THE UNIVERSITY OF SOUTH ALABAMA
SCHOOL OF COMPUTING

IMPROVING RESEARCH SOFTWARE ENGINEERING IN MATHEMATICS

BY

Abram S. Miller

An Honors Thesis

Submitted in partial fulfillment
of the requirements of the
The University of South Alabama
Honors Program and for the degree of

Bachelor of Science

in

Computer Science

May 2025

Approved: Date:

5/11/2025

Mentor: Dr. Tom Johnston

5/11/2025

Committee Member: Dr. Steven Clontz

5/11/25

Committee Member: Dr. Ryan Benton

12 May 25

Dean of the Honors College: Dr. Douglas Marshall

IMPROVING RESEARCH SOFTWARE ENGINEERING IN MATHEMATICS

An Honors Thesis

Submitted to the Graduate Faculty of
The University of South Alabama
in partial fulfillment of the
requirements for the degree of

Bachelor of Science

in

Computer Science

by
Abram S. Miller
May 2025

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to:

Dr. Steven Clontz, my mentor, for proposing the initial idea for the thesis, inviting me to the Joint Mathematics Meetings in Seattle, teaching me professional GitHub usage and etiquette, and keeping me on track throughout the thesis project.

David and Hasmin, my parents, for their emotional support, home-cooked food, and cat photos.

Mei and Mirabella, my sisters, for being my inspirations and keeping our family chats lively.

Kitkit and Yams, my cats, for being incredibly cute and photogenic little guys.

Ivan, my partner, for inspiring me with your own hard work and dedication.

Alex, my friend, for encouraging me to take breaks and making sure that I was eating, drinking, and getting enough sleep while writing this thesis.

Thank you to all of my friends and family for your support throughout this last semester of my undergrad degree.

TABLE OF CONTENTS

Page

LIST OF FIGURES v

ABSTRACT vii

CHAPTER I. INTRODUCTION 1

1.1. Terms used 2

1.2. Prior work 2

CHAPTER II. DESIGN AND METHODOLOGY 3

2.1. Survey design 3

2.2. Recruiting participants 4

CHAPTER III. ANALYSIS AND COMPARISON 6

3.1. Comparing survey results 6

3.1.1. Preprocessing and Survey Code 6

3.1.2. Survey introduction and demographics 14

3.1.3. Research question one - Software Development Practices . . 19

3.1.4. Research question two - Development Infrastructure and Tools 21

3.1.5. Research question three - Training 25

3.1.6. Research question four - Funding and Institutional Support 26

3.1.7. Research question five - Career Paths 33

3.1.8. Research question six - Credit for Software Contributions . 35

3.1.9. Research question seven - Diversity and Inclusion 38
3.2. Conclusions 40

CHAPTER IV. PI-BASE: A COMMUNITY DATABASE OF TOPOLOGICAL COUNTEREXAMPLES 41

CHAPTER V. DEVELOPMENT CONTRIBUTIONS TO π-BASE 43
5.1. Citation Button 44
5.2. External Markdown 47

CHAPTER VI. FUTURE WORK 51
6.1. Further feature improvements for π-Base 51
6.1.1. Graphical representation of π-Base data 52
6.2. Future research with survey data 52
6.3. Data Availability 53

REFERENCES 54

APPENDIX 56
Appendix A. First Appendix Title 56

BIOGRAPHICAL SKETCH 58

LIST OF FIGURES

Figure Page

1. Respondent type comparison. 14
2. Who respondents work for. 15
3. Professional title. 16
4. Age range. 17
5. Respondent experience 18
6. Team size. 18
7. Software target groups. 20
8. Use of testing. 20
9. Use of open-source license. 21
10. Project experience. 22
11. Availability of tool support. 23
12. Institutional support provided. 27
13. Institutional support needed. 28
14. The importance of various hiring concerns. 34
15. The importance of various hiring concerns about the respondent's current position. 35
16. Software credit methods. 36
17. Methods of receiving credit. 37

18. Recruiting, retaining, and including participants from underrepresented groups. 39

19. Citation button code changes 45

20. Additional citation button code changes 46

21. The citation button on the π-Base homepage 46

22. The citation button on the explore page, properly updated to link to search results 46

23. The 'Show Markdown' button showing standard formatted Markdown beneath the π-Base Markdown 47

24. External Markdown code snippet 49

25. External markdown changes to multiple files 50

26. IRB Approval form 57

ABSTRACT

Miller, Abram, B.S., University of South Alabama, May 2025. Improving Research Software Engineering in Mathematics. Mentor: Tom Johnsten, Ph.D.

Research Software Engineering is critical to modern mathematical research, enabling the creation, maintenance, and dissemination of computational tools that bridge theory and practice. However, the field faces systemic challenges, including insufficient funding, lack of institutional recognition, and gaps in training and infrastructure. This thesis investigates these challenges through two approaches: (1) a comparative survey study focused on mathematicians and (2) hands-on contributions to an open-source research software project.

The Improving Research Software Engineering in Mathematics survey, conducted from September 2024 to January 2025, adapts the survey framework developed by Carver et al. in *A survey of the state of the practice for research software in the United States* to analyze research software engineering practices among mathematicians.

The thesis details contributions to π-Base, a community-driven database of topological counterexamples. Software contributions include a dynamic citation button and an External Markdown feature, both of which were developed through collaborative workflows adhering to continuous integration and continuous deployment (CI/CD) practices. These contributions emphasize the importance of community development efforts in sustaining research software.

CHAPTER I
INTRODUCTION

Research Software Engineering, or RSE, is a discipline that is dedicated to the design, development, and maintenance of software with the goal of advancing scientific inquiry. In the field of mathematics, RSE acts as a bridge between abstract theory and concrete practice. It enables research software mathematicians to find and solve computationally complex problems, validate hypotheses, and share reproducible results with peers. RSE combines software engineering principles with knowledge specific to any given field. In mathematics, this translates into modern research tools and software. Mathematical RSE can be found in almost every modern tool for symbolic computation, number crunching, formalized proofs, databases, and more.

Beyond general software development, RSE extends to the maintenance of existing infrastructure. After creating functional research tools, software engineering practices such as version control and thorough documentation ensure the reproducibility and scalability of existing codebases and architecture. However, despite its critical role, RSE in mathematics faces systemic challenges: projects often collapse due to unstable funding, inadequate institutional recognition, and a lack of training in sustainable software practices.

This honors thesis is divided into two parts. Sections 3 to 5 detail the methodology, design, and results of the Improving Research Software Engineering survey conducted between September 2024 and January 2025 compared to the

survey conducted by Carver et al. Sections 6 to 7 of the study document technical contributions, experiences, and the process of contributing to the pi-Base community database.

1.1 Terms used

- **IRSEMath**: Improving Research Software Engineering in Mathematics. This term is the acronym for the author's survey.
- **URSSI**: US Research Software Sustainability Institute. This term is the title for the survey conducted by Dr. Carver et al.
- **RSE**: Research Software Engineers, or Research Software Engineering. Refers to individuals or the field, depending on the context.
- **RQ1, RQ2, etc**: Research Question 1, 2, etc. This term is used as shorthand for the research questions referenced in the survey analysis.

1.2 Prior work

Dr. Jeffrey C. Carver et al. studied the general field of research software engineering with their survey, *A survey of the state of the practice for research software in the United States*. In this study, Carver covered how research software engineers face numerous challenges, primarily in software sustainability and development. The key findings of the study include the need for more training, a consistent lack of funding, and the need for field recognition in software development for career advancement. The goal of Carver's study was to inform and further support the development of research software infrastructure, and to encourage national policy to increase its focus on software longevity [1].

CHAPTER II

DESIGN AND METHODOLOGY

2.1 Survey design

This survey, Improving Research Software Engineering in Mathematics, or IRSEMath, is a direct recreation of the survey conducted by Dr. Jeffery C. Carver et al. The parent survey is referred to in this study as URSSI, after the US Research Software Sustainability Institute, since the survey is titled so in the paper and on *Zenodo* [2].

Unlike the URSSI, the IRSEMath survey is not aimed at research software engineers in all fields. Instead, it is different from its predecessor in that its intended audience is research software mathematicians, a much smaller demographic. Additionally, while the IRSEMath closely mirrors the URSSI survey, it does not pose the additional follow-up questions from that survey.

The IRSEMath survey is divided into eight sections. The first section gathers the demographics of the participants. The following seven sections ask each of the research questions originally posed in the URSSI survey.

1. Software Development Practices: The activities research software developers spend time on and the impacts of those activities.

2. Development Infrastructure and Tools: The tools research software developers use and what is needed to support sustainable development practices,

3. Training: The training that is available to research software developers if it meets their needs.

4. Funding and Institutional Support: The available institutional support for research software development and the sources of institutional funding available to research software developers.

5. Career Paths: The factors that impact career advancement and hiring in research software.

6. Credit for Software Contributions: The requirements for crediting or attributing software use in research software projects and how individuals and groups are given institutional credit for developing research software.

7. Diversity and Inclusion: How current Research Software Projects document diversity statements and the support needed to further diversity initiatives.

To analyze the free-response data questions (primarily RQ4.5) from the IRSEMath and URSSI surveys, the author used the Deepseek large language model AI to cluster and rank the responses into the most commonly expressed sentiments [3]. The tool found several categories (e.g., "funding and resource scarcity," "lack of recognition," "training and expertise gaps") and highlighted several representative quotes from each. To ensure that the AI did not "hallucinate" survey data, quotes were cross-referenced with the data to ensure that they were valid. This approach streamlined the analysis of more than 200 text responses, which helped to analyze important survey data.

2.2 Recruiting participants

After receiving Institutional Review Board (IRB) approval for survey ethics, the Improving Research Software Engineering in Mathematics (IRSEMath) survey

was disseminated through a number of avenues to recruit participants. The survey recruitment was carried out through institutional and community channels. Digitally, messages were distributed to various established Google groups and Zulip servers for users and developers of mathematics research software, and participants were encouraged to share the survey with colleagues. The survey was advertised with a standardized message outlining the purpose of the study: an introduction to the survey describing the student research and a request for mathematicians to take the survey. In addition to distributing the survey digitally, the author volunteered to help host a booth in Seattle, WA at the January 2025 Joint Mathematics Meetings (JMM) conference. JMM is the world's largest mathematical gathering with more than 5400 registrations in 2025 [4]. The author volunteered at the booth for the Code4math organization, of which Dr. Steven Clontz is a member [5]. The primary goal of the volunteer work was to ensure that at least one person was available at the booth while the members of the organization hosted various workshops and attended events. While "holding down" the Code4math booth, the author engaged with convention attendees to expand the reach of the organization, and also requested survey participation from convention attendees who matched survey criteria.

CHAPTER III

ANALYSIS AND COMPARISON

3.1 Comparing survey results

In the URSSI survey by Dr. Carver et al., the raw survey data is archived on *Zenodo*. Although the URSSI survey data is open source, none of the methods used to graph or represent the data in the resulting paper is shared. Because of this, a large part of the IRSEMath thesis work was dedicated to using the *Pandas* and *Matplotlib* libraries in *Python* to create graphs and data representations for comparing data from the two surveys. However, for the purpose of this thesis, a subset of the full data was analyzed and graphed. To graph the survey questions, the author started by coding comparison graphs for specific questions, referencing the graphs provided in the previous study. After creating a set of graphs, the author repeatedly iterated on those graphs to improve the text and data representations, then rewrote those graphs into modular and reusable functions.

3.1.1 Preprocessing and Survey Code

Before the IRSEMath and URSSI survey datasets were workable, the author needed to preprocess the data, which is stored in comma separated value (CSV) sheets. This processing involved removing incomplete survey responses, aligning the question names between the files, and manually repairing the formats of several

response columns within the survey, and removing the follow-up questions in the URSSI data that were not included in IRSEMath data collection.

The code to extract and present the survey in its entirety is hosted and annotated in a readable format in the author's Improving Research Software Engineering in Mathematics repository using GitHub pages [6]. This section briefly describes and displays the functions used to print and graph the survey data.

