SURVEYMAN: Programming and Automatically Debugging Surveys

Emma Tosch Emery D. Berger

School of Computer Science
University of Massachusetts, Amherst
Amherst, MA 01003
{etosch,emery}@cs.umass.edu

Abstract

Surveys can be viewed as programs, complete with logic, control flow, and bugs. Word choice or the order in which questions are asked can unintentionally bias responses. Vague, confusing, or intrusive questions can cause respondents to abandon a survey. Surveys can also have runtime errors: inattentive respondents can taint results. This effect is especially problematic when deploying surveys in uncontrolled settings, such as on the web or via crowdsourcing platforms. Because the results of surveys drive business decisions and inform scientific conclusions, it is crucial to make sure they are correct.

We present SURVEYMAN, a system for designing, deploying, and automatically debugging surveys. Survey authors write their surveys in a lightweight domain-specific language aimed at end users. SURVEYMAN statically analyzes the survey to provide feedback to survey authors before deployment. It then compiles the survey into JavaScript and deploys it either to the web or a crowdsourcing platform. SURVEYMAN's dynamic analyses automatically find survey bugs, and control for the quality of responses. We evaluate SURVEYMAN's algorithms analytically and empirically, demonstrating its effectiveness with case studies of social science surveys conducted via Amazon's Mechanical Turk.

1. Introduction

Surveys and polls are widely used to conduct research for industry, politics, and the social sciences. Businesses use surveys to perform market research to inform their spending and product strategies [4]. Political and news organizations use surveys to gather public opinion, which can influence political decisions and political campaigns. A wide range of social scientists, including psychologists, economists, health professionals, political scientists, and sociologists, make extensive use of surveys to drive their research [3, 7, 13, 22].

In the past, surveys were traditionally administered via mailings, phone calls, or face-to-face interviews [9]. Over the last decade, web-based surveys have become increasingly popular as they make it possible to reach large and diverse populations at low cost [6, 34, 42, 43]. Crowdsourcing platforms like Amazon's Mechanical Turk make it possible for researchers to post surveys and recruit participants at scales that would otherwise be out of reach.

Unfortunately, the design and deployment of surveys can seriously threaten the validity of their results:

Question order effects. Placing one question before another can lead to different responses than when their order is reversed. For example, a Pew Research poll found that people were more likely to favor civil unions for gays when this question was asked after one about whether they favored or opposed gay marriage [27].

Question wording effects. Different question variants can inadvertently elicit wildly different responses. For example, in 2003, Pew Research found that American support for possible U.S. military action in Iraq was 68%, but this support dropped to 43% when the question mentioned possible American casualties [28]. Even apparently equivalent questions can yield quite different responses. In a 2005 survey, 51% of respondents favored "making it legal for doctors to *give terminally ill patients the means to end their lives*" but only 44% favored "making it legal for doctors to *assist terminally ill patients in committing suicide*." [28]

Survey abandonment. Respondents often abandon a survey partway through, a phenomenon known as *breakoff*. This effect may be due to *survey fatigue*, when a survey is too long, or because a particular question is ambiguous, lacks an appropriate response, or is too intrusive. If an entire group of survey respondents abandon a survey, the result can be *selection bias*: the survey will exclude an entire group from the population being surveyed.

OOPSLA '14 October 22–24, 2014, Portland, Oregon, USA

Inattentive or random respondents. Some respondents are inattentive and make arbitrary choices, rather than answering the question carefully. While this problem can arise in all survey scenarios, it is especially acute in an on-line setting where there is no direct supervision. To make matters worse, there is no "right answer" to check against for surveys, making controlling for quality difficult.

Downs et al. found that nearly 40% of survey respondents on Amazon's Mechanical Turk answered randomly [10]. So-called *attention check* questions aimed at screening inattentive workers are ineffective because they are easily recognized. An unfortunately typical situation is that described by a commenter on a recent article about taking surveys on Mechanical Turk:

> If the requester is paying very little, I will go as fast as I can through the survey making sure to pass their attention checks, so that I'm compensated fairly. Conversely, if the requester wants to pay a fair wage, I will take my time and give a more thought out and non random response. [11]

While all of the above problems are known to practitioners [21], there is currently no way to address them automatically. The result is that current practice in deploying surveys is generally limited to an initial pilot study followed by a full deployment, with no way to control for the potentially devastating impact of survey errors and inattentive respondents.

From our perspective, this is like writing a program, making sure it compiles, and then shipping it—to run on a system with hardware problems.

1.1 SURVEYMAN

In this paper, we adopt the view that surveys are effectively programs, complete with logic, control flow, and bugs. We describe SURVEYMAN, which aims to provide a scientific footing for the development of surveys and the analysis of their results. Using SURVEYMAN, survey authors use a lightweight domain-specific language to create their surveys. SURVEYMAN then deploys their surveys over the Internet, either by hosting it as a website, or via crowdsourcing platforms.

The key idea behind SURVEYMAN is that by giving survey authors a way to write their surveys that steers them away from unnecessary ordering constraints, we can apply static analysis, randomization, and statistical dynamic analysis to locate survey errors and ensure the quality of responses.

Overview: Figure 1 depicts the SURVEYMAN workflow. Survey authors create surveys using the SURVEYMAN programming language. The SURVEYMAN static analyzer checks the SURVEYMAN program for validity and reports key statistics about the survey prior to deployment. If the program is correct, SURVEYMAN's runtime system can then deploy the survey via the web or a crowdsourcing platform: each respondent sees a differently-randomized version. SURVEYMAN's dynamic analysis operates on information from the static analysis and the results of the survey to identify survey errors and inattentive respondents. Table 1 summarizes the analyses that SURVEYMAN performs.

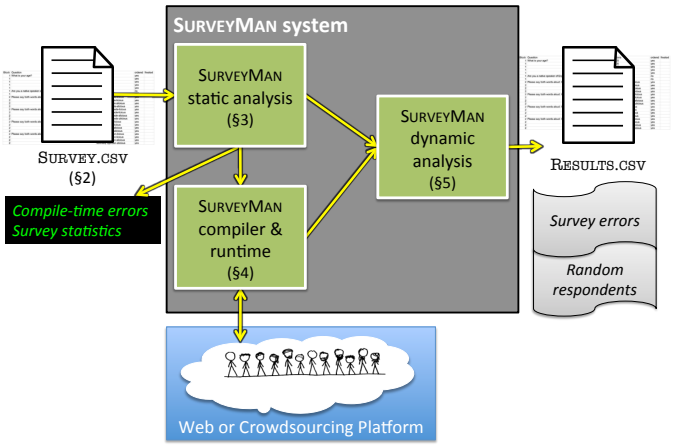


Figure 1: Overview of the SURVEYMAN system.

Domain-Specific Language. We designed the SURVEYMAN language in concert with social scientists to ensure its usability and accessibility to non-programmers (§2). The approach we take leverages the tools that our target audiences use: because social scientists extensively use both Excel and R, which both feature native support for comma-separated value files, we adopt a tabular format that can be entered directly in a spreadsheet and saved as a `.csv` file.

SURVEYMAN's domain-specific language is simple but captures most features needed by survey authors, including a variety of answer types and branching based on answers to particular questions. In addition, because SURVEYMAN's error analysis depends on randomization, its language is designed to maximize SURVEYMAN's freedom to randomize question order, question variants, and answers.

A user writing a survey with SURVEYMAN can optionally specify a partial order over questions by grouping questions into *blocks*, identified by number. All questions within the same block may be asked in any order, but must strictly precede the following block (i.e., all the questions in block 1 precede those in block 2, and so on).

