Results of University of Southampton software survey June 2019

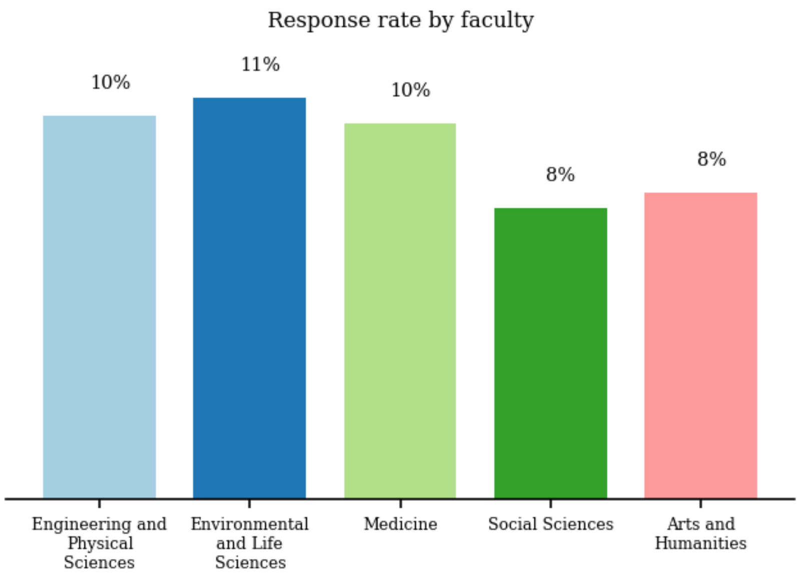
August 16, 2019

1 Introduction

According to a 2014 study, around 92% of UK researchers use software and 69% report that software is fundamental to their research. In 2019, the Southampton Research Software Group conducted a survey of all research staff. The results of this survey show that research software is near ubiquitous at the University of Southampton, that it is judged as vital to research and that the current provision for supporting software engineering could be considerably improved.

1.1 Methodology

The survey was sent to all research staff. In practice this meant sending an email invitation to the mailing lists for all staff employed on an ERE (Education, Research and Enterprise) contract and for all PhD students. The emails were sent on a faculty-by-faculty basis with the following response rate.



The survey asked 16 questions about the researcher and their use of software. This report provides a brief overview of the results.

1.2 Verification and reproducibility

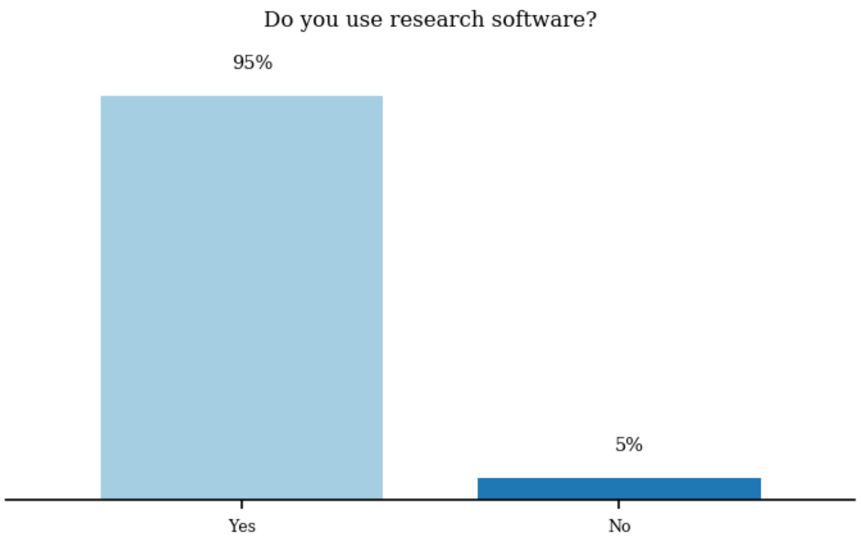
All resources related to this survey are stored on Github:

1. A pdf of the original survey
2. The anonymised data collected by the survey
3. The code used to analyse the software
4. This report on the results of the analysis

The above have been published under an open licence (BSD 3-clause for the code, CC by attribution for the data).

