

# Investigating the Adoption of Research Software: From Success Factors to Challenges Faced by Brazilian Academic Researchers


Erica Mourão  [ Fluminense Federal University | [ericamourao@id.uff.br](mailto:ericamourao@id.uff.br) ]

Daniela Trevisan  [ Fluminense Federal University | [daniela@ic.uff.br](mailto:daniela@ic.uff.br) ]

José Viterbo  [ Fluminense Federal University | [viterbo@ic.uff.br](mailto:viterbo@ic.uff.br) ]

Ana Paula Barbosa Sobral  [ Fluminense Federal University | [ana\\_sobral@id.uff.br](mailto:ana_sobral@id.uff.br) ]

Mônica da Silva  [ Fluminense Federal University | [monica\\_silva@id.uff.br](mailto:monica_silva@id.uff.br) ]

Carlos Eduardo Pantoja  [ Federal Center for Technological Education Celso Suckow da Fonseca | [pantoja@cefet-rj.br](mailto:pantoja@cefet-rj.br) ]

## Abstract

Modern research strongly depends on software that is essential in various domains, including engineering, sciences, and other fields, and most often developed within universities and used by the researchers themselves. This software, specifically referred to as research software, presents a unique set of challenges and constraints in its life cycle, methodology, practices, disclosure, and incentives, thereby distinguishing it from conventional software. However, while certain research software developed and used in universities have achieved widespread use and been adopted by the academic community, thereby considered “successful”, others have failed to secure the same level of adoption, resulting in non-adoption and abandonment software leading to an unnecessary expenditure of development time and resources. Gaining a comprehensive understanding of the factors influencing the adoption can assist in mitigating these issues. In this context, the primary aim of this study is to investigate factors influencing the adoption of research software by users and developers, to discern their relative importance, and to explore the challenges associated with fostering the adoption of such software. A survey involving 173 Brazilian academic researchers was executed to gather insights regarding the significance of the factors contributing to the adoption of research software and the obstacles encountered in its adoption. The collected data underwent rigorous statistical analysis, utilizing Independent Sample T-test and Factor Analysis to thoroughly examine the information and provide a ranking of factors. The survey responses were used to rank factors contributing to the adoption of research software, with results stratified by respondents’ profiles. This study suggests that academic researchers can optimize their use of research software by recognizing and prioritizing adoption factors, preventing non-adoption, and overcoming the identified challenges. Understanding the relative importance of each factor can enhance support for research software developers and users and, consequently, promote the broader adoption of research software.

**Keywords:** *Software Engineering Research, Empirical Software Engineering, Research Software, Software Adoption, Challenges, Academia, Universities, Survey Research, Brazilian Academic Researchers*

## 1 Introduction

Modern research strongly depends on software, that is essential in various scenarios, and most often developed within academia and universities and/or used by the researchers themselves (Katz et al., 2019). Researchers develop and/or use software across a wide range of domains, such as the sciences, computing, engineering, and humanities, among others fields, to conduct or support their research, and these software (libraries, tools, and applications) are referred to as research software (Eisty et al., 2018). Examples of research software used by researchers include interactive development environments for research (e.g., Matlab, Jupyter), workflow managers (e.g., Snakemake), frameworks to generate documentation (e.g., Overleaf), and mathematical or statistical libraries and tools (e.g., R, SPSS, scikit-learn) (Eisty and Carver, 2022; Sochat et al., 2022).

In recent years, research software used and developed in academia has been of interest to study. A recent study in the United Kingdom received responses from 417 researchers selected at random from 15 Russell Group universities. They report that 92% of academics use research software, 69% say

that their research would not be practical without it. The report also presents that 46% of those have no training in software development (Hetttrick et al., 2014). Another study expands the UK data, focusing on United States universities, with a survey conducted through 209 members of the National Postdoctoral Association. They report that 95% of academics use research software, 63% say that their research would not be practical without it. The report also presents that 54% of those have no training in software development (Nangia et al., 2017).

Software Engineering Researchers have developed several processes, methods, tools, techniques, and practices that contribute to enhancing the overall quality of software. These include documentation, design patterns, management, version control, and peer code reviews across the complete software life cycle (Heaton and Carver, 2015). However, the academic environment still presents unique differences concerning the software development life cycle (Carver et al., 2016), and that was described more than a decade ago (Segal, 2009).

Although the research community widely acknowledges the importance of research software, the creation, development, and maintenance of research software is still impro-

vised, making such infrastructure fragile and vulnerable to failure (Carver et al., 2022). A prominent issue with research software produced within academia is the varied understanding of research processes, methods, and good software practices among staff, including postdoctoral researchers, faculty members, and students, which can pose challenges to their learning and adherence to good and best software practices (Katz et al., 2019).

In various fields, research software is developed by academics who have differing levels of training, abilities, and access to expertise, resulting in a highly variable software landscape. Researchers often work under significant pressure to maintain proficiency in their respective research domains, leaving them with limited time to keep in touch with the latest software engineering practices (Carver et al., 2022).

Many research software projects have lifespans extending beyond the academic tenure of any postdoc, graduate student, staff, or faculty member. Furthermore, the majority of software is developed in an *ad hoc* manner, without reliance on established frameworks and tools that could potentially reduce the workload. In addition, best practices offering researchers methodologies for efficient development and maintenance are not widely adopted (Katz et al., 2019).

The adoption is often an important goal of any software project. It may be a measure of success for the results of an academic project to be adopted by users or an industrial partner (Tilley et al., 2003). Some empirical studies have investigated how organizations adopt the product with a focus on software development to evaluate its importance and guide the selection of such practices (Hauge et al., 2010; Campanelli et al., 2018).

Nations produce science advances and new technology by investing in scientific and technological research with the anticipation of achieving success that generates economic and social change. The production of science advances and new technology is driven by individuals, who have the natural tendency to seek out novelty and challenges, to explore, to learn, and to achieve goals. With the increasing demands driven by technological advancements, new systems are developed within the academic environment for the benefit of the wider community (Coccia, 2019). This expansion and creation of systems are critically important for academia. However, while certain research software developed and used in universities have achieved widespread use and been adopted by the academic community - thereby considered “successful” - others have failed to secure the same level of adoption, resulting in non-adoption and abandonment of software leading to an unnecessary waste of time and resources.

Given these circumstances, it becomes imperative for researchers (those who develop software in the academic context for use within the same environment and those who use the software to directly conduct or support their research) to understand the significant factors determining the adoption of research software. In this context, the aim of this study is to understand the factors that influence the adoption of research software, to evaluate their significance, and to investigate the challenges associated with promoting their adoption within the Brazilian academic environment.

Our research is validated by empirical investigations that support its findings, with a focus on software engineer-

ing (Wohlin et al., 2012). We carried out a study in two stages: in a previous study, we conducted an exploratory study using interviews with twenty (20) expert Brazilian academic researchers in the field of Computer Science (Mourão et al., 2023a). This approach was employed to identify, gain insights, and understand the factors contributing to successful research software, as well as the challenges that must be overcome.

In this study, we conducted a survey (Wagner et al., 2020) involving 173 academic researchers and discussed the results to probe the understanding of the adoption of research software in Brazilian academia. We used the factors identified in the previous study to discern the significance of each factor. We utilized close-ended questions with a five-point Likert scale to investigate the importance of factors leading to adoption and open-ended questions to capture participants’ perspectives on the challenges. We conducted statistical analyses using inferential methods, which included Independent Sample T-test and Factor Analysis (Fávero and Belfiore, 2017), used to test the statistical differences between the means of two groups (developers and users) and describe variability among observed, scrutinizing the data and generating results, thereby creating an adoption factor ranking. Moreover, we employed the Grounded Theory (GT) (Corbin and Strauss, 2014; Stol et al., 2016) to identify the primary challenges influencing the adoption of research software developed in academia, as indicated by participant responses, and to provide insight into a discussion related to these challenges.

The primary contributions of this work are:

- (i) it validates interview findings in our previous work, where we identify and understand the success factors, the non-success, and the challenges in developing successful research software.
- (ii) it extends the perspective and discussions of adopting research software among Brazilian academics by analyzing the factors that could impact and guide their choices.
- (iii) it provides a ranking of factors that may guide the development of academic research software successfully with topics that will help to community to increase the research software development;
- (iv) it improves the process of generating and sustaining research software development and all research software ecosystem regarding people that create, maintain, and use research software;
- (v) it identifies challenges that influence the success of Brazilian academic research software; and
- (vi) it discusses the findings related to academic research software to provide insight into a situation.

The remainder of this paper is structured as follows. Section 2 provides the necessary background information, and Section 3 provides related works to motivate the research questions examined in this study. Section 4 details the method, including study design, data collection, and data analysis procedures. Section 5 presents a comprehensive overview of the results. Section 6 discusses the key findings derived from the study. Section 7 contemplates potential threats to validity. Finally, Section 8 concludes the paper and outlines future work.

## 2 Background

This section presents some concepts related to research software, research software development and its communities, and software adoption.

### 2.1 Research Software Definition

In recent years, research software has gained greater importance in modern research (Katz et al., 2019). This software is referred to collectively as research software and consists of tools, libraries, and end-user software or applications to conduct or support research (Eisty et al., 2018, 2019; Eisty and Carver, 2022). In our study, we will assume this definition of research software, but due to the aspect of the software and projects, we provide an overview of other definitions.

Research software is commonly used to refer to software used and/or developed in a research context, including and not limited to scientific, non-scientific, commercial, academic, and non-academic research. Two points of view of research software definition can be considered, inclusive and exclusive, which includes source code files, algorithms, scripts, computational workflows, and executables or software services that were created during the research process or for a research purpose (Gruenpeter et al., 2021). To help understand the differences in points of view of research software definition, they use a spectrum between two types of definitions: an inclusive definition, which is closer to a usage point of view, and an exclusive definition, which is closer to a developer point of view. An inclusive definition is related to all code and software artifacts that are used, produced, or might be related to the research process or software that was not necessarily developed with the intention of being part of the research, for example, a library for interfacing with a sensor. On the other hand, an exclusive definition is related to well-identified software that is part of the research discovery process, which might require specialized domain knowledge and is by itself a contribution to science and research, or software that was developed with the intention of being part of the research.

The Journal of Open Source Software (JoSS)<sup>1</sup> is an academic journal with a formal peer review process that is designed to improve the quality of the software submitted and publishes articles about research software. It defines research software as software that: solves complex modeling problems in a scientific context (physics, mathematics, biology, medicine, social science, neuroscience, engineering); supports the functioning of research instruments or the execution of research experiments; extracts knowledge from large data sets; offers a mathematical library; or similar.

A recent work presents a Research Software Encyclopedia that provides lists of criteria and a taxonomy of domains, that a user can make a context-specific choice about a definition of research software. This taxonomy of research software provides a break of research software into sub-groups, and thus empowers people to refer to some subset. The taxonomy of research software is linked to: I) Software to directly conduct research (a) Domain-specific software, (b) General

software); II) Software to support research (a) Explicitly for research, (b) Used for research, but not explicitly for it, (c) Incidentally used for research. Thus, if exists a general definition for research software that is based on its intention, users, and impact on the research space, the taxonomy allows for a user of the definition to scope his or her definition to some subset (Sochat et al., 2022).

### 2.2 Research Software Development

The first definition of the name “Software Engineering” was proposed in 1969 at NATO Conference (McIlroy et al., 1968; Naur et al., 1976) to discuss software development problems. Then, in 1990, the definition of the IEEE Standard Glossary of Software Engineering Terminology was “Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software” (IEEE, 1990; Van Vliet et al., 2008). Research Software Engineering (RSEng) was first formalized by a group in 2012, in the UK, bringing together the definition of “Research Software” and “Software Engineering” with practitioners called Research Software Engineers (RSEs) (Brett et al., 2017; Katz and Hettrick, 2023).

Research Software Development is one of the pillars of RSEng (Cohen et al., 2021). Research software is developed daily by developers in a research center, national laboratories, industries, universities, communities, and practitioners (Lusk, 2018). The development of scientific software differs from the development of more traditional business information systems/IT software, from which many software engineering best practices and tools have been drawn (Hanay et al., 2009). These differences appear at the following phases of the software life cycle: Requirements, Design, Coding, Validation and Verification, and Deployment (Carver et al., 2016).

Researcher groups and institutes are emerging across organizations globally to address the needs around research software and in the life cycle phases of research software development. It is evidenced by the UK Software Sustainability Institute<sup>2</sup>, US Research Software Sustainability Institute (URSSI)<sup>3</sup> funded by National Science Foundation (NSF), that funds research and education in most fields of science and engineering, Netherlands eScience Center<sup>4</sup>, and Software Engineering for Science (SE4Science)<sup>5</sup>. All of them with important publications about research software that can motivate and support the members and collaborators in research software and its development.

Researcher groups undertaken efforts to promote the role of research software. They are calling them Research Software Engineering (RSE) (Baxter et al., 2012; Katz et al., 2019; Brett et al., 2017). Research Software Engineering communities can exist on different levels: local, regional, national, and international, and they exist to support researchers and RSE with the software development aspects of their roles, networking, and best practices for developing research software (Cohen et al., 2021).

<sup>1</sup><https://joss.theoj.org/>

<sup>2</sup><https://www.software.ac.uk/>

<sup>3</sup><https://urssi.us/about/>

<sup>4</sup><https://www.esciencecenter.nl/>

<sup>5</sup><https://se4science.org/>

In the academic context, research software development has received increased attention in recent years. An organizational structures model for research software development at universities that include the use of software engineering practices and process was developed to address the needs around software (Katz et al., 2019). Moreover, interviews were conducted with research software engineer developers from four institutions to examine sustainability from the perspective of research software produced in an academic environment and determine whether there is a shared understanding of what sustainability means (de Souza et al., 2019). A recent survey of the state of the practice presents findings about sustainability challenges that researchers face in developing and using research software. They identified that most research software is developed and managed in ways that are at odds with its long-term sustainability. Some of the findings include a repeated need for more opportunities and time for developers of research software to receive training, the need to cross the software lifecycle, and several types of tools (Carver et al., 2022). On the other hand, an initial common understanding of sustainable software and finding evidence that reveals the importance of promoting sustainable software development was investigated and the perception of sustainable practices for software development in the context of software development from an industry standpoint (Karita et al., 2021). These discussions reinforce the importance of sustainability in the development of research software.

