Test Anxiety and Online Testing: A Study

Amruth N. Kumar
Ramapo College of New Jersey
Mahwah, NJ 07430, USA
amruth@ramapo.edu

Abstract—Test anxiety is known to negatively affect test performance. Having students write about their testing worries before taking a test was recently shown to improve test performance by reducing test anxiety. We conducted a controlled study to replicate this result in the context of students using online Computer Science tutors unsupervised and on their own time. Instead of using open-ended expressive writing exercise, we used a multiple-choice questionnaire that addressed student anxiety. During the study, we collected data from two tutors on advanced programming concepts over three semesters. We did ANOVA analysis of the number of problems solved, score per problem and time spent per problem with treatment as the between-subjects factor. The test group solved significantly more problems and scored more points per problem than the control group on one of the two problems solved, but not the other. We discuss a possible explanation for the result and the significance of addressing test anxiety for broadening participation in Computer Science. (Abstract)

Keywords—Online testing; test anxiety; evaluation

I. INTRODUCTION

Empirical research has consistently shown a negative relation between anxiety and measures of learning [1]. Anxiety could be facilitating, i.e., task-directed, or debilitating, i.e., task-irrelevant [2]. Debilitating anxiety is known to negatively affect learning and cause poor performance. Hundreds of studies have been conducted on its causes and effects, and meta-studies have documented their aggregate results (e.g., [3]).

We revisited anxiety and its effect on test performance in the context of online software tutoring. Software tutors are increasingly being used to supplement classroom instruction – they are used to help students learn by solving problems after class, on their own time and at their own pace. They are gaining prominence in the context of flipped classrooms and MOOCs, where they provide the interaction and individual attention that are the strengths of traditional face-to-face classes.

We have been developing and deploying such software tutors for programming constructs, called problets (problets.org). These tutors have students solve problems on individual programming constructs and grade their answers. They also provide step-by-step explanation of the correct solution, which has been shown to help improve learning [4]. In order to maximize learning while minimizing the number of problems solved by the student, problets adapt to the learning needs of the student [5]. They do so by using a pre-test-practice-post-test protocol during every problem-solving session: students first solve a pre-test, which records the concepts students already know and those that they do not; students then proceed to practice solving problems on only the concepts that they do not yet know, followed by answering a post-test on the concepts they just practiced. So, the performance of the student on the pre-test determines how many problems the student ends up solving with the tutor and how long the student spends doing so. Given this setup, we wanted to eliminate confounding factors that might negatively affect the performance of the student on the pre-test and lead to the student solving unnecessary problems, e.g., in earlier studies, we found stereotype threat to be one such confounding factor [6, 7] and have since stopped asking for any demographic information before the pre-test. According to literature, test anxiety is another such confounding factor.

Numerous behavioral, cognitive, and cognitive-behavioral approaches to reduce test anxiety and improve test performance have been studied in the past and have been found to be effective [3]. In one study, when students were told to feel free to comment on items of a multiple-choice psychology test, they consistently scored higher than those who were not allowed or not asked to comment [8]. Instructions encouraging comments interacted with the level of anxiety and helped students who had high anxiety [9].

In a more recent study, simply writing about testing worries was found to improve test performance [10]. This was an adaptation of expressive writing, a practice in which subjects repeatedly write about a traumatic or emotional experience over several weeks or months in order to decrease rumination [11]. Expressive writing may ameliorate the demands placed on short term working memory by the worries generated by test anxiety [12], which in turn frees up working memory to focus on the task at hand, i.e., taking the test.

Expressive writing is particularly attractive for use in software tutors and online testing since administering it takes minimal time and no expert supervision. Given this, and our desire to eliminate test anxiety as a confounding factor, we proposed to study whether expressive writing would be beneficial to students using our software tutors.

However, this study differed from previous study [10] in several respects:
1. The previous study was conducted in-ovo, in a supervised classroom setting. Our study was conducted in-natura, i.e., students used the web-based software tutor usually on their own time, unsupervised, and as an assignment in the course. Since they received credit for completing the assignment rather than for their score on it, their incentive may have been to minimize the time they spent rather than maximize their score. However, these in-natura conditions are the norm rather than the exception for the use of software tutors, especially in higher education. Whether these conditions are a high-pressure environment is debatable. Given that expressive writing was not shown to affect performance in low-pressure situations [10], we expected this to be a confounding factor if they weren’t.

