Geek Genes, Prior Knowledge, Stumbling Points and Learning Edge Momentum: Parts of the One Elephant?

Alireza Ahadi  
University of Technology, Sydney  
Australia  
Alireza.Ahadi@uts.edu.au

Raymond Lister  
University of Technology, Sydney  
Australia  
Raymond.Lister@uts.edu.au

ABSTRACT
Computing academics report bimodal grade distributions in their CS1 classes. Some academics believe that such a distribution is due to their being an innate talent for programming, a “geek gene”. Robins introduced the concept of learning edge momentum, which offers an alternative explanation for the purported bimodal grade distribution. In this paper, we analyze empirical data from a real introductory programming class, looking for evidence of geek genes, learning edge momentum and other possible factors.

Categories and Subject Descriptors
K.3.2 [Computing Milieux]: Computers and Education - Computer and Information Science Education

General Terms
Measurement, Human Factors.

Keywords
Learning edge momentum, programming, CS1, assessment, bimodal grade distribution.

1. INTRODUCTION
Some computing academics claim that students are either born with an innate ability to program, the “geek gene”, while other students are doomed to struggle with programming. As justification, those computing academics claim to see bimodal distributions in CS1 grades.

Robins (2010) provided an alternate explanation for why CS1 grades might show a bimodal distribution. Robins introduced the concept of learning edge momentum, which works as follows. Consider a CS1 course as consisting of a sequence of topics. Suppose that two students have an equal probability of learning the nth topic but only one of them successfully learns that topic. In Robins’ model, the successful student is then a little more likely than the other student to learn topic n+1. That is, the student who learnt topic n gains a little momentum, while the student who did not learn topic n loses a little momentum. Robins constructed a computer model and showed that his model leads to bimodal grade distributions. A crucial feature of Robins’ model is that all the simulated students begin with an equal probability of learning the first topic. Thus, Robins’ simulations demonstrate that a bimodal grade distribution is not necessarily proof for a geek gene.

Apart from geek genes and learning edge momentum, there are other possible explanations for the purported bimodal grade distribution. For example, perhaps some students enter CS1 with prior programming knowledge, or superior study skills. Another possibility is what we, the authors of this paper, call “stumbling points”. These are a small number of identifiable skills and concepts (perhaps but not always threshold concepts) that can have a major impact on a student’s progress.

These hypotheses – geek genes, prior knowledge, stumbling points and learning edge momentum – are not mutually exclusive. For example, perhaps prior knowledge produces a small difference in students at the start of semester, and that difference is then amplified by learning edge momentum.

These hypotheses place differing emphasis on different points of time and different periods of time. Geek genes and prior knowledge, of course, emphasize the time before the student commenced CS1. The distinction between stumbling points and learning edge momentum is more subtle, and overlapping. The idea of stumbling points is clear cut when there appear to be a small number of identifiable difficulties that students have with learning to program, whereas learning edge momentum is clear cut when there are a large number of subtle difficulties spread uniformly across a semester.

The over-arching discussion point of this paper is this: What types of empirical data, collected from real students, might distinguish between the competing hypotheses for the purported bimodal grade distribution? Unfortunately, we cannot answer that question in this paper (if we could, then this would not be a discussion paper). Instead, we will use data we have collected to highlight the difficulty of choosing between the competing hypotheses. Our data is from a real introductory programming class, collected from four tests held at weeks 3, 4, 6 and 7 of semester.

2. TESTS 1 & 2 (WEEKS 3 & 4)
The introductory programming course in which these tests were conducted had a two hour lecture in each of the 13 weeks of semester. Each week, commencing from week 2, there was also a two hour lab and a one hour recitation (known as a tutorial in some countries).

Students completed each of these tests at the start of a lecture session. Every test question, or where applicable every part of a question, was graded as being either right or wrong – no fractional points were awarded.
Completion of the tests did not contribute to a student’s final grade, and was voluntary. However, as the lecture did not proceed until a test was over, most students completed the tests. The time students took to complete a test was not formally recorded, but each test took around 15 minutes. Students were under little time pressure. The lecturer asked for a show of hands on who needed more time, and the tests usually did not finish until no student raised their hand. There was rarely an unanswered question in the students’ submissions.

