Pass Rates in Introductory Programming and in other STEM Disciplines

Simon*
University of Newcastle
Australia
simon@newcastle.edu.au

Andrew Luxton-Reilly*
University of Auckland
New Zealand
andrew@cs.auckland.ac.nz

Vangel V Ajanovski
Ss Cyril and Methodius University
Republic of North Macedonia
vangel.ajanovski@finki.ukim.mk

Eric Fouh
University of Pennsylvania
USA
efouh@cis.upenn.edu

Christabel Gonsalvez
Monash University
Australia
chris.gonsalvez@monash.edu

Juho Leinonen
University of Helsinki
Finland
juho.leinonen@helsinki.fi

Jack Parkinson
University of Glasgow
UK
jack.parkinson@glasgow.ac.uk

Matthew Poole
University of Portsmouth
UK
matthew.poole@port.ac.uk

Neena Thota
University of Massachusetts Amherst
USA
nthota@cs.umass.edu

ABSTRACT

Vast numbers of publications in computing education begin with the premise that programming is hard to learn and hard to teach. Many papers note that failure rates in computing courses, and particularly in introductory programming courses, are higher than their institutions would like. Two distinct research projects in 2007 and 2014 concluded that average success rates in introductory programming courses world-wide were in the region of 67%, and a recent replication of the first project found an average pass rate of about 72%. The authors of those studies concluded that there was little evidence that failure rates in introductory programming were concerningly high.

However, there is no absolute scale by which pass or failure rates are measured, so whether a failure rate is concerningly high will depend on what that rate is compared against. As computing is typically considered to be a STEM subject, this paper considers how pass rates for introductory programming courses compare with those for other introductory STEM courses. A comparison of this sort could prove useful in demonstrating whether the pass rates are *comparatively* low, and if so, how widespread such findings are.

This paper is the report of an ITiCSE working group that gathered information on pass rates from several institutions to determine whether prior results can be confirmed, and conducted a detailed comparison of pass rates in introductory programming courses with pass rates in introductory courses in other STEM disciplines.

The group found that pass rates in introductory programming courses appear to average about 75%; that there is some evidence that they sit at the low end of the range of pass rates in introductory STEM courses; and that pass rates both in introductory programming and in other introductory STEM courses appear to have remained fairly stable over the past five years. All of these findings must be regarded with some caution, for reasons that are explained in the paper. Despite the lack of evidence that pass rates are substantially lower than in other STEM courses, there is still scope to improve the pass rates of introductory programming courses, and future research should continue to investigate ways of improving student learning in introductory programming courses.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

ITiCSE working group; CS1; introductory programming; pass rate; failure rate; STEM disciplines

ACM Reference Format:
Simon, Andrew Luxton-Reilly, Vangel V Ajanovski, Eric Fouh, Christabel Gonsalvez, Juho Leinonen, Jack Parkinson, Matthew Poole, and Neena Thota. 2019. Pass Rates in Introductory Programming and in other STEM Disciplines. In *2019 ITiCSE Working Group Reports (ITiCSE-WGR '19), July 15–17, 2019, Aberdeen, Scotland Uk*. ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3344429.3372502

1 INTRODUCTION

The computing education community generally accepts the view expressed by Robins et al. [34] that "Learning to program is hard … Programming courses are generally regarded as difficult, and often have the highest dropout rates." This sentiment is echoed throughout the literature with statements such as "It is well known in the Computer Science Education community that students have difficulty with programming courses and this can result in high drop-out and failure rates" [6]. Two authors of the current paper led

*Working group co-leader

ITiCSE-WGR '19, July 15–17, 2019, Aberdeen, Scotland Uk

ACM ISBN 978-1-4503-6895-7/19/07…$15.00
https://doi.org/10.1145/3344429.3372502

a 2018 ITiCSE working group [28] that conducted a broad-ranging review of the literature pertaining to introductory programming courses. The team conducting that review considered 1666 papers, and at times it seemed that almost every one of those papers had words in its introduction to the effect of 'programming is hard to learn' or 'introductory programming courses have high failure and dropout rates'. However, there are relatively few papers that have explored the empirical evidence for the claim that introductory programming courses have high failure rates.

We are aware of only three prior studies that have gathered data on pass or failure rates in introductory programming courses. Two of these [4, 5] each gathered a single year's data by surveying authors of computing education papers; the third [35] gathered data over a somewhat extended period by searching the literature for explicit mentions of pass rates in introductory programming courses. Our paper extends this previous work by reporting a worldwide selection of pass rates for introductory programming courses over the past five years and comparing these to pass rates in other introductory STEM courses.

Interestingly, although many papers claiming that introductory programming courses have high failure rates cite Bennedsen and Caspersen [4] as evidence, the authors of that paper conclude: "We did not find the failure-rate of CS1 to be alarmingly high". In a more recent paper, Bennedsen and Caspersen [5] replicate their earlier work, writing: "As we've noted, it continues to be the general view that there are high failure rates in introductory programming courses. However, to our knowledge, no worldwide statistics on failure rates, dropout rates, or pass rates for introductory programming courses at university level exist to back up this postulate." In the results of their replication, the authors found that the situation appears to have improved slightly over the last fifteen years, and conclude (once again), that failure rates in introductory programming are not alarmingly high. However, they also note several threats to the validity of their findings and suggest that a more thorough investigation of failure rates would be useful to the community.

It is our belief that discussions of whether pass rates are high or low cannot meaningfully take place in isolation, since concepts of high and low, easy and difficult, are not themselves absolute, but are relative to some assumed concept of what is normal. Further, establishing what is 'normal' as a point of comparison is context-dependent since each country and culture may have constructed its own view of what is typical in that environment. Bennedsen and Caspersen [5] reiterate a reviewer comment that "a failure rate of 28% at an elite university may be considered outrageously high and that 28% in a small university may be considered low". In other words, the context of the course has implications for how a given pass rate is interpreted. For example, some governments tie university funding to pass rates in courses [11, 12], which clearly provides institutional pressure to keep pass rates at a level that ensures funding, irrespective of real or notional standards of student achievement.

Despite such issues, it may be possible to determine if there is an empirical basis for the perception that introductory programming courses have low pass rates by comparing pass rates of programming courses with those of other courses *in the same local context*. For example, if an introductory programming course has a much lower pass rate than other courses taught in the same institution, it would be reasonable to assert that passing the introductory programming course in that institution is more difficult than passing other courses in that institution.

As we aim to compare introductory programming courses with other courses in the same cultural and institutional context, we need to consider what constitutes a fair comparison, and what conclusions we might be able to draw based on that comparison. While other comparisons might have been interesting, in this study we focus on the difference between introductory programming courses and other introductory STEM courses. Although there is some debate about whether computer science is even a 'real' science, there appears to be strong agreement that it is at least *related* to science, technology, engineering, and mathematics [14], if not composed of those subjects. We therefore aim to compare pass rates in introductory programming with those in courses in similar disciplinary areas — that is, STEM courses. The question of whether pass rates of introductory programming courses differ from those in arts, law, business, or other disciplinary areas is not addressed here.

This report analyses pass rates over the past five years, from a worldwide selection of universities, in introductory programming courses and in introductory courses in other STEM disciplines. This updates and triangulates the findings of the prior studies of pass rates in introductory computing courses, and additionally helps to establish whether pass rates in computing courses really are substantially lower than in other STEM courses, and whether this is a universal phenomenon.

The work addresses the following research questions:

- RQ1 What are the current pass rates in a selection of introductory programming courses around the world?
- RQ2 How do the pass rates in introductory programming courses compare with those in other introductory STEM courses?
- RQ3 What trends, if any, can be discerned in pass rates in introductory programming courses over the past five years?

2 RELATED WORK

In reviewing the literature, we found few studies that focus on pass or failure rates in introductory programming courses, or more broadly in introductory STEM courses. This may be because universities and educators are reluctant to publish papers indicating high failure rates or low retention rates [4, 20, 35].

In this section, we review work on pass/fail and dropout rates in introductory programming courses, as well as related work in pass/fail rates in other STEM disciplines. We also look at available national data on attrition rates in computing and other STEM disciplines. When discussing pass rates, failure rates, and related concepts in this section, we use the definitions of the authors whose work we are discussing. For example, while our work is expressed in terms of pass rates, if other authors report on failure rates or dropout rates we will use those terms when reporting their work.

2.1 Pass/Fail Rates in Introductory Programming Courses

In 2007, Bennedsen and Caspersen [4] conducted a worldwide survey of computing academics to gather the pass rates in their introductory programming courses. The survey was delivered to 479 authors of papers published at five different computing education conferences, and 63 usable responses were collected, of which 50 were from universities and the remainder from colleges. This provides a snapshot of pass rates at a single point in time. The study reported an average pass rate of 66% for university courses and 88% for college courses in introductory programming, giving an overall pass rate of 67% with large variations in the pass, fail, abort, and skip rates (the meanings of these terms are discussed in section 3.2). The authors observed that smaller classes appear to have a higher pass rate than larger classes, but no statistical analysis was conducted to validate the observation. The authors note several threats to validity, including the likely bias resulting from sampling authors who publish in the computing education community.

Comparing their findings with graduation data from UNESCO, Bennedsen and Caspersen [4] concluded that the 33% failure rate was not unusually high. However, it is problematic to compare the pass rate of introductory programming courses collected predominantly in the USA with graduation rates collected predominantly from western European countries. Further, the authors use enrolment numbers in 1999 compared with graduation numbers in 2004 to determine that 27% of students succeeded; however, given that students must complete several courses in sequence to complete their degrees, a failure rate of 33% each year would result in completion rates much lower than the reported 27%. This suggests that the failure rate of 33% found by Bennedsen and Caspersen [4] is higher than typical of computing courses throughout the degree programs recorded in the UNESCO data. It is unclear what conclusions we can draw from this comparison, or whether it informs the perception of difficulty of introductory programming courses.

Pass and failure rates in introductory programming were revisited in 2014 by Watson and Li [35], who established that pass rates in the literature on introductory programming courses showed an almost identical average of 68%. Their study analysed published accounts of pass rates from 51 institutions in 15 countries and concluded that pass rates varied by country, showed no improvement over time, and were independent of the programming language taught. They found a statistically significant difference between the pass rates reported for small classes (< 30 students) and those for larger classes (≥ 30 students), confirming the earlier observation that smaller introductory programming classes have higher pass rates than larger classes. Like the authors of the first study, they did not consider the pass rate to be alarmingly low. As there are so few studies that investigate the pass rates of introductory programming courses, any additional data is helpful. However, the papers published by academics involved in the computing education community form a biased sample that may not be representative of the community as a whole. We might speculate that authors may be using low pass rates as motivation for an intervention that they report, so the reported pass rates might be lower than those that are widespread in the community. Alternatively, those who publish papers in the computing education community might be the most effective teachers of introductory programming courses, so reported pass rates from that community might be higher than those typically found worldwide. We treat these findings with caution.

