YSTATTEST: A SYSTEM FOR AUTOMATED ONLINE TEST CREATION AND CORRECTION

David G. Whiting and Del T. Scott
Brigham Young University, United States
whiting@byu.edu

Using exclusively open-source software, we have developed and implemented an online testing system dubbed YStatTest. *This system allows instructors to create homework, quizzes, and exams by querying a database containing question templates. A template is selected based on such elements as keywords, categories, and historical difficulty. Numerical values, data, and correct answers for each question template are randomly generated upon exam creation. Thus, although two students may receive identical templates, they will likely differ in placement on the exam and most assuredly in the data and numerical values associated with the question. We discuss the impact this testing system has had upon student learning and highlight future planned development for our software.*

INTRODUCTION

As professional teachers know, test preparation, correction, and evaluation make up a significant portion of our time. Conscientious teachers are further concerned about the quality of questions asked and methods for improvement in the evaluation of students' learning. Instructors often find themselves reusing questions or whole exams, with the attendant concerns about students accidentally or purposefully finding out questions before the exam date. In this light, test creation becomes daunting, especially when the exam is to be administered to hundreds or even thousands of students. (At BYU, enrollment in our introductory statistics course, Stat 221, totals approximately 2500 students per semester.)

With these concerns in mind, over the course of the past three years we have implemented a system, tentatively dubbed *YStatTest*. Our system generates unique exams for each student, creates a web form for them to enter their answers online, and corrects them automatically. An unintended consequence of the development of *YStatTest* is the ability to (1) let students create practice exams on their own, (2) let instructors use exam questions as homework problems, and (3) let students or instructors generate quizzes and make-up exams on demand.

BUILDING THE YSTATTEST SYSTEM

The *YStatTest* system is built exclusively using open-source software. A MySQL database is used for storage and retrieval of question templates, correct answers, student answers, and other information needed for the maintenance of the system. The R package for statistical analysis is used to generate data, numerical values and correct answers for each exam. Combined with the typesetting language LaTeX, a professional quality pdf file of the exam is created. Additionally, R is used for automated correction of numeric, vector, matrix, or string answers. Finally, the html scripting language php is used to build online forms for question writing, test creation, answer entry, test correction, and other administrative tasks dealing with class information and grading.

Integrating R and LaTeX

The Sweave package, available for R, allows for the integration of R and LaTeX. Base R and LaTeX code is placed in a file with extension .Rtex, using the Scode environment and the \Sexpr command in LaTeX. Issuing the R command (in a Unix environment)

```
echo 'library(tools); Sweave(test.Rtex)|R --no-save --no-restore'
```

creates the file test.tex which can be run through LaTeX. Figure 1 shows sample code for a question template selected from the database which would be included in a hypothetical test.Rtex file. Figure 2 shows the resultant test.tex file after R has generated numeric values and the correct answer. Note that some type of an enumerate environment would be required in LaTeX in order for the code snippet in Figure 2 to be correctly typeset.

```
\item A thousand tickets
  \begin{Scode}{echo=F}
    a <- rep(0,3)
    n <- c(1,5,50)
    a[1] <- round(runif(1,500,3000),-2)
    a[2] <- round(runif(1,40,200),-1)
    a[3] <- round(runif(1,10,20))
    dollar <- round(runif(1,3,10))
    answer <- sum((n*a)/1000)-dollar
  \end{Scode}
  are sold in a lottery in which there is one top prize of
  \$\Sexpr{a[1]}, five prizes of \$\Sexpr{a[2]}, and fifty prizes
  of \$\Sexpr{a[3]}. A ticket costs \$\Sexpr{dollar}.00. If $X$ is the
  player's net gain in dollars, find the expected value of $X$.
\begin{comment}
    \begin{Scode}{echo=F}
      answer
    \end{Scode}
\end{comment}
```

Figure 1: A sample question template in .Rtex format.

```
\item A thousand tickets are sold in a lottery in which there is one
  top prize of \$1000, five prizes of \$80, and fifty prizes of \$16.
  A ticket costs \$9.00. If $X$ is the player's net gain in dollars,
  find the expected value of $X$.
\begin{comment}
    \begin{Schunk}
      \begin{Soutput}
        -6.8
      \end{Soutput}
    \end{Schunk}
\end{comment}
```

Figure 2: The Figure 1 question after being run through R.