Static Analysis. SURVEYMAN statically analyzes the survey to verify that it meets certain criteria, such as that all branches point to a target, and that there are no cycles (§3). It also provides additional feedback to the survey designer, indicating potential problems with the survey that it locates prior to deployment.

Compiler and Runtime System. SURVEYMAN can then deploy the survey either on the web with SURVEYMAN acting as a webserver, or by posting jobs to a crowdsourcing platform such as Amazon's Mechanical Turk (§4). Each survey is delivered as a JavaScript and JSON payload that

	Static Analyses	
Well-formedness	§3.1	Ensures survey is a DAG and other correctness checks
Reachability	§3.1	Ensures the possibility of seeing each question.
Survey statistics	§3.2	Min, max, and avg. # questions in survey; finds short-circuits and guides pricing
Entropy	§3.2	Measures information content of survey; higher is better
	Dynamic Analyses	
Correlated Questions	§5.1	Reports redundant questions which can be eliminated to reduce survey length
Question Order Bias	§5.2	Reports questions whose results depend *on the order* in which they are asked
Question Wording Variant Bias	§5.3	Reports questions whose results depend *on the way* they are worded
Breakoff	§5.4	Finds problematic questions that lead to survey abandonment
Inattentive or Random Respondents	§5.5	Identifies unconscientious respondents so they can be excluded in analysis

Table 1: The analyses that SURVEYMAN performs on surveys and their deployment.

manages presentation and flow through the survey, and performs per-user randomization of question and answer order subject to constraints placed by the survey author. When a respondent completes or abandons a survey, SURVEYMAN collects the survey's results for analysis.

Dynamic Analyses. To find errors in a deployed survey, SURVEYMAN performs statistical analyses that take into account the survey's static control flow, the partial order on questions, and collected results (§5). These analyses can identify a range of possible survey errors, including question order bias, wording variant bias, questions that lead to breakoff, and survey fatigue. SURVEYMAN reports each error with its corresponding question, when applicable. It also identifies inattentive and random respondents. The survey author can then use the resulting report to refine their survey and exclude random respondents.

Note that certain problems with surveys are beyond the scope of any tool to address [43]. These include *coverage error*, when the respondents do not include the target population of interest; *sampling error*, when the make-up of the respondents is not representative of the target of interest; and *non-response bias*, when respondents to a survey are different from those who choose not to or are unable to participate in a survey. SURVEYMAN can help survey designers limit non-response bias due to abandonment by diagnosing and fixing its cause (breakoff or fatigue).

Evaluation. Our collaborators in the social sciences developed a number of surveys in SURVEYMAN, which were deployed on Amazon's Mechanical Turk (§6). We describe these experiences with SURVEYMAN and the results of the deployment, which identified a number of errors in the surveys as well as random respondents.

1.2 Contributions

The contributions of this paper are the following:

- **Domain-Specific Language.** We introduce the SURVEYMAN domain-specific language for writing surveys, which enables error detection and quality control by relaxing ordering constraints (§2, §4).
- **Static Analyses.** We present static analyses for identifying structural problems in surveys (§3).
- **Dynamic Analyses.** We present dynamic analyses to identify a range of important survey errors, including question variant bias, order bias, and breakoff, as well as inattentive or random respondents (§5).
- **Experimental Results.** We report on a deployment of SURVEYMAN with social scientists and demonstrate its utility (§6).

2. SURVEYMAN Domain-Specific Language

2.1 Overview

The SURVEYMAN programming language is a tabular, lightweight language aimed at end users. In particular, it is designed to both make it easy for survey authors without programming experience to create simple surveys, and let more advanced users create sophisticated surveys. Because of its tabular format, SURVEYMAN users can enter their surveys directly in a spreadsheet application, such as Microsoft Excel. Unlike text editors or IDEs, spreadsheet applications are tools that our target audience knows well. SURVEYMAN can read in `.csv` files, which it then checks for validity (§3), compiles and deploys (§4), and reports results, including errors (§5).

A key distinguishing feature of SURVEYMAN's language is its support for randomization, including question order, question variants, and answers. From SURVEYMAN's perspective, more randomization is better, both because it makes error detection more effective and because it provides experimental control for possible biases (§5). SURVEYMAN is designed so that survey authors must go out of their way to state when randomization is *not* to be used. This approach encourages authors to avoid imposing unnecessary ordering constraints.

Basic Surveys: To create a basic survey, a survey author simply lists questions and possible answers in a sequence of rows. When the survey is deployed, all questions are presented in a random order, and the order of all answers is also randomized.

BLOCK	QUESTION	OPTIONS	EXCLUSIVE	ORDERED	BRANCH
1	What is your gender?	Male			
1		Female			
1		Other			
1	What country do you live in?	United States			2
1		India			3
1		Other			3
2	What state do you live in?	Alabama		TRUE	
2		Alaska		TRUE	
2		Arizona		TRUE	
2		Arkansas		TRUE	
2		California		TRUE	
3	How much time do you spend on Mechanical Turk?	Less than 1 hour per week.		TRUE	
3		1-2 hours per week.		TRUE	
3		2-4 hours per week.		TRUE	
3		4-8 hours per week.		TRUE	
3		8-20 hours per week.		TRUE	
3		20-40 hours per week.		TRUE	
3		More than 40 hours per week.		TRUE	
3	Check all the reasons why you use Mechanical Turk.	Fruitful way to spend time.	FALSE		
3		Primary source of income.	FALSE		
3		Secondary source of income.	FALSE		
3		To kill time.	FALSE		
3		I find the tasks to be fun.	FALSE		
3		I am currently unemployed.	FALSE		

Figure 2: An example survey written using the SURVEYMAN domain-specific language, adapted from Ipeirotis [16]. For clarity, a horizontal line separates each question, and a double horizontal line separates distinct *blocks*, which optionally define a partial order: all questions in block i appear in random order before questions in blocks $j > i$. When blocks are not specified, all questions may appear in a random order. This relaxed ordering enables SURVEYMAN's error analyses.

Ordering Constraints: Survey authors can assign numbers to questions that they wish to order; we use the terminology of the survey literature and call these numbers *blocks*. Multiple questions can have the same number, which causes them to appear in the same block. Every question in the same block will be presented in a random order to each respondent. All questions inside blocks of a lower number will be presented before questions inside higher-numbered blocks.

SURVEYMAN's block construct has additional features that can give advanced survey authors finer-grained control over ordering (§2.3).

Logic and Control Flow: SURVEYMAN also includes both logic and control flow in the form of branches depending on the survey taker's responses. Each answer can contain a target branch (a block), which is taken if the survey taker chooses that answer.

2.2 Syntax

Figure 2 presents a sample SURVEYMAN program, a survey from a Mechanical Turk demographic survey modified to illustrate SURVEYMAN's features [16].

Every SURVEYMAN program contains a first row of column headers, which indicate the contents of the following rows. The only mandatory columns are QUESTION and OPTIONS; all other columns are optional. If a given column is not present, its default value is used. The columns may appear in any order.

Column	Description
QUESTION	The text for a question
OPTIONS	Answer choices, one per row
BLOCK	Numbers used to partially order questions
EXCLUSIVE	Only one choice allowed (default)
ORDERED	Present options in order
BRANCH	For this response, go to this block
RANDOMIZE	Randomize option orders (default)
FREETEXT	Allow text entry, optionally constrained
CORRELATED	Used to indicate questions are correlated

Table 2: Columns in SURVEYMAN. All except the first two (QUESTION and OPTIONS) are optional.

Blocks. Each question can have 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. If no block column is specified, all questions are placed in the same block, meaning that they can appear in any order.