2.1 Test 1 (Week 3)

When the students did test 1 in week 3, they had completed 4 hours of lectures, 2 hours of labs and 1 hour of recitation. The material in the week 3 test was taught in the previous week’s lecture, but at the time of the test the students had not done their recitation and lab supporting that week 2 lecture.

All the questions in test 1 were about assignment statements. The most complex code presented to the students comprised three assignment statements that swapped the values between two variables. The students were required to supply a total of nine answers, which were all marked as either right or wrong. Of the nine answers, 5 required the students to trace code, 3 required the students to explain code, and 1 required the students to write code. The actual questions are provided in the appendix, along with a breakdown of how well the class answered each of the nine parts. Figure 1 shows the distribution of student scores on test 1.

2.1.1 Discussion Points on Test 1

In Figure 1, the number of students peaks at two places, test 1 scores 2 and 9 – is it therefore a bimodal distribution? Either the distribution is bimodal or it is not. If it is bimodal, and if Figure 1 is indicative of CS1 classes in general, then the hypothesis of learning edge momentum is unnecessary – any bimodal distribution observed at the end of semester was simply there from the start of semester. Thus advocates for geek genes or prior knowledge might claim that Figure 1 supports their position.

Computing academic who claim that Figure 1 is NOT a bimodal distribution, but who claim to see a bimodal distribution in their end of semester grade distribution, are faced with a conundrum – how do they justify their claim that one distribution is bimodal, while the other is not? As Schilling, Watkins, and Watkins (2002) explain, it is risky to claim a bimodal distribution from merely observing two peaks. The authors of this paper believe that the distribution in Figure 1 could equally be characterized as a noisy uniform distribution with a ceiling effect – which leads the authors of this paper to wonder whether many alleged bimodal distributions of end-of-semester CS1 grades should also be classified that way.

Irrespective of whether or not the distribution in Figure 1 is bimodal, Figure 1 certainly does show a wide variation in student scores. If, as in Robins’ model, all our students had an equal chance of answering each of our questions correctly, we would expect to see a distribution of total scores showing a central tendency, which is clearly not the case. Figure 1 suggests that either (1) the students differentiated very, very quickly, or (2) the students were a heterogeneous group from the beginning. Two possible explanations for the large variation in week 3 scores are:

- Genetics: Whether any genetic difference relates to general intelligence, or a more specific “geek gene”, it seems unlikely that a college student needs to be especially bright or especially geeky to successfully reason about assignment statements.

- Prior knowledge: Approximately 30% of our students reported prior experience in programming, which may account for 39% of the class scoring 8 or 9 on test 1, but it does not account for the wide distribution of test scores for 7 or less.

2.2 Test 2 (Week 4)

The second test was conducted one week later. As with the first test, test 2 only used assignment statements. Test 2 was short, comprising only three questions with no subparts. The first question required students to trace some code which was (apart from changes to variable names and initial values) the same type of question as question 1(d) in the first test. The second question required students to write a swap of two variables, like question 3 in test 1. The third question was relatively novel, requiring students to write code for a swap-like process among four variables. The actual questions are provided in the appendix, along with a breakdown of how well the class answered each question. Figure 2 shows the distribution of student total scores on this second test.

3. COMPARING TEST 1 AND TEST 2 Q3

In test 2, the first two questions reprised questions from test 1. Only the third question involved a problem that students had not encountered before, and so this question was considered to be the question of most interest.