In the development or maintenance of research software, the software might be released as open source (e.g., in academia and national laboratories), or it might be commercial/closed source (e.g., industry, although industry also produces and contributes to open source) (Carver et al., 2022). For open science, it is essential to publish research software in addition to research data (Hasselbring et al., 2020). Research software is fundamental for research, yet significant challenges to discoverability, productivity, quality, reproducibility, and sustainability exist (Barker et al., 2022). During research software development, to make the research software Findable, Accessible, Interoperable, and Reusable (FAIR), we need to understand the FAIR principles, which are originally intended to make data Findable, Accessible, Interoperable, and Reusable to advance the findability, reproducibility, and reuse of research results (Wilkinson et al., 2016). The FAIR for Research Software (FAIR4RS) Working Group<sup>6</sup> has adapted the FAIR Guiding Principles to create the FAIR Principles for Research Software (FAIR4RS Principles). The contents and context of the FAIR4RS Principles are summarised to provide the basis for discussion of their adoption (Katz et al., 2021; Barker et al., 2022).

In Brazil, significant efforts and resources, such as human, financial, administrative, and infrastructure, are devoted to the research software development at Brazilian universities to ensure its success. The National Fund for the Development of Science and Technology (FNDCT) provides the main source of funding for university research at the federal level. Funds from the federated States are managed by research support foundations, present in almost all Brazilian States (Lahorgue, 2006; Neave et al., 2006).

These efforts are maintained through financial resources from high-level institutions, such as development agencies, councils and coordinators, such as (Coordination of Training of Higher Education Staff (CAPES)<sup>7</sup>, The National Science and Technology Council (CNPq)<sup>8</sup>, FAPERJ<sup>9</sup> and others. As part of their activities, they offer scholarships to support the research conducted by researchers and research groups. Over decades, Brazilian organizations have produced more than 18 thousand PhDs and some 55 thousand Master's degrees annually (McMANUS and Nobre, 2017). Moreover, in the last two decade, more than 15,000 research groups was distributed across 268 institutions, 90 percent of all research groups come under the responsibility of universities, isolated colleges, and research centers and 70 percent of all scientists and engineers were employed in universities (Lahorgue, 2006).

### 3 Related Works

We searched the literature to find works that offer a comprehensive overview of research software use and development using the hybrid search strategy (Mourão et al., 2017, 2020; Wohlin et al., 2022). We applied a search string<sup>10</sup> on the Scopus database, filtering the results using our defined inclusion and exclusion criteria and performing parallel backward and forward snowballing iterations on the remaining studies, which tends to present an appropriate balance between result quality and review effort.

Hannay et al. (2009) conducted a survey with 1972 scientists who develop and use research software and focused on the following aspects: (i) the methods and timing of scientists' learning about software development/use, (ii) the importance of software development/use, (iii) the time dedicated to software development/use, (iv) the hardware used for development, (v) the size of user communities, (vi) the familiarity with software engineering concepts, and (vii) the influence of program size, time spent programming, or team size on the importance of good software development practices. The primary conclusions are that knowledge for developing and using research software is predominantly acquired from peers and self-study; the use of supercomputers among scientists is limited; most scientists predominantly rely on software with large user bases; many scientists acknowledge the importance of software testing; and scientists working on larger software development projects and teams tend to rank standard software engineering concepts higher.

Nguyen-Hoan et al. (2010) presented a survey with 60 scientific software developers to identify potential areas for improving scientific software practices. The findings indicate an increased adoption of Integrated Development Environments (IDEs) and version control tools among scientific software developers. The importance of scientific software traceability is not as significant to developers as it is to users. The production of documentation appears to be more widespread

<sup>6</sup><https://www.rd-alliance.org/groups/fair-research-software-fair4rs-wg>

<sup>7</sup><https://www.gov.br/capes/pt-br>

<sup>8</sup><https://www.gov.br/cnpq/pt-br>

<sup>9</sup><https://www.faperj.br/>

<sup>10</sup>“(software engineering” OR “development” OR “use”) AND (“research software” OR “scientific software”)

than previous studies have indicated. Lastly, there remains room for improvement in tool usage, documentation, testing, and verification activities for scientific software development.

Carver et al. (2013) conducted a survey involving 141 respondents from the Computational Science & Engineering (CSE) community has the objective of capturing respondents' perception of traditional Software Engineering concepts and their usage at three different scales: individually, in their teams, and in the CSE community as a whole. The findings revealed low scores for modern Software Engineering best practices, such as code reviews and agile methods, suggesting that collaborative development practices are not being utilized as frequently as they might be needed. This under-usage could explain the relatively low quality of many CSE software projects. Furthermore, CSE practitioners have limited formal Software Engineering training and are mostly self-taught.

Hettrick et al. (2014) conducted a survey with 417 researchers randomly selected from 15 Russell Group universities. They gathered data on the use, development, and training of research software with the aim of describing the characteristics of software usage and development within research domains. Their findings highlighted that researchers recognize software as being crucial for their research, to the point that their work would not be possible without it. Another study surveyed 209 members of the US National Postdoctoral Association to gain insights into the role of software in conducting research at US universities Nangia and Katz (2017). The survey focused on the respondents' usage of research software and the training they had received in software development. Their responses revealed that 95% of respondents use research software. Moreover, 63% stated they could not conduct their research without research software, 31% could do so but with more effort, and 6% would not find a significant difference in their research without research software. Additionally, 54% of respondents had not received any training in software development, although all respondents who developed software for research had received either self-taught or formal software development training.

AlNoamany and Borghi (2018) surveyed 215 respondents to better understand the characteristics of research software and how it is created, used, and shared by researchers. Based on the responses of the participants representing a variety of research disciplines, they found that researchers create, use, and share software in a wide range of forms for various purposes, including data collection, data analysis, data visualization, data cleaning, and organization, and automation. The results indicated that while many participants distinguish between sharing and preserving software, related practices and perceptions often did not align with those of the broader scholarly communications community.

Pinto et al. (2018) conducted a survey of 1,577 R developers who have published at least one R package, using this population as a representative sample for scientific software developers. This study replicates the previous study by Hannay et al. (2009) but was conducted several years later. The results indicated that: (1) scientists who develop software mostly work alone, (2) they independently determine their next tasks, and (3) most of their software development knowl-

edge was obtained from self-study rather than formal education. After that, Wiese et al. (2019) conducted a comprehensive qualitative study based on selected results from a prior survey. They developed a taxonomy of 2,110 reported problems and grouped them into three major axes: technical-related, social-related, and scientific-related problems.

Katz et al. (2019) presented a study that recognized software as an essential component of research and noted the emergence of RSE groups globally to address the needs around scientific software. They presented organizational structure models for research software development at universities, incorporating the use of software engineering practices and processes. They presented three different models for research software development at the University of Manchester, UK, and two US universities.

Another study that we consider relevant is one conducted by Eisty et al. (2018). They conducted a survey of 129 research software developers to gather information about their knowledge and use of code metrics and software process metrics. The results indicated that while respondents generally understood metrics, their knowledge of specific software engineering metrics was limited, and their application of these metrics was even more so. The most frequently used metrics are related to performance and testing. In a subsequent study, Eisty et al. (2019) surveyed 98 research software developers to better understand the use of the software development process for research software. The respondents expressed a strong positive perception of the value of following processes. The study concluded that to produce high-quality and reliable research software, research software developers must follow a proper software development process.

More recently, Eisty and Carver (2022) conducted interviews and a community survey of 84 research software developers to collect information about their current peer code review practices, the difficulties they face, and how they address these issues. The study found that while research software teams review a significant amount of their code, they lack a formal process, proper organization, and sufficient personnel to conduct the reviews.

Another study more recently, Carver et al. (2022), performed a survey of 1,149 researchers, primarily from the United States, about the sustainability challenges they face in developing and using research software. The study highlighted the need for more opportunities and time for research software developers to receive training. They also identified the recurring need for better funding models for research software and for providing credit to software developers to advance in their careers. The results are expected to inform future infrastructure and service support for software developers and users, as well as national research policy aimed at increasing the sustainability of research software.

Based on the papers above that focus on research software use and development, our study is broader and more up-to-date, covering the papers published in the last few years. In terms of goals, our study has the most comprehensive focus since:

- (i) it does not focus on improving scientific software practices (Nguyen-Hoan et al., 2010; AlNoamany and Borghi, 2018); and like (Carver et al., 2013) who focus

- on capture of software engineering concepts;
- (ii) it does not discuss in the context only data on the use and development of research software (Hettrick et al., 2014; Nangia and Katz, 2017), universities organizational structure models for research software (Katz et al., 2019);
- (iii) it does not discuss in the context of the United States knowledge, but instead, we surveyed Brazilian researchers. Furthermore, we use metrics and process metrics by developers (Eisty et al., 2018) and use the software development process for research software (Eisty et al., 2019) and peer review practices (Eisty and Carver, 2022).
- (iv) it does not discuss in only the context of sustainability of research software (Carver et al., 2022);
- (v) and finally, it does not present only development and its problems (Pinto et al., 2018; Wiese et al., 2019).

These previous studies indicate the need and widespread use of research software among researchers but do not delve deeply into the factors that influence its adoption, especially in the academy. Understanding the determinants of adoption would be invaluable for developing software that meets researchers' needs and encourages more widespread use. In addition, challenges related to the sustainability of research software, such as funding models and career advancement for software developers, were highlighted. This study investigates potential solutions to challenges in research software related to influences on the adoption of the software. Finally, although these studies provide an investigation of software engineering best practices, including the use of software processes, it is limited among research software developers and does not present a view of the adoption of research software by users. In this work, we provide a ranking and insights on success factors that determine the adoption of research software in an academic context that guide the development to success.

## 4 Methodology

The primary aim of our research is to investigate the factors that determine the adoption of research software in academic communities, such as universities, using an empirical research method (Wohlin et al., 2012; Creswell, 2014). To identify and investigate these factors, we carried out a study in two stages, as shown Figure 1: first, in a previous study, we conducted a qualitative exploratory study that interviews involved Brazilian researchers around research software; and in the second step we carried out a quantitative explanatory study using a survey to evaluate the factors, identified in the interviews with Brazilian researchers, that leads to adoption or non-adoption of research software, and identify challenges in adoption.

### 4.1 Qualitative Exploratory Study

In a preliminary study (Mourão et al., 2023a), we conducted a qualitative exploratory study involving interviews with twenty Brazilian academic software researchers with experience using and developing research software. This approach

allowed us to gather rich qualitative data regarding the success, non-success, and challenges of developing successful research software. We identified sixteen success factors, grouped into two categories based on the experts' responses: (i) Software Usage and Disclosure Factors and (ii) Software Quality Factors. Additionally, eight non-success factors and five challenges were identified.

Expanding upon these findings, we designed a quantitative follow-up study incorporating a survey distributed among a distinct set of researchers who did not participate in the initial interviews. The primary objective of this subsequent study is not only to validate our interview findings, including the identified factors and challenges but also to gain a deeper understanding of the significance and statistical relationships among the factors influencing the adoption of research software.

### 4.2 Quantitative and Qualitative Explanatory Study

We conducted an explanatory survey and descriptive research, that is, an empirical software engineering research method (Fink, 2003; Wohlin et al., 2012; Wagner et al., 2020) and used a select sample of the population (Sue and Ritter, 2012). We followed the research process aligned with the guidelines defined by Kitchenham and Pfleeger (2008). This approach allows us to validate and assess how the factors identified in the earlier stage of this study could be generalized to the broader Brazilian academic population.

We employed the Goal-Question-Metric (GQM) paradigm (Basili and Rombach, 1988; Basili et al., 1994) to define the goal, questions, and metrics. The GQM paradigm represents a systematic approach in which objectives are defined operationally, and refined into quantifiable questions to extract appropriate information (Basili, 1993).

According to the GQM paradigm (Basili and Rombach, 1988; Basili et al., 1994), the aim of our survey is to *analyze research software with the purpose of characterizing with respect to* determining the level of importance of the factors that influence its adoption *from the point of view of* Brazilian academic researchers in the context of software use and development in academia.

The next step in the GQM approach is to define the questions and metrics that address the defined goal. In this phase of the study, we have formulated the following RQs:

**RQ1. What is the level of importance of factors that determine the adoption of research software developed in academia?** This first research question aims to investigate how research software users and developers prioritize the importance of the factors influencing the adoption of research software. These factors were identified in our previous study as success factors (Mourão et al., 2023a). We have divided it into two sub-questions (RQ1.1 and RQ1.2) relating to categories identified in a previous study, namely (i) *Software Usage and Disclosure* and (ii) *Software Quality*.