Once the data have been extracted from the surveys in a workable dataframe format, there are two main types of functions that are used for data representation: printing and graphing. The print functions utilize the *Pandas* data framework to aggregate and display the data of the question in a text format with response counts, percentages, and totals.

```
# text function: print the answer counts, percentage, and totals for a given question
def print_question(questionID="Q0.1", survey=0, percentages=True, totals=True):
    if survey == 0 or survey == 1:
        questions = IRSEM[questionID][2:]
        if percentages:
            df = pd.concat(
                [
                    questions.value_counts(),
                    questions.value_counts(normalize=True).round(2),
                ],
                axis=1,
                keys=["IRSEM survey: counts", "percentages"],
            )
        else:
            df = pd.concat(
                [questions.value_counts()], axis=1, keys=["IRSEM survey: counts"]
            )

        if totals:
            df.loc["TOTAL:"] = df.sum(numeric_only=True, axis=0)

        print(df)

```

```python
    if survey == 0:
        print("\n - - - - -\n")

    if survey == 0 or survey == 2:
        questions = URSSI[questionID][2:]
        if percentages:
            df = pd.concat(
                [
                    questions.value_counts(),
                    questions.value_counts(normalize=True).round(2),
                ],
                axis=1,
                keys=["URSSI survey: counts", "percentages"],
            )
        else:
            df = pd.concat(
                [questions.value_counts()], axis=1, keys=["URSSI survey: counts"]
            )

        if totals:
            df.loc["TOTAL:"] = df.sum(numeric_only=True, axis=0)

        print(df)
```

Listing III.1. The functions used to print a question comparison from the survey data

```python
# text function: print a set of questions (aka a Likert Matrix)
def print_question_set(question="", statements=[], survey=0, percentages=True):
    div = "___________________"
    for i, statement in enumerate(statements):
        print(statement + ":\n")
        print_question(
            question + "_" + str(i + 1), survey=survey, percentages=percentages
        )
        if i < len(statements) - 1:
            print("\n", div * 3, "\n")
```

Listing III.2. The functions used to print a set of questions for direct comparison

The `print_question()` function works to pull the specific questions from the survey, aggregate and read them into smaller dataframes, then print them with added value counts and percentages, based on a number of parameters for each function. The `print_question_set()` function works as a "bulk printing" function to print all of the subquestions of a matrix question along with each of their associated question statements in a list format.

```
# graph question in a standard format
def graph_question(
    questionID="Q0.1",
    title="Respondent type (Demographics: question 1)",
    IRSEMsubtitle="IRSEM: Mathematicians",
    URSSIsubtitle="URSSI: Research Software Engineers",
    statement="Respondent type",
    answersIRSEM=[],
    answersURSSI=[],
    percentage=True,
    stacked=False,
    labelrotation=46,
):

    # get question data for each survey
    QUESTION_DATA_IRSEM = [(statement, IRSEM[questionID][2:])]
    QUESTION_DATA_URSSI = [(statement, URSSI[questionID][2:])]

    # get lists of possible answers for each version of the question
    # the pandas conversion is necessary to remove the 'nan' values in the list
    if not answersIRSEM:
        ANSWER_DATA_IRSEM = (
            pd.Series(IRSEM[questionID][2:].sort_values().unique()).dropna().tolist()
        )
    else:
        ANSWER_DATA_IRSEM = answersIRSEM
    if not answersURSSI:
        ANSWER_DATA_URSSI = (
            pd.Series(URSSI[questionID][2:].sort_values().unique()).dropna().tolist()
        )
    else:
```

```
        ANSWER_DATA_URSSI = answersURSSI

    # create dataframes
    if percentage:
        df1 = pd.DataFrame(
            get_percentages(QUESTION_DATA_IRSEM, ANSWER_DATA_IRSEM), index=[statement
    ]
        )
        df2 = pd.DataFrame(
            get_percentages(QUESTION_DATA_URSSI, ANSWER_DATA_URSSI), index=[statement
    ]
        )
    else:
        df1 = pd.DataFrame(
            get_ans_counts(QUESTION_DATA_IRSEM, ANSWER_DATA_IRSEM), index=[statement]
        )
        df2 = pd.DataFrame(
            get_ans_counts(QUESTION_DATA_URSSI, ANSWER_DATA_URSSI), index=[statement]
        )

    # plot
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.suptitle(title)

    ax1 = df1.plot.bar(rot=0, ax=ax1, stacked=stacked)
    ax1.set_title(IRSEMsubtitle)
    ax1.set_ylabel("Percentage %") if percentage else ax1.set_ylabel("Frequency")
    ax1.tick_params("x", labelrotation=labelrotation)
    if percentage:
        ax1.set_yticks(np.arange(0, 100.1, 5), minor=True)
    # ax1.get_legend().remove()

    ax2 = df2.plot.bar(rot=0, ax=ax2, stacked=stacked)
    ax2.set_title(URSSIsubtitle)
    ax2.set_ylabel("")
    ax2.tick_params("x", labelrotation=labelrotation)
    if percentage:
        ax2.set_yticks(np.arange(0, 100.1, 5), minor=True)

    # edit parameters
```

```
    plt.legend(bbox_to_anchor=(1.02, 1))
    plt.gcf().set_size_inches(12, 6)
    plt.show()
```

Listing III.3. The functions used to graph a question comparison from the survey data

```
# function: find counts of each answer type on each question

def get_ans_counts(questions, answerTypes):
    ANSWERCOUNTS = {category: [] for category in answerTypes}
    for label, data in questions:
        for category in answerTypes:
            count = data.value_counts().get(category, 0)
            ANSWERCOUNTS[category].append(count)
    # print(ANSWERCOUNTS)
    return ANSWERCOUNTS


# function: get the percentages from the total counts of each category's responses
def get_percentages(questions, answerTypes):
    PERCENTAGES = {category: [] for category in answerTypes}
    for label, data in questions:
        for category in answerTypes:
            percent = data.value_counts(normalize=True).get(category, 0) * 100
            PERCENTAGES[category].append(percent)
    # print(totals)
    return PERCENTAGES


# functions to graph a likert matrix

def get_likert_data(
    survey=IRSEM,
    questionID="Q0.8",
    statements=["1 Person", "2 - 5 people", "6 - 20 people", "20+ people"],
):
    DATA = []
    for i, statement in enumerate(statements):
        DATA.append((statement, survey[questionID + "_" + str(i + 1)][2:]))
```

```
        # print(statement)

    return DATA

    # returns this data format:
    #   [('1 Person',             survey["Q0.8_1"][2:]),
    #    ('2 - 5 people',         survey["Q0.8_2"][2:]),
    #    ('6 - 20 people',        survey["Q0.8_3"][2:]),
    #    ('20+ people',           survey["Q0.8_4"][2:])]


def graph_likert_matrix(
    questionID="Q0.8",
    title="Team Size (Question 0.8)",
    IRSEMsubtitle="IRSEM: Mathematicians",
    URSSIsubtitle="URSSI: Research Software Engineers",
    statements=["1 Person", "2 - 5 people", "6 - 20 people", "20+ people"],
    answers=["Never", "Sometimes", "About half the time", "Most of the time", "Always
    "],
    percentage=True,
    stacked=True,
    labelrotation=45,
):

    LIKERT_DATA_IRSEM = get_likert_data(
        survey=IRSEM, questionID=questionID, statements=statements
    )
    LIKERT_DATA_URSSI = get_likert_data(
        survey=URSSI, questionID=questionID, statements=statements
    )

    # create dataframes
    if percentage:
        df1 = pd.DataFrame(
            get_percentages(LIKERT_DATA_IRSEM, answers),
            index=[label for label in statements],
        )
        df2 = pd.DataFrame(
            get_percentages(LIKERT_DATA_URSSI, answers),
            index=[label for label in statements],
```

```
        )
    else:
        df1 = pd.DataFrame(
            get_ans_counts(LIKERT_DATA_IRSEM, answers),
            index=[label for label in statements],
        )
        df2 = pd.DataFrame(
            get_ans_counts(LIKERT_DATA_URSSI, answers),
            index=[label for label in statements],
        )

    # plot
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.suptitle(title)  # Title

    ax1 = df1.plot.bar(rot=0, ax=ax1, stacked=stacked)
    ax1.set_title(IRSEMsubtitle)
    ax1.set_ylabel("Percentage %") if percentage else ax1.set_ylabel("Frequency")
    ax1.tick_params("x", labelrotation=labelrotation)
    if percentage:
        ax1.set_yticks(np.arange(0, 100.1, 5), minor=True)
    ax1.get_legend().remove()

    ax2 = df2.plot.bar(rot=0, ax=ax2, stacked=stacked)
    ax2.set_title(URSSIsubtitle)
    ax2.tick_params("x", labelrotation=labelrotation)
    if percentage:
        ax2.set_yticks(np.arange(0, 100.1, 5), minor=True)

    # edit parameters
    plt.legend(bbox_to_anchor=(1.02, 1))
    plt.gcf().set_size_inches(12, 6)
    plt.show()
```

Listing III.4. The functions used to graph a Likert Matrix from the survey data

The `get_ans_counts()` and `get_percentages()` functions iterate through the provided question data to prepare the formatted data for graphical representation. The `get_likert_data()` is used with bulk graphing of related

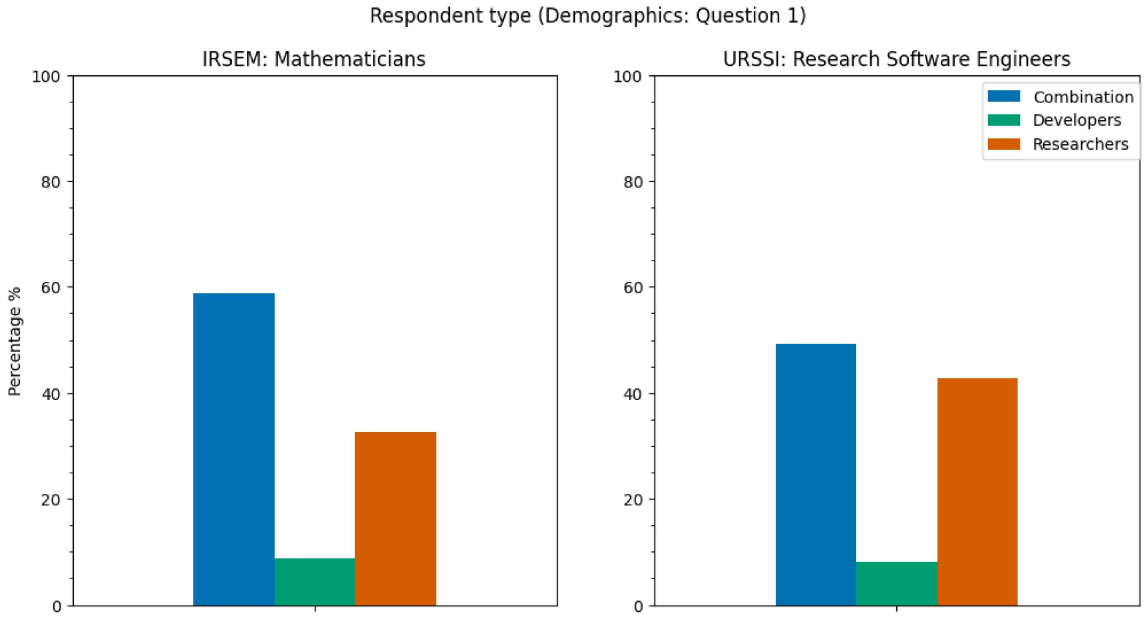

Figure 1. Respondent type comparison.

questions and aggregates results from the `get_ans_counts()` and `get_percentages()` functions. These functions are used as part of the `graph_likert_matrix()` function.

3.1.2 Survey introduction and demographics

The study opens with the consent form detailing the purpose of the survey, participation benefits and risks, compensation, confidentiality, and contact information. After the participant consents to taking the survey, their demographics are collected in the following section, Respondent Demographics. In this section, participants are asked what profile matches the work they perform, organization, professional title, age, time in research, gender, and average research team size.

Question 1 (Figure 1) asks what profile matches the work the participant performs. There are three options to this question: researchers, developers, or combination. Researchers are purely users who primarily use software to do their research. Developers mainly write software or contribute to software projects for

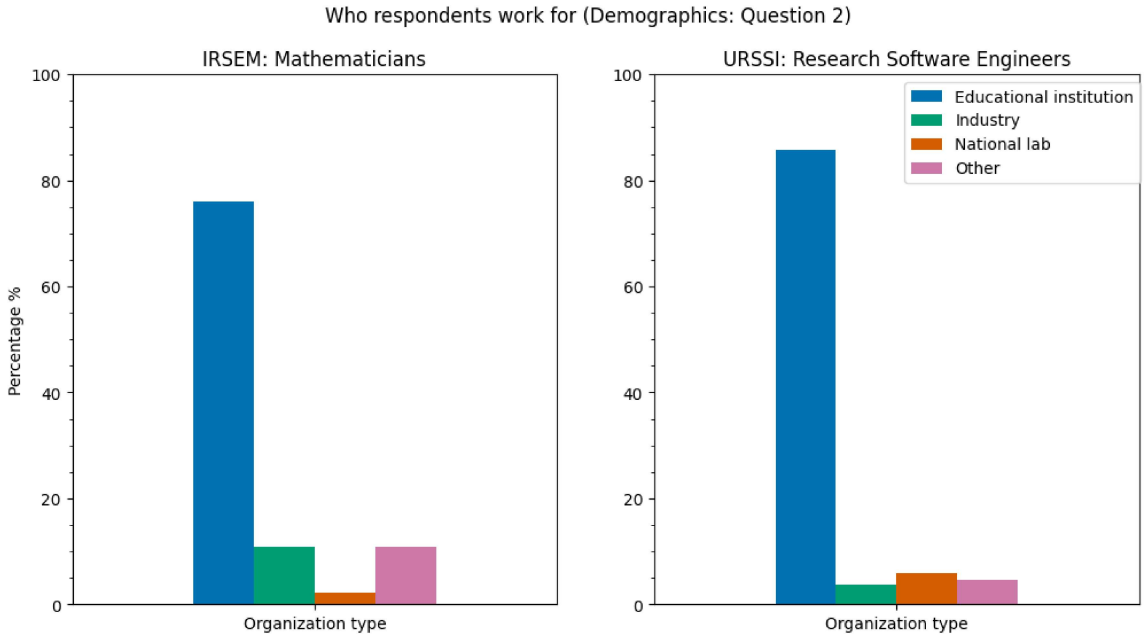

Figure 2. Who respondents work for.

others to use. Combination researcher-developers write software for both personal research and for others.

Both surveys have a similar overall distribution of respondent types. However, the ratio of combination to developer is higher at 59% (27/46) to 33% (13/46) in the IRSEMath mathematician survey , compared to 49% (544/1106) to 43% (473/1106) in the URSSI survey with researchers in general. Developers make a similar minority in both surveys at 9% (4/46)

Question 2 (Figure 2) asks what type of organization the participant works for. In both surveys, the overwhelming majority of the respondents work for educational institutions, with mathematicians at 76% (35/46) and general researchers at 86% (901/1051). For mathematicians, 'industry' and 'other' are tied at 11% (5/46), while only 4% (39/1051) and 5% (49/1051) for researchers, respectively. Only one IRSEMath participant worked in a national lab, while 6% (62/1051) did on the URSSI.