The repository can be found here: https://github.com/Southampton-RSG/soton_software_survey_analysis_2019

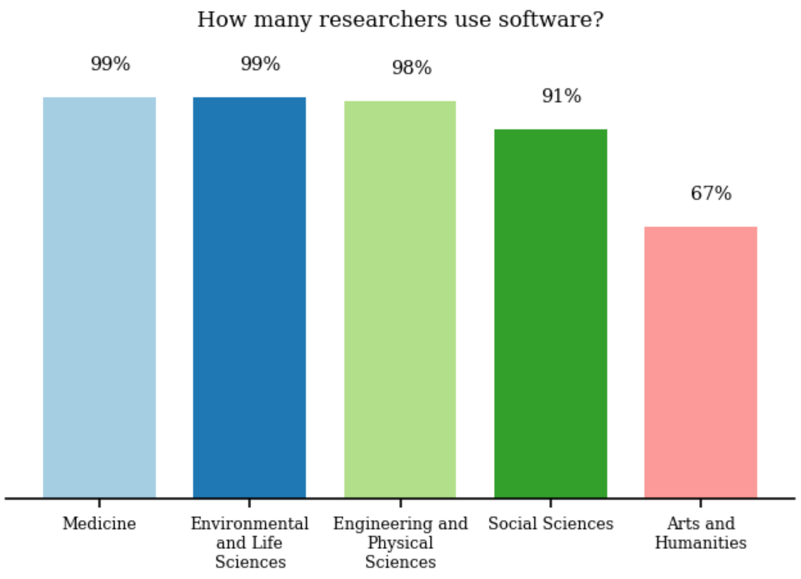
2 Is software important to Southampton researchers?



In this question research software was defined:

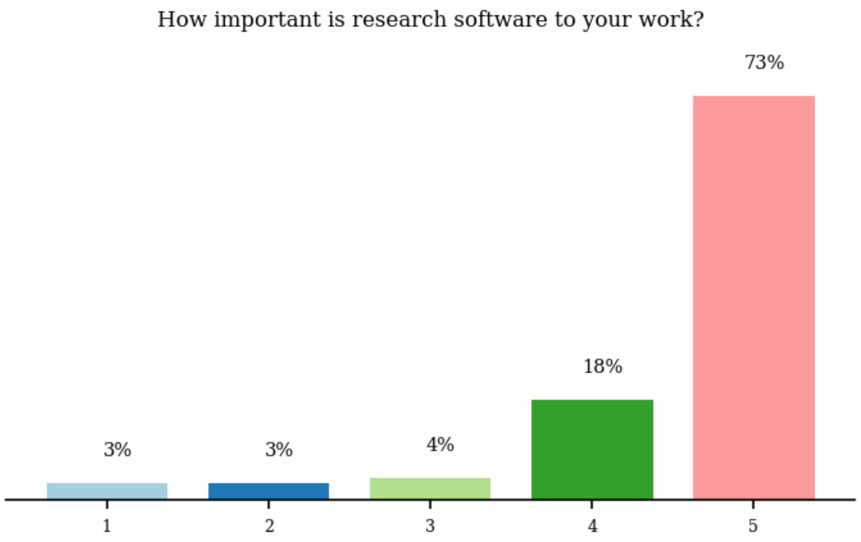
> "any software you have used in the generation of a result that you expect to appear in a publication. This might be anything from a few-line script to clean some data, to a fully fledged software suite. It includes code you have written yourself and code written by someone else."

Almost all researchers at the University use software. Only 5% report that they do not use software, but even this small number seems too large when one consider the ubiquity of software in research. More insight is provided if we segment the data across faculties.