**RQ1.1 What is the level of importance of factors regarding software use and disclosure that determine the adoption of research software developed in academia?** This sub-research question aims to investigate how research

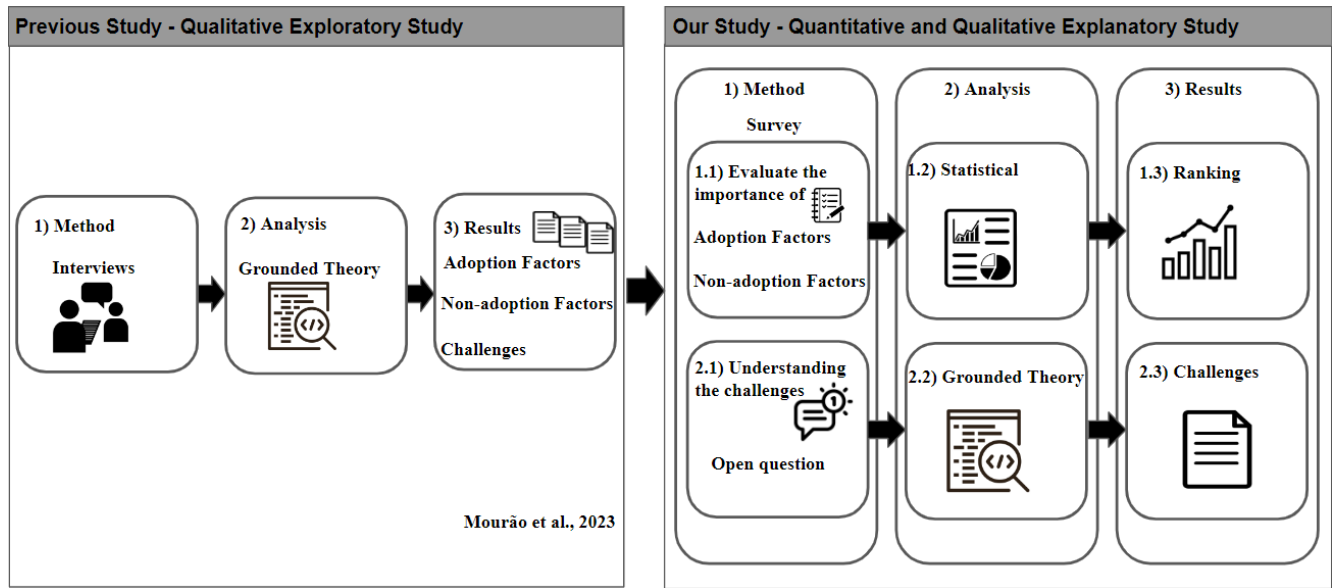


Figure 1. Overview of the study steps

software users and developers prioritize the importance of the factors regarding software use and disclosure.

**RQ1.2 What is the level of importance of factors regarding software quality that determine the adoption of research software developed in academia?** This sub-research question aims to investigate how research software users and developers prioritize the importance of the factors regarding software quality.

**RQ2. What is the level of importance of factors that determine the non-adoption of research software developed in academia?** This second research question intends to understand the level of importance of factors that influence the non-adoption of research software. These factors were identified in our previous study as non-success factors (Mourão et al., 2023a). We need to investigate whether the factors that lead to non-adoption are the opposite of those that lead to adoption.

**RQ3. What are the main challenges that determine the adoption of research software developed in academia?** This third research question examines the challenges in adopting research software. In our previous study, we identified research challenges to determine software success (Mourão et al., 2023a). In this study, we need to understand the challenges that may help drive the development of such software and improve adoption.

To answer these research questions, we planned and conducted an explanatory study using a survey instrument. The metrics were primarily based on close-ended questions, some multiple-choice type questions, and a few open-ended questions. Regarding research question RQ3, we used only an open question to get responses from respondents regarding challenges.

### 4.3 Study Design

The survey instrument consists of a total of twenty questions designed to collect data on topics including research software profile, personal and professional profile, factors that influence both the adoption and non-adoption of research soft-

ware by users and developers and challenges related to the adoption of research software.

This study focuses on the context of the Brazilian academic researchers' demographic and the questions regarding RQ1.1 (RQ1.2 and RQ2) primarily consist of closed-type questions with Single Choice (SC) and some Multiple-Choice (MC) type questions. The first multiple-choice type questions related to which research software use or have used, include a free text option, e.g., "other", so that they can express an option that is not listed. Questions related to the name of an academic research institution and main lines of research were necessary to be open questions. Related to RQ3, an Open Question where respondents can provide feedback about the challenges was provided.

Furthermore, most questions use Likert scales on an ordinal scale from 1 to 5, and define for each a minimum value (e.g., "nothing important"), a maximum value (e.g., "very important"), and a middle point (e.g., "median important"). Additionally, if the respondent lacks knowledge about the subject of the question, they can select the option (e.g., "I don't know"), which is assigned a value of 0.

At the beginning of the survey instrument, we included a consent form and a brief explain what research software is, examples, and what adoption is.

After the consent form, the first survey question asked respondents to select one of two options: "I use and/or work in the development of research software, and I agree to participate in this research" or "I do not agree to participate in this research". If respondents chose not to participate, the survey would end.

The initial question of the survey was intended to ensure that respondents fit the target audience, i.e., researchers who use and/or work in the development of research software. For those who agreed to participate, we presented questions grouped into categories. The survey comprises the following sections:

- *Researcher's profile.* This initial category investigates respondents' backgrounds related to research software,

including which research software they use or have used. Respondents are asked to select and indicate the tools they use or have used, how long they have been using the research software and their frequency of use. Additionally, respondents are asked to self-identify as either users or developers of research software. This category also includes questions about the respondents' personal and professional profiles, which coverage, gender, geographic location, education level, academic performance, academic institution, main research areas, and experience with research software, as shown Table 1.

- *Factors that determine the adoption of the research software - software usage and disclosure (RQ1.1).* This category includes eight mandatory sub-questions related to factors that influence the usage and disclosure of research software. Questions assess the importance of various factors influencing the adoption of research software, such as widespread usage within academia, comprehensive user documentation, citations in scientific articles, publicity in social networks, events and media, and recognition as a reference in the research area, as show Table 2.
- *Factors that determine the adoption of the research software - software quality (RQ1.2).* This category includes eight mandatory sub-questions related to software quality. Questions assess the importance of factors such as functional adequacy, performance, compatibility, efficiency, usability, reliability, security, maintainability, and portability, as shown Table 2.
- *Factors that determine the non-adoption of the research software (RQ2).* This category includes eight mandatory sub-questions related to factors that influence the non-adoption of research software. Questions cover issues such as ease of use, documentation, scientific disclosure, quality, open-source availability, cost, maintenance and continuous evolution, and adoption by researchers, as shown Table 2.
- *Challenges that determine the adoption of the research software (RQ3).* The final category includes an open question where respondents can provide feedback about the challenges that determine the adoption of research software, as shown Table 2.
- *Conclusion of the survey* Respondents are given the opportunity to leave their email address in one open question if they wish to receive a notification of the survey results or to be invited to participate in further stages of the research. The question of contact was "If you would like to be notified about the result and/or invited to the next stages of this research, please inform your e-mail address".

The pilot test of the survey was conducted in August 2021 and involved six researchers affiliated with the study with expert levels of experience in use and development. This testing phase was crucial to ensure the clarity, consistency, and comprehensibility of the questionnaire. Feedback and data were collected regarding the time taken to complete the survey, respondent satisfaction with the research process, and the appropriateness and content of the questions.

The feedback received was generally positive, with re-

spondents considering the number and content of the questions to be in line with the research goals. On average, it took participants 15 minutes to complete the pilot survey. However, the respondents also highlighted minor issues, such as the need for added clarifications to reduce ambiguities, improvements to the readability of some questions, and reorganization of questions into categories aligned with the study's dimensions. Respondents also suggested adapting the questionnaire to better suit the interests and style of the target audience. They recommended applying principles of simplicity, completeness, and neutrality throughout the questionnaire and reminded us to respect the time constraints of potential respondents. Based on this feedback, the final questionnaire was revised to include a request in the invitation for respondents to share the survey with colleagues and friends who also use or work with research software.

#### 4.4 Ethical Criteria and Procedures

In this subsection, we discuss the ethical criteria and procedures adopted in our research, which align with Brazilian guidelines such as Resolution No. 466/2012 (Brasil, 2013) and Resolution No. 510/2016 (Brasil, 2016) from the National Health Council. We implemented the Informed Consent Form (ICF), also known as Termo de Consentimento Livre e Esclarecido (TCLE) in Brazil, issued by the Ethics Committee (CEP) of UFF<sup>11</sup>. Although we did not submit the project to the Ethics Committee, we adhered to the established ethical principles. The ethical criteria and procedures employed in our study, particularly in Article 17 of Resolution No. 510/2016, are highlighted as follows:

- We obtained informed and voluntary consent from research participants. The invitation for participation included a detailed message that identified the responsible researchers, the researcher's supervisors, contact methods, the estimated time to respond to the answers, and the associated institution. Accompanying the invitation, participants received a brief explanation of the study's objectives.
- We ensured participants' total freedom to decide on their participation in the research. Participants were informed, through the ICF, that they could withdraw from the research at any time without any detriment.
- We assured participants of the confidentiality of their identities and privacy. Participants were informed that the collected data would be used for scientific research in the field of software engineering and that all data would be anonymized to prevent participant identification.
- Participants were aware that their contribution was voluntary and without any financial compensation. They were invited to participate in the research voluntarily and optionally through online means (e-mail and social media).

In regards to the procedures for translation and language correction, we observed that the authors utilized translation

<sup>11</sup>Ethics Committee (CEP) - <http://cep.uff.br/>

**Table 1.** Questions and metrics addressed to participants

Categories	Question	Metric
Researcher's profile	1. Which research software do you use or have used and were developed in academia?	Closed question (MC)
	2. How long have you been using research software developed in academia?	Closed question (SC)
	3. How often do you use research software developed in academia?	Closed question (SC)
	4. Regarding your experience with research software developed at the academy, do you consider yourself?	Closed question (SC)
	5. How old are you?	Closed question (SC)
	6. What is your gender?	Closed question (SC)
	7. What state do you live in?	Closed question (SC)
	8. What is your academic background?	Closed question (SC)
	9. What is your academic work?	Closed question (MC)
	10. What is your academic research institution?	Open question
	11. What is your major area of research expertise?	Closed question (SC)
	12. What are your main lines of research?	Open question
	13. How many years of experience do you have in academic research?	Closed question (SC)

<sup>1</sup> MC - Multiple Choice.<sup>2</sup> SC - Single Choice.**Table 2.** Research Questions and metrics addressed to participants

Research Question	Question related to Factors that influence the Adoption - Usage and Disclosure Factors	Metric
RQ1.1	Regarding your experience in the use of software research developed in academia for the academy, please select the degree of importance that each factor influenced the Adoption of software research.	
	1. To be widely used within academia	Closed question (SC) <sup>3</sup>
	2. To be widely used outside academia	Closed question (SC) <sup>3</sup>
	3. Have comprehensive user documentation on different media	Closed question (SC) <sup>3</sup>
	4. Have clear and consistent documentation for software use	Closed question (SC) <sup>3</sup>
	5. Be widely cited in scientific articles	Closed question (SC) <sup>3</sup>
	6. Have articles published about the software	Closed question (SC) <sup>3</sup>
	7. To be disclosed in social networks, events and media	Closed question (SC) <sup>3</sup>
	8. Be recognized as a reference for the research area	Closed question (SC) <sup>3</sup>
Research Question	Question related to Factors that influence the Adoption - Software Quality Factors	Metric
RQ1.2	Regarding your experience in the use of software research developed in academia for the academy, please select the degree of importance that each factor influenced the Adoption of software research.	
	1. Have functional adequacy (completeness, correctness)	Closed question (SC) <sup>3</sup>
	2. Have performance (response time, resource utilization)	Closed question (SC) <sup>3</sup>
	3. Have compatibility (coexistence, interoperability)	Closed question (SC) <sup>3</sup>
	4. Have usability (protection against user errors, learning effectively, efficiently, and satisfied)	Closed question (SC) <sup>3</sup>
	5. Have reliability (maturity, availability, fault tolerance, recoverability)	Closed question (SC) <sup>3</sup>
	6. Have security (confidentiality, integrity, authenticity)	Closed question (SC) <sup>3</sup>
	7. Have maintainability (modifiable, testability, reuse, modularity)	Closed question (SC) <sup>3</sup>
	8. Have portability (adaptability, installability, replaceability)	Closed question (SC) <sup>3</sup>
Research Question	Question related to Factors that influence the Non-Adoption of the Software	Metric
RQ2	Regarding your experience in the use of research software developed in academia for the academy, select the degree of importance that each factor influenced the Non-Adoption of research software.	
	1. Not having ease of use	Closed question (SC) <sup>3</sup>
	2. No documentation on its use	Closed question (SC) <sup>3</sup>
	3. No scientific disclosure about the software	Closed question (SC) <sup>3</sup>
	4. Not having quality	Closed question (SC) <sup>3</sup>
	5. Not having open source	Closed question (SC) <sup>3</sup>
	6. Not being free	Closed question (SC) <sup>3</sup>
	7. No maintenance and continuous evolution	Closed question (SC) <sup>3</sup>
	8. No adoption by researchers/professors	Closed question (SC) <sup>3</sup>
Research Question	Question related to Challenges in Adoption of the Software	Metric
RQ3	In your opinion, what are the challenges in adopting research software developed in academia?	Open question

<sup>3</sup> I don't know = 0, Nothing important = 1, Little important = 2, Median important = 3, Important = 4, Very important = 5.

tools such as Google Translator<sup>12</sup>, as well as the online grammar checker Grammarly<sup>13</sup> to perform the translation of this

article. For the procedures of language correction and agreement evaluation, artificial intelligence tools like Language

<sup>12</sup><https://translate.google.com/><sup>13</sup><https://www.grammarly.com/>

Tool<sup>14</sup> and ChatGPT Version 4<sup>15</sup> were also used.

## 4.5 Data Collection

The survey was conducted through targeted invitations, utilizing a convenience sampling approach, which is a common empirical method in software engineering research (Sjöberg et al., 2005). The target audience for the survey was defined as Brazilian academic researchers at various stages in their careers, including undergraduate and graduate students, professors, scientists, and professionals who either use or are involved in the development of research software.

Email invitations to participate in the survey were sent to (i) Members of the Brazilian Computer Society, (ii) Individuals on email lists at Fluminense Federal University (UFF), which include researchers, teachers, and undergraduate and graduate students, (iii) Coordinators of research projects and professors teaching Computer Science or Information Systems courses at public and private higher education institutions. Moreover, as the researchers conducting the study are affiliated with UFF, social media and WhatsApp groups were used to distribute the survey. Additionally, other researchers knew the authors were contacted to respond and share the survey.

In total, the survey was distributed to 64 federal public higher education institutions, 34 state public higher education institutions, 16 private institutions, and 28 federal institutes. The goal was to reach as many Brazilian academic researchers as possible who fit the target demographic. There were no restrictions based on project size or organizational culture, enabling the inclusion of researchers from a wide geographic range across Brazil. The online nature of the survey made it cost-efficient and allowed respondents to answer questions at their convenience.

