`randexam`: Randomized Exam Generation and Grading

User Manual

Version 1.13.0

2015-10-07

1 Overview

The `randexam` program allows the generation of individually-randomized exams for every student. The process starts with a *library* latex file that contains a sequence of *zones*. Each zone contains a sequence of *questions*, and each question contains one or more *variants*. If the question is a multiple-choice question, then each variant has a set of *answers*, exactly one of which is the *correct answer*.

Each randomized exam has the same set of zones as the library, and in the same order. Within each zone, the order of questions is randomized. For each question, one of the variants is selected at random, and the order of the answers is also randomized.

There are two important identifiers used by `randexam`: the *NetID* identifies students and the *exam key* identifies exams. The exam key is a string of the form `BAECCD`, where the number of letters depends on the number of randomized exams generated. The exam key uniquely identifies the randomized exam, and so it is written on the front of the exam booklet and must be copied by the student onto the Scantron sheet, where it is encoded into the last few questions on the Scantron.

The exam keys for a particular exam are generated so that they all differ by at least three letters. This means that if a student makes an error in copying their exam key, they won't accidentally produce another valid exam key. In particular, if they make only a single-letter error, then it is possible to correct this automatically, and mis-copying two letters can be detected but not automatically corrected. If they mis-copy three or more letters, however, then they may produce another valid key, and this can only be detected by observing that their score was very low (effectively random guessing). In practice this has never occurred.

2 Preparing the exams

1. Obtain a copy of the three files: `randexam`, `config.ini`, and `library.tex`.
2. On Mac or Linux, make sure `randexam` is executable, or make it so with `chmod +x randexam`.
3. On Windows, it may be necessary to rename `randexam` to `randexam.py` and to run it with `python randexam.py <arguments>` rather than `./randexam <arguments>`.
4. Set the `exam_title` in `config.ini`.

5. Set the `filename_prefix` in `config.ini` to `TAM212_F14_Midterm2` or similar, and correspondingly rename `library.tex` to `TAM212_F14_Midterm2_library.tex`. Hereafter we write `XXX` in place of the prefix, so this file is referred to as `XXX_library.tex`.

6. Edit `XXX_library.tex` to add questions and make the cover sheet. This file can be latexed to produce the library PDF file that contains all variants of all questions.

7. Run `./randexam proc-lib`. This will produce the output tex file as well as various files summarizing the randomization, question points, and so on.

8. Run `pdflatex XXX_exams.tex`, and look at the output near the end for information about exam lengths.

9. To ensure all exams have the same number of pages and the last page is blank, adjust `minimum_pages_per_exam` in `randexam` to be the smallest even number that is at least $N + 1$, where N is the maximum raw length reported by latex when processing `XXX_exams.tex`. Then re-run `./randexam proc-lib` and re-latex.

3 Tips for exam preparation

1. Try to have about three variants of each question. The variants should be of equal difficulty and should only differ in some unimportant way. For example, reverse the orientation of a figure, change the value of one number, etc.

2. Zones serve two purposes: (1) to group questions so that random permutations only occur within a zone, not between zones (allowing the exam to start with 5 easy questions, for example); and (2) to insert text at a certain point in the exam. Zones do not need to contain any questions, so they can be used just to insert text, for example a formula sheet at the start or end of the exam.

3. Points are assigned on a per-question basis, so all variants of a given question are worth the same number of points.

4. Only one answer can be marked as correct for each variant. This answer will be awarded the full points for that question, and all other answers will be worth zero points. This can be adjusted by manually editing `XXX_points.csv`, which allows points to be assigned to any answer of any variant in an arbitrary way, including negative points and real-valued points.

5. Questions do not have any text associated with them directly. If there is text that is common to all variants of a question then it must be repeated in each variant.

6. Variants are not required to have any answers. This is useful for including questions that the students must answer via some other means, such as a long-form written answer.