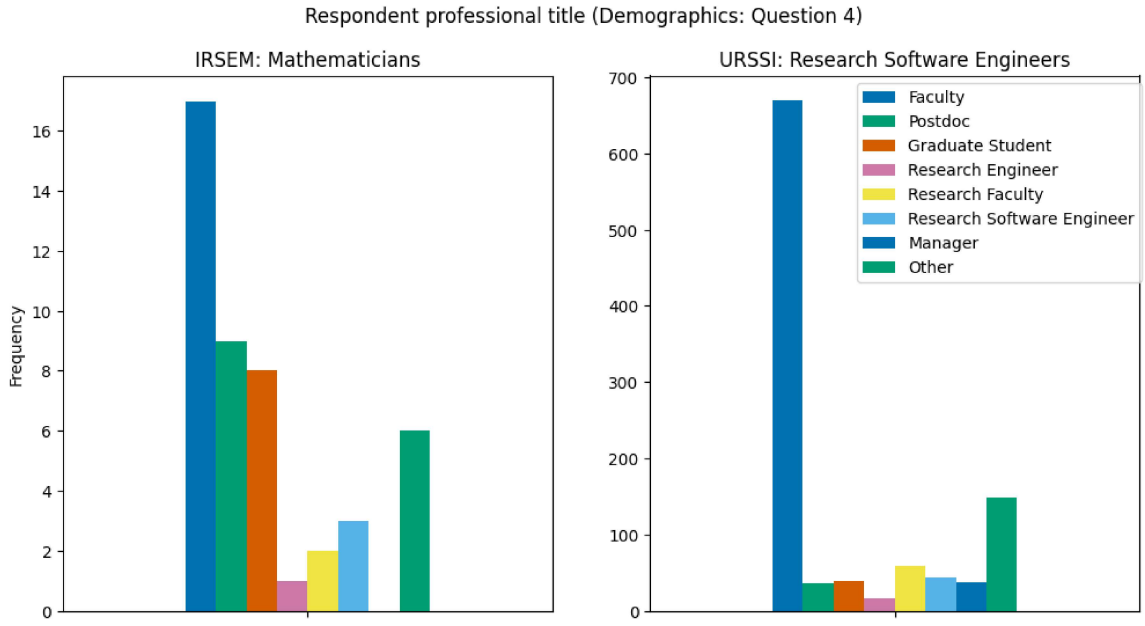

Figure 3. Professional title.

Question 3 asks the participant what country they live in. While the URSSI survey tallies to 95% (993/1041) American respondents, the IRSEMath tallies to only 61% (28/46) American. This is likely because the URSSI survey was targeted toward US-based lists, while the IRSEMath was shared to multiple decentralized communities. Notably, Germany is the second highest country (4/46 and 11/1041) on both lists.

Question 4 asks what the participant's job title is. Faculty is the primary job title in both surveys (IRSEMath: 37% at 17/46, URSSI: 64% at 669/1049). On the IRSEMath, this is followed by Postdoc at 20% (9/46) and Graduate student at 17% (8/46). However, in the URSSI, no other title comprised a significant portion of respondents.

Question 5 (Figure 4) asks the participant's age. For both surveys, the largest group was between 35 and 44 years of age (IRSEMath: 46% at 21/46, URSSI: 35% at 366/1038). However, the IRSEMath trended younger, with 25-34

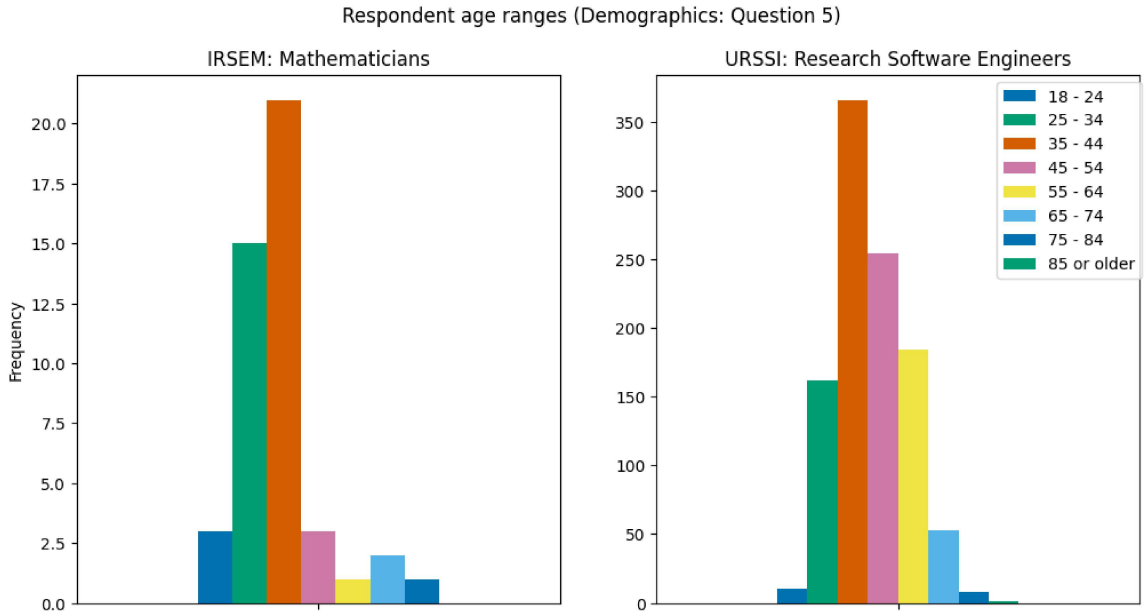

Figure 4. Age range.

being the second largest group (33% at 15/46), and the URSSI trended older, with 45-54 (24% at 254/1038) being its second most populous age group.

Question 6 (Figure 5) asks how many years of experience the participant has in research. Survey respondents in the IRSEMath survey skewed younger, with 68% (31/46) having between 1 and 15 years of experience. URSSI respondents skewed older, with 76% (800/1043) of respondents having 11 or more years of experience.

Question 7 asks the participant's gender. Both surveys were skewed majority male. On the IRSEMath, 85% (39/46) of respondents identified as male, compared to 70% (734/1042) on the URSSI. In contrast, 13% (6/46) of IRSEMath respondents and 26% (268/1042) of URSSI respondents identified as female.

Question 8 (Figure 6) asks how often the participant works on projects of varying team sizes. The average team sizes in both surveys followed a similar trend: Participants often worked alone or in groups of 2 to 5 people. A smaller percentage, even smaller on the IRSEMath, regularly had team sizes between 6 and 20 people,

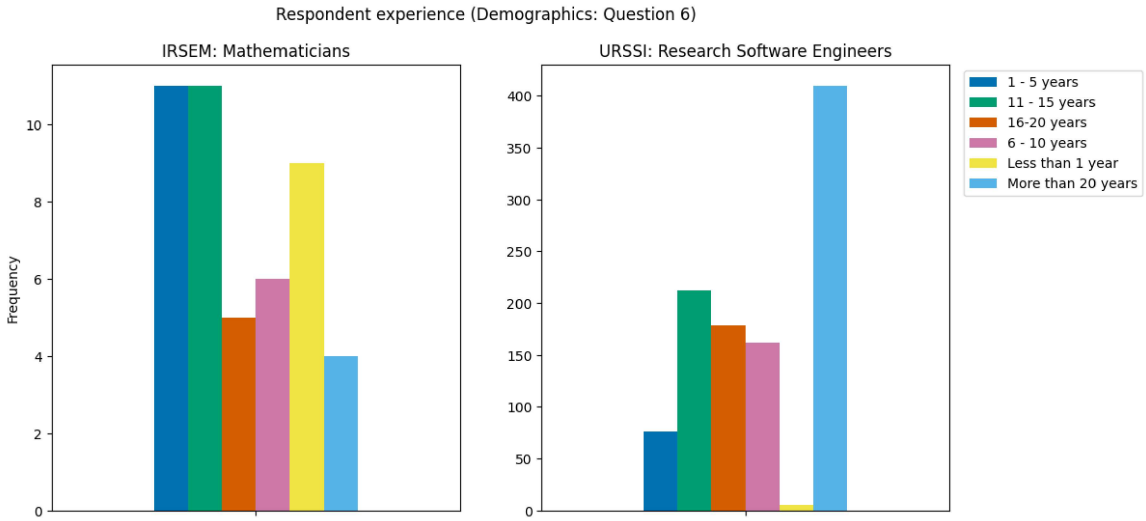

Figure 5. Respondent experience

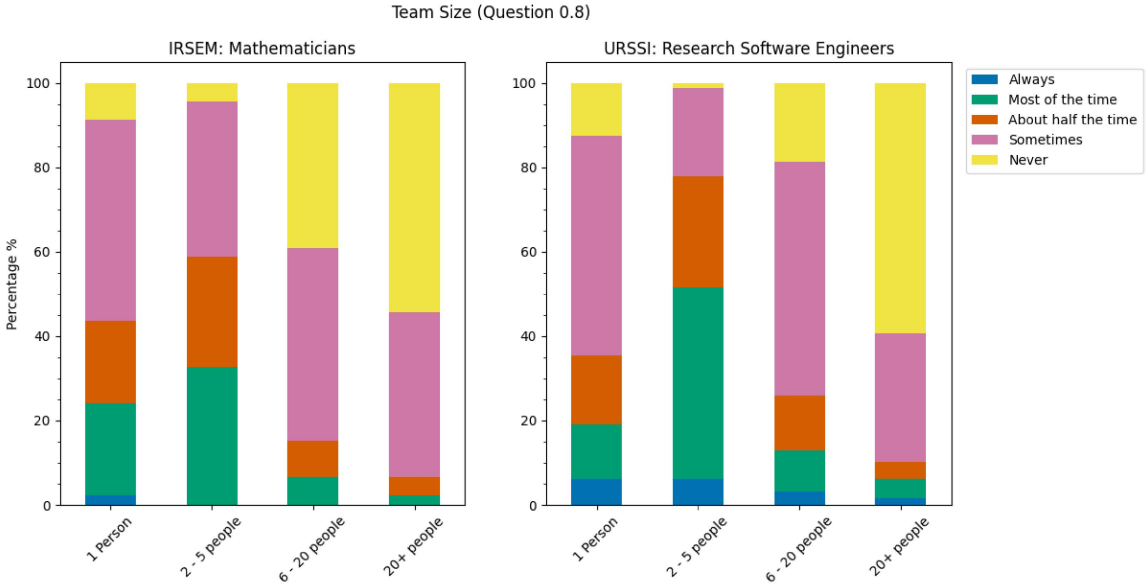

Figure 6. Team size.

or more than 20 people. Notably, on the IRSEMath, no respondent chose "always" on any question, except 2% (1/46) of the respondents who reported that they always work alone.

3.1.3 Research question one - Software Development Practices

Research question one asks what activities research software developers spend their time on and how this impacts the perceived quality and long-term accessibility of research software. This section is only shown to respondents who are either developers or combination researcher-developers.

Question 1 (Figure 7) asks to what extent the software was targeted at different groups of users. Based on IRSEMath results, mathematicians' priorities for who their software is intended for were their discipline (11/22—50%), followed by themselves (9/22—41%), multiple disciplines (8/22—36%), then their teams (1/22—5%). Teams were consistently the secondary target for software (11/22—50%). URSSI respondents followed a somewhat different trend, with the primary focus being their discipline (226/450—50%), followed by their team (205/425—48%), themselves (133/411—32%), and other disciplines (106/431—25%).

Questions 2 and 3 are omitted.

Question 4 (Figure 8) asks what types of software testing the participant uses. According to IRSEMath results, mathematicians used testing more frequently on average than the URSSI respondents. Integration testing was used most frequently, with 45% (10/22) of respondents answering 'Frequently' versus 14% (3/22) answering 'Never'. Notably, regression testing was the least used in the IRSEMath survey, with 23% (5/22) answering 'Frequently' and 41% (9/22) of respondents answering 'Never'.

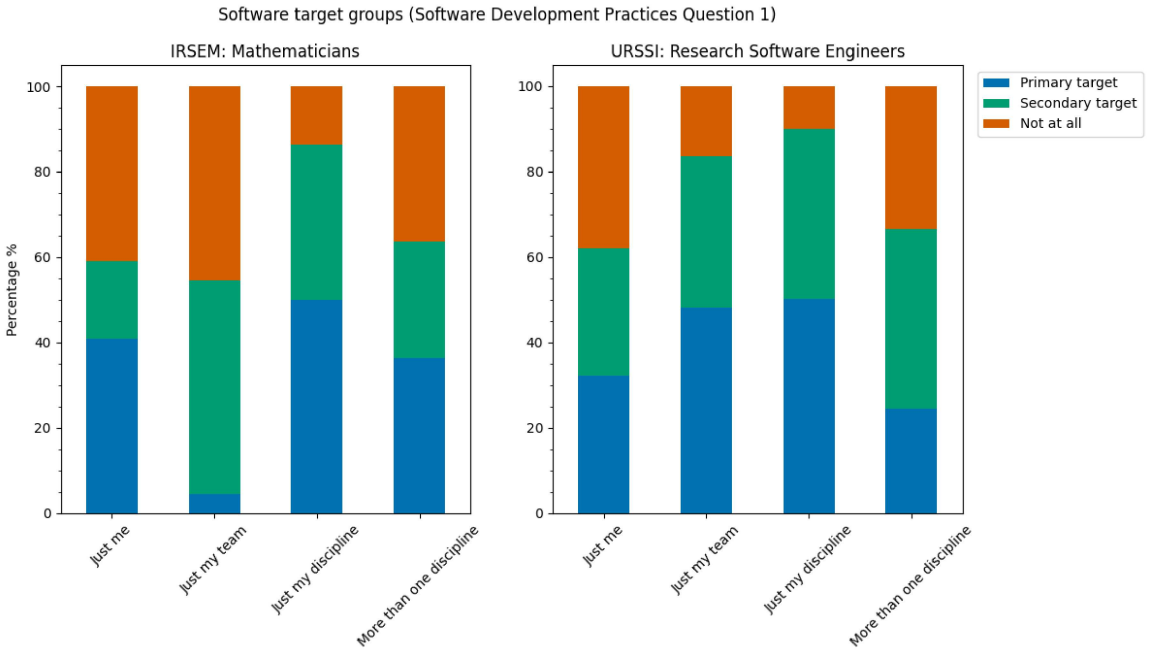

Figure 7. Software target groups.