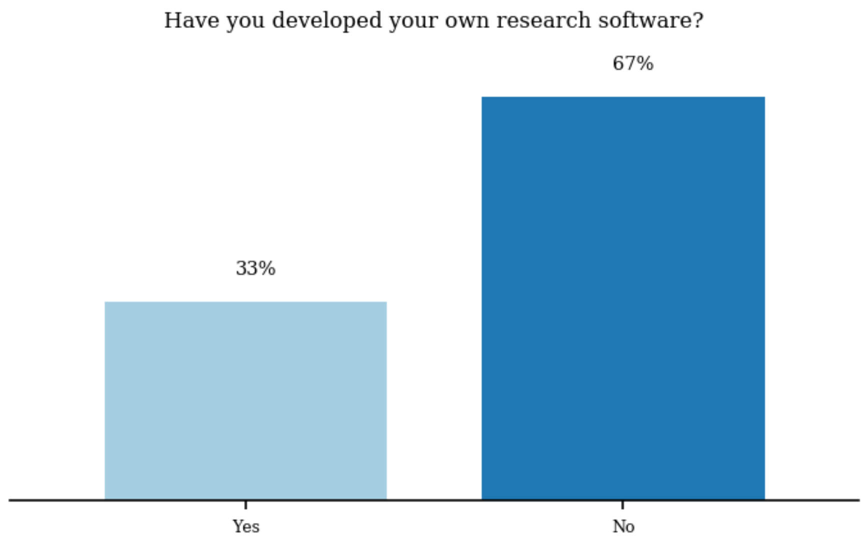


In Medicine, Environmental & Life Sciences, and Engineering & Physical Sciences, the use of software is near 100%. Software use drops slightly in Social Sciences, but it is still above 90%. In Arts and Humanities only 67% report using software in their research, which is understandable in this discipline, although it should be noted that humanities researchers rely heavily on software. The mean percentage of researchers who use software in their research across the university neglecting the contribution from Arts & Humanities is 97%.

Use of software indicates importance, but to investigate this issue further we asked researchers to rate how important their research software was to their research. A rating of 1 was described as "not at all" important and a 5 was described "vital".



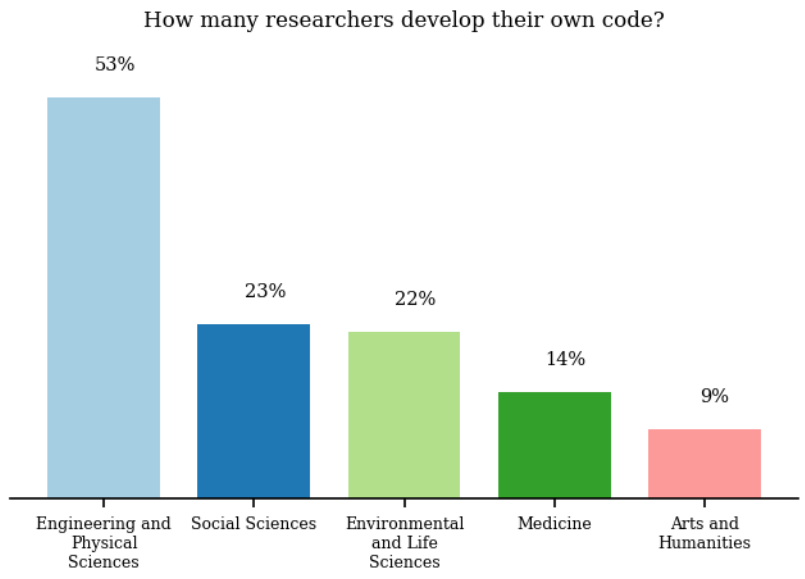
It is clear that software is of fundamental to the research conducted at the University of Southampton: 95% of researchers use software and 73% state that it is vital to their research.



Research problems are often unique, which means they require bespoke software to solve them. Consequently, software development should be encouraged in aademia because it allows researchers to produce the exact software they need, and this saves time, reduces errors and allows research to be conducted at scales that are simply not possible with off-the-shelf software. A third of Southampton researchers develop their own code. It is illuminating to convert this percentage back into numbers of people. There are 198 respondents who develop their own code and 396 who do not.

