How Demand for New Skills Affects Wage Inequality: The Case of Software Programmers

Gustavo De Souza[1], Jack Mannion[1], and Jacob Herbstman[2]

[1]Federal Reserve Bank of Chicago
[2]University of Chicago

December 11, 2024

Abstract

We study how the demand for programming skills has impacted inequality. We create a new dataset with information on wages, employment, and software of Brazilian programmers, covering the period from the birth of information technology (IT) to the rise of artificial intelligence (AI). High-ability, high-wage, and highly educated individuals in key technology hubs are more likely to become programmers. Creating software boosts both wages and career prospects of programmers, especially for those with specialized skills in AI and cybersecurity. These wage gains are concentrated among top programmers, increasing inequality within the profession. Therefore, increased demand for specialized skills in programming has contributed to wage inequality both within the programming field and between programmers and other occupations.

JEL Codes: J24, J23, O33

Key Words: technological progress, new work, AI, software

1 Introduction

Technological progress creates demand for specialized skills and even entirely new occupations (Acemoglu and Restrepo (2019), Lin (2011), Atalay et al. (2020b), Autor et al. (2024)). For example, the rise of information technology (IT) has led to the creation of the software programmer role and recent advancements in "big data" have greatly increased the demand for skills in artificial intelligence (AI), cybersecurity, and machine learning (Ţiţan et al. (2014), Eeckhout et al. (2021)). While many studies explore how new technologies replace workers, fewer focus on the positive side of technological progress—its effect on the demand for new skills. In this paper, we ask: How does the growing demand for new skills in software engineering affect the wages of programmers? Who supplies these skills and how does the demand for new skills contribute to inequality?

To answer these questions, we draw from various administrative sources to construct a dataset with information on wages, employment, and software developed by all programmers in Brazil. The data spans from the inception of the IT sector in 1987 through key technological milestones up to 2023, including the implementation of AI. We show that highly educated workers in a few major hubs tend to become programmers. Developing a software program increases wages by 5% permanently, with even higher wage increases for those with skills in cybersecurity, data management, and AI. Additionally, higher-ability programmers receive a greater return from creating new software. These findings suggest that the rising demand for specialized skills in software engineering has increased inequality not only between programmers and workers in other professions but also among programmers within the profession.

We begin by collecting a new dataset on software and its programmers, which we merge with administrative employment data. In 1987, the Brazilian government implemented a copyright law that protected the source code of computer programs. This law allowed firms to register their software with the Brazilian Patent Office, providing stronger intellectual property protection for a small fee. We collect data on all the software programs ever

registered with the Brazilian Patent Office, which includes detailed information on their ownership, programmers, applications, and methods. The data goes back to 1987, before being a programmer was a common occupation, and up to 2023, after the emergence of AI.

We identify each software programmer on a matched employer-employee administrative dataset that covers the entire formal workforce in Brazil, linking software creation to the labor market history of its programmers. The final dataset contains information on all the software registered in Brazil and the labor market history of its programmers.

Using data on programmers and their labor market histories, we show that high-ability individuals in key technology hubs are more likely to become programmers. Even before creating their first software program, programmers tend to have higher wages, more education, and greater ability than non-programmers.[1] They are concentrated in high-income technology hubs such as São Paulo, Rio de Janeiro, and Brasília. Programmers who develop specialized skills, particularly in niche programming languages or advanced software areas like AI, earn higher wages early in their careers. People from racial or ethnic minorities are underrepresented among programmers, and there is a greater likelihood that programmers are white and male. These findings emphasize the role of selection in the supply of new programming skills.

To isolate the effect of software creation from selection, we conduct a difference-in-difference comparing the career trajectories of individuals who create a software program with a matched control group of similar individuals who do not. Exploiting the richness of the Brazilian dataset, we match programmers and non-programmers based on wage, working hours, occupation, age, and region on the years prior their first software, ensuring that the control group closely mirrors the treatment group in key characteristics. A balance test and pre-period parallel trends confirm that both groups were similar before programmers created their first software.

Creating software has a significant impact on the careers of high-ability programmers

[1] Ability is calculated as the residual from a Mincer regression.

with specialized skills, leading to higher wages and higher positions within firms. On average, programmers experience a 5% wage increase after creating their first software, which grows to 8% over the next decade. However, these wage gains are concentrated among a few programmers. Those working on modern applications such as AI or cybersecurity see the largest wage increases, while the choice of programming language plays a smaller role. Programmers with wage in the top quartile have a 120% greater wage increase than programmers with wage in the bottom quartile. Therefore, increased demand for specialized skills in programming has contributed to wage inequality both within the programming field and between programmers and other occupations.

This paper shows that technological progress also affects the labor market through the demand for new skills and the selection on its supply. The emergence of IT has increased the need for programmers, with high-ability individuals supplying this demand. Continuous advancements in IT have further increased the demand for specialized skills, which has led to greater inequality within the programming profession than between programmers and other occupations.

This paper contributes to the literature studying new work, such as Acemoglu and Restrepo (2019), Atalay et al. (2020a), Autor et al. (2024), Connor et al. (2024), Kim (2022), Lin (2011), Deming and Noray (2020), Webb (2019), and Atalay et al. (2020b). This literature has mostly relied on job ads (Atalay et al. (2020a),Atalay et al. (2020b)), occupational descriptions (Autor et al. (2024),Lin (2011)), and patent data (Webb (2019), Kelly et al. (2021)) to study how technological progress affects tasks performed by workers. Like we do, Autor et al. (2024) study the rise of new work driven by AI and digitization. This literature has concluded that technological progress has replaced tasks previously done by routine low-skill workers while creating tasks performed by high-skill workers.

We contribute to the new work literature by showing that, in the case of programmers, new work not only increases wage inequality between new and old occupations but also within newly created occupations. Using a new dataset, we address key questions not yet explored:

Who selects into new work? How are their skills rewarded? How are different workers rewarded differently? We find that high-wage, high-ability, and high-education individuals are more likely to become programmers. These workers see a permanent wage increase after creating their first software program, with even greater growth for those skilled in newly emerging technologies.

This paper also relates to the literature on the effects of AI and software on the labor market, such as Acemoglu et al. (2022) ,Babina et al. (2024), and Alekseeva et al. (2021a). Using job ad data, Acemoglu et al. (2022) do not find a significant effect of AI adoption on firms. Babina et al. (2024) argue that AI drives firm growth by enabling the introduction of new products. Alekseeva et al. (2021a) show that the demand for AI skills is increasing.

We make two key contributions to this literature. First, we construct a new dataset that not only measures AI software creation but also links it to the workers responsible for developing it. Second, we examine the effect of AI through its impact on the demand for new skills.

The paper is organized as follows. The second section covers the data, with a focus on the institutional framework that allows firms to register their software with the Brazilian patent office. The third section examines the characteristics of programmers in Brazil. The fourth section introduces a difference-in-difference strategy to separate the effect of becoming a programmer from selection. The fifth section explores the impact of creating software on wages and career prospects. The final section concludes.

2 Data

We constructed a new dataset that includes information on the labor market outcomes of programmers and the software they developed, covering the period from the emergence of IT to recent innovations such as AI. Data on software were collected from the Brazilian Patent Office, which we are the first to use. Labor market information comes from RAIS, a widely

used matched employer-employee dataset (de Souza (2020), de Souza (2022), de Souza and Li (2023), Gerard and Gonzaga (2021), and Britto et al. (2022a)). Below, we describe in detail the process of constructing this dataset.

2.1 Software

2.1.1 Institutional Setting

Since 1987, the Brazilian government has allowed programmers to register their software with the patent office under a special copyright system, in a process similar to patent registration. This unique feature of the Brazilian copyright law allows us to identify programmers and their software in a period of accelerated technological development in computer science.

Most countries do not maintain a formal record of the software created. In 1995, the WTO established the first international agreement extending intellectual property protection to software. Under the WTO's Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS), computer programs are protected as literary works under the Berne Convention (1971). This means software receives the same copyright protection as books, songs, and movies. However, unlike inventions, which are patentable, software is not usually protected by patents and, therefore, does not typically leave a paper trail.

Brazilian software law provides special copyright protection for software. In 1987, the Brazilian government passed a law that created a special type of intellectual property right for software, similar to a patent.[2] The law defines software as a "set of instructions or statements, written in proper language, to be used directly or indirectly by a computer to obtain a certain result." Software owners have rights over their creation for 50 years. This protection includes preventing the unauthorized use of the software or its source code. Individuals found violating software rights can face fines and prison sentences ranging from

[2]The first software law was Law number 7,646/1987. It was later updated by Law 9,609/1998.

6 months to 4 years.

Firms register software on the Patent Office. Firms that have developed software and want to protect its source code can register it with the Brazilian Patent Office. To do this, firms submit an electronic form containing the firm's name, the software's programmers, the software's application, the methods used, and the programming languages. They also send an encrypted version of the source code, with the encryption key kept by the firm. The Patent Office only releases the source code to a specialist in the event of intellectual property litigation, proving the firm's ownership of the software. The Patent Office's website provides information about all registered software.

Litigation. If a firm suspects its source code has been stolen, for example by a former employee, it can file a lawsuit against the accused company. In such cases, the source code held at the Patent Office is compared to the code used by the accused firm. A judge then decides whether there is reasonable suspicion that the accused company stole part or all of the code.

Why are firms registering their software? Firms typically list four common reasons for registering their software: intellectual protection, government procurement, signaling to investors, the low financial cost, and the secrecy advantage.[3] The first reason aligns with the software law's purpose: registering software provides proof of ownership over the source code and offers a legal claim against unauthorized use. Second, the government requires software registration for the purchase of any software or IT-related service. Third, registration signals to potential investors the size of firm's intangible capital. In fact, the Brazilian Association of Software Firms notes that most startup investors require all firm's software to be registered.[4] Finally, the cost of registration is low. The Patent Office charges only $34 to register software.

[3] Torquato (2022)

[4] Associação Brasileira das Empresas de Software (ABES)

Additionally, unlike patents, the source code or software details are never revealed to the public, so the firm does not disclose sensitive information to competitors.

2.1.2 Software Dataset

In this section, we describe how we collect information from all software ever registered in the patent office. Moreover, we show that, because software are not usually guaranteed patent protection, the software dataset is a broader measure of the software created in Brazil than patents.

Data collection and variables. We collected from the patent's office website descriptive information on all 37,677 softwares registered. For each software we observe the name of its owners, the name of the programmers on the project, the publication date, the title of the software, the programming language, the software application field, and the software type. The application field is the sector or areas where the software is intended to be used. The program type categorize software according to their technical functions and purposes. It ranges from fundamental systems like Operating Systems (SO01) and Network Controllers (SO08) to more specialized tools such as encryption software (PD03) and artificial intelligence applications (IA01). Section A.1 and A.2, in the Appendix, lists all program types and application fields. The dataset spans from 1987 to 2023.