Questions and Answers. The QUESTION column contains the question text, which may include HTML. Users may wish to specify multiple variants of the same question to control for or detect question wording bias. To do this, users place all of the variants in a particular block and have every question branch to the same target.

The survey author specifies each question as a series of consecutive rows. A row with the QUESTION column filled in indicates the end of the previous question and the start of a new one. All other rows leave this column empty (as in Figure 2).

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. 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 presented to the respondent as 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 the user specifies a EXCLUSIVE column and fills it with `false`, the respondent can choose multiple options, which are displayed as "checkbox" questions.

Freetext Questions. The default value for FREETEXT is `false`. There are three ways of specifying freetext questions:

1. Enter `true` in the FREETEXT column. The interpreter will show an empty text area.
2. Enter a regular expression for validation. We denote regular expressions by $\#\{\langle \texttt{regexp} \rangle\}$, where $\langle \texttt{regexp} \rangle$ is validated by the Javascript runtime.
3. Enter any other string, which will be interpreted as a default value to be displayed in the text area.

When a question's FREETEXT column is not empty or `false`, SURVEYMAN ignores all other flags and ensures that the set of options is empty.

Ordering. By default, options are unordered; this corresponds to nominal data like a respondent's favorite food, where there is no ordering relationship. Ordered options include so-called Likert scales, where respondents rate their level of agreement with a statement; when the options comprise a ranking (e.g., from 1 to 5); and when ordering is necessary for navigation (e.g., for a long list of countries).

When options are unordered, they are presented to each respondent as one of $m!$ possible permutations of the answers, where m is the number of options. To order the options, the user must fill in a ORDERED column with the value TRUE. When they are ordered, they can still be randomized: they are either presented in the forward or backwards order. The user can only force the answers to appear in exactly the given order by also including a RANDOMIZE column and filling in the value FALSE.

Unordered options eliminate option order bias, because the position of each option is randomized. They also make inattentive respondents who always click on a particular option choice indistinguishable from random respondents. This property lets SURVEYMAN's quality control algorithm simultaneously identify both inattentive and random responses.

Branches. The BRANCH column provides control flow through the survey when the survey designer wants respondents who answer a particular question one way to take one path through the survey, while respondents who answer differently take a different path. For example, in the survey shown in Figure 2, only when respondents answer that they live in the United States will they be asked what state they live in.

During the survey, branching is deferred until the end of a block, as Figure 3(a) depicts. By avoiding premature branching, this approach ensures that all users answer the same questions, regardless of randomization. It also avoids the biasing of question order that would result by forcing branches to appear in a fixed position.

Branching from a particular question response must go to a higher numbered block, preventing cycles.

2.3 Advanced Features

SURVEYMAN has several advanced features to give survey authors more control over their surveys and their interaction with SURVEYMAN.

Correlated questions. SURVEYMAN lets authors include a CORRELATED column to assist in quality control. SURVEYMAN checks for correlations between questions to see if any redundancy can be removed, since shorter surveys help reduce survey fatigue. However, sometimes correlations are desired, whether to confirm a hypothesis or as a form of quality control. SURVEYMAN thus lets authors mark sets of questions as correlated by filling in the column with arbitary text: all questions with the same text are assumed to be correlated. SURVEYMAN will then only report if the answers to these questions are *not* correlated. If not, this information can be used to help identify inattentive or adversarial respondents.

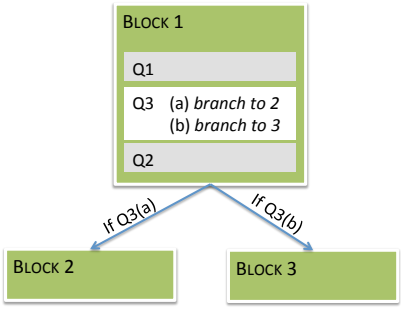
Advanced Blocks Notation

So far, we have described blocks as if they were limited to non-negative integers. In fact, SURVEYMAN's block syntax is more general[1].

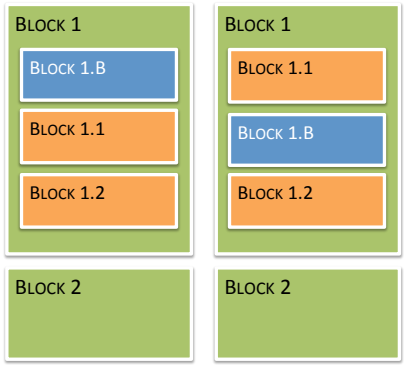
There are three kinds of blocks: *top-level blocks*, indicated by a single number; *nested blocks*, indicated by numbers separated by periods; and *floating blocks*, indicated by alphanumerics, which can move around inside their parent blocks. We describe each in terms of their interaction with other blocks and with branches. Figure 3(b) gives an example of each.

Top-level Blocks. A survey is composed of blocks. If there are no blocks (i.e., the survey is completely flat), we say that all questions belong to a single, top-level block (block `1`). Only one branch question is allowed per top-level block. Additionally, branch question targets must also be top-level blocks, though they cannot be floating blocks (described below).

[1] Formally, the syntax of blocks is given by the following regular expression: `[1-9][0-9]*(.[a-z1-9][0-9]*)*`.



(a) In this example of survey branches, the respondent is asked the branching question (Q3) as the second question. The execution of this branch is deferred until the respondent answers Q2, the last question in the block.



(b) Two different instances of a survey with top-level blocks (`1` and `2`), nested blocks (`1.1` and `1.2`), and a floating block (`1.B`). The floating block can move anywhere within block `1`, while the other blocks retain their relative orders (§2.3).

Figure 3: Examples of branches and blocks.

Nested Blocks. As with outlines, a period in a block indicates hierarchy: we can think of blocks numbered `1.1` and `1.2` as being *contained* within block `1`. All questions numbered `1.1` precede all questions numbered `1.2`, and both strictly precede blocks with numbers 2 or higher.

Floating Blocks. Survey authors may want certain questions to be allowed to appear anywhere within a containing block. For example, a survey author might want questions to appear within block `1`, but does not care whether they appear before or after other questions contained in that block (e.g., questions with block numbers `1.1` and `1.2`). Such questions can be placed in *floating blocks*. A question is marked as floating by using a block name that is an alphanumeric string rather than a number.

In this case, the survey author could give a floating question the block number `1.A`. That block is contained within block `1`, and the question is free to float anywhere within that block. The string itself has no effect on ordering; for example, blocks `1.A` and `1.B` can appear in any order inside block `1`. All questions in the same floating block float together inside their containing block.

3. Static Analyses

SURVEYMAN provides a number of static analyses that check the survey for validity and identify potential issues.

3.1 Well-Formedness

After successfully parsing its `.csv` input, the SURVEYMAN analyzer verifies that the input program itself is well-formed and issues warnings, if appropriate. It checks that the header contains the required QUESTION and OPTION columns. It warns the programmer if it finds any unrecognized headers, since these may be due to typographical errors. SURVEYMAN also issues a warning if it finds any duplicate questions. It then performs more detailed analysis of branches:

Top-Level Branch Targets Only: SURVEYMAN verifies that the target of each branch in the survey is a top-level block; that is, it cannot branch into a nested block. This check takes constant time.

Forward Branches: Since the survey must be a DAG, all branching must be to higher-numbered blocks. SURVEYMAN verifies that each branch target id is larger than the id of the block that the branch question belongs to.

