Investigating Novice Programming Mistakes: Educator Beliefs vs Student Data

Neil C. C. Brown and Amjad Altadmri
School of Computing
University of Kent
Canterbury, Kent, UK
{nccb,aa803}@kent.ac.uk

ABSTRACT

Educators often form opinions on which programming mistakes novices make most often – for example, in Java: "they always confuse equality with assignment", or "they always call methods with the wrong types". These opinions are generally based solely on personal experience. We report a study to determine if programming educators form a consensus about which Java programming mistakes are the most common. We used the Blackbox data set to check whether the educators' opinions matched data from over 100,000 students – and checked whether this agreement was mediated by educators' experience. We found that educators formed only a weak consensus about which mistakes are most frequent, that their rankings bore only a moderate correspondence to the students in the Blackbox data, and that educators' experience had *no* effect on this level of agreement. These results raise questions about claims educators make regarding which errors students are most likely to commit.

Categories and Subject Descriptors

K.3.2 [**Computers And Education**]: Computer and Information Science Education

General Terms

Experimentation, Human Factors

Keywords

Programming Mistakes; Educators

1. INTRODUCTION

Educators naturally form opinions on common mistakes that their students make when learning to program. Such opinions are reflected in textbooks – for example, in Java: "Failing to use equals() to compare two strings is probably the most common single mistake made by Java novices" [22], "The most common mistake made with an if statement is the use of a single equal sign to compare equality" [7], "A common mistake in for loops is to accidentally put a semicolon at the end of the line that includes the for statement" [4]. In this paper, we set out to determine: are such observations accurate in general? This matters for several reasons. Firstly, recent research by Sadler et al [15] suggests that knowledge of student misconceptions is important to educator efficacy, and thus it is of interest whether educators' impressions about student mistakes are correct. Secondly, when educators communicate with each other (face-to-face or online), it matters whether their opinions on common student mistakes are accurate and generalise to each other's students. Knowing which mistakes are common also informs the writing of instructional materials such as textbooks.

Previous studies that have investigated student errors during [Java] programming have focused on cohorts of up to 600 students at a single institution [1, 5, 6, 10, 11, 19]. However, the recently launched Blackbox data collection project [3] affords an opportunity to observe the mistakes of a large number of students at a variety of institutions – for example, from September to December 2013, the project collected error messages and Java code from around 110,000 users.

In this paper we describe a study to check if educators' opinions on beginners' mistakes in Java form a consensus, and whether they generalise to all students. We combine a survey of educators with data from the Blackbox project to answer the following research questions:

- Do educators' views form a consensus about the frequency of specified student Java mistakes?
- Do educators' views of mistake frequency match the observed behaviour of students in a large-scale multi-institution data set?
- Are more experienced educators' views more likely to match the students' data than novice educators?

2. RELATED WORK

2.1 Observing Student Errors

The concept of monitoring student behaviour and mistakes while programming has a long history in computing education research. The Empirical Studies of Programming [16] workshops in the 1980s had several papers making use of this technique for Pascal and other languages. More recently, there have been many such studies specifically focused on Java, which is also the topic of this study.

ICER'14, August 11-13, 2014, Glasgow, Scotland, UK.

ACM 978-1-4503-2755-8/14/08 ...$15.00.
http://dx.doi.org/10.1145/2632320.2632343.

Many of these studies used compiler error messages to classify mistakes. Jadud [11] looked in detail at student mistakes in Java and how students went about solving them. Tabanao et al. [19] investigated the assocation between errors and student course performance. Denny et al. [5] looked at how long students take to solve different errors, Dy and Rodrigo [6] looked at improving the error messages given to students, and Ahmadzadeh et al. [1] looked at student error frequencies and debugging behaviour. Jackson et al. [10] identified the most frequent errors among their novice programming students. All six of these studies used compiler error messages to classify errors. However, early results from McCall [14] suggest that compiler error messages have an imperfect (many-to-many) mapping to student misconceptions. Additionally, all six studies looked at cohorts of (up to 600) students from a single institution.

Our study is novel in that it looks at student mistakes from a much larger number of students (over 100,000) from a large number of institutions[1], thus providing more robust data about error frequencies. An earlier paper about Blackbox gave a brief list of the most frequent compiler error messages [3], but in this study we do not simply use compiler error messages to classify errors. Instead, we borrow error classifications from Hristova et al. [9], which are based on surveying educators to ask for the most common Java mistakes they saw among their students.