The MySQL Database

The MySQL *YStatTest* database contains a number of linked tables that constitute the internal engine of the testing system. Here we discuss only a few of the more important tables.

- The *question* table contains fields for an id, the question template, and epsilon, the amount a student's answer can deviate from the true answer and still be deemed correct. Supplemental information about the question template, e.g., subject matter, author, difficulty level, key words, etc. are stored in additional tables and referenced via the id variable.
- The *class* table contains information on question availability (some questions may not be relevant for certain classes), which test the question covers, and question difficulty, a historic measure which obviously differs for diverse classes, e.g., advanced vs. introductory courses.
- The eponymous *correct_answer* and *my_answer* tables are populated by the R-generated answers and the students' input answers, respectively.

INDIVIDUAL TEST CREATION

Before being included in an exam, each question template is vetted on three conditions: (1) the R code must work, (2) the LaTeX code must be error free, and (3) the computation must be correct. While the first two are done by the *YStatTest* system automatically, the third must be checked manually by the instructor.

Test creation rules are simply SQL queries set up by the instructor, based on fields in the class and related tables. Questions can be selected based on test number, difficulty level, class identification, keywords, and subject matter. It is thus easy to define a test where, e.g., three questions are about a normal distribution, two questions deal with a binomial, and five questions address either confidence intervals or hypothesis tests, where at least four of the questions have a difficulty greater than 0.6 (on a zero to one scale). Queries can, of course, be as complex as

needed. Thus, some understanding of SQL syntax is required. However, we have created a php web page which greatly simplifies the creation of these queries (see Figure 3).

Each test begins with the generation of a random seed and the random selection of questions matching the SQL query criteria. Enough information is saved in the database, as noted above, to allow for the exact regeneration of an exam. This is especially useful when a question template is found to produce a wrong result. After the template is corrected, all tests are rebuilt and once again corrected. Note that these steps have no affect on the student answers contained in the *my_answer* table.

TEST TAKING AND CORRECTION

Student exams are highly individualized, in that they have a unique combination of questions with unique numerical values and data. Students log into the system via a web browser to take the exam. Depending on the test format, the student may be allowed to print off a pdf copy of the test. Answers to individual questions are entered in an online form that is accessible only to that student. Student answers within a specified epsilon value are deemed correct. Experience has shown us that this is a good way to administer a test, even if it is a "take home" exam.

Upon test submission, the test may be automatically corrected and the results immediately made available to the student, or the instructor may opt to batch correct all tests. A *Test Status* form is available to the instructor, which allows him or her to see which students have taken the test, when they took it, whether it has been corrected, and their score. In the *Question View*, answers may be looked at individually by students (see Figure 4a) or by question. With this form a question may be completely dropped from the test (of those students who received it). Additionally, points may be manually entered by the instructor, e.g., for partial credit. We have also created online pages for *By Question* analysis and *Summary Analysis*, which allow the instructor to update question difficulties for their class (as in Figure 4b).

EXPERIENCES

We have found that test creation has, to a great extent, become a synergistic endeavor among instructors—questions added to the database are potentially available for any class. This has vastly increased both the type and the amount of questions accessible to those of us using the *YStatTest* system. The database currently contains over 800 question templates. Hence, our biggest technical issue at the moment is in creating a user interface to easily sort and choose the question templates without relying solely on SQL syntax. An eBay-like interface is being developed which would allow searches and checked boxes to specify which questions are to be used for an exam.

The biggest change, from an instructor's point-of-view, is that writing a test is a matter of selecting types of questions or topics to be covered, rather than worrying about the minutia of test creation. In other words, we spend more time thinking about the *content* of an exam than we could have before. Furthermore, because test creation and correction is so efficient, it has allowed a shift from one or two exams in a semester to four or six or even weekly exams, a situation that is virtually impossible without such a system. This system gives the instructor almost immediate feedback on the concepts students are learning and those that they are struggling with. Thus, our instructors using the testing system have been able to quickly adapt their teaching to the needs of the students, especially in devoting more time to reviewing or covering in further detail concepts deemed difficult by the students.

From a student's perspective, the online exams have forced an increase in learning. First and foremost, students can print off their own practice exams with correct answers supplied directly under each question. This gives them fair and honest study with questions comparable (and at times, perhaps similar) to what they would see on an exam. Secondly, the opportunity to cheat, either overtly or inadvertently, is essentially completely removed.