7. Using PGFPlots for graphics (based on PGF/TikZ) can result in very long compile times and errors like `TeX capacity exceeded, sorry`. To avoid this, use the PGFPlots `external` library. To do this:

   (a) Include the following lines in the library file preamble:

   ```
   \usepgfplotslibrary{external}
   \tikzexternalize
   ```

   (b) Use `\tikzsetnextfilename` to assign a unique filename for each `tikzpicture`, such as:

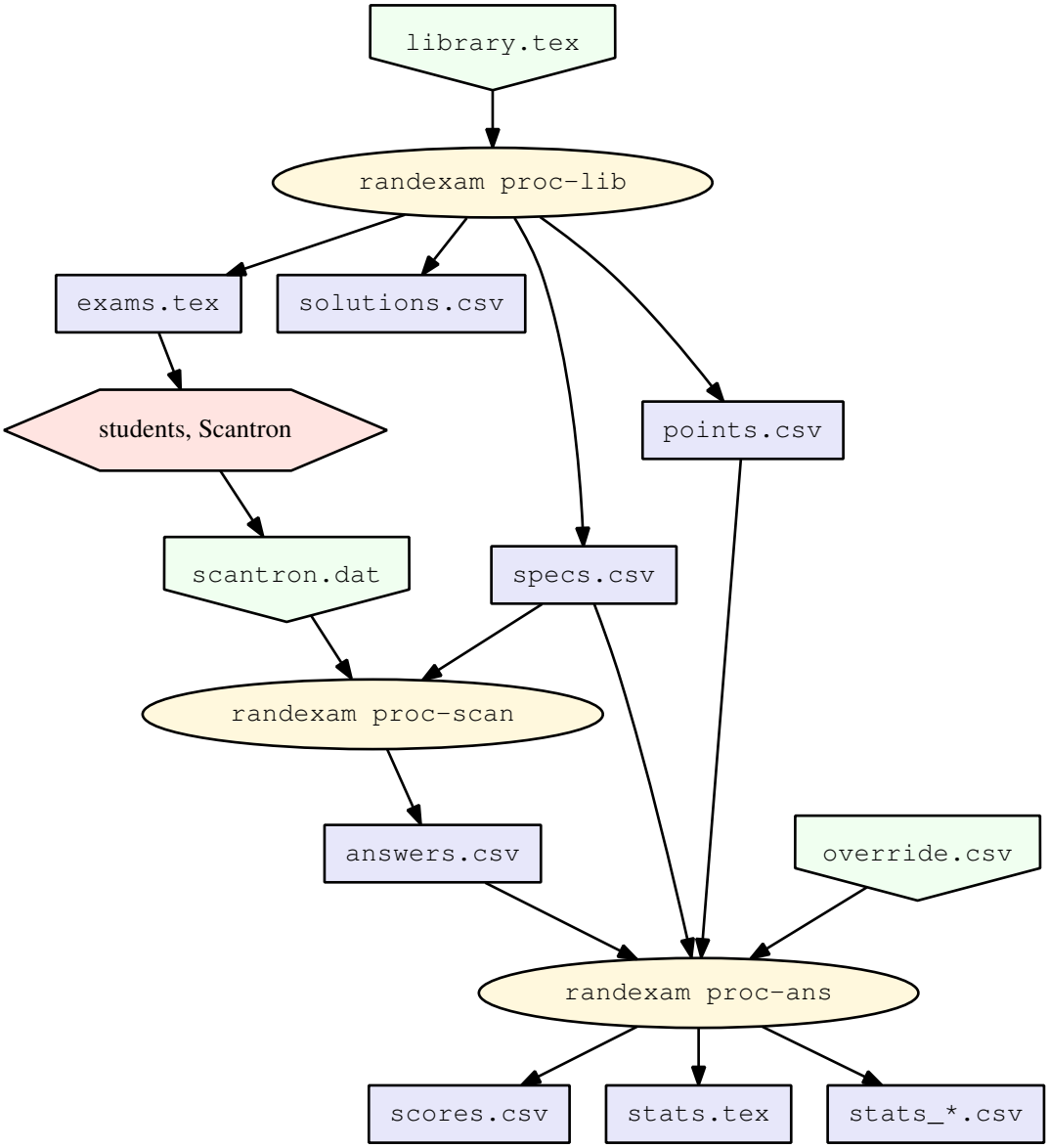

Figure 1: The pipeline for making and grading the randomized exams. The green pentagons are input files that require human effort to generate, the purple boxes are output files, the yellow ellipses are program runs, and the pink hexagon is the students sitting the exam and the scanning of the Scantron forms.

```
\tikzsetnextfilename{tikz_figname}
\begin{tikzpicture}
...
\end{tikzpicture}
```

(c) Generate library and exam PDFs with:

```
pdflatex -shell-escape filename.tex
```

(d) To generate student feedback, all the TikZ files must be in the current directory, so copy them over if they are somewhere else. This is both because latex isn't being run with `-shell-escape` when generating feedback, and because the use of `-output-directory` is incompatible with the TikZ external file support.

4 Configuration Options in `config.ini`

`answers_per_question` (default: `5`) Number of possible answers per question on the Scantron form. That is, the number of Scantron bubbles for each question.

`check_repeated_exam_keys` (default: `Yes`) Check for repeated use of each exam key, and report when this happens. If you are using individualized exams then this should not occur.

`check_valid_netids` (default: `Yes`) Check whether all students taking the exam have NetIDs listed in the `XXX_netids.txt` file.

`curve_scores` (default: `No`) Whether to transform the final exam scores by a curving function.

`curve_new_midpoint` (default: `80`) If curving is enabled, then this is the new midpoint score of the curved scores. This is normally set to the desired value of the median.

`curve_new_zero` (default: `20`) If curving is enabled, then this is the new curved score than an uncurved value of zero will map to.

`curve_old_midpoint` (default: `80`) If curving is enabled, then this is the old midpoint score of the curved scores. This is normally set to the median value of the uncurved scores.