2.2 Educators' Opinions

Spohrer and Soloway [17] proposed in 1986 to examine the accuracy of educator folk wisdom. However, they did not survey educators, and instead took two anecdotal folk wisdoms about learning to program and then compared them to actual data from students (in Pascal). Ben-David Kolikant [2] interviewed some educators about students' approaches to programming, but at a higher level, without reference to a specific language. Hristova et al. [9] surveyed a combination of local teaching assistants and students, as well as educators from 58 universities, asking about common Java mistakes. This data was combined to form a list of 20 Java mistakes that were detectable at compile-time. We base our classification of mistakes on this work, but Hristova et al. did not test these predictions themselves against actual student data. As far as we are aware, our study is the first to survey educators about Java programming mistakes and compare their opinions to actual data.

3. METHOD

3.1 Student Mistakes

We used Hristova et al's [9] twenty student mistakes as a basis for our analysis. We removed two mistakes: firstly, "leaving a space after a period when calling a method", which is a style issue not a programming mistake, and secondly, "improper casting" which was not described clearly enough for us to operationalise. We also altered one further question "invoking a class method on an object" after several of our pilot testers remarked that this was not a mistake, but the reverse (invoking an instance method on a class) was, and they had seen the latter much more often. This left eighteen misconceptions, which we labelled A through R:

A: Confusing the assignment operator (`=`) with the comparison operator (`==`).
For example: `if (a = b) ...`

B: Use of `==` instead of `.equals` to compare strings.
For example: `if (a == "start") ...`

C: Unbalanced parentheses, curly brackets, square brackets and quotation marks, or using these different symbols interchangeably.
For example: `while (a == 0]`

D: Confusing "short-circuit" evaluators (`&&` and `||`) with conventional logical operators (`&` and `|`).
For example: `if ((a == 0) & (b == 0)) ...`

E: Incorrect semi-colon after an if selection structure before the if statement or after the for or while repetition structure before the respective for or while loop.
For example:

```
if (a == b);
    return 6;
```

F: Wrong separators in for loops (using commas instead of semi-colons)
For example: `for (int i = 0, i < 6, i++) ...`

G: Inserting the condition of an if statement within curly brackets instead of parentheses.
For example: `if {a == b} ...`

H: Using keywords as method names or variable names.
For example: `int new;`

I: Invoking methods with wrong arguments (e.g. wrong types).
For example: `list.get("abc")`

J: Forgetting parentheses after a method call.
For example: `myObject.toString;`

K: Incorrect semicolon at the end of a method header.
For example:

```
public void foo();
{
    ...
}
```

L: Getting greater than or equal/less than or equal wrong, i.e. using `=>` or `=<` instead of `>=` and `<=`.
For example: `if (a =< b) ...`

M: Trying to invoke a non-static method as if it was static.
For example: `MyClass.toString();`

N: A method that has a non-void return type is called and its return value ignored/discarded.
For example: `myObject.toString();`

O: Control flow can reach end of non-void method without returning.
For example:

```
public int foo(int x)
{
    if (x < 0)
        return 0;
    x += 1;
}
```

P: Including the types of parameters when invoking a method.
For example: `myObject.foo(int x, String s);`

Q: Incompatible types between method return and type of variable that the value is assigned to.
For example: `int x = myObject.toString();`

R: Class claims to implement an interface, but does not implement all the required methods.
For example: `class Foo implements ActionListener { }`

[1] We have no way of measuring the number of institutions in the Blackbox data, but simply: the 100,000 students must be split over at least several hundred institutions.

3.2 Educators' Survey

We prepared a questionnaire that listed our eighteen student mistakes and asked respondents to rate each mistake on a scale of infrequent to frequent, by making a mark along a visual analogue scale (a straight line with endpoints, like so: ⊢——⊣). These scales were measured to the nearest $\frac{1}{100}$ of their length and recorded as a number from 0 to 100.

This paper questionnaire was given out to attendees of the ICER 2013 conference. An equivalent online electronic version of the questionnaire was also later developed, and was advertised via the SIGCSE mailing list, the UK Computing At School forum and through Twitter. (Those who had responded to the paper version were instructed not to complete the online version.)

29 participants returned a paper questionnaire and 191 started filling out the online questionnaire (although many did not complete). Only 76 participants filled in all scales for all questions (20 paper and 56 online), and this formed the sample set for all analyses. Participants were also asked about their educational experience in different sectors. 56 had experience only in the tertiary sector (age 18+), 3 only in secondary (ages 11–18), 14 in secondary and tertiary, and the remaining 3 in tertiary, secondary and primary (ages 4–11). The educators' experience is detailed and analysed in the results in section 4.1.

3.2.1 Inter-Educator Agreement

To measure agreement among educators we used Kendall's coefficient of concordance (aka Kendall's W) [12]. This statistic can be used to assess the agreement among ranks assigned by a group of raters to a set of items, by looking at the variance among the ranks of the different mistakes.

3.3 Student data

Data about student mistakes was taken from the Blackbox data set [3], which collects Java code written by users of BlueJ, the Java beginners' IDE. We used data from the period 1st Sep. 2013 to 31st Dec. 2013 (inclusive), as representing the autumn/winter term in the northern hemisphere.