Conventional software support, which is often provided by the IT department, relies on training a few people to become experts in the few available packages - Word, Excel, etc. - who then provide training to researchers and answer their questions. A proliferation of homegrown software is not suited to the conventional software support model. Instead, it is necessary to provide researchers with access to experts in general software engineering who can quickly understand and provide support with bespoke software.

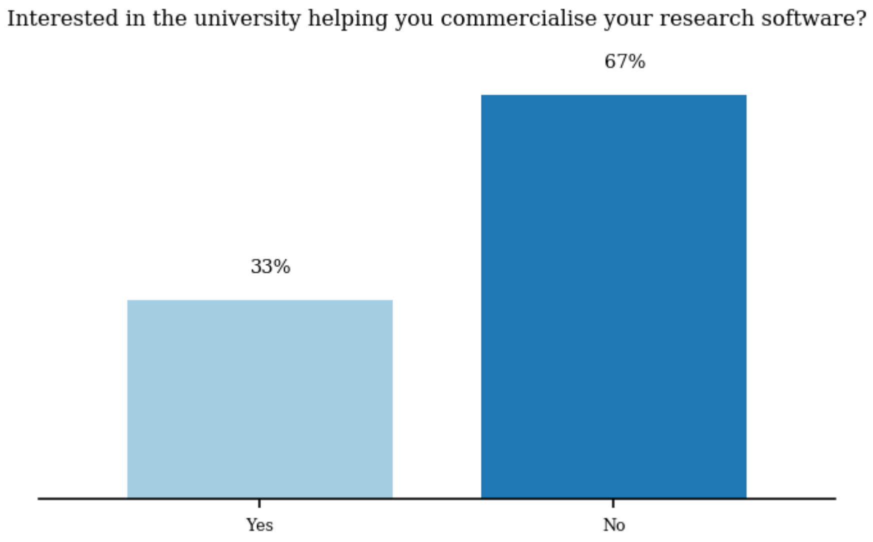
Since software development is a benefit to research, we investigated how it varied across faculties



Engineering & Physical Sciences, Environmental & Life Sciences, Medicine and Social Sciences report similar levels of *software use*, but researchers within Engineering & Physical Sciences are far more likely - over double - to write their own code. There lies in this result a significant opportunity: if researchers from the faculties with high software use can be helped to write their own code, they could significantly improve their research.

One of the obvious factors that dictates whether someone is willing to write their own software is their level of expertise.

3 Potential for commercialising software

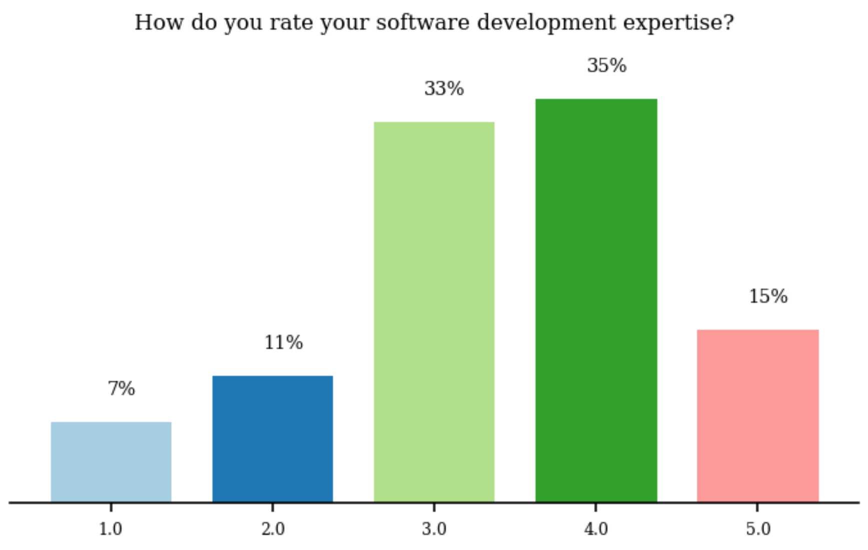


Only 33% of respondents develop their own code, and 33% of these are interested in commercialising their software. Extrapoloating this result over all University research staff indicates that 11% of research staff - around 660 people - have software that they would like to commercialise. Since much research software is developed by lone developers or by very small groups, it seems likely that there could be the potential for 300-600 opportunities for commercialisation across the University. With such a large number of potential codes to draw from, there would appear to be significant potential for commercial success as long as researchers have access to the correct guidance and expertise.