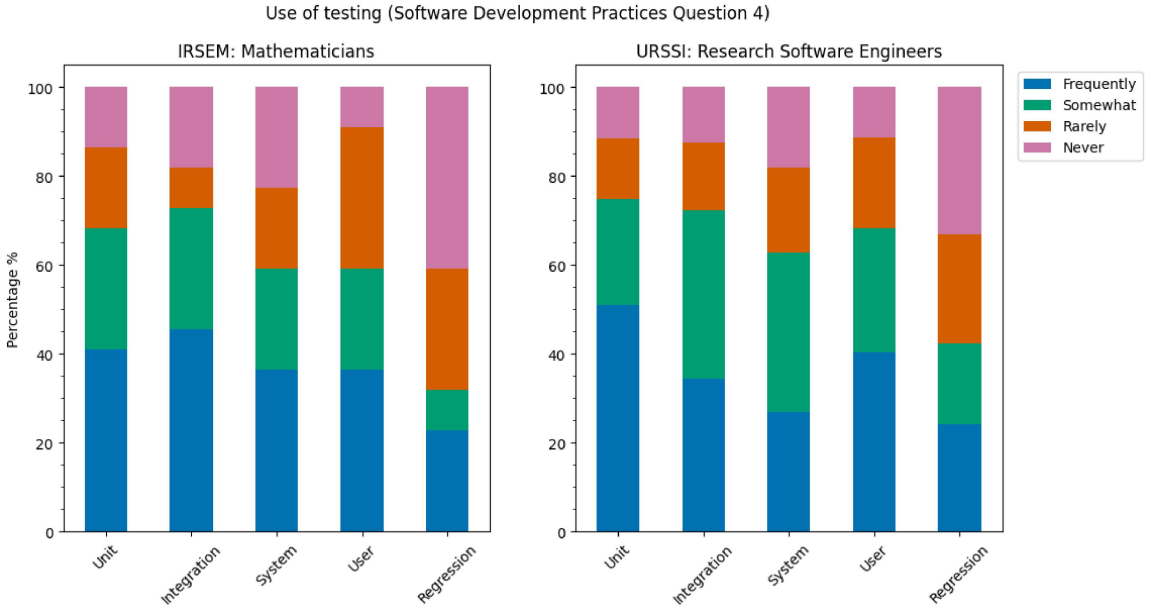

Figure 8. Use of testing.

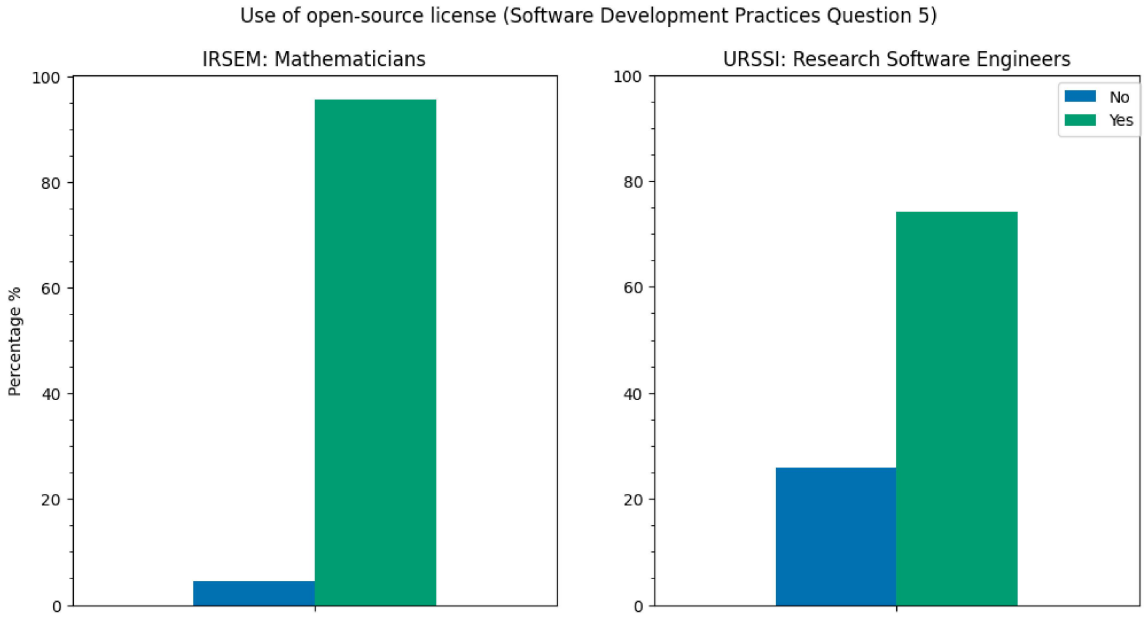

Figure 9. Use of open-source license.

Question 5 (Figure 9) asks if the participant has released their software under an open-source license. Mathematicians overwhelmingly released software with an open-source license, with 95% (21/22) of respondents answering 'Yes', versus only 74% (349/471) doing so on the URSSI.

3.1.4 Research question two - Development Infrastructure and Tools

This research question aims to find what tools research software developers use and what additional tools are needed to support sustainable development practices. The question asks about what software development infrastructure, tools, and practices respondents may use or are familiar with. This section is also only shown to developers or researcher-developers in combination.

Question 1 (Figure 10) asks how much experience the respondent has had with a selection of project types within the past five years. The project types provided to respondents were the following:

- Single developer

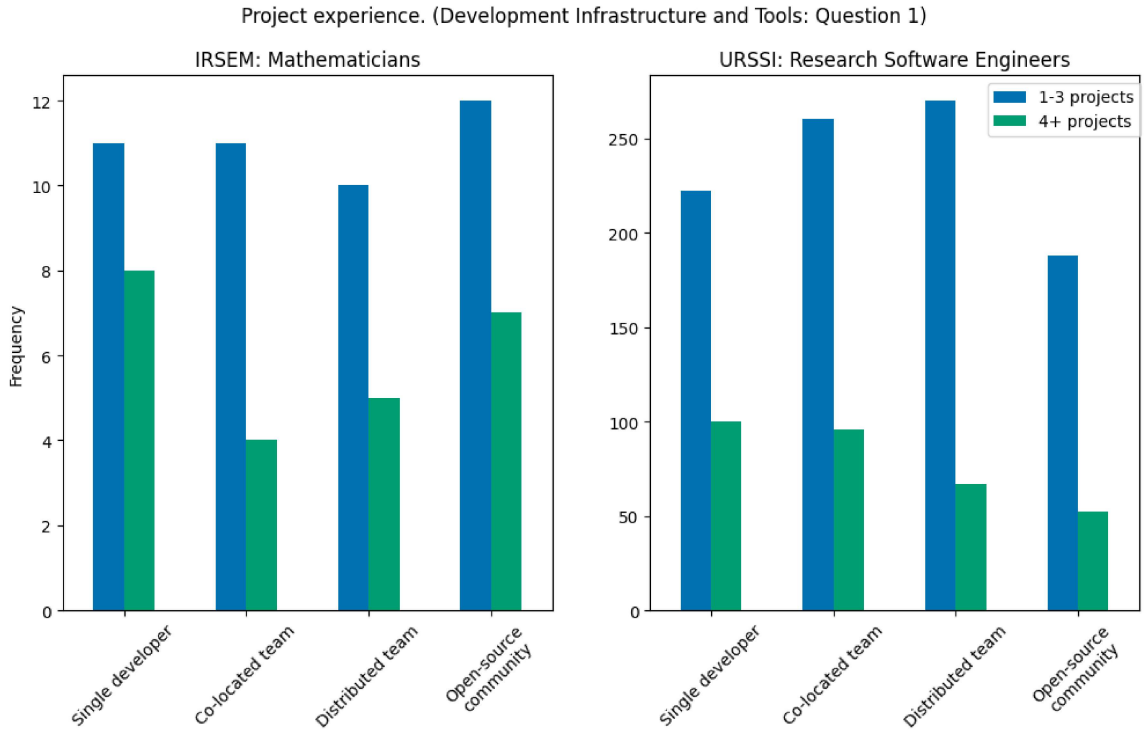

Figure 10. Project experience.

- Co-located team (i.e. a team, located in a single place or institution with well-coordinated funding/organization)
- Distributed team (i.e. a well-defined team, located in multiple places or institutions with well-coordinated funding/organization)
- Open-source community (i.e. a group of developers who are distributed and working under multiple independent funding sources)

We rank the options by the respondents' project experience:

1. IRSEMath - 4 or more projects
2. Single Developer (8/19—42%)
3. Open-source community (7/19—37%)

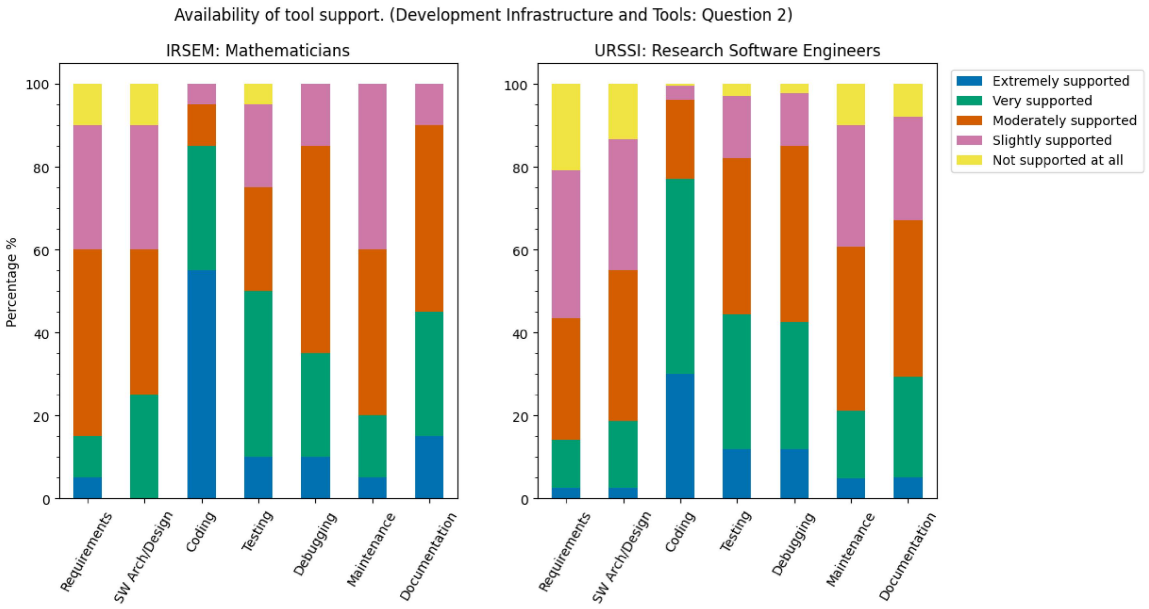

Figure 11. Availability of tool support.

4. Distributed team (5/15—33%)

5. Co-located team (4/15—27%)

1. URSSI - 4 or more projects

2. Single developer (100/322—31%)

3. Co-located team (96/356—27%)

4. Open-source community (52/240—22%)

5. Distributed team (67/337—20%)

IRSEMath respondents were more likely to have experience in four or more projects in every project type, apart from co-located teams, where the rates on both surveys matched exactly.

Question 2 (Figure 11) asks how well current tools support various research software support activities. We combine the values of 'Extremely supported' and 'Very supported' to rank the availabilities of tool support for each survey:

IRSEMath

1. Coding (17/20—85%)
2. Testing (10/20—50%)
3. Documentation (9/20—45%)
4. Debugging (7/20—35%)
5. Software architecture and design (5/20—25%)
6. Maintenance (4/20—20%)
7. Requirements (3/20—15%)

URSSI

1. Coding (341/442—77%)
2. Testing (196/442—44%)
3. Debugging (188/442—43%)
4. Documentation (130/442—29%)
5. Maintenance (93/435—21%)
6. Software architecture and design (82/439—19%)
7. Requirements (62/435—15%)

With similar results to the URSSI, IRSEMath mathematician respondents only indicated that coding practices were well supported. All other areas show a clear need for support and would benefit from such.

3.1.5 Research question three - Training

Research question three asks what training is available to research software developers and whether this training meets their needs, with the aim of understanding where developers may have received formal or informal training for software development.

Question 1 asks if the respondent has received training for software development. If they answered yes, they would be presented with a follow-up question asking where. On the IRSEMath, 42% (15/36) of mathematicians responded that they had received training, which was a higher percentage than the 35% (302/869) who answered yes on the URSSI.

In the follow-up question, 40% (6/15) of mathematicians and 38% (115/302) of RSEs reported they were trained in 'Class/school', and another 40% (6/15) of mathematicians and 30% (92/302) of RSEs reported they were trained in both 'Class/school' and through 'Online self-directed' means.

Question 2 asks whether the respondent believes that there are sufficient opportunities available to explore or obtain new software skills. Both surveys reported similar results, with 53% of mathematicians and 52% of RSEs agreeing, and the rest disagreeing or unsure.