2. In the previous study, expressive writing was open-ended: students were allowed to write openly about their thoughts and feelings regarding the test. In an unsupervised setting, this may lead to uneven response – students may be tempted to get past the expressive writing exercise with a few random keystrokes, thereby defeating the spirit of the exercise. So, we chose a multiple-choice format instead: students were presented 6 mandatory questions and were asked to respond to each question on a three-point scale. This ensured a minimal amount of participation from all the subjects in the anxiety exercise. This is also a novel adaptation of expressive writing - whereas expressive writing was adapted earlier [10] from a weeks-long writing exercise to writing once, we adapted it further from writing about testing worries to selecting the most appropriate phrases in a multiple-choice questionnaire that describe the testing worries. The underlying hypothesis is that answering multiple-choice questions on test anxiety instead of writing about test anxiety is adequate for reducing rumination, and therefore, reducing the demands placed on short term working memory by the worries generated by test anxiety, thereby enabling better performance on the test. So, positive results in this study would support not only the beneficial effects of test anxiety treatment, but also that such treatment could be delivered in multiple-choice format.

3. We did not measure the level of test anxiety of the subjects. We studied the effect of expressive writing on the performance of students on a test, without using test-anxiety measure as a covariate. Given that expressive writing was not shown to affect the test performance of low-test-anxiety students [10], we expected this to confound the results of our study, i.e., possibly not find a significant effect for expressive writing because of low-test-anxiety students in our population.

II. Protocol

For this study, two tutors were used, both dealing with the concept of functions:

- A tutor on code tracing – students were asked to identify the output of programs involving one or more functions. The tutor covered 10 concepts, including parameter passing, calling functions in expressions, multiple calls to a function, and variables with the same name in multiple functions (See Fig. 1).

- A tutor on debugging – students were asked to identify bugs in programs involving one or more functions. The tutor covered 8-9 concepts (depending on the language – C++/Java/C#), including mismatch in the number and type of actual and formal parameters, missing return statement in a function, incorrectly calling a function that returns void versus non-void, and re-declaring a formal parameter as a local variable in a function (See Fig. 2).

Both these tutors are on advanced concepts that students in introductory programming courses find to be difficult.

We used a partial cross-over design for the study: institutions using the tutors were divided into two groups; students from each group served as control subjects on one tutor and test subjects on the other.

Each tutor was configured to administer the following protocol:

- Survey: During this stage, control group answered a multiple-choice questionnaire on the design of user interface, and test group answered a multiple-choice questionnaire that addressed their anxiety.

- Pre-test – During the pre-test, the tutor presented one problem per concept. If a student solved a problem correctly, the student was given credit for the corresponding concept. No feedback was provided to the student, and no more problems on the concept were presented to the student. On the other hand, if the student solved a problem incorrectly, feedback was presented to the student immediately after the student submitted his/her solution to the problem. Additional problems were presented on the concept during the subsequent stages described below.

- Adaptive practice – During this stage, additional problems were presented to the student on only the concepts on which the student made mistakes when solving problems during the pre-test. For each such concept, the student was presented multiple problems until the student mastered the concept, i.e., solved at least 60% of the problems correctly. On each problem, the student received feedback including step-by-step explanation of the correct answer, which has been shown to improve learning [4].

- Post-test - During this stage, the student was presented test problems on the concepts that the student had mastered during adaptive practice.

- Demographics - Students were provided the option to identify their demographic information, including sex and race. Demographic information was solicited after the pre-test-practice-post-test protocol to avoid the effects of stereotype threat [7].
The entire protocol was administered online, back-to-back, with no break in between, all by the software tutor. The pre-test-practice-post-test protocol was limited to 30 minutes.

During the survey stage, control group was asked to respond to the following mandatory questionnaire on the design of user interfaces:

1. In which font size do you prefer to read text?
   - Small font / Medium font / Large font
2. In which color do you like important text highlighted?
   - Red / Green / Blue
3. Do you like to see images on the screen?
   - No images / Some images / A lot of images
4. Do you like to see animations on the screen?
   - No animations / Some animations / A lot of animations
5. Do you think instructions should be provided for new software?
   - No / Sometimes / Always
6. How frequently do you like to take a break when working online?
   - Every 15-20 minutes / Every 30-40 minutes / Every 60-75 minutes

During the same survey stage, test group was asked to respond to the following questionnaire dealing with their anxiety about the topic:

1. How important is it for you to know this material?
   - Very important / Moderately important / Not important
2. How well do you know this material?
   - Very well / A little / Not at all.
3. Are you concerned about how well you know the material?
   - Very concerned / Somewhat concerned / Not concerned
4. You will solve problems next. Does solving problems help you learn?
   - No / May be / Yes
5. This tutor provides detailed feedback. Does such feedback help you learn?
   - No / May be / Yes
6. How do you hope to feel after using this tutor?
   - More knowledgeable / More confident / More comfortable

Note that the first question was on the importance of the topic, the second one on cognitive self-assessment; the third on self-efficacy; questions 4 and 5 on meta-cognitive processes about one’s own learning and the last question was designed to be an upbeat end to the survey, with all its answering options being positive.