In a replication study in 2019, Bennedsen and Caspersen [5] received 170 responses to their survey, and found a statistically significant increase in the pass rate to 72.6% (which we will henceforth refer to as 73%). As with their previous results, there was wide variation in the pass, fail, abort, and skip rates. The mean course size had grown from 116 in 2006/07 to 196, and courses with fewer than 30 students dropped from 23% to 9% of their data set, reflecting the anecdotal reports of substantial growth in student enrolments in introductory programming courses. Unlike the earlier study, there was little difference between universities and colleges for abort, skip, and fail rates, but there was a tendency for colleges to have better pass rates than universities. As a basis for comparison, the authors tentatively established a US national average failure rate in college algebra of between 42% and 50%, and concluded that the average failure rate of 28% in introductory programming did not seem particularly high.

A number of studies [2, 7, 19, 25, 34, 35] attest to the belief that learning to program is difficult. The literature also includes a number of suggestions as to why pass rates are low in introductory programming courses. For example, Luxton-Reilly [27] suggests that "we make our introductory courses difficult by establishing unrealistic expectations for novice programming students"; Hoda and Andreae [21] suggest that the high level of attrition and failure are due not so much to incapable students as to inadequate teaching; and Parsons et al. [30] suggest that "the methods of assessment ... do not reflect the knowledge and skills that a real programmer needs to write real code." Some studies have also linked low performances in introductory programming courses to students' lack of self-regulated learning skills [6, 16].

Pass rates in courses are inextricably linked with failure rates and dropout rates. There is a substantial body of literature examining why students drop courses, and in particular why they drop introductory computing courses. Some of the reasons given are students' comfort level; expectations and perceptions of not getting enough help from course staff; difficulty in understanding course content; time management issues; and the lack of consequences of dropping out [8, 22–24]. However, the intent of this report is to attempt to measure, not to explain, and it is beyond the scope of the report to revisit the question of reasons for dropping out or to cover that literature in detail.

2.2 Pass/Fail Rates in other STEM Disciplines

There is substantially more work on failure rates in STEM courses than in introductory programming courses. Freeman et al. [18] reviewed eleven studies from 1992 to 2007 and noted that although there was not a comprehensive review of STEM pass rates, the studies they examined suggested that approximately a third of students failed in STEM gateway courses. They found reports of failure rates for introductory courses in biology, chemistry, computer science, engineering, mathematics, and physics, ranging from 85% (in biochemistry) to 25% (in biology). The reported average failure rate of 33% by Bennedsen and Caspersen [4] was similar to that reported in

Table 1: Average introductory STEM pass percentages comparing courses within specific institutions as reported in the literature

Reference	CS	Maths	Phys	Chem	Biol
Peterfreund et al. [31]	-	84	91	87	79
Liron and Steinhauer [26]	62	62	50	-	-
Chapman et al. [9]	-	75	83	80	-

introductory chemical engineering (32%) and introductory physics (33%), and better than in introductory calculus (42%).

Correlations have been found between failure rates and the instructional strategies used in STEM courses. A meta-analysis of 225 studies, reporting data on examination scores or failure rates in undergraduate STEM courses [17], found that average failure rates were 22% for courses using active learning compared with 34% for those using traditional lecturing. It is worth noting that the average failure rates for STEM courses delivered using traditional lecturing appear very similar to those of introductory programming courses.

There are very few studies that compare the percentages of students passing different courses within the same institution. Table 1 provides a summary of the rates reported in three such studies, which compared the impacts of different instructional methods on pass rates within the same institution. The number reported is the unweighted average of course pass rates: if the paper reports two different pass rates for the same subject/course, the average of those numbers is reported, regardless of the number of students in each course. In line with recommended practice, we have rounded the reported percentages to integer values. We note that the introductory programming course is equal second-lowest of ten different reported pass rates, which may be considered cause for concern. However, when we consider the individual institutions, the one introductory programming course is equal highest of the three courses reported from its institution, which is not particularly concerning. This illustrates the importance of comparing data obtained from the same institution rather than simply aggregating across all courses.

2.3 National Data on Attrition Rates in Computing and other STEM Disciplines

While our investigation compares pass rates in introductory programming courses with those in other STEM introductory courses, we thought that it might be instructive to look at publicly available attrition rates. Attrition generally refers to withdrawal from an entire program of study; obviously, students who do this while enrolled in an introductory programming course will also necessarily withdraw from that course.

In the US, an analysis of dropouts between 2003 and 2009 found that about half of STEM undergraduate students leave the field before completing a college degree [10]. The attrition rate was highest for majors in computer/information sciences and the report concluded that the 48% attrition rate for STEM undergraduates was similar to that for other fields. An analysis [15] of some of the factors that influence persistence rates in STEM fields revealed that academic preparation and entry test scores, along with students' performance in entry level classes, were important predictors of student persistence in STEM field majors.

Table 2: Percentage attrition rates for computing and other STEM disciplines

Country	CS	Maths	Phys	Biol	Eng
UK* [1]	10	5	4	7	7
USA+ [10]	31	12	18	15	20
USA++ [10]	28	26	28	30	21

* Started higher/post secondary education in 2016-2017 and left after one year without a degree.
+ Started higher/post secondary education in 2003-2004 and left within six years without a degree (2003-2009).
++ Started higher/post secondary education in 2003-2004 and switched to another degree within six years (2003-2009).

National reports of attrition rates in computing and other fields are available for the UK [1] and the USA [10], and the rates are summarised in table 2. Again, in line with recommended practice, we have rounded the reported percentages to integer values. In both countries, computing has the highest rate of students exiting higher education. In the US, computing is second only to physics in terms of students who switched to a different major. The UK data are reported after the first year, meaning that we should expect those students to have taken an introductory programming course in the year when they decided to drop their program of study. We cannot make the same assumption from the US data as the time frame spans several years. Similar data from Ireland, grouped differently [32], shows a 45% attrition rate for computing compared with 33% for engineering and 24% for science and mathematics.

We are left to ponder what impact, if any, introductory programming courses might have on these high attrition rates.

3 METHOD

The plan of our working group was to use three approaches to discovering and collecting data.

- Each member of the working group sought to gather data on pass rates in introductory courses from their own institution, both in programming and in other STEM disciplines, for the past five years. Analysis of this data would give a picture of pass rates at a small number of institutions, and would help to establish whether any trends can be discerned over the past five years.
- The working group also conducted a survey in which it asked respondents to provide the same data for just the most recent year at their own institutions. Analysis of that data would strengthen the findings from the institutions of the working group members, giving a clear snapshot of the current relationship between pass rates in introductory programming courses and those in other introductory STEM courses. The survey was circulated to the SIGCSE-members email list, which at the time the invitation was posted had 1183 subscribers.

- Finally, the working group sought publicly accessible data on pass rates reported at a state or national level, to complement the data from members' institutions and from the survey.

In the event, it was surprisingly difficult to gather the data that we sought for this project. Many universities are sensitive about their pass rates and reluctant to release that data, even in an aggregated and fully anonymous form.

Most members of the working group were required to go through detailed and time-consuming processes in order to be granted access to the data for their institutions. In many cases, these processes took several months, and included provisions that the results must be presented in such a way that it would not be possible for readers to trace them back to individual institutions. One member of the group was simply denied access to the relevant data. We are therefore limited in the contextual information we can provide about individual institutions that contributed data to the study.

Following this experience, we were not sure that anybody would be in a position to respond to the survey, and were not surprised to receive only ten responses.

3.1 Unified Data Set

During an initial meeting, members of the working group discussed how data obtained from different institutions and with different levels of detail could be combined using a common structure. After a thorough discussion about terminology (see section 3.2), a common baseline for a joint data set was defined, based on types of data that were available consistently across all institutions. All the gathered data was converted to the same form, in order to facilitate analysis both within and between institutions. The final joint data structure comprises the following attributes:

- *University code* — to be used for reporting results of analyses across institutions without revealing the institutions' names
- *Course code* — used internally to identify distinct courses within institutions
- *Course name* — used internally to gather information about the course and properly categorise it
- *Year the course was offered* — used for longitudinal analysis
- *Course offering* — number used to distinguish among several offerings of the same course during the same year, but at different periods/terms/semesters or locations
- *Enrolments* — number of students enrolled in the course at the time the pass rate is calculated
- *Passes* — number of students who passed the course
- *Category* — this attribute was added to the data set and was decided by the working group members, based on the agreed categorisation scheme explained in section 3.3.

All of the data in the joint data set was obtained directly from individual institutions or from national databases. Both data sources are combined in the joint data set and are used in the analysis reported in section 4. Data from the survey responses was not incorporated in the joint data set, but is reported separately.

3.2 Terminology

As we began to combine data from different institutions, we quickly realised that pertinent words and phrases in common use can actually mean very different things. For example, a 'pass' might come in different types, such as a 'restricted pass' that can be credited towards a degree program but does not fulfil the prerequisite requirements needed to continue to more advanced topics in that subject area. That being the case, it is not feasible to describe our method without clearly explaining our terminology. In this section, we describe how we operationalise the various concepts for the purpose of this study, and discuss how our terms differ from those in the related work.

3.2.1 Course. In this paper, we use the term 'course' to mean a single cohesive set of content, typically taught and assessed within a single semester. We believe that 'course' is understood fairly consistently, although the terms 'unit', 'subject', and 'module' are known to be used as equivalents in some institutions.

3.2.2 Introductory. Although in this paper we use the phrase 'introductory course', it is not always obvious when a course should be treated as introductory. For the purposes of this study, we consider a course to be introductory if it is a viable entry point to the subject matter and can be taken without any formal prerequisites from within the university. Courses treated as introductory are typically available for students to take in their first year of study.

Some introductory courses may require students to have taken particular courses at school, while others do not require any previous disciplinary knowledge. Some institutions have entry criteria (whether imposed at the course, discipline, or institutional level) that require a particular level of academic background, either in cognate subject areas or in general academic performance (such as a GPA), while other institutions have no such criteria for entry.

Further, some subject areas may offer several different introductory courses, acting as multiple entry points and allowing students to choose an introductory course commensurate with their experience. For example, some universities may offer a slow-paced introductory programming course for students who have not previously encountered programming, and a faster-paced introduction to programming for students who are self-taught or who have taken computing at school. We consider all such courses to be introductory, despite potential differences in their student cohorts, their pace, and their content.

3.2.3 Programming. A wide variety of courses teach programming at introductory level. There are two main factors that are relevant in knowing whether a course should be treated as an 'introductory programming' course: the target audience and the focus of the content. We make a clear distinction between courses that teach programming primarily for computing students, and those that teach programming primarily for students of other disciplines. We also distinguish between courses whose focus is general purpose programming (e.g., 'Introduction to programming in Python') and courses designed to teach programming with a more specific goal in mind (e.g., 'Programming web applications').

When examining data from multiple institutions it is not always possible to know whether a given course is available only to computing students, to other students, or both, and we are unable to determine which students actually take the course, so we cannot distinguish courses in this way. Instead we focus on the course content and course description.

Some courses are taught within computing discipline areas (that is, offered by disciplines variously called computer science, software engineering, information technology, informatics, etc.) and focus on teaching general programming skills in courses such as 'Introduction to programming' and 'Introduction to problem solving with computers'. Other courses clearly teach programming to a non-computing audience, with course titles such as 'Engineering computing' and 'Computing for physics'. Further, some courses involve programming, but in a more specialised context, such as 'Programming for creativity', 'Introduction to programming with databases' and 'Business application programming'.