Question 3 asks whether the respondent has enough time to take the types of training they need to be successful. On the IRSEMath, mathematicians were split 50/50 (18/36 yes, 18/36 no), versus a much different split of 25% (222/864) saying yes and 75% (642/864) saying no on the URSSI. Although mathematicians were split, URSSI respondents did not believe they had sufficient time to train.

Question 4 asks the types of delivery for various training techniques the respondents preferred the most. For on-site custom training, development techniques were preferred the most,

3.1.6 Research question four - Funding and Institutional Support

Research question four has two subquestions it aims to answer:

- Subquestion A asks about available institutional support for research software development.
- Subquestion B asks what sources of institutional funding are available to research software developers.

Question 1 asks if the participant has included costs for software development in a funding proposal. While mathematicians generally do not with 78% (28/36) choosing no, research software engineers were more split almost evenly with 54% (451/836) choosing no.

If the respondent chose yes, a follow-up question asks them to select from a list of uses for the software development funding in their funding proposals. IRSEMath respondents primarily included costs for developing new software (6/10—60%) or software maintenance (4/10—40%). No IRSEMath respondents mentioned reusing existing software. URSSI respondents chose to include costs for new software development (343/713—48%), software maintenance (210/713—29%), and software reuse (160/713—22%).

Question 2 is omitted.

Question 3 (Figure 12) asks if institutional support is provided for research software development activities. This question is only shown to developers or combination researcher-developers. For financial support, 42% (8/19) of mathematicians report that they are adequately funded and supported, compared to only 14% (60/420) of research software engineers. For infrastructural support, both

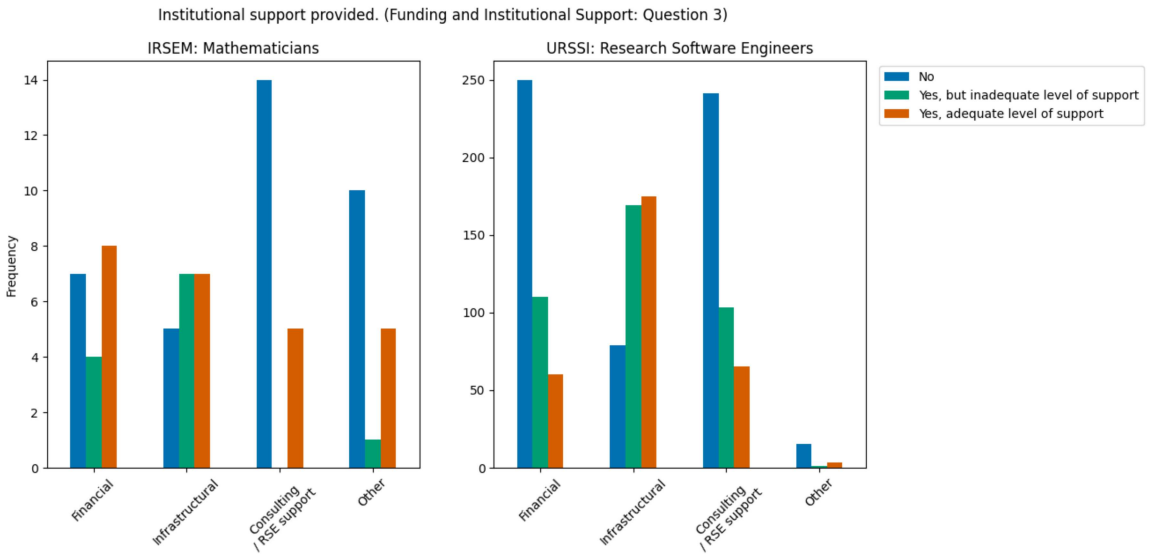

Figure 12. Institutional support provided.

mathematicians and research software engineers report similar levels of support, at 37% (7/19) and 41% (175/423), respectively. For consulting or RSE support, only 26% (5/19) of mathematicians and 16% (65/409) of research software engineers are adequately supported.

Question 4 (Figure 13) asks if the levels of institutional support actually needed for the development of research software are provided. This question is also only shown to developers or combination researcher-developers. For financial support, 37% (7/19) of mathematicians report that the funding they need is sufficient and supported, compared to only 15% (60/420) of research software engineers. For infrastructural support, both mathematicians and research software engineers report similar levels of support, at 37% (7/19) and 33% (175/423), respectively. For consulting or RSE support, only 26% (5/19) of mathematicians and 16% (65/409) of research software engineers are adequately supported.

Question 5 is a free response question that asks about the most significant gaps in the current institutional or organizational support of software-related

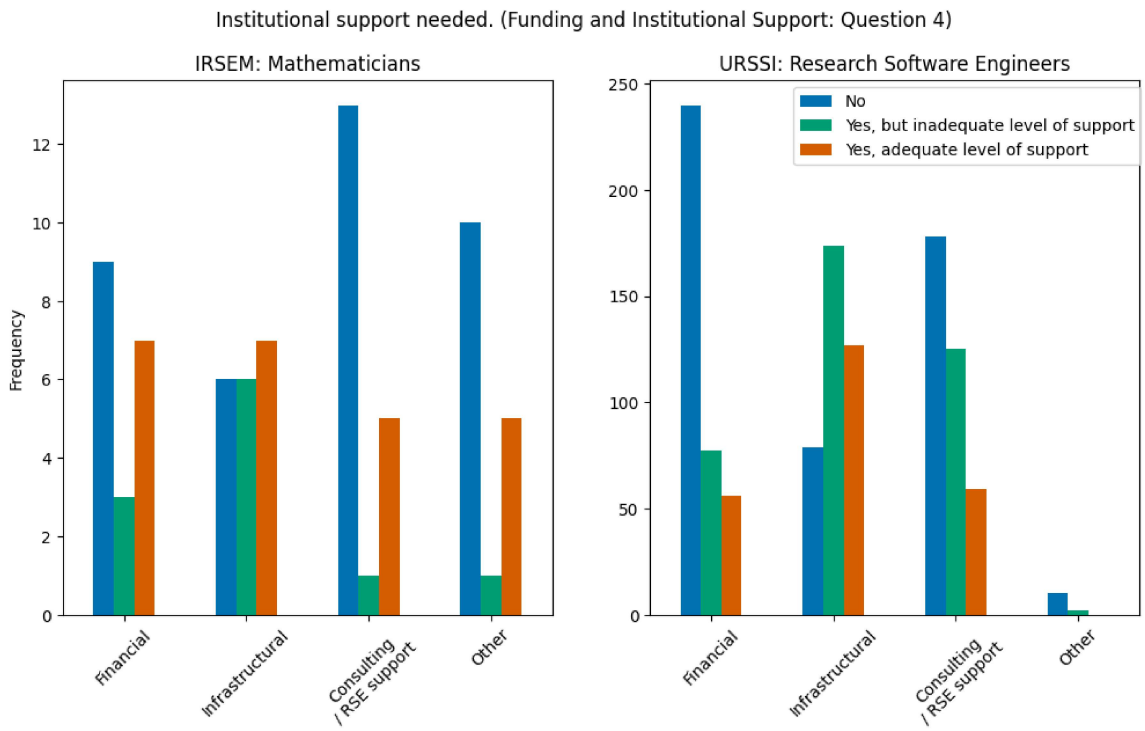

Figure 13. Institutional support needed.

research. Both URSSI and IRSEMath respondents mentioned several common sentiments in their responses to this question.

- **Funding and resource scarcity**

  URSSI responses:

  - *"There is no support to make our open source software available when grant funding is over (sustainability is lacking)."*
  - *"Creation is good, but it would be good to provide funding for documentation, more and better working examples, and for refactoring and maintenance."*
  - *"Funding is not there, and there isn't much reward for software development..."*

IRSEMath responses:

- *"No grants," "lack of money," "financial."*
- *"Lack of funding for proprietary software."*

Both mathematicians and RSEs frequently reported poor financial support for software development, maintenance, documentation, and long-term sustainability. They rely on unstable grant funding with no institutional backing, have a lack of budgets for maintenance, refactoring, or open-source sustainability, and their salaries for RSE work are not competitive compared to industry pay.

- **Lack of recognition**

  URSSI responses:

  - *"No recognition of software development / maintenance during the tenure and promotion process. If it does not have dollars associated... it is deemed less important."*
  - *"The promotional track is not as well defined for faculty who primarily write software."*
  - *"Software engineers are treated as second class citizens."*

  IRSEMath responses:

  - *"It's better to crank out research papers than code."*
  - *"Work done on developing software [...] is not appreciated enough."*

  Respondents commonly expressed that software contributions are rarely acknowledged in their tenure/promotion decisions, and that software development in general is seen as a "side activity" rather than intellectual

work. There are no clear career paths for RSEs and there are no metrics to evaluate the impact of their software.

- **Time constraints and unrealistic expectations**

  URSSI responses:

  - *"Recognition of how long it takes to make good software. Unrealistic delivery timelines."*
  - *"My own time is so limited... Collaborators often see engagement as a losing proposition."*

  IRSEMath responses:

  - *"Lack of time [...] It's better to crank out research papers."*
  - *"Arcane process for applying for reassigned time."*

  Institutions and other funders fail to account for the time required for professional software development, testing, and documentation. These time constraints pressure researchers to prioritize publications over sustainable software, leading to maintenance and support being deprioritized after the initial development process.

- **Training and expertise gaps**

  URSSI responses:

  - *"Most students only know a language. They aren't aware of... version control, unit testing, etc."*
  - *"Basic software engineering skills not introduced to new staff."*

URSSI respondents reported that institutions do not teach researchers and students the skills for modern practices, such as version control, unit testing, and continuous integration / continuous deployment (CI / CD). In addition, institutions are overly dependent on self-taught faculty and temporary students for support. Advanced training for research needs is limited. IRSEMath respondents did not address this topic when discussing gaps in support.

- **Poor infrastructure and technical support**

  URSSI responses:

  - *"Windows only platforms... No one is doing Java, no one is doing/teaching Cuda, C, etc."*
  - *"Problems with licensing software tools... denial of permission to purchase licenses."*
  - *"We don't have a repository where code can be uploaded, preserved, and run virtually."*

  IRSEMath responses:

  - *"IT does not support Linux, and a lot of mathematical research software is more easily available on Linux. My IT in particular has also failed to install version of the software that could run on Windows. :-("*
  - *"Lack of computing power."*

  Researchers are limited by outdated and restrictive systems and have subpar access to high-performance computing (HPC) and cloud computing resources. Sluggish approval processes and restrictive institutional policies for hardware

and software hinder research progress and lead to an overreliance on legacy systems.

- **Sustainability and maintenance challenges**

  URSSI responses:

  - *"Always have to 'innovate' on the next project, rather than enhance/tune/debug/support existing frameworks."*
  - *"Maintenance cost is always ignored."*

  Institutions and other funders push RSEs toward finding new ideas, rather than iterating on and improving existing technology. This is further exacerbated by the disregard for maintenance costs in project funding.

- **Cultural resistance and communication barriers**

  URSSI responses:

  - *"Lack of communication between researchers and professional software developers."*
  - *"Lead scientists are not aware of software issues... focus is always on scientific obstacles."*
  - *"Many researchers... do not align with modern good practices."*
  - *"Program managers have no concept of CI, Unit testing etc.. ie they don't know what good SE practices entail."*

  IRSEMath responses:

  - *"People aren't aware of or interested in using software in their research."*

Researchers are further restricted by a lack of effective communication. Leadership often does not grasp the importance of software in research, and thus neither incentivize nor support modern software engineering practices. Additionally, bureaucratic inertia and "not invented here" attitudes stifle progress.

3.1.7 Research question five - Career Paths

Research question five asks what factors impact career advancement and hiring in research software. Its questions are designed to provide information about career support and career pathways for research software developers.

Question 1 asks respondents to list all the available positions at their institution for people whose main role is to develop research software. IRSEMath respondents most commonly reported that the available positions for developing research software were postdocs (16/66—24%), RSEs (14/66—21%), research faculty (10/66—15%), or software developers (9/66—14%). URSSI respondents responded that those positions are postdocs (411/1969—21%), software developers (253/1969—13%), programmers (253/1969—13%), or research programmers (251/1969—13%).

Question 2 asks if there are opportunities in the respondent's institution for career advancement, such as tenure or more senior positions, for people whose primary role is to develop research software. The majority of IRSEMath respondents responded that they either did not know or would have to change institutions if they wanted advancement (25/33—76%). URSSI respondents matched this sentiment (606/783—77%).

