Building behavioral experimentation engines

Yun-En Liu

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2015

Reading Committee:

Zoran Popović, Chair

Emma Brunskill, Chair

Steve Tanimoto

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington

Abstract

Building behavioral experimentation engines

Yun-En Liu

Co-Chairs of the Supervisory Committee:
Professor Zoran Popović
Computer Science & Engineering

Assistant Professor Emma Brunskill
Computer Science, Carnegie Mellon University

Human behavior is an incredibly complex topic, given the variation between individuals and the many ways we can be influenced by our environment. This complexity, combined with the difficulty and expense of running experiments involving humans, means there are still many aspects of how we react and learn that are poorly understood. This thesis argues that the rise of online software and new machine learning algorithms has given us a new way to study these vast behavioral topics. First, by designing software people want to use, we can collect data much faster and more cheaply than ever before. Second, by identifying the objectives scientists implictly maximize when choosing experiments to run, we can invent new algorithms to maximize for these objectives directly by automatically altering the software and measuring user responses. Third, these algorithms must take into account difficulties in implementation not present in the laboratory environment, where users are paid to participate.

Four examples of algorithms, designed to automatically run experiments on online software to fulfill different scientific objectives, are presented. These algorithms can maximize for subject outcomes such as learning while discovering which factors most influence their performance, efficiently discover the most general form of scientific results and intelligently follow many simultaneous chains of research, identify and sample experimental conditions which yield the most surpris-

ing user behavior, and respect the implicit tradeoff between uncovering new scientific knowledge and providing good outcomes for users. Finally, no algorithm can make progress if the software source fails to collect useful data, so a case study of how to design new software deployment methods to collect better-quality data on algebra learning and social incentives is presented.

ACKNOWLEDGMENTS

Thank you to my advisors, Zoran Popović and Brunskill. Thank you to my collaborators and fellow graduate student friends. Thank you to my parents, Pao-Lo Liu and Shaw-Lin Han, and the rest of my family. Together, you have given me a truly valuable gift: the belief that I can do anything.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Imagine we could accurately predict how any individual person would respond to a change in their environment. Such knowledge would enable us to develop computational tools and design institutions and systems to enable people to realize their greatest potential. For example, knowing how much students would learn from intervention A compared to intervention B would allow us to optimize their educational outcomes. As such, solid understanding of human behavior has long been the primary goal of many fields of research which I collectively refer to as the behavioral sciences.

Unfortunately, this goal is difficult to achieve. Humans are expensive to study, as they must be recruited, and there are many differences between people that must be accounted for. Big data collected from electronic records is no panacea, as confounding variables and fundamental correlation-causation problems mean that the conclusions we draw may not generalize to new scenarios. Even worse, many questions are nearly impossible to answer in the laboratory: researchers cannot run randomized experiments on stock return rates because they do not control the stock market, for instance. Surpassing these limitations requires tremendous volumes of data to try many variations of interventions on many different types of people, as well as the design of new experimental methodologies or systems that enable researchers to run experiments that have previously been impossible.

My approach relies on the rise of online software such as games or e-commerce websites like Amazon, which has given us the key piece to the puzzle: the ability for computers to rapidly, cheaply, and automatically run experiments with a wide array of subjects on large-scale online systems. In my thesis, I argue that we can tackle these challenges using online software (to give us large data streams), combined with algorithmic active experimentation (to prove causation and use

data efficiently). To do so, I build software capable of generating data on the subjects we wish to study, identify the objectives scientists have when they run experiments, and use machine learning algorithms to automatically maximize for these objectives by altering this software and measuring changes in user responses. These systems are what I refer to as behavioral experimentation engines. Taking full advantage of this new way of running experiments is not trivial, however. This thesis tackles multiple challenges:

1. The most efficient way to make use of online data is not to run extremely large experiments, such as running an A/B test with many users in each condition, because this does not solve the original problem of inability to deal with large factor spaces. A better solution is to run rapid individual experiments, with each sample going to the condition that maximizes some useful scientific objective as data is collected, instead of being allocated a-priori. This requires automation, as researchers cannot make thousands of experimental decisions per day. I suggest several scientific objectives and matching optimization criteria, which allow the algorithms I develop to efficiently allocate user samples to advance our knowledge in the behavioral sciences.

2. Using online software to conduct experiments results in new classes of problems that are not normally major concerns in the laboratory setting, which I examine and address. For example, data collected from users may be delayed or dropped, or the owner of the software may not allow researchers full control over which conditions can be given to which users. These considerations affect the sampling process and can drastically degrade performance or bias results if not accounted for. Furthermore, many of these environments can be considered high-stakes - if running experiments on an e-commerce platform costs a company money, that company has no incentive to allow the research to continue. Thus, I develop new sampling methods capable of honoring this fundamental tradeoff.

3. The easiest types of software to deploy treat users as all being similar coming in, interacting with the system briefly, and then leaving. Mechanical Turk or online games follow this

model. Yet many interesting questions cannot be answered under these assumptions. For example, do boys or girls have a greater 6-month retention rate when learning fractions in a group setting? In most online research environments, we do not know users' gender, cannot track them for 6 months, and cannot easily arrange them into collaborative groups. No clever sampling algorithm can overcome this limitation; instead, clever design of new software or software delivery mechanisms is needed in order to collect data about such questions. In this thesis I describe one such new delivery mechanism, the educational campaign, which allows us to answer social incentive and demographic questions in the domain of algebra education.

The remainder of this thesis describes related work in the many areas of research from which I draw, followed by a report on five of my different projects that tackle at least one of these challenges. Four of these projects are new algorithms for automatically running experiments with different scientific objectives and constraints: automatic hierarchical experimentation to run A/B tests in a staged greedy fashion, automatic generalization to find how far an experimental effect generalizes in certain scientific domains, sampling for scientific surprise to efficiently sample conditions whose results do not match expectations, and trading off between reward and scientific knowledge with a new multi-armed bandit technique. The last project is a case study of a new mechanism for delivering educational technology, giving us access to a new data source with which we can run experiments. Many limitations and challenges remain for individual projects, which I will describe in each chapter. I will then conclude and discuss broader future directions for the behavioral experimentation engine approach in the final chapter.

Chapter 2

RELATED WORK

Building behavioral experimentation engines involves many techniques from many fields. Research from human-computer interaction, games, psychology, and education can help us build more desirable, usable software with which we can gather more data. At the same time, machine learning and statistics can help us direct our expensive experimental samples to the best conditions. Here, I will cover general related work that applies to the overarching thesis. Project-specific previous work will appear in individual chapters as needed.

2.1 Massive online experimentation

Major companies such as Microsoft [56] and Amazon [57] have performed online experiments for years. For researchers, Mechanical Turk has proven to be quite helpful to many scientists looking for cheap, high-quality user data [53]. Games have also become an increasingly popular source for behavioral data, and have been used to study the effects of optional rewards [6] and tutorials [8].

These web-based mass experimentation platforms have both benefits and drawbacks relative to standard laboratory experiments. Mechanical Turk has been shown to provide inexpensive, reliable results and has a demographic spread much wider than typical pools for social science research [24]. In the games domain, the observed behavior is "in the wild" [7], increasing external validity [110]. Experiments run online can have many more users, and better measure true task engagement; however, it may be difficult or impossible to collect rich data such as demographic or interview data [110] or control experimental setting. Additionally, simply running larger experiments grants us greater statistical certainty but does not escape from the combinatorial explosion of possible interventions as we consider more and more factors (parameters about some system researchers are interested in studying, which can be set in many ways). My algorithms are designed

to efficiently search through such combinatorial spaces efficiently by using all previously-collected data to inform each new choice.

2.2 Scientific Discovery

Researchers in AI have been working for years to develop systems capable of generating scientific knowledge. This field, known as scientific discovery, has generated many such systems aimed at automating different scientific behaviors [61]. For example, Lee et al. used a feedback loop between the RL rule induction program and expert knowledge to identify potentially carcinogenic compounds [63]. Perhaps the most comprehensive example of such a system is Robot Scientist Adam, a fully automated robot capable of the full loop of hypothesis generation, experimental design, and data analysis in yeast genomics [52].

I am most interested in automatically understanding human behavior, which introduces many problems not present in a laboratory setting and not specifically dealt with by scientific discovery researchers. For example, in some settings (ex. education) there are many experimental variables: instruction duration, number representation, ordering of concepts, problem type, hinting systems, etc., while the number of variables is generally much smaller in laboratories. As another example, I need to know how to incentivize users to interact with my systems in order to get more data, while yeast cells have no such preference. In addition, I often want to both find general rules about how different factors affect student learning, as well as the specific settings that optimize rate of learning and maximum transfer ability. This multi-objective theme will recur throughout this thesis, and necessitates different sampling strategies compared to simply maximizing scientific information in some sense.

2.3 Educational Data Mining

Several of my application domains involve running experiments to identify factors contributing to variation in learning, a common theme in the educational data mining community. Intelligent tutoring systems, especially cognitive tutors [9], have been used for many experiments: for example, Rau et al. [89] test the usefulness of multiple fraction representations with self-explanation

compared to single representations. Or in the educational games domain, Lomas et al. [74] used an educational game to run two large-scale experiments with many conditions on the effect of challenge on motivation and learning.

I am not merely proposing a new tutor or game and then using it to run standard large-scale educational experiments. Instead, I wish to develop methods that gather or use existing data from some source (such as a tutor or game) to automatically run experiments with many factors, ideally with applications for all behavioral sciences. To the best of my knowledge, other researchers in this field have not proposed or used an automatic method for choosing and running experiments in a hypothesis space. Furthermore, they have not considered several additional problems that occur once moving to the online domain that I address, such as using importance sampling to run multiple experiments on the same dataset with a different sampling distribution than desired.

2.4 Active Learning

Active learning is a branch of machine learning concerned with methods for requesting new samples in order to best improve prediction performance of a given model. This is especially useful when labeled data is expensive to come by [100]. Many types of data meet this criteria: for example, annotation of speech [124], information extraction for text [101], and cancer diagnosis [69] are all expensive. Active learning aims to choose samples which minimize future classification error. This is not generally tractable, so approximations are used, such as uncertainty sampling (choosing the datapoint of which the classifier is least certain) [66] or expected error reduction (minimizing predicted generalization error) [98]. This method has been less widely applied in the behavioral science domain, but one example comes from Lindsey et al., who used Gaussian process regression and an active learning framework to reduce the number of samples required to learn how users responded to different teaching strategies [68].

2.4.1 Optimal Experimental Design

Active learning is a highly general technique. A related subclass is the field of optimal experimental design, which aims to collect samples (run experiments) to reduce expected error, such as in the text-classification framework proposed in [97]. Bayesian experimental design, a subfield of optimal experimental design, is the most closely related to my work. It chooses the experimental design yielding the most information about a parameter vector θ [27], such as the probability of various drugs inducing a positive response. This is difficult to do when data is to be gathered from unpredictable sources such as humans, since estimation of information gain on θ depends on the data collected, and estimating the data to be collected from an experimental design requires estimating how humans will react to it. Rafferty et al. tackle this problem in the context of designing a simple educational game by modeling subject behavior with Markov decision processes [88]. An additional difficulty arises in our domain: we would like to take advantage of data gathered from previous subjects when choosing the experiment for the next subject. This *sequential design* should lead to some gains in nonlinear problems [27], and has been studied in special cases for nonparametric binary regression [59] and in the batch setting for computing seed infection rates [95].

These techniques are a useful addition to a researcher's toolkit; in this sense, I do not claim that my methods are better. However, bayesian experimental design cares about specifically gathering information to minimize error in beliefs about parameters. I believe there are several other criteria that are often of more interest to scientists, such as discovering how general a result is or collecting data about surprising conditions. In these cases, the optimization criteria is different, and so I develop new sampling methods to maximize the new criteria. As one simple example, if a researcher cares more about identifying conditions in which theory fails to predict how humans actually behave, then "surprising" conditions ought to receive more samples than "unsurprising" ones. Yet bayesian experimental design cares equally about estimating parameters of all conditions, and so will tend to spread out samples much more evenly.

2.5 Multi-Armed Bandits

As they will be important later on, a brief introduction of multi-armed bandits is in order. Multi-armed bandit problems consist of K probability distributions, $D_1, \ldots, D_K$, with expected values $\mu_1, \ldots, \mu_K$. The D_i and μ_i are not known at the start. These distributions are classically viewed as wins or losses (1 or 0) from various arms of a slot machine, but in continuous formulations can be any (bounded) user-defined function. In our scenario, we can think of the arms as different types of numberlines to be given as practice, and the reward as player success on a randomized test number line.

In the most common bandit formulation, the experimenter tries to pull arms in order to collect as much reward as possible (e.g. assign players to conditions to maximize test performance). At each turn, $t = 1, 2, ...$, we select an arm $j(t)$ and receives some reward from that arm's reward distribution $r(t) \sim D_{j(t)}$. If the D_i were known, the optimal strategy would be to choose the arm with the highest expected reward, $j(t) = \arg\max_i \mu_i$: that is, pick the most effective number line and give it to every player. Unfortunately, the D_i and μ_i are hidden. Successful bandit algorithms must navigate an *exploration-exploitation* tradeoff to discover information about the D_i while also generating high reward. In other words, successful algorithms must generate reward by pulling arms known to be good (exploitation), while also trying new arms in case they give even more reward (exploration).

In general, we will not be able to give everyone the best intervention. Define the *total expected regret* at some fixed timestep T as the loss of reward from playing non-optimally, $R^T = T \max_i \mu_i - \sum_{t=1}^{T} \mu_{j(t)}$. Lai and Robbins show that regret must grow at least logarithmically in time [60], developing a lower bound of $R^T = \Omega(log(T))$.

Not all strategies that work well in practice meet this bound. One such heuristic strategy is ϵ-greedy, which for any $0 < \epsilon < 1$ plays a random arm with ϵ probability and otherwise plays the arm with the highest empirical mean. This strategy has linear regret because it has a constant chance to play suboptimal arms. One could consider allowing ϵ to decrease over time to eliminate this linear regret term; however, this adds another parameter and does not always help in practice

[115].

A different, popular class of theoretically-motivated strategies which meet the logarithmic regret bound are the Upper Confidence Bound strategies (UCB) [13]. These algorithms exemplify the principle of *optimism under uncertainty* by pulling the arm which has the highest estimated upper confidence bound on its mean. The simplest, UCB1, works in the following way. Assume that all rewards are in the range $[0, 1]$. To initialize, pull each arm once. Then at each subsequent timestep, if the number of times an arm i has been pulled is n_i, choose the arm $j(t) = \arg\max_i \hat{\mu_i}^t + c\sqrt{\frac{2 \ln t}{n_i^t}}$. In this formula, the exploitative first term is the estimate of the arm mean, while the exploratory second term represents an uncertainty that grows slowly as other arms are pulled but decreases sharply when this arm is pulled. UCB1 provably incurs logarithmic regret when $c = 1.0$, though c is often set to be smaller for better empirical performance. For other algorithms, see [23].

Multi-armed bandits and contextual bandits are quite common in online settings. For instance, they are often used to maximize click through rates of search results or article recommendations [67]. Yet their structure is designed to maximize a reward metric such as revenue. In contrast, I care about identifying scientifically interesting results. As such, my algorithms have different aims and assign samples differently. This difference is particularly stark in the case that there is a condition with unexpectedly poor performance: an algorithm aimed at optimizing reward would avoid this condition, while I would prefer to sample it so as to learn more about it.

2.6 Bayesian Optimization

Bayesian optimization is a subfield of general optimization that constrains the underlying function with priors which are updated as samples are drawn. They generally trade off exploration and exploitation to reduce the number of necessary queries. GP-UCB, for example, optimizes black-box stochastic functions [109] using a UCB-like approach, which is arguably the most pure expression of exploration-exploitation. Thompson Sampling [112] is another bayesian technique most classically used in multi-armed bandit problems, in which arms are pulled in proportion to their probability of being optimal. This posterior sampling can be applied to a wide range of

problems, however, such as navigating exploration-exploitation tradeoffs in Posterior Sampling for Reinforcement Learning [82]. Bayesian optimization has even been used for parameter tuning for large classes of machine learning algorithms using Gaussian Processes to model the algorithm's performance [107].

Bayesian optimization is useful for identifying function optimization, such as finding the single most effective educational intervention. However, for the same reason as described in the multi-armed bandit section, it is often desirable to not simply maximize a function. Indeed, if the goal is to create new scientific knowledge, researchers would rather learn about the shapes of response functions as opposed to identifying the "best" point. It is not even clear what "best" would be in many cases: there is no ideal response time for multiple object tracking task, for example, since the task is not trying to train a skill but is rather meant to uncover information about how the brain works. That being said, the bayesian approach implicitly tries to learn general function shapes as posteriors in its goal to identify optimal points, and these posteriors could potentially serve as useful information for scientists. What types of priors and posteriors are appropriate for human response functions is an open question, however, so it is far from clear how to apply bayesian optimization methods to build experimentation engines.

Chapter 3

AUTOMATIC HIERARCHICAL A/B TESTING

3.1 Introduction

One of the most common methods designers and researchers use to improve software is through A/B testing [56, 57, 6, 8]. In this method, two or more variants of software are presented to users, and some resulting behavior of interest is measured (dwell time, click-through rate, money spent, etc.). Usually the designer chooses the best variant, then repeats with new versions of the software.

In this chapter, which is based off my published work [71], I introduce an automated hierarchical experimentation framework to address two problems with A/B testing:

1. There may be many possible factors that could vary between different versions of the software. For example, in an educational game, there are many ways to display any given fraction; at the same time, there are many types of hints that could be given if a student fails to solve a problem. As the number of factors increases, the number of possible software variants grows combinatorially: one for each combination of settings for all factors. Even Internet-scale data is insufficient to explore all combinations, so a search strategy is required.

2. Researchers often do not have full control over how software is presented. For example, a website's owner may be reluctant to allow many users to see potentially bad versions of that website. In that case, they may only allow a small percentage of users to participate in certain experimental conditions. In the extreme case, we may want to analyze already-collected data offline.

The basic version of my framework deals with the first problem by performing a greedy search over a (possibly large) space of potential factors. It runs series of A/B tests in succession, identifying the most broadly good factor setting available at each step before considering the remaining

factors. The advanced method enhances the basic version with importance sampling to handle the case where researchers lack full control over which versions of the software are distributed to each person. There are several themes in this section which will recur throughout this thesis: collecting small amounts of data from each subject across many subjects to learn about the population (though at the expense of assuming all users are alike), intelligently choosing experimental samples in a combinatorial space of possible variations, and needing to alter or improve methods to deal with real-world constraints that prevent us from having full control over sample distribution.

3.2 Treefrog Treasure

The methods presented in this thesis are applicable to many forms of online software. A common running example will be the educational game Treefrog Treasure, however, so we introduce it here. Treefrog Treasure is a game in which the player controls a frog character, who jumps through a jungle world and solves number line problems to reach an end goal. The player must navigate sticky, bouncy, and slippery surfaces and avoid hazardous lava to win successively more complex levels. Number line problems serve as barriers that the player must solve by hitting the correct target location, as shown in Figure 3.2. It has been played by over 5 million players worldwide as of September 2013 [71].

We chose number lines in general as they are a popular pedagogical tool, and a fair amount of evidence exists suggesting that much whole and rational number knowledge is organized around a mental number line [10], [103]. Many of our experiments will involve modifying how the number lines are presented, in search of which types of number lines cause users to answer future number line questions correctly. These modifications can be found in Table 3.1 and are visually represented in Figure 3.3, and will be described in more detail later.

3.3 Basic Framework

We first present a simplified version of our framework, which requires full control over how users are sampled and is designed to run one experiment at a time. We ground our discussion in the following example: investigating how varying the presentation of (fraction) number lines affects

Figure 3.1: A screenshot of Treefrog Treasure, from which we will draw much of our experimental data. Players navigate through a physics-based world, solving number line problems along the way. Notice that the number line has full tick marks, pie chart labels on the line, and a symbolic (ex. $\frac{a}{b}$) target representation. In our experiment, these are a few of the parameters we allow our system to automatically explore to determine which types of number lines lead to maximal near-transfer.

users' ability to answer future number lines. Fraction number line problems typically take the form of a line representing the reals, with at least two points marked for scale, and ask where new fractions should be placed or what fraction corresponds to some point. In the next section, we will extend this framework with importance sampling to allow us to run multiple analyses on the same dataset, and then present an implemented version of it that was able to automatically run parameter searches to find which types of number lines lead to maximal near-transfer to new number line questions, using data from the online educational game Treefrog Treasure.

First, let's consider how a researcher might run an experiment.

1. The researcher hypothesizes about how one or more factors might impact a variable of interest, e.g. different fraction representations or hinting systems might lead to different performance on future number lines.

2. If trials are expensive, she cannot test more than a few of these factors simultaneously. Instead, she must decide which factors are most interesting: perhaps she compares a few different representations with each other, with no hinting systems.

3. She decides on an experimental procedure to test the role of this factor on the variable of interest, including whatever assessments are necessary. Here, this will likely involve the choice and refinement of existing number line tests, whether or not the players should participate online or come into the lab, and so on.

4. She runs the experiment, then collects and analyzes data.

5. Assuming the results become well-established and accepted within the research community, eventually another researcher may pick other factors to investigate, holding the already-studied ones constant. For example, perhaps representation A is better in the initial experiment. Then the next experiment might test the effects of different hinting systems for number lines with A representations.

From a research perspective, perhaps the most interesting part of this process is data analysis and hypothesis generation. The rest is needed to both gather data confirming or rejecting the hypothesis, and to ensure that there is sufficient statistical power even with a small number of subjects.

However, let's assume now that we have a constant stream of new users: say, several thousand per day. Furthermore, let's assume that they are interacting with a system under the experimenter's full control and are willing to participate in the experimental conditions and take any assessments

necessary. Then, there is no difficulty in finding subjects, and the experimenter does not have to be present to run experiments. Furthermore, with so many users and control over the assignment of players to conditions, we can test many experimental conditions, not just a handful. Thus, the experimenter need not carefully select factor levels: she can simply specify the available factors and the system can explore them to identify the ones that seem most informative. Finally, statistical analysis becomes easier with clean experimental designs, though as we will show later we can continue to operate even if we do not have full control over which interventions are chosen.

Of course, the total number of experiments possible grows exponentially with the number of factors, so it is necessary to choose a search strategy. In this chapter, we propose a greedy search: at each step, we choose a parameter and its setting that leads to the best performance when randomly selecting other parameters, then proceed to recursively resolve the remaining parameters. This leads to selecting parameters which are broadly effective at the top of the experiment tree (due to the randomization of other parameters), but quickly narrows (due to the greedy setting of parameters with each step). This strategy is appropriate in a domain such as education where we both want to create generalizable knowledge about which dimensions of the parameter space are most effective, and also optimize some metric like learning gain. In other domains, such as psychology, the goal may simply be to find the factors that cause the largest differences between settings and so a different search strategy may be needed.

This search strategy is detailed in Algorithm 1. P_e and P_g refer to parameters and their associated settings which have not yet been set; the difference between them is that the *generalization parameters* P_g are always set randomly while the algorithm explicitly tests different settings of parameters in the *experimental parameters* P_e. For example, a P_e parameter might be "type of fraction representation" and therefore worth studying. In contrast, a P_g parameter might be "fraction denominator:" we would like to know that certain fraction representations are better or worse for many different fraction denominators, but are not interested in how the choice of denominator affects student learning and so it is always randomly chosen. The generalization power of most standard studies is often both implicit and minimal, in the effort to control as many variables as possible. But in our framework it is made explicit, and as we will see later can reasonably be quite

Algorithm 1 Hierarchical Experimentation Algorithm

function SEARCH(P_e,P_g,P_f,N)

▷ P_e is a map from experimental parameters to many settings each

▷ P_g is a map from generalization parameters to many settings each

▷ P_f is a map from already-fixed parameters to one setting each

q = empty PriorityQueue

for $p, settings \in P_e$ **do**

for $s \in settings$ **do**

$newP_e = P_e - p$

$newP_f = P_f + p \rightarrow s$

r = empty Queue

for N users **do**

$P =$ RANDOMIZE$(P_e \cup P_g) \cup P_f$

$o =$ EXPERIMENT(P)

r.PUSH(o)

q.INSERT$((newP_e, newP_f, P_g)$,average of $r)$

while q is not empty **do**

$(newP_e, newP_f, P_g)$ = delete max from queue

SEARCH$(newP_e, newP_f, P_g)$

end function

function RANDOMIZE(P) ▷ Chooses a randomly select setting of each parameter

end function

function EXPERIMENT(P) ▷ Test a user with parameter settings P; return an objective value

end function

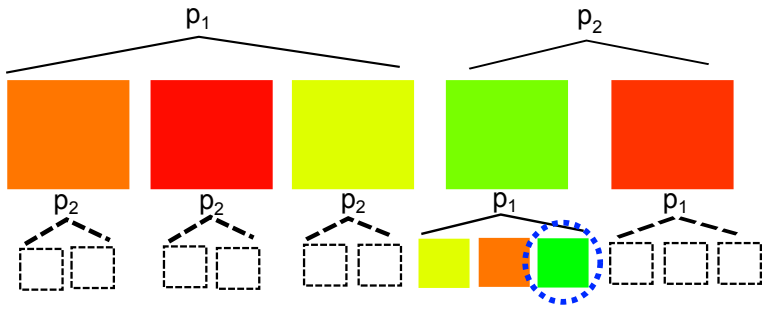


Figure 3.2: An illustration of the experimental tree built by our algorithm with two parameters, p_1 and p_2. Our algorithm first explores all individual parameter settings at the top of the tree, holding one parameter fixed to some value and randomizing over the other parameter. It orders the nodes by function value and recurses, so that we hold p_2 to its best setting and vary p_1. Since the algorithm can be terminated at any time, in this work we stop once all nodes have been set for the first time, though with an exponential number of users the algorithm could explore all nodes.

large, since we can muster so many players.

Our algorithm is intuitively simple to understand. Its goal at each step is to order the parameters and associated parameter settings by how broadly positive of an impact they have, marginalizing over all remaining parameters in the parameter space. It then sets the best parameter to its best setting and repeats the process with all remaining parameters until it hits the bottom and has to backtrack to the next best parameter setting. Given enough players, the algorithm will eventually test all experimental conditions.

The algorithm can be stopped at any point, giving the experimenter a valid experimental tree and the current-best node, B. A common stopping choice might be to have it stop once it reaches the bottom for the first time, resulting in the greedy selection of good parameter settings. In this case, if there are M experimental parameters and they each have at most K settings, the number of experiments the system will run is $O(KM^2)$. Unfortunately, if there are particularly nasty interactions between parameters, nothing short of a full search of the experiment parameter space and its associated $O(K^M)$ runtime is guaranteed to find the globally optimal setting. An easy solution is for the experimenter to combine parameters likely to interact into a single parameter with many settings. Or, if we assume that there are no more than J-way interactions between

parameters, we could allow the algorithm to explore all combinations of parameters of size J at each level. This reduces the number of recursion steps to $\frac{M}{J}$, but requires $\binom{M}{J}$ experiments per step, resulting in approximately $O(K^{\binom{M}{J}}(\frac{M}{J})^2)$ experiments. N, the number of players assigned to each condition, must be chosen carefully to make good decisions in reasonable time.

3.4 Full Framework with Importance Sampling

In this chapter, we are primarily interested in using games and online learning software. In online game environments, we have the advantage that trials are inexpensive, but the disadvantage that we may not have full control over our player sampling process. For example, game design constraints may make it difficult to give highly randomized interventions: a game with completely random levels may not be very fun to play. We also wish to be able to function in an offline setting in order to re-use existing datasets, and in some cases because it is difficult to choose parameters for each new player in an online fashion.

We deal with both problems by extending our basic framework using importance sampling. Importance sampling is a commonly-used technique (ex. [47]) that allows us to estimate an expected function value from a desired distribution of the arguments, even though we can only sample from a different distribution of the arguments. This is accomplished by weighting our function evaluations. Specifically, let $f(x)$ be the objective function, $p(x)$ the desired distribution, and $q(x)$ the actual distribution. Then $\mathbb{E}_p[f(x)] = \mathbb{E}_p[f(x)\frac{q(x)}{q(x)}] = \mathbb{E}_q[f(x)\frac{p(x)}{q(x)}]$. The last quantity is one we can estimate from data with $F_q = \frac{1}{N}\sum_{x \in X_q} f(x)\frac{p(x)}{q(x)}$, as long as we know both the sample and target distributions, and gives us an unbiased and consistent estimator. This technique allows us to run our full framework in offline situations with soft constraints on what interventions we can give to players, assuming that the dataset has non-zero probability for all possible settings of the experimental parameter set.

3.5 Evaluation

In this chapter, we want to maximize player learning in our game. There are two challenges. The first is that players may quit at any time, so that an intervention may appear to be better just

because it causes the least able players to quit. We address this by having short interventions and assessments, and assigning a score of 0 to players who quit before reaching the assessment. The second challenge is that we need to be able to measure player knowledge. This is actually a major challenge: imagine the number of players we would lose by embedding a paper-and-pencil test in a free online game. In our implementation, we mitigate this problem by both embedding the test in the game itself, and only giving players a single question. Such short assessments lead to an increase in noise of the objective function (whether or not students are able to answer the assessment question correctly), but this noise is offset by the large number of players we have. In other scenarios, longer tests may be a better choice.

This approach is not only specific to our application. Indeed, we can do this population average whenever we are interested only in comparing expected assessment scores between different experimental conditions. At any particular stage, the population we are measuring, X_k, consists of players who were directed into some particular condition C_k whose efficacy we wish to measure. The expected test score F_{C_k} for any player $x \in X_k$ for our randomized test is obtained by simply averaging over the test scores that we observe over all players, as long as we assume players are sampled independently and identically distributed (from X). This approach bears some similarity to the one taken in domain sampling theory [32], one of the classical testing theories from psychometrics. We avoid many complications because we do not need to estimate single-user scores, only population-level ones, and our choice is justified because we are sampling directly from the population of interest.

To get a sense of how reliable our results are, we would like to establish confidence intervals for our assessment objective function. This is a non-trivial task, given that there is no a-priori knowledge of the objective function distribution, and we re-weight our samples with importance sampling. If we assume a stationary distribution of player scores, we can use a general resampling method known as bootstrapping [38], which repeatedly samples from our empirical data and calculates a test statistic on these resampled batches to estimate quantities relating to the original, unknown distribution. In our case, the test statistic is the mean, and we are interested in obtaining 95% confidence intervals of the mean. Since we have no guarantee of the symmetry of our sample

mean around the true mean, we use the centered bootstrap percentile method [105]. Our framework does not depend on the method of calculating confidence intervals, however, so for certain classes of objective functions it may be considerably faster to calculate these intervals with closed-form solutions or more intelligent sampling methods.

3.6 Example implementation

Now that we have described our general automatic experimentation framework, we demonstrate its power with a full implementation in a specific setting, along with experimental results. Our platform is an educational game, with players gathered from a popular Flash game website targeted at schoolchildren and teachers [20]. Taken together, our importance sampling method and our randomized assessments over populations will allow us to run the full system on a 2 x 2 x 4 x 4 experimental parameter space on a data set collected previously for a different use. This will allow us to discover what number line properties are most likely to lead to increased player ability to solve a second, randomized test number line.