The invitation email provided essential information about the study, including its purpose, the rationale for the target audience selection, the importance of participation, the expected time commitment, and a link to the online questionnaire. It was clearly communicated that responses would be anonymized to encourage honest and open sharing of opinions about academic research software. There were no incentives offered to respondents for their participation.

The survey was designed, hosted, and administered using Google Forms<sup>16</sup>, and it remained open for four months from November 2021 to February 2022. The dataset collected in the survey and that support the findings of this study are available in Zenodo repository with the identifier, <https://doi.org/10.5281/zenodo.7362052>. The statistical analysis of the dataset that supports the findings of this study is available on GitHub, <https://github.com/ericamouro/researchsoftware>

Out of the 300 invitations sent, our survey received 173 responses, giving a response rate of approximately 57.66% (173/300). This response rate provides a robust sample for the analysis of factors affecting the adoption of research software in the Brazilian academic community.

## 4.6 Data Analysis

The analysis of survey results involved several steps and assumptions. The following assumptions were adopted to ensure the reliability of the survey results: (i) we anonymized the data, (ii) we cleaned the data to remove incomplete responses, (iii) in the results, the sum of percentages equals one hundred percent for single closed questions with a range of options and the five-point Likert Scale, (iv) we assigned unique identifiers were used for each respondent's quotes, (v) all responses have been filled in completely.

### 4.6.1 Data Analysis of RQ1 and RQ2

Regarding Research Questions RQ1 (sub-questions RQ 1.1 and RQ1.2), and RQ2, statistical analyses were employed to explore the data and generate results, necessitating the preparation of the data for descriptive and inferential statistical analysis (De Sá, 2007; Fávero and Belfiore, 2017).

The first analysis involved frequency counting for questions where respondents could choose one or more options. An example of this would be a question asking about years of experience or duration of use of research software. Frequency counts were extracted, bar charts were created, and Chi-squared tests were performed, which are more appropriate for the data. Chi-squared is commonly used to test statistical independence or association between two categorical variables.

The second analysis is related to the importance rating using the five-point Likert scale (I don't know = 0. Nothing important = 1, Little important = 2, Median important = 3, Important = 4, Very important = 5). The median for each category was calculated, and the top ratings were presented.

The third analysis involved inferential analysis. We analyze the calculation of Cronbach's alpha, a statistic measuring the internal consistency and reliability of the variables (Cronbach, 1951; Nunnally, 1994). This was followed by the calculation of Spearman's Rank correlation coefficient, a nonparametric measure that can be used to summarise the strength and direction of a relationship between two variables. The normality of data was then tested using the Kolmogorov-Smirnov and Shapiro-Wilk tests.

The fourth analysis involved the Independent Sample *t*-test, which compares values from two samples of independent groups and is the most appropriate for the data. We used a nonparametric bootstrap *t*-test with pooled resampling method for comparing paired or unpaired means and for validating the one-way analysis of variance test results for non-normal data in our study (Dwivedi et al., 2017).

Finally, the fifth analysis involved performing Factor Analysis (Johnson et al., 2002) using the principal components method to identify an importance ranking of the factors that determine the adoption of research software. This analysis used the common variance, also known as Communality  $h^2$  (which ranges between 0 and 1), to determine the rank of factors influencing the adoption and non-adoption of research software.

Factor Analysis was carried out in three steps. First, the Kaiser-Meyer Olkin (KMO) and Bartlett's Sphericity tests were performed to verify if the application of factor anal-

<sup>14</sup><https://languagetool.org/>

<sup>15</sup><https://chat.openai.com/>

<sup>16</sup><https://docs.google.com/forms/>

ysis is valid for the variables of each dimension. Second, factor analysis was performed on the factors of each dimension using the principal axis factoring extraction method with Varimax rotation (Kaiser, 1958), which is the most popular orthogonal rotation technique associated to Kaiser normalization (Fávero and Belfiore, 2017). To handle missing values in the data, list-wise deletion was used, a method that 'drop' values and prevents the overestimation of factors within the dataset. Third, communalities were analyzed, representing the total shared variance of each variable across all factors extracted from eigenvalues greater than 1. Factors were then ranked according to their profile. Each item was given a score for each factor, and factor loadings less than 0.4 were suppressed as scores greater than 0.4 are considered stable (Velicer, 1988).

The data collected was initially analyzed using the Statistical Package for Social Science (SPSS) (Sousa, 2002) trial version for Windows, Version 28, SPSS Inc., Chicago, IL, USA<sup>17</sup>. To ensure the evidence's consistency and the possibility of reproducibility of this work, Python version 3.7 was used to analyze all data.

#### 4.6.2 Data Analysis of RQ3

In addressing Research Question RQ3, the Grounded Theory approach was employed, utilizing open coding, axial coding, and selective coding methods (Corbin and Strauss, 2014; Charmaz, 2014; Stol et al., 2016) to identify the primary challenges influencing the adoption of academic research software based on participants' responses. We performed *open coding* to associate codes with quotations of respondents. After that, we related the codes through *axial coding*. In this procedure, the codes were merged and grouped into more abstract categories. Then, we used *selective coding* to identify core categories that best explain challenges. It is essential to clarify that the intention is not to create a theory, as is usual when using GT, but to explore inferential and interpretative meanings to synthesize the data.

The ensuing analysis used triangulation to substantiate the validity of the research findings. Triangulation, a strategy that entails the incorporation of varied sources and methodologies to investigate a common phenomenon (Jick, 1979), is acknowledged as a crucial validation approach in qualitative studies as it engenders diverse perspectives (Flick, 2004). By combining different types of data, the results can be categorized as convergent, complementary, or divergent (Flick, 2020). Initially introduced as a validation instrument, triangulation has since been recognized as a methodological technique that leads to a broad and in-depth investigation (Denzin, 2012). The application of this strategy can promote transparency, rigor, and reliability within the research methodology (Noble and Smith, 2015).

In the context of this study, triangulation was harnessed to ascertain the consistency and congruity of the results, validate and enhance the findings acquired via coding methods, and scrutinize them from manifold perspectives. This methodology ensured an exhaustive and dependable comprehension of the difficulties involved in the adoption of

research software, as outlined in the interviews and literature (Stol et al., 2016). The integration of diverse methodological techniques, including interviews, surveys, and literature review, facilitated the generation of convergent data — data characterized by consistency across techniques — as well as complementary data, in which the outcomes of one method addressed the inquiries posed by another, thereby enriching the overall comprehension of the research subject.

## 5 Results

In the following, we summarise our results structured according to the Respondents Demographics and Research Questions defined in Section 4. This study was responded to by one hundred and seventy three (173) participants. All completed the responses. The dataset of this work is available in GitHub and can be accessed via the link: <https://github.com/ericamourao/researchsoftware>

### 5.1 Respondents' Demographics

The participants' age brackets were as follows: of the total of 173 respondents, a majority (39.3%) fell between the ages of 30 and 39, followed by 26.6% who were between 18 and 29 years old. The next group consisted of 19.7% of participants who were between 40 and 49 years old, 10.4% were between 50 and 59 years old, and 4.0% were 60 years or older. In terms of gender, 27.75% identified as female, 71.10% identified as male, and 1.16% (or two respondents) preferred not to disclose their gender. Figure 2 shows the relationship between the gender and age of the Brazilian respondents. The data suggests that the majority of academics are male and fall within the 30 to 39 years old age bracket.

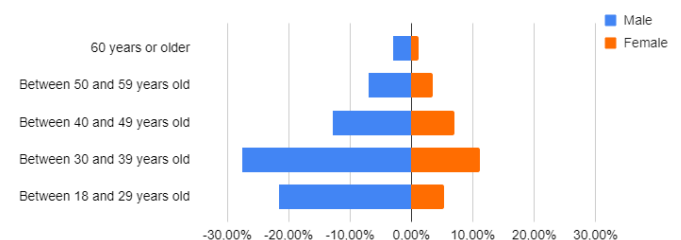


Figure 2. Distribution of age and gender of respondents

In terms of regional distribution, as shown Figure 3, the majority of respondents were from the Southeast region of Brazil (56.07%), which houses a larger number of computing institutions located in São Paulo, Rio de Janeiro, Minas Gerais, and Espírito Santo. The second largest group was from the Central-West region (14.45%). The Northeast and North regions, which include nine and seven states, respectively, each accounted for 10.40% of respondents. Finally, the South region, composed of three states, contributed 8.67% of respondents. As expected, the Southeast region, due to its higher concentration of institutions and academic researchers, produced the most respondents.

Regarding academic qualifications, as shown Figure 4, 97.11% of respondents had attained at least a graduate degree and 72.83% had a master's degree. The breakdown was as follows: graduate (15.03%), specialization (e.g., lato sensu

<sup>17</sup><https://www.ibm.com/br-pt/spss>

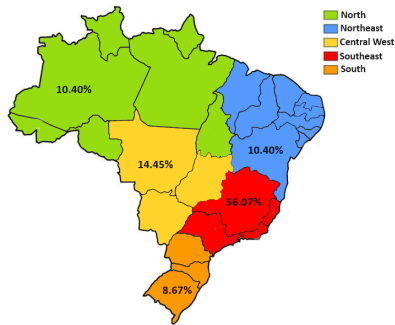


Figure 3. Regional distribution of respondents

Master of Business Administration) (9.25%), Master's degree (stricto sensu) (41.04%), PhD (21.39%), and Postdoctoral (10.40%). A smaller group (2.89%) were undergraduates. In addition, the results indicate a proportional distribution between genders, with a higher number of males at the master's degree level.

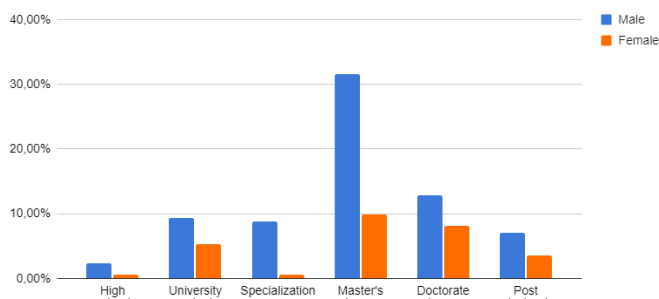


Figure 4. Distribution of age and gender of respondents

The majority of respondents earned an undergraduate degree in the area of Exact and Earth Sciences (93.64%), which included Computing (75.72%), Engineering (9.83%), and a combination of Exact and Earth Sciences (such as Computing or Engineering) with Health Sciences, Human Sciences, Applied Social Sciences, or Biological Sciences (8.09%). Other respondents were from Health Science, Human Sciences, or Applied Social Sciences (6.36%).

Regarding their experience with research software developed in academia, 143 participants (82.66%) identified as users, while 30 (17.34%) considered themselves both users and developers/participants in the development of research software. As for their years of experience in using research software, as shown in Figure 5, the majority had between one and five years of experience (38.15%), which suggests intermediate familiarity with research software. This was followed by researchers with six to ten years of experience (31.21%), who displayed more robust expertise. Those with eleven to fifteen years (9.83%), sixteen to twenty years (6.36%), and over twenty years of experience (10.40%) were also represented. Thus, those with over six years of experience (57.80%) are characterized by specialized expertise in research software. A smaller group with less than 1 year of experience (4.20% of users and 3.33% of users and developers) were considered beginners.

Chi-squared test is commonly used to test statistical independence or association between two categorical variables. We perform the test to determine whether there is an association between categorical variables, for example, years of experience "More than 20 years" and the respondent's role

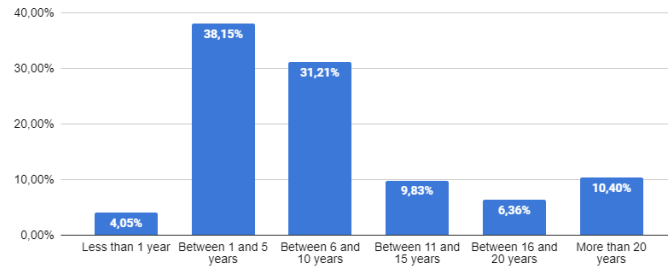


Figure 5. Years of experience using research software

"users". We adjusted the alpha level to control for the probability of committing a type I error, using the chosen significance level (Pituch and Stevens, 2015). Since the p-value is greater than our chosen significance level ( $\alpha = 0.05$ ), we do not reject the null hypothesis ( $H_0$ ), where  $H_0$ : [variable 1] is independent of (or is not associated with) [variable 2], the test shows no significant dependence, and alternative hypothesis  $H_1$ : [variable 1] is not independent of (or is associated with) [variable 2], the test shows significant dependence.

Regarding years of experience using research software, as shown in Figure 5, the Chi-squared test showed no significant dependence between years of experience and the respondent's role as a user or participant in the development of research software ( $\chi^2(5) = 3.470$ ;  $p = 0.628$ ).

Regarding the year of use of research software, shown in Figure 6, most respondents had over six years of use. Others reported three to five years of use (37.57%), one to two years of use (13.87%), and less than one year of use (8.09%). The Chi-squared test showed no significant dependence between the time of use and the respondent's role as a user or participant in the development of research software ( $\chi^2(3) = 5.866$ ;  $p = 0.118$ ).

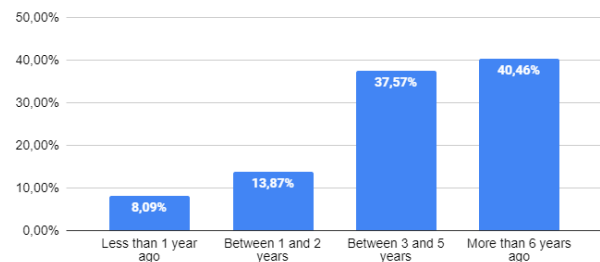


Figure 6. Years of use of research software

Regarding the frequency of use of research software, as shown the Figure 7, most respondents (38.73%) often use research software, followed by those who always use it (30.64%). A sizable group (26.01%) reported occasionally using research software. As expected, we observed high usage of such software. The Chi-squared test showed no significant dependence between the frequency of use and the respondent's role in relation to the research software ( $\chi^2(4) = 5.537$ ;  $p = 0.237$ ).