`exam_title` (default: `Exam`) Title of the exam (used in statistics reports).

`feedback_directory` (default: `feedback`) Directory name where feedback PDFs are stored.

`feedback_solutions` (default: `Yes`) Whether per-student feedback should include full solutions to all questions.

`filename_prefix` (default: *blank*) Prefix to append to all input and output filenames. This is the `XXX` in filenames like `XXX_library.tex`. The filename_prefix should uniquely specify the exam. For example, `TAM212_S13_Final` would indicate the final exam in Spring 2013 for the course TAM 212.

`last_scantron_question_number` (default: `96`) Number of the last question on the Scantron form. This is used to determine the questions that will store the exam key.

`mail_domain` (default: `illinois.edu`) Domain name to append to student NetID to produce email addresses.

`mail_max_attempts` (default: `5`) Maximum number of times to retry sending each feedback email before giving up.

`mail_max_per_second` (default: `1`) Maximum email sending rate, in emails per second.

`mail_message_text` (default: `The attached file contains your test results.`) Text to use in the feedback email.

`mail_retry_delay_seconds` (default: `30`) Number of seconds to delay between retries of failed feedback email sending.

`mail_server` (default: `smtp.illinois.edu`) SMTP mail server to use for outgoing feedback emails. Connections will be made on port 587 using TLS and authenticated with the provided username/password.

`mail_signature_file` (default: `~/.signature`) Signature file to include at the end of feedback email messages, if the file exists.

`mail_timeout_seconds` (default: `120`) Timeout for all mail operations (connecting, sending, etc). Operations that take longer than this will be considered an error.

`max_answers_per_question` (default: `3`) Maximum number of answers for a question to still receive some partial credit. Only used if `multiple_answers_per_question` is `Yes`.

`minimum_pages_per_exam` (default: `2`) Length that every exam will be padded out to. Should be the smallest even number at least equal to $N+1$, where N is the maximum raw length reported by latex when processing `XXX_exams.tex`.

