve its profit maximization problem.

Cost Minimization. The cost minimization problem in (16) over the firm's software choices $\boldsymbol{s}_{ft} = \{s_{jpt}\}_{j\in\mathcal{J}_{ft},p\in\mathcal{P}}$ and labor choices $\boldsymbol{\ell}_{ft} = \{\ell_{fst}\}_{s\in\mathcal{OS}_t\cup\{f\}}$ is subject to two constraints. First, I fix total labor $L_{ft} = \ell_{fft} + \sum_{s\in\mathcal{OS}_t}\ell_{fst}$. Second, given my parameterization of the quality production function $\Xi_{fjt}(\boldsymbol{s}_{jt}, \boldsymbol{K}_t)$ in (12), the quality constraint determines the choice between in-house and OSS for each part of each of the firm's websites.

For clarity, consider website j operated by firm f. If, in the observed data, the software $s_{jpt} \in \mathcal{OS}_t$ is OSS, then when choosing software for the same website and part, the firm must choose another OSS option in $\mathcal{OS}_t$. Conversely, if $s_{jpt} = f$ is in-house, the firm must continue choosing in-house software for that website's part under the quality constraint.

[111] Before 2012, the BLS does not separately report employment for web developers. Since my sample begins in 2014, I replace (3) with $K_{ft} = \sum_{\tau=t}^{t-2}(1-\delta_P)^{t-\tau}L_{ft}$. I estimate a high δ_P, so this ends up mattering little.

Given my wage parameterization in (8) and subject to the two constraints, an equivalent but more numerically stable version of the cost minimization problem is[112]

$$\min_{\boldsymbol{s}_{ft},\boldsymbol{\ell}_{ft}} \left\{ \log\left(\sum_{s\in\mathcal{OS}_t\cup\{f\}} \frac{\ell_{fst}}{L_{ft}} \cdot \exp\left(\text{Dev}_{fst}(\boldsymbol{\ell}_{ft},\boldsymbol{K}_t)\right)\right) + \text{Ops}_{ft}(\boldsymbol{s}_{ft},\boldsymbol{K}_t)\right\}. \tag{L1}$$

The separability of the development and operations components is convenient because it allows the software choice to be concentrated out. Since the operations component can be expressed as a traffic-weighted average over the firm's websites,[113]

$$\text{Ops}_{ft}(\boldsymbol{s}_{ft},\boldsymbol{K}_t) = \sum_{j\in\mathcal{J}_{ft}} \frac{Q_{jt}}{Q_{ft}} \cdot \text{Ops}_{fjt}(\boldsymbol{s}_{jt},\boldsymbol{K}_t), \tag{L2}$$

it suffices to, for a guess of labor $\boldsymbol{\ell}_{ft}$ and hence capital $\boldsymbol{K}_t$, find the software $\boldsymbol{s}_{jt}$ that minimizes the above website-specific operations component $\text{Ops}_{fjt}(\boldsymbol{s}_{jt},\boldsymbol{K}_t)$. Notably, solving for $\boldsymbol{s}_{jt}$ can be done *independently* for each website. Conditional on labor, this problem does not become exponentially more complex for firms with many websites.

In practice, I optimize OSS investment $\{\ell_{fst}\}_{s\in\mathcal{OS}_t}$, set $\ell_{fft} = L_{ft} - \sum_{s\in\mathcal{OS}_t}\ell_{fst}$, and concentrate out choice of software. To avoid optimization issues, I impose box constraints $\ell_{fst} \in \left[0,\, \min\left\{\bar{\ell},\, L_{ft}\times\tilde{\ell}\right\}\right]$, where $\bar{\ell}$ is the 99th percentile of observed nonzero investment ℓ_{fst} in $s\in\mathcal{OS}_t$, and $\tilde{\ell}$ is the 99th percentile of observed nonzero shares of investment ℓ_{fst}/L_{ft} in $s\in\mathcal{OS}_t$. I enforce these bounds by optimizing ℓ_{fst} in logit space.

For the first year and equilibrium iteration, I initialize each $\ell_{fst} = \underline{\ell}/2$, where $\underline{\ell}$ is the smallest increment of OSS investment estimated in Appendix G. After optimization, I retain each optimized ℓ_{fst} for "hot starts" in subsequent equilibrium iterations and years. If optimization is successful, I use the optimized OSS investment as a starting point for the next equilibrium iteration. After solving for an equilibrium, I use the equilibrium OSS investment from successful optimization as a starting point for the next year, if the firm remains in the sample. Otherwise, I revert to $\ell_{fst} = \underline{\ell}/2$.

I optimize the objective in (L1) using a JAX-compiled version of the Newton-CG trust-region algorithm (7.2) from Nocedal and Wright (2006). I also use JAX to automatically compute the gradient and Hessian-vector products for the objective, which are inputs for the optimization algorithm. SciPy's default optimization hyperparameters work well in my

[112] Investment incentives for the policy simulations in Section 7 enter additively within the log in the development component.

[113] $\text{Ops}_{fjt}(\boldsymbol{s}_{jt},\boldsymbol{K}_t) = \sum_{p\in\mathcal{P}}\alpha_p\text{Ops}_{fjpt}(s_{jpt},\boldsymbol{K}_t)$. The quality constraint also fixes traffic shares Q_{jt}/Q_{ft} in the operations parameterization; conditional on quality, traffic depends on neither software nor labor.

setting. Maximums of 500 outer and CG iterations are rarely binding, and I can impose tight radius- and gradient-based tolerances of 10^{-12}. The main challenge in optimization comes from discrete jumps in the model, requiring minor smoothing to ensure progress.