We note that a researcher will only invest time into software development if their research requirements cannot be fulfilled by software that is commercially available to them. This indicates that there is an unfulfilled market for software with that functionality. It does not indicate anything about the size and viability of that market, and conducting this market analysis is one area in which researchers will require help.

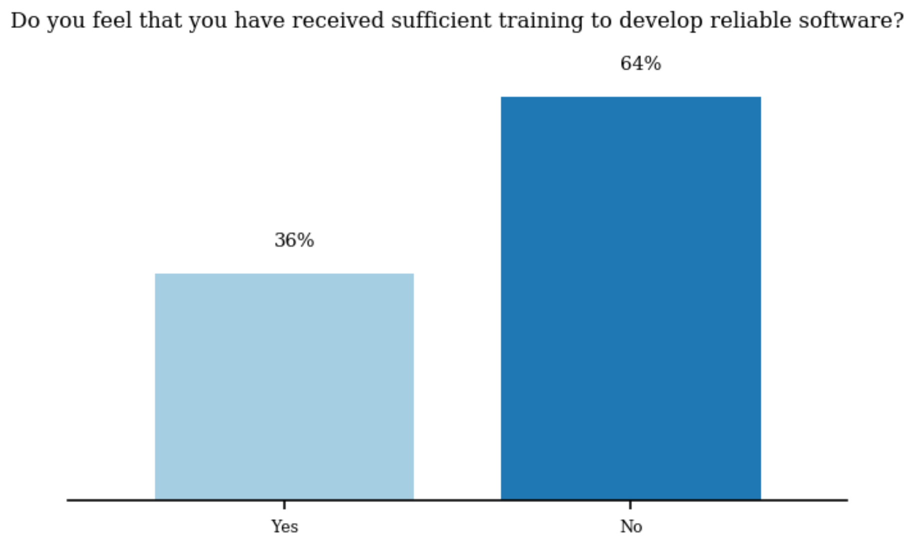
```
[ ]:
```

4 Level of expertise in software



For this question we cateogrised a 1 as “Beginner” and a 5 as “Professional”. Consequently, we can see that 15% of respondents who develop code believe their expertise is professional-level and a further 35% is near-professional level.

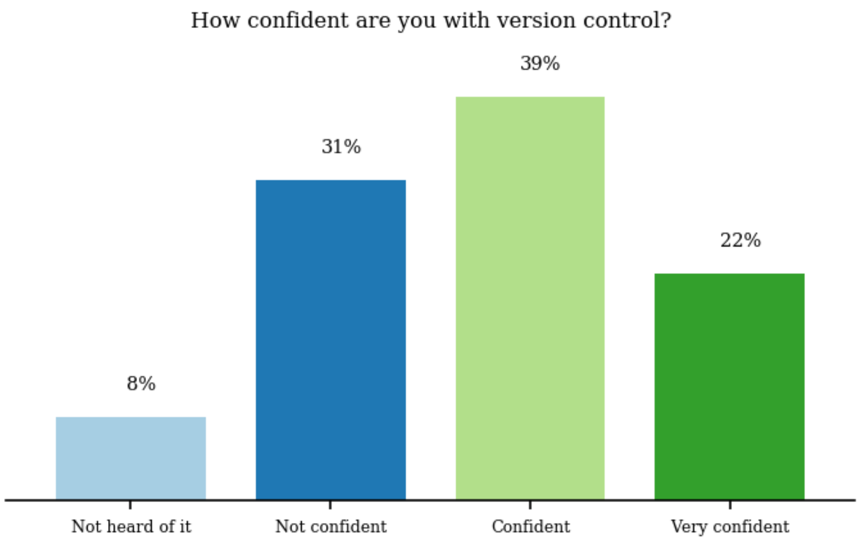
The problem with asking about self-assessed expertise is that it is answered relative to the respondent’s peer group. Someone who may be viewed as a gifted developer within one group might be seen as an amateur in another. Hence, we asked a follow-up question to investigate how confident the respondent was in their code.