We had three methods of detecting mistakes. For four of the student mistakes, *I, M, O, R*, we were able to use the compiler error message directly to detect the mistake. However, this was not possible for the other errors, as some of them are logical errors that do not cause a compiler error or warning, while in other cases the error messages do not have a one-to-one mapping to our mistakes of interest. Thus for one of the other mistakes (C) we performed a post-lexing analysis (matching brackets) and for the final thirteen we used a customised permissive parser to parse the source code and look for the errors. The source code of all the tools used will be available shortly.

We took each source file in the data set, and tracked the file over time. At each compilation we checked the source file for the eighteen mistakes. If the mistake was present, we then looked forward in time to find the next compilation where the mistake was no longer present (or until we had no further data for that source file). When the mistake was no longer found – which could have been because the mistake was corrected or because the offending code was removed or commented out – we counted this as one instance of the mistake. Further occurrences in the same source file were treated as further instances.

3.4 Educator and Student Agreement

To measure agreement between educators' ratings and the Blackbox frequencies, we used the average Spearman's ρ (rho) for pairwise comparisons between each educator and the Blackbox data (thus: one correlation per rater)[2]. (We term these pairwise correlations between educators and the Blackbox data: educator *accord.*) This use of the average was originally recommended by Lyerly [13], then explained and generalised by Taylor and Fong [21, 20] to add a significance test. In our example, Taylor's $\bar{\rho}_{t,c}$ is the average of the pairwise correlations between the Blackbox data and each educator, corrected for continuity.

3.5 Educator and Student Agreement – Effect of Experience

To check if this accord was affected by educators' experience, we used the following procedure. As described in the previous section, we first calculated educator accord, using Spearman's ρ as a measure of agreement between each educator's rankings and the Blackbox rankings (one correlation per rater). This accord was then correlated (again with Spearman's ρ) with the educators' total years of experience[3]. A significant correlation would indicate an effect of experience on educators' agreement with the Blackbox data.

4. RESULTS

4.1 Educators

Our analysis used responses from the 76 educators who gave rankings to all of the eighteen student mistakes. Educators were also asked how many years they had been an educator, to the nearest year[4]. Educators were also asked for the number of years spent teaching introductory programming, in any language or in Java, to three different age groups (4–11, 11–18, and 18+), so 6 numbers in all. We wished to combine these into a measurement of years spent teaching introductory programming in any language or in Java (i.e. collapsing across age group)[5]. Examination of the data suggested that some educators had taught some age groups simultaneously, so rather than summing across the age groups, we used the maximum figure from the three age groups. (Only 17 of the 76 had taught more than one age group.) We also capped the years spent teaching Java at 19, the language's current age, which affected two educators who claimed to have been teaching Java for more than 19 years. Frequencies for the years of experience are given in Figure 1.

4.2 Agreement Among Educators

Our analysis of Kendall's coefficient of concordance among the educators produced the result $W = 0.408$. For aid in

[2] Spearman's ρ is a correlation between the ranks of the two different variables, and thus looks only at the *ordering* of mistake frequency, not the exact frequencies nor the educator's 0–100 ratings.

[3] Our use of ρ here means that we do not look for a linear effect of experience (e.g. accuracy increasing linearly from 5 to 10 to 15 years), but rather: when educators are ordered by experience, does this match their ordering by accuracy?

[4] Since most educators begin their career in September, asking for a more precise measurement than the nearest year would not provide greater fidelity.

[5] In hindsight, we should have asked for these figures directly.

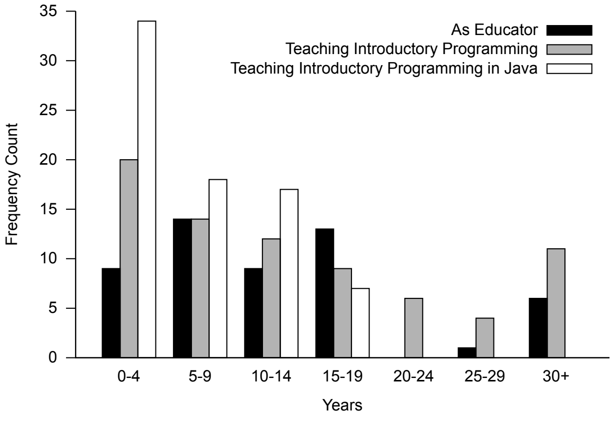


Figure 1: The distribution of years for surveyed educators: how long they have been an educator (solid black bar, $N = 56$), how many years they have spent teaching introductory programming (grey bar, $N = 76$) and how many years they have spent teaching introductory programming in Java (hollow bar, $N = 76$). Note that due to the different number of total responses, comparisons between the black bar and other bars are problematic.

interpretation, we use a conversion to Spearman's ρ correlation for ranked data [8, p313], which gives $\rho = 0.400$. Informally, this means that the educators are closer to chance agreement than they are to complete agreement. The educators we surveyed form a very weak consensus about which errors are most frequently made by students.