Regarding the distribution of years of experience and years of use of research software, as shown the Figure 8, we analyze the influence of the participants' years of experience on their years of using research software. Results show that respondents with one to five years of experience have been using research software for between three and five years. Similarly, those with over six years of experience have primarily been using research software for more than six years. We cal-

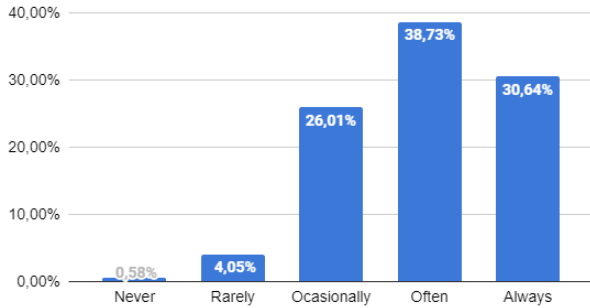


Figure 7. Frequency of use of research software

culated the Spearman's correlation coefficient ( $r$ ), which is a method of testing the strength (.00-.19 "very weak", .20-.39 "weak", .40-.59 "moderate", .60-.79 "strong", .80-1.0 "very strong") and the direction, positive or negative, of the correlation between two variables, for example, years of experience "More than 20 years" and the use of research software "More than 6 years ago". The null hypothesis ( $H_0$ ), where  $H_0$ : the correlation coefficient  $r = 0$ , means that there is no correlation, and the alternative hypothesis ( $H_1$ ), where  $H_1$ : the correlation coefficient  $r \neq 0$ , means that there is a correlation. The results suggest that the strength of the correlation is strong and it is a positive correlation between "years of experience" and "use of research software" as indicated by Spearman's correlation ( $r = 0.611$ ;  $p < 0.001$ ), where  $p$  is  $p$ -value using the chosen significance level ( $\alpha = 0.05$ ).

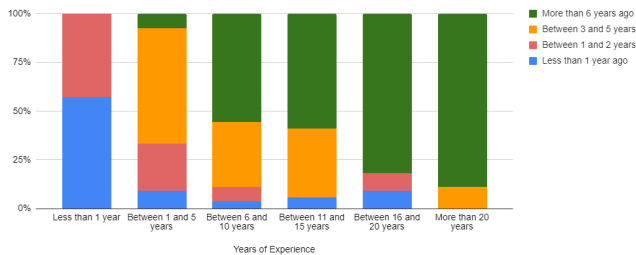


Figure 8. Distribution of years of experience and years of use of software research

Regarding the distribution of education level and years of use of research software, as illustrated in Figure 9, respondents with a master's degree have been using research software for between three and five years. Those with doctoral or postdoctoral degrees have been using research software for over six years. There is a strong positive correlation between "years of experience" and "level of education" according to Spearman's correlation ( $r = 0.609$ ;  $p < 0.001$ ).

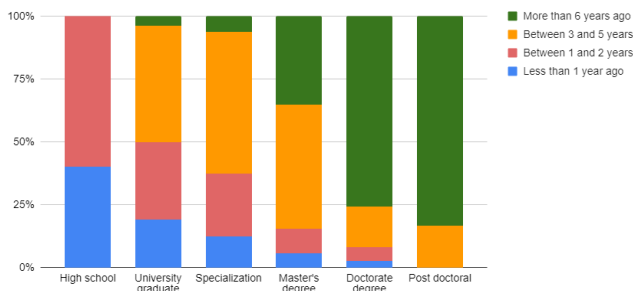


Figure 9. Distribution of level of education and years of use of software research

The following subsection describes the analysis of our Re-

search Questions (RQ).

## 5.2 Importance of Factors in Adopting Research Software (RQ1)

This section presents the results of RQ1, which was divided into two sub-questions (RQ1.1 and RQ1.2). The factors were sorted in increasing order according to the proportion of the top two ratings: Very important (Vi) and Important (I).

### 5.2.1 Software Use and Disclosure Factors in Adopting Research Software (RQ1.1)

Based on the descriptive analyses, factors with a median score above three are considered influential in terms of importance. As illustrated in Figure 10, eight factors related to software usage and disclosure were ordered.

The results show that of the total of 173 respondents, 80.93% of the participants (47.40% Vi and 33.53% I) consider the factor "Have clear and consistent documentation for using the software" to be highly influential, suggesting that it is a crucial factor impacting the adoption of research software. This is supported by the highest median score. Following this, 72.25% of the participants (31.79% Vi and 40.46% I) believe that the factor "Be widely used within the academy" significantly affects the use and disclosure of research software. Furthermore, 69.94% of the participants (34.10% Vi and 35.84% I) consider the factor "Be recognized as a reference for the research area" to be of great importance. Lastly, 67.63% of participants (29.48% Vi and 38.15% I) believe that the factor "Have comprehensive documentation in different media for the user" is of substantial importance.

Moreover, 37.57% of the participants (15.03% Vi and 22.54% I) consider the factor "Be widely cited in scientific articles" as influential in importance, while 37.00% (12.14% Vi and 24.86% I) believe that "Having articles published about the software" is important for the use and disclosure of research software. Lastly, 32.37% of participants (9.83% Vi and 22.54% I) regard "Being widely used outside academia" as important, and 26.58% of participants (10.40% Vi and 16.18% I) find "Being publicized on social networks, events, and media" to be important for the use and disclosure of research software.

Regarding the inferential analyses, the Cronbach's alpha, a statistic measuring the internal consistency and reliability of the variables, was above 0.7, which is considered satisfactory. The Kolmogorov-Smirnov and Shapiro-Wilk test results indicate non-normal distribution of data for all variables where  $p$ -value is ( $p < 0.001$ ), using the chosen significance level ( $\alpha = 0.05$ ). The nonparametric bootstrap Independent Student's  $t$ -test with pooled resampling method showed differences in participant scores for some factors between users and developers. However, the Independent Student's  $t$ -test was not statistically significant.

The Kaiser-Meyer-Olkin (KMO) test obtained values of 0.741 (users) and 0.679 (developers), indicating that a principal component analysis is suitable. Bartlett's sphericity test for the category "Usage and Disclosure" of research software showed values of ( $\chi^2(28) = 421.953$ ;  $p < 0.001$ ) for users and ( $\chi^2(28) = 100.357$ ;  $p < 0.001$ ) for developers using the

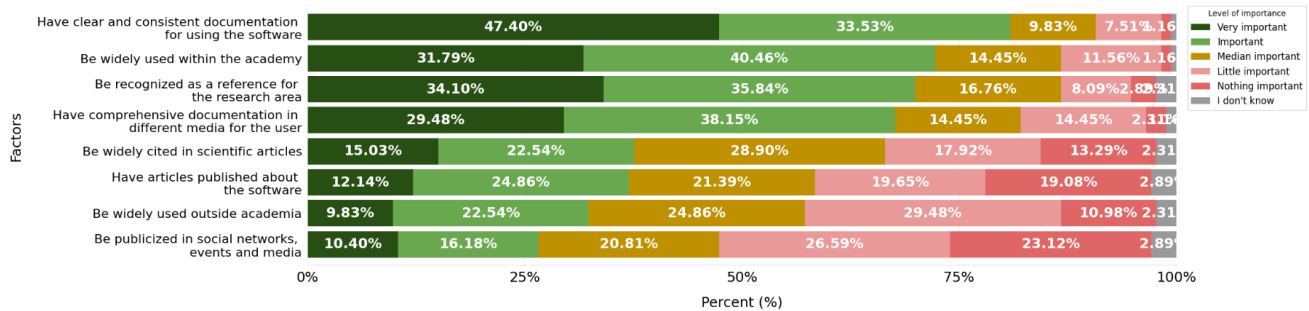


Figure 10. Factors of usage and disclosure that determine the adoption of research software

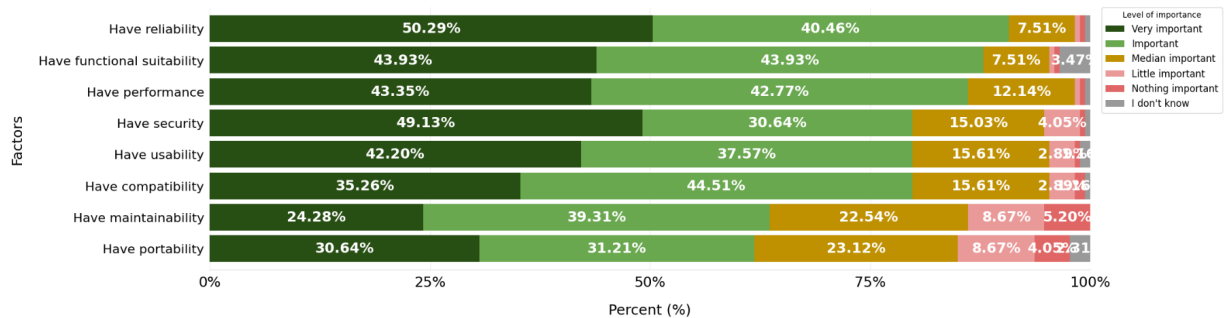


Figure 11. Factors of software quality that determine the adoption of research software

chosen significance level ( $\alpha = 0.05$ ). Thus, Factor Analysis is quite adequate.

Factor Analysis was performed on the factors of each dimension, and a principal component factor analysis with Varimax rotation was conducted. The rotated factors, which will be called F1, F2, and F3, and which represent the highest percentages of shared variance by the original variables, accounted for 30.11%, 22.04%, and 18.76% of the total variance (70.91%) for users and 30.41%, 18.48%, and 25.89% of the total variance (74.78%) for developers. The corresponding eigenvalues were greater than 1 and were used to compute the communality.

Finally, we analyzed the communality, which represents the total shared variance of each variable across all factors extracted from eigenvalues greater than 1. The factors were ranked by profile as shown in Table 3.

### 5.2.2 Software Quality Factors in Adopting Research Software (RQ1.2)

Regarding the descriptive analyses, the results indicate that factors with a median above three are considered influential in importance. As shown in Figure 11, eight factors related to software quality were sorted.

The results reveal that of the total of 173 respondents, 90.75% of the participants (50.29% Vi and 40.46% I) deem the factor “Have reliability” as influential in importance. This is followed by 87.86% of the participants (43.93% Vi and 43.93% I) who indicate that the factor “Have functional suitability” is essential for software quality in research software. Subsequently, 86.12% of the participants (43.35% Vi and 42.77% I) consider that the factor “Have performance” is crucial. This is succeeded by 70.77% (49.13% Vi and 30.64% I) who assert that the factor “Have security” is important.

Additionally, 79.77% of the participants (42.20% Vi and

35.57% I) indicate that the factor “Have usability” is important, and again 79.77% of the participants (35.26% Vi and 44.51% I) state that “Have compatibility” is crucial for software quality in research software. Finally, 63.59% of the participants (24.28% Vi and 39.31% I) consider that the factor “Have maintainability” is important, and 61.85% (30.64% Vi and 31.21% I) indicate that “Have portability” is crucial in the software quality of research software.

Concerning the inferential analyses, we employed the same statistical analysis procedures presented in the previous Research Question (RQ1.1).

The participant’s scores differed in some factors between the users and developers, and the Independent Student’s *t*-test was statistically significant. The *t*-test revealed that “Have usability” ( $p = 0.015$ ) and “Have compatibility” ( $p = 0.015$ ) scored significantly differently between users and developers with a 95% of confidence.

We performed a Factor Analysis using the principal components method to identify an importance ranking of the factors. The KMO test obtained a value of 0.854 (users) and 0.753 (developers), and the Bartlett’s sphericity test for Software Quality dimension showed the values ( $\chi^2(28) = 359.821; p < 0.001$ ) for users and ( $\chi^2(28) = 86.229; p < 0.001$ ) for developers. Thus, factor analysis is quite adequate.

The rotated factors, F1, F2, and F3, are formed by the highest percentages of variance shared by the original variables, for users (in this case 30.32%, 15.09%, and 24.82%) of the total variance of 70.24%, respectively, and developers (28.72%, 23.65%, 22.64%) of the total variance of 75.02%. The eigenvalues corresponded to values greater than 1 and will be used to create the communality.

Finally, we analyzed the communality, which represents the total shared variance of each variable across all factors extracted from eigenvalues greater than 1, and ranked the fac-

**Table 3.** Ranking of factors related to software disclosure and usage that determine the adoption by users and developers

Profile	Factor of Software Usage and Disclosure	F1	F2	F3	Communality
Users	Have clear and consistent documentation for using the software		0.923		0.851994
	Have comprehensive documentation in different media for the user		<b>0.800</b>	0.407	0.805784
	Be widely cited in scientific articles	0.895			0.800771
	Be widely used outside academia			0.873	0.762194
	Have articles published about the software	0.818			0.669142
	Be publicized in social networks, events and media	0.464		0.661	0.652485
	Be recognized as a reference for the research area	0.707			0.499548
	Be widely used within the academy	0.427	<b>0.429</b>		0.365822
Developers	Have clear and consistent documentation for using the software		0.928		0.861862
	Have articles published about the software	0.914			0.834600
	Be widely used within the academy	<b>0.601</b>		0.587	0.705759
	Be widely cited in scientific articles	0.831			0.691024
	Be widely used outside academia			0.827	0.683372
	Be publicized in social networks, events and media			0.807	0.651830
	Be recognized as a reference for the research area	<b>0.635</b>		0.421	0.581078
	Have comprehensive documentation in different media for the user		0.659		0.434792

tors by profile as shown in Table 4.

### 5.3 Importance of Factors in Non Adopting Research Software (RQ2)

Regarding the descriptive analyses, the results indicate that factors with a median above three are considered influential in their importance. As shown in Figure 12, eight factors were arranged in increasing order, as was done in RQ1.

The results reveal that of the total of 173 respondents, 89.02% of the participants (61.27% Vi and 27.75% I) consider the factor “Lack of Quality” as influential in importance. This is followed by 82.66% of the participants (52.60% Vi and 30.06% I) who indicate that “Lack of Ease of Use” is important. Then, 78.03% of the participants (43.93% Vi and 34.10% I) deem the factor “Lack of Documentation about the Usage” as important. This is succeeded by 74.57% of the participants (47.98% Vi and 26.59% I) who consider the factor “Lack of Freedom (Not Free)” as important.