III. DATA COLLECTION AND ANALYSIS

Data was collected over three semesters: fall 2012, spring 2013 and fall 2013. In each of these semesters, students from multiple institutions used the tutors over the web as shown in Table 1.

<table>
<thead>
<tr>
<th>TABLE I. NUMBER OF INSTITUTIONS THAT USED THE TUTORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracing</td>
</tr>
<tr>
<td>Debugging</td>
</tr>
</tbody>
</table>

Since this was a controlled study of the effect of using expressive writing on the test performance of students, only pre-test data was used for analysis. Only those students who had solved a majority of the pre-test problems were considered for analysis – at least 7 problems on code tracing tutor and at least 5 problems on debugging tutor. Furthermore, students who scored 100% on the pre-test were eliminated – they already knew the material, and did not need or learn from using the tutor. The number of students who remained in the control and test groups of the two tutors after these eliminations is shown in Table II.

In order to account for differences in the number of problems solved by students during pre-test, the score per problem was considered instead of the raw pre-test score. Similarly, the time spent per problem was considered instead of the total time spent on pre-test. We conducted univariate ANOVA analysis of the number of problems solved, the score per pre-test problem and the time spent per pre-test problem with treatment (user-interface versus anxiety) as the between-subjects factor.

<table>
<thead>
<tr>
<th>TABLE II. NUMBER OF STUDENTS IN THE STUDY</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
</tr>
<tr>
<td>Code-Tracing</td>
</tr>
<tr>
<td>Debugging</td>
</tr>
</tbody>
</table>

A. Code Tracing Tutor

We found a significant main effect for the number of pre-test problems solved \([F(1,220) = 4.67, p = 0.032]\): test group solved more pre-test problems than control group: 9.486 ± 0.128 problems by test group versus 9.188 ± 0.244 by control group.

We found a significant main effect for the pre-test score per problem \([F(1,220) = 5.137, p = 0.024]\): test group scored significantly more per pre-test problem than control group:
0.764 ± 0.03 by test group versus 0.693 ± 0.056 by control group. We did not find a significant effect for the time per pre-test problem \([F(1,220) = 0.033, p = 0.855]\): both the groups spent around the same amount of time per problem (83 – 85 seconds) on the pre-test.

B. Debugging Tutor

We did not find a significant effect for the number of problems solved \([F(1,304) = 0.001, p = 0.971]\): the two groups solved about the same number of pre-test problems (7.8).

We did not find a significant effect for the score per pre-test problem \([F(1,304) = 0.604, p = 0.438]\): the two groups scored about the same per problem (0.52-0.55).

We did not find a significant effect for the time spent per pre-test problem \([F(1,304) = 0.086, p = 0.770]\): the two groups spent about the same amount of time per problem (107.5-109.9 seconds).

IV. DISCUSSION

The results of the code-tracing tutor support the hypothesis that answering the questionnaire about their anxiety helped students solve more problems and score more per problem than the students who answered the questionnaire on user-interface. Whereas the results of debugging tutor do not directly support the same hypothesis, they do support that the two groups were comparable in their prior preparation. Given the partial crossover design of the study, the comparability of prior preparation of the two groups lends further credence to the results of the code tracing tutor, that the difference observed between the two groups was indeed because of the use of anxiety questionnaire.

Of course, the results of the study would have been more unequivocal and emphatic, had the test group students using debugging tutor scored more per problem than control group students. However, there may be an alternative explanation for why no difference was observed between the two groups. In prior studies, subjects fared worse when difficult items appeared early rather than appeared after easier items in a test (Lund, 1953 in [1]). Based on the facts that students took a lot longer to solve debugging problems (107.5-109.9 seconds) than code-tracing problems (83-85 seconds), and that both the groups scored far fewer points per problem on debugging tutor (0.52-0.55) than either group scored on code-tracing problems (minimum 0.693 points), we may conclude that problems presented by the debugging tutor were considerably harder than those presented by the code-tracing tutor. If a test begins with difficult questions, it may increase immediate anxiety, and negate any benefits of prior treatment.

So, while the results of this study are encouraging, they call for additional data collection and analysis to verify whether 1) expressive writing on anxiety in multiple-choice questionnaire format 2) presented before an online test/tutor 3) in in-natura environment helps students overcome their anxiety and improve their test performance.