4.3 Student Mistakes

We used our detector to look for the number of instances of mistakes, as described in the method section. The data set featured 14,235,239 compilation events, of which 7,333,201 were successful and 6,902,038 were unsuccessful. Each compilation may include multiple source files – the total number of source files considered was 17,144,721 files, of which 8,787,189 were compiled successfully and 8,357,532 were not.

We can informally categorise the mistakes as follows:

Misunderstanding (or forgetting) syntax:

- A (confusing `=` with `==`),
- C (mismatched parentheses),
- D (confusing `&` with `&&`),
- E (spurious semi-colon after if, for, while),
- F (wrong separator in for),
- G (wrong brackets in if),
- H (keyword as variable or method name),
- J (forgetting parentheses when calling methods),
- K (spurious semi-colon after method header),
- L (less-than/greater-than operators wrong),
- P (including types in actual method arguments).

Type errors:

- I (calling method with wrong types),
- Q (type mismatch assigning method result).

Other semantic errors:

- B (using `==` to compare strings),
- M (invoking instance method as static),
- N (discarding method return),
- O (missing return statement),
- R (missing methods when implementing interface).

Mistake	Frequency	Error Type
C	404560	Syntax
I	165832	Type
O	137230	Semantic
N	86107	Semantic
A	68254	Syntax
B	45012	Semantic
M	30754	Semantic
R	24846	Semantic
P	21694	Syntax
E	20264	Syntax
K	16156	Syntax
Q	14371	Type
D	11212	Syntax
J	8332	Syntax
L	1916	Syntax
F	1171	Syntax
H	415	Syntax
G	63	Syntax

Table 1: Frequency of mistakes committed by students from 1st Sep. 2013 to 31st Dec. 2013 (incl.).

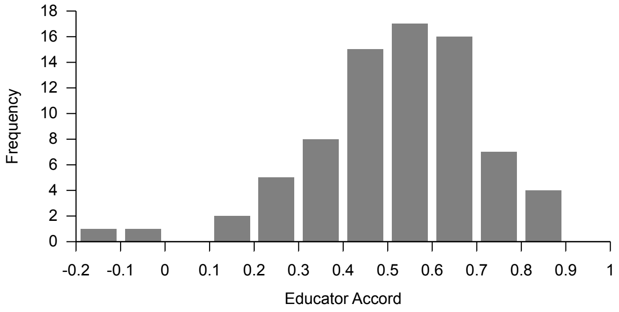


Figure 2: The distribution of accord scores. Picking ranks randomly would give an educator an average accord of 0, while 1 would indicate perfect agreement (and -1 perfect disagreement). Inspection of Q-Q plots confirms data is normal, as expected [21], with two notable outliers (visible above, far left).

Note that mistake N (ignoring the non-void result of a method) is not a compile error, and is not always an error (e.g. when you call a remove method that returns the item removed, you may not need to do anything with the return).

The frequencies of the number of instances of the different mistakes are shown in table 1, along with our informal classification of the error message. It can be seen that the type and semantic errors generally occur more frequently than the syntax errors.

4.4 Educator and Student Agreement

The average ρ for correlations between each educator and the Blackbox data was 0.514 (3 s.f.). Corrected for continuity [20], this gives a $\bar{\rho}_{t,c} = 0.514$, so $z = 18.5$ (3 s.f.) and thus $p < 0.001$. Since the standard deviation of $\bar{\rho}_t, c$ is 0.028 [21]

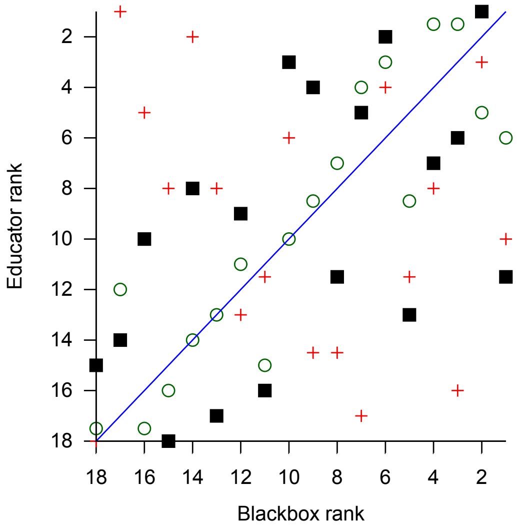


Figure 3: Examples of different accord scores. Perfect agreement is shown by the solid line, and accord is proportional to the square of the vertical distance of each point from the line. The empty circles are the ranks assigned by the educator with highest accord (0.876). The solid squares are the ranks assigned by one of the two educators surrounding the median accord (0.537). The pluses are the ranks assigned by the educator with lowest accord (-0.137).

and normality is assumed, the 95% confidence interval is [0.460, 0.569] (3 s.f.). Therefore, there was a statistically significant overall agreement, termed accord, between educators and the Blackbox data, with an average correlation of 0.514. The distribution of accord scores is shown in Figure 2. An example of accord is shown in Figure 3. The latter figure shows that while the educator with highest accord was reasonably close to the correct answer, the median educator in our sample (the solid squares) had only a moderate level of agreement, ranking the most popular Blackbox mistake as joint eleventh, and getting most other errors wrong by around four ranks. This level of agreement between educators and the student data was thus in general quite low.