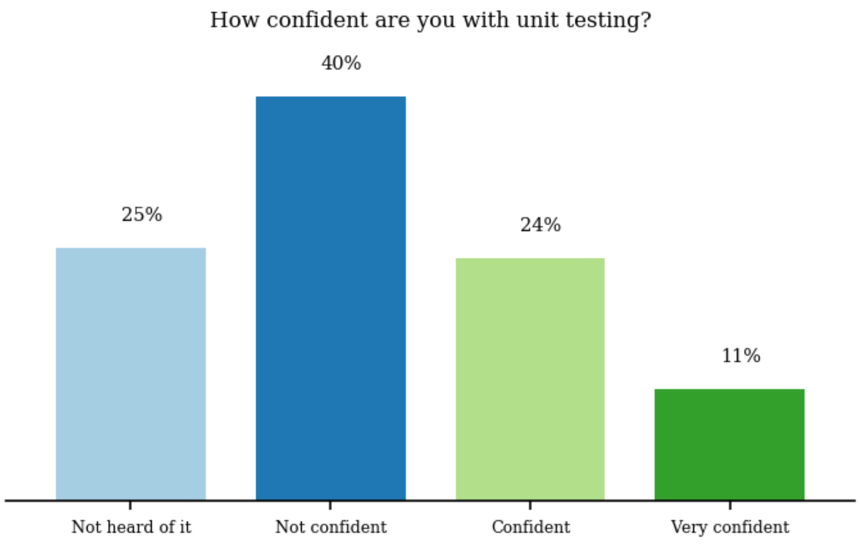


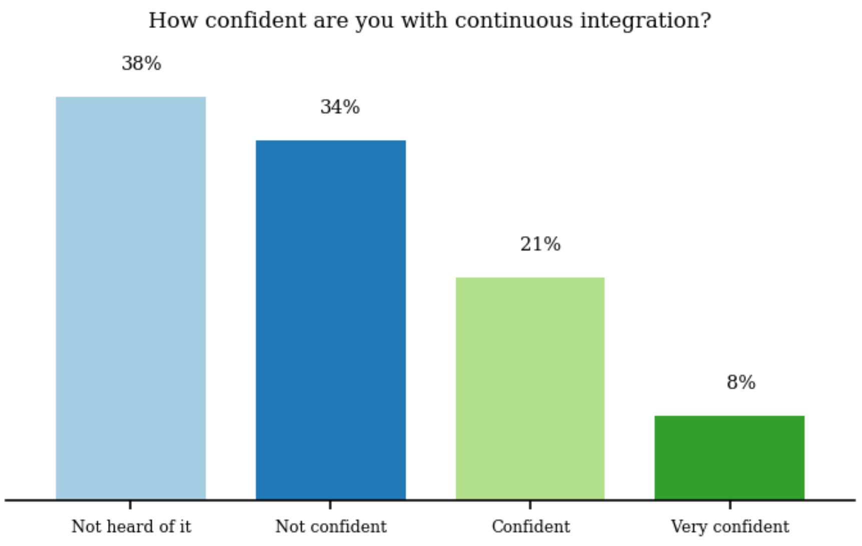
By focussing on the development of reliable software, which surely must be the minimum requirement for any software used in research, we identified that respondents who were previously confident in their development expertise were not nearly as confident that they could write reliable code.

Most researchers who have received training, have received that training in software development: the concepts of how to contruct a program and the syntax on how a programming language works. This allows a researcher to produce code that produces an answer. Software engineering is a completely different discipline to programming. It teaches practices that allow the reliability of software to be understood and maximised. Software engineering is needed to produce code that allows a researcher to produce a verifiable answer.

To complete our analysis of respodnents' expertise with software, we asked questions about three basic software engineering techniques and technologies: version control, unit testing and continuous integration. It is not necessary to understand these concepts to understand the results. It is sufficient to say that anyone who engineers code will certainly have heard of all three.