Figure 3 plots the probability of students answering test 2 question 3 correctly, versus their total score on test 1. The diameters of the circles in this figure reflect the number of students who attained those respective scores on test 1. For example, the largest circle represents the 24 students who attained a perfect 9 on test 1. The smaller circle to its left represents the 14 students who scored 8 on test 1. The two smallest circles each represent 4 students.
Advocates for **Learning Edge Momentum** will again point out that the probability of providing a correct answer to each of the questions from tests 3 and 4 rises as test 1 score rises. While the lines of best fit do not account for as much variance as before, that is to be expected with Learning Edge Momentum.

On Figure 4, advocates for the **Stumbling Point** hypothesis might perform the same type of statistical analysis as before (see Table 3) and see hints of a second jump in probabilities between test 1 scores of 4–6 and 7–8. However, they would also need to analyze test 1 and identify the qualitative difference between students who scored 4–6 and 7–8 on test 1. In Figure 5, there is not a statistically difference jump in probabilities between test 1 scores of 2–3 and 4–6 (see Table 4a), but perhaps by week 7 many students scoring 4–6 on test 1 may be floundering as much as most students who scored 2–3.

Table 1. The percentage of students who answered correctly the tracing parts of Test 1 Q1, broken down by total score. Each cell in the rows commencing “χ2, p =” show the probability of the two percentages above and below that cell NOT being statistically significant. Thus the greyed cells indicate statistically significant differences.

<table>
<thead>
<tr>
<th>Test 1 Total Score</th>
<th>N</th>
<th>Test 1 Question 1 Tracing Parts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>a</td>
</tr>
<tr>
<td>2–3</td>
<td>17</td>
<td>94%</td>
</tr>
<tr>
<td>χ2, p =</td>
<td></td>
<td>0.71</td>
</tr>
<tr>
<td>4–6</td>
<td>22</td>
<td>91%</td>
</tr>
<tr>
<td>χ2, p =</td>
<td></td>
<td>0.50</td>
</tr>
<tr>
<td>7–8</td>
<td>24</td>
<td>96%</td>
</tr>
</tbody>
</table>

Table 2. The contingency table for how students answered Test 2 Q3 vs. their total score on Test 1 (χ2 test, p < 0.01).

<table>
<thead>
<tr>
<th>Test 1 Total Score</th>
<th>N</th>
<th>Test 2 Question 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>wrong</td>
</tr>
<tr>
<td>2–3</td>
<td>17</td>
<td>12</td>
</tr>
<tr>
<td>4–6</td>
<td>22</td>
<td>6</td>
</tr>
</tbody>
</table>

4. **TEST 1 VERSUS TESTS 3 AND 4**

The advocates of the various hypotheses will each find comfort from a similar comparison of test 1 versus test 3 Q6(b) (see Figure 4) and also test 1 versus test 4 Q5 (see Figure 5):

Advocates for **geek genes or prior knowledge** will again point out that students who did very well (or very poorly) on test 1 tended to do well (or poorly) on each of the questions from tests 3 and 4.
Table 3. Two contingency tables for how students answered Test 3 Q6(b) vs. their Test 1 score. (Both χ² tests, p = 0.04).

<table>
<thead>
<tr>
<th>Test 1 Total Score</th>
<th>Test 3 Q 6(b) Total Score</th>
<th>Test 3 Q 6(b)</th>
<th>Test 3 Q 6(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>wrong</td>
<td>right</td>
<td>wrong</td>
</tr>
<tr>
<td>2 − 3</td>
<td>9</td>
<td>1</td>
<td>4 − 6</td>
</tr>
<tr>
<td>4 − 6</td>
<td>7</td>
<td>7</td>
<td>7 − 8</td>
</tr>
</tbody>
</table>

Figure 5. The relationship between student scores on test 1 and the probability of answering Test 4 Q5 correctly (N=63).

Table 4. The contingency tables for how students answered Test 4 Q5 vs. their Test 1 score.