4.5 Educator and Student Agreement – Effect of Experience

The relationship between educator accord (i.e. agreement with the Blackbox data) and years of being an educator is shown in Figure 4. The Spearman's ρ correlation was not significant at the 5% level, $\rho = -0.180, p = 0.202$. As a follow up analysis, we also examined whether years of experience teaching introductory programming or teaching introductory programming in Java had an effect (alpha corrected to 0.025 for multiple comparisons). The result for correlating years spent teaching introductory programming in any language with accord was not significant, $\rho = -0.151, p = 0.192$, and neither was the result correlating years spent teaching introductory Java programming with accord: $\rho = 0.04, p = 0.972$. Thus, there was no effect of educator experience (in any measure we tried) on an educator's level of agreement with the Blackbox data.

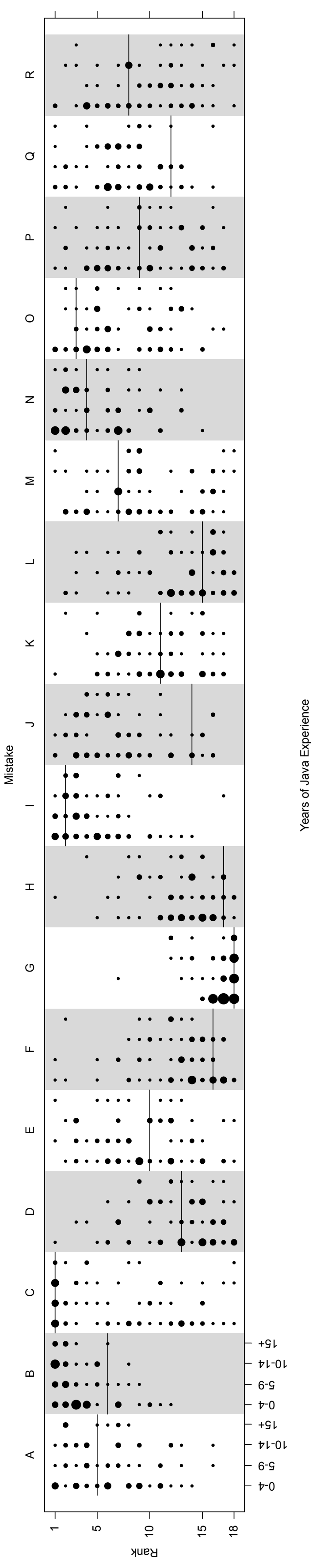
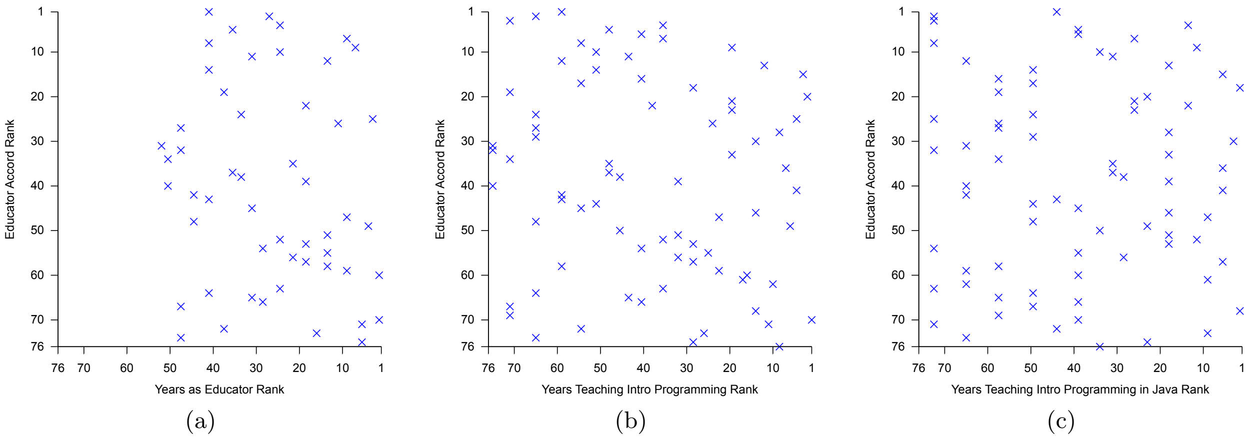