Consistent Block Branch Types: SURVEYMAN computes the *branch type* of every block. A block can contain no branches in the block (NONE) or exactly one branch question (ONE). It can also be a block where all of the questions have branches to the same destination (ALL). This is the approach used to express question variants, where just one of a range of semantically-identical questions is chosen. In this case, questions can only differ in their text; only one of the variants (chosen randomly) will be asked.

Reachability: If we define a path through a survey to be the sequence of questions answered, the union of the set of questions in all possible paths must be equal to the set of questions specified in the survey. If these two sets are not equal, then some question is unreachable. Questions can only become unreachable if they are contained in an ordered block that is skipped, due to branching. When this happens, SURVEYMAN throws an error.

3.2 Survey Statistics and Entropy

If a program passes all of the above checks, SURVEYMAN produces a report of various statistics about the survey, including an analysis of paths through the survey.

If we view a path through the survey as the series of questions seen and answered, the computation of paths would be intractable. Consider a survey with 16 randomizable blocks, each having four variants and no branching. We thus have $16!$ total block orderings; since we choose from four different questions, we end up with $216!$ unique paths.

Fortunately, the statistics we need require only computations of the *length* of paths rather than the *contents* of the path. The number of questions in each block can be computed statically. We thus only need to consider the branching between these blocks when computing path lengths.

Minimum Path Length: Branching in surveys is typically related to some kind of division in the underlying population. When surveys have sufficient branching, it may be possible for some respondents to answer far fewer questions than the survey designer intended – they may *short circuit* the survey. Sometimes this is by design: for a survey only interested in curly-haired respondents, the survey can be designed so that answering "no" to "Do you have curly hair?" sends the respondent straight to the end. In other cases, this may not be the intended effect and could be either a typographical error or a case of poor survey design. SURVEYMAN reports the minimum number of questions a survey respondent could answer while completing the survey.

Maximum Path Length: A survey that is too long can lead to survey fatigue. Surveys that are too long are also likelier to lead to inattentive responses. SURVEYMAN reports the number of questions in the longest path through the survey to alert authors to this risk.

Average Path Length: Backends such as Amazon Mechanical Turk require the requester to provide a time limit on surveys and a payment. The survey author can use the average path length through the survey to estimate the time it would take to complete, and from that compute the baseline payment. SURVEYMAN computes this average by generating a large number of random responses to the survey (currently, 5000) and reports the mean length.

Maximum Entropy: Finally, SURVEYMAN reports the entropy of the survey. This number roughly corresponds to the complexity of the survey. If the survey has very low entropy (few questions or answers), the survey will require many respondents for SURVEYMAN to be able to identify inattentive respondents. Surveys with higher entropy provide SURVEYMAN with greater discriminatory power. For a survey of n questions each with m_i responses, SURVEYMAN computes a conservative upper bound on the entropy on the survey as $n \log_2(\max\{m_i \mid 1 \leq i \leq n\})$.

4. Compiler and Runtime System

Once a survey passes the validity checks described in Section 3.1, the SURVEYMAN compiler transforms it into a payload that runs inside a web page (§4.1). SURVEYMAN then deploys the survey, either by acting as a webserver itself, or by posting the page to a crowdsourcing platform and collecting responses (§4.2). Display of questions and flow through the survey are managed by an interpreter written in JavaScript (§4.3). After all responses have been collected, SURVEYMAN performs the dynamic analyses described in Section 5.

4.1 Compilation

SURVEYMAN transforms the survey into a minimal JSON representation wrapped in HTML, which is executed by the interpreter described in Section 4.3. The JSON representation contains only the bare minimum information required by the JavaScript interpreter for execution, with analysis-related information stripped out.

Surveys can be targeted to run on different platforms; SURVEYMAN supports any platform that allows the use of arbitrary HTML. When the platform is the local webserver, the HTML is the final product. When posting to Amazon's Mechanical Turk (AMT), SURVEYMAN wraps the HTML inside an XML payload which is handled by AMT. The embedded interpreter handles randomization, the presentation of questions and answers, and communication with the SURVEYMAN runtime.

4.2 Survey Execution

We describe how surveys appear to a respondent on Amazon's Mechanical Turk; their appearance on a webserver is similar. When a respondent navigates to the webpage displaying the survey, they first see a consent form in the HIT preview. After accepting the HIT, they begin the survey.

The user then sees the first question and the answer options. When they select some answer (or type in a text box), the next button and a SUBMIT EARLY button appear. If the question is instructional and is not the final question in the survey, only the next button appears. When the respondent reaches the final question, only a SUBMIT button appears.

Each user sees a different ordering of questions. However, each particular user's questions are always presented in the same order. The random number generator is seeded with the user's session or *assignment* id. This means that if the user navigates away from the page and returns to the HIT, the question order will be the same upon second viewing.

SURVEYMAN displays only one question at a time. This design decision is not purely aesthetic; it also makes measuring breakoff more precise, because there is never any confusion as to which question might have led someone to abandon a survey.

4.3 Interpreter

Execution of the survey is controlled by an interpreter that manages the underlying evaluation and display. This interpreter contains two key components. The first handles survey logic; it runs in a loop, displaying questions and process-

ing responses. The second layer handles display information, updating the HTML in response to certain events.

In addition to the functions that implement the state machine, the interpreter maintains three global variables:

- **Block Stack:** Since the path through a survey is determined by branching over blocks, and since only forward branches are permitted, the interpreter maintains blocks on a stack.
- **Question Stack:** Once a decision has been made as to which block is being executed (e.g., after a random selection of a block), the appropriate questions for that block are placed on the question stack.
- **Branch Reference Cell:** If a block contains a branch, this value stores its target. The interpreter defers executing the branch until all of the questions in the block have been answered.

The SURVEYMAN interpreter is first initialized with the survey JSON, which is parsed into an internal survey representation and then randomized using a seeded random number generator. This randomization preserves necessary invariants like the partial order over blocks. It then pushes all of the top-level blocks onto the block stack. It then pops off the first block and initializes the question stack, and starts the survey. One question appears at a time, and the interpreter manages control flow to branches depending on the answers given by the respondent.

5. Dynamic Analyses

After collecting all results, SURVEYMAN performs a series of tests to identify survey errors and inattentive or random respondents. Table 3 provides an overview of the statistical tests used for each error and each possible combination of question types tested. The categories of data being compared determine the statistical tests that SURVEYMAN uses. These categories are known as "scales of measurement" in the behavioral sciences [38]. SURVEYMAN supports both unordered (*nominal*) and ordered (*ordinal*) scales.

Unordered data include sex, language spoken, and yes/no questions. The most powerful comparison we can make between two unordered measurements is equality.

Ordered data in surveys primarily take the form of Likert-scale questions. These questions have an ordering, but there is no meaningful interpretation for the magnitude of difference between ranks.

SURVEYMAN uses the EXCLUSIVE and ORDERED columns to determine the category of data. SURVEYMAN does not support any automated analyses on questions whose EXCLUSIVE column is set to `false` (i.e. "checkbox" questions). When a question's EXCLUSIVE column is set to `true` and its ORDERED column is set to `false`, the question's response options are unordered data. When both EXCLUSIVE and ORDERED are set to `true`, the question's response options represent ordered data.

5.1 Correlated Questions

SURVEYMAN analyzes correlation in two ways. The CORRELATED column can be used to indicate sets of questions that the survey author expects to have statistical correlation. Flagged questions can be used to validate or reject hypotheses and to help detect bad actors. Alternatively, if a question not marked as correlated is found to have a statistically significant correlation, then SURVEYMAN flags it.