Smoothing. First, software is discrete. A zero gradient of the operations component with respect to labor poses challenges for the optimizer, which expects a smooth objective. Instead of representing $\boldsymbol{s}_{jt}$ with binary indicators for whether each $s \in \mathcal{OS}_t \cup \{f\}$ satisfies the quality constraint and minimizes $\text{Ops}_{fjt}(\boldsymbol{s}_{jt}, \boldsymbol{K}_t)$ in (L2), I pass values of $-\text{Ops}_{fjt}(\boldsymbol{s}_{jt}, \boldsymbol{K}_t)$ through the softmax function

$$\text{softmax}_i(x) = \frac{\exp(x_i/\lambda_S)}{\sum_\iota \exp(x_\iota/\lambda_S)}, \tag{L3}$$

where λ_S is a smoothing parameter. Given the magnitudes of $\text{Ops}_{fjt}(\boldsymbol{s}_{jt}, \boldsymbol{K}_t)$ in my setting, I find that $\lambda_S = 0.1$ produces smoothed indicators that the optimizer can handle, while remaining close to their discrete counterparts.

Second, labor also has discrete features. OSS investment estimated in Appendix G is always at least its minimum estimated increment, $\ell_{fst} \geq \underline{\ell}$, except when it is zero, $\ell_{fst} = 0$. To capture this when simulating data, I use a modification of the smoothstep function:

$$\text{smoothstep}(x) = 3x^3 - 2x^3. \tag{L4}$$

I multiply each guess of ℓ_{fst} by the following modification, evaluated at ℓ_{fst}:

$$\text{softstep}(x) = \text{smoothstep}\left(\frac{\exp(\min\{\max\{x/\underline{\ell}, 0\}, 1\}/\lambda_L) - 1}{\exp(1/\lambda_L) - 1}\right). \tag{L5}$$

For $x \geq \underline{\ell}$, $\text{softstep}(x) = 1$, and as x approaches zero, so does $\text{softstep}(x)$. For smaller values of the smoothing parameter λ_L, $\text{softstep}(x)$ approaches a discrete step function around $\underline{\ell}$. Again, I find that $\lambda_L = 0.1$ results in simulated OSS investment that closely matches the discreteness in the data while still being smooth enough for the optimizer to handle.

Labor discreteness also affects capital. When computing the software characteristics $\boldsymbol{X}^W_{fst}(\boldsymbol{K}_t)$ in (11), I bound the OSS capital terms K_{st}, K_{cst}, and K_{fst} from below by $\underline{\ell}$. Without this, highly depreciated values could result in very negative logarithms. Specifically, when solving the firm's cost minimization problem, I replace $\log K_{st}$ and $\log K_{fst}$ in (11) with $\log(\max\{\underline{\ell}, K_{st}\})$ and $\log(\max\{\underline{\ell}, K_{fst}\})$, respectively. I replace $\mathbb{1}_{K_{fst}>0}$ with the previously discussed $\text{softstep}(\ell_{fst})$ when there were no prior firm-specific contributions (i.e., $K_{fst} = \ell_{fst}$). Similarly, I scale $K_{c(f)st}/K_{st}$ by $\text{softstep}(\ell_{fst})$ when there were no prior country-specific

contributions (i.e., $K_{c(f)st} = \ell_{fst}$).

Smoothing also improves supply estimation in Appendix K.[114] I compute matched statistics in Appendix Table K1 using the smoothed software and labor discussed earlier. I replace conditional logic regarding software in Appendix Table K1 with weighted averages across smoothed software indicators. I use the smoothed labor values, scaled by the softstep function when below $\underline{\ell}$. Additionally, I replace all log OSS labor terms with $\log(\max\{\underline{\ell}, \cdot\}) \times \text{softstep}(\cdot)$, I replace all indicators for OSS labor presence with softstep$(\cdot)$, and I scale each domestic share by softstep$(\cdot)$ evaluated at the numerator.

Profit Maximization. The profit maximization problem in (19) over $\boldsymbol{\xi}_{ft} = \{\xi_{jt}\}_{j \in \mathcal{J}_{ft}}$ is subject to the constraint that quality lies within the range of the production function $\Xi_{fjt}(\boldsymbol{s}_{jt}, \boldsymbol{K}_t)$ in (12). Since I fix total labor L_{ft} and thus private capital K_{ft}, this range consists of the qualities from each combination of in-house versus OSS—a binary choice—for the parts of the firms' websites.

Given my parameterization of the quality production function, the qualities from these combinations are distinct, so the choice of quality can be re-expressed as the choice of which combination of in-house versus OSS maximizes profit. For a given combination, it is straightforward to compute traffic $Q_{jt}(\boldsymbol{\xi}_t)$ in (14) for each of the firm's websites, scale total traffic $Q_{ft}(\boldsymbol{\xi}_t) = \sum_{j \in \mathcal{J}_{ft}} Q_{jt}(\boldsymbol{\xi}_t)$ by net marginal revenue $\hat{R}_{ft}$ estimated in Appendix K, and subtract costs evaluated at the optimal software and labor choices.

