EEF pre-pilot study on statistics with computing

This is the report of a pre-pilot course teaching statistics through computing from July 11th through the 13th.

There were not enough students signing up to justify a full study, so HEFI kindly agreed to fund this course as a pre-pilot.

Summary

In this iteration of the course, compared the first run, I spent more time on basic programming ideas, such as variables, conditionals, lists, and loops. We used these ideas to show the idea of simulation with random numbers to estimate probability, and finished by implementing the permutation test for comparison of means.

Although the students did have some grasp of the basic ideas by the end of the course, most of them were still not capable of writing fairly basic programs in their final exam.

Teaching the course was helpful in showing me some striking difficulties that the students had with `for` loops, in particular.

The course was not quite long enough to cover permutation tests at a reasonable speed.

In the next iteration, I plan to drop a couple of distracting elements, and improve the teaching of difficult areas such as `for` loops.

Background

Please see my report on the first (not EEF) pilot course at the course website.

This course differed from the original in:

- Slower pace;
- Only one dataset;
- Much more time devoted to basic programming constructs.

- We only covered the Permutation test (and not correlation by permutation, paired tests).

Methods

The course web page is https://matthew-brett.github.io/eef-pre-pilot with sources at https://github.com/matthew-brett/eef-pre-pilot.

The course consisted of:

- 12 hours of face-to-face teaching (4 hours on each of 3 days).
- 2 hours of homework.

First day

- Introductory talk emphasizing reproducibility.
- Introduction to the Hansard post-Brexit survey.
- Work in groups to use Excel find proportion of voters who admit to voting Leave. Record all steps for replication.
- Introduction to variable assignment.
- Step by step analyis of survey using Python / Pandas / Jupyter.

Homework involved reflecting on inference about the Brexit voter proportions, and some simple work in the Jupyter Notebook.

Second day

- Reflecting on the Brexit voter proportions;
- Introduction to simulation with "If a family has 4 children, what is the probability that the family has 3 girls?". Simulation with coin tosses.
- Conditionals (`if`) and `for` loops.
- Lists
- Replicating coin toss simulation with Python.

Homework was modifying notebooks to solve variations on the Number of Girls problem.

Third day

- while loops as an alternative to `for` loops;
- functions
- basic plotting.

- Inference on the proportion of Leave voters in the Brexit survey using the same simulation methods we used for the Number of Girls problem.
- Permutation.
- Slicing lists.
- Comparing two groups using permutation testing.

Exam

The exam was three notebooks, for the students to fill in:

- `rainfall.ipynb` - a version of the classic programming problem (Seppälä et al. 2015). The student gets a series of numbers, and has to take the mean of the numbers preceding the first instance of `99`. This tested iteration, lists and conditional statements.
- `widgets.ipynb` - using simulation to estimate weights and variability of bag of widgets, of different weights. This tested functions generating outcomes with given probabilities, and assembling samples from this function to get and test sampling distributions.
- `sprinting.ipynb` - implementing a permutation test similar to the one we had done in class, with a new dataset.

Results

See results notebook.

The exam scores (out of 103) were very low, with min, median, max scores of 0, 3.5, 103, and mean of 27.2. The histogram shows that 4 / 6 students failed to complete any of the questions correctly. The remaining two got 51 and 103.

There wasn't an obvious pattern in the pre-course questionnaire that predicted the outcome of the exam. In response to the question "How much computer code have you written?", one student chose "I have written a small amount of useful code" but got 4 / 103 on the exam. The student with score 103 chose "I've played with code but never wrote anything useful". The student scoring 51 on the exam took A-level maths, and got a B, but the other student who had taken A-level maths got an A and got 4 on the exam. The only student taking GCSE computer science scored 4.

Motivation for taking the course seemed to be important. 2 students chose "The 120 pound bursary" in answer to "The main reason I wanted to do this course was", and scored 3 and 4 on the exam. The remainder chose "The bursary and the learning were equally important.", and these included the students with the higher scores.

Feedback was broadly similar to the first version of the course. There was moderate agreement with "I had a better understanding of statistics after the

course" (mean 2.17 / 5, where 1=Agree), and equivocal agreement with "Writing the statistical procedures in code made them harder to understand." (mean 2.5 / 5, where 1=Agree) - although see caveat in the previous course report. Students gave mean 3.17 / 5, where 5=Agree, for the question "After the course, I feel more confident about doing my own data analysis." On the 1-5 scale about the pace of the course, with 1 being "Too fast", the mean score was 2.67.

Impressions

I started the course concentrating on reproducibility, by first asking the students to do an analysis in Excel, and then describe how to reproduce it in detail. This didn't work very well, because the task was easier in Excel than I had realized, and the description part seems immediately tedious.

As part of the same move, I went through the Pandas analysis of the same data rather slowly. I told the students they did not need to understand in detail, but of course they did want explanations. This slowed down the start of the course, and probably did not help them much, because we hardly used Pandas after that.

I was very surprised by the students' confusion about `for` loops. They found these very difficult to grasp. To work round that, I reverted to teaching `while` loops on the following day. It is still not completely clear to me why they found it so difficult, but it may be that they found the loop variable (e.g. `i` in `for i in [1, 2, 3]:`) to be rather magical.

I found myself teaching the students how to debug, using print statements. I should have had some more specific time for that.

The students were working at different paces, and I found myself having to write make up new, more difficult exercises for the faster students, while the others were working.

The exam was too difficult for most of the students.

Plan

- Drop use of Excel / reproducibility example.
- Drop in-detail walk-through of Pandas analysis.
- Spend more time on lists, and `for` loops.
- Add some scaffolding to exam to get students started.

References

Seppälä, Otto, Petri Ihantola, Essi Isohanni, Juha Sorva, and Arto Vihavainen. 2015. "Do We Know How Difficult the Rainfall Problem Is?" In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, 87–96. ACM.