Question 3 (Figure 14) asks how important different concerns are with regard to hiring and maintaining a highly qualified software development staff. Combining "very important" and "extremely important", the main hiring concerns

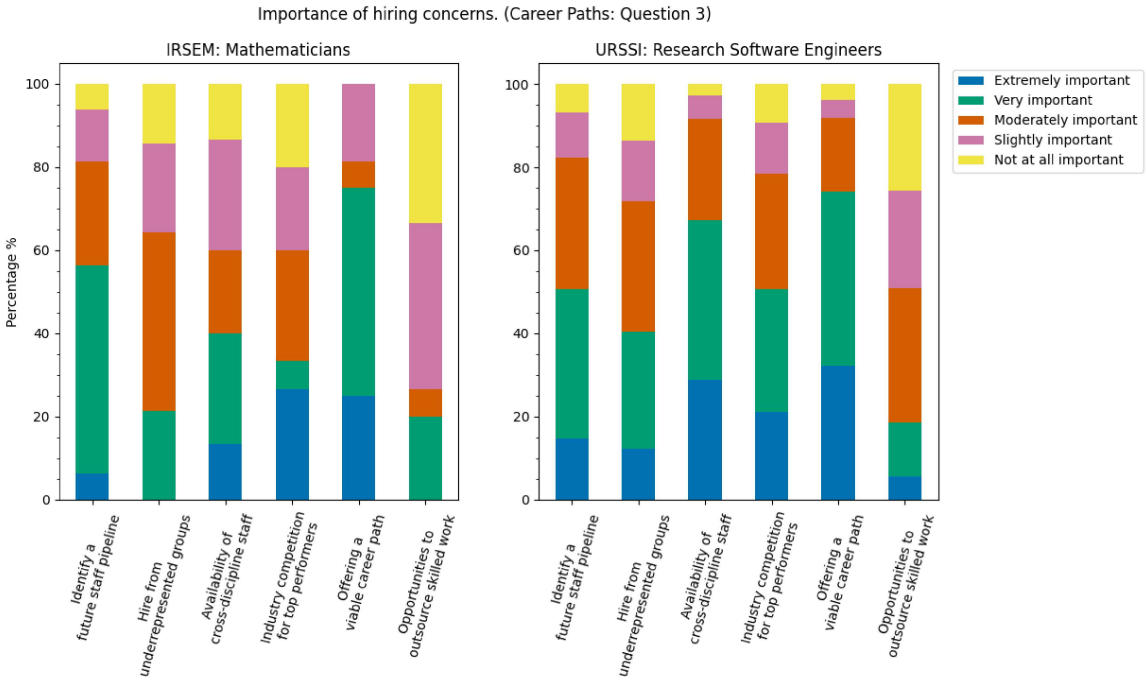

Figure 14. The importance of various hiring concerns.

mathematicians see when hiring software development staff were offering a viable career path (12/16—75%), identifying a future staff pipeline (9/16—56%), and the availability of cross-discipline staff (6/15—40%). Research software engineers had similar main hiring concerns: offering a viable career path (485/654—74%), availability of cross-discipline staff (444/661—67%), and the identification of a future staff pipeline (332/654—51%).

Question 4 (Figure 15) asks the respondent about the concerns from when they were hired to their current position and how important they were. On both surveys, diversity in the organization is not an important factor, with only 6% (1/16) of IRSEMath respondents and 14% (57/403) of URSSI respondents deeming it very or extremely important. Respondents to both surveys agreed programming or software engineering experience was somewhat important, with 46% (7/16) of mathematicians and 32% (133/406) of RSEs marking it as important. Having a

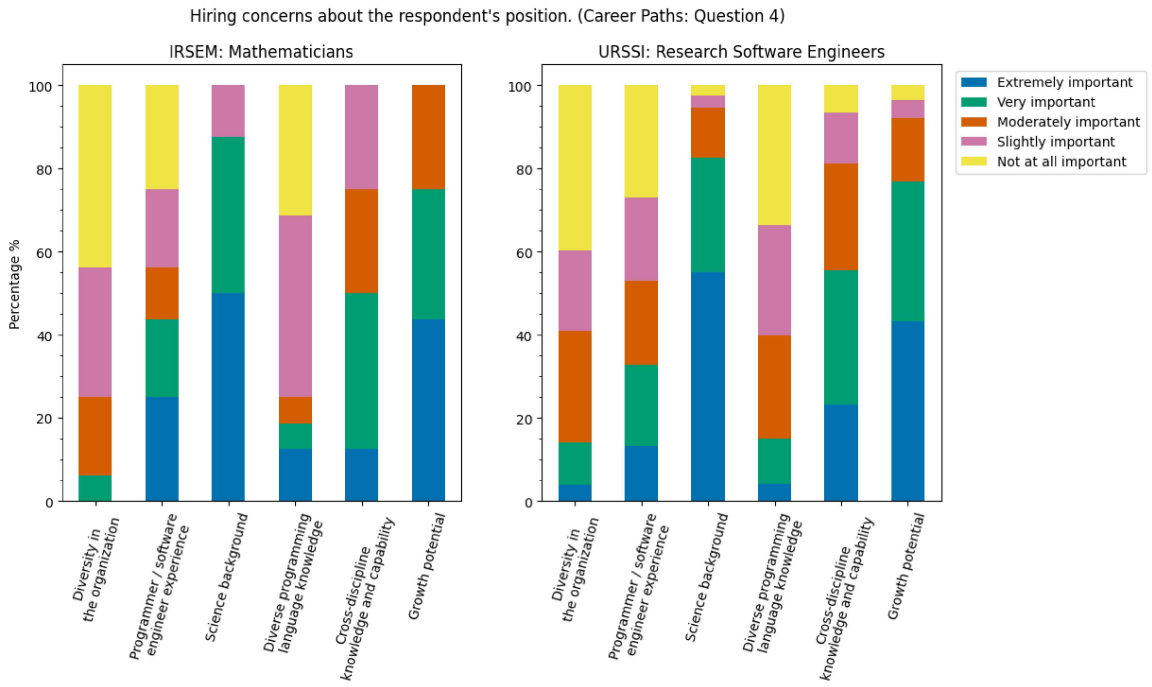

Figure 15. The importance of various hiring concerns about the respondent's current position.

scientific background was considered the most important concern for recruitment, with 88% (14/16) of mathematicians and 83% (335/406) of RSEs rating it as important. Diverse programming language knowledge was not a high priority, but was slightly more important to mathematicians (3/16—18%) than to RSEs (61/405—15%). Interdisciplinary knowledge and capability were the third most important hiring concern for both mathematicians (8/16—50%) and RSEs (225/405—55%). Potential for growth was the second most important concern to mathematicians (12/16—75%) and RSEs (311/405—77%).

3.1.8 Research question six - Credit for Software Contributions

Research question six has two subquestions that elicit information about the citation and attribution of research software:

- Subquestion A asks what research software projects require when crediting or attributing software use

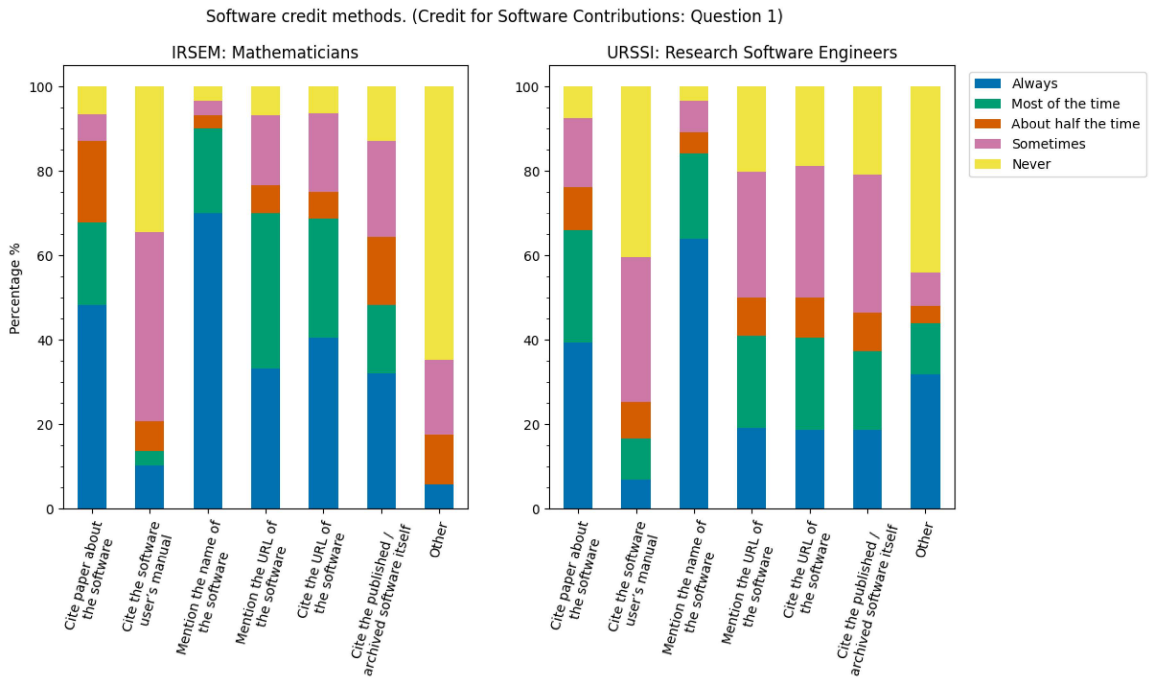

Figure 16. Software credit methods.

- Subquestion B asks how individuals and groups are given institutional credit for developing research software.

Question 1 (Figure 16) asks how survey respondents approach mentioning software when writing papers. Citing a paper about the software is a very common practice for both mathematicians (21/31—67%) and RSEs (490/746—66%). Citing the software user's manual is quite uncommon (IRSEMath 4/29—13%, URSSI 121/725—17%). Mentioning the name of the software is the most frequent practice when crediting software (IRSEMath 27/30—90%, URSSI 625/743—84%). Mentioning (IRSEMath 21/30—70%, URSSI 297/726—41%) or citing (IRSEMath 22/32—69%, URSSI 295/729—41%) the URL of the software are common practices, especially for mathematicians. Citing the published or archived software itself was common for both mathematicians (15/31—48%) and RSEs (272/729—38%). Overall, mathematicians credited software more often than RSEs.

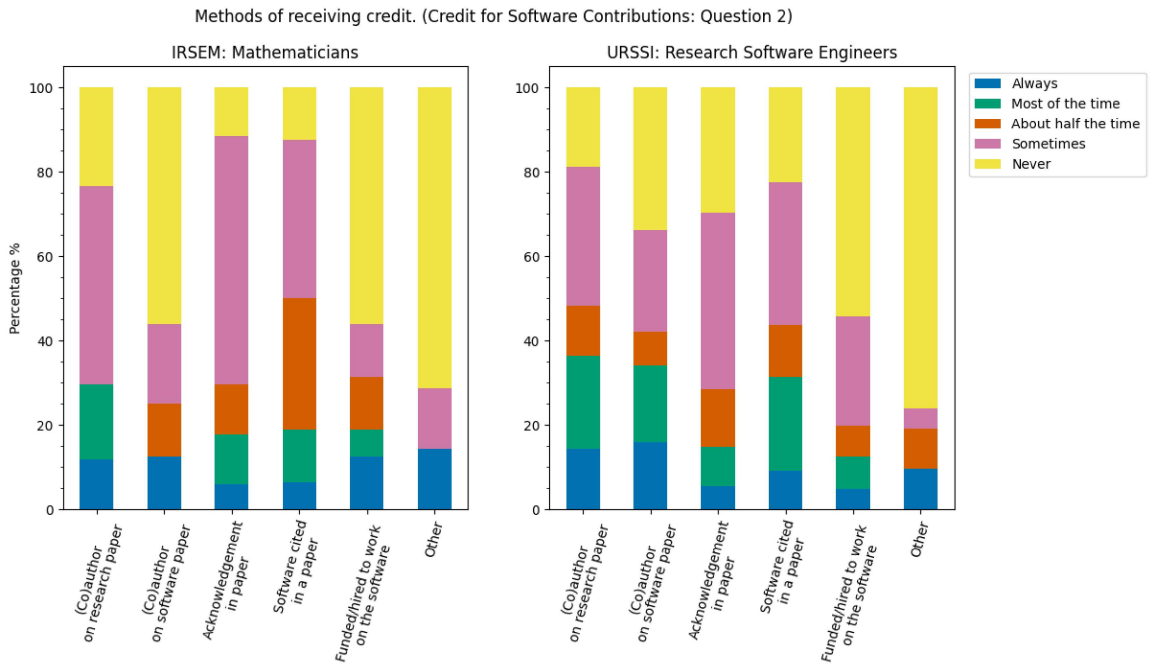

Figure 17. Methods of receiving credit.

Question 2 (Figure 17) asks how often and in what ways respondents currently receive credit for software contributions. Respondents to the URSSI and IRSEMath surveys received credit in relatively similar ways. When coauthoring research papers, mathematicians reported receiving credit slightly less often (5/17—30%) than researchers (146/402—36%). When coauthoring software papers, mathematicians reported much lower rates (2/16—12%) than researchers (135/396—34%). The acknowledgment rates in papers for the respondents of both surveys were low but similar. (IRSEMath: 3/17—18% vs. URSSI: 59/399—15%). Mathematicians reported much lower rates of their software being cited in papers (IRSEMath: 3/16—18% vs. URSSI: 124/395—31%). A somewhat larger proportion of mathematicians (3/16—18%) reported being funded or hired to work on software than researchers (49/391—13%).

Question 3 asks if the respondent's institution allows software contributions to be considered in performance reviews or promotion cases for their job role. Respondents to both surveys agreed that this was generally dependent on the positions of programmer versus faculty (IRSEMath 12/31—39%, URSSI 289/744—39%) with an inclination towards yes (IRSEMath 10/31—32%, URSSI 290/744—39%).

Question 4 asks if the respondent believes software contributions are important for their own performance reviews or promotion cases. Mathematicians believed contributions to by slightly (12/31—39%) or moderately important (11/31—35%). Research software engineers either did not believe contributions were important (265/757—35%) or believed they were only slightly important (197/757—26%).

3.1.9 Research question seven - Diversity and Inclusion

Research question 7 asks how current research software projects document diversity statements and what support is needed to further diversity initiatives based on the software project the respondent works on the most.