While our original objective of this study was to eliminate factors confounding the use of our adaptive tutors, addressing test anxiety in online software will accrue additional benefits in Computer Science. Females showed higher levels of test anxiety than males, according to a meta-analysis of hundreds of research studies [3]. The sex difference was small in early school years, peaked in grades 5-10 and declined through college. Similarly, black students displayed higher anxiety than white students, although the difference was eliminated by high school. However, a significant difference was observed between Hispanic and white students across grade levels [3]. Given these results, addressing test anxiety in online resources for Computer Science is important for broadening participation of females and minorities in Computer Science.

Given the potential for broadening participation, and on the suggestion of one of the anonymous reviewers, we reanalyzed the data, taking into account sex and race as additional factors. Since Asians are positively stereotyped for quantitative disciplines such as Computer Science (e.g., [13]), we compared Caucasians and Asians versus underrepresented groups that included Black/African American, Hispanic/Latino, Native American, Native Hawaiian/Pacific Islander and Other designations.

On code-tracing tutor, we found marginally significant interaction between treatment and sex on the number of pre-test problems solved \([F(1,188) = 3.017, p = 0.084]\): male students in test group (N=109) solved more problems than those in control group (N=20) whereas female students in test group (N=43) solved fewer problems than those in control group (N=17), as shown in Table III. Similar marginally significant interaction was observed between treatment and sex on the pre-test score per problem \([F(1,188) = 3.525, p = 0.062]\): male students in test group scored more per problem than those in control group, whereas female students in control group scored more per problem than those in test group, as shown in Table IV. Similar interactions between treatment and sex were not observed on debugging tutor.

<table>
<thead>
<tr>
<th>Problems Solved</th>
<th>Control</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>8.995 ± 0.379</td>
<td>9.466 ± 0.189</td>
</tr>
<tr>
<td>Female</td>
<td>9.439 ± 0.409</td>
<td>9.328 ± 0.302</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Score per problem</th>
<th>Control</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>0.630 ± 0.089</td>
<td>0.735 ± 0.044</td>
</tr>
<tr>
<td>Female</td>
<td>0.759 ± 0.096</td>
<td>0.717 ± 0.071</td>
</tr>
</tbody>
</table>
On debugging tutor, we found significant interaction between treatment and race on the time spent per pre-test problem \[F(1,268) = 4.185, p = 0.042\]: Caucasians and Asians in test group (N=67) spent more time per problem than those in control group (N=150), whereas underrepresented minorities in control group (N=29) spent more time per problem than those in test group (N=23), as shown in Table V.

<table>
<thead>
<tr>
<th>Time per problem</th>
<th>Control</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caucasians+Asians</td>
<td>108.243 ± 13.477</td>
<td>112.522 ± 20.004</td>
</tr>
<tr>
<td>Under-represented</td>
<td>126.741 ± 32.003</td>
<td>78.662 ± 30.563</td>
</tr>
</tbody>
</table>

The interactions between treatment and sex on code-tracing tutor and the interaction between treatment and race on debugging tutor could point to differential effect of test anxiety treatment on the two sexes/racial groups. We plan to collect additional data to test this hypothesis.

ACKNOWLEDGMENT
Partial support for this work was provided by the National Science Foundation under grant DUE-0817187.

REFERENCES

Figure 1: Snapshot of Code Tracing Tutor in Java: Problem shown in the left panel and step-by-step explanation shown in the right panel

```java
// The Java program
public class Problem {
    public static void display(int principal) {
        principal = principal - 1;
        System.out.print(principal);
        System.out.print(principal * 2);
    }
}
```

**Explanation of the correct solution:**

The program is executed starting from line 13. The method `Problem.main()` on line 13 is executed. Since the parameter `principal` is passed by value, it's copied to the formal parameter `principal` in the method `display()`. The method `Problem.display()` on line 5 is executed, `principal` is decremented to 5 on line 7. The value of formal parameter `principal`, i.e., 5 is printed on line 8. The value of formal parameter `principal`, i.e., 10 is printed on line 10. The method `Problem.display()` is exited on line 11. The formal parameter `principal` goes out of scope and is deallocated on line 11.

Upon completion of execution of method `Problem.display()`, program resumes execution from line 16. The method `Problem.main()` is exited on line 16. This completes the execution of the program.

Figure 2: Snapshot of Debugging Tutor in C++: Problem shown in the left panel and explanation shown in the right panel

```cpp
// The C++ program
#include <iostream>
using namespace std;
void compute() {
    cout << 1;
}
int main() {
    cout << 5;
    cout << 7;
    compute();
    cout << count;
}
```

**Explanation of the correct solution:**

The definition of the function `compute()` requires no formal parameter. The function call on line 14 contains one actual parameter. This is a syntax error.