3.6.1 Experimental Design

We will be using data from Treefrog Treasure, described in Section 3.2. Our dataset was collected from June 3, 2013 to June 20, 2013. Players went through several tutorial levels before reaching the experiment levels. After cleaning our data of users with dropped data and who failed to finish the tutorial and never made it into full game play, we determined that we had 34,197 players. This data set was originally collected for a different purpose, but since our full framework allows us to run our algorithm even on data sets that were collected according to a different sampling distribution, we can simulate what would have happened had we run our algorithm online (assuming either we have a large amount of data or the original sampling distribution is close to the one our algorithm would have chosen). This allows us to make maximal use out of already-collected data, an important property since human data is relatively expensive to come by compared to simulation data.

In this chapter, we wish to use this data set to find the type of number line that leads to greatest

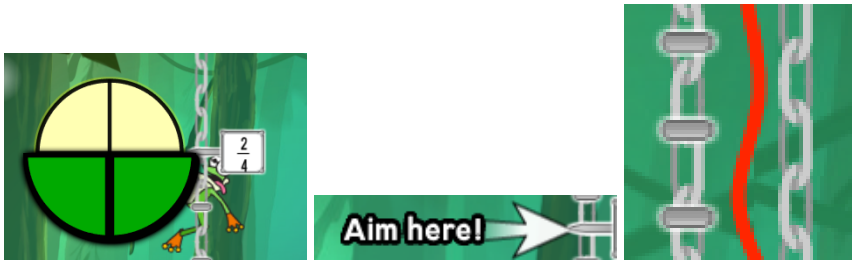


Figure 3.3: The animation condition on the left shows the player how to divide up the number line. The backoff condition in the middle fills in labels and eventually tells the player where to hit. The ticks condition on the right either divide up the number line into segments when ticks are present, or leave it empty besides the 0 and 1 labels when ticks are absent.

player performance on a randomized test number line. To do this, we consider each player's number line solving attempts as a sequence of many pairs of number line solving attempts, and treat each pair as an experimental unit. For example, if a player encountered number lines A-B-C, we would treat this player as two separate pairs: A-B and B-C. This gives us 361,738 pairs. This violates certain assumptions about the independence of variables in classical statistical tests, but greatly increases the amount of available data. We will strictly adhere to the correct assumptions when we validate our results on a new dataset, later, and we will see that our major results continue to hold.

Our experimental parameters alter the way number lines are presented, and can be seen in Table 3.1. We chose these parameters because they have all been the subject of previous research and are subject to varying amounts of controversy. Tick marks may allow students to find fractions directly through a double-counting method [62]. Hints are often considered necessary by educators [49], but can have negative effects when students "game" them [11]. Finally, there are many ways to represent fractions: pie charts and other area models, operators and linear models [77], magnitudes [103], standard symbolic notation, and so on. Showing multiple representations to students is widely thought to be useful, but may actually be worse than a single representation in some circumstances [89].

Parameter	Settings	Interpretation
Fraction	Any $\frac{a}{b} \in (0,1), 0 \leq b \leq 9$	The target fraction the player must hit; there are 36 possibilities
Ticks	Present, Absent	For target $\frac{a}{b}$, we can display tick marks for each fraction $\frac{n}{b}$.
Animations	Present, Absent	If the player misses a target $\frac{a}{b}$, they might receive a lengthy pie chart animation showing how to divide up the number line into b parts. These are similar to direct instruction.
Initial Labels	[0,1]	For target $\frac{a}{b}$, the proportion of labels of $\frac{n}{b}$ fractions shown at the start.
Backoff Hints	1, 2, 3, 4	The number of misses for target $\frac{a}{b}$ before the progressive hinting system fills in all labels for $\frac{n}{b}$ and displays the correct answer.
Target Representation	Symbolic, Pie	How the target fraction is displayed.
Label Representation	Symbolic, Pie	How fraction labels on the number line are displayed.

Table 3.1: The parameter space for our experiment.

The parameter settings of the first number line in each pair constitutes the experimental condition. The full parameter set can be seen in Table 3.1. Our experimental set consists of Ticks, Animations, Backoff Hints, Target Representation, and Label Representation. Tick marks divide up number lines into fractional segments and can be thought of as a form of scaffolding; Animations and Backoff Hints are both hinting systems that give the player additional information upon placing a fraction in the wrong location; and Target and Label Representations control whether the target fraction and the fractions on the number line itself are represented either as pie charts or in standard symbolic notation. We suspect that Target and Label representations are likely to interact, so we treat them as a single parameter, Representation, with four settings for Target and Label: Symbolic/Symbolic, Symbolic/Pie, Pie/Symbolic, Pie/Pie. This means that our results are meant to hold for different Fraction and Initial Labels values, which are always randomly chosen.

We show the results of our system on two objective functions.

Correctness 1.0 if the player answers the second number line correctly on the first try; 0.0 if they answer incorrectly or quit/restart before reaching it. This corresponds to fraction-placement ability.

Persistence 1.0 if the player eventually answers the second number line; 0.0 if they fail to answer it or quit/restart before reaching it. This corresponds to fraction-placement persistence.

3.6.2 Correcting sampling distributions

We want the number line parameter settings to be selected uniformly at random. Thus, if one experimental condition had a better objective function value than another, it would mean that some particular settings for the first number line (marginalized over the specific Fraction and Initial Label set) increased player performance across our randomized second number line.

As mentioned above, the dataset was collected for a different purpose as the exploratory phase for a reinforcement learning experiment. Thus, the actual distribution is different than our distribution of interest. In this dataset, number lines are linked, two at a time. More specifically, the first two number lines always share the same value of Ticks, Animations, Backoff Hints, Target

Representation, Label Representation, Initial Labels, and the Fraction denominator d. These are chosen uniformly at random. Then each number line in the pair has the Fraction numerator selected uniformly at random from 1 to $d - 1$. Likewise, the second pair's parameters are selected (independently) using the same process, and so on.

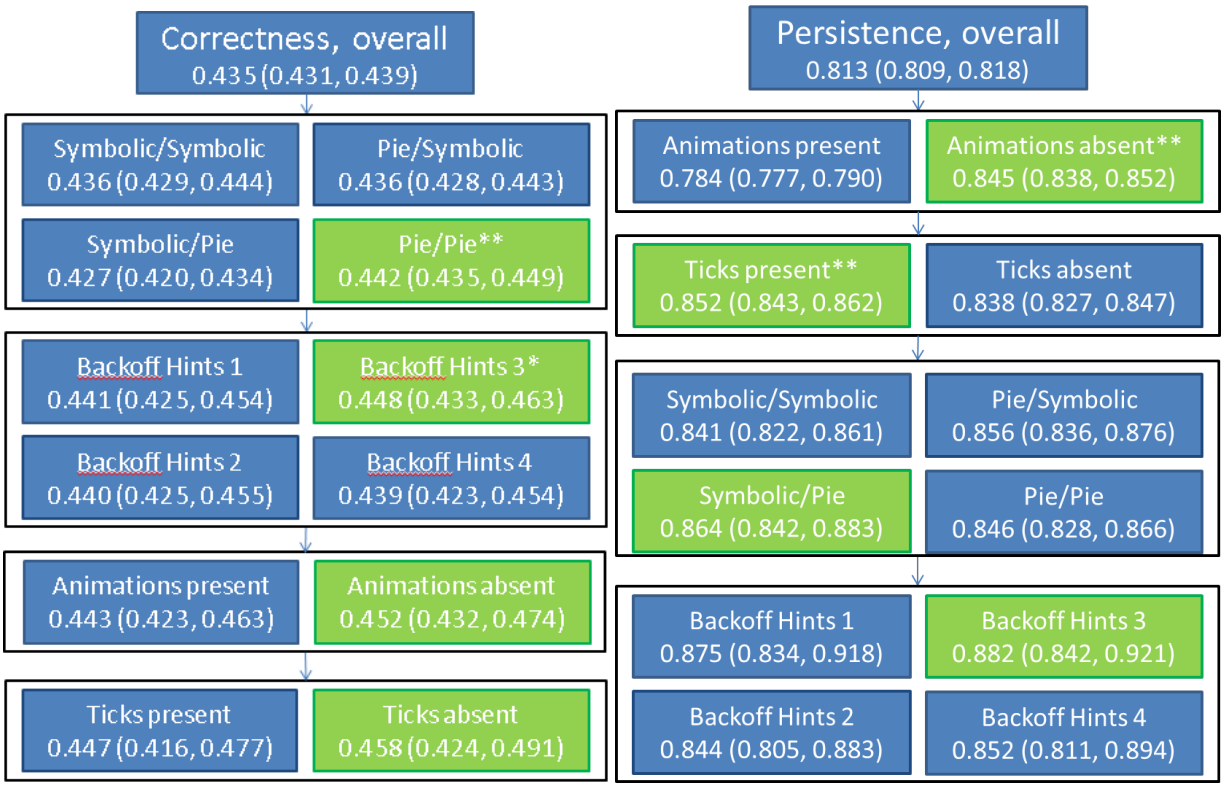
This has several implications. The first is that certain fractions are over-represented relative to the desired uniform distribution over fractions. For example, $\frac{1}{2}$ is much more likely to be selected than $\frac{1}{9}$. The second is that half of our generated pairs will match on all parameters except the Fraction numerator due to the parameter pairing, and the other half will be independently and randomly generated from the process above.

As suggested above, we can use importance sampling to address these problems by "morphing" the actual sampling distribution to the one desired by our experimentation algorithm. Since number lines in the data set come in linked pairs which share many of the same parameters, there are too many train-test number line pairs where the parameters are the same compared to ones where train-test pairs are different. Thus, importance sampling will up-weight the data samples from cross-pair links (second and third, fourth and fifth, etc.), and down-weight the oversampled intra-pair links (first and second, third and fourth, etc.). More specifically, for any experimental condition, we have a set of parameters that are already set to some known value, and a set of parameters that should be uniformly random. This gives us the desired sampling distribution over the first number line. Since we also know that our desired objective function is some measure of performance on an independent, uniformly random second number line, this specifies the full desired distribution over pairs. But since we know the original distribution used to generate the data, we simply use importance sampling as above to reweight each objective function valuation in our dataset to calculate the expected player score under our desired distribution.

3.6.3 Results

Since we have at least one player in every experimental condition, it's possible to finish the depth-first search and generate the full experimental tree. However, even just the bottom of the tree contains 64 possible parameter setting combinations, making it difficult to show the full set of

results. Instead, we show the parameters the algorithm greedily selected and its evaluation of the objective function for each of the different settings, stopping once it has set each parameter.



(a) Correctness (b) Persistence

Figure 3.4: A greedy slice of the experimental space explored, for two objective functions. Objective function evaluations are given with 95% confidence intervals, given by the centered bootstrap percentile method. Our algorithm conditions on more parameters as we go deeper down the tree, so that the results at the bottom have all experimental parameter values set according to their best observed settings. In addition, we will later statistically test the results on a separate dataset. Single asterisks mark results that will be marginally significant, $p < 0.10$. Double asterisks mark results that will be significant, $p < 0.05$.

The results are shown in Figure 3.4. Each subfigure represents a narrow, greedy slice of the experimental tree generated by our algorithm as applied to the *correctness* and *persistence* objectives defined above. They differ because the algorithm discovers that different features of number lines are more important for maximizing the two objectives: for example, choosing pie chart rep-

resentations is most important for helping players answer the next number line correctly on the first try, while turning off animations is most important for helping players persist until solving the next number line. As a reminder, at each stage our algorithm finds the single parameter setting that maximizes the objective function, while averaging over all other parameters. It then sets this parameter to the best setting and repeats this process with the remaining parameters. Thus the Representation pie/pie setting is the broadest, best parameter setting among the entire experimental set in tree generated by our algorithm applied to the *correctness* objective, as seen in Figure 3.4(a), the Backoff Hints 3 result is the broadest, best parameter setting only when the Representation is given to be pie/pie, and so on.

The confidence intervals given at each level of the tree grow wider as we go down. This is because data becomes sparser at each level, since we do not have control over the sampling distribution. In the basic online framework, the system would instead direct players to the condition in question, decreasing the amount of data near the top and increasing it near the bottom. Based on the amount of overlap present, we can guess that the results of the top and possibly second layer are reasonably trustworthy, but that we should be increasingly suspicious as we test more specific conditions.

An important question is whether the greedy method finds reasonable settings, in practice. We can of course construct examples such that for any deterministic strategy, the algorithm must perform an exponential search to find the best settings. This can be quite bad for large parameter spaces, though it is probably unlikely in practice. Since our original dataset contains samples from all over the experimental tree, we can exhaustively search all possible experimental conditions at each depth to see when the greedy search diverges. Our greedy selection of the *correctness* diverges from the global optimum on the third and fourth levels with average score 0.460 and 0.474, respectively. These values are well-within the greedy selection's confidence intervals as seen in Figure 3.4(a). Our greedy selection of in the *persistence* condition finds the globally best selection at each level of the tree. Thus we conclude that greedy selection is reliable in this particular domain, especially at the top of the tree where data are plentiful and only a few parameters are set. Even in new domains, it could be possible to run statistical tests on the greedy selection against

Repres.	Backoff Hints	Anim.	Ticks	Mean	Other mean	Statistics
pie/pie	Any	Any	Any	0.431	0.407	$\chi^2(1, N = 9675) = 4.44, p = .0035$
pie/pie	3	Any	Any	0.447	0.410	$\chi^2(1, N = 9675) = 3.13, p = .077$
pie/pie	3	No	Any	0.424	0.412	$\chi^2(1, N = 9675) = 0.18, p = .669$
pie/pie	3	No	No	0.462	0.412	$\chi^2(1, N = 9675) = 1.61, p = .204$

Table 3.2: The results for correctness on the final validation set. The first line says that interventions with double pie charts are better than all others. The second line says that interventions with double pie charts and level 3 backoff hints are better than all others, and so on.

Anim.	Ticks	Repres.	Backoff Hints	Mean	Other mean	Statistics
No	Any	Any	Any	0.890	0.864	$\chi^2(1, N = 9675) = 14.18, p < .001$
No	Yes	Any	Any	0.896	0.870	$\chi^2(1, N = 9675) = 11.84, p < .001$
No	Yes	symbolic/pie	Any	0.892	0.876	$\chi^2(1, N = 9675) = 1.33, p = .249$
No	Yes	symbolic/pie	3	0.877	0.877	$\chi^2(1, N = 9675) < 0.001, p = 0.995$

Table 3.3: The results for persistence on the final validation set.

other conditions, since the randomization means all parameter value combinations are possible. We leave this for future work.

3.6.4 Validation

Our results so far could be useful to a game designer, with appropriate validation on new players to avoid overfitting. For example, a designer might conclude that future games should rely on pie charts to represent fractions on number lines, instead of standard symbolic notation. However, we

would like even more: we want our framework to suggest important parameters and likely-effective settings for further experimentation, or even to generate research results outright. Unfortunately, our methodology makes a number of assumptions that make standard statistical tests inapplicable, and runs so many experiments that it is virtually guaranteed to find spurious results. We can, however, use our results to generate hypotheses that are testable on a new dataset.

We used a second dataset, again sampled according to the same distribution as the original. Indeed, it was collected for the same original purpose and we withheld to serve as a validation dataset. It consists of 9,675 players of Treefrog Treasure from June 20, 2013 to July 9, 2013. We did not use this dataset to help us develop our system, hence it is similar to the final test dataset common in supervised machine learning.

Recall that the first two number lines share most parameters, including all of our experimental parameters, so they can serve as our experimental condition. All other parameters are chosen independently of the experimental ones, so the conditions are comparable. The third number line is itself chosen randomly and independently from the first two, and can serve as our assessment: i.e., did players answer it correctly on the first try, or did they persist until solving it?

Thus, for any combination of parameter settings, we can ask whether players given number lines matching those criteria on the first two number lines performed better on the third than everyone else, a cross-sectional sampling scheme. We collect only one datapoint from each player, allowing us to meet the independence of samples criterion. Since both our objective functions are of the form "Pass" or "Fail", represented as 1.0 and 0.0 respectively, we use the χ^2 two sample test in each case. Other types of objective functions will in general have different appropriate tests, such as ANOVA or Mann-Whitney U.

Each experimental tree suggests a large number of potential hypotheses; here, we will simply focus on the most basic ones. Specifically, we ask whether the parameter settings chosen by our algorithm lead to increasingly "good" outcomes as we go down the tree. Each comparison will be performance of players with the settings of the experimental parameters at that point, as compared to everyone else. The results are shown in Tables 3.2 and 3.3.

Remember that the wide confidence intervals at the deep ranges of the tree with many param-

eters set led us to not trust those results. Indeed, our statistical tests confirm this fact: we achieve significantly or marginally significantly better performance following our automatically-generated results at the top two layers, but mostly do not see significant effects on the bottom two. Thus our validation results are not surprising, and underscore the need for a validation set when running the system offline to avoid overfitting. In the online case, overfitting is less of a concern since new data is drawn for each experiment rather than trying to simulate experiments over existing data.

Finally, the effects may seem relatively weak, with a 2.4% increase from using the pie/pie representation on *correctness*. However, this is because we are measuring differences of interventions consisting of two numberlines and no explicit instruction. If a 10% difference in test scores after thirty minutes of instruction is good, then a 2% improvement after one minute may be reasonable. The extension of effective short interventions to effective long interventions is not trivial, and is left to future work.

3.7 Discussion

3.7.1 Hypothesis Generation

Our primary purpose is to introduce an automatic experimentation framework. To demonstrate its utility, we have shown that we can use our implementation to discover interesting information and find potential educational hypotheses to further explore. We certainly do not claim that our findings are highly general, mature educational results. There are many caveats: the intervention is extremely short, the measured task is near-transfer onto a broadly randomized number line, the population is drawn from an online educational game, and so on.

That being said, our results suggest broader hypotheses that could now be tested either in our framework, with lengthier interventions and more comprehensive assessments, or in a standard fashion in a school or lab setting. While the expert (a member of our team) specified the parameter space, he did not need to decide particular parameter settings that were likely to perform better than others. This reduces our reliance on expert knowledge and makes it less likely that we will miss important results due to lack of extensive exploration.

As one example from our *correctness* results, we see that the pie/pie representation is significantly better than any other representation combination at improving player performance on a huge variety of number lines with both symbolic and pie chart representations for targets and labels. Educational experts that we spoke with found this to be quite interesting, since number lines almost always appear with symbolic notation. Not only is this a statistically significant result on a rare representation combination that bears further research on its own, it also has potential implications for multiple representations research in general.

To explain further, the early math educational literature generally supports the notion of multiple representations in supporting learning [64], but only in certain circumstances. Many students have difficulty converting back and forth between different representations [104]. One of the reasons multiple representations may sometimes not be beneficial are that students simply opt to ignore presented number lines or informative diagrams when they are given with no added explanation [41]. In the fraction domain specifically, other researchers have found that multiple graphical representations may actually be harmful relative to single representations [2], unless accompanied by a self-explanation prompt.

Although our intervention number lines do offer hints, our number lines have neither explanations nor prompts in the traditional sense. Yet using pie charts together with number lines leads to superior performance on the test line, compared to using number lines using the standard symbolic notation. Thus our system may have found an example where multiple representations are useful, without the additional explanation or support suggested by the literature.

We do not know why this is the case in our game, but one explanation might be that understanding symbolic notation may be more difficult than understanding pie charts, which at least are seen outside of the classroom. Then players who are not proficient with number lines may learn them faster or be more willing to play only when they can map them to a more familiar pie chart representation. The opposite possibility is that players have overfit in the classroom to number lines with symbolic notation. In this case, they would have difficulty answering the test number line questions that involve pie charts, and so the most profitable thing to practice would be the number line and pie chart combination. Though, we also note that pie/symbolic and symbolic/pie condi-

tions are worse as well; perhaps the difficulty of mapping between three representations outweighs the potential benefits of seeing a pie chart target with a standard, symbolically-labeled number line.

Regardless of the explanation, our system was able to automatically find and run an interesting experiment that we would not have thought to try. The generated results were confirmed on a separate dataset, and differ in key ways with well-accepted literature, suggesting extensions to existing theories and further research to be done. This demonstrates the exploratory power of our method.

Finally, in this chapter we have concentrated on parameters in an educational game; however, our method should be applicable to other domains, as well. For example, in the e-commerce domain, one could consider the parameter space of page layouts, checkout strategies, and item recommendation algorithm, with an objective function of clickthrough rate. Or a polling experiment on Mechanical Turk might ask which combinations of introduction, phrasing, question ordering result in the most consistent survey results. The same approach with negated objective values could even be used to find particularly "bad" conditions, if that was of interest. In all cases, the key is to have a constant stream of users, the ability to choose parameter settings for users and measure an outcome, and some sense of which outcomes are "good."

3.7.2 Limitations

Our work has important limitations. We wish to stress that the results are only strictly applicable to the user population they were generated from: in our case, players of our educational game. This can be mitigated in certain domains where demographics can be collected. When this is not possible, it may be best to treat the obtained results as hypotheses to test for future experiments on the desired population.

Furthermore, the algorithm is only as effective as the parameter space specified by the experimenter. It is entirely possible that the given parameters have negligible impacts on the objective function. In this case, the algorithm will greedily select parameter settings that appear very close to the global average, which may serve as a signal that a new parameter space should be devised. Researchers with a solid grasp of the underlying behavioral theories may be able to create more

effective parameter spaces.

We also caution that problems arise in certain platforms, especially when users are not invested in the system. In the games domain, users can quit at any time; if a long intervention is desired, changes in the objective function may be caused by survivor bias induced by particular users leaving. As an example, an extremely difficult number line might appear to have a strong test score, but only because it caused all the players bad at answering number lines to quit. We control for this effect by having extremely short interventions so that the probability of quitting is low, and giving players who quit the lowest possible score. This protects us from spurious results caused by biased patterns of quitting, but also (intentionally) entangles learning and engagement. This issue is much less prominent in Mechanical Turk or software being used in schools by teachers, where the populations are more invested in finishing the intervention.

Also, this approach is focused only on exploring hypotheses related to the overall effects of system-controllable behaviors. Many factors such as age, gender, personality, performance, etc. are not directly controlled by the system, but are frequently studied in educational literature. For example, this system cannot identify different groups of people which need different interventions based on past performance. This type of useful adaptivity is challenging to achieve with limited data, and is left for future work.

3.8 Future Work

This is a new domain, only made possible in the past few years through the increasing use of the Internet. As such, there is a tremendous number of possible ways our framework could be extended or improved. We will list only a few of them.

Our implementation currently only handles standard, categorical factor experiments. This is not a fundamental limitation, but there is more work to be done to handle ordinal or numeric factors. Our system currently cannot deal with these variables because it does not know how to find the ideal parameter setting to use in a continuous range. One solution is to sample at random from numerical factors, then chop them into ranges that best separate the data as in regression trees. Another approach is to try using one of many well-known optimization techniques which attempt

to find the best value, such as Covariance Matrix Adaptation.

As mentioned earlier, our search strategy in the space of experimental parameters is a staged, greedy selection designed for both maximizing the objective function value and finding which parameters are most important. This is appropriate in an educational domain. However, there are many other possible search strategies maximizing other goals. For example, a psychologist might care about variables causing the biggest difference in behavior, in which case a better strategy might be to choose parameters with greatest information gain, as often done in decision trees [87]. In this chapter, our results are taken from the offline case, where the search strategy is less important, but when players are committed to conditions online the search strategy is critical.

Future work includes investigating search strategies in the online case. If the researcher's interest is purely in mapping out the hypothesis space, one could imagine a search strategy that simply tries to find the most discriminative parameter at each level of the tree, using some well-known metric like information gain or Gini impurity which are used to learn decision trees [21]. And departing from standard techniques, we could imagine a system that does a soft search over the parameter space to find the discriminative parts of the experimental tree. There are many online algorithms from the active learning and multi-arm bandit communities that attempt to do similar things, which can potentially be adapted to this framework.

3.9 Conclusion

Recent years have seen the emergence of large sources of user data. In this chapter, we take advantage of these new data sources and propose a general, automated experimentation and hypothesis generation framework. This framework is specifically designed to automatically explore large hypothesis spaces in human behavioral research. Our importance sampling component allows the system to be used offline and when we have different distributions than the one of interest.

To show the usefulness of our framework, we implement it using an educational game. Using already-collected data, our system explores the hypothesis space for two alternative objective functions: maximizing player correctness and player persistence on a highly randomized test numberline. We find the most important parameters and their recommended settings and show that

the greedy selection does a good job of finding the best settings at each level. We then confirm just a few of the most promising generated hypotheses on a already-collected, different data set. One of these hypotheses, generated from an unusual method of representing fractions on a number line, seems to be in opposition to recent work, which indicates that our system is indeed capable of automatically generating and testing interesting hypotheses that may not have been otherwise discovered.

Of course, attempting to maximize some objective by studying parameters one at a time is only one potential scientific goal. The next chapter will focus on a different scientific goal: discovering the generality of a scientific result.

Chapter 4

GENERALIZING SCIENTIFIC RESULTS

4.1 Introduction

The automatic hierarchical experimentation framework introduced in Chapter 3 is primarily designed to improve software according to some objective function, such as learning or revenue. Yet this kind of optimization is not the only possible goal. Instead of striving to improve user outcomes, we could also use software as a laboratory for performing high volume, precision experiments. In this case, the primary goal switches to the advancement of behavioral science.

Recent years have seen a rising interest in using machine learning and statistical methods to automate parts of the scientific process [108]. This would free humans to specify high-level research goals, while machines and algorithms handle low-level tasks. For example, sequencing the first human genome took many researchers several years [33]; automation and new hardware means the process now takes on the order of days and at a fraction of the cost and labor [1], freeing researchers to examine the links between genes and health outcomes instead of running sequencing tasks. Besides simply speeding up data collection, such systems could even reduce the amount of data required to uncover new knowledge by choosing experimental samples in an intelligent fashion. And they are of special interest in domains where researchers can not frequently intervene, as with Mars rovers [18].

For this kind of automation, we must first formalize different scientific goals, then develop efficient algorithms to achieve them. In this chapter, I focus on the goal of "generalization." More specifically, many scientific domains have hierarchical methods of representing experimental subjects or conditions in a tree form, such as phylogenetic trees (evolutionary biology) or WordNet (computational linguistics) [76]. If we know that some effect occurs or holds for a node in the tree, we wish to discover the "generality" of this effect by finding the largest subtree for which the

same effect occurs. For example, say that priming the word "dollar" causes subjects to be more likely to make rational bets. Is the same true for all currencies, or all items with monetary value? Given a tree like WordNet classifying words into increasingly-general categories, and the strength of the observed effect, my algorithms will automatically run experiments to find the most general category of word that causes humans to behave more rationally.

In this chapter, I introduce a trio of algorithms to automatically generalize scientific results. The idea is that a scientist specifies a family of experiments and "generalization" rules to follow in case an effect is likely to hold of an experiment. These experiments and generalization rules are handed to my algorithms, which choose samples to allocate amongst the different experiments in order to quickly identify general effects. In particular:

- I introduce the algorithm MARGIN for determining if an effect holds on a set of conditions (i.e. if proportion Q of the conditions have expected value greater than a researcher-specified cutoff α with high confidence). Simulations show that MARGIN is more efficient than several baseline algorithms because it directs more samples to difficult-to-classify *necessary* conditions but fewer samples to difficult-to-classify *unnecessary* ones.

- MARGIN can choose when to advance along a chain of increasingly general condition sets, but what if there are several potential threads of research? Using the same core optimistic strategy, the algorithm MARGINCHAIN efficiently chooses between several competing chains of experiment sets and outperforms baseline strategies that focus on running one experiment at a time.

- One common form of generalization rules is a tree classification scheme, where experimental conditions can be thought of as leaves in a tree that classifies them at different levels of generality. I enhance MARGINCHAIN to work in cases where chains can join, as in a tree structure, and to choose between multiple potential cutoffs if the correct one is not known a-priori. Simulations show this algorithm, MARGINTREE, performs better than a round-robin baseline at identifying a special subtree for which effects hold. This remains true even when

the starting node or depth are unknown, and several cutoffs (only one of which is correct) are under consideration.

- As a case study, I run MARGINTREE on a simulation with parameters drawn from a real-world experiment about number line sequences for teaching fractions in an educational game. MARGINTREE is significantly faster than the standard round-robin approach at detecting a target "interesting" subtree.

4.2 Setup and Definitions

An experimental *condition* C is something we can actually sample. In this work we will consider Bernoulli conditions, so with probability $p_i = \mu_i = \mathbb{E}[C_i]$ we receive a payoff of 1 and otherwise receive a payoff of 0. This corresponds to experiment conditions with binary responses, i.e. did a user click an advertisement or not, or did a student answer a test question correctly or not.

For illustrative purposes, say we wish to study the effects of showing certain words to subjects and studying whether they choose to make a rational bet later on. Then a condition is a word such as "dollar", and sampling the condition means showing this word to a subject and measuring whether they chose to take the bet (1) or not (0). μ then corresponds to the true probability that upon being shown a word, a subject would take a rational bet later on.

An experiment *set* S is a group of conditions $C_{1...N}$, such as a set of words: e.g. "dollar", "quarter", and "dime." We say that an effect holds of a set if and only if proportion Q of the conditions has a mean greater than a specific cutoff value α. As with all experimental science, we must gather evidence to make this determination: that is, we will want to show either that the effect is likely to hold ($P(\mu_i > \alpha) \geq 1 - \delta$ for proportion Q of the conditions), or that it is unlikely to hold ($P(\mu_i < \alpha) \geq 1 - \delta$ for proportion $1 - Q$ of the conditions). Typical values for $1 - \delta$ are 0.99, 0.95, or 0.9, and will depend on the domain and researcher. For distributions over probable condition means, we use Beta posteriors to estimate these probabilities, with Beta priors always set to $(1, 1)$, and say that a condition is *classified* if $P(\mu_i > \alpha) \geq 1 - \delta$ or $P(\mu_i < \alpha) \geq 1 - \delta$.

We note that this is only one of many possible definitions of what it means for an effect to

hold of a set of conditions. Other definitions, such as having a sufficiently high expected value of randomly sampling from any C_i in the set, would require different sampling strategies. Which definition to use depends on what the researcher wants to find; ideally, researchers will have access to many such definitions and corresponding sampling algorithms. Our particular definition has the advantage of being robust to outliers, controlled by Q (since as Q shrinks more and more conditions can fail to meet the cutoff).