Figure 5: Graphic illustrating the effect of years spent teaching introductory programming in Java on mistake frequency rating. Each vertical band (alternately shaded for presentation) is a separate mistake and effectively a separate graph, with years plotted along the X axis, grouped into 4 groups, versus ranking (Y axis). The area of each circle is proportional to frequency (number of educators). If there was agreement among all raters, we would expect to see horizontal grouping around a common rating within each band. Alternatively, if the amount of experience had an effect on rating, we would expect to see a non-horizontal diagonal trend within each band. In this diagram, it can be seen that agreement of either kind is generally weak. For information, the rank derived from the Blackbox data is drawn on to each band as a horizontal line, and thus accord is (informally) the vertical distance of the circles from these horizontal lines. If experience had an effect on accord, we would see that the righthand points within each band were closer to the horizontal line than the lefthand points, but this is not the case.



(a)

(b)

(c)

Figure 4: Graphs showing ranks for educator accord (i.e. their agreement with the Blackbox data) against ranks for (a) the years they have been an educator, (b) the years they have taught introductory programming and (c) the years they have taught introductory programming in Java. Ranks are plotted such that the highest values (and thus lowest ranks) are towards the top or right of the graph. Perfect agreement would be a diagonal line from bottom-left to top-right; perfect disagreement would be a diagonal line from top-left to bottom-right. Note that for (a), only 52 of the 76 educators in our sample answered this question (whereas all answered for (b) and (c)). Rank correlations were insignificant in all three cases.

5. DISCUSSION

The answers to our research questions from section 1 are:

- Educators' views form only a weak consensus about the frequency of the specified student Java mistakes.
- Educators' views of mistake frequency have only a moderate match ("accord") with the observed behaviour of students in a large-scale multi-institution data set.
- This accord has no association with educator experience.

A graphic displaying most of the study results together can be found in Figure 5.

5.1 Inter-Educator Agreement

Our implicit expectation when beginning this research was that there would be a reasonably strong consensus among educators as to the frequency of Java mistakes. The low level of agreement between educators was a surprise – and even more so because years of experience (generally, or within Java) made no apparent difference to the ratings assigned by the educators (see Figure 5 for a visualisation of these results). This result colours some of the interpretation of the results regarding agreement between the educators and Blackbox: if educators don't agree among each other, it is unlikely that there would be very strong agreement between the educators and the Blackbox data.

Our results show that educator opinions about student mistakes are not very consistent across educators. There are several possible explanations for this. One explanation is that the educators are all correct, in their own context: each educator may be correct about their own classroom, but their opinions do not generalise to other classrooms. Further work would be needed to investigate this, either directly by getting educators to tag their own students in the data set, or indirectly by breaking down the mistakes by student and comparing the variance between students to the variance between educator ratings.

Another explanation for the lack of generalisability is that looking at frequencies of Java mistakes is the wrong level of abstraction to find common ground among educators. If educators talked about programming features at a higher level (e.g. the conceptual difficulties that students have with looping, or variables) then we might find much more agreement and consensus that looking at the lower level aspects of the frequency of specific syntax and semantic mistakes. There is a reasonable argument to be made that educators' knowledge of mistake frequencies is not important in understanding student difficulties. However, it remains the case that educators do not form a consensus. This may be because educators do not form a consistent memory of mistake frequency, or generalise inaccurately from their own experience to larger bodies of students.

5.2 Student Mistakes

Our results from the Blackbox data provide a large sample-size result for the frequency of different Java errors, distinct from compiler error messages (shown in Table 1). Mismatched brackets – a syntax error – was the most frequent type of error, but otherwise the semantic errors tended to appear higher than the syntax errors. We should not over-interpret this result, as it is greatly affected by which mistakes we included in the study – but we wonder if this hints that semantic errors are a more serious challenge than syntax errors. Recent research has suggested that Java has syntax which is not much better than an arbitrary choice of syntax for learning [18], but perhaps the syntax of a language does not matter as much as semantics. Further work could investigate whether the frequency of these mistakes change over time (i.e. do students make fewer syntactic mistakes over time?), but that is beyond the scope of this paper. We also note that some of the errors which we took from Hristova et al. [9] are very infrequent, so their list of mistakes may not be suitable when considering just the most frequent Java mistakes.

5.3 Educator-Student Accord

Our examination of educators' frequency rankings against frequency rankings from the Blackbox data set (termed *accord*) showed a significant but moderate correlation of $\rho = 0.514$. As previously mentioned, given the weak levels of consensus among educators, it would not have been possible to achieve very high levels of correlation[6]. Still, it means that for any given educator from our sample, the expected correlation to the real data would be 0.514, which is lower than we had expected. If an educator states that from their experience, students always make a certain mistake in Java, our results must suggest to be wary of such a claim.

