Programming and Debugging Surveys (Full Presentation)

Emma Tosch Emery D. Berger
University of Massachusetts, Amherst

1. Introduction

Over the last decade, web-based surveys have become increasingly popular, making it possible to reach large and diverse populations at low cost. Crowdsourcing platforms like Amazon's Mechanical Turk and SurveyMonkey's Contribute enable researchers to post surveys and recruit participants at scale. Unfortunately, there are numerous problems in the design and deployment of surveys that can threaten their validity:

Question order effects. The placement of one question before another can influence the responses.

Question wording effects. Different wording of questions can elicit different responses.

Non-response / abandonment. Web survey respondents often abandon a survey partway through, a phenomenon known as *breakoff*. This may be due to *survey fatigue*–the survey is too long–or a problem with a particular question that is ambiguous, lacking an appropriate response, or is too probing.

Inattentive or random respondents. Some respondents may become inattentive and make arbitrary choices, rather than answering the question carefully. While this problem occurs in all survey scenarios, it is especially acute in an on-line setting where there is no direct supervision. The presence of automated "bots" that fill in surveys randomly is an even greater threat.

Despite the fact that these problems are well-known, no previous work addresses them. Current practice in deploying surveys is generally limited to an initial pilot study followed by a full deployment; we view this as akin to writing a program, making sure it compiles, and then shipping it.

2. SURVEYMAN

To combat these problems, we present SURVEYMAN, a programming language and runtime system for designing, debugging and deploying surveys at scale. Using SURVEYMAN, non-programmers can easily create sophisticated surveys via a lightweight programming language. SURVEYMAN can then deploy these surveys automatically over the Internet via crowdsourcing platforms. SURVEYMAN uses randomization and statistical analysis to enable debugging and quality control of responses. To our knowledge, SURVEYMAN is the first system of its kind.

3. The SURVEYMAN Language

We designed the SURVEYMAN language in concert with social scientists to ensure its usability and accessibility to non-programmers. Because social scientists routinely use Excel and R, which both feature native support for comma-separated value files, we decided to adopt a tabular format that can be entered directly in a spreadsheet and saved as `.csv` files.

Because SURVEYMAN's error analysis depends to a large extent on randomization, its language is designed to give its runtime system the greatest possible freedom to randomize question order, question variants, and answers. A user writing a survey with SURVEYMAN must explicitly state whenever randomization is *not* to be used.

Every SURVEYMAN program contains a first row of column headers, which indicate the contents of the following rows. The columns must at least include questions and options. If a given column is not present, the default value is used. The columns may appear in any order. We only describe the most important options here.

Questions and Answers. The QUESTION column contains the question text, which may include HTML. Newline characters must be quoted in the usual way. Users may wish to specify multiple variants of the same question to control or detect question wording bias. To do this, users place all of the variants in a unique block and set every question to branch to the same target (we discuss branching below).

OPTIONS are the possible answers for a question. SURVEYMAN treats a row with an empty question text field as belonging to the question above it. Depending on other entries discussed below, these may be rendered as radio buttons, checkboxes, or freetext. If there is only one option listed and the cell is empty, this question text is interpreted as if it were instructions.

Radio Button or Checkbox Questions. By default, all options are EXCLUSIVE: the respondent can only choose one. These correspond to "radio button" questions. If this column is filled with `false`, the respondent can choose multiple options, which are displayed as "checkbox" questions.

Ordering. By default, options are unordered; this corresponds to categorical answers, such as a respondent's favorite food. Ordered options include Likert scales, where respondents rate their level of agreement with a statement, or when the options comprise a ranking (e.g., from 1 to 5).

Randomization. The default setting is `true`. If options are unordered, they are presented to each respondent as one of $m!$ possible permutations of the answer labels. If they are ordered, then there are just 2 possible permutations.

This approach eliminates option order bias. More importantly, it also makes lazy respondents who click on a particular option choice indistinguishable from random respondents, which makes it possible for SURVEYMAN's quality control algorithm to find and eliminate their responses.

Blocks. Each question has an optional BLOCK number associated with it. Blocks establish a partial order. Questions with the same block number may appear in any order, but must precede all questions with a higher block number.

Branches. The BRANCH column provides control flow through the survey: there is a partition amongst the answer options such that respondents who answer a particular question differently are sent to different sets of questions, according to the partition. Branching occurs from a particular question response to a higher numbered block.

4. Runtime System

The SURVEYMAN system comprises several components. These include the language parser and validator, a system that deploys the surveys on-line to the desired sample population and collects responses, and error detection analysis.

After collecting survey responses, SURVEYMAN performs error analysis off-line and emits two reports. The first contains any identified errors in the survey itself, including question order bias, question wording bias, and breakoff. The second report contains the results of the survey, with each response tagged to indicate the computed likelihood that the respondent behaved randomly; this number is given as a percentage between 0% and 100%. Payment decisions are made on the basis of this information.

5. Error Detection

SURVEYMAN implements four error detection algorithms to identify survey errors and identify random respondents.

Question Order Bias: The placement of one question before another can influence the latter question's answer distribution. SURVEYMAN uses the Kruskal-Wallis ANOVA test to determine whether a question's answer distribution remains invariant under ordering. If question order does not have an impact, then we expect to detect no statistically significant difference in the distributions of the question's answers under different orderings.

Question Wording Variants: SURVEYMAN uses the Kruskal-Wallis test to detect two effects of question wording variants : between variants' answer distributions to determine whether a particular variant causes a change in the answer distribution, and between variants' breakoff frequencies to determine whether a particular variant causes increased breakoff.

Breakoff: To identify problematic questions, SURVEYMAN counts the number of respondents who ended the survey at each question, ignoring questions that were in the final position. To identify breakoff due to survey length, it counts the number of questions answered at each position except the last. In both cases it uses a one-sided hypothesis test to detect unusually high breakoff.

Inattentive or Random Respondents: As long as a survey is sufficiently long and there is some bias in the responses, SURVEYMAN can reliably identify random respondents or bots. If there are n unordered questions, each having k least popular elements, then k_i/m_i is the probability that a bot will choose a least popular element of question i. Let X_i denote the event that a respondent chooses a least popular option for question i and let X be the total number of questions that are answered with a least popular option. Then the expected number of questions answered with the least popular option for bots will be $\mathbb{E}(X) = \mu = \sum_{i=1}^{n} \frac{k_i}{m_i}$.

A respondent who differs from μ by more than a certain amount is, with high probability, a human: this happens when a respondent answers significantly fewer of the questions with the least popular answer. We use Chernoff bounds to describe the relationship between the expected number of least-popular responses, deviation from that expectation, and our confidence that a response that deviates is indeed a human and not a bot. SURVEYMAN's bot detection algorithm uses this bound to classify bots. Note is that μ is tunable: it is always possible to add options to a question to increase the likelihood of it containing a least popular response.

6. Conclusion and Future Work

This paper reframes surveys and their errors as a programming language problem. It presents SURVEYMAN, a programming language and runtime system for implementing surveys and identifying their errors. Pervasive randomization prevents small biases from being magnified and enables rigorous statistical analyses, which in turn allows researchers to identify serious flaws in survey design. We believe that SURVEYMAN has the potential to have significant impact on the reliability and reproducibility of social science research. We are currently working with psycholinguistics researchers who are writing surveys in the SURVEYMAN language and deploying surveys through our runtime system on Mechanical Turk.