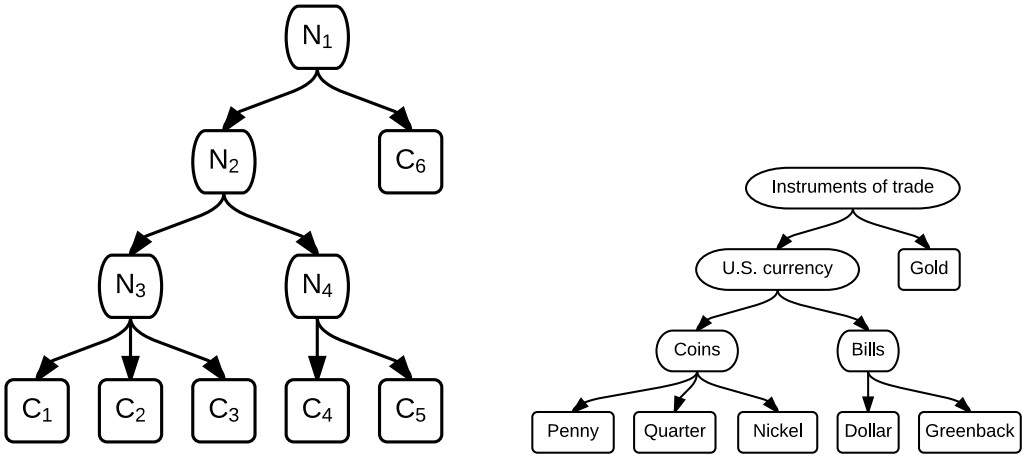
An experiment *chain* is an ordered list of experiment sets $S_{1...M}$. These chains correspond to a plan about how to generalize: if we believe that the effect holds for the first experiment set, we can add all the conditions associated in the second set and ask if the effect holds for this combined set, and so on until we reach the end of the chain or reach a point when the effect fails to hold for our current combined set. For example, S_1 might consist of words pertaining to U.S. currency, S_2 might consist of words pertaining to all currencies, and S_3 might consist of words pertaining to all financial instruments. Perhaps subjects are more likely to make rational bets if shown words such as "dollar" and "quarter" but not "franc" or "pound;" then the effect likely holds for S_1 but not S_2 and would be an interesting research result.

Finally, an experiment *tree* is a tree whose leaves are conditions. A *node* N in the tree thus defines a set, whose member conditions are the leaves of the subtree rooted at that node. A path from any node to an ancestor node defines a chain, whose sets consist of union of their children's sets when we step up from a node to its parent (see Figure 4.1).

4.3 Margin

Our first goal is to develop a strategy for determining if an effect holds for an experiment set. For the sake of the following discussion, say we have conditions C_1, C_2, C_3 with means $\mu_1 = 0, \mu_2 = 0.51, \mu_3 = 1$. Assume a cutoff $\alpha = 0.5$. For example, perhaps we have three coins that land heads 0%, 51%, and 100% of the time, and we wish to identify whether proportion Q of the coins more likely to land heads than tails.

Without any information about how the conditions are related, the standard approach is to sample in a round-robin fashion. We continue sampling until $P(\mu_i \geq \alpha) \geq 1 - \delta$ or $P(\mu_i <$



(a) Abstract

(b) Currency

Figure 4.1: Experimental conditions C can be thought of as leaf nodes in a tree which classifies them into groups of different generality. Each node N defines a set S of conditions within the subtree rooted at N, ex. N_3's set is C_1, C_2, C_3. A path from any node to an ancestor defines a chain of experiment sets, such as $N_3 \Rightarrow N_2 \Rightarrow N_1$, which specify increasingly general sets of conditions to check. An example of a currency word tree from the rational betting example with the same structure as the abstract tree is shown.

$\alpha) \geq 1 - \delta$ for all conditions (that is, no conditions are unclassified). Here, classifying a condition corresponds to deciding that with high probability its mean is less than or greater than the cutoff value. This will require many samples primarily because C_2 is difficult to classify (μ_2 is close to α). The key insight is that we need only identify proportion Q of the conditions as meeting the cutoff, or $1 - Q$ proportion of conditions as failing to meet the cutoff. Importantly, we do not need to classify all conditions, nor do we need to send the same number of samples to all conditions.

There are two primary ways in which we can save samples. The first is to avoid sampling conditions when they are not needed to determine if an effect holds. For example, say $Q = \frac{1}{3}$. To say the effect holds for the set means, by definition, that we amass evidence to prove least one of the coins has greater than a 50% chance of coming up heads. To do this, we need not sample C_1 and can sample either C_2 or C_3. The less-difficult condition, C_3, is a better choice: fewer samples will be required to show $P(\mu_3 \geq \alpha) \geq 1 - \delta$. Or in other words, it takes many fewer flips to show that a coin always landing heads is biased towards heads, as compared to the number of flips needed to show a coin landing heads 51% of the time is biased towards heads. Since we only need to show one coin is biased towards heads, we might as well pick the easier one.

Paradoxically, the second way to save samples is to direct *more* samples to difficult conditions, when they are required to decide if the effect holds. For example, say $Q = \frac{2}{3}$. We must show that C_2 and C_3 both meet the cutoff to show the effect holds. In this case, we should sample C_3 only a few times, focusing our attention on C_2. That is, since it's easy to show that the always-heads coin is biased towards heads, and we need to prove that both it and the mostly-heads coin are biased towards heads, we should focus our experiments on the mostly-heads coin.

We propose a new Bayesian sampling strategy following these principles, MARGIN, which is described in Algorithm 2. This algorithm uses the principle of "optimism under uncertainty" by directing samples to conditions it believes could require the fewest number of samples to classify. MARGIN keeps an estimate of how many heads responses $hOpt_i$ from C_i would be required to believe that $P(\mu_i \geq \alpha) \geq 1 - \delta$, and likewise tails responses $tOpt_i$ for $P(\mu_i < \alpha) \geq 1 - \delta$. It then chooses to sample conditions which require the fewest number of heads or tails responses to classify in a new way. For example, if we are unsure if C_i meets the cutoff, optimistically

$\min(hOpt_i, tOpt_i)$ flips are required for a classification. Alternatively, if we believe $P(\mu_i \geq \alpha) \geq 1 - \delta$, more heads flips will not change our minds, so $tOpt_i$ is the minimum number of samples to make a new classification. Finally, while the strategy can continue sampling forever, a convenient halting criteria we will often use is to stop once the algorithm believes the effect holds or fails to hold for the set.

MARGIN's construction means that it has two useful properties. Among conditions it must sample, it will direct more towards the difficult conditions; once a condition is classified we are unlikely to direct more samples to it because many samples will be needed to reverse our decision. At the same time, MARGIN avoids sampling unnecessary difficult conditions: as more samples pass without a classification, the associated $hOpt$ and $tOpt$ values increase and make that condition unattractive to sample further unless we cannot classify enough of the other conditions.

4.3.1 Simulation results

We test MARGIN against several other possible strategies. We measure how long it takes each algorithm to gather enough evidence to decide that the effect holds of the experiment set, and how it distributes samples between the conditions. We set $\alpha = 0.5$, with $1 - \delta = 0.99$ to avoid cost-accuracy tradeoffs between the different algorithms (none of them made mistakes on any run). All ties are broken at random.

The other sampling strategies are as follows:

RoundRobin Sample the minimum-sampled condition.

RoundRobinTerminate Sample the minimum-sampled *unclassified* condition.

ProbabilityMass Similar to MARGIN, but instead of using the minimum number of samples, use the maximum probability mass in favor of a change in classification. That is, set $tOpt = \text{Beta}(h, t).\text{CUMULATIVEPROBABILITY}(\alpha)$ and $hOpt = 1 - tOpt$, then sample in proportion to the mass returned instead of picking the minimum.

Algorithm 2 Margin

function MARGINSET($H_{1,\ldots,N}, T_{1,\ldots,N}, \alpha, \delta$)

▷ H_i and T_i are counts of heads and tails for C_i

▷ Returns the condition to sample next

return $\arg\min_i$ MINTOCLASSIFY$(H_i, T_i, \alpha, \delta)$

end function

function MINTOCLASSIFY(h, t, α, δ)

$hOpt = \min\{n \mid \text{CLASSIFY}(h + n, t, \alpha, \delta) = \text{“Meets”}\}$

$tOpt = \min\{n \mid \text{CLASSIFY}(h, t + n, \alpha, \delta) = \text{“Fails”}\}$

$s = \text{CLASSIFY}(h, t, \alpha, \delta)$

if $s =$ “Meets” **then return** $tOpt$

else if $s =$ “Fails” **then return** $hOpt$

else return $\min(tOpt, hOpt)$

end function

function CLASSIFY(h, t, α, δ)

p = Beta(h, t).CUMULATIVEPROBABILITY(α)

if $p \leq \delta$ **then return** “Meets”

else if $p \geq 1 - \delta$ **then return** “Fails”

else return “Unknown”

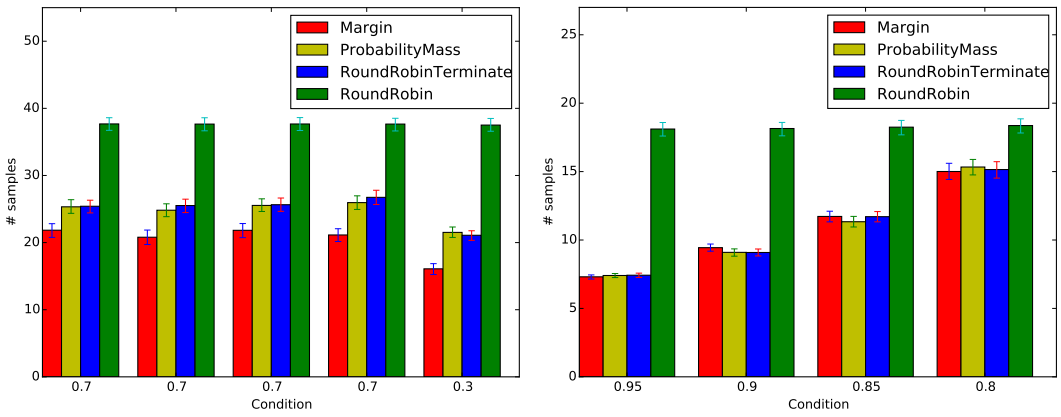
end function

As seen in Figure 4.2, ROUNDROBINTERMINATE and PROBABILITYMASS are able to direct more samples to difficult *necessary* conditions. This is because they are much less likely to sample already-classified conditions. However, their performance degrades in the presence of difficult *unnecessary* conditions (Figure 4.2(c)). The other strategies do not realize that some conditions are likely to require many samples to classify, and so continue to sample them until the easier options are exhausted. For example, PROBABILITYMASS cannot distinguish between a well-sampled and a sparsely-sampled condition if both have $p(\mu_i \geq 0.5) = .70$, while MARGIN would prefer the sparsely-sampled one.

4.4 MarginChain

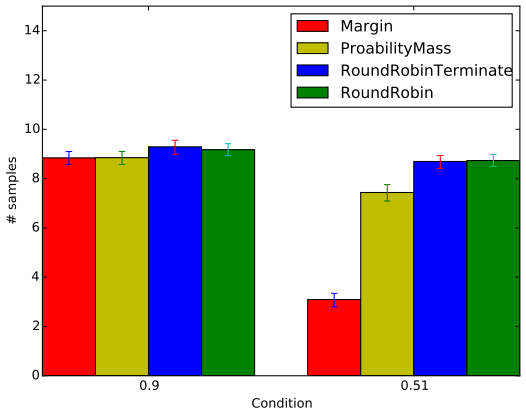
MARGIN allows us to make generalization decisions when given a single experiment chain, which is a list of increasingly large experiment sets to try if we decide the desired effect holds on previous sets. However, it is not clear what to do if we have multiple experiments running at once. Given limited resources, how do we choose which one to sample? A human scientist might choose a single chain and follow it until failing to find the effect, but such a strategy could easily spend many samples struggling on a difficult set in one chain while other experiments would have concluded much faster.

We can use the same idea from MARGIN to design an algorithm to decide between competing experiment chains, called MARGINCHAIN. Remember that a chain consists of a series of sets to add to consideration if we believe the effect holds for all preceding sets. Thus at any moment a chain has an "active" experiment set. MARGINCHAIN chooses the chain whose active set would optimistically require the fewest number of total samples to determine if the effect holds. For each chain's active set, we calculate the minimum number of heads flips $HOpt$ required to confirm that proportion Q of the conditions have $P(\mu_i \geq \alpha) \geq 1 - \delta$, and likewise $TOpt$ for confirming proportion $1 - Q$ of the conditions have $P(\mu_i < \alpha) \geq 1 - \delta$. As with MARGIN, the number of samples needed to make a new classification about an unclassified set is $min(HOpt, TOpt)$, while if we believe the effect does not hold for a set we use $HOpt$ and vice versa. Once a chain is chosen, we use MARGIN to pick a condition to sample.



(a) Standard

(b) Varied



(c) Difficult

Figure 4.2: Distribution of samples by different algorithms, before deciding that the experimental criteria is met. The x axis labels indicate the expected value of sampling each condition in the set; the y axis is the number of times that each algorithm had to sample that condition before deciding the effect held for the entire set. Values of Q used were $0.59, 0.99, 0.49$ for Standard, Varied, and Difficult. We see that MARGIN is superior to other strategies because it explicitly avoids sampling unnecessary difficult conditions (Figure 4.2(c)).

4.4.1 Simulation results

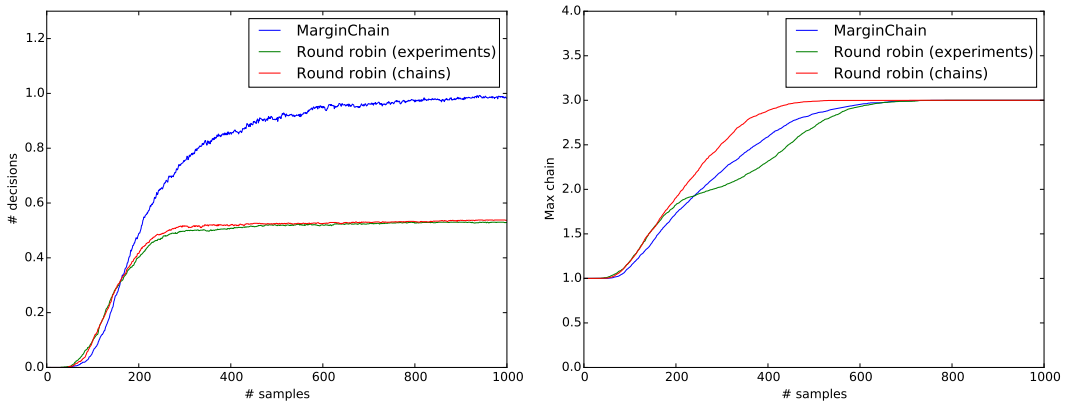
We compare MARGINCHAIN against two other baselines for selecting which experiment chain to work on. All algorithms use the MARGIN sampling strategy once a chain is selected.

RoundRobinChain Samples one chain at a time until termination by reaching the end or deciding the effect fails to hold for the current set, randomly selecting if all chains have terminated.

RoundRobinSet Samples a chain's current set until a classification decision for the set is reached, as for MARGIN, then moves on to the next active chain instead of continuing on the same chain.

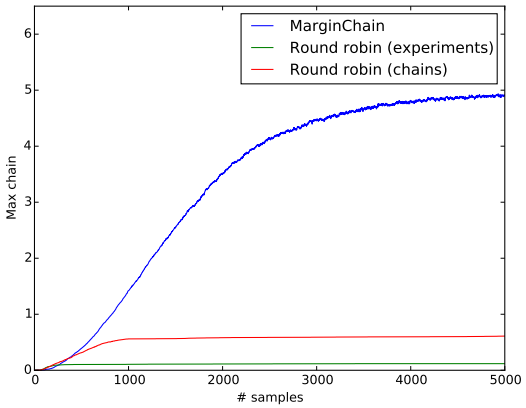
We evaluate the performance of these algorithms in three simulations, shown in Figure 4.3. Remebering that chains are just a series of sets, we will use two different sets to construct chains for the purposes of evaluation. These sets are named *Easy*, with condition means $(0.7, 0.7, 0.7, 0.7, 0.3)$; and *Hard*, with condition means $(0.51, 0.51, 0.51, 0.51, 0.49)$. In all cases, $1 - \delta = 0.99$, $\alpha = 0.5$, and $Q = 0.79$, so the correct decision is always to generalize and advance along the chain. These sets are so named because the "easy" set is full of conditions which do not need to be sampled very much to tell if their expected values are less than or greater than α, while the "hard" set is full of conditions whose expected values are very difficult to distinguish from α.

The results are shown in Figure 4.3. The *Varied* simulation has two chains of length one, one of which is *Easy* and the other of which is *Hard*. MARGINCHAIN is able to reliably find and quickly classify the *Easy* chain, while the other strategies do not know to stop working on the *Hard* one. This difference can be made arbitrarily large by "hiding" a long (length five) *Easy* chain amongst many (eight) length one *Hard* chains. Even in the situation where RoundRobinChain is optimal, with all chains being length three *Easy* experiments, MARGINCHAIN still does very well. It takes additional samples to recognize that one chain's first experiment generalizes, but once that determination is made its optimistic nature prefers to continue sampling that chain and so it quickly makes additional generalization decisions.



(a) Varied

(b) Same



(c) Single easy, many hard

Figure 4.3: Performance of MARGINCHAIN against different round-robin strategies. MARGINCHAIN is superior at finding and making easy decisions when possible. The difference is most extreme with one easily-classified chain and many difficult ones, where MARGINCHAIN is able to identify and grow the easy chain quickly (Figure 4.3(c)). And even when all chains are identical, its optimistic nature causes it to find and focus on a single chain nearly as quickly as the optimal strategy (Figure 4.3(b)). All graphs generated from 1000 runs.

4.5 MarginTree

Given one or more pre-existing chains of increasingly general experiments, MARGIN can automatically continue to generalize until it believes the effect no longer holds. And in the case where we do not know which of several chains is best to explore, MARGINCHAIN can decide which chain to investigate. Thus, if a reasonable α to search for is known for each chain, we can start many possible chains and allow MARGINCHAIN to advance the promising ones.

But what if we do not even know a-priori how to set α? One approach, similar to depth-limited search, is to begin with a high α. Once all chains terminate (because the effect does not hold in their active experiment sets), we can then reduce α. This runs into the problem that if one of the cutoffs happens to make many conditions difficult to classify, MARGINCHAIN will get bogged down and spend many samples, whereas a lower cutoff might have led to the discovery of many interesting subtrees; in our simulations, difficult α values can cause MARGINCHAIN to do no better than round-robin (Figure 4.4). Instead, using the same idea behind MARGINCHAIN, we can simultaneously consider many cutoffs $\alpha_{1,\ldots,N}$. As long as one of them leads to easy classifications, we will tend to direct samples there.

MARGINTREE is thus MARGINCHAIN with two modifications. The first is that since chains consist of a path of nodes to the tree root, many of the chains eventually overlap as they progress up the tree. Thus, each time we sample, we check if any chain's active set is completely subsumed by another's; if so, we remove it from consideration. The second is that we allow the researcher to specify several cutoffs $\alpha_{1,\ldots,N}$. Then, we pick the chain that has the lowest optimistic number of samples required to change classifications for *any* of these cutoffs, as opposed to a single pre-specified one. We offer one word of caution: choosing a very low cutoff causes MARGINTREE to generalize these low cutoffs all the way to the top, which is unhelpful in distinguishing nodes. It is better to err on the side of $\alpha_{1,\ldots,N}$ being higher rather than lower.

4.5.1 Simulation results

We first constructed a binary tree of depth six. We randomly selected a depth three node N^* which we wanted our algorithms to identify; the effect we wished to find was defined by $\alpha = 0.5$, $1 - \delta = 0.95$, and $Q = 0.8$. N^* was the root of a subtree with a set of experimental conditions: we set 80% of these leaves to have means selected uniformly from $[0.6, 0.7]$ and the remainder to have means of $[0.3, 0.4]$. For all other subtrees at the same depth, we set 20% of their leaves to have means in $[0.6, 0.7]$ and the remainder in $[0.3, 0.4]$. We then found a randomly-chosen chain from N^* to some leaf, such that the desired effect held for the set of leaves of each node in the chain. We measured MARGINTREE's performance by looking at the highest node in this special chain for which it was able to collect enough data to "prove" that the effect held for the leaf conditions of that node, testing the following versions:

MarginTree, depth+cutoff Begins the chains at nodes of depth three with $\alpha = 0.5$.

MarginTree, bottom+cutoff Begins the chains at leaf nodes with $\alpha = 0.5$.

MarginTree, bottom+backoff Begins the chains at leaf nodes, choosing the next α from the list $(0.8, 0.7, 0.6, 0.5)$ whenever all chains terminate.

MarginTree, bottom+simul Begins the chains at leaf nodes, but simultaneously considers all values of α from the list $(0.8, 0.7, 0.6, 0.5)$ as described above.

RoundRobinLeaf Round-robin sampling.

As shown by the results in Figure 4.4, extra information helped us improve our performance. If the experimenter knows what level of generality she is interested in, starting at higher subtrees is useful. Likewise if the experimenter knows what cutoffs are interesting, potentially due to running a few experiments in the past, this helps save samples as well. And if the cutoffs are unknown, simultaneously choosing between multiple cutoffs is much better than running them one at a time until termination, a strategy which gets bogged down by the difficult 0.7 and 0.6 cutoffs. These

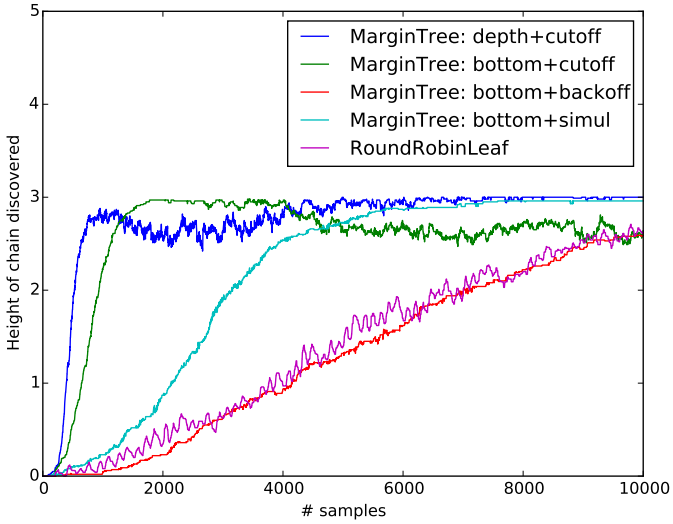


Figure 4.4: Ability of MARGINTREE variants to sample and identify the subtree of N^*, averaged over 100 runs. Extra information, such as starting chains at the correct tree depth or knowing a good cutoff to search for, was useful. But even simultaneously running multiple cutoffs, some of which were difficult, was better than round-robin sampling.

cutoffs are difficult because many leaf conditions had means in the $[0.6, 0.7]$ range and would have required many samples to determine if they were under or over the cutoff. This improvement reflects the same core idea behind MARGIN and MARGINCHAIN: we can make faster progress by identifying difficult decisions and delaying them in favor of easier ones.

4.6 Number line sequences

As a concrete example of how a scientist might use our algorithms, consider once again the educational game Treefrog Treasure, described in Section 3.2. We will be using a filtered, randomized dataset consisting of 26,014 players.

One natural question a researcher might ask is how one should order number lines such that players are most likely to subsequently answer a randomly-generated test number line correctly. For example, number lines could represent fractions visually as pie charts, or in standard symbolic notation. Furthermore, each number line had two places where fractions needed to be represented:

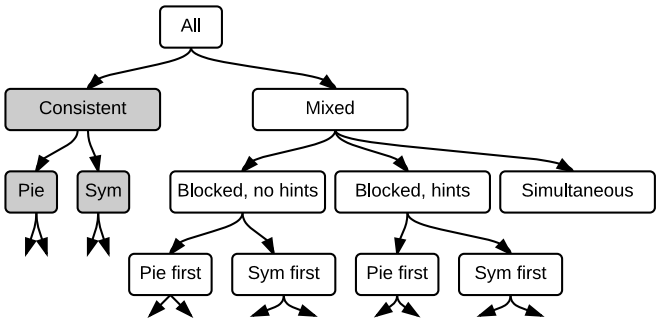


Figure 4.5: A partial tree classifying number line sequences. Not shown are additional branches representing different denominator choices for the first pair of number lines (2-9). Shaded nodes correspond to a special subtree with especially good player performance.

a target fraction the player had to aim for, and label fractions on the line itself to direct them. In addition, some number lines had animated hints which would show both representations simultaneously. We will say that a sequence was successful if the player was able to reach and solve a randomly-generated test number line after this sequence, and otherwise is a failure.

For simplicity, we will restrict ourselves to practice sequences of length four. In this scenario, we could simply consider all possible sequences of possible target and label representations independently, but it is likely there is more structure to the problem than that. For instance, sequences involving both types of representations most likely have different pedagogical properties than those where are all fractions are represented the same way. And of mixed-representation sequences, consider *blocked* sequences where both target and label were one type of representation for the first pair of lines, then were the other for the second pair. These might have distinct effects compared to sequences in which pie charts and symbolic representations showed up in each number line, either as part of the animated hints or in the target and label representations [90]. A partial tree structure representing these classifications is represented in Figure 4.5.

Since we have already collected data on this problem, we can estimate from our data the probability that a student seeing one of these sequences was able to successfully solve the test number line. The μ_i are difficult to distinguish: in our dataset, the vast majority lie in $[0.8, 0.9]$. It turns

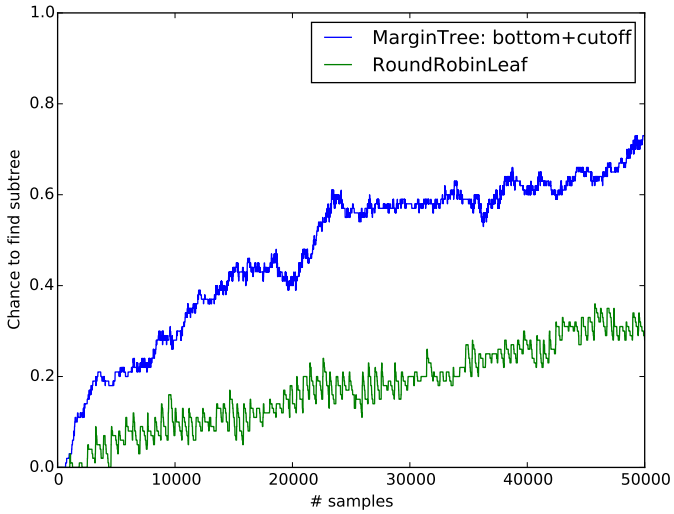


Figure 4.6: MARGINTREE is faster at gathering evidence confirming the presence of a high-performance subtree representing sequences with consistent, all-symbolic fraction representations and no animated hints. $1 - \delta = 0.9$, $P = 0.5$, $\alpha = 0.89$; results averaged over 100 runs.

out that the single-representation symbolic sequences are quite good: 75% of the sequences of this type have mean performances of over 0.89, with the runner-up node (single-representation pie chart) having 37.5% of their associated conditions having means greater than 0.89. And in simulations with the μ_i estimated from real data, we see that MARGINTREE is significantly faster than typical round-robin experimental sampling to detect this special node, as seen in Figure 4.6. We note that round-robin will eventually detect this node, but simply takes much longer.

4.7 Future work

Q and $1 - \delta$ are domain- and researcher-specific parameters that describe what kinds of results are interesting or acceptable. However, the starting node or depth, and α are better-characterized as prior knowledge or tuning parameters, and are conceivably ones we could learn from data. In this chapter, we show MARGINTREE can avoid requiring a starting node or depth by starting chains at all nodes of any depth and merging them as generalization decisions are made. However, this strategy will not work if there are so many leaves that it is infeasible to sample them all, such

as the space of all genetic sequences. In these cases, instead of offering a hard guarantee that Q conditions are likely to meet the cutoff, we could instead offer a probabilistic guarantee, reducing the number of samples required. Alternatively, we might need to truncate the tree, measuring the performance of a truncated node with a random rollout as in monte carlo tree search strategies.

More serious is the reliance on knowing a reasonable set of cutoffs. The performance of the MARGIN strategies can be sensitive to difficult choices of cutoffs; offering the ability to run many simultaneous cutoffs helps, but does not eliminate this issue. We doubt, for example, that a researcher would have guessed to try 0.89 as a reasonable value for the Treefrog test with no prior knowledge. There may be a strategy that allows us to adaptively choose better cutoffs as we sample, perhaps by looking for midpoints in large gaps between condition means (since cutoffs are easiest to verify when condition means are far away). Another possibility is for the researcher to directly decide how much a sample "costs" relative to generalization decisions, so the algorithm can decide that a particular cutoff is unfruitful relative to this metric and attempt a different one.

An even more difficult problem involves the tree structure inherent to the algorithm. Not all domains have such a structure; while they can be learned via agglomerative clustering or by relative orderings of the importance of different factors, there is no guarantee these strategies or expert knowledge will produce "correct" trees. In the worst case, when trees completely misrepresent the true underlying experimental structure, the generalization results of any sampling algorithm will be nonsense. It may be possible to reduce our vulnerability to badly-structured trees by sampling across several possible trees or searching over potential changes in tree structures to avoid worst-case behavior.

4.8 Conclusion

By automating scientific behaviors, we can free researchers to focus on high-level scientific tasks while decreasing the time and cost of generating scientific knowledge. In this chapter, we focus on experimental *generalization*, taking advantage of tree-structure classifications of experiment sets that tell us how to attempt to generalize a known result. Our algorithms, MARGIN, MARGINCHAIN, and MARGINTREE, efficiently allocate samples between conditions within a set,

sets within many potential chains, and chains within an experiment tree for different effect cutoff values. We show in simulations that our algorithms are much more efficient than round-robin sampling and more robust to difficult-to-classify conditions, sets, and cutoffs than other baseline strategies. Finally, we demonstrate how MARGINTREE is much faster than standard round-robin sampling at detecting an effective class of number line teaching sequences in a simulation based on a data from an educational game. Future work includes better identification of reasonable cutoffs and probabilistic guarantees instead of hard ones if trees are large, as well as trying this approach on other domains.

While generalizing scientific results is undoubtedly an important scientific endeavor, it is only really possible if there is already an existing result to generalize. Therefore, the next chapter introduces a method designed to generate interesting scientific results that could then be generalized in the future. Since "interesting" is an ill-defined term, we will focus on specifically finding "surprising" results.

Chapter 5

SAMPLING FOR SCIENTIFIC SURPRISE

5.1 Introduction

As I have argued in previous chapters, the rise of online software has led to a potential explosion in the amount of data available for large-scale experiments in the behavioral sciences. One advantage I have not pressed, however, is that in the past fields such as psychology have traditionally relied on relatively costly methods of recruiting subjects from pools that may not be demographically diverse [48]. This scarcity means that researchers must usually carefully and cautiously plan the specific experiments they wish to run. But now, for fields in which humans can both remotely perform interesting tasks and are willing to do so, online software may allow us to rapidly sample broad experimental spaces to identify areas of interest. This type of opening salvo could help suggest which lab studies would be most worthwhile to run, or even generate scientific results outright.

The previous chapter dealt with choosing between multiple potential experiments to identify maximally general results. This is not the only potential metric an experimental scientist might care about, however. In this chapter, I focus on the opposite problem: identifying experimental conditions for which existing theory does not hold (i.e. ways in which theory does *not* generalize). Specifically, I propose that a good way to use online data sources is to identify conditions which are scientifically *surprising*, which I define as having a result that is different than predicted. By sampling for surprising conditions, the hope is to find "edge cases" in which existing theory fails. This objective function is different than criteria used in previous automated experimentation work, such as learning parameters of all conditions with high confidence or optimizing for some reward function like revenue. In particular, our objective function is nonstationary, as once a condition is known to be surprising, the value of sampling it further decreases. Such a method is designed to be

used as an "opening salvo" by rapidly covering a large set of experimental conditions, identifying and collecting data on surprising conditions which would be worth studying in more detail.