For two questions such that at least one of them is unordered, SURVEYMAN returns the χ^2 statistic, its p-value, and computes Cramér's V to determine correlation. χ^2 is the standard statistic for comparing categorical (unordered) data. Each variable can only take on one of a set of values and we expect the underlying distribution of values (question responses) for each variable (question) to be independent.

For any two questions, we find the subset of respondents who answered both questions. We fill a contingency table with those counts and compute the χ^2 statistic. Cramér's V scales the χ^2 statistic for use as a correlation coefficient. SURVEYMAN also uses Cramér's V when comparing an unordered and an ordered question.

Ordered questions are compared using Spearman's ρ. This is a commonly used statistic for comparing ranked data. It measures the strength of a monotonic relationship between two ranked variables.

In practice, SURVEYMAN rarely has sufficient data to return confidence intervals on such point estimates. Instead, it simply flags the pair of questions with the value computed.

The survey author can act on this information in two ways. First, the survey author may decide to shorten the survey by removing one or more of the correlated questions. It is ultimately the responsibility of the survey author to use good judgement and domain knowledge when deciding to remove questions. Second, the survey author could use discovered correlations to assist in the identification of cohorts or bad actors by updating the entries in the CORRELATED column appropriately.

5.2 Question Order Bias

To compute order bias, SURVEYMAN uses the Mann-Whitney U test for ordered questions and the χ^2 statistic for unordered questions. In each case, SURVEYMAN attempts to disprove the assumption that the distributions of responses are the same regardless of question order.

For each question pair (q_i, q_j), where $i \neq j$, SURVEYMAN partitions the sample into two sets: $S_{i<j}$, the set of questions where q_i precedes q_j, and $S_{j<i}$, the set of questions where q_i follows q_j. Each pair of observations will correspond to an individual respondent. We assume that respondents do not collude; this allows SURVEYMAN to assume each set is independent. We outline below how to test for bias in q_i when q_j precedes it (the other case is symmetric), both for ordered and unordered questions.

ERROR	BOTH ORDERED	BOTH UNORDERED	ORDERED-UNORDERED
Correlated Questions	Spearman's ρ	Cramér's V	Cramér's V
Question Order Bias	Mann-Whitney U-Test	χ^2	N/A
Question Wording Variant Bias	Mann-Whitney U-Test	χ^2	N/A
Inattentive or Random Respondents	Nonparametric bootstrap over empirical entropy	Nonparametric bootstrap over empirical entropy	N/A

Table 3: The statistical tests used to find particular errors. Tests are conducted pair-wise across questions; each column indicates whether the pairs of questions both have answers that are ordered, both unordered, or a mix.

Ordered Questions: Mann-Whitney U Statistic

1. Assign ranks to each of the options. For example, in a Likert-scale question having options *Strongly Disagree*, *Disagree*, *Agree*, and *Strongly Agree*, assign each the values 1 through 4.
2. Convert each answer to q_i in $S_{i<j}$ to its rank, assigning average ranks to ties.
3. Convert each answer to q_j in $S_{j<i}$ to its rank, assigning average ranks to ties.
4. Compute the U statistic over the two sets of ranks. If the probability of computing U is less than the critical value, there is a significant difference in the ordering.

Unordered Questions: χ^2 Statistic

1. Compute frequencies $f_{i<j}$ for the answer options of q_i in the set of responses $S_{i<j}$. We use these values to compute the estimator.
2. Compute frequencies $f_{j<i}$ for answer options q_i in the set of responses $S_{j<i}$. These form our observations.
3. Compute the χ^2 statistic on the data set. The degrees of freedom will be one less than the number of answer options, squared. If the probability of computing such a number is less than the value at the χ^2 distribution with these parameters, there is a significant difference in the ordering.

SURVEYMAN computes these values for every unique question pair, and reports questions with an identified order bias.

5.3 Question Wording (Variant) Bias

Wording bias uses almost the same analysis approach as order bias. SURVEYMAN compares the response distributions of $\binom{k}{2}$ pairs of questions, where k corresponds to the number of variants. As with order bias, SURVEYMAN reports questions whose wording variants lead to a statistically significant difference in responses.

5.4 Breakoff vs. Fatigue

SURVEYMAN identifies and distinguishes two kinds of breakoff: breakoff triggered at a particular *position* in the survey, and breakoff at a particular *question*. Breakoff by position is often an indicator that the survey is too long. Breakoff by question may indicate that a question is unclear, offensive, or burdensome to the respondent.

Since SURVEYMAN randomizes the order of questions whenever possible, it can generally distinguish between positional breakoff and question breakoff without the need for any statistical tests. To identify both forms of breakoff, SURVEYMAN reports ranked lists of the number of respondents who abandoned the survey by position and by question. A cluster around a position indicates fatigue or that the compensation for the survey, if any, is inadequate. A high number of abandonments at a particular question indicates a problem with the question itself.

5.5 Inattentive or Random Respondents

Prior Approaches

Inattentive and random respondents are known threats to validity in the survey literature. These threats are amplified for web surveys. Researchers in the social sciences already employ *ad hoc* quality control mechanisms to address these threats. These include the following:

- *Catch trials*, questions with only one legitimate response [5];
- *Attention-check questions*, which are repeated, sometimes with different wording, to check for consistency [31, 41];
- *Instructional manipulation checks*, where the question text indicates how to answer a question (e.g., that the respondent should pick option (b)) [26]; and
- *Good-faith reminders*, exhortations to the survey taker to pay attention and answer truthfully [23].

These quality control mechanisms are vulnerable to malicious respondents. For example, catch trials and manipulation checks are easily spotted and thus can be gamed. Conversely, honest respondents who experience fatigue may inadvertently fail to answer these correctly and be incorrectly excluded [24].

In addition to *ad hoc* in-survey quality control measures, researchers also employ *post hoc* statistical analyses to identify inattentive respondents, also known as *insufficient effort responding* [14]. Approaches to identifying inattentive respondents include the following:

- Picking a threshold number of failed catch trials;

- Identifying outlier responses at the question level; and
- Identifying outlier responses using the multi-dimensional Mahalanobis distance over the entire survey [14].

These approaches can be useful, but do not generalize. Although general catch trials exist (e.g., asking for a respondent's date of birth twice, worded differently), they are easily spotted and gamed. Domain-specific catch trials are not reusable across surveys. Mahalanobis distance and other outlier based approaches assume that legitimate survey responses all form one cluster. This assumption can easily be violated if survey respondents constitute multiple clusters. For example, politically conservative individuals might answer questions on a health care survey quite differently from politically liberal individuals. In any event, there is no compelling reason to believe clusters will be distributed normally around the mean.

Other *ad hoc* techniques aimed at identifying random or low-effort respondents include measuring the amount of time spent completing a survey, adding CAPTCHAs, and measuring the entropy of responses, assuming that low-effort respondents either choose one option or alternate between options regularly [45]; none of these reliably distinguish lazy or random respondents from real respondents.

Many *post hoc* statistical analyses must exclude respondents with partial responses or must somehow impute the missing data. It is also unclear how to take into account surveys with branches into these extant statistical analyses; different respondents will take different paths, making them incomparable.

SURVEYMAN's Approach

SURVEYMAN's quality control provides a general solution to finding inattentive respondents; by virtue of SURVEYMAN's randomization of answers, it also automatically identifies random respondents.

SURVEYMAN uses an approach inspired by the entropy calculation in its static analysis, but that represents scaled likelihood. SURVEYMAN first computes the empirical probabilities for each question's answer options.

Let n be the number of questions in a survey. For every response r, SURVEYMAN calculates a score based on entropy. Since respondents may exit the survey at any point, r is a set such that $|r| \leq n$. We define the following terms:

- n_r: the total number of questions answered in response r;
- q_i: the i^{th} question; and
- o_{r,q_i}: the answer option chosen for q_i.