For the purposes of this paper, we will use the phrase 'introductory programming' to refer to courses that appear to focus on general-purpose programming for computing students, even though students from other disciplines might also enrol in these courses. We distinguish these courses, which are often referred to as CS1 courses, from introductory courses that appear to teach programming in other contexts or for other student groups, which we hereafter refer to as 'cross-disciplinary programming' courses.

Courses that teach computing concepts other than programming (such as 'Computer architecture' or 'Database design') are categorised as computing courses, but not as programming courses.

While this study, like those that precede it, makes an assessment of pass rates in introductory programming courses, we emphasise that there is no such thing as 'the introductory programming course'; there are probably as many forms of introductory programming course as there are offerings. All of the courses analysed in this study have been identified as introductory programming courses, but we do not know their content, their level of difficulty, the number of hours of teaching that they involve, what programming languages they use, what teaching approaches they use, how many students they have, the number, nature, and weighting of their assessment items, or the many other factors that undoubtedly influence their pass rates. Therefore, when we discuss introductory programming courses, we urge readers to remain aware that we are in fact discussing a great variety of courses that happen to fit the same generic term.

3.2.4 Enrolment. The interpretation of student enrolment is critical to our analysis since we are interested in the number of students that pass a course as a percentage of the number of students enrolled in that course.

Although the concept of enrolment may initially appear to be straightforward, there are several factors related to institutional contexts that affect how enrolment is calculated, and consequently the basis on which pass rates are calculated. For example, we need to consider how to address the situation where a student initially decides to take a course but then changes their mind.

Typically, if a student changes their mind before the course starts then we would not want to count them as having enrolled but failed to pass the course. It is less clear how to record the situation when a student remains in the course for a period of time and then decides not to continue. If a student changes their mind early in the course (immediately after the first class, for example), then it does not seem appropriate to record them as a 'drop' or a 'failure to complete'. However, if a student changes their mind a day before the end of the course, we would want to record that as a failed attempt at the course. Additionally, a student may choose to enrol in a course after the start of the course. In this case, we would want to ensure that the student contributes not only to the count of outcomes, but also to the count of enrolments. Deciding when to record the student as making a genuine attempt at a course is somewhat problematic, particularly as not all institutions record the same data, and those that do record the same data do not always record it in the same way.

Many institutions provide a 'grace' period after the start of the course in which students may change their enrolment. For some institutions involved in the study, this period is two weeks, while other institutions allow up to four weeks for a student to change courses. Many institutions do not record a student as enrolled in a course if they have removed themselves from the course within the grace period. At the end of the grace period there is a 'census' date after which a student may not change their enrolment. There are also institutions that consider students to be enrolled only if they are still active at the end of the teaching period; for example, when the final exam is held.

For institutions where we have sufficient data, we define the enrolment of a course to be the number of students that are in that course at the census date for their institution (which is typically 2-4 weeks after the start of classes). If an institution has no grace period in which students can change their course selection, we consider the census date to be the start of the course (or the time of enrolment if no subsequent changes are permitted). However, for some institutions we have no way of knowing at what point the number of enrolled students is assessed, and we must simply accept the pass rate as provided without knowing how it was calculated.

3.2.5 Withdraw. Several institutions allow students to formally 'withdraw' from a course after a census date has passed. Typically this will not result in a refund of any fees that students might have paid, but it is recorded on their academic record as a 'withdraw' rather than a fail. This is equivalent to the term 'abort' used by Bennedsen and Caspersen [5], but we note that is is variously referred to as 'drop/drop-out' or 'non-completion'; at the same time, we acknowledge that 'drop' is often used to refer to withdrawal from an entire program of study, or to a change of major, rather than withdrawal from a single course.

In some institutions, for example in England and Wales, to withdraw from a course one would have to withdraw from the whole degree program, and the reasons that a student might choose to drop their program are not necessarily based on their performance in or their response to a single course such as introductory programming.

A withdrawal might or might not affect the GPA of a student (depending on the country and institution), but many students prefer to have a withdraw rather than a fail recorded on their academic transcript. Additionally, some institutions prefer students to withdraw rather than fail due to the reporting requirements of the institution; for example, an institution may report a higher pass rate if it is based on the number of students who attempt the final exam. From the data available, we cannot know why a student has withdrawn, but we do know that a student who has withdrawn from a course has not passed that course.

3.2.6 Skip. Bennedsen and Caspersen [5] use the term 'skip' to refer to students who were permitted to attempt the final exam but did not do so (for courses that have final exams). In other institutions, this may be reported as 'did not sit (DNS)' or 'absent fail'. In some institutions, a range of other categories of non-completion are recorded, including 'did not complete' and 'not satisfied requirements (NSR)' (for courses that have completion requirements that are not met, such as undertaking a given number of hours of practical work). In other institutions, students in this position are simply given fail grades.

This description assumes that where students have missed the exam for unavoidable external reasons such as illness, their cases have been resolved — for example by the offer of a subsequent exam — and they have subsequently been classified into one of the other categories such as pass or fail.

We did not capture 'skip' rates and cannot report them separately; instead, students who might be classified as 'skip' in related work are included here as students who *do not pass* a course. In our analysis we focus on pass rates and do not distinguish between the different ways that a student might not pass.

3.2.7 Fail. A wide variety of outcomes are treated as fail grades. In some places, an 'F' is an explicit fail, while others assign different levels of failure, such as distinguishing between 'D−', 'D' and 'D+', which are all considered failing grades in some places.

3.2.8 Pass. Passing grades are typically awarded with variations in achievement (e.g.,letter grades, numeric grades, or named grades such as high distinction and distinction), but can sometimes include non-graded courses for which students are awarded only a pass or a fail. In some institutions, a letter grade of D is considered to be a passing grade, but one that does not qualify the student to continue to subsequent courses in the same subject area.

In this paper, we treat any grade that can be credited towards completion of a degree program as a passing grade, even though in some cases the grade will be insufficient for the student to continue to more advanced material in that subject area.

Because the notion of failing a course is fraught with inconsistency (for example, is withdrawing from a course the same as failing a course?), it is difficult to report failure rates consistently across multiple institutions. On the other hand, we believe that the notion of a pass is relatively clear: a grade that permits the course to count towards a student's degree completion requirements. In this paper, we focus on the number of students who are enrolled in a course at the census date for that institution if it has one, and the number of students who pass the course; and we report pass rates rather than fail (or abort, or skip, or drop) rates.

3.2.9 STEM. In choosing to compare pass rates in introductory programming courses with those in other introductory STEM courses, we acknowledge that there is no universally agreed definition of STEM. The acronym represents the phrase 'science, technology, engineering, and mathematics'; but all four of those terms are themselves open to multiple interpretations. Indeed, while it appears to be generally accepted that computing falls into the technology category, there are people who question whether computing is part of STEM, and a web search readily finds universities using the wording 'STEM and computer science'. Further, it is not clear whether medicine and related disciplines are considered part of science, and thus of STEM.

Members of this working group used their own understanding of STEM courses when collecting data from their institutions and from public sources. This is not considered a threat to validity, as it was clearly never going to be possible to gather data for every introductory STEM course at every institution. The outcome of this study might therefore be described more precisely as a comparison of pass rates in a selection of introductory programming courses with pass rates in a selection of STEM and related courses at the same institutions.

Table 3: Course category keywords used in this report

Keyword	Description
prog-comp	Introductory programming courses in computing (introductory programming)
prog-other	Introductory programming courses for students in other disciplines (cross-disciplinary programming)
comp	Non-programming courses in computing
maths	Mathematics
stats	Statistics
phys	Physics
chem	Chemistry
biol	Biology
earth	Earth sciences
psych	Psychology
health	Health & medical sciences
eng-ee	Electrical & electronic engineering
eng-other	Other engineering disciplines
other	Other (including computing skills for students in other disciplines)

3.3 Data Categorisation

While our expressed intention was to compare pass rates in introductory programming courses with those in other introductory STEM courses, we saw potential benefit in breaking down those other STEM courses into recognisable subject areas. Before collating all of the data, the group chose the categories into which courses would be assigned and discussed the classification of some oddities and cross-disciplinary courses. Once categories had been established, each member assessed a subset of the data, assigning each course to a category based on a combination of the course's delivering department (where this was captured), course title, course code (which often includes hints such as BIOL or CHEM), and in some cases the public course descriptions supplied by the institutions. Table 3 shows the categories into which the courses were classified.

3.3.1 Classification Reliability. We conducted an inter-rater reliability test on our classification of courses into subject groupings, using the Fleiss-Davis kappa [13], a chance-corrected measure of reliability for the situation in which each classifier independently classifies each item into one of a fixed set of categories. As test data

we used a set of 41 courses, from a university that was not part of our data set, that were possibly introductory STEM courses. None of the team were familiar with these courses.

Each member of the team independently classified each of the courses into one of 15 categories: the 14 listed in table 3, and *not-intro*, a category for courses that would not be included in our data set, either because they are not considered STEM or because they are not genuinely introductory courses.

The Fleiss-Davies kappa for our classification was 0.66. Readers unfamiliar with chance-corrected inter-rater reliability will consider this low, but it is generally accepted that values between 0.4 and 0.75 are "fair to good" [3], and our measure is approaching the high end of this range.

On examining the classifications it appeared that the greatest disagreement lay with the application of the *not-intro* classification. There was some disagreement as to whether certain subjects (notably anthropology, archaeology, and science communication) should be considered STEM; and more obvious disagreement as to whether certain courses are genuinely introductory. For example, our test data includes the courses *Comp1110 Structured programming* and *Comp1140 Structured programming (advanced)*. Assuming that Comp1110 is an introductory course, Comp1140 could be either a follow-on course, and thus clearly not introductory; or a parallel introductory course for students deemed on entry to be more advanced[1]. It would often be possible to resolve such ambiguities by examining the website of the university in question; however, with nearly 1500 distinct courses to consider (see section 4), we did not always do this, instead relying on existing knowledge of the courses where possible, and otherwise on the course codes and names.

To test the hypothesis that disagreements concerning *not-intro* were substantially affecting the kappa value, we replaced every classification other than *not-intro* with a generic *intro* classification and measured the reliability again; the kappa value was now 0.29, indicating no real agreement beyond chance and thus supporting the hypothesis.

In an attempt to eliminate this particular source of disagreement, we then applied a majority rule to courses for which there was uncertainty concerning *not-intro*. If the majority had classified a course as *not-intro*, the other classifications were adjusted to *not-intro*. Conversely, if a minority had classified a course as *not-intro*, those classifications were replaced with the discipline that the majority had chosen. A third test, on the data adjusted as described, gave a kappa value of 0.93, high in the excellent agreement band.

These measures establish that we have excellent agreement on the classification of courses into disciplines, but poor agreement on which courses should be included in our analysis, generally because of uncertainty about whether a course is genuinely introductory. If the pass rate in a non-introductory course is likely to be higher than in an introductory course, inclusion of non-introductory courses might inflate the average pass rate, as might exclusion of introductory courses. The classification of our test courses thus suggests a possible threat to the validity of our findings.