To accomplish this goal, I define a new optimization metric which preferentially samples conditions which are more surprising and of which we are more uncertain, and show how such a metric can be calculated based on subject responses to queries in our task. Next, I propose the stochastic sampling method UWPS, inspired by Thompson sampling from multi-armed bandits [112], and build simulations of our environment to study how it behaves compared to greedy and round-robin approaches. I show that algorithms optimizing for our proposed metric are better than round-robin sampling for identifying and allocating samples to one surprising condition from many unsurprising ones. I also study the effect of delayed feedback, in which the system must decide what to do with new users before all data about past users has been gathered. This simulation shows that even mild delay significantly degrades the performance of the greedy sampling strategy, but not UWPS.

To show the usefulness of our proposed *surprise* criteria and our algorithm, I implement and run UWPS online, using Mechanical Turk as the source of subjects. This algorithm is able to automatically explore a 180-condition experimental space in the field of summary statistics, identifying over a dozen experimental conditions in which subjects responded significantly differently than the standard summary statistics task. In this task, subjects are shown a visual display and answer a question requiring perception and recall from memory; further details are given in Section 5.3.1. Finally, to demonstrate the usefulness of the data collected by our system, I analyze a few of the main effects. Among other things, *Uwps* finds evidence that subjects may be down-weighting circles which are translucent, near the edge of the display, and which appear first, with ordering effects by far the strongest and confirmed to be surprising by a colleague researching summary statistics. This proves that the sampling strategy is able to automatically sample for surprising scientific conditions, gathering data from which researchers can discover novel effects in the domain of summary statistics.

5.2 General Algorithm

5.2.1 Setup

Assume we have some number of experimental conditions C_i, $i = 1, \ldots N$ we wish to explore. On each timestep $t = 1 \ldots T$, an individual arrives, we assign them to one or more experimental conditions, and they produce a scalar observation o_i iid from a fixed but unknown distribution $\mathcal{D}_i$. We assume that the time horizon T is unknown. We also assume we have a prior theory τ which predicts behavior in each condition. We want to identify which conditions C_i cause subjects to behave in a way that is unexpected, that is, different what from our prior theory would predict. We quantify this with a difference function D_i which takes as input an estimated mean and the theoretical mean, $\hat{\mu}_i$ and $\mu_{\tau,i}$. In certain cases, we might care about a more nuanced metric of distance which takes into account the other moments of the distribution, but this is left for future work. We want to reduce uncertainty about the conditions we find, so that we can report them with accuracy. To quantify this, we assume a Bayesian framework where one has an initial prior over the mean, which together with the data forms a posterior P_i. Calculating the variance over this posterior quantifies our uncertainty U_i.

We desire an algorithm for allocating conditions to subjects that tries to find surprising conditions, while making certain that they are surprising. One potential algorithm might be to minimize $\sum_{i=1}^{N} D_i^{c_d} U_i^{c_u}$, where c_d and c_u control how much we care about distance vs uncertainty. Since our uncertainty tends to zero as more samples are gathered, an algorithm which optimizes for this objective has a number of desirable properties. If it is similarly certain about all conditions, it will prefer to sample those which are the most unexpected (larger D_i). If all conditions are similarly different from expected, it will prefer to sample those of which it is least certain (larger U_i). Finally, given infinite data and finite distances, such an algorithm will eventually learn the means of all conditions to an arbitrary degree of certainty.

It is important to note that this objective is nonstationary, which means most standard bandit algorithms will not work. One set of potential exceptions are adverserial bandit algorithms. However, while they can handle nonstationary reward distributions, these variants are meant to be

general for many types of nonstationarity. Our nonstationarity has a specific form we can take advantage of for better performance.

5.2.2 Algorithms

Computing the uncertainty about a condition, U_i, is straightforward. However, D_i, our estimate of the distance of subject responses to what was expected, will always be inaccurate due to the stochasticity in the observations, which makes it non-trivial to decide which experimental conditions to give out to incoming subjects. More subtly, we may need to assign conditions to new subjects while old subjects are still completing an experiment; this *delay* can cause poor performance in algorithms that do not account for it [28]. The problem of delay is complicated further if subjects choose not to complete their tasks, causing some samples never to return an observation. The fundamental issue is that bandit algorithms will sample arms that have already been sampled and simply have not yet returned their data, when it might have been better to explore or exploit other arms instead. This problem is particularly acute with deterministic bandits; our approach is to use stochastic bandit algorithms to avoid pathologically bad behavior when feedback is delayed.

In this chapter, we will describe three potential algorithms for selection of experimental conditions and examine their behavior under simulation, before choosing the best one and deploying it online.

RoundRobin Choose a random ordering, then cycle through the order. This tends to be a good strategy when the uncertainty U_i term dominates the distance term D_i, since evenly distributing samples will decrease uncertainty about all conditions. Especially when the underlying variance of each distribution is similar, this is close to what a Bayesian experimental design approach would do, as it focuses on becoming certain of the means of all conditions. In our experiments, ROUNDROBIN always outperforms simple random sampling, so we use it as our baseline.

Greedy Sample each condition once in a random order. Then for future subjects, calculate D_i using the maximum a posteriori probability (MAP) estimate of the condition mean $\hat{\mu_i}$, and

then choose the condition $\arg\max_i D_i^{c_d} U_i^{c_u}$, or in our case $\arg\max_i \sum_{k=1}^{7} D_i^{c_d} U_i^{c_u}$. This method performs well in ideal environments where we receive all information from each user before the next user arrives, but as we will see degrades rapidly as the delay in feedback increases because it directs too many samples towards conditions which are already in the process of being sampled. Another disadvantage is that if $D_i = 0$ by chance for any condition at any time, it will never be sampled again as long as any other condition has a positive D_i. More generally, it will tend to undersample arms that have small differences due to chance, especially if c_d is high compared to c_u.

Uncertainty-Weighted Posterior Sampling (UWPS) This is a stochastic method inspired by Thompson sampling from multi-armed bandits [112]. Generally, if we maintain a posterior distribution of distances P_i based on a prior and the collected observations, we can draw $\hat{\mu}_i \sim P_i$ to calculate D_i. Then we pick the condition $\arg\max_i D_i^{c_d} U_i^{c_u}$. Thompson sampling has been shown to better handle delay due to the fact that its stochasticity prevents it from oversampling poor arms [28]. Its good performance in the bandit setting [28] means that if c_d is high compared to c_u, it will trade off exploration and exploitation much better than a greedy approach.

One complication is that in many settings, the prior theory does not issue precise predictions; rather, it has some parameters that need to be learned from the data. In this case, one will devote additional samples to the standard condition to learn these parameters, and then use estimates (MAP in the case of greedy or sampled in the case of UWPS) to calculate the differences D_i.

5.3 Summary Statistics

5.3.1 Background

The scientific domain we choose to study, summary statistics, comes from the field of human visual perception. Over the past fifteen years, behavioral research in this area has shown that the human visual system, when presented with an array of similar visual stimuli, possesses a surprising ability

to quickly and accurately extract the average value of that array along a variety of dimensions. For example, behavioral lab studies have shown that the visual system computes and represents the mean size [12, 30], mean orientation [84], mean position [4], and mean color of a group of objects [37]. This ability of the visual system to extract the average value of a set of items is made especially interesting by findings that it can operate even with very short ($\leq$ 200ms) presentation durations [30] and simultaneously across multiple sets of items [19], or across feature dimensions [3]. Additionally, summaries of groups of items can often be obtained with very little memory for the individual items in the group [12, 46], and even for groups of items that the user is not paying attention to [4, 5]. A good review of the area can be found in [119].

The presumed purpose of this ability is efficient representation of visual stimuli. The stream of information that confronts the visual system at any waking moment is far too large for the brain to fully and deeply process, and summary statistics are thought to afford compression, since the typical visual environment is highly regular and redundant. For example, the many leaves contained in a region of the visual field corresponding to the boughs of an oak tree might be accurately represented in the brain by single summary statistics for mean leaf size (0.5°of visual angle) and mean leaf color (deep green). Such a representation strategy is much more efficient than storing individual size and color values for each leaf. This is especially true if there is not a great deal of variation in either of those properties or if close inspection of the leaves is not required at a given moment.

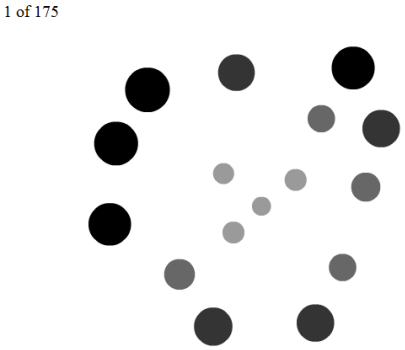
Though much research has shown that humans can compute these summaries, very little is known about how the computation is made for different types of stimuli. If such an ability is regularly used in real-world situations as many believe, it needs to be robust to a great deal of variation in what the to-be-summarized items look like. But since a typical lab study of summary statistics uses only a single set of stimulus parameters for item shape, color, spatial distribution, temporal distribution, etc., very little is known about how the summary is affected by different stimulus properties. For example, how is the summary affected when the items arrive over time, or are different shapes or are presented to peripheral versus central vision? Answers to these and related questions would tell us much about how summary statistics are computed and how useful they are

in real-world behavior, such as intepretation of graphs or other visualizations. However, the space of possible stimulus properties that needs mapping is large and exploring it with traditional lab studies is costly and time consuming. Thus, we would like an automatic method of running experiments, capable of identifying factors which skew or disrupt humans' ability to calculate summary statistic information.

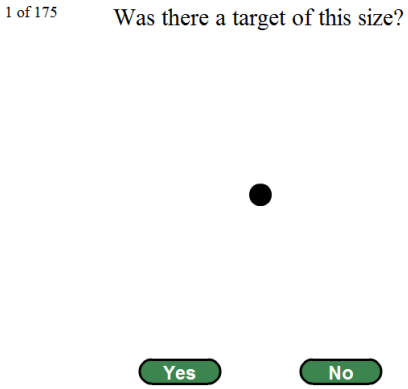
5.3.2 Setup

In order to maximize for scientific surprise, the distance and uncertainty functions D_i and U_i must be defined. To do this, we should first know the tasks subjects will perform and what aspect of their responses are measured. In our summary statistics task, which borrows heavily from the standard method used by Ariely [12] and others in the field, subjects are briefly shown an array of circles. Afterwards, they are shown a query circle, and asked whether a circle of that size was present in the array. An example task, query, and response are shown in Figure 5.1. The question we wish to answer, which is so far not addressed in this literature, is whether certain ways of displaying objects, such as varying their opacity, distance from the center of the visual field, order of appearance, or whether or not they are filled, skews the summary humans compute from these displays. For example, in conditions where large objects are opaque and small objects are translucent, does the summary computation process place less weight on the small objects and therefore produce a skewed or erroneous summary of what was present? This would be of great interest not only to the human perception community, but also to researchers in HCI, especially those working in data visualization. After all, different features of datasets are often conveyed through size, color, opacity, display order, location, or other properties, and thus it is important to understand these choices affect what users perceive.

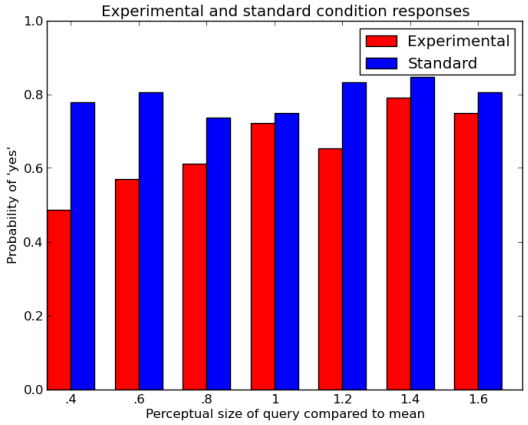
Since the summary statistics literature has not yet studied how altering these factors changes responses, our best guess might be that they cause no effect. In that case, D_i to be the difference between how users respond to queries in an experimental condition and how they respond to the standard condition, in which circles are all opaque, placed randomly, appear simultaneously, and are filled. This corresponds to the sum of the differences in response rates across query sizes, as



(a) Example display



(b) Example query



(c) Example response

Figure 5.1: Each experimental condition consists of flashing a set of 16 circles. The different conditions display the circles in different ways; in this condition, smaller circles are more translucent and closer to the center of the display. Subjects are then asked if a circle of one of 7 sizes relative to the mean circle size had appeared in the display. Using data gathered from our experiment, we can learn a probability of users answering "yes" for the seven query sizes and contrast them to the responses to a standard condition. Here we see that this condition causes people to be more likely to think smaller circles were present.

seen in Figure 5.1(c). One complication we suffer from is the problem that subjects may vary substantially in their age, screen size, amount of recent sleep, distance from eyes to screen, or any number of other factors we cannot easily control or reliably measure. However, we can partially control for these effects in the following manner: whenever we assign condition C_i to user U_t, we also assign a *standard* summary statistics task, C_0. Thus, data gathered about any experimental condition will always compared to data about the standard condition gathered from the same set of subjects, helping us to control for unobserved between-subject variables.

As noted above, we must define a distance metric D_i comparing subjects' responses and expected responses, and we also must define the priors to construct the posterior over which to compute uncertainty. Remember that the response we measure from subjects is the probability they believe one of seven query target sizes was present in the display. As boolean responses, these can be thought of as Bernoulli random variables Q_{ik}, $k = 1, \ldots, 7$ with unknown probabilities of "yes" responses, which we wish to learn. Per the Bayesian approach, we keep 14 Beta(β) posteriors over the likelihood that subjects will believe an object of one of the seven query target sizes was present. There are 7 for the standard condition C_0 and 7 for the experimental condition C_i; these correspond to the 14 bars in Figure 5.1(c). In this work we use $\beta(1, 1)$ priors, corresponding to assuming each query size for each condition has already received one "yes" and one "no" psuedocount before the experiment begins. We can incorporate existing knowledge by altering these β priors if there are already existing studies about specific conditions. Examining one particular condition with random variable Q_{ik}, let Y_{ik} be the number of "yes" responses seen up to this time for query size k in condition C_i, and let N_{ik} be the number of "no" responses. Then we obtain a posterior distribution over the mean of Q_{ik}, $\beta(Y_{ik} + 1, N_{ik} + 1)$. For a particular query size k, the distance between the standard condition C_0 and experimental condition C_i is then just $D_{ik} = |E[Q_{ik}] - E[Q_{0k}]| = |E[\beta(Y_{ik}+1, N_{ik}+1)] - E[\beta(Y_{0k}+1, N_{0k}+1)]| = |\frac{Y_{ik}+1}{Y_{ik}+N_{ik}+2} - \frac{Y_{0k}+1}{Y_{0k}+N_{0k}+2}|$. This uses the Betas to get a MAP estimate of the distance. Likewise, the uncertainty of our estimate for one query size is $U_{ik} = var(\beta(Y_{ik}+1, N_{ik}+1))$. Finally, we weight distance and variance ($c_d = 1, c_u = 1$) equally, thus our goal is to sample to minimize $\sum_{i=1}^{N} \sum_{k=1}^{7} D_{ik} * U_{ik}$. Though the details may be somewhat complicated, the general goal is not: we prefer to sample the conditions in which subjects are much

more or less likely to believe objects of a particular size are present than in the standard condition, and we prefer to sample conditions in which we are not sure of our results.

5.3.3 Sampling algorithms

The implementation of the sampling strategies we described earlier is now straightforward. We choose $c_d = 1, c_u = 1$ to put equal weight on distance and variance, meaning our objective is $\sum_{i=1}^{N} \sum_{k=1}^{7} D_i U_i$, for the true distances D_i.

RoundRobin Remains the same.

Greedy After the initial sampling phase, greedily choose the condition $\arg\max_i \sum_{k=1}^{7} D_{ik} * U_{ik}$, where D_{ik} is computed using the MAP estimate of D_{ik} described above.

UWPS The Thompson-like strategy corresponds to drawing $\hat{\mu}_{ik}$ and $\hat{\mu}_{0k}$ from the experimental and standard beta posteriors we have for each query size, respectively, calculating $D_{ik} = |\hat{\mu}_{ik} - \hat{\mu}_{0k}|$, and choosing condition $\arg\max_i \sum_{k=1}^{7} D_{ik} U_{ik}$.

5.4 Simulation

There are two primary questions we wish to answer. The first is if minimizing for our suggested metric is useful for identifying surprising conditions, as opposed to merely sampling evenly to gather information about all conditions. The second is which algorithm performs best at minimizing our suggested metric. Both questions are complicated by the fact that, in an online environment, we do not receive samples in an ideal fashion. In particular, we suffer from the problem of *delay*; since we are gathering so many subjects so quickly, we may be required to give out conditions to new subjects before we have received data from previous ones.

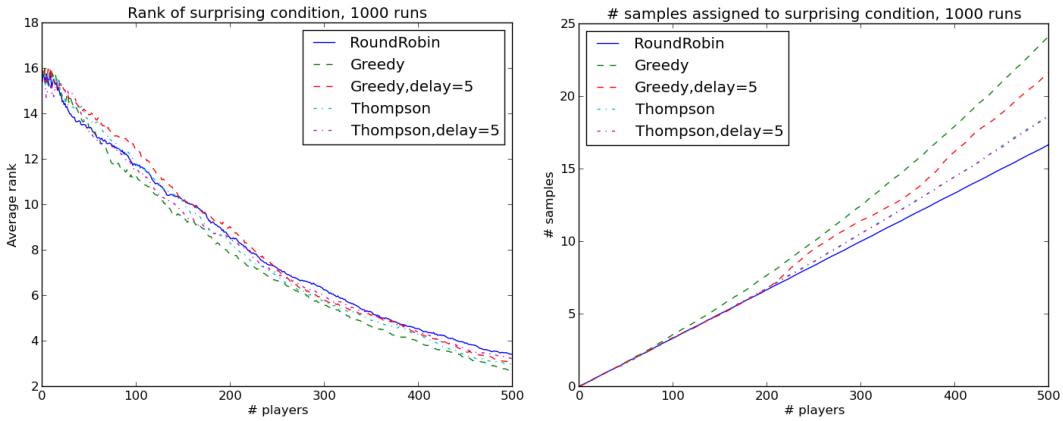
To answer the first question, we first created a simulation of our summary statistics environment. Remember that the parameters of interest for each condition are the probabilities that subjects will respond "present" to each of the seven query sizes. We created a standard condition based on the the normal distribution pdf ϕ, giving us seven probabilities [$\phi(-1)$, $\phi(-0.66)$, $\phi(-0.33)$,

$\phi(0)$, $\phi(0.33)$, $\phi(0.66)$, $\phi(1)$]. Then, 29 experimental conditions were created with exactly the same parameters as the standard condition. One experimental condition was designated "surprising" and had a bimodal response pattern: its query probabilities were set at [$\phi(-1)$, $\phi(0)$, $\phi(1)$, 0, $\phi(-1)$, $\phi(0)$, $\phi(1)$]. We assumed that each subject, once assigned an experimental condition, would be queried five times for each query size for both the experimental and standard conditions, for a total of 5*7*2 = 70 datapoints per subject.

The algorithms were run 100 times in this simulated environment assuming we would get a total of 500 subjects. After each subject, we graphed both the average number of samples allocated to the surprising condition and the average rank of the surprising condition when sorted by empirical distancefrom standard condition. An ideal algorithm would tend to assign more samples and have a lower rank (ideally 1) for the surprising condition. Finally, to study how these algorithms would run in an realistic environment, we also introduced some delay. In an environment with delay d, the algorithms were only allowed access to data gathered up to d players ago. This had no effect on ROUNDROBIN, which ignores incoming data, but affected the performance of GREEDY and to a lesser extent UWPS.

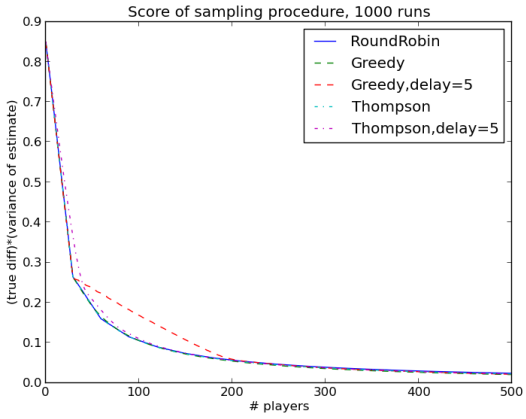
To understand the effects of delay, we created a less extreme simulation with the same standard condition. Here, 15 experimental conditions were identical to the standard condition, and 15 experimental conditions were made by introducing uniform noise to each probability in $[-0.2, 0.2]$, clamping probabilities to $[0, 1]$ as needed. Armed with this ground truth data, we can then calculate the true distance from standard for each condition, and at every timestep can graph the sum of (true distance)*(variance of beta posterior) across the experimental conditions. Note that the true distance of each condition identical to standard is zero, so sampling them is not helpful at reducing the objective.

The results of both simulations are shown in Figure 5.2. In the first environment, we see that ROUNDROBIN is slightly worse at finding the interesting condition and significantly worse at allocating samples to it. This is because it spends too many samples on the uninteresting ones. This suggests that, if we are most interested in identifying places where current theory fails, attempting to minimize distance*uncertainty is a useful metric. In the second environment, we see that in



(a) Rank (Environment 1)

(b) Samples (Environment 1)



(c) Performance (Environment 2)

Figure 5.2: Simulation results. In the first environment where we want to identify the single "surprising" condition, we see that both GREEDY and UWPS strategies are slightly better than ROUNDROBIN at identifying the surprising condition, and significantly better at allocating more samples to the surprising condition, even in the face of delay. In the second environment, we see that GREEDY is very vulnerable to delay. The *Thompson* strategy, on the other hand, is barely affected by delay and performs very well.

online environments with delay, the performance of GREEDY degrades rapidly. This is because it chooses a single condition repeatedly until data returns after the delay - most of those samples would have been better served reducing the variance of other conditions. UWPS is robust to this delay since it spreads samples among many conditions during the delay period.

5.5 Experiment

5.5.1 Task and Parameter Description

Subjects were recruited from Mechanical Turk from September 16, 2014, to September 19, 2014. Each subject went through a short tutorial explaining the task. They could not pass the tutorial without answering each question correctly. The tutorial included displays with circles appearing sequentually, as outlines, and with varying levels of opacity to prepare them for all possible conditions. They could not continue to the experimental conditions until they passed all three tutorial levels correctly.

In each trial, a grand mean circle diameter was chosen between 10 and 15 pixels. Users were exposed to a fixation point at the center of the screen for 500ms, followed by a display of circles lasting from 400ms to 4800ms depending on the condition. All circle centers appeared within 200 pixels of the center of the display. There was then a 500ms gap, followed by a query of one of seven circle sizes at 40%, 60%, 80%, 100%, 120%, 140%, and 160% of the perceptual size of the mean. Users then answered either "yes" if they thought a circle of that size had been present, or "no" otherwise. The next trial's fixation period then began immediately.

Here, "perceptual" size refers to an adjustment made because humans do not perceive circle sizes as growing linearly with respect to radius or area, but instead as at a rate of $A^{0.76}$ [111]. Thus, the query circle at size 40% was the circle whose perceived area was 40% of the mean circle size's perceived area, according to this adjustment. The display of circles always consisted of four groups of four circles. In each group, specific circle sizes were drawn from perceptual sizes 40%-60%, 70%-90%, 110%-130%, and 140%-160% randomly such that the mean perceptual circle size of each group was always 50%, 80%, 120%, and 150% of the grand mean circle size, respectively.

The randomness of the grand mean and circle size in any group was done to make it more difficult for users to learn patterns about circle sizes from one trial to the next. Note also that no circle of exactly the same size as a query circle ever appeared in the display, and in fact no circle within 10% of the perceptual size of the grand mean ever appeared.

We deployed an experiment with 180 conditions, with condition 1 being considered the standard. In the following description, parameters for the four circle groups are always given in order of smallest to largest size. The experimental space was constructed from the following factors:

Outline All circles in the display and the query are either *Filled*, or *Outline*.

Opacity *LargeOpaque* sets the alphas of the groups to $(0.4, 0.6, 0.8, 1.0)$, so that the largest circles are most solid. *SmallOpaque* sets the alphas of the groups to $(1.0, 0.8, 0.6, 0.4)$. *AllOpaque* sets all alphas to 1.0.

Distance *LargeCenter* constrains the pixel distances of the circles from the center to be ($[0, 100]$, $[100, 150]$, $[150, 175]$, $[175, 200]$), while *SmallCenter* constrains the distances to the reverse, ($[175, 200]$, $[150, 175]$, $[100, 150]$, $[0, 100]$). *Random* selected random distances for all circles from $[0, 200]$.

Ordering There are multiple orders in which circles can appear. In *Simultaneous400* and *Simultaneous800* they are all flashed together for 400ms or 1200ms. In *Group200LargeFirst* and *Group800LargeFirst*, the two larger groups are shown first, then the two smaller ones, for 200ms and 800ms (so the entire display lasts 400ms and 1600ms), and likewise with order reversed for *Group200SmallFirst* and *Group800SmallFirst*. Finally, in *Sequential150LargeFirst* and *Sequential300LargeFirst*, circles are shown one at a time with no overlap or gap between them. The first four come from the largest group, the second four come from the next largest group, and so on, with each circle lasting 150ms and 300ms, respectively (so the entire display lasts 150*16=2400ms and 300*6=4800ms). The orders are reversed for *Sequential150SmallFirst* and *Sequential300SmallFirst*.

These parameters were chosen because their effects have not yet been studied, to the best of our knowledge, but might impact how users perceive the display. For example, the standard condition is *Filled,AllOpaque,Random,Simultaneous400*. If we then make the smaller circles translucent, will subjects be less likely to notice them and so be less likely to think that small circles were present? Or if we make large items appear near the periphery, will they be harder to see? Figure 5.1(b) contains subject responses collected from Mechanical Turk for the display shown in Figure (which originates from the condition *Filled,LargeOpaque,SmallCenter,Simultaneous400*). These results suggest that the combination of *SmallCenter* and *LargeOpaque* causes subjects to be less likely to think that smaller items were present.

Subjects were assigned experimental conditions by using UWPS, ranking the conditions, and selecting the top four and the standard condition. Each query size in each condition was queried five times, giving a total of 5*7*5 = 175 trials per subject and lasting about 15 minutes. Subjects were compensated $1, and were free to leave or reject the task at any time; for efficiency reasons, UWPS only used their data once they finished all trials.

5.5.2 Results

The output of our system can be visualized by ordering conditions from most to least surprising (largest to smallest distance from standard), along with a description of the condition and a graph showing the differences between them. This is shown and described in Figure 5.3.

It is difficult to calculate statistical significance in closed form since different conditions have different numbers of samples, each condition consists of seven other conditions, and we have partial violation of independence of samples because each person contributes multiple samples to multiple conditions. With all of that being said, we can still get a good sense of what sorts of distances might be likely to happen by chance. We do this by estimating the best parameters describing subject responses to the standard condition, then drawing from it for the minimum number of samples observed for any condition. We can then measure the distance between the parameters we would have estimated from the samples and the true parameters, repeating this process one million times. Using the Bonferroni correction, we see that distances of greater than 0.832 have less than a $\frac{0.05}{180}$

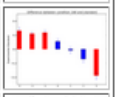
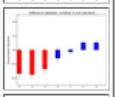
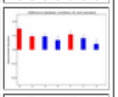
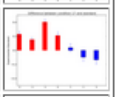
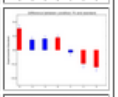
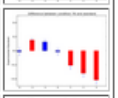
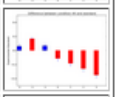
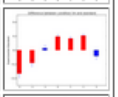
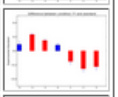
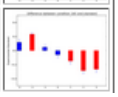
Condition ID	Difference	# Samples	Opacity	Distance	Ordering	Outline	Graph
168	1.361111	504	SmallOpaque	SmallCenter	Sequential,300,LargeFirst	Outline	
8	1.311688	539	LargeOpaque	LargeCenter	Group,200,SmallFirst	Filled	
45	1.270270	259	AllOpaque	Random	Sequential,150,LargeFirst	Filled	
17	1.269231	364	LargeOpaque	LargeCenter	Sequential,300,LargeFirst	Filled	
75	1.258065	434	SmallOpaque	SmallCenter	Sequential,150,LargeFirst	Filled	
78	1.250000	504	SmallOpaque	SmallCenter	Sequential,300,LargeFirst	Filled	
49	1.196970	924	SmallOpaque	Random	Sequential,300,LargeFirst	Filled	
54	1.194444	504	AllOpaque	Random	Sequential,300,SmallFirst	Filled	
77	1.186916	749	AllOpaque	SmallCenter	Sequential,300,LargeFirst	Filled	
165	1.156863	714	SmallOpaque	SmallCenter	Sequential,150,LargeFirst	Outline	

Figure 5.3: A screenshot of output generated by our experiment. Bars highlighted in red are those which are statistically significant using Fisher's exact test; these are meant primarily as guidelines rather than true statistical confidence given the violation of the independence of samples criteria and the large number of experiments being run. We can also run simulations to estimate a measure of statistical significance. The most conservative estimate is that any distance greater than 0.832 is statistically significant, which includes all the ones seen here.

probability of occurring by chance.