Coverage and comparison to alternative datasets. Previous studies measuring software creation have relied on patents or job ads, but neither is as comprehensive as our dataset. Our software dataset provides broader coverage than software patents because software is typically not patentable. Under U.S. law, software can only be patented if it does more than cover an abstract idea. For example, software that controls a robotic arm is patentable, but software that calculates optimal pricing is not.[5] The U.S. Supreme Court has ruled that

[5]For instance, the Supreme Court case Parker v. Flook (1978) ruled that an invention containing only a new mathematical algorithm is not patentable.

certain types of software, such as AI software combining images (Digitech Image Technologies, LLC v. Electronics for Imaging, Inc.), online transaction software (buySAFE, Inc. v. Google, Inc.), and software implementing numerical algorithms (Gottschalk v. Benson and Parker v. Flook (1978)), are not patentable. However, any of these could be registered with the Brazilian Patent Office if created by a Brazilian company.

Table 1 shows that firms are more likely to register their software with the Brazilian Patent Office than to seek patent protection. In total, our sample includes 37,677 software, while only 445 software patents were granted during the same period. Additionally, line 6 indicates that even among firms that have registered software, a software patent is obtained only 10% of the time. These statistics demonstrate that software registered with the Patent Office covers a much broader range of software compared to those that are patented.

Others have attempted to measure software creation and adoption using job ads. Job ads targeting programmers or requiring AI skills can indicate a company's intent to create software or adopt AI. However, our measure offers two advantages. First, we track the actual software output of programmers, not just the intent to create it. Second, we capture software created by incumbent employees, not only those being hired.

Table 1: **Comparison Between the Coverage of Software and Software-Patent**

Number of Software	37,677
Number of Software-Patent	445
Number of firms with software	8,640
Number of firms with software-patent	290
Percentage of firms with software-patent with at least one software	87%
Percentage of firms with software with at least one software-patent	10.72%

Note: This table shows statistics from the software database and from patents. Patent data comes from de Souza (2022). Following Webb (2019), we say that a patent is a software patent if the title contains the words "software", "computer", or "program" and not the words "chip", "semiconductor", or "circuit".

Summary statistics. Figure C1 in the appendix shows that the number of software programs have increased over time. As shown in Figure C2, most applications come from firms

in support sectors, such as IT, and from universities. According to Figure C3, the most common types of programs are related to business software, data management, and automation.

2.2 Labor Market Data

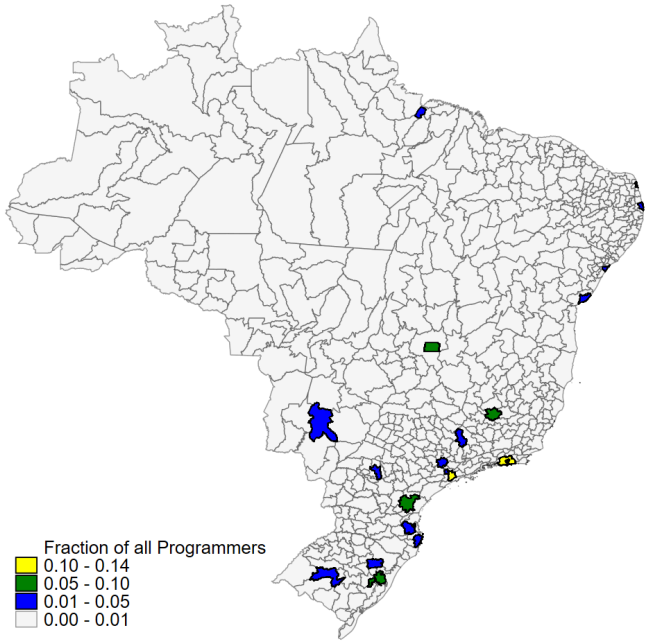
Matched Employer-Employee. Labor data comes from the matched employer-employee dataset RAIS (*Relação Anual de Informações Sociais*), an administrative dataset collected by the Brazilian Ministry of Labor. RAIS follows the universe of formal firms and workers over time, starting in 1985, linking them to their tax identifiers. RAIS contains information on wages, occupation, education, sector, location, and other demographic information. This will allow us to follow workers over time and across firms to measure the effects of publishing software on the likelihood of changing firms, for example. Section A.3 describes the steps to merge RAIS to the software dataset.

3 Characteristics of Programmers

In this section we show statistics of the labor market of programmers. We call a programmer an individual that has ever registered software. Programmers have higher wages, education, and ability than non-programmers; they are regionally concentrated in a few technology hubs; and programmers with modern skills have higher wages. These results suggest that, as technology advances, programmers with specialized skills and favorable locations will earn higher wages, widening wage inequality within the profession and compared to other workers.

Programmers are concentrated in a few technology hubs. Figure 1 shows the geographical distribution of programmers in Brazil. Dense and high-income microregions concentrate most of the programmers. About 30% of all Brazilian programmers live in São Paulo, Rio de Janeiro or Brasília, the three richest microregions.

Figure 1: **Geographical Distribution of Programmers**



Note: This figure plots a heat map of the location within Brazil of programmers in the year prior to their first software at the microregion level who register their software at the INPI (Institute Nacional da Propriedade Insturial).

Programmers have higher wages, education, and ability than non-programmers. Table 2 compares programmers to non-programmers. Column 1 presents the characteristics of a random sample of workers in Brazil, while Column 2 shows the characteristics of programmers one year before publishing their first software. On average, programmers earn higher wages and have more years of education, even though they haven't created software at that point in time. The general population has slightly less than the required years for a high school diploma, whereas programmers have nearly enough years for a college degree.

The higher wages of programmers cannot be explained by age or location. Line 3 of Table 2 highlights the difference in ability between programmers and non-programmers, which is the portion of hourly wage not explained by age or region.[6]

Minorities are underrepresented among programmers. Programmers are 10% more likely

[6]We define ability as the residual of log hourly wages regressed on age × year and microregion × year fixed effects.

to be white and 25% more likely to be male compared to the general population.

Table 2: **Statistics of Programmers and Non-Programmers**

	(1) Non-Programmers	(2) Programmers	(3) Difference
Real Hourly Wage	45.90	204.85	158.95***
	(101.08)	(269.76)	(0.00)
Years of Education	11.07	15.70	4.63***
	(3.48)	(1.85)	(0.00)
Ability	-0.00	1.46	1.46***
	(0.75)	(0.70)	(0.00)
Age	35.69	36.12	0.43**
	(11.39)	(10.03)	(0.03)
Percentage Male	0.59	0.83	0.25***
	(0.49)	(0.37)	(0.00)
Percentage White	0.54	0.63	0.10***
	(0.50)	(0.48)	(0.00)
Observations	5,657,062	2,675	5,659,737

Note: This table presents summary statistics for programmers and non-programmers. The statistics for programmers are calculated for the year prior to their first software registration, while the statistics for non-programmers are based on a 1% random sample. Standard deviations are shown in parentheses in Columns 1 and 2, and p-values are reported in parentheses in Column 3. Row 1 displays the real hourly wage of Brazilian programmers and non-programmers, row 2 the years of education, row 3 the ability score, which is calculated as the residual from a Mincer regression of age × year and microregion × year fixed effects on hourly wage. Row 4 shows the average age of the individuals, Row 5 the proportion of males, and Row 6 the proportion of the whites. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$.

Programmers with skills in niche programming languages have higher wages. Table 3 shows the average wage of programmers according to the programming languages used in their first software. Niche programming languages, such as Fortran, Matlab, and C, are among the highest paid programming languages. In contrast, programmers using more popular languages, like Java, JavaScript, and SQL, tend to earn relatively lower wages.

Figure C4 in the Appendix shows the correlation between the average hourly wage of programmers and the characteristics of the programming languages they use.[7] Panels (a) and (b) of Figure C4 show that programmers working with niche languages used in scientific

[7]See Section C.2 for more details on how the characteristics of programming languages were constructed.

computing tend to have higher hourly wages. Figure C4 shows a negative correlation between the popularity of a programming language and hourly wages.

Table 3: **Wage and Years of Education by Programmers of Different Languages**

	(1) Wage	(2) Yrs. of Education	(3) Percent Male
Fortran	14377.78	16.69	0.83
Matlab	12075.90	17.22	0.87
Python	11365.57	16.11	0.75
R	10469.45	16.29	0.71
C++	10104.15	16.11	0.90
C	9842.82	15.94	0.91
Java	8928.96	16.05	0.81
HTML	8409.78	16.14	0.80
Others	8326.61	15.72	0.84
SQL	8067.74	15.71	0.80
Javascript	7572.28	15.75	0.83
PHP	7503.43	15.73	0.79
CSS	7237.16	15.91	0.83
Delphi	7163.16	15.08	0.89
C#	7059.20	15.64	0.88
Excel	6946.69	16.50	0.88
Basic	6806.09	15.32	0.87
N	6905	6849	6886

Note: In column 1, this table shows the average monthly wage by programming language. Column 2 shows the average number of years of education by programming language. Finally, column 3 shows the fraction of programmers that are male by programming language. Statistics are calculate according to characteristics of the first software program on the year before its registration.

Programmers with skills in modern types, such as AI, tend to earn higher wages. Table 4 shows the average hourly wage and years of education for programmers working on different types of software.[8] Advanced methodologies, such as simulation, modeling, and artificial intelligence, have programmers with highest wage. In contrast, programmers who register software of less advanced types, such as administrative work or entertainment, earn lower wages in average.

[8]Program types categorize software based on their technical functions and purposes. Section A.1 lists all program types.

Table 4: **Wage and Years of Education by Program Type**

	(1) Wage	(2) Yrs. of Education	(3) Percent Male
Simulation and Modeling	12736.45	16.82	0.87
Artificial Intelligence	10544.68	16.12	0.85
Instrumentation	10404.18	15.95	0.81
Technical Scientific	9889.11	16.34	0.81
Development Support	9595.82	16.18	0.89
Report Generation	7881.71	15.87	0.83
Languages	7854.22	15.78	0.95
Communication Technology	7731.48	14.90	0.82
Data Management	7518.68	15.62	0.83
Operation Systems	7257.82	15.44	0.88
Applications	7243.25	15.61	0.82
Utilities	7223.28	15.64	0.83
Data Communication	7174.65	15.45	0.92
Administrative	7069.28	15.61	0.88
Entertainment	6980.02	16.13	0.79
Data Protection	6897.28	15.38	0.83
Teleinformatics	6867.72	15.44	0.85
Automation	6707.38	15.32	0.85
N	9155	9131	9122

Note: In column 1, this table shows the average monthly wage by program type. Column 2 shows the average number of years of education by program type. Finally, column 3 shows the fraction of programmers that are male by program type.

Selection and skill premium. Programmers in tech hubs earn higher wages, particularly those with niche or modern skills. These findings are driven by two factors: selection and the skill premium. High-ability individuals may choose to become programmers, which could explain the patterns we have observed. In the next section, we present an event study to separate the effect of rewards for specific technological skills from the influence of selection.

4 Empirics

In this section, we outline an empirical strategy to separate the reward for creating software from the effects of selection. To achieve this, we match software programmers with similar individuals who have not created software. Then, we compare the wage growth of programmers to their similar counterpart after a software program is created. To understand how the returns from software creation vary based on a programmer's skills or characteristics,

we explain how to calculate heterogeneous treatment effects using random forest.

4.1 Matching

Matching procedure. To identify the causal effect of creating software, we match each programmer in the database to a set of similar individuals that did not register any software. Our approach, which follows the methodology outlined in Iacus et al. (2012), creates a comparison group for each individual creating software.