<table>
<thead>
<tr>
<th>Test 1 Total Score</th>
<th>Test 3 Q 6(b) Total Score</th>
<th>Test 3 Q 6(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>wrong</td>
<td>right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4a. χ² test, p = 0.31

### Table 4b. χ² test, p = 0.03

5. TEST 2 VERSUS TESTS 3 AND 4

The advocates of Learning Edge Momentum will react enthusiastically to comparisons of test 2 versus test 3 Q6(b) (see Figure 6) and also test 2 versus test 4 Q5 (see Figure 7). The advocates of the other hypotheses may point out that test 2 only has three questions, and both Figures 6 and 7 only have three data points of non-trivial sample size. Thus, so long as the probability associated with a test 2 score of 2 falls somewhere in the middle, a line of best fit is always going to be a good fit. The advocates of Learning Edge Momentum might then concede that the R² values are not to be taken too seriously, but never the less in both Figures the probability values rise with test 2 score — and that supports Learning Edge Momentum.

6. GENERAL DISCUSSION

A curious issue is that tests 1 and 2 were all about assignment statements, but assignments statements are not used in the questions from tests 3 and 4 (at least not explicitly). We don’t know which hypotheses might be supported by that curious issue.

In an Indian folk story, blind men touch different parts of an elephant and describe the elephant in different ways. One blind man feels a leg and claims the elephant is like a pillar; while another feels the tail and claims the elephant is like a rope, and so on. Likewise, what we have demonstrated in this paper is that advocates of the various hypotheses – Geek Genes, Prior Knowledge, Stumbling Points and Learning Edge Momentum – can all find support for their respective hypotheses, in aspects of the data in this paper. What we need is someone with the vision required to generate a different type of data — a better way of studying the whole elephant. What is that data?

Some readers may believe it is futile to look for empirical data to distinguish between these hypotheses. Our first response to that is the old saying “If you can’t measure it, then it doesn’t exist”. Our second response is more specific to this context: “If you believe there is no empirical way of choosing between these competing hypotheses, then why believe in any one of them?”

7. ACKNOWLEDGMENTS

Support for this research was provided by the Office for Learning and Teaching, of the Australian Government Department of Industry, Innovation, Science, Research and Tertiary Education.

8. REFERENCES


9. **APPENDIX: THE FOUR TESTS**

9.1 Test 1 (week 3)

**Q1**

(a) In the boxes, write the values in the variables after the following code has been executed:

```java
int a = 1;
int b = 2;
a = 3;
```

The value in `a` is ___ and the value in `b` is ___

(b) In the boxes, write the values in the variables after the following code has been executed:

```java
int r = 2;
int s = 4;
r = s;
```

The value in `r` is ___ and the value in `s` is ___

(c) In the boxes, write the values in the variables after the following code has been executed:

```java
int p = 1;
int q = 8;
p = q;
p = q;
```

The value in `p` is ___ and the value in `q` is ___

(d) In the boxes, write the values in the variables after the following code has been executed:

```java
int x = 7;
int y = 5;
int z = 3;
x = y;
z = x;
y = z;
```

The value in `x` is ___ and `y` is ___ and `z` is ___

(e) In the boxes, write the values in the variables after the following code has been executed:

```java
int x = 7;
int y = 5;
int z = 0;
z = x;
x = y;
y = z;
```

The value in `x` is ___ and `y` is ___ and `z` is ___

(f) In part (e) above, what do you observe about the final values in `x` and `y`? Write your observation (in one sentence) in the box below.

**Sample answer:** The values in `x` and `y` were swapped.

**Q2**

The purpose of the following three lines of code is to swap the values in variables `a` and `b`, for any set of possible values stored in those variables. Assume that variables `a`, `b` and `c` have been declared and initialized.

```java
c = a;
a = b;
b = c;
```

(a) In one sentence that you should write in the box below, describe the purpose of the variable “c” in the above code.

**Sample answer:** It is used to hold a value temporarily.

(b) In one sentence that you should write in the box below, describe the purpose of the following three lines of code, for any set of possible initial integer values stored in those variables. Assume that variables `i`, `j` and `k` have been declared and initialized.

```java
j = i;
i = k;
k = j;
```

**Sample answer:** The code swaps the values in `i` and `k`.

**Q3**

Assume the variables `first` and `second` have been initialized. Write code to swap the values stored in `first` and `second`.