CONCLUSIONS

Ironically, this project started over three years ago as an effort to decrease the amount of time spent writing exams. However, through its development we have noticed the positive impact

it has had on teaching and learning in our statistics curriculum, and it has become a labor of love to continue with its development.

Because the components of our system are based on open-source software, we plan to continue in that vein. Our most recent efforts have been in making the system more bulletproof and user-friendly. Currently, we plan to release a beta version sometime in the coming year. In the long term, we can envision an online statistics question database with contributed questions and programmers more expert than us contributing their skills. Imagine a world-wide database of statistics questions!

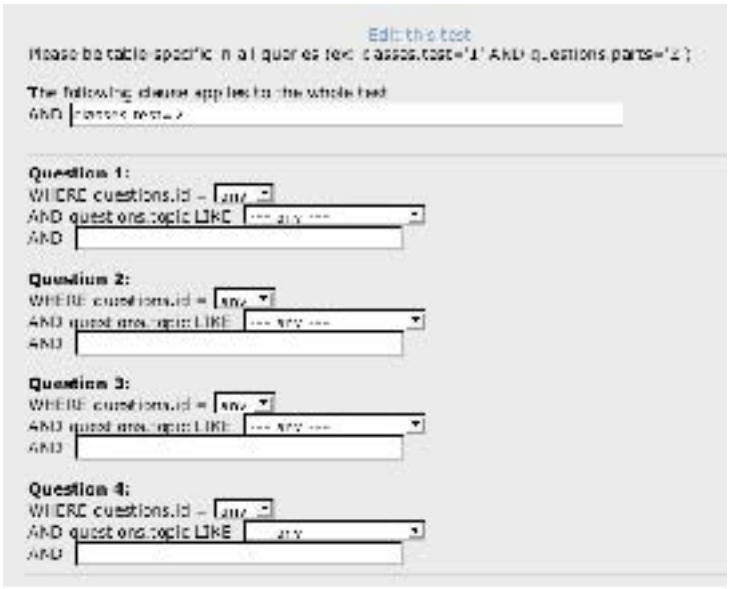

Figure 3. A php form for building SQL select statements for *YStatTest* test creation

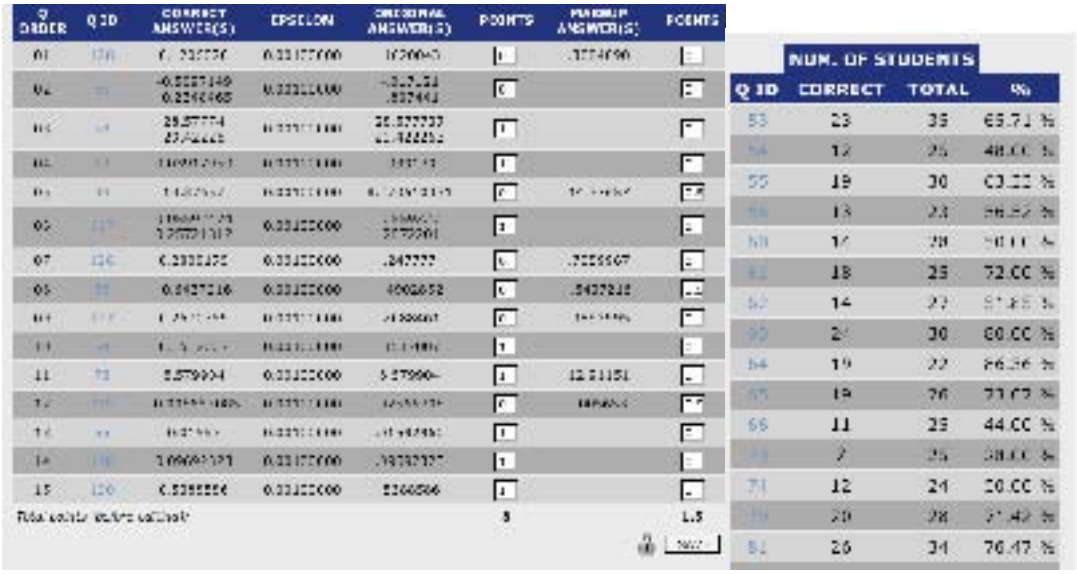

Figure 4. A test correction “by student” page and a simple question difficulty web page