Additionally, 72.83% of the participants (38.15% Vi and 34.64% I) state that the factor “Not having Maintenance and Continuous Evolution” is important, and 61.85% of the participants (30.06% Vi and 31.79% I) consider “Not having Adoption by Researchers/Professors” as crucial to the non-adoption of research software. Finally, 46.83% of the participants (21.97% Vi and 24.86% I) regard the factor “Not having Scientific Disclosure about the Software” as important, and 30.05% of the participants (16.18% Vi and 13.87% I) find “Not having Open Source” important for the non-adoption of research software.

Concerning the inferential analyses, we employed the same statistical analysis procedures presented in the previous Research Questions RQ1.1 and RQ1.2.

The participants’ scores differed in some factors between users and developers, and the nonparametric bootstrap Independent Student’s *t*-test was statistically significant. The *t*-test revealed that “Not having documentation about the Usage” ( $p = 0.009$ ) and “Not having Open Source” ( $p = 0.026$ )

scored significantly differently between users and developers with a 95% confidence interval.

We performed a Factor Analysis using the principal components method to identify an importance ranking of the factors. The KMO test obtained a value of 0.810 (users) and 0.615 (developers), and the Bartlett’s sphericity test for non-adoption of research software showed the values of users ( $\chi^2(28) = 331.691; p < 0.001$ ) and developers ( $\chi^2(28) = 56.383; p < 0.001$ ). Thus, factor analysis is quite adequate.

Afterward, we performed a factor analysis on the factors of each dimension, similar to what was done in RQ1. The rotated factors for users accounted for 26.51%, 21.66%, and 19.67% of the total variance of 67.85%, respectively, and for developers, they accounted for 25.03%, 20.72%, and 20.42% of the total variance of 66.18%. The communality is shown in Table 5.

### 5.4 Challenges in Adopting of Research Software (RQ3)

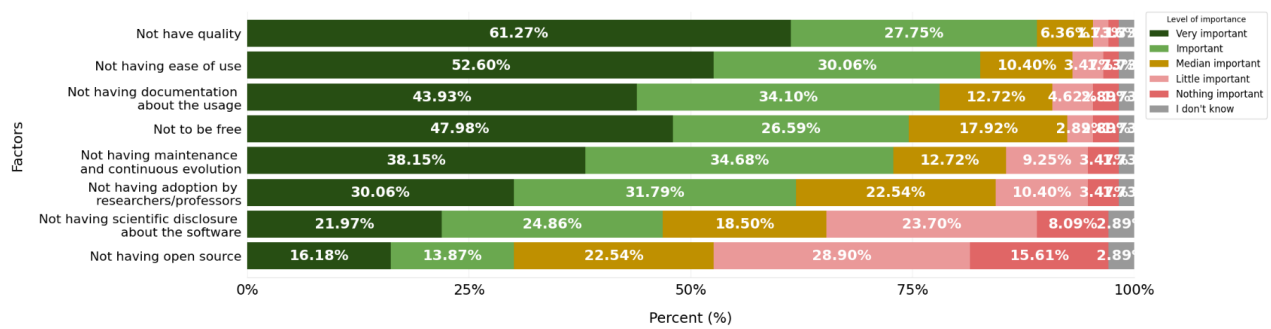
This question aims to acquire data for conducting a triangulation with the data obtained from interviews conducted with research software experts in order to verify and substantiate the information gathered. Additionally, it seeks to ascertain if there were any other challenges not mentioned by the interviewees.

We identified challenges related to research software adoption as shown Table 6 and 7. The challenges shown in Table 6 are aligned with those identified in our previously published study, where data were collected through twenty Brazilian academic research experts interviews (Mourão et al., 2023a). In this previous study, a target population included Brazilian researchers and professors who are users and or work in software research development. They have experience in research software, domain, and knowledge in their academic projects.

Therefore, the results present in Table 6 corroborate the assertions regarding previously identified challenges, such as: “Funding”, “Maintenance”, “Engagement”, “Disclosure”,

**Table 4.** Ranking of factors related to software quality that determine the adoption by users and developers

Profile	Factor of Software Quality	F1	F2	F3	Communality
Users	Have functional suitability		0.929		0.862206
	Have usability	0.436		<b>0.744</b>	0.743267
	Have reliability			0.856	0.732645
	Have portability	0.824			0.679557
	Have maintainability	0.791			0.624942
	Have performance	<b>0.555</b>	0.431		0.493613
	Have security			0.690	0.464518
	Have compatibility	0.682			
Developers	Have usability	<b>0.758</b>	0.502		0.827547
	Have functional suitability		0.898		0.805712
	Have portability			0.875	0.765805
	Have performance	0.430	<b>0.738</b>		0.729495
	Have compatibility	0.547		<b>0.596</b>	0.654566
	Have reliability	0.786			0.618608
	Have maintainability			0.718	0.515514
	Have security	0.706			0.497449

**Figure 12.** Factors that determine the non-adoption of research software**Table 5.** Ranking of factors that determine the non-adoption by users and developers

Profile	Factor of Non Adoption	F1	F2	F3	Communality
Users	Not having adoption by researchers/professors			0.820	0.672157
	Not to be free	0.815			0.663462
	Not having scientific disclosure about the software			0.802	0.643801
	Not having open source		0.763		0.582907
	Not having documentation about the usage		0.760		0.577650
	Not having maintenance and continuous evolution	<b>0.553</b>	0.438		0.497271
	Not having ease of use	0.694			0.481248
	Not have quality	0.657			0.431186
Developers	Not having adoption by researchers/professors	<b>0.652</b>		0.599	0.783759
	Not having documentation about the usage	0.868			0.753445
	Not have quality			0.860	0.739538
	Not having open source	0.460	<b>0.716</b>		0.723322
	Not to be free		0.806		0.649769
	Not having maintenance and continuous evolution	0.665			0.442186
	Not having scientific disclosure about the software			0.611	0.373321
	Not having ease of use				0.000000

and “Quality”.

The term “Funding” is related to the lack of funding and incentives for research or lack of alignment with real issues in an academic environment. In academic research, it is crucial to have funding for research and this resource to be maintained. “Maintenance” is related to have maintenance after the development and implementation period. “Engagement” is related to people engagement in adoption and maintenance.

“Disclosure” is related to and “Quality”

Participants answered open-ended questions in the survey implemented in this study. Respondents in this survey are users and/or developers and have different levels of knowledge in research software. We extract key challenges from feedback from participating research software developers and users. Among the total, we selected some more expressive comments from which the categories were generated.

**Table 6.** Challenges that determine the adoption of research software - similar to previously

Challenge	Comment of respondents
Have funding.	<i>"The biggest challenge is economic, followed by quality software. A lot of useful software lacks quality because it doesn't have adequate financial incentive for development." (P21), "The challenges are inherent in the lack of investment in dedicated teams dedicated to the development and maintenance of applications. Often the development of software in the academy is limited to the execution of a team of no more than three members, since the applications, in general, are the result of theses and dissertations by a single professor." (P119), "... investments are needed in communication platforms, good service to users, promotion of updated and adequate tutorials to use the software." (P61), "Partnerships, ..., and investments." (P14), "Some tools also have limited costs or licensing, which may require investments for full use." (P107), and "...it needs to increase public investment, the incentive to create certain software." (P07)</i>
Have maintenance.	<i>"...As a developer, the biggest difficulty is to maintain the developed software. As most of the software is developed in research/TCC projects, after the student leaves, a replacement with knowledge of the technology and the domain must be found (this replacement is usually not easy and in some cases it is not feasible)." (P19), "... Most software developed at the academy does not have maintenance after the promotion period. This makes promising software not properly maintained." (P26), "Continuity of these tools (evolution) and support to users." (P63), "Maintain maintenance and be able to answer user questions." (P76), "The continuity of excellent academic projects, which end up ending and the software does not have continuity..." (P104), and "Lack of a broad community and ongoing support for the software." (P84)</i>
Have engagement.	<i>"end users are more interested in its use." (P03), "depends almost exclusively on the interest of the student/researcher." (P65), "The desire to learn something new and get out of the comfort zone." (P86), "I believe that the main challenge for software developed at the academy is the recognition of the potential and its possible adoption by the academy itself, since it is currently easy to create a system. I imagine the hard part is to maintain and get recognition especially for new tools." (P124), "Fostering researchers' interest in learning a new tool that will indirectly help in research (e.g. less time or easier to do something) when they are concerned with learning new concepts and techniques that will directly influence research." (P126), and "Fear of novelty." (P41)</i>
Have disclosure.	<i>"More dissemination, not only in the scientific community already formed, but also among students in training. The incorporation of these types of software from an early age, since graduation." (P37), "Disclosure: Sometimes we don't know about a particular software until we read about it in an article or hear about it at an event. Perhaps social networks can become a more frequent source of systematic dissemination of these resources." (P87), "There are few ways to publicize the systems made by the academy, in my opinion betting on scientific journals for this is a serious mistake on the part of the academy. ...they do not have visibility outside the academy or its niche." (P89), "Wide-scale distribution and dissemination. Export to communities other than academic." (P93), "Lack of disclosure and practicality in inserting the tools into the routine." (P106), and "Community support, in the sense of publicizing, encouraging use, training workshops." (P109)</i>
Have quality.	<i>"...code not available/closed; low usability." (P02), "Difficulty in use and not finding documentation..." (P28), "Ease of use, simplified reporting and not consuming so much processing resources." (P33), "...development of more user-friendly interfaces to involve a greater number of researchers." (P34), "People don't know it, sometimes it's difficult to use because ... it doesn't have good usability and ease of learning. Sometimes it's not so intuitive." (P39), "The main challenge is to develop solutions that are good in terms of functionality and quality of use, ensuring their continuity and availability in an open and free way. With these characteristics achieved, it is possible that the adoption will be promoted by the scientific communities themselves." (P48)</i>

Due to text length constraints, we have restricted ourselves to featuring six comments per challenge.

From a total of 143 participants, 113 identified as users, while 30 considered themselves both users and developers/participants in the development of research software. Moreover, those with over six years of experience demonstrated specialized knowledge in research software.

Table 7 shows challenges not previously mentioned in our study of interviews. These findings complement the earlier ones, introducing new assertions related to the challenges identified earlier, such as: "Understanding User Needs", "Training", "Documentation", and "Cost".

The term "Understanding User Needs" is related to understanding what exactly researchers need. "Training" is related to specialization courses in handling learning and knowledge of tools. "Documentation" is related to extensive and detailed documentation for users and developers. Finally, "Cost" is

related to gratuity and paid to have research software.

## 6 Discussions

In this section, we discuss the implications of our findings from RQ1 (RQ1.1 and RQ1.2), RQ2, and RQ3. Our study identifies gaps between the topics explored in academia and practices implemented in research software. We offer suggestions to bridge this gap in the following subsections.

### 6.1 Implications of Adoption Factors

**Software Use and Disclosure Factors.** The results for RQ1.1 of the study show that the ranking of factors determining adoption places "Having clear and consistent documentation for using the software" in the first position. Moreover, as

**Table 7.** Challenges that determine the adoption of research software - not was previously

Challenge	Comment of respondents
Understand the needs.	<i>"The activities of each area and often of the different groups within each area often differ considerably and standardization by a generic software tool ends up being unfeasible." (P11), "find the right software for each problem. Adaptations often need to be made." (P32), "The main challenge is to find the most suitable for your needs." (P38), "Identify what exactly researchers need and what software meets those needs." (P42), "I believe the biggest challenge is for the software to be "generic" enough to meet different demands. They are often developed for such a specific niche, or to meet such a specific need, that they end up not being used. In addition, they are often developed in a way that does not have as many differences in relation to what already exists." (P57), "Lack of standardization/specification: lack of specific software that can be used to accelerate scientific production in a given area. Usually the software is more generic to cover a larger audience, implying the lack of specific tools for each area, which have metrics, standardizations and differentiated evaluation processes In the end, generic software cannot fully meet any research group." (P74)</i>
Training.	<i>"Knowledge of the tools, there is a lack of indications, disclosures and video-class material such as a user manual and for what purposes such software is proposed." (P54), "High level of domain, software variability and diversity, specialization courses in handling learning and software flexibility and adaptability." (P60), "It takes time to get to know and use the tools quickly. Time is often wasted learning the functionality..." (P118), "Generally, the knowledge of which software to use is passed on through advisors and colleagues, but there is no training to use the software." (P68), "The biggest challenge would be the beginning, that is, how much time is needed to understand the tool and if this can complicate the development of the research." (P51), and "Often the system is made for a particular area and all the semantics that the system carries is taken as tacit, that is, many terms that are super familiar in one area are totally obscure in another." (P50)</i>
Documentation.	<i>"Insufficient or scarce documentation..." (P02), "...In addition to making the software available, a usage guide must be made available. It doesn't have to be extensive and detailed documentation, but rather a quick guide that teaches the new user how to install and integrate the tool into the research environment...Finally, I consider that support for the use of the tool is extremely important, regardless of whether the support is provided by the community or by the developers themselves.." (P46), "Code replication and dissemination and excellent documentation." (P56), "Extensive or unattractive documentation, which does not facilitate learning about the use of the software." (P69), "Interoperability and accessible documentation are the two most sought after qualities of software of this type." (P91), and "People don't know it, sometimes it's difficult to use because it doesn't have documentation or a guide that makes it easy to use..." (P39)</i>
Cost.	<i>"Gratuity is one of the main problems for the use/adoption of research software." (P67), "Being paid and difficult to use." (P58), "It's often the cost." (P75), "Easily organize and visualize results, create connections between results/documents, some are paid." (P62), "I believe there are several challenges. The first one is to make the software available in a public repository, so that anyone can access and use the software. We often find articles that describe software, but without directing a link to get the software." (P46), and "Discontinuity, change to billing and with that expenses..." (P71)</i>

shown Table 3, from the point of view of both user and developer profiles, the importance of comprehensive and understandable documentation for the use of research software is recognized. A recent study indicates features of sustainability, such as software documented, that in particular converges on the theme of factors important for adoption research software (de Souza et al., 2019) and other finding out even more insights and evidence that may reveal the importance of promoting the sustainable software development field including good practices of Software Engineering such as clear and consistent documentation (Karita et al., 2021). Thus, this corroborates the importance of the success factor that determines the adoption of research software.