[1]Subsequent investigation reveals these two courses to be the same course, with students attending the same classes, but with some more challenging assessment items in *Comp1140*.

3.4 Data Aggregation

To analyse the data, we need to aggregate the pass rates from the original data set. There are two general ways to calculate an aggregate over a group of percentages. The terms *unweighted* and *weighted* are used in this context to describe how the percentages are aggregated.

Unweighted aggregate pass rates are calculated using the percentage of students passing each course as a single data point (that is, dividing the number of students passing a course by the number of students enrolled in the course). This means that a course with 500 enrolments and an 80% pass rate will have exactly the same effect on the aggregate data as a course with 50 enrolments and an 80% pass rate. We believe that this approach to aggregation may better reflect *instructor* perceptions, since instructors are focused on course delivery.

A potential limitation of this approach may arise if courses are recorded in the data for administrative purposes in a way that differs from the actual delivery of the course. This may occur, for example, if a single course is divided into multiple streams and each stream is recorded as a separate course instance; or if a virtual course offering is created for students with timetable clashes even when they attend the same course as other students. See also the footnote in the preceding column. Our data strongly suggest that these practices are followed at three of the institutions from which we have large numbers of course offerings for analysis.

Weighted aggregate pass rates are calculated using the raw numbers of students enrolled and passed in all courses of a given category (that is, the sum of all students who passed courses in a given category divided by the sum of all students enrolled in courses belonging to that category). We believe that this approach better represents the overall *student* perspective, since it captures the total percentage of students who have succeeded in a course in the subject area. This has the effect of giving more weight to courses with higher enrolments, but it solves potential issues that may arise with many small courses being artificially created for administrative purposes.

We have chosen to present both analyses, but with a preference for the weighted representation when examining overall trends since it mitigates the potential impact of organisational and administrative differences between institutions.

4 RESULTS

Our principal data set comprises data from 17 universities from eight countries: Australia, Finland, New Zealand, North Macedonia, Norway, two countries from the UK, and the USA. The set covers 5646 offerings of 1406 distinct courses over the five-year span from 2014 to 2018. Most offerings cover the full five-year span, but there are exceptions — for example, for courses that were launched or discontinued during the span. There are 232 course offerings categorised as introductory programming, and 131 offerings categorised as cross-disciplinary programming. A total of 990,569 enrolments were recorded across all courses and institutions, including 92,607 enrolments in introductory programming courses.

We cannot give details of the data from each institution, as that might encourage attempts to identify the institutions. Instead we provide descriptive information about the nature and size of the

Table 4: Minimum, maximum, and median number of courses, course offerings over five years, and total student enrolments over five years, for the 17 institutions

	min	max	median
Intro programming			
Courses	2	12	3
Course offerings	10	37	20
Enrolments	1,802	11,370	4,414
Other STEM			
Courses	1	673	38
Course offerings	10	1,915	204
Enrolments	1,966	138,803	40,029

Table 5: Number of unique courses, distinct offerings of those courses, and total enrolments in each category

Category	Courses	Offerings	Enrolled
prog-comp	34	232	61,580
prog-other	31	131	31,027
comp	81	398	80,312
maths	193	921	156,824
stats	48	278	74,087
phys	128	506	78,502
chem	113	472	92,309
biol	176	628	133,493
earth	149	517	44,843
psych	26	114	34,802
health	127	385	59,769
eng-ee	45	160	23,444
eng-other	159	622	94,604
other	96	282	24,973
Total	1,406	5,646	990,569

data set. Table 4 shows that data from some institutions is very broad, comprising hundreds of courses from various STEM fields, while other institutions provided much more limited data — in one instance, courses only in programming and mathematics.

Table 5 offers a more detailed breakdown of the data, showing the numbers of courses, offerings, and enrolments in each subject category.

4.1 Longitudinal Results

We calculated the overall pass rates for programming courses and other STEM courses by year for the period 2014–2018. The trends are shown in figure 1 (unweighted average pass rate per group per year) and figure 2 (weighted average pass rate per group per year). Based on the data available to us, the average pass rates for both programming courses and other STEM courses have been quite stable, hovering around 75%.

Based on the unweighted aggregate data, introductory programming courses appear to have very similar pass rates to cross-disciplinary programming courses and other STEM courses. When

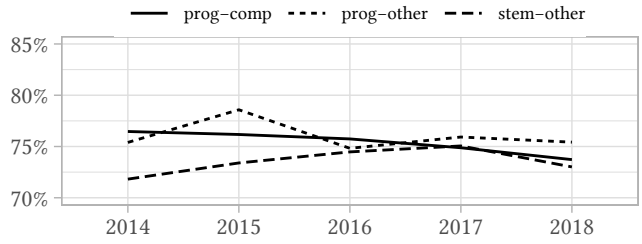

Figure 1: Unweighted average pass-rate trends for introductory programming courses and other STEM fields in the period 2014–2018

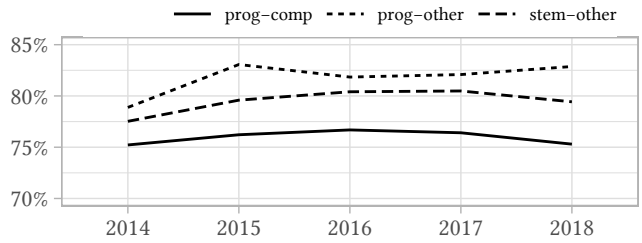

Figure 2: Weighted average pass-rate trends for introductory programming courses and other STEM fields in the period 2014–2018

considering the weighted averages, the pass rate for introductory programming courses is about the same as the unweighted rate, but the rates for cross-disciplinary programming and other STEM courses are substantially higher. This suggests that pass rates in cross-disciplinary programming and other STEM courses are higher in courses with higher enrolments. However, this is based only on the data that we have to hand, and we are not in a position to speculate on the reasons.

Both the weighted and unweighted measures of introductory programming are close to the 73% average pass rate identified by Bennedsen and Caspersen in their 2019 study [5].

4.2 Cross-Category Results

Figure 3 shows the unweighted average pass rate of each of the disciplines that we have identified in our analysis, and figure 4 zooms in to show the difference (as an absolute percentage) between each other discipline and introductory programming. The figures show that introductory programming is fairly centrally placed, with four disciplines having higher unweighted average pass rates and nine having lower rates. The colour scheme used in these figures is consistent in all subsequent figures: light grey for the reference category *prog-comp*, orange for the categories having pass rates above the reference, and purple for the categories having pass rates below the reference.

Interestingly, two of the four categories with higher pass rates than introductory programming are the other computing categories, the remaining two being psychology and health.

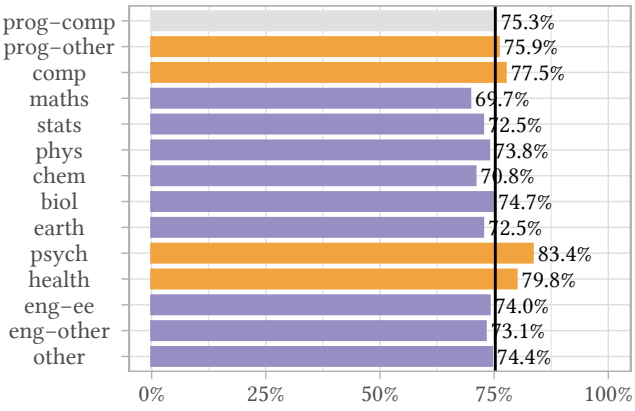

Figure 3: Unweighted pass rates by category with all categories represented (the comparison line indicates the pass rate in *prog-comp*)

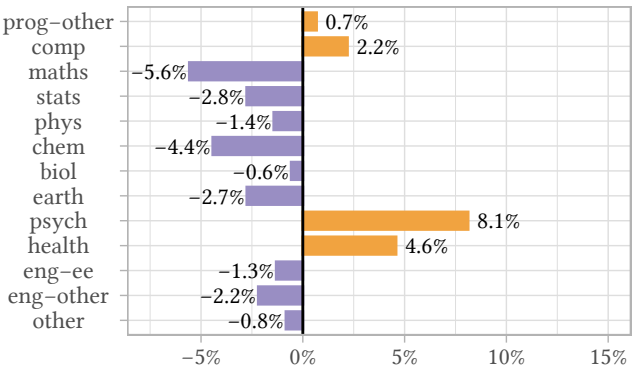

Figure 4: Differences between unweighted pass rates in other STEM courses and introductory programming courses (*other course – prog-comp*)

A different picture is painted when we consider the weighted average pass rates per category. Figures 5 and 6 show that all STEM courses except mathematics and statistics have higher weighted average pass rates than introductory programming courses. In both the unweighted and the weighted data, mathematics and statistics courses have lower pass rates than introductory programming courses.

4.3 Institutional Results

Although aggregating pass rates across all institutions provides results that can be compared with related work, we hypothesised that there would be substantial differences between institutions. Given the anticipated impact of local context, we believe that it is important to compare introductory programming courses with other STEM courses at the *same* institution. Figure 9 shows a box plot of the unweighted pass rates for STEM courses at each institution, and the unweighted pass rate for introductory programming courses (represented as a circle); and figure 10 breaks down the pass rates

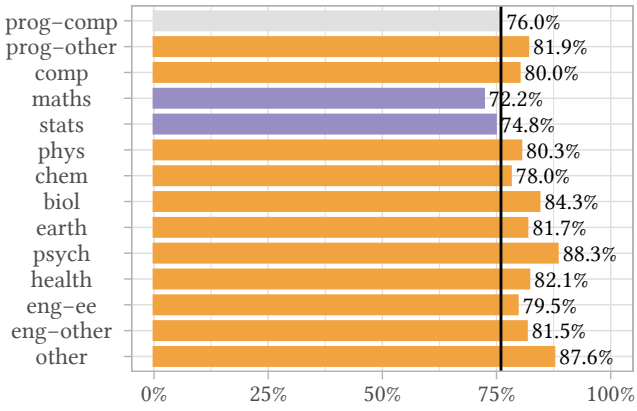

Figure 5: Weighted pass rates by category with all categories represented (the comparison line indicates the pass rate in *prog-comp*)

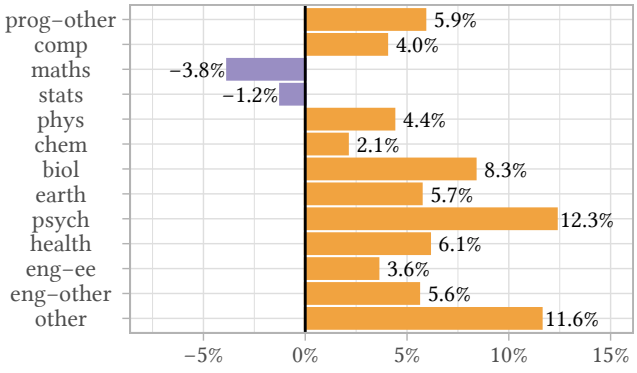

Figure 6: Differences between weighted pass rates in other STEM courses and introductory programming courses in computing (*other course – prog-comp*)

from each university into disciplines. It is clear that disciplinary pass rates vary substantially between institutions.