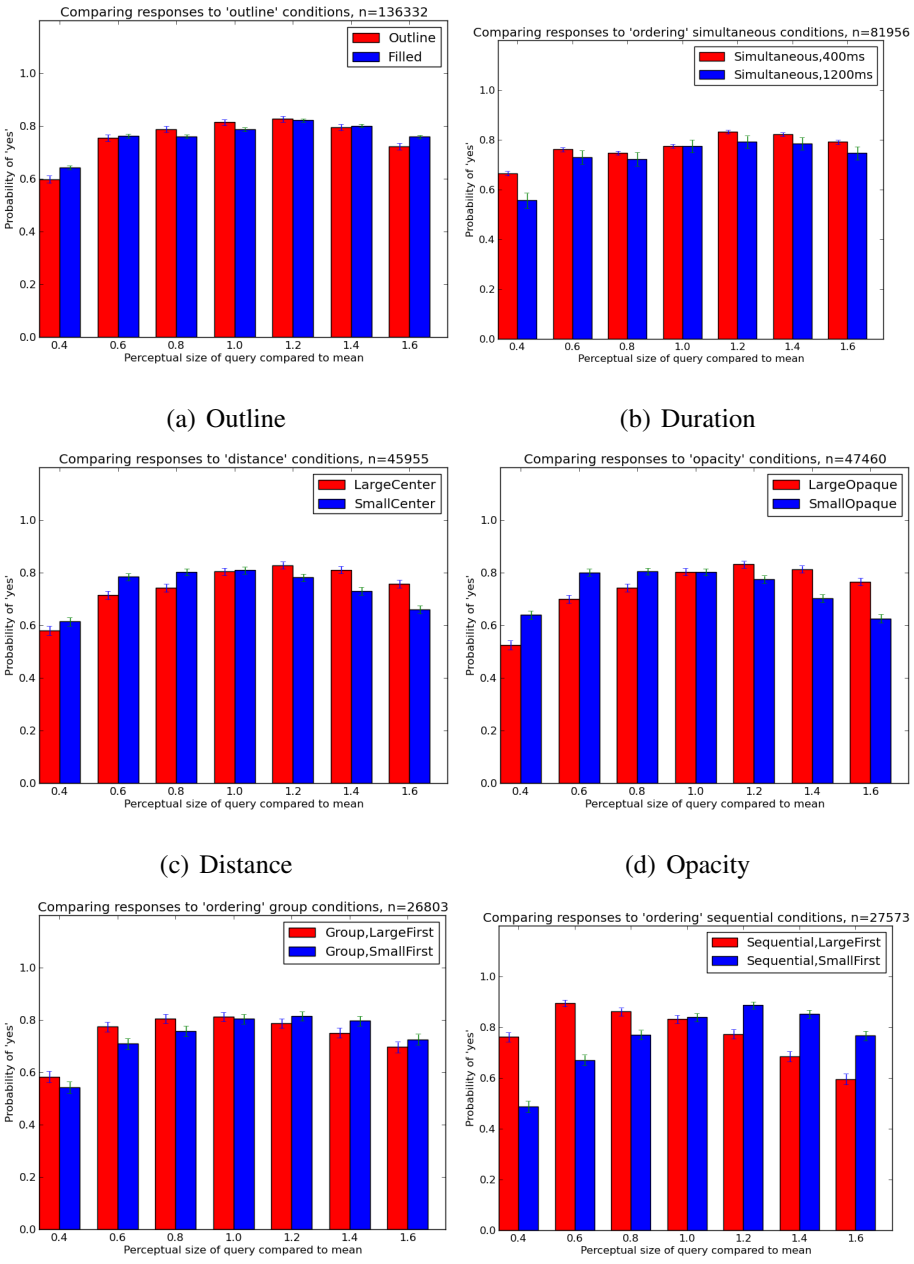
5.5.3 Analysis

Our system neither requires nor uses any structure in the experimental conditions. Nothing stops the researcher from specifying distinct, uncomparable conditions, in which case attempting to find patterns in the results will require prior knowledge of underlying domain theories. However, in this experiment our condition space was generated combinatorially from a number of factors. This gives us the ability to perform some simple analysis to discover general trends.

For space reasons, we restrict ourselves to just a few interesting comparisons, shown in Figure 5.4. Figure 5.4(a) suggests that changing circles to be filled or outlines does not change the pattern of responses. In other words, the summary computation process appeared to treat outlined and filled circles similarly. On the other hand, Figure 5.4(b) suggests that, at least in simultaneous displays, longer displays reduce users' likelihood of believing circles of extreme size were present. Figure 5.4(c) suggests that subjects preferentially weight items in the center when computing the mean size, since placing large objects in the center increases the probability of believing large objects were present and vice versa. Likewise, Figure 5.4(d) leads us to believe that subjects more highly weight opaque items in the summary statistic computation. Finally, Figures 5.4(e) and 5.4(f) suggest a recency effect, where the summary users compute is biased towards the most recently seen circles, and that this effect is stronger when items are presented one at a time. To the best of our knowledge (and as verified by a domain expert), none of these effects have previosuly been documented in the literature on summary statistics, and thus represent promising directions for future study in this field.

5.6 Discussion

We were able to automatically discover many scientifically interesting conditions that were extremely unlikely to have occurred by chance, so in this sense the experiment was a success. The key question, however, is whether or not the same thing could have been done in a standard fash-



(a) Outline (b) Duration

(c) Distance (d) Opacity

(e) Group (f) Sequential

Figure 5.4: Simple analysis of data gathered during the summary statistics experiment. Probabilities of response are computed by marginalizing over all other factors; for example, data in the *Group,LargeFirst* includes all data where the **Ordering** factor was set to *Group,400,LargeFirst* or *Group,1200,LargeFirst*. 95% Wald confidence intervals are given. The number of samples is the total number of trials used to generate the graph, with any given subject contributing many trials. This number varies because different conditions were assigned to different numbers of people, with the standard condition being assigned to everyone. As a reminder, the *group* conditions have the circles appearing in two groups, while *sequential* conditions have circles appearing one at a time.

ion. Have we gained anything from using this method? Unfortunately, it is difficult to compare the data we gathered from our experiment to data that might have been gathered in the lab. On the one hand, we gathered 110 subject-hours of data over a period of two and a half days by a single student, for approximately $500. An equivalent set of lab experiments would cost approximately $1100 and would require about 160 hours of researcher time.

On the other hand, the data we gathered may be noisier, with many additional factors (screen size, background distractions) unmeasured and uncontrolled. While using only within-subjects comparisons with random ordering of trials mitigates this problem, factors such as being distracted by a television are likely to reduce effect sizes by adding independent noise to all conditions. Another downside is that web-based data collection prohibits use of techniques such as eye-tracking or subject interviews. In exchange, we may be getting data from a wider pool of subjects [24] and avoiding or mitigating certain effects from participating in an experiment in a lab [110]. This makes the two data types difficult to compare.

Furthermore, due to resource constraints, a normal researcher would never even attempt to run a series of experiments in this fashion. The standard laboratory experimental approach in behavioral research is to test one or two factors at a time, greatly reducing the number of conditions and required data, as opposed to the "shotgun" approach taken in this chapter. As a result, we believe that our method should be considered a new tool for running experiments in conjunction with existing ones, with its own unique sets of advantages and disadvantages.

That being said, our experiment generated a great deal of data about this domain, some of which may be valuable for HCI researchers, especially those working in data visualization. We will thus take a moment to reflect on what a researcher could do with the data we gathered.

Given the difficulty of applying normal statistical tests to data gathered by our algorithm, and the noise introduced by failing to control potentially important variables in the online setting, some caution is warranted. The most cautious use of our algorithm is to find experimental conditions worth exploring more fully in a controlled laboratory setting. Our initial examination of the effects shown in Figure 5.4 suggests that order effects, opacity effects, distance effects, and display time effects all affect how the human visual system computes summaries of what it has seen. This tells

us which experiments might be worth running, in what order, and provides an estimate of the likely results.

A more ambitious use of our algorithm is to trust the results it produces outright, in which case the researcher could consider extensions to existing theory and generate entirely new experiments to run using these as a launching point. Figure 5.4(b) tells us that displaying the objects for 1200ms instead of 400ms decreased the probability that subjects would think any particular object was present, especially for more extreme query sizes. One possibility is that subjects had more time to examine objects and store detailed information, in which case there should be a limit to this effect once sufficient time is given to devote attention to every object or once working memory capacity has been reached. To test this, we could run an experiment with longer displays or fewer objects and see at what point no more decrease is observed. Indeed, it may be possible to adapt work from the optimization literature capable of identifying and running such experiments automatically. Another possibility is that, while the visual system appears to have built-in "hardware" that computes average sizes quickly, a slower mental process may be able to compute other statistical properties such as variance. With more time, these estimates might become more accurate. This could be tested by asking subjects to estimate the number of objects of different sizes in the display, and studying how statistical properties of their estimates changes depending on the duration and distribution of object sizes in the display.

Alternatively, Figures 5.4(e) and 5.4(f) suggest recency effects are present when calculating average sizes from stimuli presented across time. Since the sequential results are much more pronounced, we might theorize that the visual cognition system applies a discount factor to objects in the summary statistics computation, such that objects seen in the far past have the least weight. Is the function linear, exponential, or something else? Does the same effect occur when instead of only one circle being seen at a time, subjects pay attention to only one circle at a time? If so, and there are regularities in the order in which objects catch our attention (for example, most colorful first), there may be predictable biases in how we perceive groups of objects in particular contexts. These are questions that could be answered with more focused experiments, but knowledge of these effects already improves our understanding of how summary statistics are computed.

Regardless of the explanations for the results that we found, our method was able to provide us with novel information about how different methods of displaying objects affect subjects' beliefs about what items were present. Furthermore, we only examined a few of the simplest possible effects; there are many more potential comparisons that could be run, including interaction effects. We also have completely ignored more fine-grained data such as the time it took for users to respond, which could give us information about which displays make subjects more or less uncertain. We also have the specific combination of conditions given out to each user, which could tell us if some displays have lasting effects that affect summary statistics computations on future ones. This proves the potential usefulness of our system as a new tool in the scientific toolbox for researchers in behavioral science.

5.7 Future Work

There are many potential avenues for future work, such as running more extensive simulations, improving the sampling algorithm, and proving theoretical guarantees in the face of delay as has been done recently in multi-armed bandits [51]. We will list only a few of the more interesting ones here.

Any automatic experimentation method requires some method of allocating samples. We have proposed *scientific surprise* as an appropriate and interesting metric to optimize. This method is appropriate when there is a concept of "distance" from an expected result. Yet such a distance metric may not exist, or may be difficult to compute, such as when existing theories provide only qualitative predictions. Other optimization criteria may be appropriate in this case, such as identifying conditions which share most factor settings yet have very different results; this is conceptually related to the splitting criteria in decision trees using information gain [87]. Furthermore, there are other criteria in many sources of online data, such as maximizing learning in educational technology [73]; further work is required to develop algorithms capable of trading off between scientific surprise and other objectives.

The algorithms we have proposed here are highly general, in the sense that they takes any number of arbitrary experimental conditions. However, this generality is also a weakness, because

we cannot take advantage of structure that relates conditions to each other. Experimental spaces grow exponentially with the number of factors under consideration, and become infinite with the introduction of continuous variables. In these situations, we must make simplifying assumptions, such as assuming conditions sharing many factor values are similar. Work from bandits in metric-spaces [54], hierarchical automatic experimentation [72], or optimal experimental design are likely to be useful for this purpose.

5.8 Conclusion

Recent years have seen an explosion of data available to researchers in the behavioral sciences. Taking full advantage of this data requires new, automated methods for exploring experimental spaces. In this chapter, we propose optimizing for scientific surprise, which we define as finding experimental conditions whose results are most different than expected. Identifying surprising experimental conditions would point to future experiments to run in more controlled settings, or point to flaws or gaps in existing theories that can be improved. We show under simulation that optimizing for a combination of distance from expected and uncertainty is more effective at identifying and assigning samples to scientifically surprising conditions. Furthermore, we show how delayed feedback, a common issue in automatic experimentation, degrades performance of a greedy optimization algorithm. We propose a stochastically greedy algorithm called UWPS inspired by Thompson Sampling algorithms from multi-armed bandits, and show its robustness to delay in another simulation.

Next, we implement and deploy our system online. We study a 180-condition experimental space composed of four different factors in the domain of summary statistics, drawing data from users of Amazon's Mechanical Turk. Our system is able to identify over a dozen interesting experimental conditions to which users respond significantly differently than a standard condition. Furthermore, we are able to pool data together and examine just a few of many possible comparisons. We discover that displaying outlined instead of filled circles has little appreciable effect, while lengthening displays decreases the probability of subjects believing extreme-sized objects to have been present in the display. Furthermore, our results suggest that objects which are translu-

cent, located peripherally, or appear earlier are all downweighted in the estimation of the mean size. These are all scientifically novel findings, and our ability to uncover them shows the effectiveness of using online data sources to rapidly map out experimental spaces in general, as well as our optimization criteria and sampling algorithm specifically.

One overarching theme of this thesis has been the need for new experimental sampling strategies if we are to take advantage of online software as a new source of data. This chapter has shown that delay can be a problem in online environments, as samples are distributed so rapidly that previous ones do not have time to finish. However, a different fundamental problem that has so far been ignored is that most users of online software are unpaid and expect to receive something from that software; should we assign users to unpleasant conditions (even if they would be valuable to study), those users may leave and never return. This is the subject of the next chapter.

Chapter 6

THE REWARD-KNOWLEDGE TRADEOFF

6.1 Introduction

So far, I have introduced systems which run experiments for a single purpose, such as optimization for some objective function like user learning or discovery of scientifically interesting information. Yet in many domains (ex. education), researchers may care about both objectives. Just as systems like ASSISTMENTS promise to both educate and assess simultaneously [92], ideally we should both educate and experiment at the same time. With programmatic control of educational material and automatic data collection, such systems could provide ever more effective educational experiences while also generating scientific knowledge as they collect data about the comparative effectiveness of different representations of knowledge or teaching strategies. This knowledge could then be used to inform educational theories, potentially producing new and better options to be investigated by the system.

Now an additional complication arises due to the high-stakes nature of the domain: users of educational software should learn from the system, so testing ineffectual conditions can cause real harm. At the same time, knowing certain conditions are ineffective may help us draw conclusions about how to deliver future content. This inescapable tradeoff suggests the experimenter should be allowed to explicitly specify the relative worth of teaching users (by giving out the best conditions) against the gain of scientific knowledge (by giving out sub-optimal conditions to better assess their worth). Then an algorithm can sample the different experimental conditions, with a bias in favor of finding and exploring the better ones depending on the specified weighting. As the algorithm obtains better estimates of the effects of different conditions, it should eventually converge to placing all students in the most effective condition. This problem fits nicely into a multi-armed bandit formulation, where the arms are conditions and the reward is user learning, so I attack it

from this angle.

My contributions are as follows. First, I formulate a dual-objective bandit to optimize a weighted combination of the 95% confidence interval sizes around the condition means, and user test performance. I then introduce UCB-EXPLORE, a modification of an existing multi-armed bandit algorithm that tries to directly optimize for this user-specified tradeoff. Second, I analyze the performance of this algorithm in a 64-condition simulation with parameters learned from a real-world experiment involving different ways of displaying number lines. UCB-EXPLORE better optimizes the weighted objective compared to existing bandit algorithms, and in the simulation appears fairly robust to changes in its parameters. Finally, I show how to use the samples generated from our algorithm to identify likely and unlikely two-way interactions between factors, and validate our hypotheses on a separate dataset.

6.2 Related Work

6.2.1 Adaptive Trials

The most relevant related work comes from clinical trials, another example of high-stakes domains where it may be undesirable or unethical to assign patients to certain experimental conditions. Over the years, researchers have developed different methods for minimizing patient harm while trying to identify the best treatments. Some examples include play-the-winner, drop-the-losers, sample size re-estimation, adaptive treatment-switching, and so on; for a review, see [31] or [17]. The adaptive randomization designs are closest in spirit to our work: they bias the randomization in favor of successful conditions and away from failed ones [123] [118].

These strategies are often heuristic and offer no guarantees. However, clinical trials can also be formulated as multi-armed bandit problems, for which algorithms with theoretical performance guarantees are known. The closest work in this space is perhaps by Kuleshov et al., who propose the use of existing multi-armed bandit algorithms for the allocation of users to experimental conditions and show simulations suggesting that more patients are successfully treated [58]. Similar empirical investigations of bandits have been undertaken in the domain of web content retrieval by Vermorel

et al. [115]. They focus on the standard bandit formulation in which the only objective is to maximize reward; in this chapter, we also care about scientific knowledge.

6.3 UCB-Explore

Our algorithm, called UCB-EXPLORE, is a variant of UCB1. For a description of both bandits and UCB1, see Section 2.5. As noted in that section, changing the scaling factor c on UCB1's confidence bounds often leads to improved performance in practice: thus, the key idea behind UCB-EXPLORE is to self-adjust c in response to mistakes. Our algorithm takes as input a set of arms with unknown reward distributions D_i, a function CI to calculate confidence interval sizes, a weight w controlling the tradeoff between reward and confidence interval sizes around the arm means, and a multiplier m that controls how quickly c changes. Good choices of CI in general depend on the shape of the reward distributions D_i, though methods such as the centered percentile bootstrap [105] allow reasonable estimates in most situations. In our example, the arms are Bernoulli processes generating "success" or "failure" depending on whether the student answers a test question correctly, so a reasonable choice for CI is the Wilson score interval [120].

Let N be the number of arms, r^t be the reward received on pull t, and Δ_j^t be the size of the 95% confidence interval of arm j on round t. Our goal is to maximize the expression $w \sum_{t=1}^{T} r^t - (1 - w) \sum_{j=1}^{N} \Delta_j^t$. That is, we want to maximize the total reward received, but minimize the sizes of the 95% confidence intervals sizes, with some weight w between both goals. This type of goal makes sense if the experimenter is able to assign "worth" to confidence interval sizes, in terms of reward. For example, an educational institution might be given funding based on students' standardized test scores, and be willing to pay a certain amount of money for smaller confidence intervals about certain educational interventions. As an alternative interpretation, note that reward grows without bound while the confidence interval sizes cannot go below 0, so w can be thought of as a rough "switching point" after which the algorithm will aim primarily to gather more reward. Say that the experimenter knows a reasonable reward is 0.6, would roughly like the reward term to dominate after n pulls, and calculates that after this many pulls the confidence intervals can be expected to decrease by about 0.3 per pull. Then setting $w = 0.33$ causes the reward term to overtake the

confidence intervals after about n pulls.

Note that this objective is difficult to optimize directly. This can be seen by considering the case $w = 1.0$, where we are focused on reward: it is computationally intractable to optimally pull arms to optimize the objective [85]. It must therefore be similarly intractable to optimize in the general case. As such, we propose UCB-EXPLORE as a heuristic algorithm which lacks guarantees but seems to work well in our scenario. Our algorithm is shown in Algorithm 3. Let $c^1 = 1.0$ be our initial scaling factor, as in UCB1. We first pull each arm once; at subsequent times t, we choose the arm $j(t) = \arg\max_i \hat{\mu_i}^t + c^t\sqrt{\frac{2\ln t}{n_i^t}}$. When c^t is very large, $\hat{\mu_i}^t$ has little effect, and we will tend to choose arms that have fewer pulls (more exploratory). When c^t is very small, $\hat{\mu_i}^t$ dominates, so that we tend to pull only the arms with highest empirical means (more exploitative). So far, then, our algorithm is simply a tuned variant of UCB1.

The key change in our algorithm is that we increase or decrease c if we choose the wrong arm to pull. Say that arm i has been pulled at times $t_1, t_2, \ldots$. Then the rewards of all pulls of arm i up until this time are $R_i^t = r_i^{t_1}, r_i^{t_2}, \ldots$. Let $s_i^t = w\hat{\mu_i}^t + (1-w)(CI(R_i^t) - \mathbb{E}_{r_i^t}[CI(R_i^t + r_i^t)])$: that is, s_i^t is the weighted combination of the expected reward and the expected decrease in confidence interval size if we pull arm i. The calculation of $\mathbb{E}_{r_i^t}[CI(R_i^t + r_i^t)]$ in full generality requires a posterior estimate of D_i; since we are working with Bernoulli trials, we can estimate $p(r_i^t = 1) = \hat{\mu_i}^t$ and calculate the expectation directly.

When should we adjust c? In UCB-EXPLORE, we ask whether we should have picked the arm with the second-highest upper confidence bound. Let $b_i^t = \hat{\mu_i}^t + c\sqrt{\frac{2\ln n}{n_i^t}}$ for each arm i. Without loss of generality, assume that $b_1^t \geq b_2^t \geq b_i^t$, $i = 3, \ldots, K$. If $s_2^t > s_1^t$, then our algorithm has made an error: it could have (greedily) obtained a better tradeoff respecting the researcher's decision of w by pulling the second best arm. If $\hat{\mu_1}^t > \hat{\mu_2}^t$, then the algorithm was exploiting too much, so we set $c^{t+1} = mc^t$. If $\hat{\mu_2}^t \leq \hat{\mu_1}^t$, then the algorithm was exploring too much, so we set $c^{t+1} = c^t/m$. We then pull the arm $j(t)$ and continue. It is important to note that this algorithm is heuristic in nature, but seems to work well in our simulation. It may be possible to develop a more theoretically-motivated algorithm to maximize for this weighted goal, which we leave to future work. In either case, if the algorithm respects w in its behavior and seems relatively robust to the

Algorithm 3 UCB-Explore

Require: a tradeoff w, multiplier m, bandit arms $A_{1,\ldots,N}$

$c = 1.0$

for $j = 1$ to N **do**

$r_j = \textsc{Pull}(A_j)$

$\mu_j = r_j$

$n_j = 1$

$R_j = \{r_j\}$

for $t = N + 1$ to ∞ **do**

for $k = 1$ to N **do**

$b_k = \mu_k + c\sqrt{\frac{2 \ln t}{n_k}}$

$c_j = \textsc{CalculateCI}(R_j)$

$e_j = \textsc{CalculateExpectedNextCI}(R_j)$

$s_k = w\mu_i + (1 - w)(c_j - e_j)$

$u = \arg\max_i b_i$

$v = \arg\max_{i \neq u} b_i$

if $s_v > s_u$ **then**

if $\mu_u > \mu_v$ **then**

$c = mc$

else

$c = \frac{c}{m}$

$r_j = \textsc{Pull}(A_u)$

$\mu_j = n_j\mu_j + r_j$

$n_j = n_j + 1$

$R_j = R_j \cup r_j$

choice of m, then we will have achieved our goal. We will see in Section 6.5 that this is the case in our scenario.

6.4 Example application

We will examine the performance of our algorithm with a 64-arm simulation whose parameters are drawn from real-world data. This will demonstrate the feasibility of our approach in a real-world situation. In this simulation, we will try to identify how the appearance of a "practice" number line affects player performance on a randomized "test" number line, a particularly challenging problem since we expect the effect sizes to be small given that the intervention is one number line long. We choose number lines as they are a well-studied and commonly-used pedagogical tool, and a fair amount of evidence suggests that much whole and rational number knowledge is organized around mental number lines [10], [103]. We will first describe the game from which we collected our data, as well as the factors that vary between number lines.

6.4.1 Treefrog Treasure

Once more we will be relying on data from Treefrog Treasure, described in Section 3.2. Data for this experiment is drawn from BrainPOP [20], an educational website aimed at school-aged children. Our dataset consists of 34,197 players, who played from June 3, 2013 to June 20, 2013.

As before, we consider each player as a sequence of many pairs of number lines, and treat each pair as an experimental unit. This gives us 361,738 pairs. This potentially violates independence assumptions in classical statistical tests, but greatly increases the amount of available data we can use to estimate our arm reward distributions. We will strictly adhere to the correct assumptions when we attempt to generate hypotheses and validate them on a new dataset, later.

The appearance of the first number line in each pair constitutes the experimental condition - the full set of factors is specified in Table 3.1, with illustrations in Figure 3.3. We care primarily about Ticks, Animations, Backoff Hints, Target Representation, and Label Representation. This gives us 64 separate conditions, one for each combination of factor settings.

There are additional complexities in the sampling distributions in this dataset that are not relevant to this work; for a more thorough explanation, refer to Section 3.6.2. The important point is that we can obtain an unbiased estimate, for each experimental condition, of the probability that a player receiving a number line with those parameters will reach and solve the randomized next "test" number line correctly on the first try. For our simulation we will assign each arm the associated mean estimated from our data, and draw simulated samples from the arms by flipping coins with the specified probability of success. These probabilities range from 0.38 to 0.47, with the vast majority falling in $[0.41, 0.45]$. Since the arms are Bernoulli in nature and the probabilities are close to 0.5, the variance is nearly the maximum possible for distributions in $[0, 1]$.

6.5 Simulation

Empirical MAB research such as [58] and [115] indicates that MAB algorithm performance is very sensitive to the exact values of parameters, and that tuned simple algorithms, such as ϵ-greedy, outperform theoretically-motivated algorithms such as UCB1. One reason that tuning affects reward is that different settings of these parameters can result in a tradeoff between identifying the best mean and exploiting the current best. Although these parameters do not explicitly optimize the tradeoff between confidence interval size and reward, it is often the case that more exploratory parameter settings will do a better job of minimizing confidence interval size. But it is not immediately obvious how to set these parameters for any particular tradeoff - what ϵ should we choose if we want to weight confidence interval size and reward equally? In contrast, UCB-EXPLORE allows us to explicitly set this trade off and optimize for it more directly. We will compare how UCB-EXPLORE trades off reward and scientific knowledge compared to MAB algorithms ϵ-GREEDY, UCB1, and UCB1-TUNED, ignoring the fact that the UCB-EXPLORE parameter is given by the objective while the other parameters may be more difficult to choose.

ϵ-GREEDY is a simple and straightforward method for balancing the dual goal of learning about the arm means and also maximizing reward. It has the additional advantage of ϵ being easy to interpret: the proportion of players who will be devoted to exploring the non-optimal arms. UCB1 is also capable of trading off between learning and reward by scaling the bounds: making them very

Parameter	Values (right to left)
ϵ-greedy, ϵ	0.3, 0.03, 0.01, 0.001, 0.0001
UCB1, c	1.0, 0.2, 0.15, 0.1, 0.03, 0.01
UCB1-TUNED, c	1.0, 0.2, 0.15, 0.1, 0.03, 0.01
UCB1-EXPLORE, w	0, 0.01, 0.03, 0.06, 0.1, 0.3, 0.6, 1

Table 6.1: The parameter settings generating the tradeoff graphs in Figure 6.1. All UCB1-EXPLORE variants in the graph, which have different values of the scaling multiplier m, are generated from the same group of w.

large causes the algorithm to prefer exploration, while making them small causes the algorithm to focus on the highest empirical mean and prefer exploitation. UCB1-TUNED replaces the loose bounds of UCB1 with ones that depend on empirical variance of the arms, which usually works better in practice [13].

We generate a tradeoff curve between average reward per pull and the sum of the sizes of the 95% confidence intervals around each estimated arm mean, shown in Figure 6.1. Each point for each algorithm is the average reward and interval size for 1000 trials of 10,000 pulls, for the parameters shown in Table 6.1. Points that are up and to the right are better. We see both that UCB-EXPLORE tends to have superior performance, especially when one does not care entirely about reward, and also that ϵ-GREEDY appears much worse than both strategies once we scale the bounds calculated by UCB1. In addition, the different UCB-EXPLORE curves are generated by different values of m - in our problem, it appears that the choice of m has little impact within a wide range, with perhaps the exception of $m = 1.01$.

To gain some intuition about how UCB-EXPLORE behaves and how one should choose w, it is useful to examine the behavior of the scaling factor c that controls the size of the bounds. This is shown in Figure 6.2. As $w \to 0$, the algorithm cares less and less about generating reward, so we expect it to explore more: as expected, c increases. Furthermore, the gains from reward are constant over time, but the gains from reducing estimated confidence interval sizes shrink as we

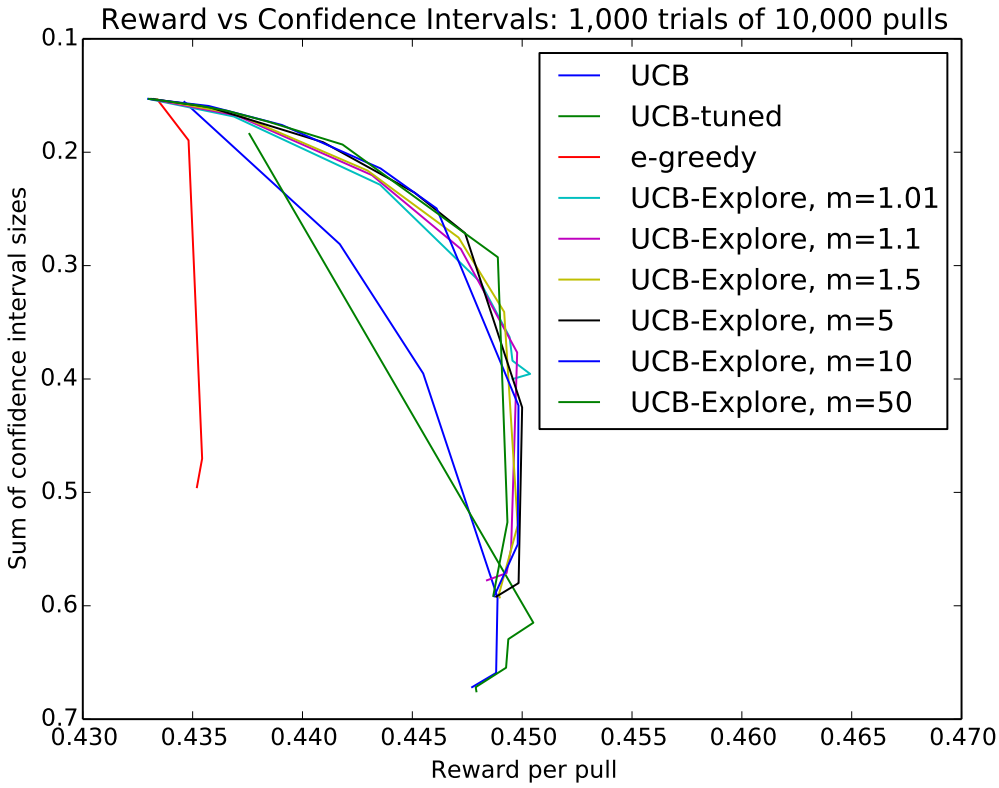


Figure 6.1: Reward vs confidence interval sizes. Up and to the right is ideal; we see that UCB-EXPLORE is typically better at generating high reward and learning the various arm means than other algorithms. ϵ-GREEDY performs poorly overall.

continue to sample. Thus as time passes we expect UCB-EXPLORE to focus more and more on reward as long as $w > 0$, and indeed this is the behavior we observe by the shrinking values of c. In the case of $w = 1.0$, c actually shrinks so fast that the algorithm exploits too quickly before it has a chance to explore and identify the best arms, explaining the small dips in performance of both UCB-EXPLORE and UCB1 when tuned to be very exploitative.

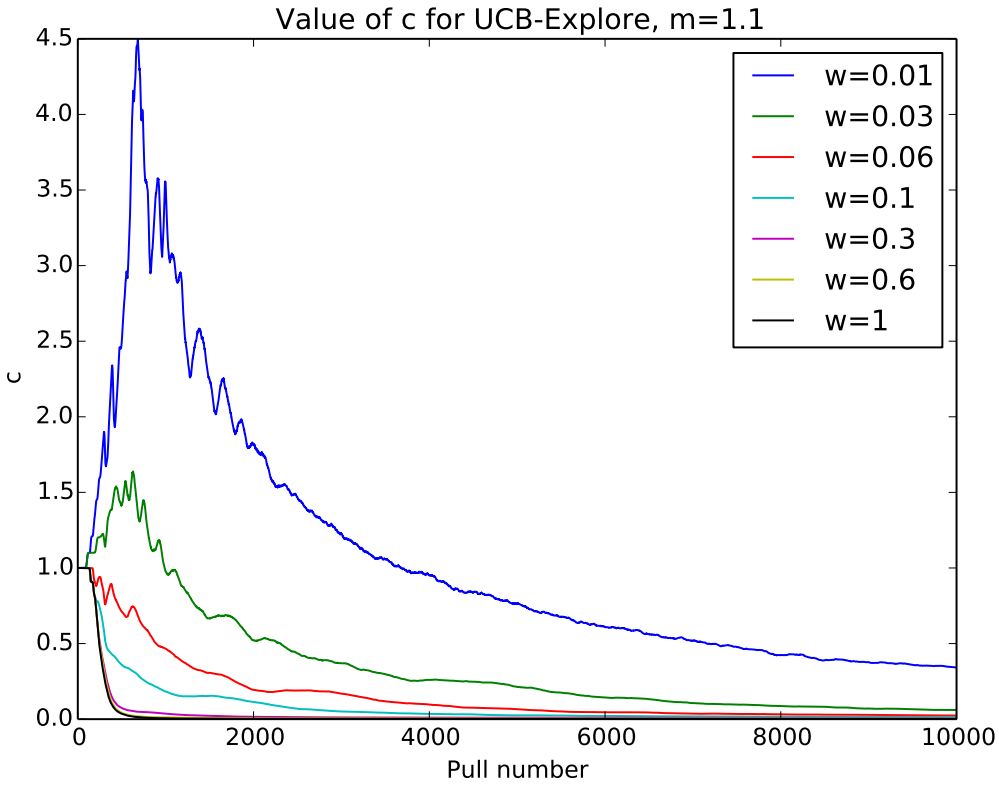


Figure 6.2: The value of the scale factor on the confidence bounds in UCB-EXPLORE as the algorithm pulls more arms. When $w = 0$, c increases quickly and does not stop, reflecting the fact that the algorithm cares only about exploration.