Regarding the factor "Being widely used within academia", it ranks second in importance as shown Figure 10. However, the most detailed ranking analysis present that users perceive this as less important than developers do, as shown Table 3. Developers working in academia understand that for software to be adopted, it must be extensively used within academic circles. Corroborating this evidence, a study that discusses the importance of scientific software indicates that scientists developers spend on average about 50% more of their total work time developing research software and 100% more of their total work time using scientific software than expected (Hannay

et al., 2009). Our results suggest that some users utilize the software but do not participate in its development within academia. Therefore, their perception of its importance is lower, as they are not necessarily part of the academy to adopt the software.

The factor "Having comprehensive documentation in different media for the user" ranks second in importance for users but is considered less significant by developers as shown Table 3. Developers do not consider it crucial to provide documentation in multiple media formats. However, for users, access to documentation across various platforms, such as computers, cell phones, CDs, e-books, online content, and physical books, is deemed essential. Industry studies also reveal that technical documentation holds significance for developers and can pose challenges for software maintenance (Santos et al., 2022).

The factors "Being recognized as a reference for the research area" and "Being publicized on social networks, events, and media" hold the same position in the ranking for both profiles as shown Table 3, indicating a strong consensus on their importance, but in lower positions. Understanding the role of FAIR principles for research software can maximise research value and its importance to provide impact in works (Barker et al., 2022). The authors highlight the importance of the FAIR4RS Principles and the positive signals

of adoption that demonstrate high levels of community support. We suggest that the use of FAIR4RS, such as a globally unique and persistent identifier for software, for example, software versioned with DOI from Zenodo, and rich meta-data that includes the identifier that is searchable and indexable, could provide the research software more recognized as a reference for the research area to achieve the successful. Regarding publicizing, events such as congresses and social networks, such as instagram, youtube, and facebook, would help to reach out to users in the community to adoption of research software. We suggest that developers have a guide to help this be one of the development items.

Finally, regarding “Be widely cited in scientific articles” and “Have articles published about the software”, the ranking has differences between the profiles, as shown Table 3. We understand that users search for articles cited in the literature, and developers aim to publish work related to research software. However, software citations should be of the same importance as citations of other research products, such as publications and data (Smith et al., 2016). Moreover, academic researchers at all levels, including students, postdocs, faculty, and staff, should be credited for the software products they develop and contribute to, particularly when those products enable or further research done by others (Smith et al., 2016). We suggest that the increase of the importance of these factors could be done also by following Software Citation Principles and FAIR movement (Katz et al., 2021; Barker et al., 2022).

**Software Quality Factors.** The results for RQ1.2 of the study show that the ranking of factors determining adoption indicates that “Having functional suitability” followed by “Having usability” are top concerns for users. Functional suitability refers to the extent to which a product or system provides functions that meet both stated and implied needs under specified conditions. Usability represents how effectively, efficiently, and satisfactorily a product or system can be used by specific users to achieve specific goals in a specified context. Both factors are in the first positions. We can note that while our results related to software quality confirm the factors presented in the work of (Nguyen-Hoan et al., 2010) published over a decade ago, such as factors related to reliability, functionality is still the most important as shown Figure 11 and in both profiles analyzed, as show Table 4.

The “Maintainability” factor, which refers to the effectiveness and efficiency with which a product or system can be modified, corrected, or adapted to environmental and requirement changes, is considered less important by developers compared to users. This indicates that users are more inclined to adopt software that offers consistent maintenance. This result differs from a previous study where portability ranked third (Nguyen-Hoan et al., 2010). Conversely, developers do not place significant emphasis on this factor, and studies in the Brazilian industry have identified it as a challenge (Santos et al., 2022).

Finally, the “Security” factor, which refers to the extent to which a product or system safeguards information and data to ensure appropriate levels of data access based on authorization, is a critical aspect to consider. Both profiles, as indicated in Table 4, currently rank last among developers and second-to-last among users. However, it is crucial to ac-

knowledge that the insufficient focus on security in these profiles may lead to system vulnerabilities, resulting in unauthorized data access and the potential disclosure of sensitive user information (Rahman and Farhana, 2021). Therefore, it is essential to address these security and privacy concerns comprehensively, especially since academic systems must also comply with data protection laws in the countries where they will be utilized (da Silva et al., 2018).

## 6.2 Implications of Non-Adoption Factors

The ranking of factors that determine the non-adoption indicates the first position, the factor “Not having adoption by researchers/professors” is the same for both profile users and developers. On the other hand, the factor “Not to be free” is the second for users but the fifth for developers. Sharing research software in an open repository, after the project ends, increases the chance it will be found and reused (de Souza et al., 2019) and may help to provide the adoption of research software.

Regarding the factor “Not having open source” is in the same position for both profiles. For users, the factor “Not having scientific disclosure about the software” is more important to determine the non-adoption than “Not have quality” for users; that is different for developers. Software is a product of research, and by not citing it we leave holes in the record of research of progress in those fields (Smith et al., 2016), as show Table 5.

Finally, the factors “Not having maintenance and continuous evolution” are the same position for users and developers. Followed by “Not having scientific disclosure about the software” by developers and “Not having ease of use” by users. Then, “Not have quality” by the users are the factors that developers believe in the last position of importance that determine the non-adoption. The sustainability of software concerns characteristics of the software artifact itself, and includes factors relating to how the code is written and documented, and the software quality (de Souza et al., 2019). Moreover, the software is developed, maintained, and published in different ways than data, often in the open on development platforms that encourage sharing and collaboration (Katz et al., 2021).

Our study shows that many factors of non-adoption are what inversely lead to adoption. Therefore, focusing on adoption factors and avoiding non-adoption factors can significantly impact software research usage.

## 6.3 Implications of Adoption Challenges

The triangulation identified in this study produced convergence and complementarity. Convergence indicates a high degree of overlap and consistency between data sets collected in the previous study (using interviews) and the new study of the same phenomenon using a survey.

Regarding convergence, we identified five challenges: (i) having funding, (ii) having maintenance, (iii) having engagement, (iv) having disclosure, and (v) having quality, as shown Table 6. Complementarity indicates that the research results from different methods enhance each other. We identified four challenges in this regard: (i) understanding the

needs; (ii) training, (iii) documentation, and (iv) cost, as shown Table 7.

We discussed the complementary challenges found in our previous study (Mourão et al., 2023a). Therefore, we will now discuss the new challenges found in this work.

*Understanding the needs.* pertains to identifying exactly what researchers need and which software meets those needs. Often, software is more generic to cover a larger audience, resulting in a lack of specific tools for each area. The application of requirements elicitation could support the identification of these needs before developing the research software (Segal, 2009). Moreover, more than a decade ago, the success of an iterative and incremental model of development is still heavily dependent on users being effectively engaged and giving meaningful feedback at the end of each iteration (Segal, 2009).

*Training.* In the context of research software, knowledge about which software to use is typically transferred through advisors and colleagues, but formal training on using the software is lacking. Studies present that there is a general lack of formal training in programming and software development among scientists; training that scientists do receive is often supplied by a computer science department, which gives general software courses that scientists might not see the relevance of, and scientists may not see the need for more formal training in software development. (Hannay et al., 2009). Other studies also an evident lack of training in software development present among researchers (Nangia et al., 2017). Providing such training for use and development could enhance the quality of software produced in research.

*Documentation.* Generally, there doesn't need to be extensive and detailed documentation, but rather a quick guide that instructs the new user on how to install and integrate the tool into their research environment. We suggest that comprehensive and clear documentation across different types of media could enhance user adoption of the software (de Souza et al., 2019; Karita et al., 2021).

*Cost.* This challenge is typically associated with the availability of free software. The software can be accessible through a public repository, enabling unrestricted access and usage. While specific software may offer free functionalities, more critical features are often restricted and require payment for access. It is important to note that the evaluation of technical debt, which affects maintenance, testing, and documentation, was beyond the scope of this study and remains an ongoing challenge for software developed in academia (Detofeno et al., 2023).

We want to emphasize that the context greatly influences the solution and that these elements interact with each other. For instance, a software that is customizable to address a specific market segment or a strategy that targets a market with low costs and focuses on a niche for very high quality. Another perspective involves introducing practices and recommendations for developers who are academic researchers and users of research software.

In general, we believe that future studies should expand on this research by conducting a more comprehensive survey of researchers in Brazil. We recommend that future studies work to assess the quality of software produced in research settings. These studies could provide evidence that software

is important in modern research and are starting points for future research.

## 7 Threats to Validity

Though we aimed to mitigate the threats to the validity of our study methodology, some decisions may have influenced the results of the survey. We will discuss the threats to validity based on the framework presented in (Wohlin et al., 2012):

Concerning *External Validity*, which pertains to the generalizability of our results, the population and sample size of the respondents is based on the number of emails sent to researchers. Given that 173 out of 300 invitees from educational institutions contributed to the survey, the sample size equates to roughly 57.66% (173/300). This sampling rate suggests our results are representative of the population we aimed to generalize.

With regards to *Internal Validity*, which speaks to causality, we adhered to the guidelines proposed by Kitchenham and Pfleeger (2008) and analyzed other similar questions in surveys to create new questions aligned with our research. We invited motivated researchers and volunteers to participate. Therefore, we believe surveys are a valid approach.

Regarding *Construct Validity*, which pertains to the generalizability of the study results, a potential threat lies in the context of Brazilian academics. To mitigate this, we sought a broad sample for the survey, evident in the extensive number of participants. However, it's possible that the study participants are not representative of the broader population of research software developers and users. As we collected perceptions that determine adoption and non-adoption, these could vary from person to person. Thus, the results of this study may not be generalizable to all research software users and developers.

Concerning *Conclusion Validity*, which speaks to the reliability of our findings, the statistical test had an 85.00% power and revealed a consistent pattern in the data. Moreover, to ensure a normal distribution, we used a bootstrap test and ensured robust testing. The survey received 173 responses. We verified that all respondents were aware of software research by asking an initial question to check whether they were users and/or involved with software research. If the answer was negative, the survey concluded.

## 8 Conclusion and Future Works

The aim of this study was to understand factors that influence the adoption of research software and its challenges in the Brazilian academic environment. Overall, the findings from this study provide a representative snapshot of the perspectives of 173 academic research software users and developers, covering diverse aspects such as frequency of use and experience. This constitutes a number of responses that can represent the viewpoint of Brazilian researchers.

However, we acknowledge the limitations of our study. Firstly, our results emerge from a reasonable yet limited sample of Brazilian researchers, necessitating further replications in diverse contexts. It would be interesting to include in

the sample respondents from other countries to prevent the results from being influenced by a particular culture. With this, it is suggested that this study be conducted in other samples. Secondly, there are few studies specifically addressing the successes and challenges in research software, perhaps due to the complexity of measuring the human aspects involved in such studies. The fact that evaluating the importance of the factors identified in our previous study may influence the research since respondents who give more importance to other success factors may not use these factors. Hence, we do not claim our study to be exhaustive. It serves as a stepping stone towards understanding the multi-faceted factors that influence research software. Future work should aim for a more comprehensive description, which could culminate in a guideline, principles, checklist, or framework and a possible evaluation using the Technology Acceptance Model (TAM) (Cruz et al., 2022).

Another limitation of this study is that our statistical analysis is not intended to be predictive. As an exploratory technique, it offers a diagnostic view of the variables' behavior in our sample. These factors can be used as explanatory variables of a given phenomenon in confirmatory multivariate models, like multiple regression models, without multicollinearity.

Although this study contributes to a better understanding of the importance of factors in research software development and such adoption, we plan to conduct a Systematic Literature Mapping (SLM) (Petersen et al., 2015) to categorize and summarize studies related to guidelines, models, checklist, and principles in research software engineering. This will support academic researchers who develop and use research software. We will employ a hybrid search strategy, using *Database Search* in Scopus and parallel *Backward and Forward Snowballing* (Mourão et al., 2017, 2020). This hybrid search strategy tends to strike an appropriate balance between result quality and review effort. It will contribute to valuable insights into the factors that determine the adoption and non-adoption of research software. Researchers should revisit and refine their initial strategies to develop and adapt to the scientific demands in academia, ensuring the delivery of sustainable and widely usable software. Moreover, strategies for managing the expertise gap must address research software needs and constraints, seeking to understand the requirements of all stakeholders. The findings from the factors related to academic research software are valuable and useful for both software developers and users, and complements another investigation with developers only related to technical, organizational and people factors (Mourão et al., 2023b). In future work, we also intend to consider other communities abroad by replicating and comparing these experiments and factors to highlight the success and challenges of adopting research software from a multicultural perspective. Thus, this study contributes as a preliminary exploration of the ranking of factors that can guide the development of research software in academia.

## Acknowledgements

This study was financed in part by the Fundação de Amparo à

Pesquisa do Estado do Rio de Janeiro (FAPERJ) and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. The authors would like to thank the volunteer participants in the survey, FAPERJ, and CAPES.

## Data Availability

The dataset collected in the survey and that support the findings of this study are available in Zenodo repository with the identifier, <https://doi.org/10.5281/zenodo.7362052> The statistical analysis of dataset that support the findings of this study are available in GitHub, <https://github.com/ericamourao/researchsoftware>

## Conflict of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Authors' Contributions

Erica Mourão: Conceptualization, Data curation, Investigation, Software, Validation, Writing - original & draft, Writing - review & editing. Daniela Trevisan: Conceptualization, Methodology, Supervision, Writing - original draft, Writing - review & editing. José Viterbo: Conceptualization, Methodology, Supervision, Writing - original draft, Writing - review & editing. Ana Sobral: Methodology, Writing - original draft, Writing - review & editing. Monica da Silva: Writing - review. Carlo Eduardo Pantoja: Writing - review & editing.

