Randomising assignments with SageTEX

Sabri W. Al-Safi

Abstract

SageTEX allows for the processing of SageMath code embedded in a TEX document, and the automated generation of TEX code to display SageMath objects. This article reports on the use of SageTEX to generate individualised coursework assignments (with corresponding answers) for students in an undergraduate mathematics module.

1 Motivation

One of the modules I teach to third year undergraduate mathematics students involves an individual coursework assignment; this is perhaps the type of assessment that is most vulnerable to concerns about plagiarism. In previous years, the same assignment was given to every student, which did little to help things. Hence I was curious to know if there was a relatively easy way to randomise the assignment so that each student is given a unique paper, and their solutions could be marked according to a corresponding markscheme (a.k.a. list of answers).

Various packages exist which can be used to generate randomised problems. The e-assessment system Numbas [4] is a popular example in which authors can define randomised variables when displaying mathematical objects in a question. Students can refresh the question to re-randomise these variables at will, and an automated script can be used to mark answers according to the variables' values.

Numbas is well-suited to computer-based assessment, but there are obstacles to using it for the formal assessment of advanced mathematics. Firstly, I wanted to hand out a neatly typeset document for an examination or assignment. Whilst Numbas can handle TEX syntax (and can even be tweaked to create printable PDF worksheets and markschemes), its typesetting capabilities are limited, and I would not have as much fine control over the output as when using TEX on my own computer.

A second drawback to Numbas is the lack of a sophisticated computer algebra system. The package contains a custom algebra system which includes some basic methods, but does not possess the advanced capabilities of, say, Mathematica. I would be forced to use a degree of effort and cunning to test my students' ability to transform a matrix into Jordan Canonical Form.

2 SageTEX

The TEX package SageTEX [1, 2] overcomes these drawbacks by integrating SageMath into LATEX documents. SageMath [5] is a powerful computer algebra system which is a free, open source alternative to Maple, Mathematica, Matlab or Magma. SageTEX allows a user to embed SageMath code into a TEX document, have that code executed, and automatically generate TEX code based on the results of those computations.

TUGboat has previously featured a comprehensive review of SageTEX [3], so I will cover only the salient aspects. I'll assume that both LATEX and SageMath are installed on the system (it's possible to run SageMath remotely, but we won't get into that here). The following line must be added to the preamble:

```
\usepackage{sagetex}
```

The SageMath code is usually written into either a `sageblock` or a `sagesilent` environment in the TEX document. When LATEX is run, all text within these environments is copied to a single, separate `.sage` file for execution (`sageblock` additionally typesets the code verbatim into LATEX's output). The code within these environments is collectively considered as a single SageMath program. As an example, if I want to assign to the variable A a diagonalizable 3×3 matrix with randomised integer entries and integer eigenvalues, I can write, anywhere in the TEX document (broken across lines here for *TUGboat*):

```
\begin{sagesilent}
import sage.matrix
A = random_diagonalizable_matrix(
    MatrixSpace(ZZ, 3))
\end{sagesilent}
```

In SageMath, each object has associated TEX code which renders an accurate representation of that object in math mode. One can automatically generate this code via the `\sage` macro. This allows the results of calculations to be displayed without having to do those calculations yourself, which is especially useful if your objects have been randomly generated. Thus, continuing, if I want to display a list of the eigenvalues of our matrix A later in the same document, I can write:

```
The eigenvalues of
$\sage{A}$ are $\sage{A.eigenvalues()}$.
```

The `\sageplot` command instructs Sage to generate graphical plots which are then included in the document. This circumvents some of the need to fuss with graphics files or `\includegraphics` commands (since SageTEX does it for you). Again, this can also be useful when applied to functions that depend on randomly generated variables.

```
\begin{sagesilent}
r = ZZ.random_element(10)
\end{sagesilent}
\sageplot{plot(sin(r*x), x, 0, 2*pi)} .
```

Getting SageMath to evaluate the SageMath code, produce plots and generate TEX code requires an additional couple of steps when compiling the document. When LATEX is first run on the .tex file, an auxiliary file with the extension .sagetex.sage is generated. The user must then run SageMath on this file, before running LATEX once again on the original .tex file. This series of steps can become tedious, although it is straightforward to automate (as I describe in the next section) and is only necessary when the SageMath commands are changed.

3 Method for individualising assignments

The assignment and the markscheme were written as separate TEX documents (Questions.tex and Answers.tex) to avoid having to split several PDF files into two. They were individualised by heavy use of randomly generated SageMath objects, so that although the students were required to use the same mathematical techniques, they would be applying them to different numbers, matrices and functions.