We match programmers to non-programmers based on monthly wage, hours of work, occupation, age, and microregion for the two years leading up to the first software. Specifically, for each individual who authored a software project, we identify a group of people who, in the past two years, fell within the same wage quintile, had similar hours of work and age, and were in the same microregion and occupation.

Matching on such large number of variables is possible due to the detailed nature of the Brazilian data. Wage is included because, when conditioned on occupation, age, sector, and region, it serves as a standard measure of ability. Matching on occupation helps account for differences in wage profiles across various jobs.[9] Age is important since it affects wage growth, which tends to be higher for younger individuals. Hours of work are matched to capture differences in employment types, while matching on region control for potential regional shocks.[10]

Balance test. Table 5 shows no statistical difference between matched programmers and non-programmers on a set of targeted and non-targeted characteristics. Column 1 presents the statistics for the matched sample of non-programmers, which we also call the control group. Column 2 displays the statistics for individuals who will eventually create software, known as the treatment group. These statistics are calculated for the year prior to the

[9] We call a programmer anyone that has authored software. On the data, those could be software engineer, managers of software engineers, CEO of a IT company, among many others. For this reason, we also match on the occupation observed in the matched employer-employee dataset.

[10] In practice, we brake wages and age in quintiles. We match individuals on 4-digit occupational codes.

programmers' first software. The control and treatment groups are similar not only in the matched variables but also in untargeted demographic characteristics. This supports our assumption that both groups were following similar trends before the software creation.

Table 5: **Difference Between Programmers (Treatment) and Matched Non-Programmers (Control)**

	(1) Control	(2) Treatment	(3) Difference
Targeted Variables			
Real Hourly Wage	271.56	274.45	2.89
	(558.90)	(497.56)	(0.92)
Years of Education	15.89	15.91	0.02
	(1.49)	(1.53)	(0.85)
Age	40.08	39.86	-0.22
	(9.01)	(8.50)	(0.66)
Weekly Hours	40.51	40.46	-0.06
	(5.04)	(4.35)	(0.84)
Not Targeted Variables			
Percentage Male	0.84	0.84	0.01
	(0.37)	(0.37)	(0.80)
Percentage White	0.81	0.82	0.01
	(0.39)	(0.38)	(0.72)
Log(Avg. Firm Size)	7.74	7.79	0.05
	(2.07)	(2.11)	(0.72)
Log(Avg. Firm Wage Bill)	16.16	16.25	0.09
	(2.32)	(2.49)	(0.55)
Firm Age	34.04	35.01	0.97
	(15.57)	(16.05)	(0.31)
Observations	65,822	322	66,144

Note: This table shows summary statistics of characteristics among the matched programmers and matched control group. All summary statistics are computed in the year prior to publishing the first software. Standard deviations are reported in parenthesis in columns 1 and 2. P-values are reported in parenthesis in column 3 and standard errors are clustered at the individual level. Rows 1,2,3, and 4 show the balance table results for variables that are targeted in our matching strategy including real hourly wage, years of education, age, weekly hours, respectively. Rows 5,6,7,8, and 9 show the balance table results for variables that are not targeted in our matching strategy including the percentage of the population that are male and white, the log of the average number of workers of the employee's firm, the log of the average wage bill of the employee's firm, and the average age of the employee's firm. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$.

4.2 Empirical Model

Main empirical model. We identify the effect of creating software on the career of programmers using

$$y_{i,t} = \beta \times \mathbb{1}_i\{\text{Treatment}\} \times \mathbb{1}_{i,t}\{\text{Aft. First Software}\} + \mu_i + \mu_{g(i),t} + \epsilon_{i,t}, \tag{1}$$

where $y_{i,t}$ is an outcome of individual i in year t, $\mathbb{1}_i\{\text{Treatment}\}$ is a dummy variable that takes one if individual i is a programmer and zero otherwise, $\mathbb{1}_{i,t}\{\text{Aft. First Software}\}$ is a dummy that takes one after individual i published their first software, μ_i are individual fixed effects, $\mu_{g(i),t}$ are time-group fixed effects that are common for all individual matched to group $g(i)$. The parameter of interest, β, captures the effect of publishing software on variable $y_{i,t}$.

Identifying assumption: unpredictability of software completion dates. The parameter of interest is β, which captures the effect of creating software on the career trajectory of programmers. The identifying assumption is that the treatment and control groups are in parallel trends. In other words, without the creation of the software, programmers would be expected to follow the same career path as their matched non-programmer counterparts.

The key assumption is that the timing of a software's completion is uncertain. This uncertainty means that a software completion does not correlate with other events affecting individual workers, like scheduled promotions or finishing a degree.

Research on project management has long argued that software completion times are uncertain, which supports the identification strategy. Up to 80% of software projects experience schedule overruns (Moløkken and Jørgensen, 2003; van Genuchten, 1992). Delays are often due to unpredictable requirements and technical complexities (Kitchenham et al., 2004; Standish Group, 1994; Jones, 2006; DeMarco and Lister, 1987; Marinho et al., 2014; van Genuchten, 1992).[11] According to Brooks (1975), a former software manager at IBM,

[11] See Marinho et al. (2014) and van Genuchten (1992) for a review of factors contributing to software

managers often underestimate the time needed to finish projects—a problem that additional workers cannot solve due to coordination issues.

Many well-known software projects have experienced significant delays, supporting the assumption that software completion times are often uncertain. For instance, Windows Vista was planned for release in 2003 but only launched in 2007. Similarly, Steve Jobs announced Mac OS X for 1999, though it didn't come out until 2001. Another example is the video game Final Fantasy XV, originally set for 2006; after many delays, it finally launched in 2016. Meanwhile, fans have been waiting for GTA VI since 2013, with no release date in sight.

Dynamic Model. To test for pre-period parallel trends, we use the following dynamic model

$$y_{i,t} = \sum_j \beta_j \times \mathbb{1}_{i,t}\{j \text{ Yrs to First Software}\} \times \mathbb{1}_i\{\text{Treatment}\} + \mu_i + \mu_{g(i),t} + \epsilon_{i,t} \qquad (2)$$

where $\mathbb{1}_{i,t}\{j \text{ Yrs to First Software}\}$ is a dummy taking one j years after the first software by programmer i. If parallel trends in the pre-period is valid, it should be the case that $\beta_j \approx 0$ for all $j < 0$.

4.3 Heterogeneity in the Treatment Effect

We are also interested in how the effect of creating software varies based on the characteristics of the software or its creator. This is important for understanding how the demand for specialized skills is contributing to growing inequality among programmers. In this subsection, we explain the methods we use to estimate the heterogeneity in the treatment effect.

Heterogeneity by characteristic of the software. To identify heterogeneity based on software type or programming language, we limit the sample to programmers (and their

development delays.

matched controls) whose first software falls within a specific language or type. For example, to examine the effect of creating AI software on wage growth, we run regression 2 only on programmers (and their matched group) whose first software is in AI.

Heterogeneity by characteristic of the programmer using random forest. To calculate heterogeneity on the treatment effect according to the characteristics of the programmer, we use random forest

We calculate heterogeneity on the treatment effect using random forest (Athey et al. (2019),Wager and Athey (2018)). We re-write model (1) as

$$\Delta y_{i,t} = \beta\left(X_i\right) \times \mathbb{1}_i\{\text{Treatment}\} + \mu_{g(i),t} + \epsilon_{i,t}, \tag{3}$$

where $\Delta y_{i,t}$ is the long-run difference in outcome $y_{i,t}$ between the year prior the software creation and three years after that. $\beta\left(X_i\right)$ is the effect of creating software on individuals with characteristic X_i, where X_i is a vector containing individual i's gender, age, and an indicator if individual i lives in in São Paulo, Rio de Janeiro, Paraná, or Brasília in the year prior to treatment. X_i also includes individual i's wage and years of education from the first year that they appear in RAIS.[12] Appendix B.1 explains the computational details of estimating the function $\beta\left(X_i\right)$ using random forests.

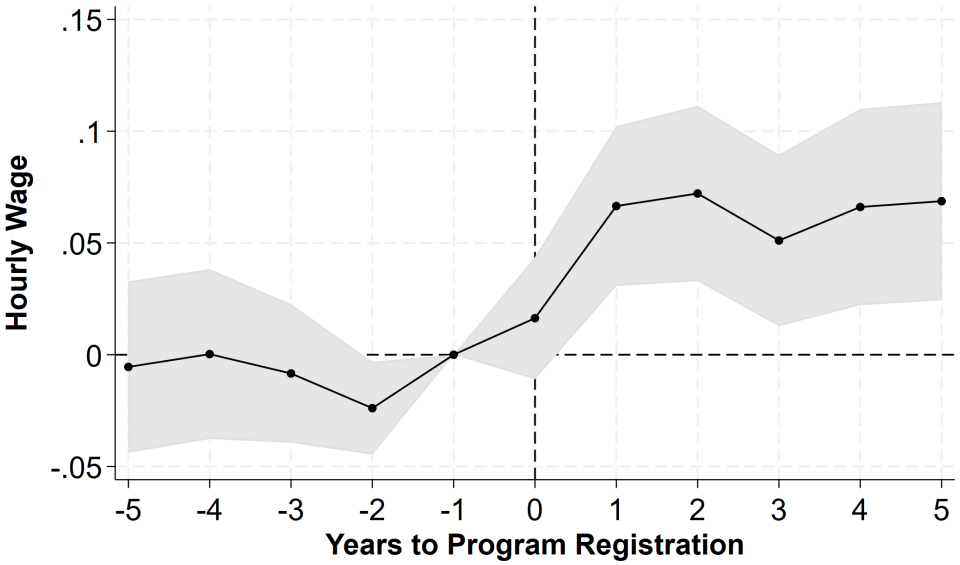
5 Effect of Software on the Career of Programmers

5.1 Average Effect of Software Program

In this section, we show how creating a software program affects the career trajectories of programmers. After programmers create software, their wages increase, and they move into higher positions within their companies.

[12] As our main outcome of interest is the effect of software on hourly wage, we use the first wage to avoid a mechanical correlation between the effect of software and its correlation with initial wage.

Figure 2: **Dynamic Effects of Software on Hourly Wage**



Note: The figure above show the dynamic effects of publishing software on log hourly wage. The x-axis measures the distance to the year of the software being published. Each dot is an estimated coefficient and the grey area is a 90% confidence interval. Standard errors clustered at the individual level.

Software increases wage and position within the company. Figure 2 shows the dynamic estimates of the effect of the first software on the hourly wage of programmers. Notice that, before the publication of the software, programmers and their matched counterpart have similar wage growth even in periods in which they were not matched. However, after programmers finish their first software, their hourly wage grows by 5% on average. Figure C5, in the appendix, shows that this effect is persistent: 10 years after the software publication, programmers have a 8% higher hourly wage than similar individuals that never wrote a computer software.

Table 6 shows that creating a software program advances the careers of programmers. After developing their first software, programmers earn higher wages, as shown in column 1. Additionally, column 2 indicates that creating software increases the likelihood of being promoted to a professional role by 5%.[13] Column 3 shows that software does not affect the

[13]The professional hierarchy is the highest level a worker can achieve prior to entering management, therefore representing a significant career achievement. Data on occupation hierarchy comes from Poliquin

education of workers. Column 4 shows that programmers are less likely to change employers after creating a software, which could be explained by the creation of firm specific human capital.