A visual inspection of figure 5 shows the big "hits and misses" from the educators. Mistake *B*, the use of `==` to compare strings, was clearly overrated by most, as was mistake *E* (spurious semi-colons after if, for and while). Mistakes *J* (forgetting parentheses for a method call) and *Q* (incompatible types when assigning a method call result) were also overrated in terms of frequency. Of course, some of these mistakes are relative to the use of the underlying constructs: if students wrote fewer method calls than educators expected, this would explain the lower frequency of mistakes in making method calls. Similarly, the frequency of mistake R (not implementing all methods from an interface) will depend heavily on the frequency with which the sampled students implement interfaces (a relatively advanced feature for novices).

The mistakes in this study were automatically detected using a set of tools. If the tools were not sufficiently accurate in picking up mistakes, this could have affected our measure of educator accord. To this end, we have made the source code available (see section 3.3) in case further investigation is warranted. As well as the tool, the choice of mistakes has an effect on educator accord. Looking at Table 1, it is clear that some of the mistakes (especially *L*, *F*, *H* and *G*) are particularly low frequency, in contrast to Hristova et al.'s [9] aim to find the most common errors. Figure 5 confirms that, especially for *G*, educators predicted these low frequency items consistently – if *G* were excluded from the analysis, educator accord would reduce. As an illustration, if mistakes *L, F, H* and *G* are removed from the analysis, the average educator accord would drop from 0.514 to 0.311 – a much weaker level of accord. A more principled search for the top mistakes, and accurate classifiers for these, may provide a better picture of educator accord, but this may decrease educator accord rather than increase it. Educators seem to be accurate at identifying rare mistakes as rare but not at comparing higher frequency mistakes.

5.4 Affect of Experience on Educator-Student Accord

Our analysis, depicted visually in Figure 4, found that there was no effect of educator experience (as measured by years as an educator, years teaching introductory programming in any language, or years teaching introductory programming in Java) on educator accord. That is: no matter how many years the educator had been teaching, it made no effect on their accuracy in predicting the frequencies from the Blackbox student data. This is a very surprising result. Our expectation had been that experience would surely provide some sort of increase to accuracy.

The results show that experience has no effect on the task of rating mistakes by frequency. Further work could investigate more closely: for example, some tertiary educators lecture but do not supervise programming classes directly, so this could have an effect on accuracy. However, an initial analysis of our results suggests this may not be the issue: the average accord of educators who had taught in secondary education was 0.509, while the average accord of educators who had taught only in tertiary education was 0.516.

If one assumes that educator experience must make a difference to educator efficacy, then this would imply that ranking student mistakes is therefore unrelated to educator efficacy. However, work from Sadler et al. [15] in physics found that "a teacher's ability to identify students' most common wrong answer on multiple-choice items... is an additional measure of science teacher competence." While picking answers to a multiple choice question is not exactly the same as programming mistakes, there is a conflict here – either the Sadler et al. result does not transfer and ranking common student mistakes is not a measure of programming teacher competenece, or experience has no effect on teacher competence. The first option seems more likely.

6. CONCLUSIONS

Our study investigated educators' opinions about the frequency of eighteen mistakes in Java among programming novices. Our first finding was that educators have only a weak consensus about these frequencies. It remains possible that these frequencies are contextual, and thus each educator is correct for their own students. Regardless, this suggests that in cases where educators communicate with each other (e.g. via online communities), talking about which mistakes student make will not provide much agreement. Our further result that educators are not very accurate compared to a large data set of students suggests that educators are also not accurate about the frequencies of these mistakes, so any claims that "students always make mistake X" are unlikely to be accurate. Of course, this may just require that a different level of discourse is required: educators may still be accurate about the cause of mistakes and student's conceptions of the mistakes, but just not about the frequency of such mistakes.

Our most surprising result was that an educator's level of experience (as measured by years as an educator, years teaching introductory programming in any language, or years teaching introductory programming in Java) had no effect on how closely the educator's frequency rankings agreed with those from the Blackbox data. A strong interpretation of this result would be that experience has little effect on educator efficacy. However, it must be remembered that this task (ranking mistakes by frequency) is not necessarily aligned with educator efficacy – this result only shows that experience has no effect on ranking mistake frequency. Instead, this result may indicate that such a task is therefore not representative of educator efficacy (which we would expect to increase with experience).