6.6 Validation

6.6.1 Interaction testing

Our algorithm is a method of allocating samples that respects the tradeoff between encouraging player learning and getting more accurate estimates of experimental condition means. Had this experiment been run online, a researcher could directly analyze the data gathered. However, we have been running our algorithm in a simulated environment with parameters drawn from real-world data, leaving us vulnerable to overfitting. To demonstrate how one might use data gathered from our algorithm, we will instead use these samples to generate hypotheses to validate on a separate

dataset: this is similar in principle to how a researcher might run multifactorial experiments in an online setting to find promising hypotheses to test in a more focused setting, as suggested by Stamper et al. [110].

UCBEXPLORE can be thought of as a biased method of drawing samples from different experimental conditions. One natural analysis would be to ask if one condition is significantly better than another. Many of our factors are specific implementation choices in our game, so such comparisons may not lead to very generalizable insights. Instead, we will attempt to identify likely interactions using the samples generated by our proposed method. In particular, we will search for two-way interactions which seem relatively large or small; we study interactions instead of main effects because we already examined main effects in Chapter 3. Then we will use the likelihood ratio test to determine if models learned with these interaction terms fit our validation data significantly better than models where these terms are set to zero. We stress that samples generated from ϵ-greedy and UCB1 could also be used in the same way and would likely lead to the same results, though in light of our simulation results either more samples would be required or more damage would be done to players in that case.

To do this, we run UCB-EXPLORE with m=1.1, w=0.001, and 100,000 pulls. We then calculate all main effects and two-way factor interactions as is done in the ANOVA test [121]. Our data is not normally distributed and the variances are unequal, violating ANOVA assumptions, but we can still consider which interaction effects seem relatively large or small. In our case, for each pair of factors, we can calculate the average magnitude of the interaction effects between all combinations of settings for those factors. We see that Target and Label have the largest average magnitude at 0.007, while Animation and Ticks have the smallest average magnitude at 0.0003. Thus, we suspect that Target and Label are much more likely to interact than Animation and Ticks.

To test these hypotheses, we will use a held-out validation dataset. This dataset consists of 9,675 players of Treefrog Treasure from June 20, 2013 to July 9, 2013. Unlike the dataset used to estimate parameters of our simulation in the previous section, we will consider only the first three number lines for each player: the first two are treated as the intervention, and the independently and randomly chosen third as the assessment of learning. For any given pair of factors, we can

	Target: Pie	Target: Symbolic
Label: Pie	0.431	0.410
Label: Symbolic	0.388	0.422

Table 6.2: Proportion of players in the validation set able to reach and answer the randomized test number line correctly on the first try. Our simulation results suggested that these parameters might interact; in fact, they interact very strongly.

attempt to fit a model with only main effects, or a model with main effects and two-way interaction effects. Since these models are nested, we can use the likelihood ratio test if the interaction model is a significantly better fit, given the increase in degrees of freedom.

For Target and Label, we have that $\chi^2(1, N = 9675) = 7.555, p < 0.006$. For Animation and Ticks, we have that $\chi^2(3, N = 9675) = 0.204, p = 0.977$. Thus, it is very likely that the effects of Target and Label representation on numberlines should not be considered separately, while we have no evidence that our Animation and Ticks hinting systems need to be modeled simultaneously.

6.6.2 Target/Label representation

In this chapter, our goal is to advocate the creation of algorithms which allow experimenters to trade off user learning and scientific knowledge, and the introduction of such an algorithm. The validation is meant to show that this approach generates samples that can lead to interesting hypotheses, so we do not claim them to be mature educational results. We will, however discuss them briefly.

The presence of significant interaction terms means that the factors involved should only be interpreted together. As a reminder, the Target factor refers to the representation of the fraction the player is asked to hit on the number line, while the Label factor refers to the representation of the fractions on the number line itself. The proportions of successful players for the different representation combinations can be seen in Table 6.2. The nature of the interaction is immediately apparent: players are more likely to reach and answer the next number line correctly if the target

and label have the same representation, and Pie chart targets and Symbolic labels are much worse than other conditions. Even when we ignore players who quit before reaching the second number line, these effects persist.

We do not know why this is the case. In our game, player ability to answer number line questions correctly is mostly a function of knowing where to hit, as the game informs the player where they will intersect the number line before they click to jump. In addition, the representation has no effect on game mechanics. Thus, the difference is most likely due to how players perceive the different fraction representations. One possibility is that number lines in classrooms are generally presented with symbolic labels only, so that the mix of familiar and new combinations of representations is particularly confusing to players. As with nearly all online experiments, we do not have access to players' thought processes, only their actions, so a more carefully designed study or a think-aloud in a classroom might be profitable. Regardless, our algorithm was able to generate samples that we could analyze for interesting factor interactions, suggesting that it is a viable method of adaptively randomizing experimental conditions.

6.7 Limitations

While our results seem promising, there are some limitations. It is extremely unlikely that UCB-EXPLORE is ideal for all bandit arm configurations; it seems to perform well on many-arm Bernoulli distributions with similar means, but some simulations suggest that it may not perform as favorably in very different cases. One example where our approach fails outright, as do ϵ-greedy and UCB1, is in the case that there are more arms than subjects - the algorithm expends all its subjects on the initialization phase when pulling each arm once. This problem occurs most obviously in the presence of continuous factors, in which case there are an infinite number of arms. It is less likely to occur in standard categorical experimental designs, though the limit can still be reached if researchers want to study something like the exponential space of all possible problem sequences.

Furthermore, while the reward portion of our dual objectives can be any measurable function of subject behavior, there may be other ways to define "scientific knowledge" or navigate the tradeoff between the two. For example, "scientific knowledge" might be the probability that the ordering

of arms is correct. Or instead of assigning some weighting between reward and knowledge, an experimenter might have constraints of the form "maximize reward subject to at least x knowledge." Some of these are relatively easy to incorporate into our framework. The example constraint might be handled by forcing the scale factor to stay at 1.0 until enough information has been collected, for instance. Other types of constraints or tradeoffs may require entirely different algorithms: knowing the number of subjects in advance, for example, leads to very different bandit algorithms than the infinite horizon variant we have presented here.

6.8 Future work

UCB-EXPLORE appears to outperform UCB1, UCB1-TUNED, and ϵ-greedy in our problem for most tradeoffs between reward and knowledge. However, our modifications remove any theoretical performance guarantees. It may be possible to alter the algorithm in a principled way to maintain its good performance and guarantee logarithmic regret. Extensive simulation on other problems with more or fewer arms and different reward variances would also be useful to understand when it or another method of allocating samples is preferable. We would also like to test if this algorithm is robust to unusual continuous reward distributions.

In addition, there are other problem formulations that would be interesting to investigate. In many practical cases (such as delayed rewards), fully online learning is infeasible; for these, we could adapt Bayesian techniques such as probability matching [99], which do not depend on online performance. Also, in cases where there is a finite budget of users, the algorithm will need to be modified based on work in mortal bandits to exploit more as the experiment draws to a close [26]. Lastly, in reality the arm distributions may be nonstationary, which might require adaptation of work in dynamically changing bandits [42].

More generally, UCB-EXPLORE only makes sense in situations where we have enough users to get substantial information about each condition. If we have many factors, we may not be able to get information about each specific condition, but may still be able to determine the best settings of the most important factors. A similar problem arises when we want to explore sequences of interventions, in which case techniques from Monte Carlo Tree Search may be most applicable.

And if one of the factors is continuous, the algorithm will not be able to make progress: here, it could be useful to adapt work on bandits in general metric spaces [54], or modifying a function approximation scheme as in [68] to incorporate the reward-knowledge tradeoff.

6.9 Conclusion

The rise of online educational software with massive numbers of users promises to change the experimental paradigm in educational research. With access to so many users and individualized control over what educational experiences they receive, it is now possible to automatically run complicated, multi-factor experiments quickly and at relatively low cost. However, education is a high-stakes domain: in many situations we have the ability to cause harm by placing students in sub-optimal conditions. Because of this we want to automatically put less students into harmful conditions while simultaneously discovering which are harmful and which are beneficial.

In this chapter, we propose allowing researchers to explicitly weight subject welfare against the amount of generalizable knowledge gained from the experiment. We show how the problem of allocating subjects to experimental conditions can be thought of as a multi-armed bandit problem with a dual objective of gaining maximum reward and minimizing the sizes of 95% confidence intervals around the arm means. We propose a new algorithm, UCB-EXPLORE, which takes a user-specified weight on the relative value of reward and confidence interval size, and adaptively adjusts its optimistic bound estimates to explore or exploit more when it makes a mistake with regards to this weight function. We analyze the behavior of our algorithm and compare it to tuned versions of other common bandit algorithms in a 64-arm simulation with parameters drawn from real-world data, showing that our algorithm is able to interpolate between these two goals much more effectively than standard algorithms. We use the simulated results of running our algorithm to generate some hypotheses about factor interactions, and confirm these results on a separate validation dataset, showing that the generated samples are useful from a research perspective.

This and previous chapters have shown the value of algorithms capable of automatically running experiments with users of online software. However, there has been one major assumption: that for the research domain of interest, online software collecting data on this domain already ex-

ists. This is by no means assured. For example, users often play free online educational games for short periods of time, so it is difficult to use these data sources to study how students learn multi-part concepts because they take too long to teach. In that case, no amount of machine learning will help us overcome this fundamental difficulty. Thus, a critical part of creating experimentation engines is the design of the software to serve as a good data collection device. This is one of the primary motivations behind educational campaigns, the subject of the next chapter.

Chapter 7

LARGE-SCALE EDUCATIONAL CAMPAIGNS

7.1 Introduction

So far, I have assumed that the data sources are given. This can be a reasonable assumption for certain tasks; many psychology studies can be run directly on Mechanical Turk, as I did in Chapter 5. Yet in some cases, such data sources may not exist. For instance, if we wish to study learning over longer timescales with access to demographic data, free online educational games are unlikely to give us the type of data we want. Studying these questions requires the creation of new software that can give us access to data that we wish to capture: as one example, LabInTheWild uses the innate human love of comparison to other humans in order to collect demographic information while simultaneously generating visual preference and other behavioral data [94]. This chapter is a published case study [70] of a new method of delivering educational technology to students: the educational campaign.

As quick background, recent years have seen a surge of interest in educational technology as a means of delivering scalable, adaptive content to students, especially those who have not historically had access to high-quality resources [45]. Massive open online courses (MOOCs) have been especially popular, but researchers have developed many other technologies that also qualify: intelligent tutoring systems [9, 34, 113], adaptive curriculum content [83, 102, 116], and educational games [79, 75] are all examples of educational technologies meant to both scale and improve student engagement and/or learning outcomes. The increasing availability of high-speed Internet has only accelerated this trend.

To encourage teachers and students to adopt and use educational systems, technologists must develop effective methods for packaging and delivering their content. In this chapter, I study a new method of delivering educational content at scale: the state- or country-wide educational

campaign. A campaign is a focused, widespread event where students from many schools engage in an educational activity over a short timespan. For example, in the Algebra Challenges I describe in this chapter, students in grades K-12 were challenged to solve as many algebraic equations as possible in the educational game DragonBox Adaptive over a week-long period. And while I focus primarily on educational technology, similar methods can be used by others to distribute other types of technology benefiting from concentrated collaborative or simultaneous use.

Campaigns are fundamentally different from typical methods used to deliver scalable educational technology. The simplest way to provide access to educational technology is by making it freely available online and allowing interested teachers to incorporate it into their classrooms, an approach taken by intelligent tutoring systems [34] or websites such as Khan Academy[1]. Another approach is to encourage participation by directly replacing the traditional classroom with an educational technology: MOOCs, for example, have attracted massive numbers of learners from diverse backgrounds with this approach [22].

Unlike these other distribution mechanisms, campaigns explicitly use media and publicity to encourage the participation of schools and classrooms that might not otherwise seek out innovative educational technologies. They also typically involve substantial social interaction in the forms of both intra-classroom cooperation and inter-classroom competition. These factors make campaigns a unique method of distributing educational technologies, and have distinct advantages and disadvantages compared to more common distribution mechanisms.

There have been few large-scale campaigns involving educational technology; as a result, very little is known about their properties. In this chapter, I explore data collected during three Algebra Challenges we conducted in Washington state, Norway, and Minnesota state. I describe the iteratations on the designs of competitive incentives and adaptation to player ability between each of the three Challenges, and analyze the resulting student outcomes. Based on analysis of the data from these Challenges, I provide a set of design recommendations for researchers and practitioners considering using similar campaigns as a method of delivering educational technology to many stu-

[1] http://www.khanacademy.org/

dents. In particular, I focus on how student behavior in an educational game during our campaigns differs from behavior in games released to free online websites, how the incentive structures can improve student engagement but cause undesirable "gaming" behavior, the usefulness of inserting "pretest" levels to allow students to skip content they may already know, and advantages and disadvantages of using campaigns as an experimental platform to conduct educational research.

7.2 Large-Scale Campaigns

In this work, we present a case study of three large-scale educational campaigns that we conducted on the topic of algebra. We define *campaigns* as events designed to encourage a large number of students to use an educational technology over a short period of time. Campaigns are designed to run at a state-wide or country-wide scale, attracting a large, diverse population of students. They are promoted through media and press releases, and possibly supported by visible figures such as politicians or industry leaders, as a way to recruit participants. Due to their limited timeframes, campaigns are designed to target a specific educational topic.

These types of educational campaigns have multiple goals. One goal is to generate awareness about both the educational technology being promoted and the organization that runs the event. Another goal is to motivate students and create excitement around the targeted educational topic. Educational campaigns are also designed to teach students during the event, and may have defined learning goals such as helping students achieve mastery on a certain concept or skill. Finally, large-scale campaigns can generate extensive data showing how a diverse population of students learns a topic or uses an educational technology. As a result, campaigns can be used as an opportunity to conduct controlled experiments on student learning.

The concept of using a short-term large-scale campaign for a targeted purpose is not new. A similar format has been used extensively to raise awareness about wide variety of topics, from eating disorders[2] to bullying[3]. Awareness campaigns typically focus on information dissemination, with the goals of educating the public about a particular topic, encouraging behavior change, and

[2]http://www.http://nedawareness.org

[3]http://www.bullyingawarenessweek.org/

raising money to support related research. One of the most well-known awareness campaigns is Breast Cancer Awareness Month[4], which was founded in 1985. Primarily focusing on awareness and fundraising, Breast Cancer Awareness Month includes walks for the cure, fundraising events, and extensive publicity and media events. As one example, the National Football League runs "A Crucial Catch Day" to promote breast cancer awareness by having players, coaches, and referees wear pink game apparel[5]. Studies show that Breast Cancer Awareness month helped increase the number of breast cancer diagnoses [50], and a comparative study shows that the large scale and extensive publicity of Breast Cancer Awareness Month has made it more effective than the smaller-scale Bowel Cancer Awareness Month [86].

While health-related awareness campaigns are the most well-known campaigns, this model has been used to promote formal educational goals as well. Geography Awareness Week[6] uses the campaign model to encourage teachers to focus on geography-related lessons during one week in November. More recently, educational technology has begun to appear in campaigns. For example, approximately six months after our Washington Algebra Challenge, Code.org ran the *Hour of Code*[7] campaign and encouraged students across the United States to participate in an hour-long programming activity during Computer Science Education Week. Code.org partners also created 20 hours of interactive tutorial activities that teachers and students could choose from[8]. Unlike the Algebra Challenge campaigns we discuss in this work, the *Hour of Code* campaign focused on outreach instead of specific learning goals: on this measure they were highly successful, with 15 million students in 170 countries completing an hour of code during the one-week campaign[9].

Campaigns hold promise as a delivery vehicle for scalable educational content, but in this context are essentially unstudied. To the best of our knowledge, this work represents the first in-

---

[4]http://www.nationalbreastcancer.org/breast-cancer-awareness-month/

[5]http://www.nfl.com/pink

[6]http://www.geographyawarenessweek.org

[7]http://code.org/hourofcode

[8]http://codeorg.tumblr.com/post/73414288834/2013update

[9]http://codeorg.tumblr.com/post/70175643054/stats

depth analysis of a large-scale campaign centered around educational technology. In this chapter, we detail both the logistical and technical considerations of running this type of event, and discuss key findings from three campaigns of our own: the Washington, Norway, and Minnesota Algebra Challenges. We discuss lessons we learned through the process of organizing these campaigns, and present design recommendations for other researchers or organizations interested in conducting similar events.

7.3 Educational Technology Distribution Methods

Educational technology has been developed and studied for decades; however, the increasing availability of high-speed Internet has recently made it feasible for free educational systems to scale to massive numbers of students. There are a number of methods of providing access to educational technologies and encouraging students to use these types of resources. In this section, we describe existing methods of distributing content at scale, and discuss the advantages and disadvantages of each technique compared to the large-scale educational campaign method that we study in this work. It is important to note that we are more interested in the distribution mechanism than the content or technology delivered; the distinction has not always been made clear in previous research, but in general most forms of educational technologies could be delivered through most of these distribution methods.

7.3.1 Purchased Software

The traditional method of distributing educational technology is as downloadable software that can be installed and run on computers in either a school or at home by purchase. This is the method that has historically been used to provide access to intelligent tutoring systems (ITSs). These advanced systems emulate one-on-one human tutoring using a cognitive model of the knowledge acquisition process [9]. They are designed to help students as they work to master new problem-solving skills in a given domain. Most ITSs are intended to be used as a part of a course, and students are expected to have access to a textbook, an instructor, and possibly other resources [113]. ITSs

have been evaluated in the classroom with success; for example, students have shown significant learning gains when using ITSs as compared to paper-and-pencil activities in both algebra [55] and physics [114]. These systems were originally developed before the advent of the Internet, and were designed to be run as desktop applications.

Schools and school districts also buy other types of educational software, such as reading programs, math games, or typing programs. Unfortunately, the market for such software and other educational tools can be hostile, especially to small groups and entrepreneurs [39]. One of the primary reasons for this is that the effectiveness of popular software is often mixed, increasing suspicion among potential buyers: the popular algebra program *I Can Learn* was shown to raise test scores [15], while the popular reading program *Fast ForWord* did not lead to increases in broader language acquisition or reading skills [96]. Unfortunately, evaluating educational technology is expensive even for researchers, let alone educational companies, since they require developing relationships with decision makers at the school level, creating assessments, and organizing and analyzing the experiment [29]. Similarly, selling and deploying such software can be challenging due to market fragmentation, differences in procurement methods between school districts, diverse curricula, and slow purchase cycles [35]. These and other factors, such as the propensity for running small resource-consuming pilots when start-up companies need larger deployments to remain viable, make it difficult for new entrants to create and market educational software [16].

Campaigns serve a fundamentally different purpose from traditional methods of distributing software. They can avoid some of the problems plaguing the traditional educational software market, primarily due to their short duration and the fact that participation is free. This bypasses procurement problems that make selling software to idiosyncratic school districts difficult [35], and also involves a type of time-pressure and social incentive for schools to participate in an "exciting event" that is not usually present for purchased software. Campaigns should thus be considered a new tool for distributing educational technology, with different properties than existing methods. Intriguingly, their characteristics may make them useful for addressing some of the problems in current software markets: it may be possible to use them to run relatively inexpensive, large-scale studies or pilot tests to study their effectiveness or convince school districts to buy the underlying

technology. We will touch on their viability as experimental vehicles later in this chapter, but much research remains to be done.

7.3.2 Websites and Apps

Nowadays, one of the most common methods of releasing educational technology is through freely accessible webpages. Arguably the most popular of these websites is Khan Academy, which follows this model to deliver lectures and scaffolded exercises to students. We are not aware of any controlled studies of Khan Academy's educational effectiveness, though preliminary research by SRI International suggests that its use is associated with increased student test scores [78]. The same study mentions that though Khan Academy is primarily used on an individual basis, its use as a supplemental resource in classrooms is increasing: the authors discovered that teachers use a variety of different methods to integrate Khan Academy materials within their classrooms.

Educational video games, which have experienced a surge of interest in recent years, are another type of technology that is typically delivered through free websites or app stores. Their popularity is in part due to their ability to engage players and motivate them to perform complex, time-consuming tasks [43], and in part due to the observation that learning is an essential part of gameplay [43, 44]. In contrast, traditional education has been criticized for only successfully engaging a small portion of students [106]; as a result, there is a growing interest in leveraging games to address the problem of student motivation in educational environments [43, 79, 75].

Campaigns tend to be more top-down events in which schools and teachers encourage students to participate for a short period of time on a focused topic. Thus, like content available freely or cheaply online, they are likely to be used in a supplemental fashion. Unlike online content, however, campaigns are also designed to increase student motivation and excitement through social pressure and collaborative or competitive rewards for participation. That is, they "push" content to students more, compared to the usual "pull" model of a student choosing to visit a website or download an app to view content.

7.3.3 Structured Online Courses

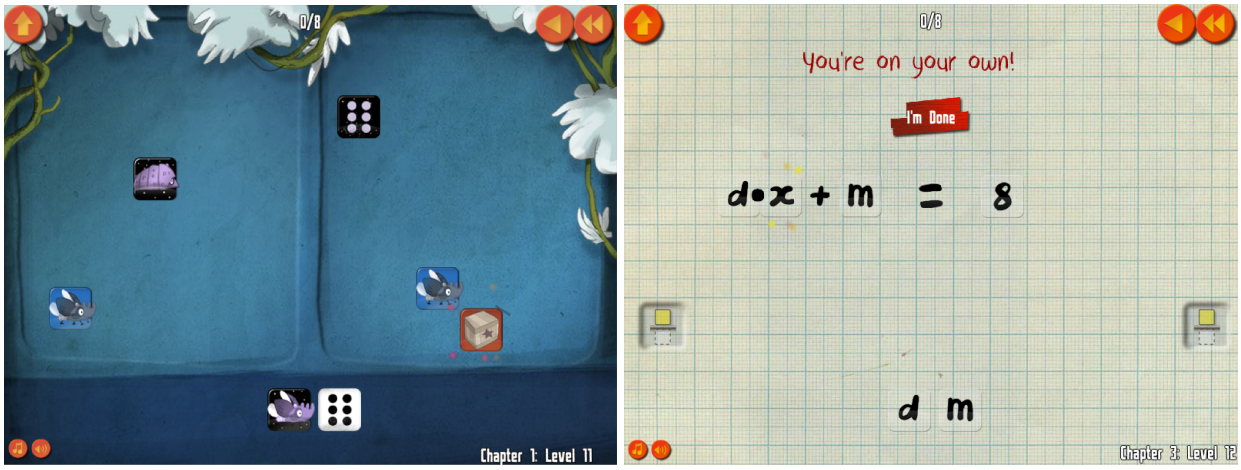
Over the past few years, Massively Open Online Courses (MOOCs) have become the central focus of discussions about learning at scale. These courses are designed to replace traditional classrooms with massive online classrooms, hopefully enabling the participation of diverse groups of students who have not previously had access to high-quality post-secondary education [22]. Hundreds of thousands of students have earned certificates through MOOCs in diverse topics such as music, systems biology, and computer science [65]. In contrast, campaigns are much more limited in scope, and are not designed to replace traditional classes in any sense. Thus they do not have lecturers or staff, as MOOCs do, and do not necessarily need to assign grades or credit for work completed (though they can do so if the campaign is run as a competition). More fundamentally, campaigns are by their nature exciting community-wide events, and one of their primary goals is to generate student interest in a particular topic. MOOCs are less concerned with this sort of outreach, and more about teaching already-interested students deep knowledge about a topic.

7.4 The Algebra Challenges

This chapter is primarily a presentation of a case study about three campaigns run by our research group: the Washington, Norway, and Minnesota Algebra Challenges. These campaigns were run sequentially, in order to give us a chance to make changes and improve outcomes from one to the next. I first describe the educational game delivered to students through the campaign, DragonBox Adaptive, then give some basic information about the campaigns themselves and how their structures differed. Later, I will take a closer look at the data we gathered during the Algebra Challenges and offer design advice to other researchers considering such campaigns, either as ways to deliver educational content or as experimental platforms for studying educational interventions.

7.4.1 DragonBox Adaptive

For the educational content delivered through our Challenges, we used a game called DragonBox Adaptive [25], evolved from the original game DragonBox [117]. The game is designed to teach



(a) Simple (b) Mastery

Figure 7.1: Screenshots of DragonBox Adaptive, ©(2013) UW Center for Game Science / We-WantToKnow AS [117]. Figure 7.1(a) shows an early level of the game with the equation `a-b=-6+a+x`. The DragonBox, on the bottom right, must be isolated on one side to pass the level. The game increases in complexity and gradually begins to look more like standard algebra, as can be seen in one of the mastery test levels in Figure 7.1(b).

algebra for all students in the K-12 spectrum, and can be seen in Figure 7.1. Each level represents an algebra equation to solve, with left and right sides filled with cards that represent numbers and variables. The key card is the Dragon Box, which must be isolated on one side for the player to win (much like X in an algebraic equation). Players can perform algebraic operations by simplifying cards in the equation area (e.g., '`-a+a`' → '`0`'), eliminating identities (e.g. '`0`' → ' '), or using cards in the deck at the bottom of the screen to add, divide, or multiply both sides.

An important concept in the version of DragonBox Adaptive we used in the Algebra Challenges is *mastery*, one of our outcome measures. Students demonstrated mastery by successfully answering a series of three test levels designed to look very similar to standard paper-and-pencil algebra problems. An example of one of these tests can be seen in Figure 7.1(b). Test levels offered no hints, did not force students to keep both sides balanced, and allowed students to submit answers at any time. Furthermore, students had to submit correct answers to three tests in a row under a certain time limit in order to achieve mastery. We will see in Section 7.5.4 that students

were largely unable to pass these tests when randomly given out early in the game, but had much better success rates when given sufficient practice; thus, at the very least, students in the Challenges improved at solving these mastery test levels.

As the name implies, DragonBox Adaptive differs from its parent game, DragonBox. The primary difference is adaptivity: depending on how they performed on embedded assessments, students were given different sets of dynamically-generated problems for additional practice. These assessments measured overall success, time to completion, and number of required actions to make these decisions. Furthermore, there were two types of assessment levels: those which looked and functioned like normal levels with in-game scaffolding, and the test levels mentioned above that looked like algebra problems and had no scaffolding.

All three Challenges gave mastery test levels at a particular point in the progression (about 50 levels in), and if the player failed the test they were sent backwards in the progression to play a few more levels before trying the test again. This additional practice usually amounted to about three levels, though occasionally could be more if a student performed very poorly. Norway and Minnesota were enhanced with a second set of even more difficult mastery tests; again, if students failed the tests we generated new practice problems for them.

All three Challenges also made use of the assessments that looked like regular game levels. In Washington, these levels were restricted to later on in the progression just before the mastery tests, so that all their extra practice tended to be extra levels involving relatively difficult concepts. In Norway and Minnesota we added more of these assessments for even basic concepts, so that students who needed additional practice early had the opportunity to play extra levels. In all cases, we used a "generative" adaptivity where new problems were generated for each student depending on which assessment they failed; this is in contrast with the common tutoring system strategy of drawing from an existing and finite pool of problems.

7.4.2 Algebra Challenge overview

We ran three Challenges in sequence: Washington, Norway, and Minnesota. Washington and Norway were designed to run for one week, and while we set a long cutoff date for Minnesota,

Location	# Users	Time (sec.)	Mastery (all/1.5h)	Partner	Start	End
Washington	4200	4199	52%/96%	Technology Alliance	2013-06-03	2013-06-08
Norway	36100	7375	65%/96%	We Want to Know	2014-01-12	2014-01-17
Minnesota	6900	3698	48%/96%	Technology and Information Education Services	2014-02-03	2014-07-01

Table 7.1: Some basic statistics about the three campaigns we ran. Note that the bulk of Minnesota players played in the first week. Time is the average length of time players were actively playing. The two mastery rates are the overall rate, and rate among students who actively played at least 1.5 hours (the length of time we asked teachers to devote). More in-depth discussion and statistical analysis will be given in Section 7.5.

the bulk of the activity occurred during the first week there, as well. To recruit teacher and student participants, we partnered with different organizations with existing connections to schools: we did not pay them, and indeed in some cases they paid us. In general, we found that it was quite easy to find teacher organizations and educational technology promotion groups eager to run these types of campaigns. These groups contacted all interested teachers through mailing lists, bulletins, social media, or through word-of-mouth. We did not place any grade restrictions on signups, so we received a wide spread of students. Future campaigns with specific pedagogical goals may wish to target content towards a particular grade; we do not know how easy or difficult this would be, but imagine that schools would be less eager to participate if only a fraction of students are allowed to participate. We also had some media coverage and promotion by political figures: for example, we received a letter of support from Washington Governor Jay Inslee, and the Norwegian Digital Learning Arena tweeted a video of the Norwegian Prime Minister, Erna Solberg, playing DragonBox Adaptive as part of the Challenge. This support may have lent legitimacy to our efforts and increased participation, though the strength of this effect is difficult to measure.

Before examining more specific information about the Challenges, what makes campaigns suc-

Figure 7.2: The Algebra Challenges included a great deal of supporting infrastructure, including the "Challenge Dashboard." This website functioned as a leaderboard that teachers and administrators could use to track mastery, number of equations solved, and play time at the class, school, or district level. Furthermore, teachers were able to track student progress within their own classroom.

cessful overall? We are not equipped to answer this question with certainty, but can speculate. Certainly the Algebra Challenges had a great deal of supporting infrastructure, such as collaborative or competitive goals and websites to track student or class progress, registration and organization pages, and server architecture. The Challenge Dashboard seen in Figure 7.2 seemed quite popular among teachers, for instance. Yet at the end of the day the primary consideration for any campaign is how many schools, teachers, and ultimately students choose to participate.

Our sense from communications with politicians, technology advocacy groups, administrators, and teachers, is that everyone benefited in different ways. Politicians could generate positive press with their association with an innovative, large-scale educational effort. Advocacy groups could both tout the benefits of educational technology and increase their own influence by adding a major campaign to their list of accomplishments. Teachers and schools seemed excited to generate enthusiasm among students by being part of a "big" event involving an educational game. For everyone, the Challenges inspired a sense of being a part of a cutting-edge, important project run by a prestigious research institution. In addition, even though the content was experimental and not guaranteed to deliver learning gains, the limited time commitment and use of a modified popular algebra game, which students would enjoy regardless, meant there were few potential downsides. If true, this argues in favor of both short-duration and game-based campaigns when content is relatively untested.