Table 6: **Effect of Software on the Career of Programmers**

	(1) *Log Hourly Wage*	(2) $\mathbb{1}\{$*Professional*$\}$	(3) *Log Yrs. Educ.*	(4) $\mathbb{1}\{$*Chng. Emp.*$\}$
$\mathbb{1}\{$Software$\}$	0.0584**	0.0488***	-0.00445	-0.0401**
	(0.0234)	(0.0177)	(0.00554)	(0.0190)
R^2	0.838	0.818	0.712	0.648
Observations	1,289,724	799,710	1,287,388	1,266,009

Note: This table presents the results of the differences-in-differences baseline specification shown in Model (1). Standard errors are reported in parenthesis and are clustered at the individual level. Column 1 shows the effect of software on the log hourly wage of programmers. Column 2 shows the effect on an indicator variable of whether or not a programmer's occupation hierarchy is classified as professional. Column 3 shows the effect on the log number of years of education of a programmer. Finally, column 4 shows the effect on the probability of a programmer changing employers relative to their employer in the year prior to software registration. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$.

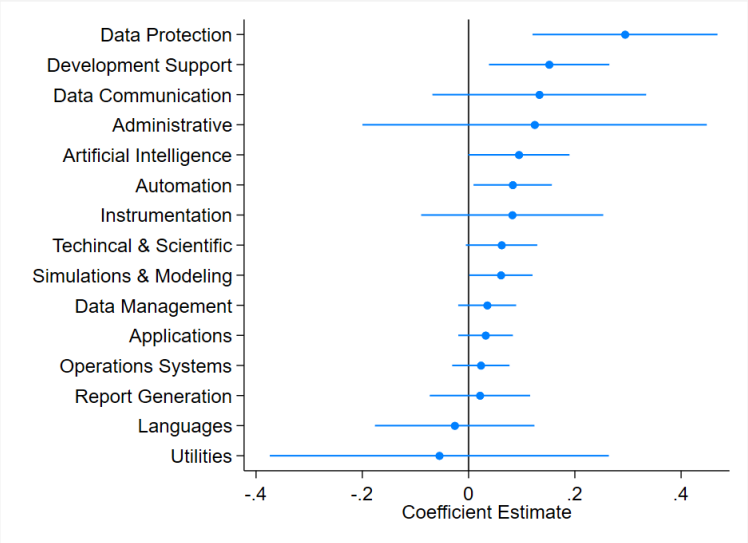
Demand for programming skills increases inequality. These results show that programmers, compared to individuals with similar skills and occupations, experience a significant and lasting increase in wages and hierarchical position after their first software. Therefore, the high-ability individuals who became programmers saw larger wage increases than those who did not, contributing to this growing gap.

5.2 Heterogeneity According to Software Characteristic

In this subsection, we show that the impact of software creation on a programmer's career is greater for software related to AI or cybersecurity, which are modern, cutting-edge applications. However, this effect does not depend on the programming language used.

(2020)

Figure 3: **Effect of Software on Wage According to Software Type**



Note: This figure shows the effect of creating a software, according to model (1), on log hourly wage limiting the sample to different program types. Standard errors are clustered at the individual level. 90% confidence intervals are shown.

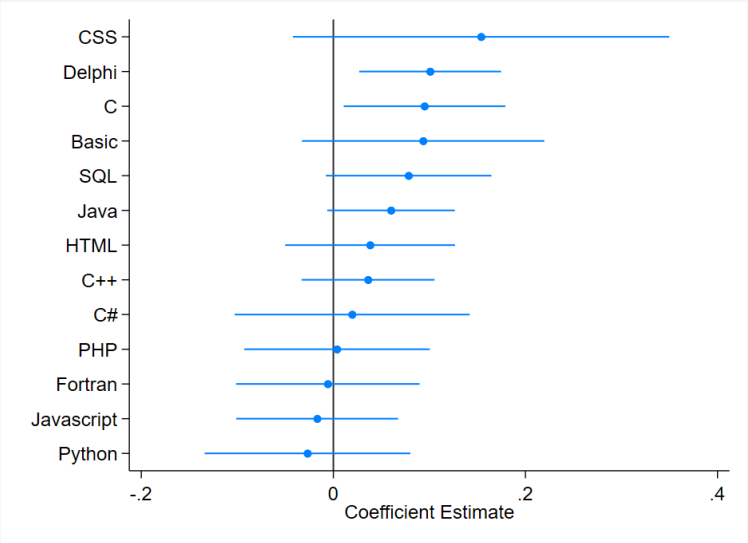
Programmers with modern software applications get larger wage gains. Figure 3 shows significant variation in the effect of software based on its type. Programmers developing software for data protection, development support, or data communication see a 20% increase in their hourly wage. Software related to AI or automation results in a 10% wage growth.[14] Other software types lead to little wage growth, with overall wage increases close to zero. The standard deviation in wage growth across different software types is about 12%.

Weak heterogeneity on programming languages. Figure 4 shows the effect of software of different programming languages on wages. Software written in CSS, Delphi, and C leads to a larger wage increase, averaging about 10%. However, overall, there is no significant difference in wage growth based on the programming language used.

Demand for modern programming skills increases inequality within-programmers. These results suggest that knowledge of basic tools, such as common programming languages,

[14]Our results are similar in magnitude to the ones presented by Alekseeva et al. (2021b), who document an 11% within-firm and a 5% within-job-title wage premium associated with AI skills.

Figure 4: **Effect of Software on Wage According to Programming Language**



Note: This figure shows the effect of creating a software, according to model (1), on log hourly wage limiting the sample to different program types. Standard errors are clustered at the individual level. 90% confidence intervals are shown.

does not lead to significant wage growth for most programmers. What matters for wage and career growth is how programmers apply these languages to advanced, frontier applications, like AI and data protection. Programmers who gain expertise in cutting-edge software applications can achieve up to 20% higher wage growth compared to other programmers. As a result, technological progress increases inequality more within the group of programmers than between programmers and non-programmers.

5.3 Heterogeneity According to Programmer Characteristics

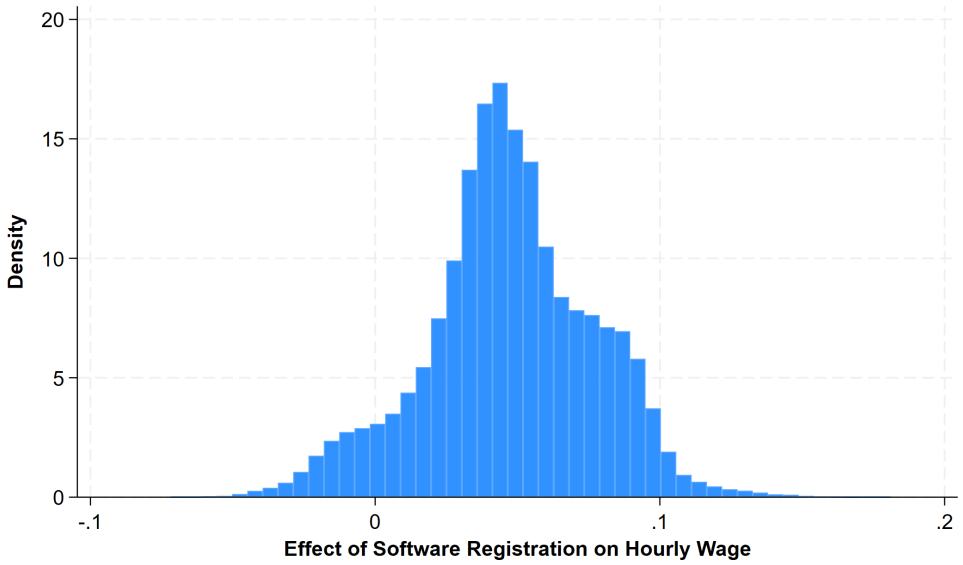
We have shown that creating software leads to significant differences in wage growth depending on characteristics of the software. In this subsection, we show that high-wage programmers have larger wage gains compared to low-wage programmers.

Large heterogeneity on the effect of software. Figure 5 shows the distribution of the effect of software on wages. To create this distribution, we estimate the heterogeneity of software effects using the random forest method, as explained in Section 4.3. Random forest

allows us to calculate a conditional treatment effect for each individual in the sample, based on their characteristics. We plot that distribution in 5.[15]

Figure 5 shows significant variation in the effect of software on wages. The median individual experiences a wage growth of around 5%, while some individuals see expected wage growth of over 12%.

Figure 5: **Distribution of the Effect of Software on Hourly Wage**



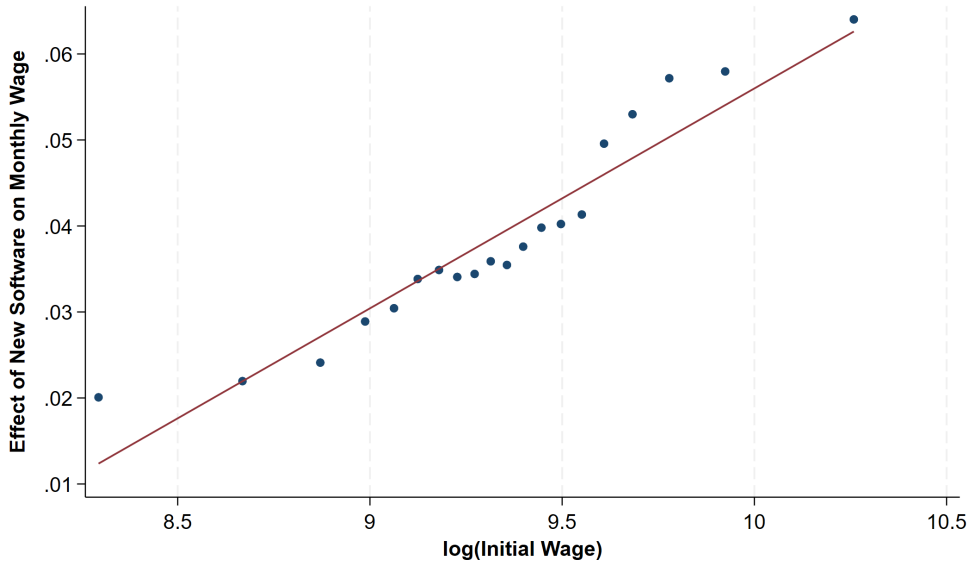
Note: This figure shows a histogram of the treatment effect estimates on log hourly wage after a programmer's first software from the causal forest as described in Appendix B.1.

High-wage individuals have even larger wage gains. Figure 6 shows that initial differences in wages explain most of the variation in the effect of software on wages. The figure plots the effect of software on hourly wages against ability, which is measured as the residual of a Mincer regression. To prevent bias, the regression only includes data from before the first software.

[15]We estimate $\beta(.)$ in equation 3 using a random forest model. In this context, $\beta(X_i)$ represents the effect of software on an individual's wages, based on their characteristics X_i. By calculating $\beta(X_i)$ for each person in the sample, we obtain a unique treatment effect for each individual.

Figure 6 shows that most of the variation in the effect of software on wages is explained by differences in initial wage. Programmers on the top quartile of wage before the software creation experienced more than a 100% larger wage increase compared to those on the bottom quartile. As a result, high-wage individuals further widen the wage gap between themselves and other programmers.