Question 1 (Figure 18) asks how well the respondent's project recruits and retains participants from underrepresented groups, includes those people in governance or leadership positions, and maintains a culture of inclusion. Mathematicians found that the efforts to recruit participants from minority groups were average (11/22—50%) or poor (9/22—41%), compared to RSEs determining that the efforts were good (152/563—27%) or average (193/563—34%). Retaining these participants was average (10/21—48%) or poor (5/21—24%) for mathematicians, and good (173/541—32%) or average (194/541—36%) for RSEs. Including them in governance or leadership positions was average (7/21—33%) or poor (9/21—43%) for mathematicians, and good (137/494—28%) or average

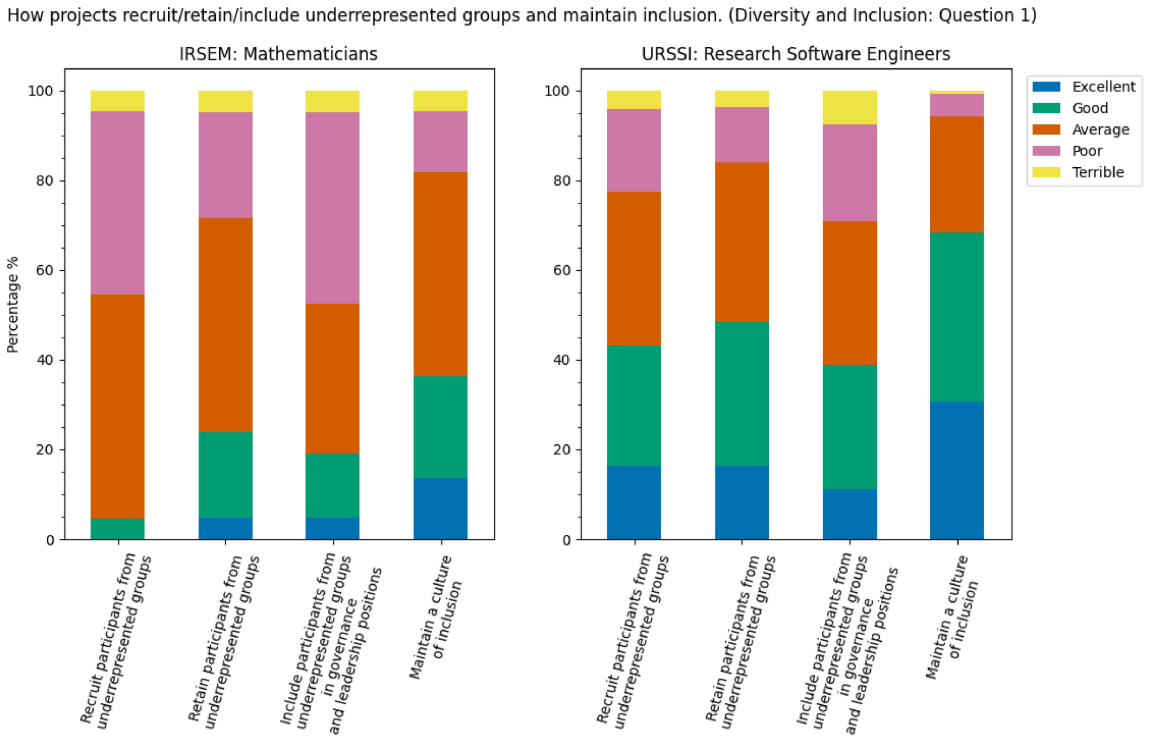

Figure 18. Recruiting, retaining, and including participants from underrepresented groups.

(159/494—32%) for RSEs. Maintaining a culture of inclusion was better for mathematicians (good: 5/22—23%, average: 10/22—45%), but RSEs reported even better results (excellent: 175/573—31%, good: 216/573—38%).

Question 2 is a free response question that asks the challenges researchers face in creating and maintaining a culture of inclusion. Several different themes appeared in a brief analysis of participants' responses to both the IRSEMath and URSSI surveys. The main challenges facing the participants of the URSSI survey revolve around systemic pipeline issues, institutional resistance, poor leadership, systemic bias, and resource limitations in terms of salary and location. In comparison, IRSEMath participants refer to the same challenges of pipeline issues and systemic biases, but also add problems of individual behavior. Deeper analysis

and comparisons can be performed on the vast amount of responses to the surveys, especially the URSSI data.

Question 3 is a free response question that asks if the researcher's project makes any measurements, tracking, or analysis of developer or user communities with regard to diversity and/or inclusion. IRSEMath respondents universally disagreed or expressed distaste for the notion. URSSI respondents also mainly disagreed, but a number of respondents referred to specific efforts to collect demographic data to satisfy federal research requirements or for professional or institutional research purposes.

3.2 Conclusions

The analysis of the IRSEMath and URSSI surveys reveals both shared and distinct challenges in RSE between mathematics and broader scientific communities. Although mathematicians exhibit stronger adoption of open-source licensing (95% vs. 74%), they face harsh institutional barriers, including inadequate funding for long-term software maintenance (only 42% report sufficient financial support) and minimal recognition of software contributions in career advancement (76% of mathematicians cite unclear pathways to career advancement). These findings align with broader RSE struggles but are exacerbated by the culture of mathematicians, where software development is often perceived as inferior to theoretical work.

CHAPTER IV

PI-BASE: A COMMUNITY DATABASE OF TOPOLOGICAL COUNTEREXAMPLES

π-Base is a community database of topological counterexamples, which models mathematical research on topological spaces, their properties, and theorems connecting those properties to each other [7]. To describe the topics that the database describes, we can use an analogy with geometry, considering geometric shapes in place of topological spaces.

Definition 1. *A polygon is a* ***square*** *provided it has with four sides of equal length, and four right angles.*

Theorem 1. *Every square is a rhombus.*

If an object has properties, we can use the database to find the other implied properties using logical reasoning.

A computer can also do that, given the data and a set of rules. The π-Base is a database that collects and categorizes hundreds to thousands of these relations between spaces and connects them in a computer-readable format. In other words, if one were to look up a shape that is recorded to be a square, pi-base would also report that it is also a rhombus and a rectangle. The opposite is true as well, where looking up a shape that is tagged as a rhombus and a rectangle would return that it is also a square.

The π-Base database and software application are maintained by Steven Clontz and James Dabbs [8]. The π-Base would neither function nor exist without its supporting social infrastructure. At regularly scheduled "community calls", volunteer contributors meet to discuss the database's software, mathematical content, events, and important subjective decisions. Community calls are crucial to actively maintaining the database and improving the way data is represented to users. Not all discussions are mathematical questions, but they are important for database sustainability. Recent examples include discussions of new terminology when labeling spaces, how to assign who reviews pull requests, adding a badge feature to ID links, and a lively discussion about choosing the preferred logical representation of theorems when both options are equivalent through De Morgan's laws. This topic was discussed in particular because there were equivalent representations in the database of theorems in the forms "NOT (p AND q)" and "(NOT p) OR (NOT q)". After a consensus was reached, the internal representations were updated.

CHAPTER V

DEVELOPMENT CONTRIBUTIONS TO π-BASE

During the Fall 2024 semester, the author worked with committee member Dr. Steven Clontz to develop multiple contributions to the π-Base's open-source repository hosted on GitHub. Built using Svelte (a web framework leveraging TypeScript), the platform follows a continuous integration and continuous deployment (CI/CD) pipeline, enabling seamless implementation of new features to its hosting service Cloudflare without any service interruptions. As an open-source project, π-Base encourages community participation through its freely accessible GitHub repository, where contributors can directly propose features or access the complete codebase.

The π-Base GitHub repository utilizes an issues page that functions as a public task board for changes, fixes, and feature requests. Any member of the community can claim or propose issues. Once assigned, contributors can create a personal fork of the main repository branch to begin work. Often, these issues require investigation to find relevant files, make modifications, and add any new files to implement the feature or fix [9].

Reviewing code is easily done using GitHub.dev, a lightweight web version of Visual Studio Code (VS Code), which enables easy navigation of the repository's file structure. However, non-trivial contributions using GitHub.dev are inadvisable, as code cannot be tested in that environment. This is best performed using GitHub Codespaces, which enables browser access to a virtual machine in the cloud that is

accessed through a browser-based VS Code instance. With GitHub Codespaces, changes can be easily developed and tested [10].

Once the work has been completed in a codespace, the changes can be published to a dedicated branch of the main project linked to the original issue. Automated CI/CD pipeline checks then validate the code against standardized tests, ensuring compatibility with existing systems. If successful, builds are peer reviewed by a project maintainer. After successful automated and human validation, approved branch changes can finally be merged into the main branch, deploying seamlessly to the live repository.

Throughout the development and research portion of this honors thesis, the author learned the development workflow, both choosing and being assigned features to develop for the website, which were integrated into the site's codebase using standard development pipeline procedures.

This honors thesis involved active engagement with π-Base's development workflow, where the author gained proficiency in GitHub-based development processes. By implementing assigned features, the author contributed multiple enhancements to the live codebase. These contributions adhered to the project's CI/CD pipeline protocols, ensuring peer-reviewed and tested integrations with the production environment.

5.1 Citation Button

On every page on the π-Base website, there is a line that gives the citation for the given page, along with a button to copy that citation in Markdown format. There was an issue documented on the π-Base GitHub where the button to create the citation for the current web page was not updating according to which part of the website the user was accessing.

```
packages/viewer/src/components/Shared/Cite.svelte

@@ -3,7 +3,8 @@
3   3     import CopyButton from './CopyButton.svelte'
4   4     let url = ''
5   5     let markdown = ''
6     -   export let title: string | undefined = undefined
    6 +   export let title: string =
    7 +     'π-Base, a community database of topological counterexamples.'
7   8     onMount(() => {
8   9       url = window.location.href
9   10      markdown = `[${title}](${url})`
@@ -15,12 +16,7 @@
15  16      Cite as:
16  17      <span class="text-muted">
17  18        The pi-Base Community.
18    -     {#if title}
19    -       <cite>{title}.</cite>
20    -     {:else}
21    -       <cite>π-Base, a community database of topological counterexamples.</cite
22    -       >
23    -     {/if}
    19 +     <cite>{title}.</cite>
24  20        Available at: {url}
25  21        (Accessed: {new Date().toISOString().split('T')[0]}).
26  22      </span>
```

Figure 19. Citation button code changes

To find what was causing the bug, the author with his mentor researched relevant Svelte documentation, along with exploring the code on other parts of the π-Base web repository. From this exploration, the author found that the markdown citation button was not updating properly because it was initialized at startup (onMount). The author then implemented a solution to extract the website URL information from the live website data rather than using hardcoded variables, which eliminated several blocks of unnecessary logic (Figure 19, 20). Once the changes passed the CI / CD tests and manual approval, the website was updated to implement the changes and fix the button to update properly as users navigated the website (Figures 21, 22) [11].

```
packages/viewer/src/components/Shared/Cite.svelte

@@ -1,14 +1,9 @@
<script lang="ts">
-   import { onMount } from 'svelte'
+   import { page } from '$app/stores'
  import CopyButton from './CopyButton.svelte'
-   let url = ''
-   let markdown = ''
  export let title: string =
    'π-Base, a community database of topological counterexamples.'
-   onMount(() => {
-     url = window.location.href
-     markdown = `[${title}](${url})`
-   })
+   $: markdown = `[${title}](${$page.url})`
</script>

<div class="text-center p-2">
@@ -17,7 +12,7 @@
      <span class="text-muted">
        The pi-Base Community.
        <cite>{title}.</cite>
-       Available at: {url}
+       Available at: {$page.url}
        (Accessed: {new Date().toISOString().split('T')[0]}).
      </span>
    </small>
```

Figure 20. Additional citation button code changes

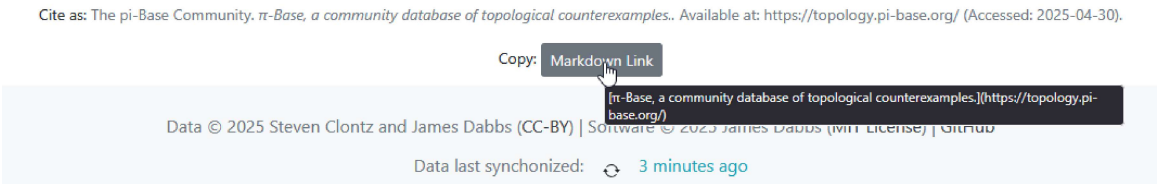

Figure 21. The citation button on the π-Base homepage

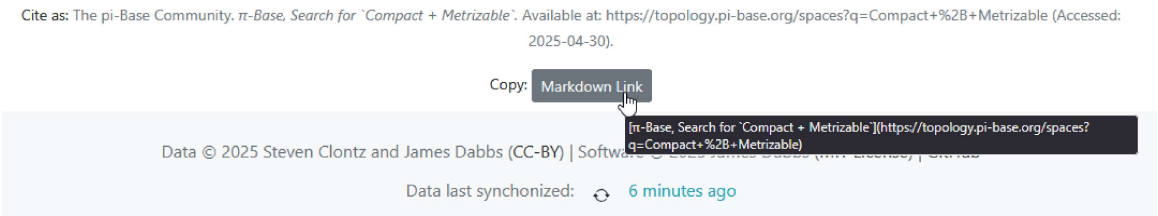

Figure 22. The citation button on the explore page, properly updated to link to search results