A close examination of figure 10 shows that psychology is the only category whose pass rate is higher at every institution than introductory programming. For every other category there is at least one institution where that category has a lower pass rate than introductory programming. However, it is clear that the majority of subject categories at universities (138 of 166 data points) have higher pass rates than introductory programming.

4.4 Course Sizes and Pass Rates

Since some of the prior work found differences in pass rates in relation to class sizes, we ran Spearman's correlation to investigate the relation between course size (as measured by enrolments) and pass rates. In the case of introductory programming courses, we found no significant correlation between course size and pass rate, as can be seen in figure 7.

Similar investigation for the other course categories found a significant slightly positive monotonic correlation in the general

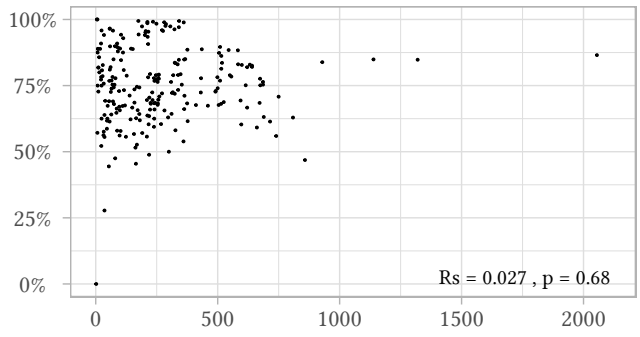

Figure 7: Relationship between course size (number of enrolments) and pass rate of introductory programming courses, measured using Spearman's rank correlation

case, but interestingly, the larger the course, the higher the pass rate, as shown in figure 8.

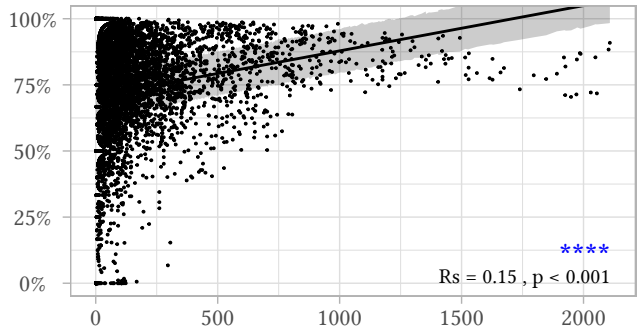

Figure 8: Relationship between course size (number of enrolments) and pass rate of introductory courses in STEM excluding introductory programming, measured using Spearman's rank correlation

Further analysis is provided in figures 11 and 12, which show the relationship between course size and pass rate for each institution, for introductory programming courses and for other STEM courses respectively. The correlation between course size and pass rate for introductory programming courses is significant at six of the institutions, of which three show positive correlations and three negative. For other STEM subject areas, the correlation between course size and pass rate is significant at nine of the 17 institutions, with only two having positive correlations and seven negative. This may suggest that in many places, fewer students pass as STEM courses grow larger, but this is not typically the case for introductory programming courses. This may also explain the difference observed between the weighted and unweighted analyses.

4.5 Survey Data

As indicated earlier, the responses to the survey are so few and so varied in detail that they cannot be usefully analysed. Instead we provide here a brief description of the responses received. Class sizes are rounded in the descriptions to help preserve the anonymity of the respondents' institutions.

One of the ten responses gives the pass rate for a single course, but with insufficient description for us to determine in which discipline (table 3) the course lies.

At the other extreme, a response from the UK provides pass rates for introductory programming (250 students, 93% passed), physics (150, 81%), mathematics (150, 92%), and biology (250, 96%). Withdrawal is not really a meaningful concept in these courses, as it would entail withdrawal from the entire program of study, so all students who did not pass the courses failed them.

Two responses provide pass rates in programming and one other STEM discipline. A response from Australia shows withdrawal rates of about 20%, and gives pass rates in terms both of original enrolment and of students who did not withdraw. The courses are programming (350 students, 52%/65% passed) and chemistry (650, 63%/82%). A response from the USA shows withdrawal rates a little under 10% and gives pass rates for programming (450 students, 79%) and mathematics (600, 81%).

One response from Germany covers one programming course and one *comp-other* course, in databases. The withdrawal rates here are far higher than in the other responses, 60% in programming and 40% in databases, and the pass rates are very low: 22% of 1500 students in programming and 35% of 700 students in databases.

The remaining responses, all from the USA, give figures only for one or more introductory programming courses, and so provide no basis for comparison with other introductory STEM courses. The nine pass rates from these five responses are 65%, 71%, 75%, 83%, 85%, 90%, 91%, 92%, and 92%. Class sizes range from 10 to 350, and withdrawal rates are all less than 10%.

While these responses are too few to give the snapshot of current pass rates that we hoped to elicit, they nevertheless manage to encapsulate the substantial problem of comparability of pass rates. How is it meaningful to compare pass rates from around the world – or to present their averages, as we have done in this report – when the courses themselves are offered, presented, and assessed on such different bases? How are we to compare the introductory programming pass rates in the UK, where students take most or all of their courses as mandatory components of their chosen degree; in Australia, where withdrawal is a standard technique to avert the award of a fail grade in a course; and in Germany, where courses can be selected almost on a trial basis and dropped without penalty?

5 DISCUSSION

In this section we reflect on the questions that we sought to answer:

RQ1 What are the current pass rates in a selection of introductory programming courses around the world?
RQ2 How do the pass rates in introductory programming courses compare with those in other introductory STEM courses?
RQ3 What trends, if any, can be discerned in pass rates in introductory programming courses over the past five years?

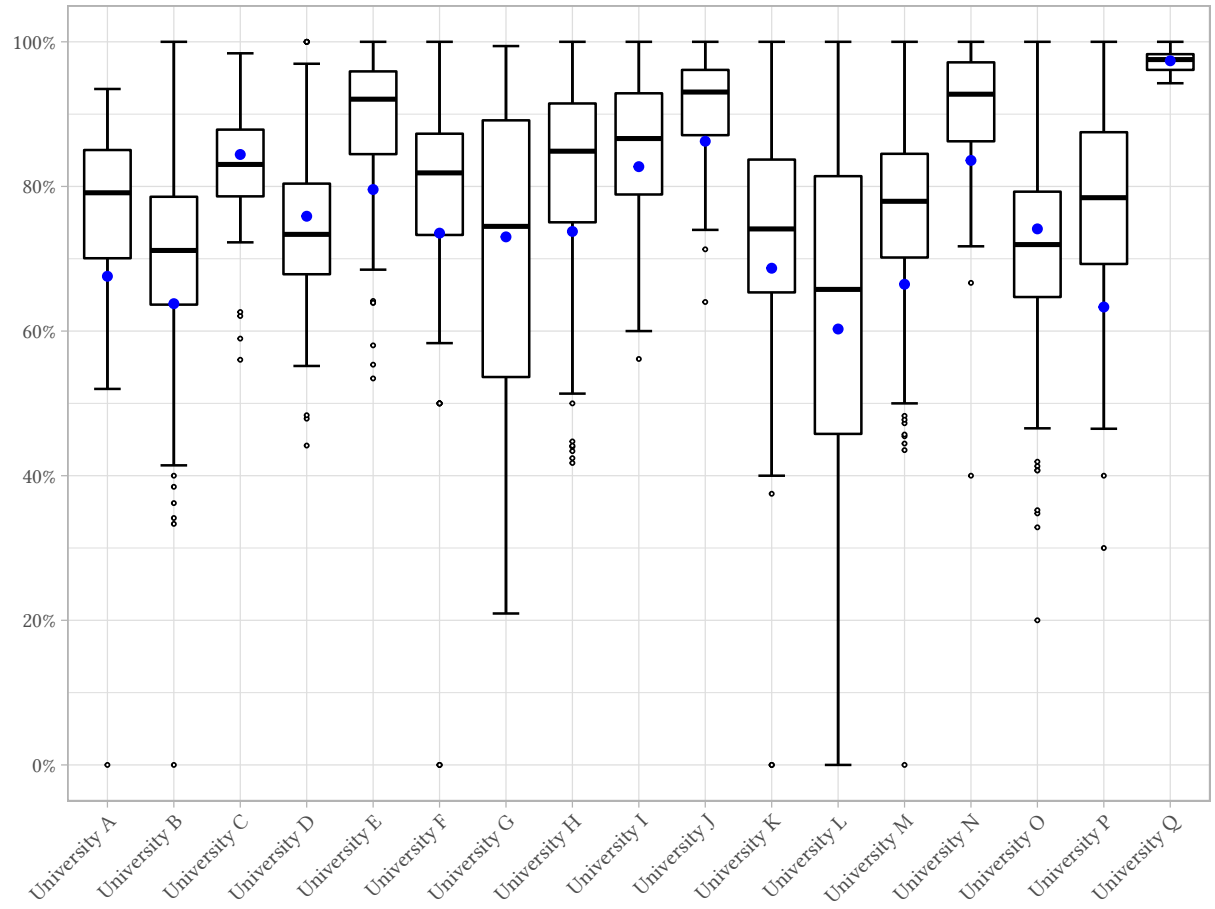

Figure 9: Unweighted pass rates by institution for introductory programming (*prog-comp*: large circles) and other STEM courses

5.1 Current Pass Rates in a Selection of Introductory Programming Courses around the World

In our analysis, introductory programming courses have an average pass rate of around 75% (75.3% for the unweighted average and 76.0% for the weighted average). This average is only slightly higher than that reported in the most recent analysis of pass rates by Bennedsen and Caspersen [5], despite the data sources and data collection process being very different. Most of the data collected in the Bennedsen and Caspersen study [5] derived from US institutions, while most of the data reported in this paper is obtained from institutions in other countries. Triangulating the pass rates through data collected from very different sources, and finding very little difference in the average pass rates, provides some confidence in these findings.

Figures 13 and 14 show that pass rates in cross-disciplinary programming courses (prog-other) are somewhat higher than in introductory programming courses delivered to computing students. This is an interesting finding, replicated across both weighted and unweighted comparisons. We can speculate as to why this might be the case: typically, cross-disciplinary programming courses are considerably more practically oriented and goal-focused, with the intention of training participants in the use of a language or framework to achieve specific goals in their particular context. Compare this with the typically broad and theoretical basis of introductory programming found in computing, focused as much on instilling concepts as on practical skills, and one can imagine that these courses might be more difficult to grasp. It is also possible that the expected standard of attainment is lower in cross-disciplinary programming courses than in courses designed to teach programming to students who will need to further develop their knowledge in subsequent courses. Another speculation that can be made is that almost every computing program will have its introductory programming course in the first year, whereas programming courses in other STEM areas may be taken by more experienced students in later years of study, possibly boosting the pass rates in those courses.

However, while these suggestions are interesting, they are purely speculative. We can tell little about these programming courses for other disciplines except that they appear to have higher pass rates than introductory courses for computing students. This might be an interesting topic for future investigation.