Figure 6: **Heterogeneity of the Effect of Software According to Initial Wage**



Note: This figure depicts individual treatment effect coefficients for log monthly wages on the y-axis plotted against log initial monthly wage vigintile bins on the x-axis. Initial wage is the wage of an individual in the year prior to treatment. Individual treatment effects estimated via a causal forest. Each point on the graph is the average β coefficient from model (1) for a given log initial monthly wage vigintile. The monthly wage coefficient estimates and initial wage are residualized against a 6-digit occupation fixed effect before plotting.

Wage, age, and education predict larger wage gains. Table 7 shows the OLS estimates of the correlation between the effect of a software on wage and characteristics of each individual. Column 4 shows that high-wage, older, and more educated programmers receive larger wage gains from creating a software.

Table 7: **Treatment Effect Correlations**

	(1) *Wage Estimate*	(2) *Wage Estimate*	(3) *Wage Estimate*	(4) *Wage Estimate*
Log Initial Wage	0.0256***	0.00787***	0.00554***	0.00728***
	(0.000229)	(0.000210)	(0.000225)	(0.000219)
Log Age		0.101***	0.103***	0.104***
		(0.000560)	(0.000561)	(0.000543)
Log Education			0.0580***	0.0442***
			(0.00209)	(0.00204)
Log Firm Size				-0.00278***
				(0.0000433)
R^2	0.358	0.580	0.585	0.611
Observations	61,696	61,690	61,690	61,690

Note: This table presents results of regressions showing the correlations between the causal forest estimated treatment effect on monthly wages as outlined in model (1) against individual and firm characteristics. A 6-digit occupation fixed effect is included. Standard errors are in parenthesis. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$.

Demand for modern programming skills increases inequality among programmers. These results suggest that high-ability individuals are the ones acquiring the skills needed for modern software applications. These skills are highly valued in the job market, leading to larger wage gains for individuals with already high wage. As a result, wage inequality increases both among programmers and between programmers and other occupations.

6 Conclusion

In this paper, we have shown that the growing demand for specialized skills in software programming has significantly increased wage inequality, both within the programming profession and between programmers and workers in other fields. Using a comprehensive dataset

on Brazilian software development, combined with administrative labor market data, we find that creating software greatly boosts wages and career prospects for those with high levels of skill, education, and ability. This effect is especially strong in emerging technologies like AI and cybersecurity.

We highlight two main mechanisms driving wage inequality. First, the selection process into programming favors individuals who are already highly educated and high-wage earners, often concentrated in key technology hubs. Second, the wage gains from software creation are unevenly distributed among programmers. While all programmers benefit, those with specialized skills in frontier technologies see the largest wage increases. This suggests that the demand for new skills amplifies wage disparities within the profession, as top performers in high-demand fields like AI and cybersecurity capture the highest returns.

References

Acemoglu, Daron and Pascual Restrepo, "Automation and new tasks: How technology displaces and reinstates labor," *Journal of economic perspectives*, 2019, *33* (2), 3–30.

_ , David Autor, Jonathon Hazell, and Pascual Restrepo, "Artificial intelligence and jobs: Evidence from online vacancies," *Journal of Labor Economics*, 2022, *40* (S1), S293–S340.

Alekseeva, Liudmila, José Azar, Mireia Giné, Sampsa Samila, and Bledi Taska, "The demand for AI skills in the labor market," *Labour economics*, 2021, *71*, 102002.

_ , José Azar, Mireia Giné, Sampsa Samila, and Bledi Taska, "The demand for AI skills in the labor market," *Labour Economics*, August 2021, *71*, 102002.

Associação Brasileira das Empresas de Software (ABES), "Por que registrar meu software?," 2024. Accessed: 2024-11-14.

Atalay, Enghin, Phai Phongthiengtham, Sebastian Sotelo, and Daniel Tannenbaum, "The evolution of work in the United States," *American Economic Journal: Applied Economics*, 2020, *12* (2), 1–34.

_ , S Sarada et al., *Firm technology upgrading through emerging work*, Research Department, Federal Reserve Bank of Philadelphia, 2020.

Athey, Susan, Julie Tibshirani, and Stefan Wager, "Generalized random forests," *The Annals of Statistics*, April 2019, *47* (2).

Autor, David, Caroline Chin, Anna Salomons, and Bryan Seegmiller, "New frontiers: The origins and content of new work, 1940–2018," *The Quarterly Journal of Economics*, 2024, p. qjae008.

Babina, Tania, Anastassia Fedyk, Alex He, and James Hodson, "Artificial intelligence, firm growth, and product innovation," *Journal of Financial Economics*, 2024, *151*, 103745.

Britto, Diogo G. C., Paolo Pinotti, and Breno Sampaio, "The Effect of Job Loss and Unemployment Insurance on Crime in Brazil," *Econometrica*, 2022, *90* (4), 1393–1423.

_ , _ , and _ , "The Effect of Job Loss and Unemployment Insurance on Crime in Brazil," *Econometrica*, 2022, *90* (4), 1393–1423.

Brooks, Frederick P., *The Mythical Man-Month: Essays on Software Engineering*, Reading, MA: Addison-Wesley, 1975.

Connor, Dylan Shane, Tom Kemeny, and Michael Storper, "Frontier workers and the seedbeds of inequality and prosperity," *Journal of Economic Geography*, 2024, *24* (3), 393–414.

de Souza, Gustavo, "Employment and Welfare Effects of the Quota for Disabled Workers in Brazil," Working Paper Series WP 2023-11, Federal Reserve Bank of Chicago October 2020.

_ , "The Labor Market Consequences of Appropriate Technology," Working Paper Series WP 2022-53, Federal Reserve Bank of Chicago 2022.

_ **and Haishi Li**, "Robots, Tools, and Jobs: Evidence from Brazilian Labor Markets," Working Paper Series WP 2023-42, Federal Reserve Bank of Chicago 2023.

DeMarco, Tom and Timothy Lister, *Peopleware: Productive Projects and Teams*, New York, NY: Dorset House Publishing, 1987.

Deming, David J and Kadeem Noray, "Earnings dynamics, changing job skills, and

STEM careers," *The Quarterly Journal of Economics*, 2020, *135* (4), 1965–2005.

Eeckhout, Jan, Christoph Hedtrich, and Roberto B. Pinheiro, "IT and Urban Polarization," Working paper (Federal Reserve Bank of Cleveland) 21-18 September 2021.

Gerard, François and Gustavo Gonzaga, "Informal Labor and the Efficiency Cost of Social Programs: Evidence from Unemployment Insurance in Brazil," *American Economic Journal: Economic Policy*, August 2021, *13* (3), 167–206.

Iacus, Stefano M, Gary King, and Giuseppe Porro, "Causal inference without balance checking: Coarsened exact matching," *Political analysis*, 2012, *20* (1), 1–24.

Jones, C., "Social and technical reasons for software project failures," 06 2006, *19*, 4–9.

Kelly, Bryan, Dimitris Papanikolaou, Amit Seru, and Matt Taddy, "Measuring technological innovation over the long run," *American Economic Review: Insights*, 2021, *3* (3), 303–320.

Kim, Gueyon, "Trade-Induced Adoption of New Work," Working Papers 2022-007, Human Capital and Economic Opportunity Working Group March 2022.

Kitchenham, Barbara, David Budgen, and Pearl Brereton, *Evidence-Based Software Engineering and Systematic Reviews*, Boca Raton, FL: Chapman & Hall/CRC, 2004.

Lin, Jeffrey, "Technological adaptation, cities, and new work," *Review of Economics and Statistics*, 2011, *93* (2), 554–574.

Marinho, Marcelo Luiz, Suzana Sampaio, Telma Lima, and Hermano Moura, "A Systematic Review of Uncertainties in Software Project Management," *International Journal of Software Engineering Applications*, 12 2014, *5*.

Meyerovich, Leo A. and Ariel Rabkin, "How not to survey developers and repositories: experiences analyzing language adoption," in "Proceedings of the ACM 4th annual workshop on Evaluation and usability of programming languages and tools" ACM Tucson Arizona USA October 2012, pp. 7–16.

_ **and Ariel S. Rabkin**, "Empirical analysis of programming language adoption," in "Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications" ACM Indianapolis Indiana USA October 2013, pp. 1–18.

Mincer, Jacob, "Investment in Human Capital and Personal Income Distribution," *Journal of Political Economy*, August 1958, *66* (4), 281–302.

Moløkken, Kjetil and Magne Jørgensen, "A review of software surveys on software effort estimation," *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.*, 2003, pp. 223–230.

Poliquin, Christopher W, "The wage and inequality impacts of broadband internet," *University of California. Retrieved August*, 2020, *19*, 2021.

Standish Group, "The CHAOS Report," Technical Report, Standish Group International 1994. Available from: https://www.standishgroup.com.

Ţiţan, Emilia, Andreea Burciu, Daniela Manea, and Andreea Ardelean, "From traditional to digital: the labour market demands and education expectations in an EU context," *Procedia Economics and Finance*, 2014, *10*, 269–274.

Torquato, Pablo, "Importance of Registering a Software in Brazil," October 2022. Accessed: 2024-11-14.

van Genuchten, Michiel, *Why is Software Late ?*, Dordrecht: Springer Netherlands,

Wager, Stefan and Susan Athey, "Estimation and Inference of Heterogeneous Treatment

Effects using Random Forests," *Journal of the American Statistical Association*, July 2018, *113* (523), 1228–1242.

Webb, Michael, "The impact of artificial intelligence on the labor market," *Available at SSRN 3482150*, 2019.

A Appendix to Data Section

A.1 Program Types

The list of program types include: Operating System, External Device Management, Hard Drive Management, External Information Exchange, System User Management, Administrative Permissions, Industrial System Management, Network Management, Code Executor, Programming Language, Compiler, Assembly Compiler, Intermediate Compiler, Compiler Translation, Intermediate Processor, Code Interpreter, Procedural Programming Language, Non-Procedural Programming Language, Personal Information Manager, Database Management, Screen Control, Report Generation, Data Description, Data Entry, File Organization, Data Recovery, Data Transmission, Remote Computer Control, Remote Data Transmission, Hardware Management, Data Network Management, Local Network Management, Troubleshooting Software, Word Processors, Electronic Spreadsheets, Graph Generators, Development Support, Self-Generating Software, Computer Aided Programming, Software Development Systems, Code Libraries, Coding Productivity Tools, Coding Documentation Tools, System Replacement Tools, Computer Performance Evaluation Tools, Computer Resource Management, Security & Data Protection, Password Management, Cryptography, Data Consistency, Permissions Management, Simulation & Modeling, Motor Vehicle Simulator, Virtual Machine, Engineering & Robotics Software, Artificial Intelligence, AI Human Replication, Natual Language Processing, Software Performance Metrics, Software Performance Evaluation, Biomedical Device Management, Other Device Management, Automation, Office Automation, Commercial Automation, Banking Automation, Industrial Automation, Industrial Process Monitoring, Robotic Manufacturing Automation, Automotive Electronics, Teleinformatics, Computer Terminal, Data Transmission, Network Coverage, Phone Call Connections, Additional Functions Implementer, Operation & Maintenance Manager, Central Operation & Maintenance Terminal, System Software Management, Data Compressor, Media Converter, Computer Internal Error Correcting, Temporary Data Storage, File Transfer, Functional Software, Planning Software, Control Software, Audit Software, Accounting Software, Technical-Scientific Software, Business Management Software, Pattern Recognition, Image Processing, Entertainment Software, Arcade Games, AI Art Generator, and

Leisure Simulator.