`multiple_answers_per_question` (default: `No`) Whether multiple answers per question are permitted (if more than one answer is given, then partial credit is awarded in inverse proportion to the number of given answers).

`number_of_exams` (default: `1`) Number of randomized exam papers to generate.

`one_page_per_question` (default: `No`) Whether the randomized exams should be formatted with every question on a different page.

`random_seed` (default: `7`) Seed value for the random number generator that creates the randomized exams. Random numbers are only generated during `randexam proc-lib`. The same value of random_seed will result in the same randomized exam generation, assuming the same version of Python on the same computer.

`randomize_answers` (default: `Yes`) If `Yes` then choose answer order randomly, otherwise leave answers in the library order.

`randomize_questions` (default: `Yes`) If `Yes`, then choose the question order randomly within zones, otherwise leave questions in the library order.

`randomize_variants` (default: `Yes`) If `Yes`, then choose question variants randomly. If `No`, then choose question variants in order, so exam 1 has variant 1 of each question, exam 2 has variant 2 of each question, etc. After exhausting all variants of a question, the selection will cycle back to variant 1.

`raw_scan_directory` (default: *blank*) Directory name where image scans of the Scantrons forms are stored. If blank, then these are ignored. If non-blank, these scans are attached to feedback emails.

`score_decimals` (default: 1) The number of decimal places to which total scores are truncated.

5 Administering the exams

1. Send the `XXX_exams.pdf` file containing all the exams to Document Services, via a department secretary. The instructions should be something like:

   ```
   Printing type:  double-sided, staple every 11 sheets of paper (that is, every
   22 PDF pages) to form 700 total exam papers
   Deliver to:  MEB 160 by Thursday morning (Oct 16)
   ```

   Here the number 22 is the value of `minimum_pages_per_exam` and 11 is half of this value. In the case the PDF file should have $22 \times 700 = 15\,400$ total PDF pages.

2. Obtain Scantron sheets from CITL Exam Services (Armory 247). These are free.

3. Hand out the exam booklets and Scantron sheets to students. Instruct them to fill in their information on the Scantron sheet and on the exam booklet. They should pay special attention to completing their NetID on the Scantron and correctly encoding the exam key from the exam booklet onto the last few questions on the Scantron.

4. Have the students do the exam.

5. Collect both the Scantron sheets and the exam papers (with names on them) from the students, so that exam papers can be later matched to students if needed. Ensure that the NetID and Exam Key (the last few questions on the Scantron) are filled in for every student.

6. Arrange the Scantron forms so that the cut corners are all aligned.

7. Take the Scantrons to CITL Exam Services (Armory 247). Tell them that you want the "dot dat" file sent to you. This refers to a filename like `jd17699.dat` that contains the raw scantron data.

6 DRES students

1. Inform DRES students before the exam that they should schedule their exams at the DRES facility. Tell DRES that the allowed window for the exam is from the afternoon on the day of the exam to the evening of the day following.

2. Extract individual exam PDFs from the large `XXX_exams.pdf` file and call these `DRES1.pdf`, `DRES2.pdf`, etc. This can be done using Adobe Acrobat Pro, or on Mac OS X using Preview and Print to PDF.

3. Email these individual PDFs to DRES and tell DRES that the students need to use an orange 96-question Scantron form, and that you want both the exam paper and the Scantron form returned to you.

7 Conflict exams

1. Schedule conflict exams starting 2 hours before the real exam (assuming a 2-h exam) until the afternoon of the following day. A good timeslot is 7 am the morning after the real exam, which ensures that only students who really need the conflict will use it.

2. Conflict exams can use the same exam as the real exam, as we rely on the randomization to make cheating difficult.

3. Conflict exams should be administered using printed copies of the exam from the main printing.

8 Grading the exams

1. Obtain the `jdXXXX.dat` file from Exam Services and rename it to `XXX_scantron.dat` or similar.

2. Extra exams, such as DRES or conflicts, can be hand-entered into the `scantron.dat` file. The format of lines in this file is described in detail in `randexam-dev-manual.pdf`.