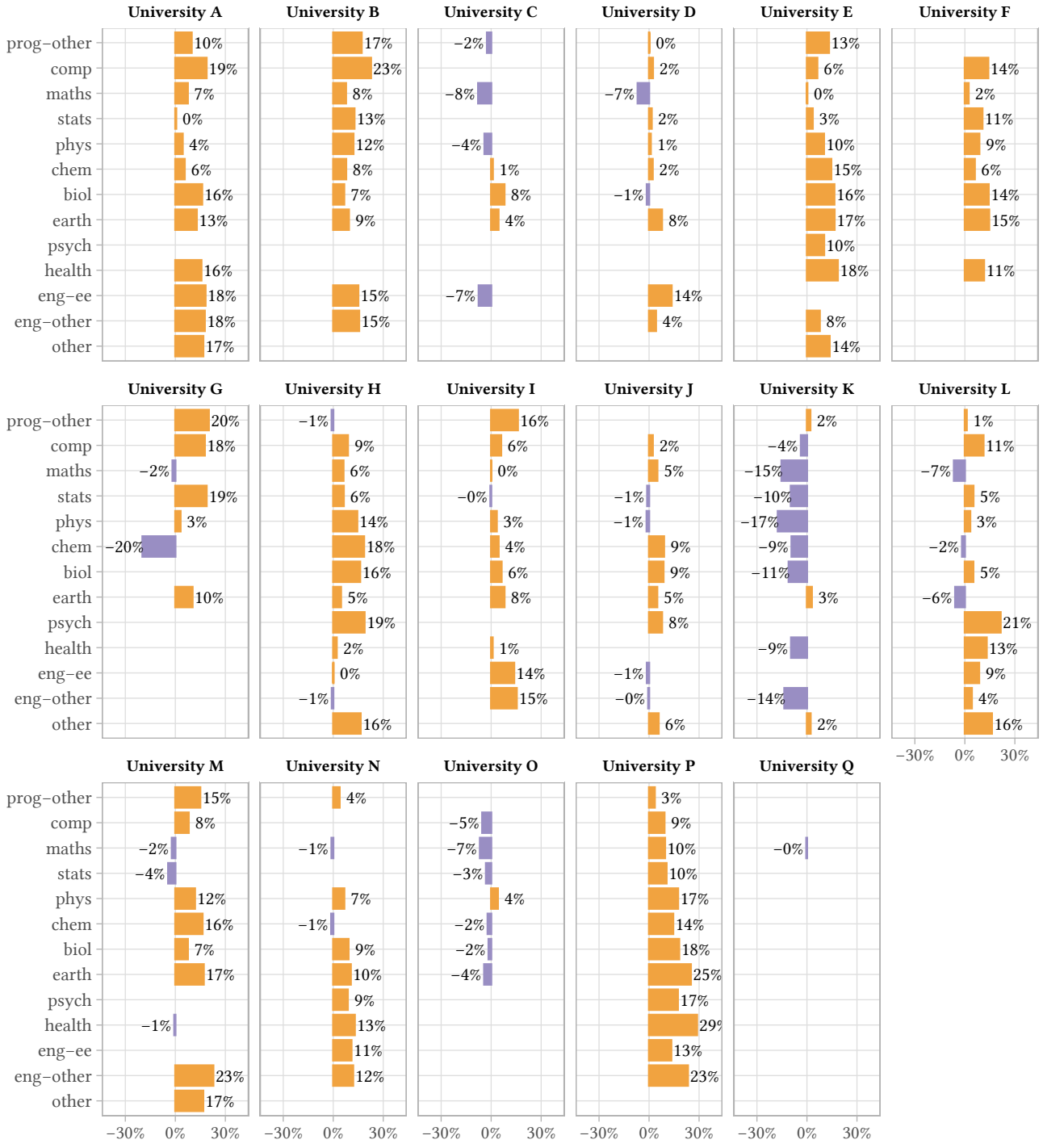

Figure 10: Differences between weighted pass rates in other STEM areas and introductory programming, for each university; a near-zero difference in pass rate is represented by a very thin bar (e.g., stats at University A or maths at University Q) and a percentage; the absence of a coloured bar and percentage means that we have no data for courses in that discipline at that university

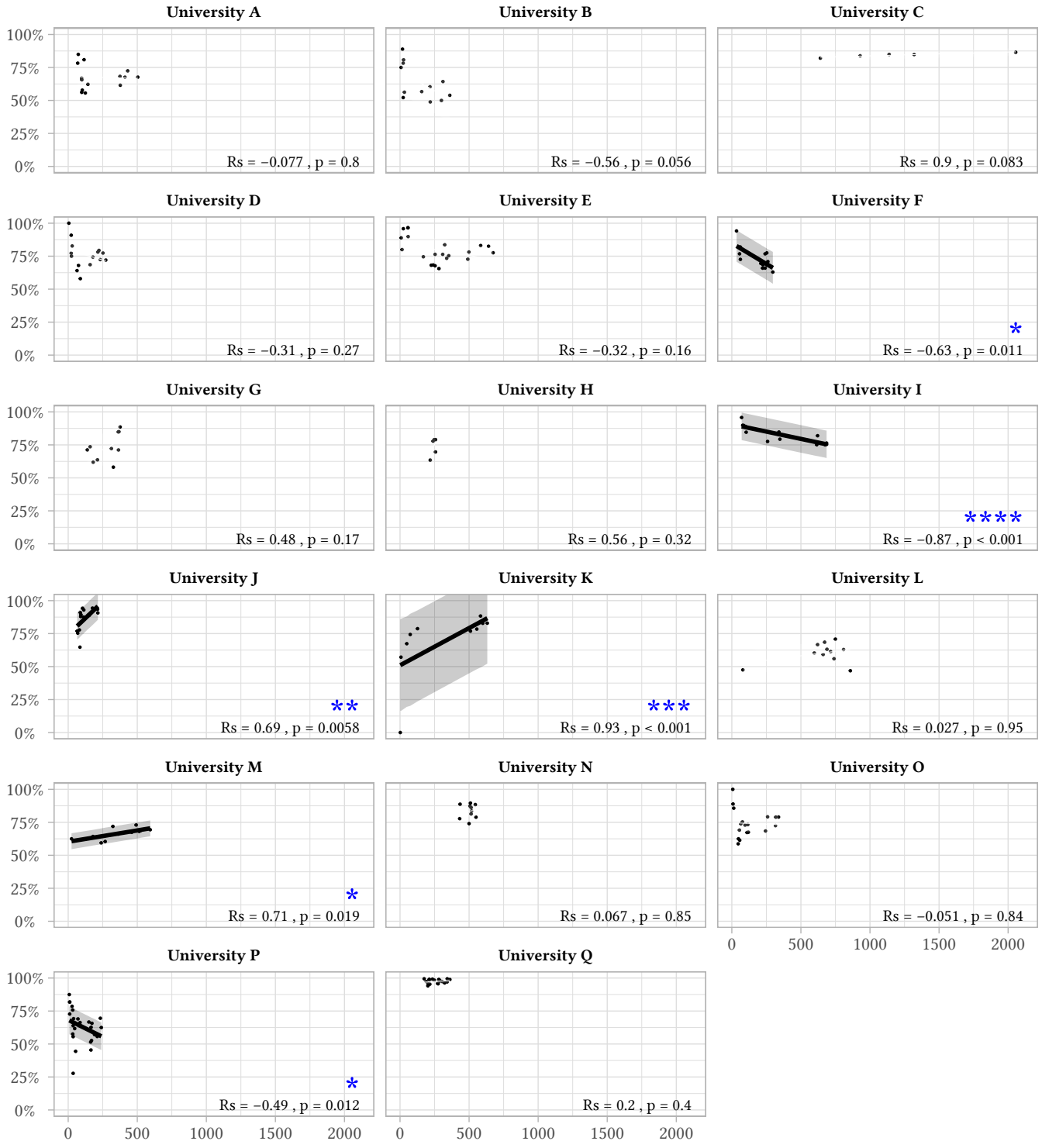

Figure 11: Relationships between course size (number of enrolments) and pass rate of introductory programming courses measured using Spearman's rank correlation (significance level denoted by asterisks: * for $p < 0.05$; ** for $p < 0.01$; * for $p < 0.001$; and **** for $p < 0.0001$)**

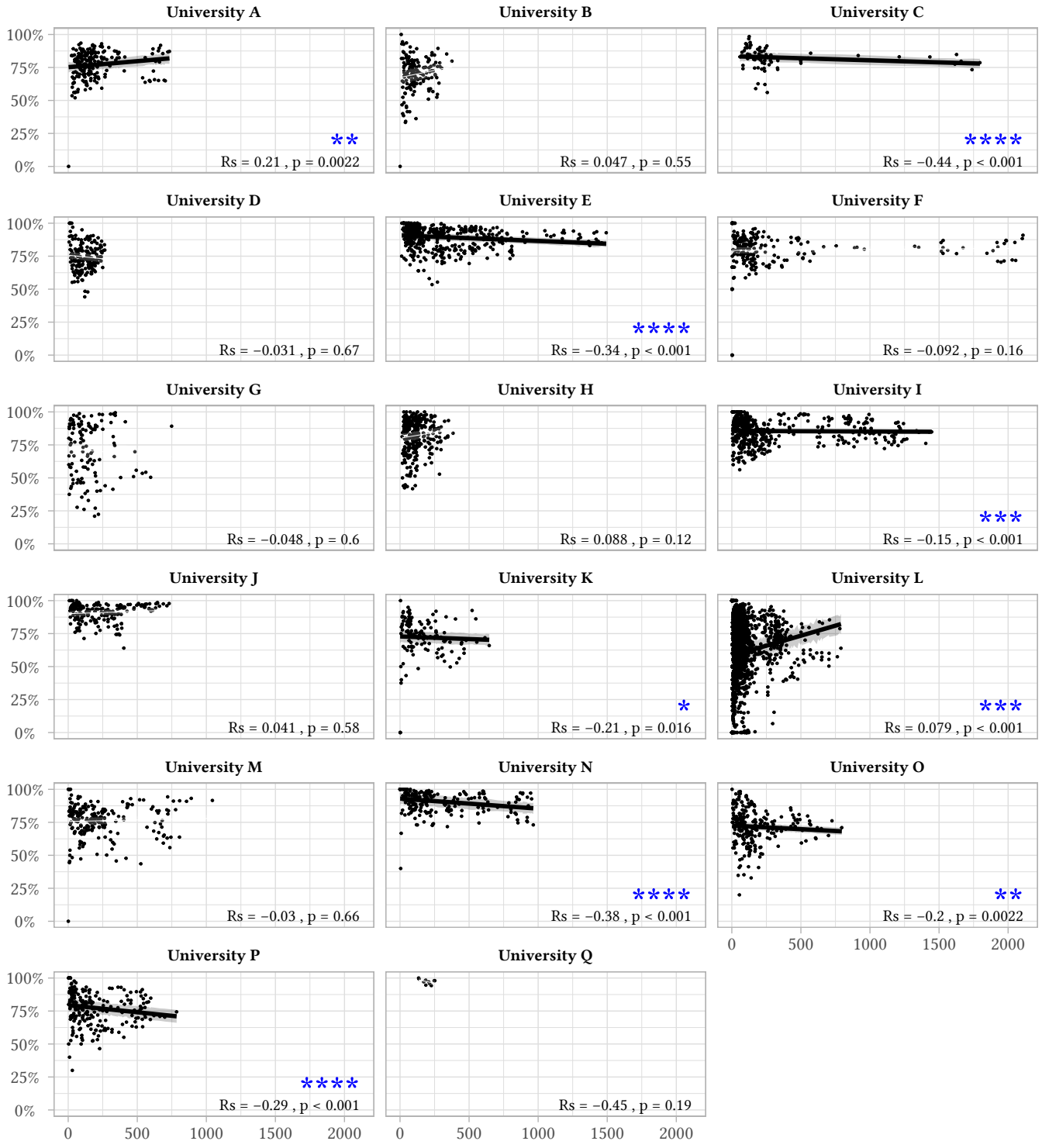

Figure 12: Relationships between course sizes (number of enrolments) and pass rates for introductory courses in other STEM areas measured using Spearman's rank correlation (significance level denoted by asterisks: * for $p < 0.05$; ** for $p < 0.01$; * for $p < 0.001$; and **** for $p < 0.0001$)**