How confident are you with version control?
8%
31%
39%
22%
Not heard of it
Not confident
Confident
Very confident

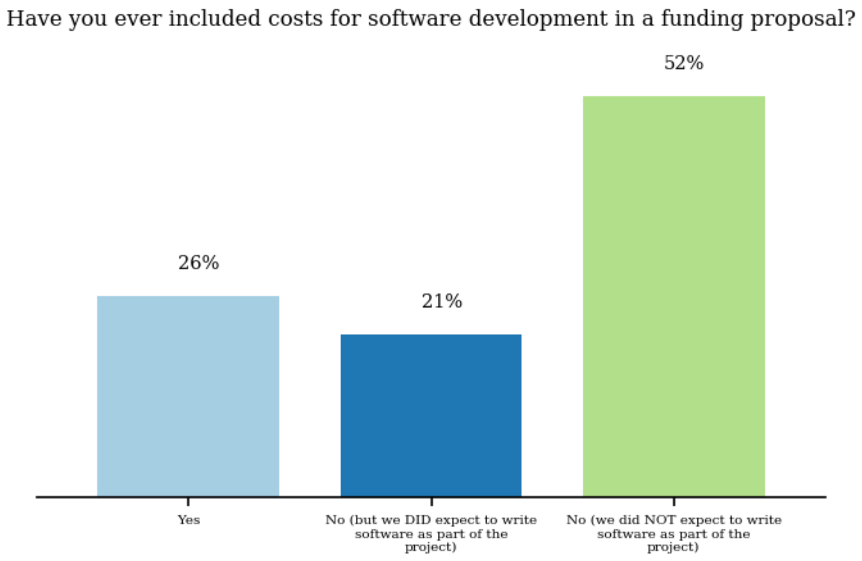

How confident are you with unit testing?
25%
40%
24%
11%
Not heard of it
Not confident
Confident
Very confident

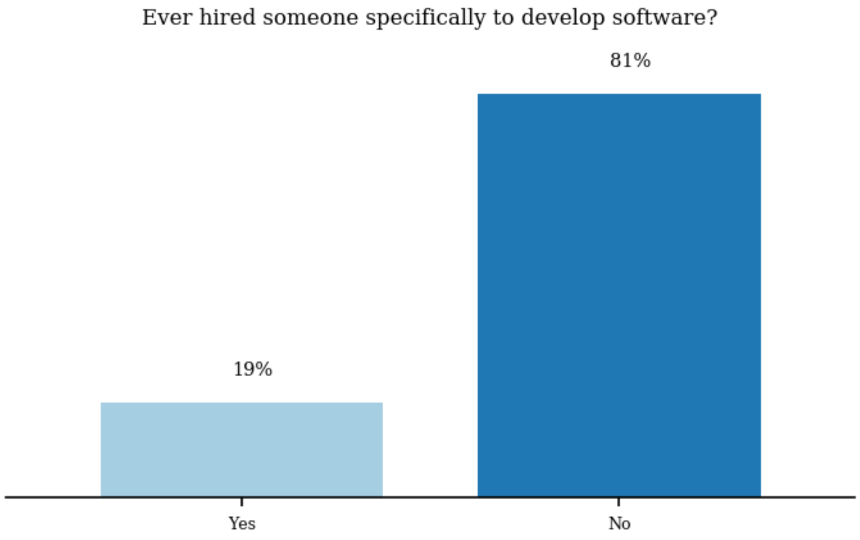


Version control is arguably the most basic technique, followed in complexity by unit testing and then continuous integration. Based on the results above, respondents to the survey appear to agree with this argument.

The above results indicate that, despite the reasonably high level of self-assessed software development expertise reported in the survey, a significant number of researchers do not possess basic software engineering expertise. Since it is software engineering that relates to reliable software, this could be a significant issue for hte university.

5 Access to software engineering expertise





Despite

6 References

1. Github repository for this project: https://github.com/Southampton-RSG/soton_software_survey_analysis_2019
2. National software survey: