AS1: A/B Testing Implementation

Due: 11:59 PM CST on Mar 8, 2018

Upload submission to Blackboard

This problem set is for the graduate 422M class only.

This problem set is the first in a three-part series (AS1, AS2, AS3) in which you will conduct a controlled experiment comparing two different implementations of the same basic user interface.

1. In this problem set, you will implement the B version of your selected interface.
2. In AS2, you'll actually run the experiment on some people and collect data.
3. In AS3, you'll analyze the data and write a report about your findings.

When you run your A/B test in future weeks, the baseline A user data will be collected and shared by course staff. You will test your own B interface.

Choose the Interface

We have provided A versions for two interfaces: a Sudoku interface and a Recurrent Calendar Event interface. Decide which problem you will work on. **You only need to choose one.**

- SudokUI (Zip file)
- Recurrent Calendar Event (Zip file)

Set Up Data Logging

We have set up a shared logging structure for the A version of each interface. Included in the A version is a framework that logs events into a Google spreadsheet. The logs record raw data in a form that you can export into an analysis tool like a spreadsheet, a python script, or R.

To use this infrastructure, you will need to set up your own Google Form. Take time to read and understand the logging.js file included with each A interface. Then you should

- Set up a Google Form with seven short-text fields as described in logging.js, and direct its data to a responses spreadsheet.
- Modify your logging.js to log data to your form. You can edit in your form ID by hand, or use googlesender.py to generate the code. (See the comment at the bottom of logging.js for instructions.)
- Verify that with your modified logging.js, the A interface generates logged data in your spreadsheet. Every click should immediately add a row of data.

Your modified logging.js can be used in your B interface.

Implement the B Interface

Next, you will create an improved interface.

- Design and implement an interface that significantly improves efficiency for at least one of the tasks you can think of. You can use the A version as a starting point, or you may create a B implementation that is very different.
- Your B implementation should solve the same user problem as the A version, and it should log enough data to allow fine-grained comparisons to the A version.
- If your interface requires explanation, consider modifying the introduction message so that users can test it properly.
- Copy the logging.js that you tested with the A interfaces on your B interface, so that it logs to your spreadsheet. Be sure to set `LOG_VERSION='B'` in this file within your B version, so that logged events can be distinguished from A version events.
- In your user interface, record **at least two new** types of log events that are relevant to the interface. You can use code of the form `$(document).trigger('log', ['myevent', {key1: val1, key2: val2}]);`. By default, logging.js already logs most mousedown and keydown events in a low-level form, so you should consider logging higher-level events for your B interface that may be hard to infer from the raw keystrokes. For example, if you are improving error feedback, you might log an event when the feedback is shown and another when it is removed. If you are adding new keyboard navigation, you might log events for different types of focus changes.

Note that you do not need to run users, collect real data, or analyze data for this assignment. You'll do that in AS2 and AS3. When you hand in this assignment, however, your software must be *ready* to run users.

Ideas for Improvements

Some ideas for improving the interface:

- Reduce the number of clicks needed to do the task.
- Add or improve keyboard support to improve efficiency.
- Improve visible affordances and feedback to aid learnability.
- Adjust sizes, hit regions, and layout to exploit Fitts's law.
- Provide immediate input checking to help users avoid mistakes.
- Use visual cues to focus user attention on salient information.
- Reduce the cognitive load by simplifying the interface.
- Reduce the cognitive load by making the interface smarter: prepopulate fields, make calculations automatically, etc.
- Dynamically adjust the interface based on the input (e.g., hide fields when they become irrelevant, or show new options when they become applicable).

Collect Example Data Logs

Your software does not need to do its own analysis or generate its own graphs. We'll use other tools for that. It just needs to record raw data in a form that will be easy to analyze. **Use the logging.js script we have provided in the A versions of the interfaces**, modified to point to your own Google Form.

In your submission, include a file named **example-data.csv** containing sample data recorded by your software, just using yourself as a pilot participant. This file should contain:

- A sequence of recorded user actions in both the A and B interfaces.
- A version column that distinguishes the A and B interface.
- Columns that allow you to track timing (timestamp), distinguish users (uid), and identify types of events (included if you use logging.js as-is).
- An example of each of your new custom event types in your B interface.

You should verify that your recorded data demonstrates logging of your two new custom events.

Prepare your Report

Prepare a report as a pdf file called **as1-report.pdf** with a description and analysis of your interface. Include the following information in this report, in this order. You will be marked off it is unclear whether your experiment meets the criteria, so please be diligent in ensuring that the grader has everything they need to evaluate the submission.

1. **Problem.** Which interface you chose to improve (calendar event or sudoku).
2. **Collaborators.** A list of the people you discussed this assignment with, or else an **explicit statement that you had no collaborators.** This is an individual assignment, so be aware of the course's collaboration policy.
3. **Illustration.** Screenshots of your B interface. Highlight key design features in the screenshot.
4. **Experimental Hypothesis.** Up to three sentences describing the efficiency improvement you hope to achieve, and how you expect to be able to see this improvement when your logged data is compared to A version logs.
5. **New Logged Events.** A list of custom events that you are logging, and a description of the purpose of each event. You should have at least two new custom events.

Your report should be as succinct as possible.

What to Hand In

Package your completed assignment as a zip file that contains:

- your **as1-report.pdf** report.
- **example-data.csv** containing a sample of data recorded by your software
- a directory containing all of your **source code** for your B implementation.

Submit your zip file on Blackboard.

Grading

This assignment will be judged on four dimensions:

- Functionality (40%): does your B implementation work?
- Testability (40%): does your B implementation include all the necessary elements to conduct an experiment, and does it log data that allows it to be compared to the A version? Did you add at least two custom events?
- Presentation (20%): does your hand-in (report, sample data, source code, and zip file) make it easy to judge functionality, completeness, and testability? Is your report neatly organized?