I needed the markscheme to use the same randomised objects as the assignment, therefore I needed control over the random state of the SageMath program. This was achieved by setting the same random seed at the beginning of Questions.tex and Answers.tex, which was read from a text file named random_seed.txt:

```
\begin{sagesilent}
set_random_seed(
  int( open('random_seed.txt').read() ) )
\end{sagesilent}
```

In addition, any creation of a random object in Questions.tex had to be mirrored by the exact same operation in Answers.tex. I was otherwise free to use SageMath to compute and display anything I wanted in the markscheme. This even included plotting functions, against which the student's own plots could be checked. The random seed approach had the added bonus that in order to re-generate the assignment or markscheme for a given student at any time, one could simply write the appropriate number into random_seed.txt.

The random seeds were chosen to correspond to the students' university numbers so that they were unique to each student and easy to recover. It was straightforward to export the relevant list of student numbers from the university's online learning environment, strip out all alphabetic characters, and save it to the file student_numbers.csv. The subsequent 6-digit numbers were used in turn to set the random seed at the beginning of both Questions.tex and Answers.tex. For both of these files the "TEX–SageMath–TEX" compilation procedure was carried out as outlined in the previous section, and the output PDF was renamed according to the student number. This whole process was entirely automated by the following Python script, named (for myself) compile.py:

```
import subprocess
import os

def CompileSageTex(fileName):
 subprocess.call(['pdflatex', fileName+'.tex'])
 subprocess.call(['sage',
                  fileName+'.sagetex.sage'])
 subprocess.call(['pdflatex', fileName+'.tex'])

for seed in open('student_numbers.csv')
             .read().split(','):
 open('random_seed.txt', 'w+').write(seed+'\n')
 CompileSageTex('Questions')
 CompileSageTex('Answers')
 os.rename('Questions.pdf',
           'Assignment_'+seed+'.pdf')
 os.rename('Answers.pdf',
           'Markscheme_'+seed+'.pdf')
```

The assignments were then ready to be sent to the students using a mail merge.

4 Observations

One of the drawbacks to SageMath and SageTEX is the initial difficulty of installation. SageMath is most at home in a Unix environment; using it on a Windows system requires a virtual machine, which makes SageTEX a much less viable option. It is possible to run SageMath remotely, but it may not be very easy to do this in an automated fashion as presented here. However, once I had configured SageTEX on my system, I found it intuitive and powerful.

SageTEX adds a significant amount of time to the compilation process. On my laptop running GNU/Linux, a test run of compile.py using a string of ten student numbers took a little over two minutes to produce assignment and markscheme documents with sizes roughly 123kb and 152kb respectively. If the CompileSageTex function was stripped down to a single call to pdflatex, it took about five seconds.

5 Discussion

SageMath is not the only language that can be integrated with TEX. Haskell and R are two notable examples for which similar tools exist to mix their code with TEX code, and generate TEX code on the fly. I used SageTEX primarily because:

- I already had some familiarity with Python and SageMath;
- I knew that SageMath was adept at linear algebra and symbolic computation;
- I could instantly see how to use SageTEX to achieve my goals.

I haven't tried the other alternatives myself, and a cursory search does not reveal any rigorous comparison of the available tools. Before deciding on one or another, consideration should probably be given to the differing capabilities of, and to one's prior familiarity with, each language.

My use of text files and a Python script is unlikely to be the fastest or most efficient method, although it served my immediate purposes. If the size of the assignment and the number of students is large, then disk space may be an issue. In this case, it may be desirable to print or email to each student the assignment immediately after compilation, and then delete the associated files.

Any use of SageMath aids its proliferation as a free alternative to proprietary software. The lead developer, William Stein, has written about his considerable efforts to improve and sustain its user base, especially for undergraduate teaching [6, 7]. If the method described in this article were somehow to be made simpler and faster, I believe teachers would find it very appealing as a way of cutting down on plagiarism, generating original problem sets, and maintaining strict correctness between questions and solutions.

References

[1] Drake, Dan, et al. "The SageTEX Package" (2010). `http://w.astro.berkeley.edu/~domagalski/latex/sagetexpackage.pdf`

[2] Drake, Dan, et al. SageTEX on CTAN. `http://ctan.org/pkg/sagetex`

[3] Joshi, Manjusha. A dream of computing and LATEXing together: A reality with SageTEX. `http://tug.org/TUGboat/tb32-3/tb102joshi.pdf`

[4] Numbas. `https://numbas.mathcentre.ac.uk`

[5] SageMath. `http://www.sagemath.org`

[6] Stein, William. "What is SageMath's strategy?" `http://sagemath.blogspot.co.uk/2015/09/what-is-sagemaths-strategy.html`

[7] Stein, William. "You don't really think that Sage has failed, do you?" `http://sagemath.blogspot.co.uk/2014/08/you-dont-really-think-that-sage-has.html`

◇ Sabri W. Al-Safi
School of Science & Technology,
Nottingham Trent University,
Burton Street, Nottingham,
NG1 4BU, UK
`Sabri dot Alsafi (at) ntu dot ac dot uk`