The score assigned to response r is then

$$\sum_{i=1}^{n_r} -\mathbb{P}(o_{r,q_i}) \log_2(\mathbb{P}(o_{r,q_i}))$$

.

Once SURVEYMAN has computed the scores for every respondent, it classifies outlying scores as bad actors. SURVEYMAN cannot perform outlier analysis on the scores directly. Breakoff and diverging paths through the survey make the scores on each response incomparable.

Instead, for each response r, SURVEYMAN selects the subset of responses that contain r's questions and recomputes the scores for comparison. SURVEYMAN uses the bootstrap method to define a one-sided confidence interval provided by the user (the default is 95%). This confidence interval provides the threshold for classifying respondents as bad actors.

Unlike previous methods, SURVEYMAN's approach is well-suited to capture malicious actors who purposely choose low-frequency answer options. Malicious respondents pose a significant and real threat to validity, especially for web-based survey respondents [32]. Respondents who frequently choose low-frequency options will almost certainly be marked as invalid.

SURVEYMAN's approach is complementary to existing techniques such as catch trials because it focuses its quality control efforts on a different type of behavior. Extant in-survey quality control questions have one feature in common: the expected answer frequencies for valid responses are highly peaked. Heuristics for deciding whether a respondent was paying sufficient attention often focus on these questions alone, ignoring the quality of the respondent's other questions.

SURVEYMAN's approach considers all answered questions in its classification and weighs these highly peaked questions less than questions with more evenly distributed response options. While it might seem counterintuitive to use a metric that discounts the value of catch trials, this approach imposes less of a penalty on good actors who make an occasional mistake. In any event, researchers must use their best judgment when faced with conflicting classifications from multiple quality control sources.

6. Evaluation

We evaluate SURVEYMAN's usefulness in a series of case studies with surveys produced by our social scientist colleagues. We address the following research questions:

- **Research Question 1:** Is SURVEYMAN usable by survey authors and sufficiently expressive to describe their surveys?
- **Research Question 2:** Is SURVEYMAN able to identify survey errors?
- **Research Question 3:** Is SURVEYMAN able to identify random or inattentive respondents?

6.1 Case Study 1: Phonology

The first case study is a phonological survey that tests the rules by which English-speakers are believed to form certain

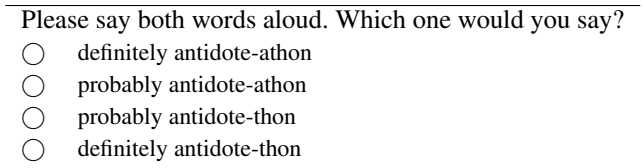


Figure 4: An example question used in the phonology case study (§6.1).

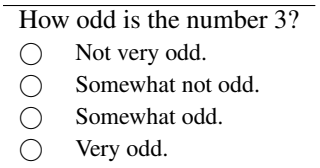


Figure 5: An example question used in the psycholinguistics case study (§6.2).

word combinations. This survey was written in SURVEYMAN by a colleague in the Linguistics department at the University of Massachusetts with limited guidance from us (essentially an abbreviated version of Section 2).

The first block asks demographic questions, including age and whether the respondent is a native speaker of English. The second block contains 96 Likert-scale questions. The final block consists of one freetext question, asking the respondent to provide any feedback they might have about the survey.

Each of the 96 questions in the second block asks the respondent to read aloud an English word suffixed with either of the pairs "-thon/-athon" or "licious/-alicious" and judge which sounds more like an English word. An example appears in Figure 4.

Our colleague first ran this survey in a controlled experiment (in-person, without SURVEYMAN) that provides a gold-standard data set. We ran this survey four times on Amazon's Mechanical Turk between September 2013 and March 2014 to test our techniques.

Static Analysis: This survey has a maximum entropy of 195.32; the core 96 questions have a maximum entropy of 192 bits. There is no branching in this survey, so without breakoff, the minimum, maximum, and average path lengths are all 99 questions long.

Dynamic Analysis: The first run of the survey was early in SURVEYMAN's development and functioned primarily as a proof of concept. There was no quality control in place. We sent the results of this survey to our colleagues, who verified that random respondents were a major issue and were tainting the results, demonstrating the need for quality control.

The latter three runs were performed at different times of day under slightly different conditions; all three permitted breakoff. All three were analyzed using two quality control algorithms: the version described in this paper, and an older version that explicitly modeled positional preference (e.g., always clicking the first option) and random behavior. This contrasts with our current method, which looks for outliers in the distribution of a metric.

We present the results from both algorithms. In all three runs, the current quality control mechanism detected nearly all of the cases detected in the previous version. The cases that the new approach missed were missed were borderline in the old approach.

First run: We used minimal Mechanical Turk qualifications (at least one HIT done with an 80% or higher approval rate) to filter the first survey, so we expected to see fewer random respondents. 26 out of 49 (53%) respondents were classified as bad actors using our active quality control algorithm. The older version found only 6% of the respondents to be bad actors.

Second run: In the second run, we removed the qualification requirements and launched the survey in the morning on a work day. We obtained 116 responses. The older quality control algorithm found that qualifications made no difference: the second run produced similar results to the first. The currently used classifier found that results actually improved: only 50 (43%) respondents were classified as bad actors.

Third run: The third run was launched on a weekend night. The older quality control classifier found approximately 15% of respondents were bad actors due to randomness or positional preference. The current quality control algorithm found 44 out of 98 (about 45%) of respondents to be bad actors.

We believe the high rate of bad actors signals the high cognitive load of the survey. Although some questions have obvious answers, others require more focus. Difficult questions may cause respondents to put less effort into responding. Previous work has shown that survey respondents exhibit this *satisficing* behavior under similar circumstances [18].

Because the survey lacks structure—it consists of only one large block, with no branching, and has roughly uniform questions—we did not expect to find any biases or other errors, and SURVEYMAN did not report any.

6.2 Case Study 2: Psycholinguistics

The second case study is a test of what psychologists call *prototypicality* and was written in the SURVEYMAN language by another one of our colleagues in the Linguistics department; as with our other colleague, she wrote this survey with only minimal guidance from us.

In this survey, respondents are asked to use a scale to rank how well a number represents its parity (e.g., "how odd is this number?"). The goal is to test how people respond to a categorical question (since numbers are only either even or odd) when given a range of subjective responses.

There are 65 questions in total. The survey is composed of two blocks. The first block is a floating block that contains 16

floating subblocks of type ALL; that is, only one randomly-chosen question from the possible variants will be asked. Every question in one of these floating blocks has slightly different wording for both question and options. The other block contains one question asking the respondent about their native language. Because the first block is floating, this question can appear either at the beginning or the end of the survey.

We launched this survey on a weekday morning. It took about a day to reach a total of 149 respondents.

Static Analysis: The maximum entropy for the survey is 34 bits. Since there is no true branching in the survey, every respondent sees the same number of questions, unless they submit early. The maximum, minimum, and average survey lengths are all 17 questions.

Dynamic Analysis: SURVEYMAN found no significant breakoff or order bias, but it did find that several questions exhibited marked wording variant bias. These were for the numbers 463 and 158. The pairs having a statistically significant difference in the distribution of their responses were:

1. *How good an example of an odd number is the number 158?* and *How odd is the number 158?*
2. *How well does the number 463 represent the category of odd numbers?* and *How good an example of an odd number is the number 463?*
3. *How odd is the number 463?* and *How good an example of an odd number is the number 463?*

While these questions may have appeared to the survey author to be semantically identical, they in fact led to substantial differences in the responses.