This problem becomes exponentially more difficult for firms with many websites, so I impose the constraint that for a given firm-year-part, the firm make the same binary choice of in-house versus OSS for all of its websites. Since there are $|\mathcal{P}| = 3$ parts, under this constraint I solve the cost minimization problem $2^3 = 8$ times, compute profit for each combination, and keep the result from the one with the highest profit.[115]

M. Intuition from Stylized Models

In this appendix, I use simple parameterizations of the model from Section 4 to illustrate the effects of government involvement in the private provision of OSS.

First, I consider the trade-off between in-house vs. open-access provision within a single

[114] Without smoothing and with a finite number of simulation draws, the first-stage objective function in (K1) features discrete jumps. This is less of a concern for global Bayesian optimization, which is robust to non-smooth objectives, but it does complicate local fine-tuning that uses gradients and performs better with smoother objective functions.

[115] I expect this restriction to have little practical impact; across firm-year-parts with multiple websites, 87% either only use in-house software or only OSS.

country. Second, I explore the international version of this trade-off: domestic vs. foreign provision. For each case, I first examine the effects of government involvement without competition, then introduce competition to illustrate the effects of business stealing.

In-House vs. Open-Access. There is one OSS, o, and two firms, $f \in \{a, b\}$. Each chooses to use $s_f \in \{o, f\}$ and whether to contribute to the OSS, $\ell_f \in \{0, 1\}$. Payoffs are

$$\Pi_f = \pi_O \cdot \underbrace{\mathbb{1}_{s_f=o}}_{\substack{\text{Use}\\ \text{OSS}}} + \pi_L \cdot \underbrace{\ell_f}_{\substack{\text{Invest}\\ \text{in OSS}}} + \pi_{\&} \cdot \underbrace{\mathbb{1}_{s_f=o} \cdot \ell_f}_{\substack{\text{Use \& invest}\\ \text{in OSS}}} + \pi_E \cdot \underbrace{\mathbb{1}_{s_f=o} \cdot \ell_{-f}}_{\substack{\text{Externality from}\\ \text{other's OSS investment}}}. \tag{M1}$$

Assume it is more profitable to use in-house software when the OSS has no contributions, $\pi_O < 0$, and that the cost of investing in OSS exceeds any direct benefits, $\pi_L < 0$. However, there is a private benefit from investing in OSS when it is also being used, $\pi_{\&} > 0$, and a public benefit from the other firm's investment, $\pi_E > 0$.

The payoff matrix is in Appendix Table M1. If using and investing in OSS is unprofitable without the externality, $\pi_O + \pi_L + \pi_{\&} < 0 < \pi_O + \pi_L + \pi_{\&} + \pi_E$, then there are either one or two pure strategy Nash equilibria. The top-left corner is an equilibrium where both firms use in-house software and neither invests in OSS. The bottom-right is an equilibrium where both firms use and invest in OSS, but only if the private benefits of investing in OSS are large enough, $\pi_L + \pi_{\&} > 0$. Otherwise, each firm has an incentive to free-ride.

Appendix Table M1: In-House vs. Open-Access

Π_f or (Π_a, Π_b)		$s_b = b$:		$s_b = o$:	
		$\ell_b = 0$	$\ell_b = 1$	$\ell_b = 0$	$\ell_b = 1$
$s_a = a$:	$\ell_a = 0$	0	$(0, \pi_L)$	$(0, \pi_O)$	$(0, \pi_O{+}\pi_L{+}\pi_{\&})$
	$\ell_a = 1$	$(\pi_L, 0)$	π_L	$(\pi_L, \pi_O{+}\pi_E)$	$(\pi_L, \pi_O{+}\pi_L{+}\pi_{\&}{+}\pi_E)$
$s_a = o$:	$\ell_a = 0$	$(\pi_O, 0)$	$(\pi_O{+}\pi_E, \pi_L)$	π_O	$(\pi_O{+}\pi_E, \pi_O{+}\pi_L{+}\pi_{\&})$
	$\ell_a = 1$	$(\pi_O{+}\pi_L{+}\pi_{\&}, 0)$	$(\pi_O{+}\pi_L{+}\pi_{\&}{+}\pi_E, \pi_L)$	$(\pi_O{+}\pi_L{+}\pi_{\&}, \pi_O{+}\pi_E)$	$\pi_O{+}\pi_L{+}\pi_{\&}{+}\pi_E$

Government subsidies that increase the private benefits of investing in OSS can be effective for two reasons. First, increasing π_L so that $\pi_L + \pi_{\&} > 0$ makes the payoff-maximizing bottom-right an equilibrium. Second, even if the bottom-right is an equilibrium, the status quo may be the top-left. Temporarily increasing π_L so that $\pi_L > 0$ may shift the equilib-

rium to the bottom-right. The economic impact of such government policies depends on the strength of *private* incentives to invest in OSS and the *public* benefits of that investment.

Business Stealing. Payoffs in (M1) and Appendix Table M1 feature only one positive externality $\pi_E > 0$. Now, assume that both firms compete in the same market, which introduces the term $\pi_B \cdot \mathbb{1}_{s_{-f}=o} \cdot \ell_f$, representing a business-stealing externality π_B. This externality could be either positive or negative, depending on whether competitors' best responses to increased OSS investment are to reduce or improve the quality of their websites. The matrix of payoff changes is in Appendix Table M2.

Appendix Table M2: Business Stealing

$\Delta\Pi_f$ or $\Delta(\Pi_a, \Pi_b)$		$s_b = b$:		$s_b = o$:	
		$\ell_b = 0$	$\ell_b = 1$	$\ell_b = 0$	$\ell_b = 1$
$s_a = a$:	$\ell_a = 0$	0	0	0	0
	$\ell_a = 1$	0	0	0	0
$s_a = o$:	$\ell_a = 0$	0	0	0	$(0, \pi_B)$
	$\ell_a = 1$	0	0	$(\pi_B, 0)$	π_B

With business stealing, the bottom-right remains an equilibrium only if private benefits of investing in OSS are even larger or smaller, $\pi_L + \pi_\& + \pi_B > 0$, depending on the sign of π_B. For subsidies to be effective in a competitive environment, they must offset this additional externality, which scales with the intensity of competition and with how much OSS affects website quality.

Domestic vs. Foreign. Assume there are four firms, two in each country $c(f) \in \{U, C\}$. All four use OSS, but there are now two to choose from, o_U and o_C. Each firm chooses to use $s_f \in \{o_U, o_C\}$ and whether to contribute to it, $\ell_f \in \{0, 1\}$. Payoffs are

$$\Pi_f = \pi_D \cdot \underbrace{\mathbb{1}_{s_f = o_{c(f)}}}_{\substack{\text{Use} \\ \text{domestic OSS}}} + \pi_L \cdot \underbrace{\ell_f}_{\substack{\text{Use \& invest} \\ \text{in OSS}}} + \pi_\& \cdot \underbrace{\mathbb{1}_{s_f = o_{c(f)}} \cdot \ell_f}_{\substack{\text{Use \& invest} \\ \text{in domestic OSS}}} + \sum_{-f \neq f} \pi_{E,-f \to f} \cdot \underbrace{\mathbb{1}_{s_f = s_{-f}} \cdot \ell_{-f}}_{\substack{\text{Externality from} \\ \text{others' OSS investment}}}. \quad \text{(M2)}$$

Assume it is more profitable to use domestic OSS when there are no contributions, $\pi_D > 0$, and that the cost of investing in OSS outweighs any direct benefits, $\pi_L < 0$. However, there

is a private benefit from investing in domestic OSS, $\pi_{\&} > 0$, and a public benefit from other firms' investments, $\pi_{E,-f\to f} = \pi_E \cdot \mathbb{1}_{c(-f)=c(f)} + \pi'_E \cdot \mathbb{1}_{c(-f)\neq c(f)}$, where the externality $\pi_E > \pi'_E > 0$ is stronger when contributions come from the same country. In this case, public benefits are *localized*.

I consider pure strategy Nash equilibria where firms in the same country make the same choices, $s_f = s_{c(f)}$ and $\ell_f = \ell_{c(f)}$, and hence have the same payoffs, $\Pi_f = \Pi_{c(f)}$. The payoff matrix is in Appendix Table M3. Identifying equilibria requires considering payoffs from unilateral deviations not shown in the table.

Appendix Table M3: Domestic vs. Foreign

Π_f or (Π_U, Π_C)		$s_C = o_C$:		$s_C = o_U$:	
		$\ell_C = 0$	$\ell_C = 1$	$\ell_C = 0$	$\ell_C = 1$
$s_U = o_U$:	$\ell_U = 0$	π_D	$(\pi_D, \pi_D{+}\pi_L{+}\pi_{\&}{+}\pi_E)$	$(\pi_D, 0)$	$(\pi_D{+}\pi_E, \pi_L{+}\pi_E)$
	$\ell_U = 1$	$(\pi_D{+}\pi_L{+}\pi_{\&}{+}\pi_E, \pi_D)$	$\pi_D{+}\pi_L{+}\pi_{\&}{+}\pi_E$	$(\pi_D{+}\pi_L{+}\pi_{\&}{+}\pi_E, 2\pi'_E)$	$(\pi_D{+}\pi_L{+}\pi_{\&}{+}\pi_E{+}2\pi'_E, \pi_L{+}\pi_E{+}2\pi'_E)$
$s_U = o_C$:	$\ell_U = 0$	$(0, \pi_D)$	$(2\pi'_E, \pi_D{+}\pi_L{+}\pi_{\&}{+}\pi_E)$	0	$(0, \pi_L{+}\pi_E)$
	$\ell_U = 1$	$(\pi_L{+}\pi_E, \pi_D{+}\pi_E)$	$(\pi_L{+}\pi_E{+}2\pi'_E, \pi_D{+}\pi_L{+}\pi_{\&}{+}\pi_E{+}2\pi'_E)$	$(\pi_L{+}\pi_E, 0)$	$\pi_L{+}\pi_E$

If the private benefit from domestic investment exceeds the direct costs, $\pi_L + \pi_{\&} > 0$, but public benefits from foreign investment are not too strong, $2\pi'_E < \pi_D + \pi_L + \pi_{\&} + \pi_E$, then there are either one or two equilibria. If the externality from foreign investment is relatively weak, $2\pi'_E < \pi_D + \pi_L + \pi_{\&}$, the only equilibrium is "decoupled," meaning each country builds its own OSS. Otherwise, two equilibria exist, where one country builds OSS while the other free-rides. Without interventions like subsidies to π_L as discussed earlier, it is not an equilibrium for both countries to invest in and use the same OSS.

Domestic-focused government policy can be effective when two equilibria exist. In the web development industry, the equilibrium closest to the current status quo may be one where the US builds its own OSS while China free-rides. Temporary subsidies or restrictions that encourage Chinese firms to use and invest in Chinese OSS could shift the equilibrium to the decoupled one and eventually to a China-preferred equilibrium, where China builds its own OSS and the US free-rides. The economic impact of such policies depends on the

strength of private incentives to invest in domestic vs. foreign OSS and the *localization* of resulting public benefits investment.

International Business Stealing. Again, payoffs in (M2) and Appendix Table M3 feature only positive externalities. Now, assume all firms compete in the same market, which adds $\sum_{-f \neq f} \pi_{B,-f \to f} \cdot \mathbb{1}_{s_{-f}=s_f} \cdot \ell_f$, with business stealing $\pi_{B,-f \to f} = \pi_B \cdot \mathbb{1}_{c(-f)=c(f)} + \pi'_B \cdot \mathbb{1}_{c(-f)\neq c(f)}$. We expect this effect to be stronger within a country, $|\pi_B| > |\pi'_B|$, due to "home bias." The matrix of payoff changes is in Appendix Table M4.

Appendix Table M4: International Business Stealing

$\Delta\Pi_f$ or $\Delta(\Pi_U, \Pi_C)$		$s_C = o_C$:		$s_C = o_U$:	
		$\ell_C = 0$	$\ell_C = 1$	$\ell_C = 0$	$\ell_C = 1$
$s_U = o_U$:	$\ell_U = 0$	0	$(0, \pi_B)$	0	$(0, \pi_B{+}2\pi'_B)$
	$\ell_U = 1$	$(\pi_B, 0)$	π_B	$(\pi_B{+}2\pi'_B, 0)$	$\pi_B{+}2\pi'_B$
$s_U = o_C$:	$\ell_U = 0$	0	$(0, \pi_B{+}2\pi'_B)$	0	$(0, \pi_B)$
	$\ell_U = 1$	$(\pi_B{+}2\pi'_B, 0)$	$\pi_B{+}2\pi'_B$	$(\pi_B, 0)$	π_B

With negative business stealing $\pi_B < \pi'_B < 0$, firms are more likely to deviate from the equilibrium where each country builds its own OSS because incentives to free-ride are stronger. The opposite holds if business stealing is positive. Temporary subsidies or restrictions that encourage Chinese firms to use and invest in Chinese OSS must overcome this additional externality from business stealing within China. Similarly, subsidies would need to be either larger or smaller to maintain an equilibrium where both countries use and invest in the same OSS. The required change in subsidies depends on the intensity of domestic and foreign competition.

N. Additional Policy Simulations

In this appendix, I show how OSS policy outcomes vary with policy details and the model's empirical content by rerunning the simulations from Section 7 using different policy and model parameters.

Stronger Localized Benefits. In column (i) of Appendix Table N1, identical to column (i) in Table 5, simulated Chinese restrictions on foreign collaboration prove ineffective at

promoting investment into domestic OSS. Given the model's estimates, China's coordination problem is not severe enough for penalizing foreign collaboration to significantly shift investment from foreign-built to domestic OSS.

In column (*ii*) of Appendix Table N1, I rerun the same simulation with increased localized public benefits from OSS investment in China. I scale the estimates in $\hat{\boldsymbol{\gamma}}_S$ on $\log K_{st}$ and $K_{c(f)st}/K_{st}$ by five for Chinese firms, increasing public benefits and their localization. With a more significant coordination problem, restrictions are more likely to increase Chinese OSS investment and even lower domestic costs.

Stronger Private Benefits. In column (*iii*) of Appendix Table N1, identical to column (*iv*) of Table 5, a global subsidy for OSS investment is highly effective at lowering costs. Since the private benefits of OSS investment are much smaller than the public benefits, subsidies help reduce underprovision by closing the gap between them.

In column (*iv*) of Appendix Table N1, I rerun the simulation after increasing the private benefits from OSS investment for all firms. I scale the estimate in $\hat{\boldsymbol{\gamma}}_S$ on $\log K_{fst}$ by five, increasing the private benefits from investing in OSS. With public benefits unchanged, the gap between public and private benefits shrinks, making subsidies less effective. Each dollar subsidized reduces costs by around four times less.

Usage-Weighted Subsidies. In columns (*i*) and (*iii*) of Appendix Table N2, identical to columns (*iii*) and (*iv*) of Table 5, subsidies for OSS investment effectively lower costs. On the left, US-focused subsidies are particularly cost-effective. Since firms disproportionately use US-built OSS, subsidizing investment in these OSS leads to larger global cost reductions compared to investing in OSS built elsewhere.

A natural question is whether targeting OSS with higher usage could further improve subsidy cost-effectiveness. In column (*ii*) of Appendix Table N2, instead of weighting by domestic capital shares, I instead weight by the share of usage for each OSS, across all website parts in the sample. Cost reductions per dollar subsidized are about 30% higher, except for China. In column (*iv*), weighting global subsidies by usage increases cost-effectiveness by about 80% globally.

Different Subsidy Sizes. In column (*ii*) of Appendix Tables N3 and N4, identical to columns (*iii*) and (*iv*) of Table 5, subsidies of \$100 per hour of investment are fairly cost-effective. A second natural question is how cost-effectiveness changes with the size of the subsidy.

In columns (i) and (iii) of Appendix Tables N3 and N4, I instead consider per-hour subsidies of \$50 and \$150 that are 50% lower and higher than the baseline of \$100, respectively. Cost effectiveness decreases with the size of the subsidy, suggesting that those OSS with the highest private benefits of investment—which are affected first by smaller subsidies—are, in general, those that also generate the greatest public benefits. This is also why the decline in cost-effectiveness is steeper for the global subsidies: as discussed above, the domestic subsidies already do much of the work to select OSS with large public benefits because these OSS tend to be made in the US.

Business Stealing. In column (i) of Appendix Table N5, identical to column (iv) in Table 6, policy-driven profit changes are driven by cost reductions, and the confidence interval for the change in consumer surplus includes zero. The estimated effect of OSS on website quality, $\hat{\beta}_S$, is too weak for business stealing concerns to be a first-order issue.

In column (ii) of Appendix Table N5, I rerun the same simulation after increasing the amount of business stealing. I set the impact of OSS on quality to $\hat{\beta}_S = \hat{\beta}_{V1} > 0$, the strong domestic preference for non-US and non-Chinese consumers. Policy-driven increases in OSS investment decline as firms internalize that their investment will lead competitors to use more OSS and steal profits. Business stealing concerns mitigate cost reductions and nearly eliminate profit gains. However, since consumers now prefer OSS, they are much better off.

"Foreign Bias". In column (iii) of Appendix Table N5, I instead replace consumer "home bias" with "foreign bias." Specifically, I flip the signs in $\hat{\boldsymbol{\beta}}_V$. Policy-driven changes in OSS investment and firm costs remain similar to the case with home bias, but both increases in profit and decreases in consumer surplus are attenuated. With more competition, firms are less able to compound their lower costs into greater profit, and consumers are better able to substitute away from websites with reduced quality.

Appendix Table N1: Cost Minimization with Stronger Localized and Private Benefits

	Stronger localized public benefits for all Chinese firms		Stronger private benefits for all firms	
	(*i*) Unchanged	(*ii*) Stronger	(*iii*) Unchanged	(*iv*) Stronger
Restrict investment in *foreign* web OSS	China	China		
Subsidize investment in *all* web OSS			Global	Global
Investment in web OSS	−0.2%	−0.3%	+57.9%	+13.4%
	[−0.5, −0.0]	[−0.5, −0.1]	[+49.9, +63.9]	[+12.5, +14.9]
↪ US	+0.0%	+0.0%	+39.8%	+8.6%
	[−0.1, +0.1]	[−0.1, +0.0]	[+35.7, +41.8]	[+8.2, +9.0]
↪ China	−3.1%	−5.3%	+57.4%	+16.6%
	[−5.1, −1.4]	[−6.6, −2.5]	[+42.6, +63.6]	[+15.6, +18.1]
Domestic web OSS investment	−0.0%	+0.1%	+44.8%	+8.9%
	[−0.1, +0.1]	[−0.1, +0.2]	[+40.1, +46.6]	[+8.5, +9.4]
↪ US	+0.0%	+0.0%	+38.6%	+8.1%
	[−0.0, +0.1]	[−0.1, +0.1]	[+34.4, +40.2]	[+7.7, +8.6]
↪ China	−0.4%	+0.1%	+65.2%	+10.7%
	[−1.0, +0.5]	[−0.7, +1.0]	[+60.1, +72.9]	[+9.6, +11.9]
Investment incentives, millions			+$6.2	+$11.2
			[+5.1, +7.1]	[+10.3, +12.0]
↪ US			+$2.6	+$5.5
			[+2.1, +2.8]	[+5.2, +5.8]
↪ China	−$0.1	−$0.1	+$0.5	+$0.8
	[−0.1, −0.1]	[−0.1, −0.1]	[+0.4, +0.6]	[+0.8, +0.9]
Firm costs, per dollar of incentive	+$7.2	−$0.6	−$26.1	−$6.4
	[+1.6, +16.4]	[−11.9, +13.6]	[−28.3, −25.1]	[−7.4, −5.6]
↪ US	+$4.2	+$7.3	−$20.9	−$4.2
	[+0.3, +11.5]	[−4.2, +13.9]	[−22.4, −19.8]	[−4.7, −3.7]
↪ China	+$1.8	−$2.4	−$1.4	−$0.4
	[+1.3, +2.5]	[−8.7, +1.0]	[−1.5, −1.3]	[−0.5, −0.3]

Column (*i*) is identical to column (*i*) in Table 5, while column (*ii*) in this table is the same but scales the estimates in $\hat{\gamma}_S$ on $\log K_{st}$ and $K_{c(f)st}/K_{st}$ by five for Chinese firms. Column (*iii*) is identical to column (*iv*) in Table 5, while column (*iv*) in this table is the same but scales the estimate in $\hat{\gamma}_L$ on $\log K_{fst}$ by five.

Appendix Table N2: Cost Minimization with Usage-Weighted Subsidies

	Subsidy weighting			
	(*i*) Domestic	(*ii*) Usage	(*iii*) Unweighted	(*iv*) Usage
Subsidize *domestic* OSS investment	China & US			
Subsidize investment in *all* web OSS		China & US	Global	Global
Investment in web OSS	+9.1%	+1.8%	+57.9%	+4.8%
	[+8.0, +10.0]	[+1.4, +2.2]	[+49.9, +63.9]	[+4.0, +5.4]
↪ US	+16.2%	+3.5%	+39.8%	+3.4%
	[+13.9, +17.1]	[+2.9, +4.0]	[+35.7, +41.8]	[+2.9, +3.9]
↪ China	+21.4%	+2.4%	+57.4%	+2.5%
	[+19.2, +25.3]	[+1.7, +3.5]	[+42.6, +63.6]	[+1.6, +4.1]
Domestic web OSS investment	+21.2%	+3.8%	+44.8%	+3.8%
	[+18.6, +22.5]	[+3.2, +4.2]	[+40.1, +46.6]	[+3.3, +4.3]
↪ US	+19.6%	+4.3%	+38.6%	+3.7%
	[+17.0, +20.9]	[+3.4, +5.0]	[+34.4, +40.2]	[+3.1, +4.4]
↪ China	+46.7%	+2.6%	+65.2%	+2.7%
	[+41.9, +50.1]	[+1.3, +3.8]	[+60.1, +72.9]	[+1.2, +3.5]
Investment incentives, millions	+$0.9	+$0.2	+$6.2	+$0.4
	[+0.8, +1.0]	[+0.2, +0.2]	[+5.1, +7.1]	[+0.4, +0.5]
↪ US	+$0.8	+$0.2	+$2.6	+$0.2
	[+0.6, +0.9]	[+0.2, +0.2]	[+2.1, +2.8]	[+0.2, +0.2]
↪ China	+$0.1	+$0.0	+$0.5	+$0.0
	[+0.1, +0.1]	[+0.0, +0.0]	[+0.4, +0.6]	[+0.0, +0.0]
Firm costs, per dollar of incentive	−$30.4	−$41.6	−$26.1	−$46.2
	[−36.6, −26.9]	[−51.3, −31.6]	[−28.3, −25.1]	[−58.1, −35.8]
↪ US	−$25.5	−$34.5	−$20.9	−$35.9
	[−30.4, −22.7]	[−44.3, −27.0]	[−22.4, −19.8]	[−45.5, −27.1]
↪ China	−$1.7	−$1.9	−$1.4	−$2.1
	[−1.8, −1.5]	[−2.3, −1.5]	[−1.5, −1.3]	[−2.7, −1.8]

Columns (*i*) and (*iii*) are identical to columns (*iii*) and (*iv*) in Table 5, respectively, while columns (*ii*) and (*iv*) in this table instead weight subsidies for investing in each $s \in \mathcal{OS}_t$ by its share of usage among $\mathcal{OS}_t$.

Appendix Table N3: Cost Minimization with Different Domestic Subsidies

	China & US subsidy size		
	(*i*) \$50	(*ii*) \$100	(*iii*) \$150
Subsidize *domestic* OSS investment	China & US	China & US	China & US
Investment in web OSS	+4.9%	+9.1%	+13.3%
	[+4.1, +5.3]	[+8.0, +10.0]	[+11.0, +14.4]
↪ US	+8.6%	+16.2%	+23.4%
	[+7.3, +9.2]	[+13.9, +17.1]	[+20.2, +24.8]
↪ China	+12.1%	+21.4%	+30.5%
	[+10.0, +14.2]	[+19.2, +25.3]	[+26.0, +34.4]
Domestic web OSS investment	+11.6%	+21.2%	+30.3%
	[+10.2, +12.2]	[+18.6, +22.5]	[+26.4, +31.8]
↪ US	+10.3%	+19.6%	+28.4%
	[+8.6, +11.3]	[+17.0, +20.9]	[+24.4, +29.9]
↪ China	+26.3%	+46.7%	+61.9%
	[+23.0, +29.5]	[+41.9, +50.1]	[+57.0, +68.6]
Investment incentives, millions	+\$0.4	+\$0.9	+\$1.5
	[+0.4, +0.5]	[+0.8, +1.0]	[+1.2, +1.7]
↪ US	+\$0.4	+\$0.8	+\$1.3
	[+0.3, +0.4]	[+0.6, +0.9]	[+1.0, +1.4]
↪ China	+\$0.1	+\$0.1	+\$0.2
	[+0.0, +0.1]	[+0.1, +0.1]	[+0.2, +0.2]
Firm costs, per dollar of incentive	−\$36.2	−\$30.4	−\$27.2
	[−42.5, −30.2]	[−36.6, −26.9]	[−30.7, −24.7]
↪ US	−\$30.2	−\$25.5	−\$22.9
	[−36.2, −24.6]	[−30.4, −22.7]	[−25.9, −20.7]
↪ China	−\$1.9	−\$1.7	−\$1.5
	[−2.2, −1.7]	[−1.8, −1.5]	[−1.7, −1.4]

Column (*ii*) is identical to column (*iii*) in Table 5, while columns (*i*) and (*iii*) in this table instead subtract and add \$50 to the baseline domestic-focused subsidy of \$100 per hour.

Appendix Table N4: Cost Minimization with Different Global Subsidies

	Global subsidy size		
	(*i*) \$50	(*ii*) \$100	(*iii*) \$150
Subsidize investment in *all* web OSS	Global	Global	Global
Investment in web OSS	+33.0%	+57.9%	+78.8%
	[+27.7, +36.2]	[+49.9, +63.9]	[+69.0, +90.6]
↪ US	+22.6%	+39.8%	+54.5%
	[+19.3, +23.7]	[+35.7, +41.8]	[+49.5, +57.3]
↪ China	+31.2%	+57.4%	+77.0%
	[+26.4, +36.3]	[+42.6, +63.6]	[+59.5, +85.5]
Domestic web OSS investment	+25.0%	+44.8%	+60.4%
	[+22.1, +26.5]	[+40.1, +46.6]	[+55.8, +63.0]
↪ US	+21.6%	+38.6%	+52.7%
	[+18.7, +22.9]	[+34.4, +40.2]	[+48.2, +55.0]
↪ China	+38.9%	+65.2%	+85.4%
	[+34.7, +42.4]	[+60.1, +72.9]	[+78.7, +92.9]
Investment incentives, millions	+\$2.6	+\$6.2	+\$10.5
	[+2.1, +2.9]	[+5.1, +7.1]	[+8.7, +12.3]
↪ US	+\$1.1	+\$2.6	+\$4.2
	[+0.9, +1.2]	[+2.1, +2.8]	[+3.4, +4.7]
↪ China	+\$0.2	+\$0.5	+\$0.8
	[+0.2, +0.2]	[+0.4, +0.6]	[+0.7, +0.9]
Firm costs, per dollar of incentive	−\$37.6	−\$26.1	−\$20.8
	[−40.2, −35.0]	[−28.3, −25.1]	[−22.1, −19.4]
↪ US	−\$29.9	−\$20.9	−\$16.3
	[−32.2, −27.7]	[−22.4, −19.8]	[−17.5, −15.0]
↪ China	−\$1.9	−\$1.4	−\$1.1
	[−2.1, −1.8]	[−1.5, −1.3]	[−1.2, −1.0]

Column (*ii*) is identical to column (*iv*) in Table 5, while columns (*i*) and (*iii*) in this table instead subtract and add \$50 to the baseline global subsidy of \$100 per hour.

Appendix Table N5: Profit Maximization with Business Stealing and "Foreign Bias"

	(*i*) Unchanged	(*ii*) Business stealing	(*iii*) "Foreign bias"
Subsidize investment in *all* web OSS	Global	Global	Global
Investment in web OSS	+85.7%	+50.4%	+84.0%
	[+68.5, +107.2]	[+33.2, +64.6]	[+71.1, +104.0]
↪ US	+43.0%	+30.2%	+46.0%
	[+35.5, +51.5]	[+23.6, +36.7]	[+38.3, +53.2]
↪ China	+69.7%	+36.8%	+67.5%
	[+51.7, +85.5]	[+16.1, +47.7]	[+53.7, +84.3]
Domestic web OSS investment	+44.0%	+28.5%	+43.7%
	[+33.1, +53.2]	[+20.8, +33.1]	[+37.6, +52.9]
↪ US	+24.5%	+21.2%	+26.9%
	[+18.7, +28.4]	[+16.0, +24.6]	[+22.0, +29.0]
↪ China	+48.4%	+18.2%	+46.0%
	[+39.5, +60.6]	[+8.7, +26.2]	[+28.3, +56.5]
Investment incentives, millions	+$47.0	+$30.3	+$53.0
	[+34.1, +56.3]	[+23.4, +37.4]	[+38.8, +56.2]
↪ US	+$10.8	+$9.5	+$11.7
	[+8.8, +12.3]	[+7.8, +10.7]	[+9.5, +12.8]
↪ China	+$5.0	+$2.9	+$5.1
	[+3.6, +6.0]	[+2.4, +3.6]	[+3.7, +6.1]
Firm costs, per dollar of incentive	−$9.7	−$7.2	−$8.8
	[−11.0, −7.9]	[−8.8, −4.8]	[−9.3, −6.8]
↪ US	−$6.5	−$4.8	−$5.2
	[−7.4, −5.0]	[−6.1, −3.2]	[−6.0, −4.3]
↪ China	−$1.2	−$0.8	−$1.1
	[−1.5, −1.0]	[−1.1, −0.5]	[−1.4, −0.9]
Firm profit, per dollar of incentive	+$13.8	+$2.4	+$9.2
	[+10.2, +20.0]	[−4.1, +8.8]	[+7.5, +10.0]
↪ US	+$8.7	+$0.4	+$5.7
	[+6.9, +11.7]	[−5.8, +3.3]	[+4.2, +7.3]
↪ China	+$1.5	+$0.8	+$1.1
	[+0.9, +1.8]	[+0.2, +1.4]	[+0.9, +1.5]
Consumer surplus, per capita	−$19.2	+$32.3	−$6.6
	[−80.3, −0.1]	[−12.4, +99.4]	[−17.6, +7.3]
↪ US	−$80.1	+$84.5	−$10.8
	[−249.6, −9.1]	[−26.1, +344.4]	[−56.3, +26.5]
↪ China	−$21.0	+$29.0	−$9.7
	[−120.7, +10.0]	[−49.3, +118.7]	[−33.8, +22.0]

Column (*i*) is identical to column (*iv*) in Table 6. Column (*ii*) in this table is the same but sets the effect of OSS on quality to $\hat{\beta}_S = \hat{\beta}_{V1}$, the domestic preference for non-US and non-Chinese consumers. Column (*iii*) leaves $\hat{\beta}_S$ equal to this value and flips the sign of $\hat{\boldsymbol{\beta}}_V$.