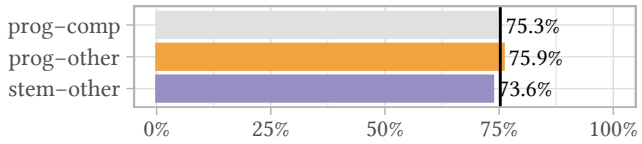

Figure 13: Unweighted average pass rates for introductory programming, cross-disciplinary programming, and other STEM fields

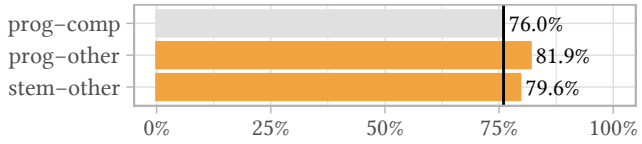

Figure 14: Weighted average pass rates for introductory programming, cross-disciplinary programming, and other STEM fields

5.2 Pass Rates in Introductory Programming Courses Compared with those in other Introductory STEM Courses

Figures 2 and 14 suggest that, weighted by student numbers, the average pass rate in introductory programming is somewhat lower than that in the rest of STEM. Figures 5 and 6 indicate that among specific STEM categories, only mathematics and statistics have lower weighted average pass rates than introductory programming. Furthermore, as indicated by figure 10, introductory programming has the lowest weighted average pass rate of any STEM area at five of the 17 participating institutions, and is in the bottom half for all but three institutions.

For further examination of figure 10, we define pass rates at an institution to be balanced if the average of all the weighted differences from *prog-comp* is less than 3%. By this criterion, universities C, D, O, and Q can be considered balanced. Of the remaining 13 institutions, only institution K has pass rates for introductory programming courses that are substantially higher than for other course categories. At the remaining 12 institutions, pass rates for introductory programming courses are substantially lower than for other course categories.

At the level of individual disciplines, mathematics, statistics and introductory programming are generally the three STEM fields with considerably lower pass rates than other STEM fields. One possible explanation for the lower pass rates could be that these three are commonly offered as service courses to be taken as minor subjects by students in other disciplines. For example, it might be more likely for biology students to take mathematics as a minor than for mathematics students to take biology as a minor.

Ultimately, though, as can be seen in figures 13 and 14, there is no great difference in pass rates between introductory programming and the rest of STEM. As a consequence, perhaps the community's attentions on the difficulty of delivering introductory programming courses are incorrectly directed, given that STEM disciplines on the whole appear to have similar pass rates. Perhaps instead of trying to change our approaches to teaching the introductory programming course, universities should focus on strengthening the skills required throughout STEM to improve outcomes across the board.

We acknowledge reports of the impact on pass rates of particular practices, innovations, or interventions. In computing, the positive impact of peer instruction (PI) [33] on four different courses spanning 16 PI course offerings over 10 years of instruction showed that course failure rates were reduced substantially when PI was used (from a failure rate of 24% using traditional delivery to 10% using peer instruction). Similarly, the use of pair programming in computing [29] has shown positive results, with students working in pairs significantly more likely to complete the course (91% versus 80%) and more likely to pass the course (73% versus 63%). Although looking at pass rates in general gives an overview of how the pass rates of programming may be experienced by instructors and students alike, it is worth noting that there are teaching approaches that may impact substantially on course pass rates. Further work that focuses on the relationship between course delivery and pass rate within institutional contexts would be interesting.

5.3 Trends in Pass Rates in Introductory Programming Courses over the Past Five Years

The original pass rate reported by Bennedsen and Caspersen [4] was 67%, while the more recent replication of the study [5] reported 73%, an increase of 6% over a 15-year period. Although this suggests an increasing trend, we found that the pass rates remained quite stable over the most recent five years (figure 1). There are several possible explanations for the difference in trends.

One possible explanation is that the data analysed in this study is from a time period that is too short to display trends. For example, the difference observed between our studies might be explained by a significant change in the delivery of introductory programming courses, or among the student cohorts enrolling in those courses, at some point between 2007 and 2014. An alternative explanation is that the data sources for the two studies by Bennedsen and Caspersen were different, so the studies have differences due to variation between the populations that contributed data, while our study reports longitudinal data from individual institutions. While the previously reported increase in pass rates may lead to speculation about grade inflation, we do not see evidence of an increase in pass rates in introductory programming courses or in other STEM courses over the past five years.

The differences between weighted and unweighted comparisons of the data have been described in section 4. This distinction was maintained throughout the discussion of the results in order to ensure that multiple interpretations of the data can be considered. One might assume that the fairest way to present this data would be to use the weighted aggregates only, letting every individual student pass carry the same weight. On the other hand, this would allow courses with high enrolments and perhaps non-normally distributed grades to skew data in ways that are of interest, but that do not represent the aggregates across course-level performance. This ignores a level of nuance offered by the unweighted data, effectively overwhelming the results observed in offerings with lower

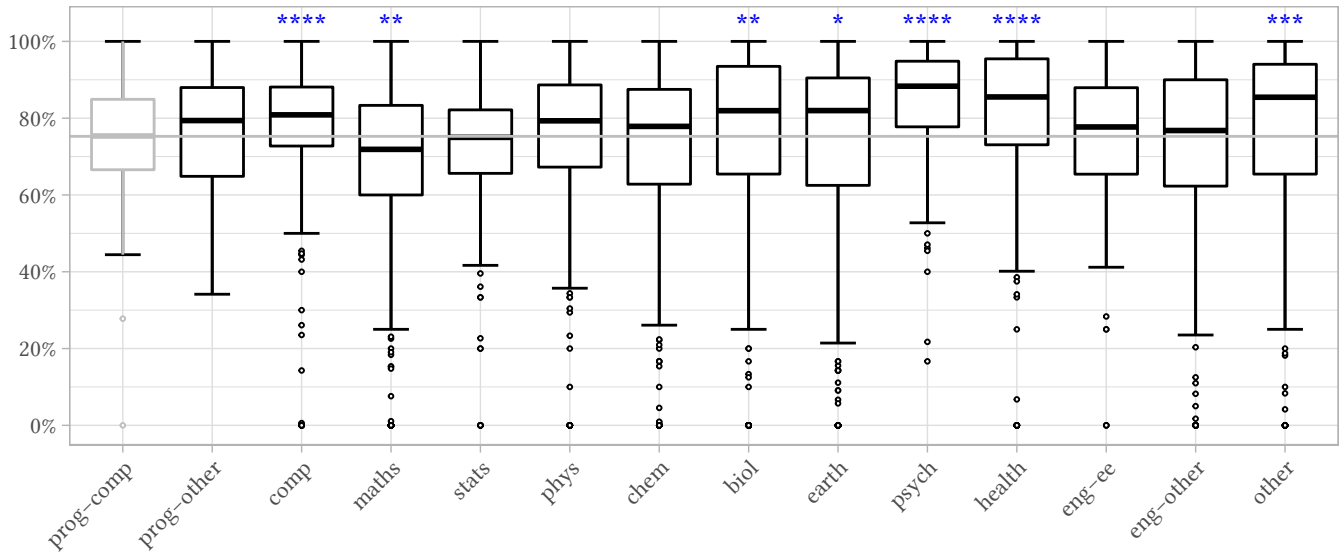

Figure 15: Unweighted pass rates by categories, comparing *prog-comp* with each other category (significance level denoted by asterisks: * for $p < 0.05$; ** for $p < 0.01$; * for $p < 0.001$; and **** for $p < 0.0001$)**

enrolments. Thus we saw value in working with and reporting on both forms of the data.

5.4 Further Observations

The authors of this report experienced considerable trouble obtaining the data required to compare pass rates. In some countries, such as Norway, the pass rates for individual courses are a matter of public record and are accessible through a public web interface. In other countries, such as New Zealand, pass rates at the course level are reported nationally and are available upon request in anonymous aggregate form. However, in many countries information about pass rates is very difficult to access, being perceived to be both academically sensitive and commercially sensitive.

Although for some institutions we had access to fine-grained data such as grade breakdowns, gender data, and/or data on majors, we were unable to use this data in our analysis since it was important to be able to report measurements as consistently as possible across the institutions. Attrition rate is another data set that would have been of particular interest. This might have helped to explain some of the larger gaps in aggregate pass rates between different subject areas, but the data was not available from enough institutions to permit such analysis.

Although we have reported pass rates calculated by aggregating the data, we appreciate that there is limited value in aggregating data on pass rates collected from very different contexts. For example, if one institution has a culture of offering free education to all citizens with no restrictions on entry, and subsequently reports a pass rate of 50%, while another institution has highly competitive entry criteria and reports a pass rate of 80%, how meaningful is it to report an average pass rate of 65%? Studies that report data that is aggregated across different contexts need to explicitly acknowledge the impact of the aggregation. The studies by Bennedsen and Caspersen [4, 5] and Watson and Li [35] all aggregate data without addressing the different contexts of the institutions that provided the data. It is difficult to interpret average pass rates of a single course such as introductory programming, so our comparison with other courses in the same institution acts as a control for some of the cultural and institutional factors that may bias the findings.

6 THREATS TO VALIDITY

The authors of this paper collected data from their own institutions. In some cases, the authors were involved in one or more of the introductory programming courses that appeared in the data set, which might indicate a bias in the data similar to that of previous research on pass rates [4, 5, 35], where the data was sourced from people involved in the computing education community. However, in this study the authors taught only a fraction of the 232 introductory programming course offerings recorded in the data collection, which limits the potential bias of course selection.

The quantitative results presented in this report are accurate for the data that we have gathered. However, although we have up to five years' data from 17 institutions in eight countries, we have no evidence that our data set is representative of introductory courses world-wide.

Even for the few institutions from which we have data, the variability in terminology and in marking and grading practices lead us to conclude that there is no single uniform understanding of what it means to pass a course. This particular threat to validity applies equally to prior work on pass or failure rates in introductory programming courses, notably that of Bennedsen and Caspersen [4, 5] and Watson and Li [35], and also more broadly to literature on pass rates in STEM courses.

The collected data did not include course categories. The authors classified the courses in the data set based on the course titles, course codes, and institutions that offered the course. Coming from computing, the authors are not experts in the fields corresponding

to the chosen set of categories, and so were unlikely to be 100% accurate. Furthermore, each course offering can deviate from its curriculum depending on the prioritisation of the course topics by the teacher, and the competencies/needs of the majority of students enrolled in the course, in that year.