3. By default, `randexam` will look for a file called `XXX_netids.txt` in the current directory. This should contain the class roster, listed as one NetID per line, and is used to check validity of the NetIDs in `scantron.dat`. If this checking is not desired then set `check_valid_netids: No` in `randexam`.

4. Run `./randexam proc-scan` and clean up errors in the `scantron.dat` file as needed. It is essential that every student has a correct exam key for grading to occur, and their NetID should also be correct for email sending and gradebook uploads. Other data (name, student ID, etc) is only for informational and checking purposes.

5. If desired, hand-edit the `XXX_points.csv` file to change the number of points awarded for each variant of each question. This is a good way to give all students full points for a given question, if a question was badly worded or incorrect.

6. Run `./randexam proc-ans` to perform the actual grading. This will generate exam statistics in tex file and the final scores in the files `XXX_scores.csv` and `XXX_gradebook.csv` (two different formats of the same information).

7. Run `pdflatex XXX_stats.tex` to generate a PDF of the exam statistics.

9 Curving exam scores

1. If score curving is enabled (by the `curve_scores` configuration parameter) then the final student scores are transformed using the piecewise linear function shown in Figure 2 with the following parameters:

quantity	config option	description
M_0	`curve_old_midpoint`	midpoint of old (uncurved) scores
M_1	`curve_new_midpoint`	desired midpoint of new (curved) scores
Z_1	`curve_new_zero`	desired new (curved) score for old zero score

2. While there is no requirement to do so, it is a good choice to use the median of the uncurved scores for the old midpoint, and the desired median of the curved scores for the new midpoint.

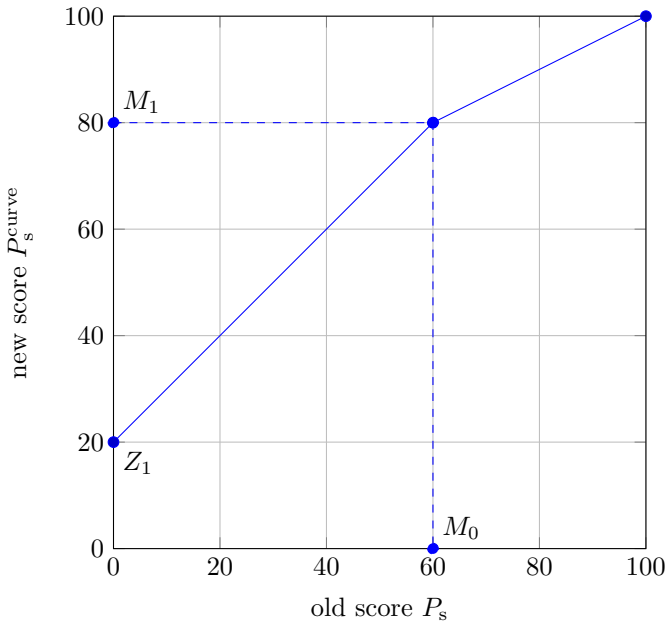

Figure 2: Curving function that transforms old scores to new scores.

10 Allowing partial credit for multiple answers

1. If desired, `randexam` can allow students to fill in multiple answers for a single question, and then give partial credit for that question if one of the provided answers was correct. To achieve this, several changes need to be made to the exam process, as follows.

2. When processing the Scantrons at CITL Exam Services, they need to scan the exam in a "multiple answer" format. This alters the `.dat` file format that they return.

3. Before running `./randexam proc-scan`, edit `config.ini` and set `multiple_answers_per_question: Yes`.

4. If multiple answers are given for a question, then the student is given partial credit if one of them is correct. The amount of credit given is inversely proportional to the number of answers they gave, so half credit for two answers, one-third credit for three answers, etc. In all cases, if the correct answer was not one of the given answers then no points are given. The maximum number of answers for which partial credit will be awarded is controlled by the `max_answers_per_question` variable.