In addition, SURVEYMAN identified 53 bad actors out of 149 total (35.5%). A visual inspection of the data revealed that many of those classified as bad actors did in fact appear to be answering randomly; most tellingly, they frequently chose the wrong parity for a number.

6.3 Case Study 3: Labor Economics

Our final case study is a survey about wage negotiation, conducted with a colleague in Labor Studies. The original design was completely flat with no blocks or branching. There were 39 questions that were a mix of Likert scales, checkbox, and other unordered questions. The survey asked for demographic information, work history, and attitudes about wage negotiation.

Our collaborator was interested in seeing whether complete randomization would have an impact on the quality of results. We worked with her to write a version of the survey that comprised both a completely randomized version and a completely static version: a respondent is randomly given one of the two versions, depending on how they answer the first question. This approach lets us to collect data under nearly identical conditions, since each respondent is equally likely to see the randomized or the static version.

We launched this survey on a Saturday and ran the survey for approximately a week. We were able to obtain 134 responses. We observed extreme survey abandonment: none of the respondents completed the survey. The maximum number of questions answered was 35.

Static Analysis: SURVEYMAN reports that this survey has a maximum entropy of 80.45 bits. Every path is of length 40: in addition to the original 39 questions and the introductory question that routes respondents, we added an instructional final question to join the two paths.

Dynamic Analysis: SURVEYMAN automatically identified 19 bad actors out of 134 (14.18%). Although this is a low rate of bad actors, every respondent abandoned the survey before completion, giving SURVEYMAN too little information to construct a robust classifier.

The analysis of breakoff revealed both positional and question breakoff. A remarkable 78.4% of the breakoff occurred in the first six positions. Over 20% of the breakoff was in the first question. We believe this was caused by the absence of a "breakoff notice," to inform the respondent that they will be paid commensurate with their work. The base price for this survey was $0.10 USD. 55% of the instances of breakoff also occurred in just four of the questions, shown in Table 6.

These results illustrate the impact of randomization on diagnosing breakoff. Since we expect an equal number of respondents to answer the ordered questions as the unordered, we would expect an approximately equal amount of breakoff at each position. For positions 2 through 4, at least as many respondents who saw the fully randomized survey broke off as those who saw the static survey. A small number of the randomized cohort in these positions broke off at demographic questions. Positions 5 and 6 were dominated by the demographic questions of the static survey. These questions asked about country of origin and current residency and appear to cause breakoff.

6.4 Summary

SURVEYMAN addresses the three research questions stated at the beginning of this section as follows:

Research Question 1: Expressiveness and Usability. Our close collaboration with a variety of researchers in the social sciences led us to design a language that could express a wide range of surveys. SURVEYMAN's blocking, branching, and randomization abstractions form the foundation for this expressiveness. None of the surveys we examined throughout the course of this project were inexpressible in SURVEYMAN. Speaking to usability, one of our collaborators in linguistics found that, "writing the CSVs was easy" [30]. Her work at the time required complex branching and blocking.

Regarding SURVEYMAN's general approach, our collaborator in labor studies said,

Question	Count	Version
Please choose one.	28	All
In which country were you born?	26	Static
In which country do you live in now?	12	Static
In what year were you born?	8	Static

Figure 6: Over half of observed breakoff in the labor economics case study (§6.3) occurred in just four questions.

> I strongly think this speaks to the need to revisit the assumptions that the social sciences have about survey methodology, especially when researchers are operating in online environments [...] what is often a best practice offline does not translate to online tools [17].

Research Question 2: Bugs in the Survey.

- **Breakoff:** We found surprisingly high breakoff in the wage survey. Our colleague from labor studies has used this information to "debug" her survey [17].
- **Wording Bias:** Our colleague from linguistics hypothesized that in the prototypicality survey, all variants would have the same distribution of answers. SURVEYMAN found this to not be true for two sets of variants.
- **Order Bias:** SURVEYMAN found no statistically significant pariwise order bias in any of our case studies. We expected to find order bias in the wage survey, but did not have sufficient samples along the fully randomized path to actually test for it.

Research Question 3: Bugs in the Data. SURVEYMAN was able to identify random respondents and respondents with positional preferences. The SURVEYMAN approach is comparable to a classifier that modeled this behavior explicitly.

7. Related Work

7.1 Survey Languages

Table 4 provides an overview of previous survey languages and tools and contrasts them with SURVEYMAN.

Standalone DSLs. Blaise is a language for designing survey logic and layout [25]. Blaise programs consist of named blocks that contain question text, unique identifiers, and response type information. Control flow is specified in a rules block, where users list the questions by unique identifier in their desired display order. This rules block may also contain data validation rules, such that an Age field be greater than 18. Similarly, QPL is a language and deployment tool for web surveys sponsored by the U.S. Government Accountability Office [44]. Neither language supports randomization or identifies survey errors.

Embedded DSLs. Topsl [19] and websperiment [20] embed survey structure specifications in general-purpose programming languages (PLT Scheme and Ruby, respectively). Topsl is a library of macros to express the content, control flow, and layout of surveys; Websperiment provides similar functionality. Both provide only logical structures for surveys and a means for customizing their presentation. Neither one can detect survey errors or do quality control.

XML-Based Specifications. At least two attempts have been made to express survey structure using XML. The first is SuML; the documentation for this schema is no longer available and its specification is not described in its paper [1]. A recent XML implementation of survey design is SQBL [36, 37]. SQBL is available with a WYSIWYG editor for designing surveys [35]. It offers default answer options, looping structures, and reusable components. It is the only survey specification language we are aware of that is extensible, open source, and has an active community of users. Unlike SURVEYMAN, none of these provide for randomization or error analyses.

7.2 Web Survey Tools

Table 4 lists some of the tools available for deploying web-based surveys. Quite a few of these tools and services have recently added features similar to those provided by SURVEYMAN, such as question randomization. Some of these features are only available in premium versions of the software. None of the services available perform analyses on the quality of the responses, nor do they consider that the survey itself may be flawed.

7.3 Survey Analyses

Despite the fact that surveys are a well-studied topic—a key survey text from 1978 has over 11,000 citations [9]—there has been surprisingly little work on approaches to automatically address survey errors. We are aware of no previous work on identifying any other errors beyond breakoff. Peytchev et al. observe that there is little scholarly work on identifying breakoff and attribute this to a lack of variation in question characteristics or randomization of question order, which SURVEYMAN provides [29].

7.4 Randomized Control Flow

As far as we are aware, SURVEYMAN's language is the first to combine branches with randomized control flow. Dijkstra's Guarded Command notation requires non-deterministic choice among any true cases in conditionals [8].

LANGUAGE	TYPE	LOOPS	QUESTION RANDOMIZATION	ERROR DETECTION	RANDOM RESPONDENT DETECTION	PARTICIPANT POOL
Blaise [25]	Standalone DSL					
QPL [44]	Standalone DSL					
Topsl [19]	Embedded DSL	✓				
websperiment [20]	Embedded DSL	✓				
SuML [1]	XML schema					
SQBL [37]	XML schema	✓				
TOOL						
Lime Survey [39]	Web-based	✓	✓			
Qualtrics [31]	"	✓	✓			
SocialSci [33]	"	✓	✓			✓
instant.ly [15]	"					✓
SurveyMonkey [41]	"		✓			
SurveyGizmo [40]	"	✓	✓			
Google Consumer Surveys [12]	"					✓
SURVEYMAN	Standalone DSL		✓	✓	✓	

Table 4: Feature comparison of previous survey languages/tools and SURVEYMAN.