**Sample answer:**

```java
int temp = first;
first = second;
second = temp;
```

**Table A1. The percentage of students who answered correctly each question in test 1 (N=98)**

<table>
<thead>
<tr>
<th>Question (a)</th>
<th>1(a)</th>
<th>1(b)</th>
<th>1(c)</th>
<th>1(d)</th>
<th>1(e)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percent correct</td>
<td>92%</td>
<td>83%</td>
<td>66%</td>
<td>62%</td>
<td>55%</td>
</tr>
</tbody>
</table>

| Question (f) | 1(f) | 2(a) | 2(b) | 3 | — |
| Percent correct | 50%  | 63%  | 45%  | 52% | — |

9.2 Test 2 (week 4)

**Q1**

In the boxes, write the values in the variables after the following code has been executed:

```java
int a = 23;
int b = 11;
int c = 61;
a = b;
c = a;
b = c;
```

The value in `a` is ___ and `b` is ___ and `c` is ___

**Sample answer:** The values in `x` and `y` were swapped.
Q2
Assume the variables black and white have been initialized. Write code to swap the values stored in black and white.

**Sample answer:**
```java
int temp = black;
black = white;
white = temp;
```

Q3
Suppose there are four variables, a, b, c and d as depicted below:

```
 a   b   c   d
```

**Write code** to move the values stored in those variables to the left, with the leftmost element being moved to the rightmost position as depicted by this diagram:

```
 a   b   c   d
   temp
```

For example, if a=1, b=2, c=3 and d=4 (as shown above), then after your code is executed those variables should contain a=2, b=3, c=4 and d=1. (But your code should work for all possible values.)

**Sample answer:**
```java
temp = a
a = b
b = c
c = d
d = temp
```

Table A2. The percentage of students who answered correctly each question in test 2 (N=98)

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percent correct</td>
<td>86%</td>
<td>61%</td>
<td>64%</td>
</tr>
</tbody>
</table>

9.3 Test 3 (week 6) Question 6
Consider the following block of code, where variables a, b and c each store integer values:

```
if (a > b)
    if (b > c)
        System.out.println(c);
    else
        System.out.println(b);
else if (a > c)
    System.out.println(c);
else
    System.out.println(a);
```

(a) In relation to the above block of code, which one of the following values for the variables will cause the value in variable b to be printed? Draw a circle around the appropriate answer, (i), (ii), (iii) or (iv).

(i) a = 1; b = 2; c = 3;
(ii) a = 1; b = 3; c = 2;
(iii) a = 2; b = 1; c = 3;
(iv) a = 3; b = 2; c = 1;

(b) In one sentence that you should write in the box below, describe the purpose of the above block of if/else statements. Do NOT give a line-by-line description of what the code does. Instead, tell us the purpose of the code:

**Sample answer:** It prints the smallest value.

83% of 87 students answered test 3 question 6a correctly, while 57% of those students answered 6b correctly.

9.4 Test 4 (week 7) Question 5
In one sentence that you should write in the empty box below, describe the purpose of the following code. Do NOT give a line-by-line description of what the code does. Instead, tell us the purpose of the code. Assume that the variables y1, y2 and y3 are all variables with integer values. In each of the three boxes that contain sentences beginning with “Code to swap the values ...”, assume that appropriate code is provided instead of the box – do NOT write that code.

```
if (y1 < y2) {
    Code to swap y1 and y2 goes here.
}
if (y2 < y3) {
    Code to swap y2 and y3 goes here.
}
if (y1 < y2) {
    Code to swap y1 and y2 goes here.
}
```

48% of 63 students answered test 4 question 5 correctly.

**Sample answer:** It sorts the numbers into ascending order.

Note: Actually, two versions of this question were used. Around half the students answered another version where the numbers were sorted into descending order. The performance of students on both versions was not statistically different, so the data for the two versions was combined.