As indicated in section 3.3, the decision about whether a course is an introductory STEM course is subject to error, and we have no way of knowing the extent, if any, to which this might have skewed our findings.

7 CONCLUSIONS AND FUTURE WORK

This ITiCSE working group set out to compare pass rates in introductory programming courses with those in other introductory STEM courses, by way of both a current snapshot from many institutions and five years of longitudinal data from a smaller number of institutions.

The intended current snapshot is based on data from only a small number of institutions. Our survey elicited only ten responses, which are so few in number, and so different from the remainder of our data set, that we could not meaningfully include them in our analysis. Our remaining data, while covering more than 200 offerings of introductory programming courses, is limited by coming from only 17 institutions. Even so, it suggests an average pass rate of close to 75%, which is not far removed from that in the most recent work of Bennedsen and Caspersen [5], despite coming from a very different global distribution of institutions.

The comparison with other STEM courses is perhaps the greatest contribution of the paper. Covering more than 5000 offerings of many hundreds of STEM courses, our data set appears to permit a reasonable comparison, over the past five years, of pass rates in introductory programming with those in other introductory STEM courses.

On the broadest scale, we found the pass rates in both introductory programming courses and other introductory STEM courses to be reasonably consistent over the five years, at values very close to 75%. This contributes to knowledge in this field by confirming similar findings to those reported by other authors.

At the scale of individual institutions, we found that there is a slight tendency for pass rates in introductory programming courses to be at the lower end of the range of STEM pass rates, but not substantially lower than in several other disciplines. This was not sufficiently strong to warrant the general belief that programming is hard to learn and programming is hard to teach. While authors of computing education papers have for decades perpetuated the belief that introductory programming course are hard to pass (in comparison, presumably, with other courses), we have established that it is neither possible nor meaningful to measure this precisely; but that the imprecise work we have done suggests that the belief is ill founded.

In many countries, computing is currently being introduced at various primary and secondary school levels. It would be interesting to replicate the study in some five years or so, to assess whether this has led to a measurable change. Such a further study might also be designed from the outset to seek additional data, such as on gender and whether the students are from the country of study or from other countries. While it would be interesting to have included an analysis of such data in the current study, we did not initially request the data, and it would now be too challenging to go back and try to acquire it.

At the risk of oversimplifying, this paper finds no evidence that pass rates for introductory programming courses are substantially lower than for other STEM courses. The authors of this report would find it gratifying if in future they were able to discern a reduction in the number of computing education papers that begin with the sentiment 'introductory programming courses are hard to pass'. However, the pass rates that we found clearly leave scope for improvement in the teaching and learning of introductory programming, and future research should continue to investigate approaches that might enhance our understanding and delivery of introductory programming courses.

ACKNOWLEDGMENTS

We thank the ten people who responded to our anonymous survey; we do understand that the data we sought was not easy to obtain. We are particularly grateful to Madeleine Lorås, who was prevented by circumstances from joining the working group, but nevertheless contributed substantially to its work.

REFERENCES

[1] 2019. Higher Education Statistics Agency (HESA). https://www.hesa.ac.uk/news/07-03-2019/non-continuation-tables. Accessed 13 Jul 2019.

[2] Gillian Bain and Ian Barnes. 2014. Why is programming so hard to learn?. In *19th Conference on Innovation & Technology in Computer Science Education (ITiCSE 2014)*. ACM, 356–356.

[3] Mousumi Banerjee, Michelle Capozzoli, Laura McSweeney, and Debajyoti Sinha. 1999. Beyond kappa: a review of interrater agreement measures. *Canadian Journal of Statistics* 27, 1 (1999), 3–23. https://doi.org/10.2307/3315487

[4] Jens Bennedsen and Michael E Caspersen. 2007. Failure rates in introductory programming. *SIGCSE Bulletin* 39, 2 (June 2007), 32–36. https://doi.org/10.1145/1272848.1272879

[5] Jens Bennedsen and Michael E Caspersen. 2019. Failure rates in introductory programming: 12 years later. *ACM Inroads* 10, 2 (April 2019), 30–36. https://doi.org/10.1145/3324888

[6] Susan Bergin, Ronan Reilly, and Desmond Traynor. 2005. Examining the role of self-regulated learning on introductory programming performance. In *First International Workshop on Computing Education Research (ICER 2005)*. ACM, 81–86.

[7] Richard Bornat, Saeed Dehnadi, and Simon. 2008. Mental models, consistency and programming aptitude. In *10th Australasian Computing Education Conference (ACE 2008)*. Australian Computer Society, Inc, 53–61. http://dl.acm.org/citation.cfm?id=1379249.1379253

[8] Roger Boyle, Janet Carter, and Martyn Clark. 2002. What makes them succeed? Entry, progression and graduation in computer science. *Journal of Further and Higher Education* 26, 1 (2002), 3–18.

[9] Elisabeth Chapman, Elisabeth M Wultsch, Jan DeWaters, John C Moosbrugger, Peter R Turner, Michael W Ramsdell, and Robert P Jaspersohn. 2015. Innovating engineering curriculum for first-year retention. In *122nd ASEE Annual Conference, Seattle WA*. 26.967.1–26.967.24.

[10] Xianglei Chen. 2013. STEM attrition: college students' paths into and out of STEM fields. Statistical analysis report. NCES 2014-001. *National Center for Education Statistics* (2013), 1–96.

[11] Alison Clear, Janet Carter, Amruth Kumar, Cary Laxer, Simon, and Ernesto Cuadros-Vargas. 2015. Global perspectives on assessing educational performance and quality. In *20th Conference on Innovation and Technology in Computer Science Education (ITiCSE 2015)*. ACM, 326–327. https://doi.org/10.1145/2729094.2754843

[12] Alison Clear and Tony Clear. 2014. Introductory programming and educational performance indicators – a mismatch. In *2014 ITx Conference (ITx 2014)*. CITRENZ, New Zealand, 123–128.

[13] Mark Davies and Joseph L Fleiss. 1982. Measuring agreement for multinomial data. *Biometrics* 38, 4 (1982), 1047–1051. http://www.jstor.org/stable/2529886

[14] Peter J Denning. 2005. Is computer science science? *Communications of the ACM* 48, 4 (April 2005), 27–31. https://doi.org/10.1145/1053291.1053309

[15] Ronald G Ehrenberg. 2010. Analyzing the factors that influence persistence rates in STEM field, majors: Introduction to the symposium. *Economics of Education Review* 29, 6 (2010), 888 – 891.

[16] Katrina Falkner, Rebecca Vivian, and Nickolas JG Falkner. 2014. Identifying computer science self-regulated learning strategies. In *19th Conference on Innovation & Technology in Computer Science Education (ITiCSE 2014)*. ACM, 291–296.

[17] Scott Freeman, Sarah L Eddy, Miles McDonough, Michelle K Smith, Nnadozie Okoroafor, Hannah Jordt, and Mary Pat Wenderoth. 2014. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences* 111, 23 (2014), 8410–8415.

[18] Scott Freeman, David Haak, and Mary Pat Wenderoth. 2011. Increased course structure improves performance in introductory biology. *CBE – Life Sciences Education* 10, 2 (Summer 2011), 175–186.

[19] Mark Guzdial. 2010. Why is it so hard to learn to program? In *Making Software: What Really Works, and Why We Believe It*, Andy Oram and Greg Wilson (Eds.). O'Reilly Media, 111–124.

[20] Mark Guzdial. 2019. A biased attempt at measuring failure rates in introductory programming. https://computinged.wordpress.com/tag/failure-rates/.

[21] Rashina Hoda and Peter Andreae. 2014. It's not them, it's us! Why computer science fails to impress many first years. In *16th Australasian Computing Education Conference (ACE 2014)*. Australian Computer Society, Inc, 159–162. http://dl.acm.org/citation.cfm?id=2667490.2667509

[22] Philip R Ventura Jr. 2005. Identifying predictors of success for an objects-first CS1. *Computer Science Education* 15, 3 (2005), 223–243.

[23] Päivi Kinnunen and Lauri Malmi. 2006. Why students drop out CS1 course?. In *Second International Workshop on Computing Education Research (ICER 2006)*. ACM, 97–108. https://doi.org/10.1145/1151588.1151604

[24] Päivi Kinnunen and Lauri Malmi. 2008. CS Minors in a CS1 Course. In *Fourth International Workshop on Computing Education Research (ICER 2008)*. ACM, 79–90.

[25] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A study of the difficulties of novice programmers. *SIGCSE Bulletin* 37, 3 (June 2005), 14–18. https://doi.org/10.1145/1151954.1067453

[26] Caroline Liron and Heidi M Steinhauer. 2015. Analyzing longitudinal performance from multi-course alignment for 1st year engineering students: calculus, physics, and programming in MATLAB. In *122nd ASEE Annual Conference, Seattle WA*. 26.216.1–26.216.10.

[27] Andrew Luxton-Reilly. 2016. Learning to program is easy. In *21st Conference on Innovation and Technology in Computer Science Education (ITiCSE 2016)*. ACM, 284–289. https://doi.org/10.1145/2899415.2899432

[28] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A Becker, Michail Giannakos, Amruth N Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory programming: a systematic literature review. In *ITiCSE 2018 Working Group Reports (ITiCSE-WGR 2018)*. ACM, 55–106. https://doi.org/10.1145/3293881.3295779

[29] Charlie McDowell, Linda Werner, Heather E Bullock, and Julian Fernald. 2003. The impact of pair programming on student performance, perception and persistence. In *25th International Conference on Software Engineering (ICSE 2003)*. IEEE Computer Society, 602–607.

[30] Dale Parsons, Krissi Wood, and Patricia Haden. 2015. What are we doing when we assess programming?. In *17th Australasian Computing Education Conference (ACE 2015)*. Australian Computer Society, Inc, 119–127. http://crpit.com/confpapers/CRPITV160Parsons.pdf

[31] Alan R Peterfreund, Kenneth A Rath, Samuel P Xenos, and Frank Bayliss. 2008. The impact of supplemental instruction on students in STEM courses: results from San Francisco State University. *Journal of College Student Retention: Research, Theory & Practice* 9, 4 (2008), 487–503. https://doi.org/10.2190/CS.9.4.e arXiv:https://doi.org/10.2190/CS.9.4.e

[32] Victor Pigott and Denise Frawley. 2019. An analysis of completion in Irish higher education: 2007/08 entrants. *Higher Education Authority* (2019).

[33] Leo Porter, Cynthia Bailey Lee, and Beth Simon. 2013. Halving fail rates using peer instruction: a study of four computer science courses. In *44th ACM Technical Symposium on Computer Science Education (SIGCSE 2013)*. ACM, 177–182.

[34] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: a review and discussion. *Computer Science Education* 13, 2 (2003), 137–172.

[35] Christopher Watson and Frederick WB Li. 2014. Failure rates in introductory programming revisited. In *19th Conference on Innovation & Technology in Computer Science Education (ITiCSE 2014)*. ACM, 39–44. https://doi.org/10.1145/2591708.2591749