A.2 Application Fields

The list of application fields include: Organizational Development, Government Planning, Organizational Analysis, Public Administration, Business Organization, Factory Planning, Personnel Planning, Material Planning, Asset Inventory, Marketing, Office Services, Agricultural Planning, Agronomy, Rural Farms, Agricultural Business, Other Agriculture, Rural Construction, Soil Conservation, Plant Diseases, Plant Science, Animal Science, Forest Science, Aquaculture, Plant Extraction, Animal Hunting, Social Structure, Social Policy, Community Development, Popular Culture, Theology, Physical Anthropology, Social Sciences, Urban Studies, Rural Studies, Territorial Organization, Demography, Population Dynamics, Other Studies, Biology, Genetic Engineering, Cell Biology, Bacteriology, Human Body Systems, Other Human Body Systems, Biochemistry, Biomechanics, Plant Biology, Botanical Studies, Food, Plant Taxonomy, Metaphysics, Social Science Research, Linguistics, Human Communications, Artistic Creation, Politics, Civil Construction, Construction Methods, Construction Organization, Civil Engineering, Structural Analysis, Construction, Anchoring, Building Maintenance, Hydraulic Work, Earthwork, Law, Constitutional Law, Other Law, Biotic Relationships, Ecophysiology, Ecodevelopment, Plant Ecology, Migration, Economics, Microeconomics, Microeconoimc Theory, Production Economics, National Accounting, Currency, Demand and Supply, Consumer Goods, Economic Engineering, Regional Economics, Capital Ownership, Economic Relations, Monetary Economics, Business Economics, Education, Supplementary Education, Teaching Administration, Teaching Methods, Education Reform, Pedagogy, Energy Policy, Energy Resources, Energy Sources, Technology and Energy, Microelectronics, Nuclear Energy, Public Budget, Financial Institutions, Financial Resources, Financial Administration, Accounting, Particle Physics, Sound, Waves, Measurement, Kinematics, States of Matter Physics, Heat Physics, Quantum Physics, Magnetism, Surface Physics, Radiation Physics, Spectrography, Molecular Physics, Chemical Physics, Polymer Chemistry, Physical-Chemical Analysis, Organic Compounds, Chemical Elements, Physiography, Population Geography, Geographic Zones, Cardinal Points, Astronomy, Planimetry, Aerial Photogrammetry, Photogram, Cartography Methods, Carto-

graphic Plane, Erosion, Cryology, Geotectonics, Marine Geology, Paleontology, Petrology, Geochemistry, Housing, Housing Zoning, Water, Flooding, Fluviometry, Oceanology, Industrial Policy, Technology Policy, Technical Drawing, Mineral Extraction, Processing Industry, Scientifc and Technical, Information Analysis, Photocopying, Material Information, Library Administration, Archival Science, Information Theory, Library, User Profiles, Data Processing, Mathematics, Algebra Mathematics, Geometry Mathematics, Real Analysis, Calculus, Probability, Environmental Policy, Natural Resource Protection, Pollution Control, Environmental Quality, Applied Physics, Atmospheric Science, Climate Science, Soil Science, Soil Formation, Political Theory, Poliitcal System, Social Security, Retirement, Medical, Behavioral Science, Human Behavior, Psychology, Sanitary Engineering, Garbage, Public Cleaning, Water Supply Systems, Sewage Service, Health Policy, Health Administration, Congenital Disease, Disabilities, Outpatient Support, Therapy, Preventative Treatment, Medical Specialties, Biomedical Engineering, Pharmaceutical Assistance, Oral Health, Public Services, Private Insurance, Commercial Policy, Tourism Policy, Telecommunications, Radio Communication, Communication Lines, Telecommunications Equipment, Work Organization, Human Resource Development, Labor Market, Working Conditions, Occupational Structure, Vacations, Transport Policy, Transportation Infrastructure, Freight Transportation, Human Transportation, Transport Modes, Urban Art, Urban Property, Urbanization, Urban Transportation, and Architecture.

A.3 Merging Software Data with RAIS

In this section, we explain how the software dataset is merged with RAIS. From the public records at the Brazilian Patent Office, we can only observe the firm's name and the names of the programmers involved in the software creation. We merge this information with RAIS using both firm and worker names. The merging process follows three steps. First, we compile a list of different spellings for each firm name. Second, we create a list of variations in the spelling of each worker's name. Finally, we use exact matching to merge the two datasets.

Constructing a list of firm names and their ids. We compile a list of different spellings of firm names using data from RAIS and the Firm Registry. Each year, firms in RAIS must report their tax ID, which uniquely identify firms, and name. Because names are often spelled differently by various employees over time and across locations, we create a list of names associated with each tax ID. We supplement this data with official firm names from the Firm Registry, where each firm registers both a legal name and a marketing name. By combining these two datasets, we collect a comprehensive list for each firm, including various spellings, the official firm name, and the legal name.

Constructing a list of worker names and their ids. We follow a similar process to create a list of different spellings for workers' names. In RAIS, firms report the names of all their employees along with their tax ID, which uniquely identifies each worker. Since firms may abbreviate or spell the same worker's name differently, there is more than one spelling of a name associated to each tax ID over time. Exploiting that, we create a list of the various spellings associated with each worker's tax ID.

Constructing a list of firm owner's names and their ids. It could be the case that the firm has some of the firm owners among the programmers. To also find the tax ids of these individuals, we use the Firm Registry dataset to construct a panel dataset with the name of the owners of each firm, their tax id, and the tax id of the firm that they own.

Matching RAIS to the software data. From the previous steps, we have a dataset with 1. different spellings of worker's (owner's) name, 2. the tax id of the worker (firm owner), 3. different spellings of the firm's name, and 4. the firm tax id. We match a software programmer to a worker if 1. the software programmer and the worker have the same name in at least one spelling, 2. the name of the software owner matches the name of the firm that the worker is employed, and 3. each programmer is matched to only one tax identifier.

A.4 Matching Validation

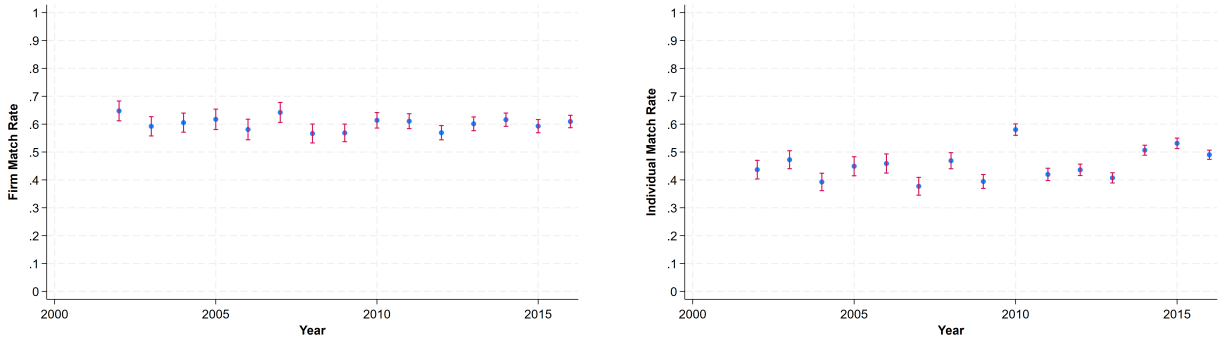
In this section, we compare the characteristics of the software that we can match to a firm id or that we can match to a worker id. We show that the matching rate is stable across time and characteristics of the software.

In panel A of figure A1 we look at the percentage of firms who are missing in each year of our sample. A good matching strategy will have a stable amount of missing values in each year, meaning certain years aren't more reliable for matching than others. We can see the firm missing rate is very stable at around 60%. We can do the same for individual programmers. Panel B of Figure A1 shows an individual missing rate that is also stable around 40-50%.

We compare the matched and unmatched samples along the program type dimension to see if the two groups have very different compositions of program types. Ideally, our matching success is not dependent on what program type the software is labeled as, which would imply a similar fraction of each program type in the two samples. In panel A of Figure A2, we look at the proportion of program types at matched and unmatched firms, and see no glaring differences. We then look at program types by individual programmer in panel B of Figure A2, and again see very stable proportions of each program type in each group.

Finally, we can repeat the above exercise with programming languages instead of program types as the attribute we look at across our matched and unmatched samples. In panel A of figure A3 we look at matched and unmatched firms and once again see a very stable proportion of each programming language in the two groups. We do the same for individuals in panel A of figure A3 and see no notable differences between our matched and unmatched groups.

Figure A1: Firm and Individual Level Missing Rates by Year

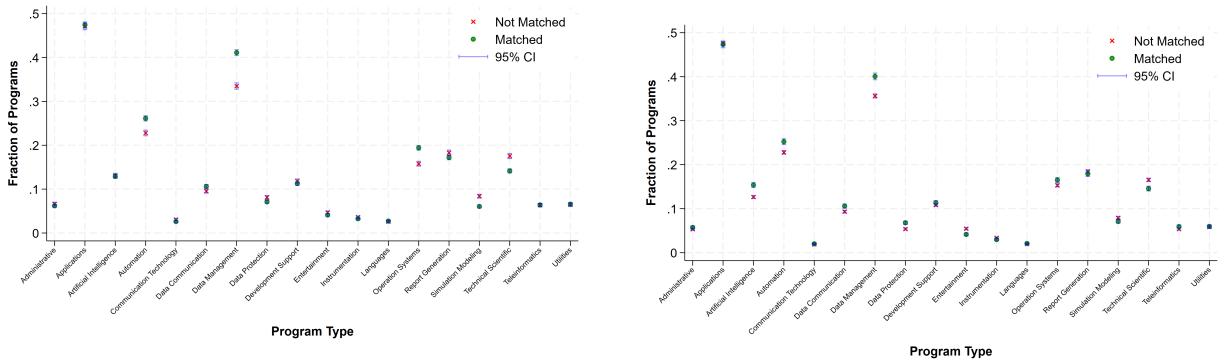


(a) Firm missing rate

(b) Individual missing rate

Note : This figure displays the percentage of firms and individuals, respectively, who have missing values in each year. We see no relationship between match rates and years, giving us confidence that there is no differential likelihood of matching based on year.

Figure A2: Matched vs. Unmatched Program Type Proportions

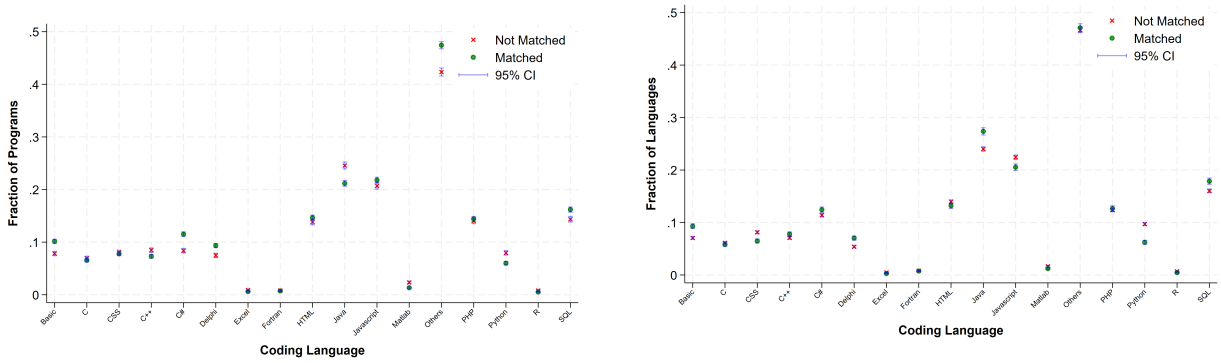


(a): Firm match rate

(b) Individual match rate

Note : This figure plots the percentage of softwares in each program type category for the matched and unmatched groups of softwares. We see no large differences, indicating that there does not appear to be selection into our matched sample based on the program type of the software.