7.5 Crowdsourcing and Quality Control

Barowy et al. describe AUTOMAN, an embedded domain-specific language that integrates digital and human computation via crowdsourcing platforms [2]. Both AUTOMAN and SURVEYMAN have shared goals, including automatically ensuring quality control of respondents. However, AUTOMAN's focus is on obtaining a single correct result for each human computation. SURVEYMAN instead collects *distributions* of responses, which requires an entirely different approach.

8. Conclusion

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 survey errors and inattentive respondents. Pervasive randomization prevents small biases from being magnified and enables statistical analyses, letting SURVEYMAN identify serious flaws in survey design. We believe that this research direction has the potential to have significant impact on the reliability and reproducibility of research conducted with surveys.

SURVEYMAN is available for download at `http://www.surveyman.org`.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CCF-1144520. The authors would like to thank our social science collaborators at the University of Massachusetts, Sara Kingsley, Joe Pater, Presley Pizzo, and Brian Smith; John Foley, Molly McMahon, Alex Passos, and Dan Stubbs; and fellow PLASMA lab members Dan Barowy, Charlie Curtsinger, and John Vilk for valuable discussions during the evolution of this project.

References

[1] M. Barclay, W. Lober, and B. Karras. SuML: A survey markup language for generalized survey encoding. In *Proceedings of the AMIA Symposium*, page 970. American Medical Informatics Association, 2002.

[2] D. W. Barowy, C. Curtsinger, E. D. Berger, and A. McGregor. AUTOMAN: A platform for integrating human-based and digital computation. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA '12, pages 639–654, New York, NY, USA, 2012. ACM.

[3] A. J. Berinsky, G. A. Huber, and G. S. Lenz. Evaluating online labor markets for experimental research: Amazon.com's Mechanical Turk. *Political Analysis*, 20(3):351–368, 2012.

[4] G. A. Churchill Jr and D. Iacobucci. *Marketing research: methodological foundations*. Cengage Learning, 2009.

[5] A. M. Colman. *A dictionary of psychology*. Oxford University Press, 2009.

[6] M. P. Couper. *Designing Effective Web Surveys*. Cambridge University Press, New York, NY, USA, 1st edition, 2008.

[7] D. De Vaus. *Surveys in social research*. Psychology Press, 2002.

[8] E. W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM*, 18(8):453–457, Aug. 1975.

[9] D. A. Dillman. *Mail and telephone surveys*, volume 3. Wiley New York, 1978.

[10] J. S. Downs, M. B. Holbrook, S. Sheng, and L. F. Cranor. Are your participants gaming the system?: Screening Mechanical Turk workers. In *Proceedings of the SIGCHI Conference on*

Human Factors in Computing Systems, CHI '10, pages 2399–2402, New York, NY, USA, 2010. ACM.

[11] G. Emanuel. Post A Survey On Mechanical Turk And Watch The Results Roll In: All Tech Considered: NPR. http://n.pr/1gqklTx, Mar. 2014.

[12] I. Google. Google consumer surveys. http://www.google.com/insights/consumersurveys/home, 2013.

[13] J. J. Horton, D. G. Rand, and R. J. Zeckhauser. The online laboratory: Conducting experiments in a real labor market. *Experimental Economics*, 14(3):399–425, 2011.

[14] J. L. Huang, P. G. Curran, J. Keeney, E. M. Poposki, and R. P. DeShon. Detecting and deterring insufficient effort responding to surveys. *Journal of Business and Psychology*, 27(1):99–114, 2012.

[15] Instant.ly. Instant.ly. http://instant.ly, 2013.

[16] P. Ipeirotis. Demographics of Mechanical Turk. Technical Report NYU working paper no. CEDER-10-01, 2010.

[17] S. C. Kingsley. Personal Communication, August 2014.

[18] J. A. Krosnick. Response strategies for coping with the cognitive demands of attitude measures in surveys. *Applied cognitive psychology*, 5(3):213–236, 1991.

[19] M. MacHenry and J. Matthews. Topsl: A domain-specific language for on-line surveys. In O. Shivers and O. Waddell, editors, *Proceedings of the Fifth ACM SIGPLAN Workshop on Scheme and Functional Programming*, pages 33–39, Snowbird, Utah, Sept. 22, 2004. Technical report TR600, Department of Computer Science, Indiana University. http://www.cs.indiana.edu/cgi-bin/techreports/TRNNN.cgi?trnum=TR600.

[20] G. MacKerron. Implementation, implementation, implementation: Old and new options for putting surveys and experiments online. *Journal of Choice Modelling*, 4:20–48, 2011.

[21] E. Martin. Survey questionnaire construction. Technical Report Survey Methodology #2006-13, Director's Office, U.S. Census Bureau, 2006.

[22] W. Mason and S. Suri. Conducting behavioral research on Amazon's Mechanical Turk. *Behavior research methods*, 44(1):1–23, 2012.

[23] A. S. McKay. Improving data quality with four short sentences: How an honor code can make the difference during data collection. 2014.

[24] A. W. Meade and S. B. Craig. Identifying careless responses in survey data. *Psychological methods*, 17(3):437, 2012.

[25] S. Netherlands. Blaise : Survey software for professionals. http://www.blaise.com/ShortIntroduction, 2013.

[26] D. M. Oppenheimer, T. Meyvis, and N. Davidenko. Instructional manipulation checks: Detecting satisficing to increase statistical power. *Journal of Experimental Social Psychology*, 45(4):867–872, 2009.

[27] Pew Research Center. Question Order | Pew Research Center for the People and the Press. http://www.people-press.org/methodology/questionnaire-design/question-order/, 2014.

[28] Pew Research Center. Question Wording | Pew Research Center for the People and the Press. http://www.people-press.org/methodology/questionnaire-design/question-wording/, 2014.

[29] A. Peytchev. Survey breakoff. *Public Opinion Quarterly*, 73(1):74–97, 2009.

[30] P. Pizzo. Personal Communication, August 2014.

[31] I. Qualtrics. Qualtrics.com. http://qualtrics.com, 2013.

[32] J. P. Robinson-Cimpian. Inaccurate estimation of disparities due to mischievous responders several suggestions to assess conclusions. *Educational Researcher*, 43(4):171–185, 2014.

[33] SocialSci. Cambridge, ma, usa. http://www.socialsci.com, 2014.

[34] D. J. Solomon. Conducting web-based surveys, August 2001.

[35] S. Spencer. Canard question module editor. https://github.com/LegoStormtroopr/canard, 2013.

[36] S. Spencer. A case against the skip statement, 2013. Unpublished.

[37] S. Spencer. The simple questionnaire building language, 2013.

[38] S. S. Stevens. On the theory of scales of measurement. 1946.

[39] L. Survey. Lime survey 2.05. https://www.limesurvey.org/en/, 2014.

[40] L. SurveyGizmo. Surveygizmo. http://http://www.surveygizmo.com/, 2014.

[41] SurveyMonkey, Inc. Surveymonkey. http://surveymonkey.com, 2013.

[42] R. Tourangeau, F. Conrad, and M. Couper. *The Science of Web Surveys*. Oxford University Press, 2013.

[43] P. D. Umbach. Web surveys: Best practices. *New Directions for Institutional Research*, 2004(121):23–38, 2004.

[44] U.S. Government Accountability Office. Questionnaire programming language. http://qpl.gao.gov/qpl6ref/01.php, 2009.

[45] D. Zhu and B. Carterette. An analysis of assessor behavior in crowdsourced preference judgments. In *Proceedings of the SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation (CSE 2010)*, pages 21–26, 2010.