Another important question to ask is the following: how easy would it be for other groups to run such campaigns? This is very difficult for us to answer. From our perspective, running the Challenges required some amount of effort and coordination, but was not tremendously difficult. The most important part, teacher recruitment, was mostly handled by other parties, such as the press generated by politicians or mailing lists of the technology groups we partnered with. At the same time, our group is relatively well-known for other projects, has well-connected funders, and was using an advanced version of a popular existing math game. If none of these factors had been in play, we may not have been able to convince these groups that the effort of running such events would have been worthwhile. We doubt, for example, that a research team consisting of five people with no existing infrastructure, connections, or credentials would be able to run a successful

campaign. Speculation aside, the best way to answer this question is for other research groups to attempt to run their own campaigns and report their experiences.

Now that we have covered our general observations about running campaigns, some basic statistics about the Challenges themselves can be seen in Table 7.1. As a reminder, "Mastery" refers to the ability of players to pass three difficult test levels in a row, under a certain time limit. Since many of these players failed to reach mastery because they did not play long enough, we also include mastery rates for students who played at least the 1.5 hours we requested of students. These are quite high, but we note that this is no guarantee that if students who stopped were compelled to play longer they would reach similar rates of mastery. Another important point is that higher play times are not necessarily better: they could be indicative of more engagement, but could also reflect an excess of time spent on easy problems. For example, the average time to mastery in the Washington, Norway, and Minnesota Challenges was 2626, 2820, and 2515 seconds, respectively: thus, the Minnesota Challenge may have been more efficient in some sense. That being said, the data in the table suggest that Norway was a clear outlier both in terms of scope and player behavior. There are several possible explanations for the difference in time played and mastery rate in Norway as compared to Washington and Minnesota. Since this is a case study and many variables changed each campaign, we cannot definitively answer this question, though we can still examine individual hypotheses.

One possible explanation is that Norway students tended to be older, as can be seen in Figure 7.3. We can test this hypothesis by running an ANCOVA on Time Played (in seconds) with independent variable Location and covariate Grade. There is a significant difference between the three Algebra Challenges ($F(2, 47197) = 250.154, p < .000$), while the association between Time Played and Grade is not significant ($F(1, 47197) = 1.361, p = .243$), suggesting that the population age is not the sole driver of the increase. Only 2.8% of the variance is accounted for by Location, controlling for Grade ($R^2 = .028$), so other unmeasured factors may be responsible. To complete the analysis, we also find that Norway students (Mean=7375 seconds) play statistically significantly longer than Washington (Mean=4199 seconds) and Minnesota students (M=3698 seconds) at the .01 level ($p < .000$), while the difference in Time Played between Washington and

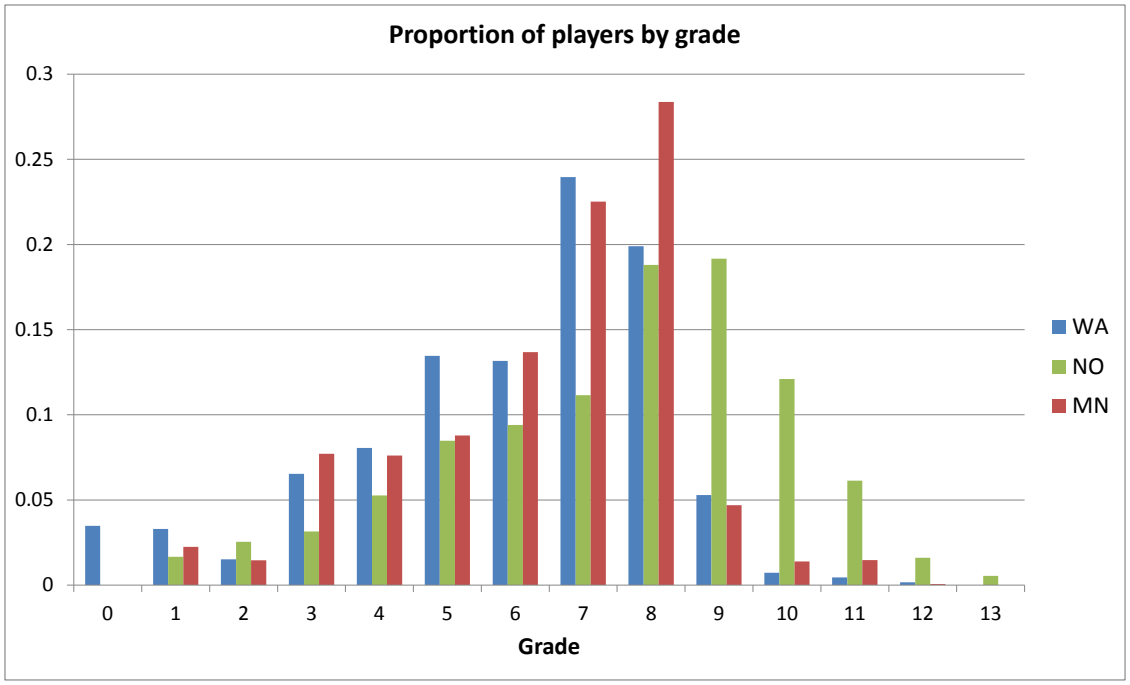


Figure 7.3: The average grade levels for Washington, Norway, and Minnesota are 5.9, 7.2, and 6.4, respectively. An ANOVA shows these differences are significant ($F(2, 47198) = 1335.626, p < .000$). The mean Norway grade is higher than Minnesota's ($p < .000$) and Washington's ($p < .000$), and the mean Minnesota grade is higher than Washington's ($p < .000$). This is a potential confound to keep in mind.

Minnesota is not significant at the .05 level ($p = .058$).

Another intriguing possibility has to do with how we encouraged participation. The incentive scheme in Norway was different than the one used in Washington, and potentially more motivating; Minnesota had no incentive scheme at all. Could the change in incentive structure explain why students in Norway played longer than those in Washington and Minnesota? We will analyze the data in light of this hypothesis in Section 7.5.2.

7.5 Lessons

Campaigns seem like a promising way to deliver certain types of educational technology to many students. To help other researchers or organizations who wish to run similar events, we will share

some of our observations from the Algebra Challenges. We will discuss potential differences between student data generated by campaigns compared to free game websites, the effects of different incentives on student behavior, what factors we observed were important to account for if attempting to run a randomized experiment in a campaign, and the results of one randomized study we ran in the Minnesota Algebra Challenge when attempting to fastforward students through content using early mastery tests.

7.5.1 Comparison to Online Educational Game Releases

To better understand how campaigns function as a mechanism for distributing educational technology and encouraging participation from a broad population of students, we were interested in comparing the DragonBox data collected during our campaigns to data collected from educational games released through online game portals. This type of analysis allows us to highlight features of campaigns that make them different from standard distribution methods, and define the properties that educational systems should have to be appropriate for distribution through a large-scale campaign.

DragonBox Adaptive has not been released on any free online game portals and the maker of the original game DragonBox does not track engagement statistics, so we are unable to directly compare student behavior in this game across distribution methods. However, many other educational games have been released online, so we can compare high-level features of the data collected with these games to data collected with DragonBox Adaptive through our campaigns. This will give us an initial understanding of the differences between free game portals and campaigns as a distribution method for educational games.

In this analysis, we compare the Algebra Challenge data to data collected with games released to the popular educational website BrainPOP[10]. BrainPOP provides an educational games portal designed for use in the classroom, where games can be freely accessed at any time. We look at three popular BrainPOP games: Refraction[11], which teaches fraction concepts through splitting lasers,

[10] http://www.brainpop.com/

[11] http://www.brainpop.com/games/refraction/

and Treefrog Treasure[12] and Battleship Numberline[13], which both teach number line concepts. All three games were developed for research purposes, and student behavior in each game has been studied extensively [74, 80].

Students who participated in the Algebra Challenge played DragonBox for much longer periods of time on average than students played the other three educational games on BrainPOP. In the Washington, Norway, and Minnesota Challenges, students played for a median time of 54 minutes, 82 minutes, and 47 minutes respectively. In comparison, published literature suggests students on BrainPOP play Refraction for a median of 3 minutes [80], Treefrog Treasure for a median of 8 minutes [80], and Battleship Numberline for a median of 2-3 minutes [74] (reported numbers in this paper are reaction times, so true time played is higher).

We caution that these numbers cannot be directly compared, given that the underlying games are different. Furthermore, it is not easy to track students on free game websites across different sessions without login information, so most research on free online games measures time played during the first session only. We have no such problem tracking across sessions in our Challenges and so can measure complete time played. With these caveats in mind, there are good reasons to believe that students may play more seriously in campaigns compared to online games. Teachers may plan for students to play the educational game for a set period of time during class when participating in a campaign, for example. There may also be effects from collaborative goals or rewards for campaign winners, though we note that students still played quite long in the Minnesota Challenge, in which there were no explicit incentives. Whether or not campaigns truly promote greater engagement than online distribution methods, and exactly how students use the technology differently depending on the distribution method, are questions we leave for future research.

Another difference between the data collected during the Algebra Challenges and through online game portals is the type of demographic data collected. During the Challenges, we were able to collect gender, grade, and school information about students, given by teachers when they registered their classes. This data is likely to be reliable because it is entered by teachers rather than stu-

[12] http://www.brainpop.com/games/treefrogtreasure/

[13] http://www.brainpop.com/games/battleshipnumberline/

dents, in stark contrast to the data collected through online educational game portals. O'Rourke et al. note that it is challenging to collect demographic data from students who play games on BrainPOP [81], and demographic information is not presented in any of the work studying Refraction, Treefrog Treasure, or Battleship Numberline [74, 80, 7]. We have tried collecting demographic information in educational games released online through embedded, skippable questionnaires, but have found that these surveys cause the majority of players to quit; more complex methods, such as giving players the opportunity to enter demographic information to compare their responses to similar players' responses [93], may be required.

These comparisons show that it is easier to collect "complete" data through campaigns than free online games. That being said, campaigns are also more expensive to conduct. By our estimates, each campaign cost around $45,000 to conduct, while releasing a game to an online portal is free. However beyond just the cost, there are other features of campaigns that educational technologists should consider. Since participants play for a very long time on average, the content of the educational technology being distributed must be extensive enough to support multiple hours of play. Depending on the stated goals of the campaign, near-infinite content may be desirable. In the Norway Challenge, for example, classes were rewarded for completing the largest number of equations: this meant that DragonBox Adaptive needed to support an infinite amount of play time to fairly support this competition. We accomplished this by procedurally generating levels from templates, to ensure that students could continue playing for as long as they wanted; indeed, one student in Norway played almost 23,000 levels.

Even without such outliers, the fastest 5% of players in the Norway Challenge achieved mastery in 64 levels, while the slowest 5% of players achieved mastery in 224 levels. If such content generation or adaptivity is not possible, organizers of campaigns may wish to promote completion of the content instead of duration or volume of interaction; however, our experience suggests that having generative adaptivity may be useful to accommodate the wide spread of student abilities.

Location	Collaborative	Competitive	Prizes
Washington	250,000 equations, Class mastery	Class mastery	One tablet per grade per class size
Norway	400,000 equations, Class mastery	Equations solved	One tablet per student of the winning class
Minnesota	250,000 equations, Class mastery	None	None

Table 7.2: Incentive structures in the Algebra Challenges. The collaborative goal was a target number of equations to be solved across all students participating in the campaign, and progress towards the goal was listed on the campaign website. The competitive goal was a reward given to classes that had the highest mastery rates (in Washington) or the most equations solved (in Norway).

7.5.2 Incentives

Educational technology is only valuable if students feel motivated to use it. Incentives for using a given system can vary widely. Students may have a desire to learn, may be seeking accreditation in the form of a badge or certificate, may enjoy the activity, or may be encouraged or pressured to participate by teachers, parents, or peers. Luckily, campaigns are uniquely suited to take advantage of many of these incentives. In our three Algebra Challenges, we used a mix of incentives not frequently found in other methods of delivering educational content at scale. In each Challenge, we set a collaborative goal of solving a certain number of equations across all participants. We also experimented with different types of competitive goals in the Washington and Norway Challenges as a way to leverage peer excitement and peer pressure at the classroom level to increase student engagement.

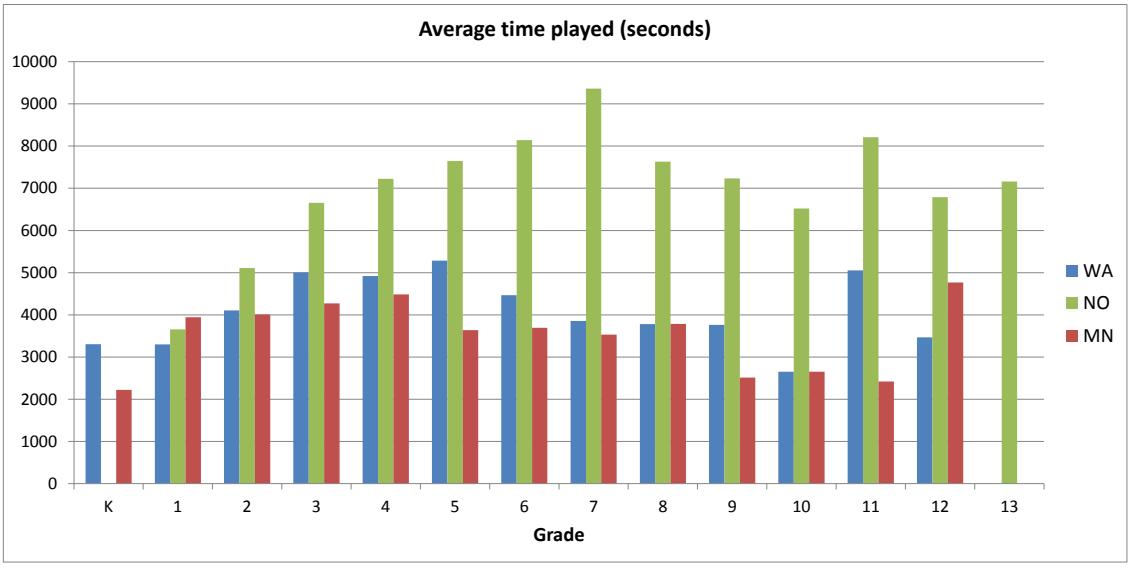
The incentives we used for each campaign can be seen in Table 7.2. There are a number of differences between the competitive incentives used in Washington and Norway. In Washington, we incentivized mastery rate, or students' ability to reach a certain point in the game. We chose to reward mastery because the educational goal of the Challenge was to help students master algebra

concepts. We awarded one tablet to the classes that had the highest overall mastery rate at each grade level. In addition, some classes had many more students participate than others, and it seemed unfair to compare large classes to small ones. We therefore awarded prizes to one class of each size (extra small, small, medium, or large) at each grade level, for a total of four winning classes in each of the grades K through 12.

In Norway, the sponsors requested we change the incentive structure to instead reward the total number of equations solved by each class. The goal of this modification was to reward behaviors that students could see and directly control. It is very clear when a student solves an equation, because this is equivalent to completing a level in the game; mastery, on the other hand, is a much more nebulous concept that was more difficult for students to grasp and work towards. This idea is similar to one proposed by Fryer, who found that giving students financial incentives for reading books significantly improved outcomes, while providing the same incentives for either increasing classroom grades of state test scores produced no significant improvement [40]. We therefore hypothesized that a similarly concrete incentive structure might be more effective in our campaigns. We further decided to reward only the single class that solved the most equations, and we gave each student in the winning class a tablet; this was designed to increase the personal incentive for students to play.

As seen previously, the descriptive statistics in Table 7.1 and the ANCOVA results from Section 7.4.2 immediately suggest two possible conclusions. First, we found no significant change in how long students played between the Washington and Minnesota Algebra Challenges even after controlling for grade (Figure 7.4). Given that we did not advertise any incentives in Minnesota at all, this calls into question the usefulness of the Washington incentive model. Second, students in Norway put much more time into the Challenge: they played for a much longer period of time on average, even when controlling for grade. This suggests that the more personalized and direct incentive structure may have been effective at influencing student behavior.

We caution that the higher time played and mastery rate during the Norway Challenge could be the result of any number of factors. Perhaps the schools involved in the campaign were richer, or maybe DragonBox is better-known in Norway. Unfortunately, it is difficult to study the impact



(a) Time

(b) Mastery

Figure 7.4: Differences in player outcomes persist even when accounting for the fact that Norway students were older.

Location	Egregious skipping	Any skipping
Washington	0 students (0%)	7 students (0.2%)
Norway	2111 students (5.8%)	10133 students (28%)
Minnesota	0 students (0%)	0 students (0%)

Table 7.3: Occurrence of intentional failure by "skipping" a problem (submitting an answer without making any moves) This is an example of undesirable "gaming the system" behavior, potentially a side effect of the Norway incentive scheme rewarding total levels completed and giving tablets to each student of the winning class.

of different incentive schemes, because it is challenging to conduct randomized experiments in campaigns. We cannot run two simultaneous campaigns in the same schools and same location with different incentive schemes, for example. Thus all analysis that follows should be taken primarily as suggestions for future research directions, possibly in a more controlled environment.

Caveats aside, the time played and mastery rate in the Norway Challenge are encouraging, and suggest that our equation-based competitive incentive structure may have been effective. For example, in Norway approximately 44% of levels played were played outside of regular school hours; in Washington, this figure was a still-respectable but lower 21%. However, the effects were not all positive. We also uncovered evidence of undesirable behavior while performing statistics to find the winning classroom. In the Washington and Minnesota Challenges, the maximum number of levels played by any single student were 1,634 and 3,000, respectively. In Norway, however, 90 players played more than 3,000 levels, and the maximum number of levels played was 22,960. This far exceeded the amount of practice we desired from students, and strongly affected the total class equation counts used to determine the winner.

When we examined the data more closely, we discovered that some players were "gaming the system." There were two different types of gaming behavior that we observed. The more mild form involved players submitting incorrect answers to game levels. This behavior would cause the game to restart the level with a potentially easier format, allowing students to boost their numbers of equations solved by completing many easy levels. The more egregious form of gaming behavior

involved players deliberately failing a mastery test, causing them to be sent back to easier levels that could be solved very quickly before reaching (and failing) another mastery test. In other words, these players used the ability to replay easier levels to artificially inflate the number of equations solved.

A natural question is whether this type of gaming behavior occurred at the same rate in all three Challenges, or whether the rate was elevated during the Norway Challenge. A reasonable proxy for intentionally giving up on a regular game level (mild gaming) or a mastery test (egregious gaming), is to measure how often participants submit an answer without having made a single move. Table 7.3 shows the number and percentage of players who engaged in this behavior at least once for arbitrary levels or mastery levels. Given that almost no intentional failure occurred in either Washington or Minnesota, it seems highly likely that this undesirable gaming behavior occurred as a direct result of the different incentive structure used in the Norway Challenge.

While these results are suggestive, more research is needed to understand how best to incentivize participation in these types of campaigns. We were not able to determine from this analysis whether rewarding individual over joint achievement was more important, or rewarding total levels completed rather than mastery rate was more important. Furthermore, while a sizable portion of students in the Norway Challenge engaged in gaming behavior, the majority did not. It is possible that there is a way to retain the gains in time played and mastery for the majority of the population, while reducing opportunities to game the system. Can the game rules be structured to continue to offer additional support for struggling players while punishing intentional player failure? Or can we detect gaming behavior when it happens, as has been tried in Intelligent Tutoring System [14]? Regardless of the answers to these questions, the results from our analysis suggest that some types of incentives can affect student engagement and participation in campaigns; thus, these incentives must be designed carefully to encourage desirable behaviors.

7.5.3 Campaigns as an experimental platform

Educational software is increasingly being used to run experiments, as has been the case in the e-commerce and game industries for a long time [57]. Notably, several studies using data drawn from

educational games have been published in the past few years (ex. [74, 71]). Previous researchers have considered using online experiments, combined with more traditional sources, to construct new types of experiment frameworks [110]. Challenges could conceivably be used for this purpose as well, but have their own unique set of strengths and weaknesses relative to other content delivery mechanisms.

Clear strengths of campaigns as experimental platforms are as follows:

- Relatively high rate of data (36,000 players in Norway over 5 days).
- Enhanced persistence of students compared to games released on free educational websites.
- Access to some demographic information about students, entered by teachers (note that this mitigates a key drawback of internet data, which generally does not contain rich data [110]).
- Relative ease of some randomization and data collection (updating a webpage and consulting a database, respectively).
- Potential ability to run social or multiplayer interventions, since it is very likely that students will be participating as part of a group.

Clear weaknesses of campaigns as experimental platforms are as follows:

- When cast as competitions, unfairness and negative press can become problematic. It would have been nearly impossible to run any kind of substantial randomization in Norway, for example, as running a competition requires that every class be using the same game and level progression to fairly compare classes against each other. This is much less of an issue if the only stated goals are collaborative: this was the case in Minnesota, and as we will discuss in Section 7.5.4 we were able to run a randomized experiment with fastforward mastery test levels.

- Violation of independence of samples, since students may play together. Precisely measuring how often this occurs is difficult, but two statistics are indicative: in the Washington Challenge, 80% of levels were played during standard school hours (8:00-15:00), and 98.5% of levels were played within one minute of another student from the same class playing a level. This problem is particularly troublesome if different players in the same class are put into different experimental conditions, since students may notice that they are not performing the same task.

Other characteristics of campaigns are as follows:

- Each Challenge cost approximately $40,000 in salary and $4,000 in server costs, not including the development of the logging infrastructure and Dragonbox Adaptive.

- Our campaigns only run a few days: the upside is that there is less time for other factors to interfere (such as learning outside of school), but the downside is that it is difficult to measure retention over long timescales or adjust for any problems that occur in the middle of the campaign.

- It may be possible to interview individual students, since the participating schools and classes are known. We did not do so during our campaigns, as our primary focus was on running them smoothly and with maximum participation; we suspect interviews would have been possible, but coordinating permissions and timing with instructors might have been difficult. It will be important to investigate the ease of collection and usefulness of qualitative data in future campaigns.

To mitigate the effect of unfairness while retaining the highly motivating competitive incentives, one possible solution might be to run competitions at the school level instead of globally. Thus school A and school B might have their own distinct prizes to award, and could be safely assigned to different experimental conditions. For that matter, the same could be done at the teacher level, allowing randomization between teachers within the same school. This would be ideal, but

only if the school or teacher have little effect on student behavior so that students in different experimental conditions would be a priori similar. Likewise, we might also wonder whether other student characteristics, such as gender and socioeconomic status, seem to affect their behavior. To answer these questions, we will attempt to model the effects of school, teacher, grade, gender, and socioeconomic status in the Algebra Challenges. The results are useful in understanding how researchers should try to randomize students into conditions, and which covariates may be important to pay attention to in the process.

Hierarchical Linear Model

How much of an effect on total time played do school and teacher have in our dataset? To answer this question, we used Hierarchical Linear Modeling (HLM), a complex form of ordinary least squares (OLS) regression, which analyzes the variance in the dependent variable when predictors are at varying levels: e.g., students in a classroom share variance as a function of their common teacher and common classroom. While this data could be analyzed ignoring the nested structure of the data, for example using fixed parameter simple linear regression, this approach would be insufficient due to its disregard for the shared variance [122], aggregation bias and misestimated precision [91].

More precisely, a 3-level Hierarchical Linear Model was employed to analyze how much of an effect schools and teachers had on the amount of time played, in seconds, during each of the three Algebra Challenges. Student was Level-1 of the model, Teacher Level-2, and School Level-3. Our dependent variable for these models is Total Time Played, measured in seconds. Our independent variables are Grade, Free and Reduced Lunch (Washington and Minnesota only), and Gender (Norway only). Free and reduced Lunch is the percentage of students in a school that participate in the Free and Reduced Lunch program, a common socioeconomic status indicator in education research, and was collected from school reports posted online.

Due to the fact that the data available for each Campaign varies, our models for Washington and Minnesota are slightly different from the Norway model. For Washington and Minnesota we added Grade as a predictor in the Student Level and Free and Reduced Lunch as a predictor of the

School Level. Norway had Gender data for a portion of their students, so we added Gender as a predictor in the Student Level along with Grade; however, Free and Reduced Lunch data was not readily available for Norway, and so was not included as a predictor in the School level.

Results of the HLM analyses are provided in Table 7.4; the findings vary by location. In Washington, the effect of Grade on Time Played was not significant, while the effect of Free and Reduced Lunch was statistically significant at the .01 level ($t = -4.125, p < .000$). This means that for Washington, the Grade level does not impact the length of time a student plays the game, and the higher a school's free and reduced lunch participation rate, the less length of time students from that school play. In Minnesota, the effect of Grade does have a statistically significant impact at the .05 level ($t = -2.265, p < .024$), where higher grade levels play less time than lower ones, while the impact of Free and Reduced Lunch does not have a statistically significant impact. In Norway, while Grade is not significant, Gender is at .05 level ($t = -0.571, p = .568, t = 2.270, p = .023$, respectively), with female students playing longer than male students.

With regards to the influences of teachers and schools, the analyses showed that the proportion of variance within teacher is more than 33% for all locations (Washington 45.60%, Minnesota 42.33%, Norway 33.82%). As compared to variance within teacher, the proportion of variance among teacher within schools is lower for Minnesota (33.24%), but marginally higher in Norway (38.51%) and Washington (48.03%). In contrast, the proportion of variance among schools in Washington (6.38%) is much lower, while the proportion of variance among schools in Minnesota (24.44%) and Norway (27.68%) are still high. Thus teacher and school account for a great deal of variance and appear to impact students' Time Played, and potentially the school matters less for explaining student engagement in Washington.

Discussion

Regardless of location, we found that teacher and school were highly significant factors on how long students play. This means, for researchers who wish to run randomized experiments using campaigns, randomizing within schools or even within teachers is ideal. If the researcher wishes to use competitive incentives and the conditions are very different, then this may not be possible. Fur-

Fixed Effect	Coefficient	Standard Error	DF	t-ratio
WA				
Intercept	5402.42	539.13	61	10.021**
Grade	-134.57	112.68	2281	-1.194
Free/reduced lunch	-3629.95	880.06	61	-4.125**
MN				
Intercept	4006.83	791.89	53	5.060**
Grade	-70.40	31.08	3801	-2.265*
Free/reduced lunch	-1395.19	1742.95	53	-0.800
Norway				
Intercept	9163.89	541.60	287	16.920**
Grade	-105.84	185.50	13464	-0.571
Gender	444.11	195.63	13464	2.270*

Table 7.4: *$p < 0.05$, **$p < 0.01$. 3-level HLM results, with student nested in teacher, and teacher nested in school (intercepts and standard errors given in seconds). Highly significant intercepts suggest that the school and teacher impact how long students participated, which may be useful for future experimental designs using campaign data.

ther research is required to understand how characteristics of schools and teachers predict student behavior, so that conditions can be balanced as evenly as possible. Furthermore, the characteristics that are important may vary by location: Free and reduced lunch rates, for example, are significant predictors in Washington but not Minnesota.

At the student level, we discovered that gender was a significant predictor in Norway, with females playing longer than males. In Washington and Minnesota, grade was either not significant or had a small effect; this is not to say that grade level is not important, but rather that once the school and teacher are known the grade level has little effect. Grade was both significant and had a larger impact in Norway, with higher grades playing less time once the school and teacher were known. Why exactly this is and why the results are different between the three locations deserves further research; we note in passing that primary school in Norway typically spans 7 years, while primary school in the U.S. typically spans 5 years, so that there may be more opportunity for grade to make a difference within Norwegian schools.

Besides providing guidance to future researchers wanting to use campaigns as a source of experimental data, our results also suggest potential improvements to the design of campaigns themselves. We motivated this analysis by asking whether or not researchers could avoid the need to randomize within schools or teachers, in order to take advantage of competitive incentives without causing unfairness. Ideally, though, we could design other non-competitive incentives with equal or greater effects than prizes for top classes, in which case this may no longer be a concern. Furthermore, the significant result of gender in Norway and free and reduced lunch percentages in Washington schools suggest that campaigns may not be "fair." In particular, one of the essential motivations for scalable learning technologies is the ability to reach otherwise disadvantaged students: unfortunately, our results suggest that students at Washington schools of low socioeconomic status participate for less time. Future campaigns may thus wish to provide additional outreach or support to these disadvantaged students to ensure they have the chance of equal participation.

7.5.4 Fastforwarding

The population of students who participated in our Algebra Challenges were very diverse in both their ages and their incoming understanding of algebra: looking at the wide grade spread in Figure 7.3 should be evidence of this. Another piece of evidence is that we saw a wide range of times to achieve mastery in the Washington and Norway Challenges: for example, the 10th, 50th and 90th percentiles of time to achieve mastery in Norway were 1322 seconds, 2545 seconds, and 5070 seconds respectively.

Given that educational campaigns draw many types of students, and that students engage the material for a fair amount of time, some form of adaptivity is likely to be important to prevent large groups of students from being bored or frustrated. One possible method is to give a test early on, and fastforward the student to a more difficult part of the progression if the test is passed. We were able to study the effects of such a "fastforward" strategy of adaptivity in Minnesota by giving random mastery tests, and will analyze the results in this section.

We provided two types of adaptivity in DragonBox Adaptive, as was described in Section 7.4.1. These two types of adaptivity were both designed to provide *additional* practice for players who were struggling. However, players who were already adept at solving algebraic equations could become bored while working their way through the 52 minimum levels required to reach the mastery tests: that is, the progression might have been too slow for players who already understood algebra. An important component of games and other optional educational technologies is that they must be engaging, or else students may simply stop and do something else. This led us to study a third type of adaptivity in the Minnesota Challenge: with 0.65% probability, the mastery tests would be given after any level before the normal mastery test location. If passed, the player would be fastforwarded to the progression after the test; if failed, the player would be returned to the original point and continued to play as normal (possibly receiving more fastforward tests later). This had the potential to greatly reduce the number of levels required to achieve mastery.

Unlike most of the other data analysis in this chapter, which is confounded by simultaneous changes in incentives, population, and level structure, these fastforward tests can be considered a

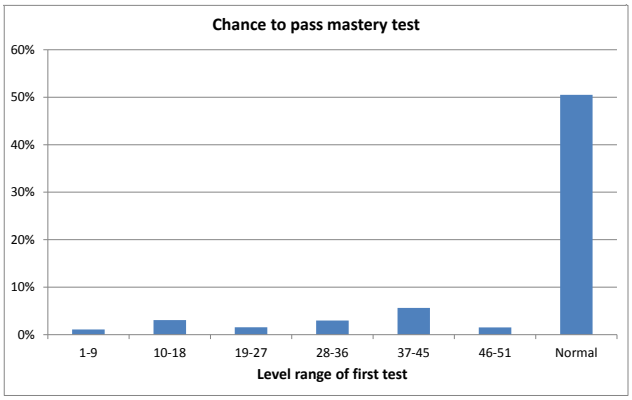
randomized experiment. By analyzing the data from the Minnesota Challenge, we hope to answer the following questions:

1. How much was the game teaching? For example, if pass rates for fastforward Challenges were the same if received in the first few levels compared to the standard point in the progression, this would be evidence the game was not improving student ability. Much like a pretest, the overall pass rate early on could also give us an indication of how much the underlying player population knew.