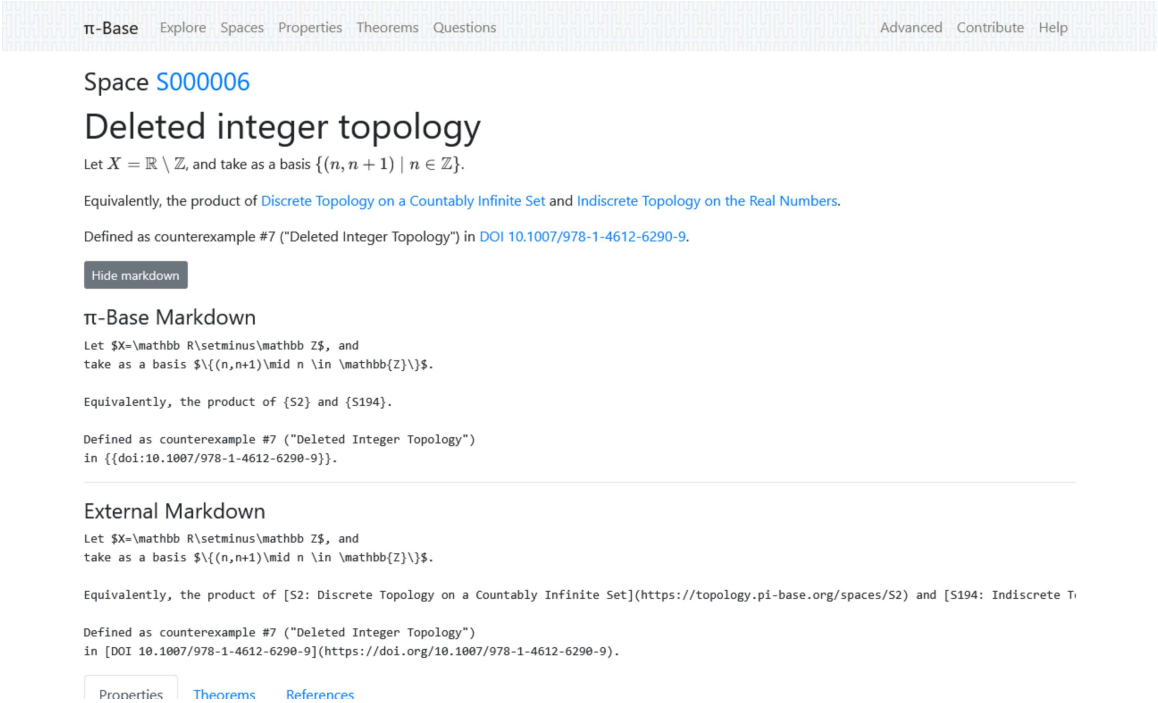

Figure 23. The 'Show Markdown' button showing standard formatted Markdown beneath the π-Base Markdown

5.2 External Markdown

The second feature the author designed for π-Base was the External Markdown feature. The feature is utilized in every location where markdown can be previewed, and can be found on the π-Base editor preview page, under the website's "Advanced" section. On the editor preview page, there is an editable text box that shows several prewritten examples of custom "π-Base-flavored Markdown", a guide for the π-Base syntax, and a live text box showing a preview of the Markdown. To put it simply, this tool takes the internal π-Base formatting and converts it to a standard format that can be copied and pasted onto websites such as *StackExchange*.

The External Markdown feature is enabled by the "Show markdown" button and displays an additional preview below the π-Base Markdown that reformats the custom Markdown into the standard format (Figures 23).

In designing this feature, the author reworked code snippets used to build the citation button mentioned previously in combination with existing code used to render π-Base Markdown to add a new toggle to render the external markdown beneath the π-Base markdown. These changes were across several files, as the external markdown button appears on several different π-Base object pages, including spaces, properties, and theorems (Figures 24, 25) [12] [13].

1 file changed +29 -8 lines changed

packages/viewer/src/components/Shared/ExtMarkdown.svelte

```
export let source: string
const modify = (s: string) => {
-  const singlecurly = /{(?<type>[SPT])(?<id>\d+)}/g
-  const doublecurlies = /{{(?<prefix>\w+):(?<id>[^{]+)}}/g
+  const internal_trait = /{S(?<spaceID>\d+)\|P(?<propID>\d+)}/g
+  const internal_SPT = /{(?<type>[SPT])(?<id>\d+)}/g
+  const external = /{{(?<prefix>\w+):(?<id>[^{]+)}}/g

  // Replace double curly braces
-  s = s.replace(doublecurlies, (_, prefix, id) => {
+  s = s.replace(external, (_, prefix, id) => {
    // Perform replacement based on prefix and id
    const ref = Ref.format({
      kind: prefix,
@@ -16,15 +19,33 @@
    return `[${ref.title}](${ref.href})`
  })

-  // Replace single curly braces
-  s = s.replace(singlecurly, (_, type, id) => {
+  // Replace single curly braces - check traits, then space/property/theorem
+  // Reuses logic from https://github.com/pi-base/web/blob/0c340a5e5c400047acd21acc1ea7a3f75c06733d/packages/viewer/src/parser/internalLinks.ts
+  // TODO: replace duplicate code
+  s = s.replace(internal_trait, (_, sid, pid) => {
+    const space = $spaces.find(Number(sid))
+    const property = $properties.find(Number(pid))
+    const trait = space && property && $traits.find(space, property)
+    const connector =
+      trait?.value === true ? 'is' : trait?.value === false ? 'is not' : '?'
+    return `[S${sid}|P${pid}: ${
+      space ? space.name : 'S' + sid
+    } ${connector} ${
+      property ? property.name : 'P' + pid
+    }](https://topology.pi-base.org/spaces/S${sid}/properties/P${pid})`
+  })
+  s = s.replace(internal_SPT, (_, type, id) => {
    // Perform replacement based on type and id
+
    if (type === 'S') {
-      return `[S${id}](https://topology.pi-base.org/spaces/S${id})`
+      const name = $spaces.find(`S${id}`)?.name
+      return `[S${id}: ${name}](https://topology.pi-base.org/spaces/S${id})`
    } else if (type === 'P') {
-      return `[P${id}](https://topology.pi-base.org/properties/P${id})`
+      const name = $properties.find(`P${id}`)?.name
+      return `[P${id}: ${name}](https://topology.pi-base.org/properties/P${id})`
    } else {
-      return `[T${id}](https://topology.pi-base.org/theorems/T${id})`
+      const name = $theorems.find(`T${id}`)?.name
+      return `[T${id}: ${name}](https://topology.pi-base.org/theorems/T${id})`
```

Figure 24. External Markdown code snippet

```
packages/viewer/src/components/Properties/Show.svelte
26  26
27  27    <section class="description-markdown">
28      -   <Source source={property.description} />
    28  +   <Source source={property.description} internal external />
29  29    </section>
30  30
31  31    <Tabs {tabs} {tab} {rel} />
```

```
packages/viewer/src/components/Shared/Source.svelte
    @@ -2,6 +2,8 @@
2   2       import ExtMarkdown from './ExtMarkdown.svelte'
3   3       export let source: string
4   4       let showMarkdown = false
    5   +   export let internal = false
    6   +   export let external = false
5   7     </script>
6   8
7   9     <button
    @@ -13,8 +15,14 @@
13  15      {#if showMarkdown}Hide{:else}Show{/if} markdown
14  16    </button>
15  17    {#if showMarkdown}
16      -   <h4>π-Base Markdown</h4>
17      -   <pre>{source}</pre>
18      -   <hr />
19      -   <ExtMarkdown {source} />
    18  +   {#if internal}
    19  +     <h4>π-Base Markdown</h4>
    20  +     <pre>{source}</pre>
    21  +     {#if external}
    22  +       <hr />
    23  +     {/if}
    24  +   {/if}
    25  +   {#if external}
    26  +     <ExtMarkdown {source} />
    27  +   {/if}
20  28    {/if}
```

```
packages/viewer/src/components/Spaces/Show.svelte
    @@ -25,7 +25,7 @@
25  25    </section>
26  26
27  27    <section class="description-markdown">
28      -   <Source source={space.description} />
    28  +   <Source source={space.description} internal external />
29  29    </section>
30  30
31  31    <Tabs {tabs} {tab} {rel} />
```

Figure 25. External markdown changes to multiple files

CHAPTER VI

FUTURE WORK

6.1 Further feature improvements for π-Base

The external markdown feature can be improved upon. One known issue is that the code added to implement the feature replicates logic from other parts of the website. To fix this, the code can be rewritten to replace the current logic with a reusable function fitted to both locations of the replicated code. This is known as the Don't Repeat Yourself (DRY) technique, which is an important software development principle. Additionally, there is a documented bug with the current implementation for converting π-Base Markdown into standard Markdown on the editor preview page where if a user links to a part of the website that does not exist, the website freezes and requires a reload for the user [14].

Not all contributions to π-Base during the research and development portion of the thesis were successful. One of the planned features was the addition of a new "any" modifier to the database search logic. π-Base has an 'Explore' page that supports search queries from which a user can list a set of properties connected by logic and receive a list of database entries that match the logic. These properties are 'true' by default and can have modifiers added to change the state of the search term to 'unknown' or 'false.' The goal of this contribution was to add a new 'any' modifier that would include all three variations of a property (true, false, unknown) to broaden the search results.

After experimenting with initial search logic changes, it was quickly determined that the scope of this modification was much more complex than originally intended. The necessary changes to support the new logic would require a broader investigation of the π-Base infrastructure before rewriting the database search functionality.

6.1.1 Graphical representation of π-Base data

One of the biggest "wishlist" features for the π-Base community is a graphical representation of the website's data as an interconnected map of nodes that can be explored and interacted with. This feature already has some of its groundwork already laid out by James Dabbs and further proofs of concept by Danflapjax [15] [16]. Once completed, it would greatly help to map the database, quickly find relationships, and potentially discover new theorems needed to fill in missing connections between the numerous spaces and properties in the database.

6.2 Future research with survey data

Both the IRSEMath and the URSSI survey data have great potential to be further studied to find important insights into researchers' needs in the fields of mathematics and RSE. Indeed, this thesis only begins to recreate all the investigations of the URSSI survey results that were conducted by Carver et al. For example, URSSI data were often partitioned by developer type or sex to create more detailed graphs and show the disparities and limitations in support of various demographics. Additionally, the omitted research questions about time spent and research funding sources for mathematicians still need to be given proper data representation and analysis.

6.3 Data Availability

The data and tools used in this survey adhere to open science principles, ensuring full reproducibility and transparency. Pre-processed versions of the IRSEMath and URSSI datasets are hosted on a dedicated public GitHub repository under a Creative Commons License. This repository includes the full original code, experimentation, and methods used to generate the graphs and comparisons used in this survey [6].

REFERENCES

REFERENCES

[1] J. C. Carver, N. Weber, K. Ram, S. Gesing, and D. S. Katz, "A survey of the state of the practice for research software in the united states," *PeerJ Computer Science*, vol. 8, p. e963, 2022.

[2] J. Carver, S. Gesing, D. S. Katz, K. Ram, and N. Weber, "URSSI Community Survey 2018 Raw Data," Sept. 2021.

[3] Z. Xie, "DeepSeek: Into the Unknown," 2025.

[4] "2025 Joint Mathematics Meetings," 2025. https://JointMathematicsMeetings.org.

[5] K. Bercic, S. Clontz, J. Dabbs, D. Lowry-Duda, and C. Vincent, "code4math: Consortium of Digital Ecosystems for Mathematics," 2025.

[6] A. Miller, "Survey for Improving Research Software Engineering in Mathematics," 2025.

[7] S. Clontz, "Database-Driven Mathematical Inquiry," 2024.

[8] J. Dabbs and S. Clontz, "pi-Base: a community database of topological counterexamples," 2025.

[9] J. D. Steven Clontz, "Issues · pi-base/web," 2025.

[10] S. Clontz, T. Chih, and O. Levin, "Github for Mathematicians," Feb. 2025.

[11] A. Miller and S. Clontz, "Fix citation button on explore page by Not-Abram · Pull Request 170 · pi-base/web," 2024.

[12] A. Miller and S. Clontz, "Added toggles to external and internal markdown previews by Not-Abram · Pull Request 185 · pi-base/web," 2024.

[13] A. Miller and S. Clontz, "External markdown enhancements by Not-Abram · Pull Request 193 · pi-base/web," 2024.

[14] S. Clontz, "DRY up trait naming · Issue 195 · pi-base/web," 2024.

[15] J. Dabbs, "Draw derived proofs by jamesdabbs · Pull Request 81 · pi-base/web," 2023.

[16] danflapjax, "Pi-Base tools / danflapjax — Observable," 2024.

APPENDIX

Appendix A

First Appendix Title

irb@southalabama.edu

UNIVERSITY OF SOUTH ALABAMA

TELEPHONE: (251) 460-6308
AD 240 · MOBILE, AL. 36688-0002

INSTITUTIONAL REVIEW BOARD
September 20, 2024

Principal Investigator:	Abram Miller, Student		
IRB # and Title:	IRB PROTOCOL: 24-459 [2192308-1] Improving Research Software Engineering in Mathematics		
Status:	APPROVED	Review Type:	Exempt Review
Approval Date:	Sep. 20, 2024	Submission Type:	New Project
Initial Approval:	Sep. 20, 2024	Next Report Due:	Sep. 20, 2025
Review Category:	45 CFR 46.104 (d)(2): Research that only includes interaction involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures or observation of public behavior (including visual or auditory recording): i. Information obtained is recorded by the investigator in such a manner that the identity of human subjects cannot be readily ascertained, directly or through identifiers linked to the subjects		

This panel, operating under the authority of the DHHS Office for Human Research and Protection, assurance number FWA 00001602, and IRB #00000286 or #00011574, has reviewed the submitted materials for the following:

1. *Protection of the rights and the welfare of human subjects involved.*
2. *The methods used to secure and the appropriateness of informed consent.*
3. *The risk and potential benefits to the subject.*

The regulations require that the investigator not initiate any changes in the research without prior IRB approval, except where necessary to eliminate immediate hazards to the human subjects, and that **all problems involving risks and adverse events be reported to the IRB immediately!**

Subsequent supporting documents that have been approved will be stamped with an IRB approval and expiration date (if applicable) on every page. Copies of the supporting documents must be utilized with the current IRB approval stamp unless consent has been waived.

Notes:

- 1 -

Generated on IRBNet

Figure 26. IRB Approval form

BIOGRAPHICAL SKETCH

BIOGRAPHICAL SKETCH

Abram Miller was born and raised in Slidell, LA. He received his certificate in A.I., minor in Mathematics, minor in Russian, and B.S. in Computing from the University of South Alabama.