11 Overriding question scores

1. For questions that must be hand-graded, or for questions where the automatically computed scores are incorrect for some reason, it can be necessary to manually override the scores.

2. To override the scores for some students on some questions, create a file called `<filename_prefix>_override.csv` in the same directory as `randexam` and enter the override scores. If present, this file will be automatically read by `randexam` whenever scores are being computed (e.g., with the `proc-ans` command).

3. The override file should be formatted as a CSV file with the first row being a header and then having one row per student. The first column must be the student NetID (capitalized) and then there is one column per question. The question number is given in the first row of the column, and then later rows specify the override scores for each student. Any students or questions not listed will not be overridden. An entry of `-1` means to not override that score, while any non-negative number is an override.

4. For example, consider the override file:

```
NetID,8,13
RDEVILLE,0.4,7
MWEST,0,-1
```

   Here two students have override scores specified for two questions. Student `RDEVILLE` will have a score of 0.4 for library question $Q = 8$ and a score of 7 for question $Q = 13$. Student `MWEST` will have a score of zero for question $Q = 8$, but for question $Q = 13$ his score will not be overridden, meaning it will be computed from the answer and points data as usual.

12 Interpreting the exam statistics

1. The `XXX_stats.pdf` file contains summary statistics for the exam and questions.

2. Check that the median score is reasonable (typically around 80 to have half the class in the A/B range), and curve the scores if desired.

3. Check that all question have variants that are fairly similar to each other in relative points. If R_{QV} is less that about 80% or more than about 120%, then investigate why some variants are significantly different and consider awarding all students full points for that question.

4. Check that all questions have discrimination values of at least 20% (higher is better). Very easy questions (difficulty below 10%) can legitimately have low discrimination without this being a problem. If a question has difficulty over 10% and discrimination below 20%, however, then investigate to see whether the question is poorly worded or incorrect, and consider awarding all students full points for that question.

13 Uploading scores to Compass

1. Go to `Grade Center` → `Full Grade Center` in the left-hand menu.

2. Back up the current data with `Work Offline` → `Download`.

3. Edit the `XXX_gradebook.csv` file generated by `randexam` to add a new first line consisting of:

```
"Username","Test Name"
```

   where `Test Name` is replaced by something like `Midterm 2` or `Final`.

4. Go to `Work Offline` → `Upload` and upload the edited `XXX_gradebook.csv` file.

5. In the grade center, set `Sort Columns By: Date Created` and `Order: Ascending`.

6. Select `Edit Column Information` from the drop-down menu next to the newly-uploaded column.

7. Set:
    - `Primary Display: Score`
    - `Category: Test`
    - `Points Possible: 100`
    - `Show Statistics: Yes`

14 Sending feedback emails to students

1. Run `./randexam proc-feedback` to generate per-student feedback PDFs. These are created in the directory named by the `feedback_directory` variable in `config.ini` (by default, a `feedback/` subdirectory). If there are extra files needed to latex the feedback, such as figures, then they need to be copied by hand into the feedback directory before running `./randexam proc-feedback`.

2. If desired, create a `rawscan` directory with PDFs of individual Scantron sheets to be included in email feedback and correspondingly set the `raw_scan_directory` variable in `config.ini`.

3. Run `./randexam proc-email`. This will ask for the NetID and password of the account that is sending the email.

   To be able to send email directly through the UIUC email servers, your NetID needs to be authorized by CITES. This only needs to be done once, as it will stay authorized. To do this, send email to the CITES Help Desk at `consult@illinois.edu` with the message "I request IMAP access for my email."

   After each email has been send, a file is created in the feedback directory called `<NetID>.email_sent`. If `./randexam proc-email` is re-run, then students with such a file won't be sent further email. Thus email sending can be retried repeatedly until all emails are successfully sent. Deleting the `*.email_sent` files will re-enable sending to those students.