Figure A3: Matched vs. Unmatched Program Language Proportions



(a) Firm match rate

(b) Individual match rate

Note : This figure plots the percentage of softwares written in each programming language for the matched and unmatched groups of softwares. We see no large differences, indicating that there does not appear to be selection into our matched sample based on the programming languages used.

B Appendix to Empirics Section

B.1 Causal Forest for Heterogeneity Analysis

We use casual forests to identify the sources of heterogeneity in our estimates of the effects of publishing software. We estimate the Conditional Average Treatment Effect (CATE):

$$\mathbb{E}[Y_{1,i} - Y_{0,i} \mid X_i = x] \tag{4}$$

where $Y_{1,i}$ and $Y_{0,i}$ are the potential outcomes of programmer i with and without publishing software, while X is a set of observable characteristics. The causal forest approach allows for a fully non-parametric relationship between the treatment effect and the controls X (Wager and Athey, 2018).

As in Britto et al. (2022b), we re-write our main estimating equation from 2 in long-differences as:

$$\Delta y_i = \theta \mathbb{1}_i\{\text{Software Published}\} + \mu_{g(i)} + \epsilon_i \tag{5}$$

We then re-write this again as:

$$\Delta y_i - \mathbb{E}[\Delta y_i | g(i)] = \theta(X_i) \left(I_i\{\text{Software Published}\} - \mathbb{E}[I_i\{\text{Software Published}\}|g(i)]\right) + \epsilon_i \tag{6}$$

where $\theta(X_i)$ is the CATE of the publishing software on the outcome of interest given a set of covariates X_i.

In our case, X_i includes an indicator variable for whether the programmer is located in one of the largest urban Brazilian states[16], their gender, their residualized ability score, and the log of the programmers age, the log of their years of education at the start of our sample, and the log of their initial observed wage in our sample.

In the causal forest approach, $\theta(X_i)$ is calculated as the average of several causal trees.

[16]Large urban Brazilian states are defined as: São Paulo, Rio de Janeiro, Paraná, and Distrito Federal.

Each causal tree is calculated as follows. First, the sample is randomly divided into two groups: one is used to estimate the sample splits (leafs); the other, used for estimation of the CATE, which is called "honest approach". Second, a random set of the covariates X_i is selected. Third, the algorithm searches for a split of the sample to maximize the difference in treatment effects in each of the sub-groups, ensuing that in each leaf there are treatments and controls. Forth, the process continues until the leaf or the heterogeneity in treatment effects between leafs is too small. This process is repeated 1,000 times and averaged out on the estimation sample.

C Appendix to the Summary Statistics Section

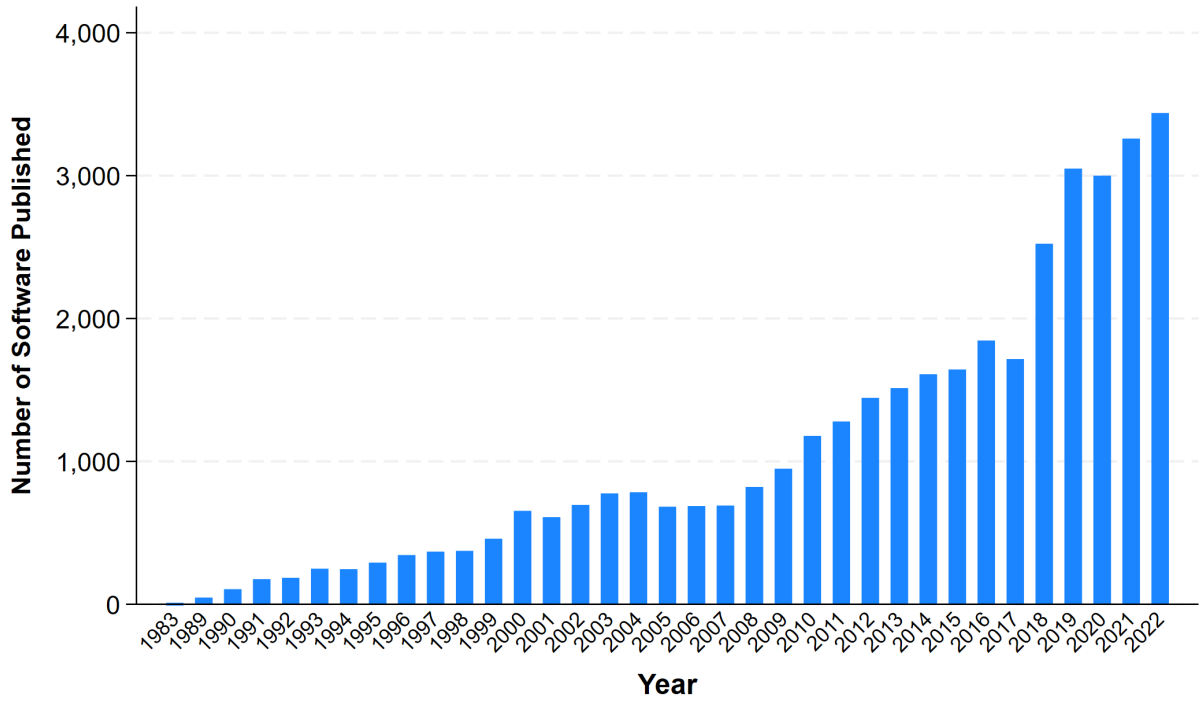
C.1 Additional Summary Statistics

Figure C1 shows the number of softwares in our sample by year. It grows substantially over time from almost zero in 1989 to over 3,000 per year by 2019.

Figure C2 shows which sectors the most softwares originate from. The vast majority come from Support Services (such as IT and administration) and another large fraction come from educational services, which mostly comes from universities.

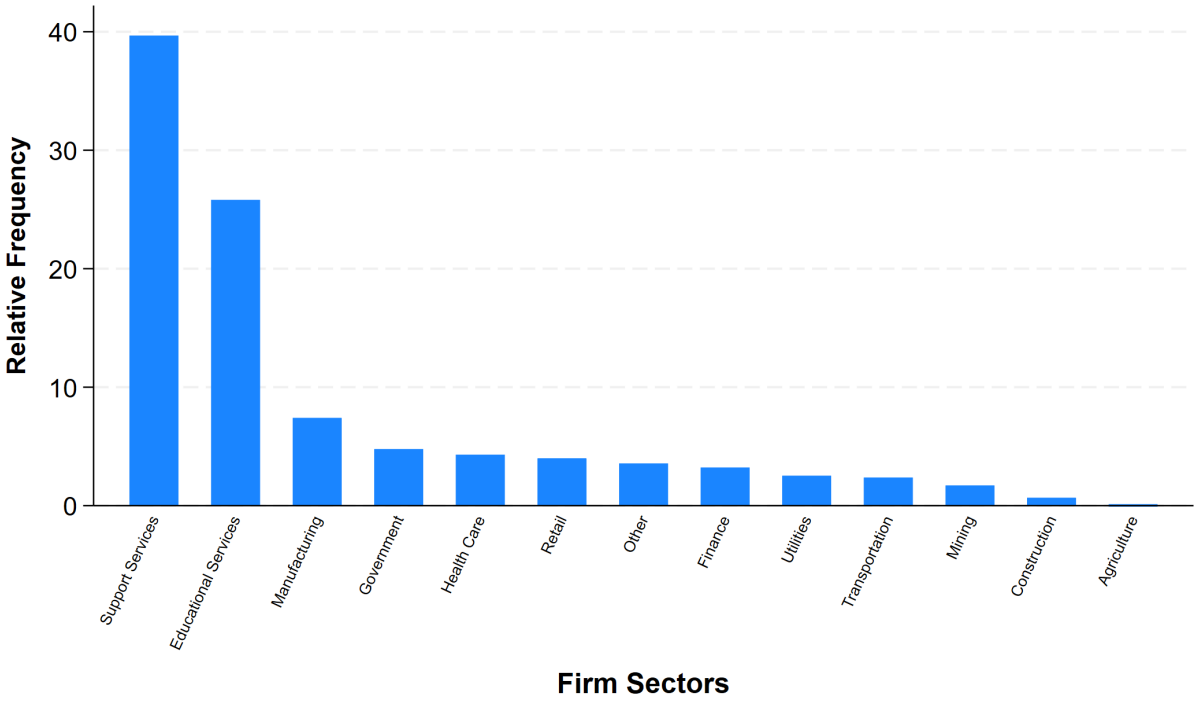
Figure C3 shows the most common use-cases for published softwares. Applications, data management, and automation purposes are the most common.

Figure C1: Number of Registered Software over Time



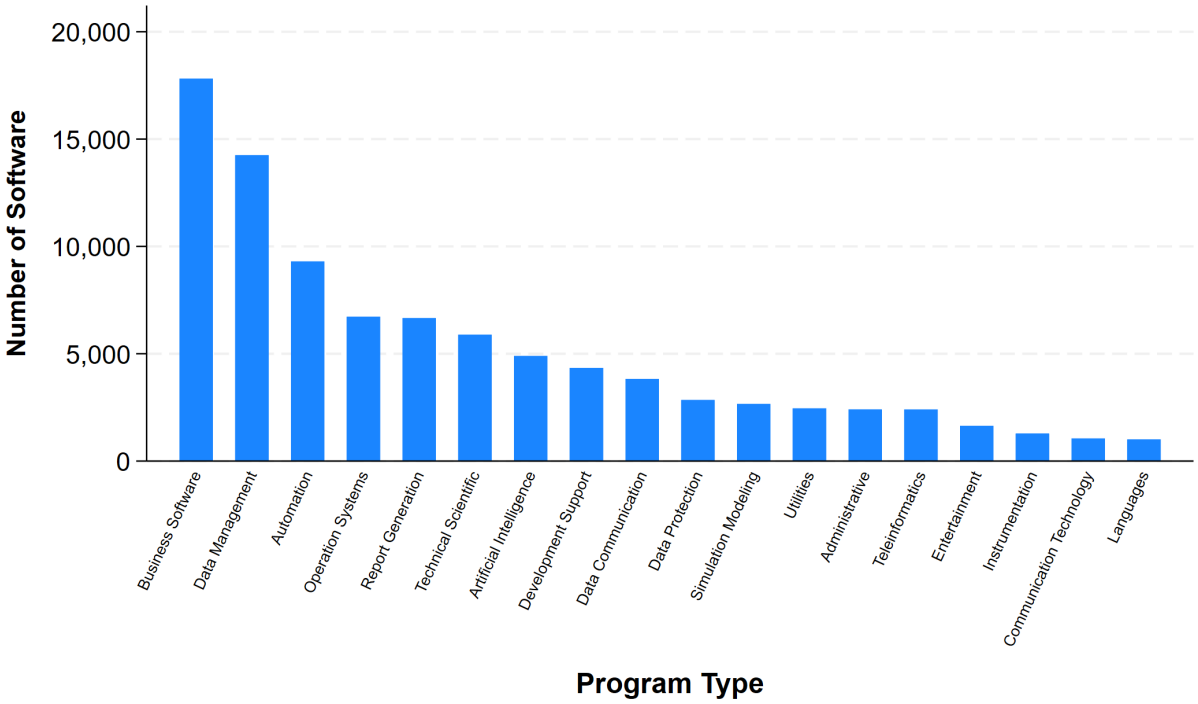
Note: This figure shows the number of software that are registered with the Brazilian patent office each year in our sample.