## References

- AlNoamany, Y. and Borghi, J. A. (2018). Towards computational reproducibility: researcher perspectives on the use and sharing of software. *PeerJ Computer Science*, 4:e163.
- Barker, M., Chue Hong, N. P., Katz, D. S., Lamprecht, A.-L., Martinez-Ortiz, C., Psomopoulos, F., Harrow, J., Castro, L. J., Gruenpeter, M., Martinez, P. A., et al. (2022). Introducing the fair principles for research software. *Scientific Data*, 9(1):622.
- Basili, V. R. (1993). Applying the goal/question/metric paradigm in the experience factory. *Software quality assurance and measurement: A worldwide perspective*, 7(4):21–44.
- Basili, V. R., Caldiera, G., and Rombach, H. D. (1994). Goal, question metric paradigm. *encyclopedia of software engineering*, vol. 1.
- Basili, V. R. and Rombach, H. D. (1988). The tame project: Towards improvement-oriented software environments. *IEEE Transactions on software engineering*, 14(6).
- Baxter, R., Hong, N. C., Gorissen, D., Hetherington, J., and Todorov, I. (2012). The research software engineer. In *Digital Research Conference, Oxford*, pages 1–3.

- Brasil (2013). Resolução nº 466, de 12 de dezembro de 2012. *Diário Oficial da União*, 12:59–59. <https://conselho.saude.gov.br/resolucoes/2012/Reso466.pdf>.
- Brasil (2016). Resolução nº 510, de 7 abril de 2016. <https://conselho.saude.gov.br/resolucoes/2016/Reso510.pdf>.
- Brett, A., Croucher, M., Haines, R., Hettrick, S., Hetherington, J., Stillwell, M., and Wyatt, C. (2017). Research software engineers: state of the nation report 2017.
- Campanelli, A. S., Camilo, R. D., and Parreiras, F. S. (2018). The impact of tailoring criteria on agile practices adoption: A survey with novice agile practitioners in brazil. *Journal of Systems and Software*, 137:366–379.
- Carver, J., Heaton, D., Hochstein, L., and Bartlett, R. (2013). Self-perceptions about software engineering: A survey of scientists and engineers. *Computing in Science & Engineering*, 15(1):7–11.
- Carver, J. C., Hong, N. P. C., and Thiruvathukal, G. K. (2016). *Software engineering for science*. CRC Press.
- Carver, J. C., Weber, N., Ram, K., Gesing, S., and Katz, D. S. (2022). A survey of the state of the practice for research software in the united states. *PeerJ Computer Science*, 8:e963.
- Charmaz, K. (2014). *Constructing grounded theory*. sage.
- Coccia, M. (2019). Why do nations produce science advances and new technology? *Technology in society*, 59:101124.
- Cohen, J., Katz, D. S., Barker, M., Chue Hong, N., Haines, R., and Jay, C. (2021). The four pillars of research software engineering. *IEEE Software*, 38(1):97–105.
- Corbin, J. and Strauss, A. (2014). *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.
- Creswell, J. W. (2014). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, Thousand Oaks, CA.
- Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *psychometrika*, 16(3):297–334.
- Cruz, M., Mourão, É., Bernardini, F., Viterbo, J., and Trevisan, D. (2022). Uso do tam—technology acceptance model—no ciclo de design de aplicações computacionais. In *Anais Estendidos do XXI Simpósio Brasileiro de Fatores Humanos em Sistemas Computacionais*, pages 3–4. SBC.
- da Silva, M., Viterbo, J., Bernardini, F., and Maciel, C. (2018). Identifying privacy functional requirements for crowdsourcing applications in smart cities. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 106–111. IEEE.
- De Sá, J. P. M. (2007). *Applied statistics using SPSS, statistica, Matlab and R*. Springer Science & Business Media.
- de Souza, M. R., Haines, R., Vigo, M., and Jay, C. (2019). What makes research software sustainable? an interview study with research software engineers. In *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 135–138. IEEE.
- Denzin, N. K. (2012). Triangulation 2.0. *Journal of mixed methods research*, 6(2):80–88.
- Detofeno, T., Malucelli, A., and Reinehr, S. (2023). Technical debt guild: managing technical debt from code up to build. *Journal of Software Engineering Research and Development*, pages 1–1.
- Dwivedi, A. K., Mallawaarachchi, I., and Alvarado, L. A. (2017). Analysis of small sample size studies using non-parametric bootstrap test with pooled resampling method. *Statistics in medicine*, 36(14):2187–2205.
- Eisty, N. U. and Carver, J. C. (2022). Developers perception of peer code review in research software development. *Empirical Software Engineering*, 27(1):1–26.
- Eisty, N. U., Thiruvathukal, G. K., and Carver, J. C. (2018). A survey of software metric use in research software development. In *2018 IEEE 14th international conference on e-Science (e-Science)*, pages 212–222. IEEE.
- Eisty, N. U., Thiruvathukal, G. K., and Carver, J. C. (2019). Use of software process in research software development: A survey. In *Proceedings of the Evaluation and Assessment on Software Engineering*, pages 276–282.
- Fávero, L. P. and Belfiore, P. (2017). *Manual de análise de dados: estatística e modelagem multivariada com Excel®, SPSS® e Stata®*. Elsevier Brasil.
- Fink, A. (2003). *The survey handbook*. sage, Thousand Oaks, California.
- Flick, U. (2004). Triangulation in qualitative research. *A companion to qualitative research*, 3:178–183.
- Flick, U. (2020). *Triangulation*. Springer.
- Gruenpeter, M., Katz, D. S., Lamprecht, A.-L., Honeyman, T., Garijo, D., Struck, A., Niehues, A., Martinez, P. A., Castro, L. J., Rabemanantsoa, T., et al. (2021). Defining research software: a controversial discussion. *Zenodo* <https://doi.org/10.5281/zenodo.5504016>.
- Hannay, J. E., MacLeod, C., Singer, J., Langtangen, H. P., Pfahl, D., and Wilson, G. (2009). How do scientists develop and use scientific software? In *2009 ICSE workshop on software engineering for computational science and engineering*, pages 1–8. Ieee.
- Hasselbring, W., Carr, L., Hettrick, S., Packer, H., and Tiropanis, T. (2020). From fair research data toward fair and open research software. *it - Information Technology*, 62(1):39–47.
- Hauge, Ø., Ayala, C., and Conradi, R. (2010). Adoption of open source software in software-intensive organizations—a systematic literature review. *Information and Software Technology*, 52(11):1133–1154.
- Heaton, D. and Carver, J. C. (2015). Claims about the use of software engineering practices in science: A systematic literature review. *Information and Software Technology*, 67:207–219.
- Hettrick, S., Antonioletti, M., Carr, L., Chue Hong, N., Crouch, S., De Roure, D. C., Emsley, I., Goble, C., Hay, A., Inupakutika, D., et al. (2014). Uk research software survey 2014.
- IEEE (1990). *IEEE Std 610.12-1990: IEEE standard glossary of software engineering terminology*. IEEE.
- Jick, T. D. (1979). Mixing qualitative and quantitative methods: Triangulation in action. *Administrative science quar-*

- terly, 24(4):602–611.
- Johnson, R. A., Wichern, D. W., et al. (2002). *Applied multivariate statistical analysis*, volume 5. Prentice hall Upper Saddle River, NJ.
- Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3):187–200.
- Karita, L., Mourão, B. C., Martins, L. A., Soares, L. R., and Machado, I. (2021). Software industry awareness on sustainable software engineering: a brazilian perspective. *Journal of Software Engineering Research and Development*, 9:2–1.
- Katz, D. S., Gruenpeter, M., and Honeyman, T. (2021). Taking a fresh look at fair for research software. *Patterns*, 2(3):100222.
- Katz, D. S. and Hettrick, S. (2023). Research software engineering in 2030. In *2023 IEEE 19th International Conference on e-Science (e-Science)*, pages 1–2. IEEE.
- Katz, D. S., McHenry, K., Reinking, C., and Haines, R. (2019). Research software development & management in universities: case studies from manchester’s rds group, illinois’ ncsa, and notre dame’s crc. In *2019 IEEE/ACM 14th International Workshop on Software Engineering for Science (SE4Science)*, pages 17–24. IEEE.
- Kitchenham, B. A. and Pfleeger, S. L. (2008). Personal opinion surveys. In Shull, F., Singer, J., and Sjøberg, D. I. K., editors, *Guide to Advanced Empirical Software Engineering*, pages 63–92. Springer London, London.
- Lahorgue, M. A. (2006). Institutional autonomy and impact of research funding the case of brazilian public universities. *Knowledge, Power and Dissent*, page 255.
- Lusk, K. (2018). Lessons learned in effective community-university-industry collaboration models for smart and connected communities research.
- McIlroy, M. D., Buxton, J., Naur, P., and Randell, B. (1968). Mass-produced software components. In *Proceedings of the 1st international conference on software engineering, Garmisch Pattenkirchen, Germany*, pages 88–98.
- McMANUS, C. and Nobre, C. A. (2017). Brazilian scientific mobility program science without borders preliminary results and perspectives. *Anais da Academia Brasileira de Ciencias*, 89:773786.
- Mourão, E., Kalinowski, M., Murta, L., Mendes, E., and Wohlin, C. (2017). Investigating the use of a hybrid search strategy for systematic reviews. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 193–198. IEEE.
- Mourão, E., Pimentel, J. F., Murta, L., Kalinowski, M., Mendes, E., and Wohlin, C. (2020). On the performance of hybrid search strategies for systematic literature reviews in software engineering. *Information and Software Technology*, 123:106294.
- Mourão, E., Trevisan, D., and Viterbo, J. (2023a). Understanding the success factors of research software: Interviews with brazilian computer science academic researchers. In *ICITS’23 - 6th International Conference on Information Technology & Systems*. Springer.
- Mourão, E., Trevisan, D., Viterbo, J., and Pantoja, C. E. (2023b). Investigating developers’ perception on success factors for research software development. In *Accepted for publication in 22nd IFIP Conference e-Business, e-Services, and e-Society I3E2023 – IFIP WG. 6.11 (IFIP I3E 2023)*. Springer.
- Nangia, U. and Katz, D. S. (2017). Track 1 paper: Surveying the u.s. national postdoctoral association regarding software use and training in research.
- Nangia, U., Katz, D. S., et al. (2017). Track 1 paper: surveying the us national postdoctoral association regarding software use and training in research. In *Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE 5.1)*.
- Naur, P., Randell, B., and Buxton, J. N. (1976). *Software engineering: Concepts and techniques*.
- Neave, G. R. et al. (2006). *Knowledge, power and dissent: critical perspectives on higher education and research in knowledge society*. Unesco.
- Nguyen-Hoan, L., Flint, S., and Sankaranarayana, R. (2010). A survey of scientific software development. In *Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement*, pages 1–10.
- Noble, H. and Smith, J. (2015). Issues of validity and reliability in qualitative research. *Evidence-based nursing*, 18(2):34–35.
- Nunnally, J. C. (1994). *Psychometric theory 3E*. Tata McGraw-hill education.
- Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and software technology*, 64:1–18.
- Pinto, G., Wiese, I., and Dias, L. F. (2018). How do scientists develop scientific software? an external replication. In *2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER)*, pages 582–591. IEEE.
- Pituch, K. A. and Stevens, J. P. (2015). *Applied multivariate statistics for the social sciences: Analyses with SAS and IBM’s SPSS*. Routledge.
- Rahman, A. A. U. and Farhana, E. (2021). An empirical study of bugs in covid-19 software projects. *J. Softw. Eng. Res. Dev.*, 9:3–1.
- Santos, I., M. Melo, S., S. L. Souza, P., and R. S. Souza, S. (2022). A survey on the practices of software testing: a look into brazilian companies. *Journal of Software Engineering Research and Development*, 10:11:1 – 11:15.
- Segal, J. (2009). Some challenges facing software engineers developing software for scientists. In *2009 ICSE workshop on software engineering for computational science and engineering*, pages 9–14. IEEE.
- Sjøberg, D. I., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N.-K., and Rekdal, A. C. (2005). A survey of controlled experiments in software engineering. *IEEE transactions on software engineering*, 31(9):733–753.
- Smith, A. M., Katz, D. S., and Niemeyer, K. E. (2016). Software citation principles. *PeerJ Computer Science*, 2:e86.
- Sochat, V., May, N., Cosden, I., Martinez-Ortiz, C., and Bartholomew, S. (2022). The research software encyclopedia: a community framework to define research soft-

- ware. *Journal of Open Research Software*.
- Sousa, A. M. d. M. B. (2002). *Guia prático de utilização do SPSS: análise de dados para ciências sociais e psicologia*.
- Stol, K.-J., Ralph, P., and Fitzgerald, B. (2016). Grounded theory in software engineering research: a critical review and guidelines. In *Proceedings of the 38th International conference on software engineering*, pages 120–131.
- Sue, V. M. and Ritter, L. A. (2012). *Conducting online surveys*. Sage.
- Tilley, S., Huang, S., and Payne, T. (2003). On the challenges of adopting rots software. In *3rd International Workshop on Adoption-Centric Software Engineering*.
- Van Vliet, H., Van Vliet, H., and Van Vliet, J. (2008). *Software engineering: principles and practice*, volume 13. John Wiley & Sons Hoboken, NJ.
- Velicer, E. G. W. (1988). Relation of sample size to the stability of component patterns. *Psychological Bulletin*, 103(2):265.
- Wagner, S., Mendez, D., Felderer, M., Graziotin, D., and Kalinowski, M. (2020). Challenges in survey research. In *Contemporary Empirical Methods in Software Engineering*, pages 93–125. Springer.
- Wiese, I., Polato, I., and Pinto, G. (2019). Naming the pain in developing scientific software. *IEEE Software*, 37(4):75–82.
- Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., et al. (2016). The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9.
- Wohlin, C., Kalinowski, M., Felizardo, K. R., and Mendes, E. (2022). Successful combination of database search and snowballing for identification of primary studies in systematic literature studies. *Information and Software Technology*, 147:106908.
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.