Our data also provides frequencies from a large data set (over 100,000 students) for our chosen eighteen novice mistakes. Given the lack of agreement between educators and

[6] As an illustration, if the Blackbox data had had frequencies ranked in order of the average ranks assigned by the educators, the average accord would have been 0.692. Although this is not necessarily the mathematical maximum of the average correlation, it can be used as an informal indicator of the highest levels of accord that could have been achieved.

this data (see figure 5), we can therefore state that the results will be surprising to most (and that any educators who claim they are "exactly as expected" were not participants in our study!). Mismatched brackets – a syntax error – were the most frequent type of mistake, and otherwise many of the top mistakes were not syntactic. Notable mistakes that were mispredicted by our sample of educators (and thus probably the most surprising results) were as follows. Mistake B, the use of == to compare strings, was clearly overrated by most, as was mistake E (spurious semi-colons after if, for and while). Mistakes J (forgetting parentheses for a method call) and Q (incompatible types when assigning a method call result) were also overrated in terms of frequency. One hypothesis is that educators overrated in terms of frequency those mistakes which students have most trouble with understanding, or were likely to spend most time on fixing. We aim to investigate this possibility in future work.

Acknowledgements

The authors are grateful to Kristina Dietz for her help with the statistical analysis methodology, to Sally Fincher for her advice on the research design, to Davin McCall for his knowledge of the area, and to Michael Kölling and Ian Utting for their observations about untested educator folk wisdom that inspired the research.

7. REFERENCES

[1] M. Ahmadzadeh, D. Elliman, and C. Higgins. An analysis of patterns of debugging among novice computer science students. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '05, pages 84–88, New York, NY, USA, 2005. ACM.

[2] Y. Ben-David Kolikant. Computer science education as a cultural encounter: a socio-cultural framework for articulating teaching difficulties. *Instructional Science*, 39(4):543–559, 2011.

[3] N. C. C. Brown, M. Kölling, D. McCall, and I. Utting. Blackbox: A large scale repository of novice programmers' activity. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 223–228, New York, NY, USA, 2014. ACM.

[4] R. Cadenhead. *Sams Teach Yourself Java in 21 Days.* Pearson Education, 2012.

[5] P. Denny, A. Luxton-Reilly, and E. Tempero. All syntax errors are not equal. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '12, pages 75–80, New York, NY, USA, 2012. ACM.

[6] T. Dy and M. M. Rodrigo. A detector for non-literal Java errors. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, pages 118–122, New York, NY, USA, 2010. ACM.

[7] C. Hoisington. *Android Boot Camp for Developers using Java.* Cengage Learning, 2012.

[8] D. C. Howell. *Statistical Methods for Psychology.* Duxbury, fifth edition, 2002.

[9] M. Hristova, A. Misra, M. Rutter, and R. Mercuri. Identifying and correcting Java programming errors for introductory computer science students. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '03, pages 153–156, New York, NY, USA, 2003. ACM.

[10] J. Jackson, M. Cobb, and C. Carver. Identifying top Java errors for novice programmers. In *Frontiers in Education, 2005. FIE '05. Proceedings 35th Annual Conference*, Oct 2005.

[11] M. C. Jadud. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the Second International Workshop on Computing Education Research*, ICER '06, pages 73–84, New York, NY, USA, 2006. ACM.

[12] M. Kendall. *Rank Correlation Methods.* Edward Arnold, fifth edition, 1990.

[13] S. B. Lyerly. The average spearman rank correlation coefficient. *Psykometrika*, 17:421–428, 1952.

[14] D. McCall. Improving the experience of novice programmers – language and tools. Technical report, University of Kent, 2013.

[15] P. M. Sadler, G. Sonnert, H. P. Coyle, N. Cook-Smith, and J. L. Miller. The influence of teachers' knowledge on student learning in middle school physical science classrooms. *American Educational Research Journal*, 50(5):1020–1049, 2013.

[16] E. Soloway and S. Iyengar, editors. *Empirical Studies of Programmers: Papers Presented at the First Workshop on Empirical Studies of Programmers.* Intellect Books, 1986.

[17] J. C. Spohrer and E. Soloway. Novice mistakes: Are the folk wisdoms correct? *Commun. ACM*, 29(7):624–632, July 1986.

[18] A. Stefik and S. Siebert. An empirical investigation into programming language syntax. *Trans. Comput. Educ.*, 13(4):19:1–19:40, Nov. 2013.

[19] E. S. Tabanao, M. M. T. Rodrigo, and M. C. Jadud. Predicting at-risk novice Java programmers through the analysis of online protocols. In *Proceedings of the Seventh International Workshop on Computing Education Research*, ICER '11, pages 85–92, New York, NY, USA, 2011. ACM.

[20] W. L. Taylor. Correcting the average rank correlation coefficient for ties in rankings. *Journal of the American Statistical Association*, 59(307):872–876, 1964.

[21] W. L. Taylor and C. Fong. Some contributions to average rank correlation methods and to the distribution of the average rank correlation coefficient. *Journal of the American Statistical Association*, 58(303):756–769, 1963.

[22] P. van der Linden. *Just Java 2.* Pearson Education, 2004.