Figure C2: Common Sectors



Note: This figure shows the relative frequency of each of the sectors that each firm that registered software with the Brazilian patent office belongs to.

Figure C3: Software Program Types



Note: This figure shows the number of software registered with the Brazilian patent office that belong to each of the displayed program types.

C.2 Language Characteristics

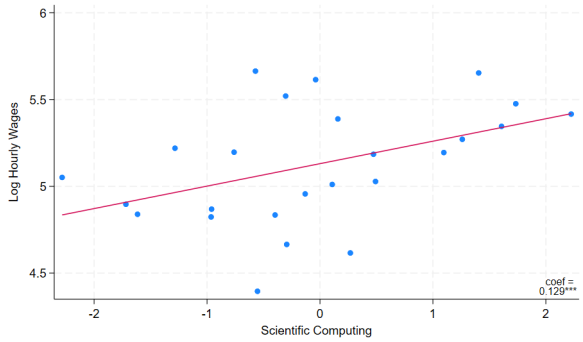
We plot correlations of our primary outcome of interest, hourly wages, against characteristics of programming languages. We obtain these characteristics from a survey from Meyerovich and Rabkin (2013). In this paper, the authors survey programmers on 51 different programming languages, and ask how they rate statements about each language, such as "This programming language is mainstream" and "This programming language is good for scientific computing". Their methodology is described in further detail in an accompanying paper. Meyerovich and Rabkin (2012).

Panel (a) of Figure C4 shows that languages that are suitable for scientific computing, such as Matlab and Fortran, are positively associated with hourly wages as well.

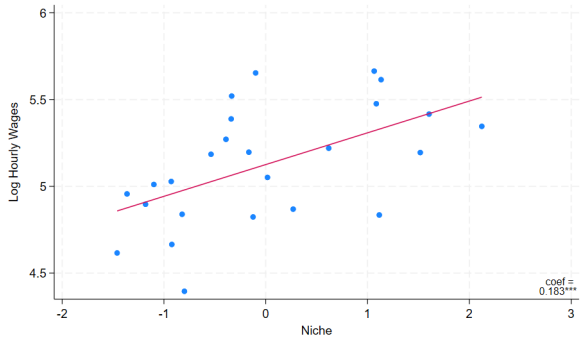
Panel (b) of Figure C4 shows a positive correlation between the hourly wages of programmers and how "niche" a programming language is. Niche languages have a higher wage premium due to less people being able to use them, which is borne out in the data.

Panel (c) of Figure C4 shows a negative correlation between the hourly wages of programmers and how "mainstream" a programming language is, illustrating that more common languages have less of a wage premium due to the amount of programmers with knowledge of it.

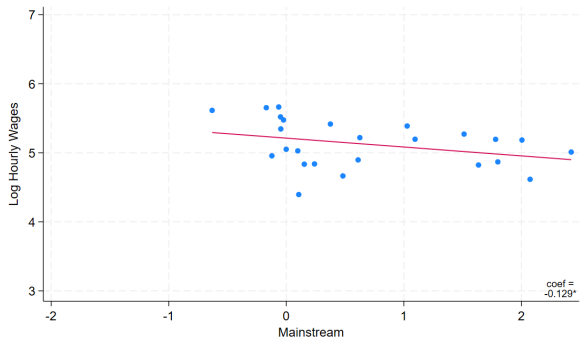
Figure C4: Correlations with Program Language Characteristics



(a) Scientific Computing



(b) Niche



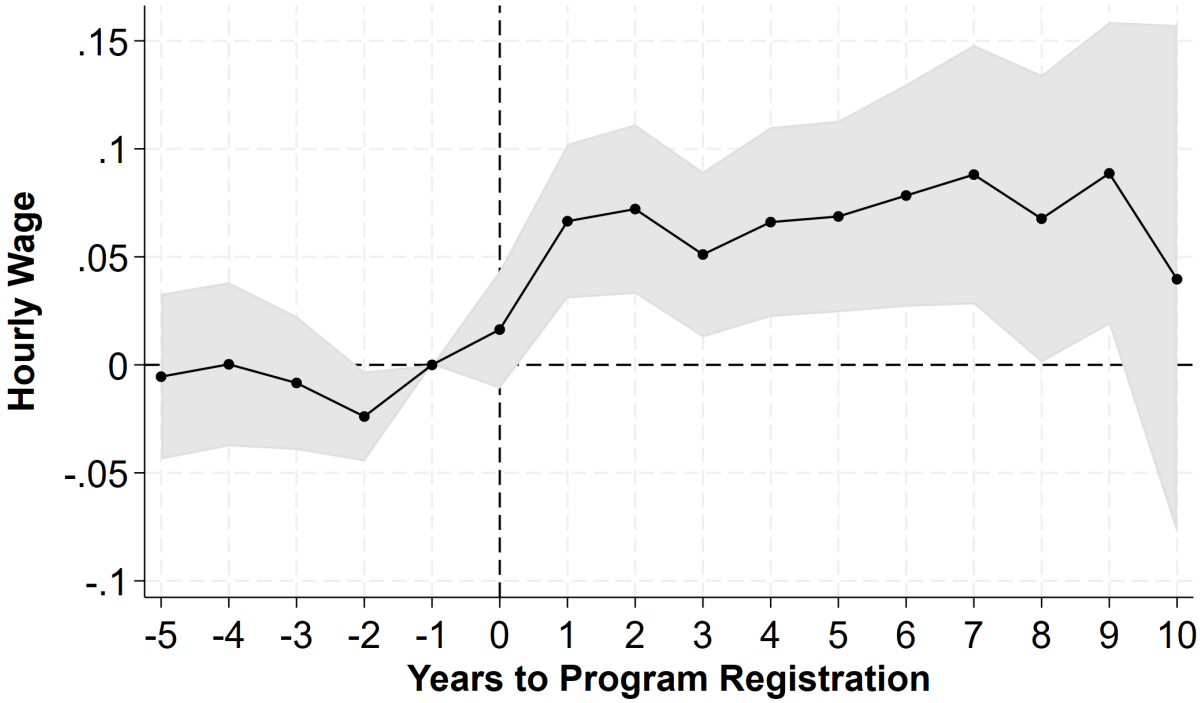
(c) Mainstream

Note : This figure plots programmer log hourly wages against different characteristics of the programming languages used in their softwares. Panel (a) plots log hourly wages against how good the programming languages are for scientific computing, panel (b) plots log hourly wages against how niche the programming languages are, and panel (c) plots log hourly wages against how mainstream the programming languages are.

C.3 Wage Effect Persistence

Figure C5 shows the dynamic response to publishing software over a longer, 10-year time horizon. We can see the response is very stable and significant up to 8 years after the software is published, showing the persistence in gains for these programmers.

Figure C5: Dynamic Long-Run Response of Hourly Wages



Note: The figure above shows the dynamic effects of registering software on hourly wage. The x-axis measures the distance to the year of the software being registered. Each dot is an estimated coefficient and the grey area is a 90% confidence interval. Standard errors are clustered at the individual level.

D Appendix to Results Section

D.1 Mincer Regressions

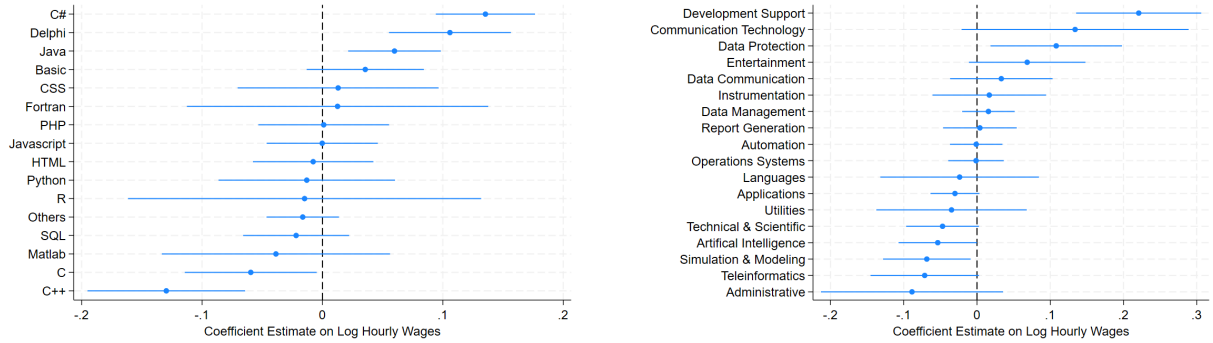
Here we present the results of Mincer earnings regressions (Mincer, 1958). To explore the heterogeneity of softwares on earnings by the coding languages used and the program type of the softwares, we saturate the regression by including dummies for each programming language, program type, and also controls for education, age, and race. We also control for a rich set of fixed effects including 2-digit occupation, region, and individual-level fixed effects. Formally, our estimating equation is:

$$y_{i,t} = \beta_1 \times +\mathbb{1}_i\{\text{Code Language}\} + \beta_2 \times \mathbb{1}_i\{\text{Program Type}\} + X_i + \mu_i + \epsilon_{i,t} \tag{7}$$

where $\beta_1$and β_2 are the coefficients of interest on code languages and program types of the softwares, respectively, X_i is a vector of demographic controls, μ_i are the set of fixed effects, and ϵ_i are standard errors clustered at the individual level. We do not include a treatment indicator in this specification since the indicator variables for code languages and program types only take the value 1 in the treated time periods, making a separate treatment variable redundant.

From the mincer earnings regressions coefficients in figure D1, we can see evidence for selection into more lucrative types of softwares, as high-paying languages like C, C++, and Matlab have negative coefficients when controlling for individual fixed effects, and high-paying program types, such as Simulation and Modeling and Artificial Intelligence also seeing negative coefficients. This evidence of selection motivates our main identification strategy of a differences-in-differences approach to see if certain types of softwares have a causal effect on wages, or if high-ability individuals simply select into certain types of softwares.

Figure D1: Mincer Regression Coefficients



(a) Programming language coefficients

(b) Program type coefficients

Note : Panel (a) shows the coefficient estimates for programming languages of a mincer regression of log hourly wages on indicator variables for each programming language and program type with controls for education, age, and race, along with individual, region, and sector fixed effects. Standard errors are clustered at the individual level. Panel (b) shows the coefficient estimates of a mincer regression for program types of log hourly wages on indicator variables for each programming language and program type with controls for education, age, and race, along with individual, region, and sector fixed effects. Standard errors are clustered at the individual level.