2. What effect do early mastery tests have on player mastery rates and engagement (measured in total time played), among players who pass them? If the boredom hypothesis is correct and a significant problem, then early mastery tests should increase both player engagement and mastery rate by preventing boredom in players who pass the tests.

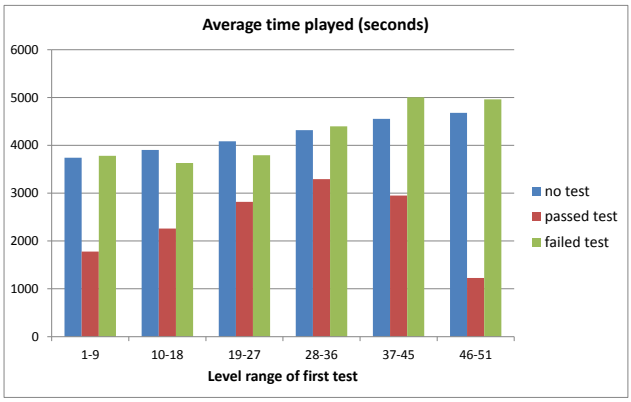
3. What effect do early mastery tests have on player engagement among players who fail them? According to the theory of flow [36], boredom and frustration are at odds: receiving and failing a difficult test could discourage players, possibly overpowering any positive effects of fastforwarding. In the best case – players are unaffected by failing a test – we would even have supporting evidence for using these types of difficult mastery tests as pretests in a campaign-game environment. This could potentially overcome one of the primary difficulties with game-based research: the difficulty in measuring learning gains due to lack of pretest-posttest data.

We hypothesized that effects on mastery rate and student engagement might change depending on when the fastforward tests showed up, with the strongest effects early on. In the standard Minnesota progression, when no random early tests were given, players had to play a minimum of 52 levels to reach the three mastery tests. Though we note that due to adaptivity which would insert extra levels, students often played considerably more than the minimum number of levels: on average students receiving the mastery tests at the normal point had to play 99 levels to do so.



(a) Pass rates

(b) Overall mastery



(c) Engagement

Figure 7.5: Results of randomly giving out mastery tests at the beginning of the game, where "no test" means that students did not receive an early test level by the end of that level block (though they might receive one later). Pass rates are very low compared to pass rates when reaching the tests through the normal progression; early tests appear to have little effect on eventual mastery rates; and failing tests has little effect on player engagement.

We then asked what effect triggering the mastery test on the n'th level would have. Given the low probability of receiving the test at any particular time, we binned the early levels into groups of 9. For any given bin, such as 10-18, we collected all the players who played at least 18 levels, then divided them into two groups: those who received no tests after the first 18 levels, and those who received no tests after the first 9 levels and at least one test after levels 10-18 (further subdivided into those who passed the tests and those who didn't). These groups correspond to the control, the highly adept who might have been bored, and those who might have been harmed by being given the test too early. The last bin, 45-51, is shortened to avoid considering players who played perfectly and received a normal test after level 52.

The results can be seen in Figure 7.5. By checking how likely players were to pass tests when given in different bins, we see from Figure 7.5(a) that players were extremely unlikely to pass the tests when given very early on, especially when compared to pass rates on tests reached normally through the progression. We found this quite surprising. Note that while it was possible for students to receive a test by level 52, very few students actually did so because the adaptive level progression would give them extra practice when they had trouble solving levels. Thus, nearly every student who received an early test, even in the 45-51 bin, had not yet reached levels with key concepts used in the mastery test. This fact, combined with the low overall pass rates, implies both that players did not already know everything the game could teach, and also that we would expect any potential gains from fastforwarding players to be minimal because so few players pass.

This turns out to be the case. By checking eventual mastery rates of players given tests in the various bins, we see from Figure 7.5(b) that the eventual mastery rates of those given or not given the test in any particular bin are very similar. Note that the mastery rates continue to increase from bin to bin because we only study players who make it to the end of each bin, so that at later bins players who quit before achieving mastery are removed. χ^2 tests confirm that giving early tests has no effect on eventual mastery rates, as seen in the upper part of Table 7.5. We conclude that our fastforward tests did not achieve the intended purpose of increasing mastery rates by allowing players to skip boring content.

We also asked whether there would be negative effects if students were given very difficult

Early test bin	χ^2 statistic	p-value ($\alpha = 0.004$)
1-9	$\chi^2(1, N = 6804) = .159$	.365
10-18	$\chi^2(1, N = 6168) = .706$	.218
19-27	$\chi^2(1, N = 5582) = .330$	.305
28-35	$\chi^2(1, N = 4952) = 2.194$	.078
36-45	$\chi^2(1, N = 4371) = .375$	.375
46-51	$\chi^2(1, N = 4059) = 5.105$	.018
Early test bin	Wilcoxon Z statistic	p-value ($\alpha = 0.004$)
1-9	$Z = -.125$	.901
10-18	$Z = -2.373$	.018
19-27	$Z = -.593$	.553
28-35	$Z = -.215$	.830
36-45	$Z = -1.213$	.225
46-51	$Z = -.189$	.850

Table 7.5: Post-hoc analyses run to understand the effect of giving early mastery tests; with 12 comparisons, the Bonferroni correction gives us critical $\alpha = 0.004$. The top table compares eventual mastery rates of players given early tests against those not given tests at those times. The bottom table compares time played by students who are given early tests and fail them against those not given tests; the Wilcoxon rank sum test was used due to non-normality of the data. No results are significant.

levels early on. This can be analyzed by comparing total time played between students who were not given tests and students who were, as seen Figure 7.5(c). We caution that only a handful of students were able to pass the fastforward tests. These students tended to play fewer levels, but there is insufficient data to draw firm conclusions; we can only speculate that they may have simply had less content because they skipped many levels, or perhaps the tests merely identified students who would not have played long anyways because they knew the content the game was meant to teach.

Interestingly enough, we discovered no statistically significant effect on engagement when students failed the early tests, as seen in the bottom half of Table 7.5. If true, this result suggests that giving pretest levels very early may be a viable way to estimate student learning without causing negative consequences for player engagement; it could also mean that games or adaptivity schemes will not cause too much harm to the student population when students are occasionally given extremely difficult problems *if they can submit an incorrect answer without penalty*.

Regardless of the specific lessons learned from our fastforward mastery test, a more general lesson learned from the Minnesota Algebra Challenge is that campaigns can be successfully used to run randomized controlled experiments. Without any competitive incentives, we were able to add extra assessment levels that could have drastic impacts on players' progress through the game, and received no complaints about this behavior. We caution, however, that players were all basically performing the same task and would have been very unlikely to notice or discuss differences in the levels they saw. Experiments with drastically different conditions would be more difficult to run.

7.6 Conclusion

In this chapter, I discussed a new method for delivering educational technology: the *campaign*. Campaigns are focused, relatively short events designed to promote and encourage the use of educational technology across a wide variety of students, and are quite different from other popular methods of delivering such technology or content, such as downloadable software, free online websites, or MOOCs. In light of the high rate of mastery (96%) among students in our campaigns who played the 1.5 hours we requested, they seem deserving of research as educational tools. Further-

more, while we have framed campaigns as being useful for delivering educational technology, it may be possible to use them with other types of content or software. This would require careful incentive design for users to participate, which we leave to future work.

To better understand the properties of campaigns, we presented a case study using three campaigns we conducted with the educational game DragonBox Adaptive: the Washington, Norway, and Minnesota Algebra Challenges. We detailed the costs and logistics of running campaigns, and described basic information about student participation and achievement in the form of time played and ability to achieve mastery of the content. To help others planning on running campaigns, we shared several of our observations. First, players played our game orders of magnitude longer than they have played other educational games offered on free websites. Second, collaborative incentives and rewards to classes for achieving mastery of the content may not have much effect on how long students play, but competitive incentives and rewards to students for finishing levels may have large effects while also leading to undesirable "gaming" behavior. Third, running experiments using campaign data can be challenging due to the difficulty of randomizing within schools or teachers, and campaigns may have differential effects depending on student gender and socioeconomic status. Finally, giving students "pre-test" levels they are overwhelmingly likely to fail does not necessarily cause frustration or other negative effects. Each of these findings requires further study, and suggests other interesting lines of research, but taken together are a promising first step in understanding how campaigns can be used to achieve learning at scale.

Chapter 8
CONCLUSION

The rise of online software has opened up new opportunities for collecting data on human behavior. Furthermore, the scale of internet data and the ability for algorithms to control the sampling process allow us to explore much larger intervention spaces than has traditionally been possible in the laboratory or by passively collecting datasets from existing websites. The tremendous increase in our ability to run experiments on education, behavioral economics, and psychology should greatly enhance our understanding of human nature. In turn, the lessons we learn from these data can then be used to improve the design and organization of our software, institutions, and even economies.

In this thesis, I argued that fulfilling this vision requires the creation of behavioral experimentation engines: algorithms capable of altering software, measuring user responses, and adapting future experiments for the sake of generating insights in the behavioral sciences. These experimentation engines are promising, but raise new challenges that must be addressed before they can become viable. My thesis tackles several of them.

- Data often comes in very quickly - easily thousands of people per day for even moderately popular games. This makes it infeasible to have a human direct the sampling process and constantly choose new experiments to run. Yet the most obvious alternative, to run all experiments simultaneously, is also untenable due to the combinatorial nature of experimental spaces. To take full advantage of these data sources, we must design algorithms that can intelligently choose experiments to run in order to achieve scientific goals. To that end, in my work on automatic hierarchical A/B testing, generalization of results, and sampling for scientific surprise, I formalize different possible scientific aims and develop machine learning algorithms capable of maximizing for these goals independently. This can even be done

when we lack full control over the software in question (through importance sampling), or in the face of delayed responses (through posterior sampling), two additional problems normally not present in standard experimental methodologies.

- Many users and software companies have certain expectations about how their software should behave. For example, students paying to take a massive open online course have a reasonable expectation that they will learn the material. An experiment drastically reducing learning outcomes for many students would both be harmful to current users and drive away future ones. For this reason, online experimentation engines run into a tradeoff between maximizing for some objective (such as learning or revenue), while collecting data that will be useful for researchers; unfortunately, these goals are usually contradictory. In my work on UCBEXPLORE, I allow the researcher to specify in advance the relative values between maximizing reward and getting better estimates of human responses to different experimental means. Over time, the algorithm self-tunes to become more or less exploratory to in order to maximize for this tradeoff.

- While online software promises data at scale, the nature of online environments often makes it hard to get certain types of data. Demographic, longitudinal, and social data are particularly hard to come by in the standard paradigm of free online software that anyone is allowed to use (and stop using) at any time. Yet clever design of new forms of software or software delivery mechanisms can alleviate many of these problems. This is the reasoning behind my work on large-scale educational campaigns. These campaigns are a new paradigm for delivering educational technology, and give us much larger volumes of interaction information on individual participants, as well as allowing us to capture and study demographic information and effects of social incentives that would have been nearly impossible to study in more standard online environments.

8.1 Future Horizons

All my algorithmic work has relied on a researcher specifying the factors worth studying. These factors form an experimental space over which my algorithms then search. If the space is too large, performance suffers. A better scenario would be to automatically identify which factors are worth studying; in this case, the researcher could specify a (possibly extremely large) superset instead of carefully curating the available factors, leaving the algorithm to decide which subspaces to explore. As we know from the optimization literature, this is not generally possible without assumptions about how factors interact with each other. Therefore, we will first need to understand what types of interactions arise naturally in the behavioral sciences, then incorporate this information in the forms of priors or dimensionality reduction techniques to automatically choose appropriate experimental spaces.

Another fundamental challenge is how a behavioral experimentation engine should interact with researchers to maximize scientific insights. Right now, they simply generate data that the experimenter must sift through, a difficult and tedious task. The data are not random, however. They are the result of precise sampling processes influenced by previous experiments. Automatic generation of experimental narratives, much as would be found in laboratory notes, might make the data more easily digestible by researchers. Likewise, integration or improvements of data visualization tools should help researchers better interpret the output from these systems and generate new hypothesis spaces to be fed into the next engine data collection cycle or specific experiments to be run in the lab.

Since my goal is to automate low-level behavioral experimentation, an even deeper question is to identify what exactly it is that scientists want, and what metrics we should maximize for to achieve these aims. While it seems clear that scientists are interested in identifying surprising new results or finding the generality of existing results, these are very loosely defined terms. I have offered some precise definitions in this thesis, but others are certainly possible. Furthermore, different fields may have entirely different criteria, for which new sampling techniques will be needed: for example, psychometricians might be interested in identifying test questions with cer-

tain statistical properties. None of my proposed sampling techniques are designed to efficiently sample for this purpose, and so new ones will need to be invented.

Finally, with the exception of social incentives in educational campaigns, my thesis research studied user-level behavior, not system-wide behavior. Studying networks through automated mass experimentation is significantly harder than studying individuals. For example, it is straightforward to ask 1,000 individual people how their saving behavior would change at different interest rates. It would be nearly impossible to find 1,000 different economies and alter interest rates in all of them to study the change in overall growth rates or wealth distribution. These network questions are of enormous importance in day-to-day life, however. Building network experimentation engines will require fundamental algorithmic changes that either make assumptions about how humans behave in groups (reducing the number of required datapoints to draw firm conclusions), or by clever design changes that allow each experiment to yield more information (ex. by partitioning an economy into many sections and simultaneously changing variables in each partition). While complicated to design and model, I am optimistic that my methods can be extended to study these questions and optimize political, economic, educational, and other social systems.

BIBLIOGRAPHY

[1] Illumina introduces the hiseq x ten sequencing system. http://www.businesswire.com/multimedia/home/20140114006291/en/#.VN0ajPl4oUo, 2014. Accessed: 2015-02-12.

[2] Shaaron E Ainsworth, Peter A Bibby, and David J Wood. Analysing the costs and benefits of multi-representational learning environments. *Learning with multiple representations*, pages 120–134, 1998.

[3] AliceR. Albrecht, BrianJ. Scholl, and MarvinM. Chun. Perceptual averaging by eye and ear: Computing summary statistics from multimodal stimuli. *Attention, Perception, & Psychophysics*, 74(5):810–815, 2012.

[4] George A. Alvarez and Aude Oliva. The representation of simple ensemble visual features outside the focus of attention. *Psychological Science*, 19(4):392–398, 2008.

[5] George A. Alvarez and Aude Oliva. Spatial ensemble statistics are efficient codes that can be represented with reduced attention. *Proceedings of the National Academy of Sciences*, 106(18):7345–7350, 2009.

[6] Erik Andersen, Yun-En Liu, Richard Snider, Roy Szeto, Seth Cooper, and Zoran Popović. On the harmfulness of secondary game objectives. In *FDG*, 2011.

[7] Erik Andersen, Yun-En Liu, Richard Snider, Roy Szeto, and Zoran Popović. Placing a value on aesthetics in online casual games. In *CHI*, 2011.

[8] Erik Andersen, Eleanor O'Rourke, Yun-En Liu, Rich Snider, Jeff Lowdermilk, David Truong, Seth Cooper, and Zoran Popović. The impact of tutorials on games of varying complexity. In *CHI*, 2012.

[9] John R. Anderson, Albert T. Corbett, Kenneth R. Koedinger, and Ray Pelletier. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2):167–207, 1995.

[10] Daniel Ansari. Effects of development and enculturation on number representation in the brain. *Nature Reviews Neuroscience*, 9(4):278–291, 2008.

[11] Amy Arbreton. Student goal orientation and help-seeking strategy use. In *Strategic help seeking: Implications for learning and teaching*, pages 95–116. Lawrence Erlbaum Associates Publishers, 1998.

[12] Dan Ariely. Seeing sets: Representation by statistical properties. *Psychological Science*, 12(2):157–162, 2001.

[13] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[14] Ryan Shaun Baker, Albert T Corbett, and Kenneth R Koedinger. Detecting student misuse of intelligent tutoring systems. In *Intelligent tutoring systems*, pages 531–540. Springer, 2004.

[15] Lisa Barrow, Lisa Markman, and Cecilia E Rouse. Technology's edge: The educational benefits of computer-aided instruction. Technical report, National Bureau of Economic Research, 2008.

[16] Larry Berger and David Stevenson. K-12 entrepreneurship: slow entry, distant exit. 2007.

[17] Donald A Berry. Adaptive clinical trials in oncology. *Nature reviews Clinical oncology*, 9(4):199–207, 2012.

[18] J.J. Biesiadecki, E.T. Baumgartner, R.G. Bonitz, B.K. Cooper, F.R. Hartman, P.C. Leger, M.W. Maimone, S.A. Maxwell, A. Trebi-Ollennu, E.W. Tunstel, and J.R. Wright. Mars exploration rover surface operations: driving opportunity at meridiani planum. *Robotics Automation Magazine, IEEE*, 13(2):63–71, June 2006.

[19] Timothy F. Brady and George A. Alvarez. Hierarchical encoding in visual working memory: Ensemble statistics bias memory for individual items. *Psychological Science*, 22(3):384–392, 2011.

[20] BrainPOP. `http://www.brainpop.com/`. Accessed: 2015-06-11.

[21] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[22] Lori Breslow, David E. Pritchard, Jennifer DeBoer, Glenda S. Stump, Andrew D. Ho, and Daniel T. Seaton. Studying learning in the worldwide classroom: Research into edxs first mooc. *Research and Practice in Assessment*, 8:13–25, 2013.

[23] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

[24] Michael Buhrmester, Tracy Kwang, and Samuel D. Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on Psychological Science*, 6(1):3–5, 2011.

[25] Center For Game Science. Dragonbox adaptive, May 2014.

[26] Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *NIPS*, pages 273–280, 2008.

[27] Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3):pp. 273–304, 1995.

[28] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *NIPS*, pages 2249–2257, 2011.

[29] Aaron Chatterji and Benjamin Jones. Harnessing technology to improve k-12 education. *Policy Brief*, 5, 2012.

[30] Sang Chul Chong and Anne Treisman. Representation of statistical properties. *Vision Research*, 43(4):393 – 404, 2003.

[31] Shein-Chung Chow, Mark Chang, et al. Adaptive design methods in clinical trials-a review. *Orphanet J Rare Dis*, 3(11), 2008.

[32] Jr. Churchill, Gilbert A. A paradigm for developing better measures of marketing constructs. *Journal of Marketing Research*, 16(1):pp. 64–73, 1979.

[33] Francis S. Collins, Michael Morgan, and Aristides Patrinos. The human genome project: Lessons from large-scale biology. *Science*, 300(5617):286–290, 2003.

[34] A. Corbett, K. R. Koedinger, and J. R Anderson. Intelligent tutoring systems. In M. Helander, T. K. Landauer, and P. Prahu, editors, *Handbook of Human-Computer Interaction, Second Edition*. Elsevier Science, Amsterdam, 1997.

[35] Council of Economic Advisers. Unleashing the potential of educational technology. 2011.

[36] Mihaly Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. Harper & Row Publishers, Inc., New York, NY, USA, 1990.

[37] Vincent de Gardelle and Christopher Summerfield. Robust averaging during perceptual judgment. 108(32):13341–13346, 2011.

[38] Bradley Efron and Robert Tibshirani. *An introduction to the bootstrap*, volume 57. CRC press, 1993.

[39] Dominique Foray. Educational innovation: An economists perspective. *Rigour and Relevance in Educational Research*, page 35, 2011.

[40] Roland G Fryer. Financial incentives and student achievement: Evidence from randomized trials. *The Quarterly Journal of Economics*, 126(4):1755–1798, 2011.

[41] Athanasios Gagatsis and Iliada Elia. The effects of different modes of representations on mathematical problem solving. In *IGPME*, volume 2, pages 447–454, 2004.

[42] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*, 2008.

[43] James P. Gee. *What Video Games Have to Teach Us About Learning and Literacy*. St. Martin's Press, revised and updated edition. edition, March 2008.

[44] James Paul Gee. Learning by design: Good video games as learning machines. *E-Learning and Digital Media*, 1(1):5–16, 2005.

[45] Philip J. Guo and Katharina Reinecke. Demographic differences in how students navigate through moocs. In *Proceedings of the First ACM Conference on Learning @ Scale Conference*, L@S '14, pages 21–30, New York, NY, USA, 2014. ACM.

[46] Jason Haberman and David Whitney. Rapid extraction of mean emotion and gender from sets of faces. *Current Biology*, 17(17):R751 – R753, 2007.

[47] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[48] Joseph Henrich, Steven J Heine, and Ara Norenzayan. The weirdest people in the world. *Behavioral and Brain Sciences*, 33(2-3):61–83, 2010.

[49] Gregory Hume, Joel Michael, Allen Rovick, and Martha Evens. Hinting as a tactic in one-on-one tutoring. *Journal of the Learning Sciences*, 5(1):23–49, 1996.

[50] Grant D. Jacobsen and Kathryn H. Jacobsen. Healt awareness campaigns and diagnosis rates: Evidence from national breast cancer awareness month. *Journal of Health Economics*, 30(1):55 – 61, 2011.

[51] Pooria Joulani, Andras Gyorgy, and Csaba Szepesvari. Online learning under delayed feedback. In *ICML 2013*, pages 1453–1461, 2013.

[52] Ross D. King, Jem Rowland, Stephen G. Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N. Soldatova, Andrew Sparkes, Kenneth E. Whelan, and Amanda Clare. The automation of science. *Science*, 324(5923):85–89, 2009.

[53] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *CHI*, 2008.

[54] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. STOC '08, pages 681–690. ACM, 2008.

[55] Kenneth R. Koedinger, John R. Anderson, William H. Hadley, and Mary A. Mark. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8:30–43, 1997.

[56] Ron Kohavi, Thomas Crook, Roger Longbotham, Brian Frasca, Randy Henne, Juan Lavista Ferres, and Tamir Melamed. Online experimentation at microsoft. In *Third Workshop on Data Mining Case Studies and Practice Prize*, 2009.

[57] Ron Kohavi, Randal M. Henne, and Dan Sommerfield. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *KDD*, 2007.

[58] Volodymyr Kuleshov and Doina Precup. Algorithms for the multi-armed bandit problem. *Journal of Machine Learning*, 2000.

[59] Lynn Kuo. Bayesian bioassay design. *The Annals of Statistics*, 11(3):886–895, 09 1983.

[60] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[61] Pat Langley. *Scientific discovery: Computational explorations of the creative processes*. MIT press, 1987.

[62] Carol Novillis Larson. Locating proper fractions on number lines: Effect of length and equivalence. *School Science and Mathematics*, 80(5):423–428, 1980.

[63] Yongwon Lee, BruceG. Buchanan, and JohnM. Aronis. Knowledge-based learning in exploratory science: Learning rules to predict rodent carcinogenicity. *Machine Learning*, 30(2-3):217–240, 1998.

[64] Richard Lesh, Tom Post, and Merlyn Behr. Representations and translations among representations in mathematics learning and problem solving. *Problems of representation in the teaching and learning of mathematics*, pages 33–40, 1987.

[65] Tamar Lewin. Education site expands slate of universities and courses. *The New York Times*, September 2012.

[66] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[67] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670, New York, NY, USA, 2010. ACM.

[68] Robert Lindsey, Michael Mozer, William J Huggins, and Harold Pashler. Optimizing instructional policies. In *Advances in Neural Information Processing Systems*, pages 2778–2786, 2013.

[69] Y. Liu. Active learning with support vector machine applied to gene expression data for cancer classification. *J Chem Inf Comput Sci*, 44(6):1936–1941, 2004.

[70] Yun-En Liu, Christy Ballweber, Eleanor O'rourke, Eric Butler, Phonraphee Thummaphan, and Zoran Popović. Large-scale educational campaigns. *ACM Trans. Comput.-Hum. Interact.*, 22(2):8:1–8:24, March 2015.

[71] Yun-En Liu, Travis Mandel, Emma Brunskill, and Zoran Popović. Towards automatic experimentation of educational knowledge. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3349–3358. ACM, 2014.

[72] Yun-En Liu, Travis Mandel, Emma Brunskill, and Zoran Popović. Towards automatic experimentation of educational knowledge. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3349–3358. ACM, 2014.

[73] Yun-En Liu, Travis Mandel, Emma Brunskill, and Zoran Popović. Trading off scientific knowledge and user learning with multi-armed bandits. In *EDM*, 2014.

[74] Derek Lomas, Kishan Patel, Jodi L. Forlizzi, and Kenneth R. Koedinger. Optimizing challenge in an educational game using large-scale design experiments. In *CHI*, 2013.

[75] Merrilea J. Mayo. Video Games: A Route to Large-Scale STEM Education? *Science*, 323:79–82, 2009.

[76] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[77] Joan Moss and Robbie Case. Developing children's understanding of the rational numbers: A new model and an experimental curriculum. *Journal for Research in Mathematics Education*, 30(2):pp. 122–147, 1999.

[78] Robert Murphy, Larry Gallagher, Andrew E. Krumm, Jessica Mislevy, and Amy Hafter. Research on the use of khan academy in schools: Implementation report. Implementation report, SRI International, 2014.

[79] Harold F. O'Neil, Richard Wainess, and Eva L. Baker. Classification of learning outcomes: evidence from the computer games literature. *The Curriculum Journal*, 16(4):455–474, 2005.

[80] E. O'Rourke, E. Butler, Y. Liu, C. Ballweber, and Z. Popović. The effects of age on player behavior in educational games. In *Foundations of Digital Games*, FDG '13, 2013.

[81] Eleanor O'Rourke, Kyla Haimovitz, Christy Ballweber, Carol S. Dweck, and Zoran Popović. Brain points: A growth mindset incentive structure boosts persistence in an educational game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, 2014.

[82] Ian Osband, Dan Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013.

[83] Alexandros Paramythis and Susanne Loidl-Reisinger. Adaptive learning environments and elearning standards. *Electronic Journal of eLearning*, 2:181–194, 2004.

[84] Laura Parkes, Jennifer Lund, Alessandra Angelucci, Joshua A Solomon, and Michael Morgan. Compulsory averaging of crowded orientation signals in human vision. *Nature neuroscience*, 4(7):739–744, 2001.

[85] Warren B Powell and Ilya O Ryzhov. *Optimal learning*, volume 841. John Wiley & Sons, 2012.

[86] A.M. Pullybank, N. Dixon, and A.R. Dixon. The impact of bowel cancer awareness week. *Colorectal Disease*, 4(6):483–485, 2002.

[87] John Ross Quinlan. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[88] Anna N. Rafferty, Matei Zaharia, and Thomas L. Griffiths. Optimally designing games for behavioural research. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 470(2167), 2014.

[89] Martina A. Rau, Vincent Aleven, and Nikol Rummel. Intelligent tutoring systems with multiple representations and self-explanation prompts support learning of fractions. In *AIED*, 2009.

[90] Martina A. Rau, Vincent Aleven, and Nikol Rummel. Blocked versus interleaved practice with multiple representations in an intelligent tutoring system for fractions. In *Proceedings of the 10th International Conference on Intelligent Tutoring Systems - Volume Part I*, ITS'10, pages 413–422, Berlin, Heidelberg, 2010. Springer-Verlag.

[91] S.W. Raudenbush. Educational applications of hierarchical linear models: A review. *Journal of Educational Statistics*, 13(2):85–116, 1988.

[92] Leena Razzaq, Mingyu Feng, Goss Nuzzo-Jones, Neil T. Heffernan, Kenneth Koedinger, Brian Junker, Steven Ritter, Andrea Knight, Edwin Mercado, Terrence E. Turner, Ruta Upalekar, Jason A. Walonoski, Michael A. Macasek, Christopher Aniszczyk, Sanket Choksey, Tom Livak, and Kai Rasmussen. Blending assessment and instructional assisting. In *Proceedings of the 2005 Conference on Artificial Intelligence in Education: Supporting Learning Through Intelligent and Socially Informed Technology*, pages 555–562, Amsterdam, The Netherlands, The Netherlands, 2005. IOS Press.

[93] Katharina Reinecke and Krzysztof Z. Gajos. Quantifying visual preferences around the world. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 11–20, New York, NY, USA, 2014. ACM.

[94] Katharina Reinecke and Krzysztof Z. Gajos. Labinthewild: Conducting large-scale online experiments with uncompensated samples. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 1364–1378, New York, NY, USA, 2015. ACM.

[95] M. S. Ridout. Three-stage designs for seed testing experiments. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 44(2):pp. 153–162, 1995.

[96] Cecilia Elena Rouse and Alan B Krueger. Putting computerized instruction to the test: a randomized evaluation of a scientifically based reading program. *Economics of Education Review*, 23(4):323–338, 2004.

[97] Nicholas Roy and Andrew McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, 2001.

[98] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 441–448, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[99] Steven L Scott. A modern bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[100] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52:55–66, 2010.

[101] Burr Settles, Mark Craven, and Lewis Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, pages 1–10, 2008.

[102] Valerie Shute and Brendon Towle. Adaptive e-learning. *Educational Psychologist*, 38(2):105–114, 2003.

[103] Robert S. Siegler, Clarissa A. Thompson, and Michael Schneider. An integrated theory of whole number and fractions development. *Cognitive Psychology*, 62(4).

[104] Anna Sierpinska. On understanding the notion of function. *The concept of function: Aspects of epistemology and pedagogy*, 25:23–58, 1992.

[105] Kesar Singh and Minge Xie. Bootstrap: A statistical method, 2008.

[106] E. A. Skinner and M. J. Belmont. Motivation in the classroom: Reciprocal effects of teacher behavior and student engagement across the school year. *Journal of Educational Psychology*, 85(4):571– –581, 1993.

[107] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

[108] Andrew Sparkes, Wayne Aubrey, Emma Byrne, Amanda Clare, Muhammed N Khan, Maria Liakata, Magdalena Markham, Jem Rowland, Larisa N Soldatova, Kenneth E Whelan, et al. Review towards robot scientists for autonomous scientific discovery. *Autom Exp*, 2, 2010.

[109] Niranjan Srinivas, Andreas Krause, Matthias Seeger, and Sham M. Kakade. Gaussian process optimization in the bandit setting: No regret and experimental design. In Johannes Frnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1015–1022. Omnipress, 2010.

[110] John C Stamper, Derek Lomas, Dixie Ching, Steven Ritter, Kenneth R Koedinger, and Jonathan Steinhart. The rise of the super experiment. In *EDM*, pages 196–200, 2012.

[111] Martha Teghtsoonian. The judgment of size. *The American Journal of Psychology*, 78(3):pp. 392–402, 1965.

[112] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

[113] Kurt VanLehn. The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16:227–265, 2006.

[114] Kurt Vanlehn, Collin Lynch, Kay Schulze, Joel A. Shapiro, Robert Shelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill. The andes physics tutoring system: five years of evaluations. In *In Proceedings of the 12th international conference on Artificial Intelligence in Education*, pages 678–685. IOS Press, 2005.

[115] Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Machine Learning: ECML 2005*, pages 437–448. Springer, 2005.

[116] K. Wauters, P. Desmet, and W. Van den Noortgate. Adaptive item-based learning environments based on the item response theory: possibilities and challenges. *Journal of Computer Assisted Learning*, 26(6):549 – 562, 2010.

[117] We Want To Know. Dragonbox, May 2014.

[118] L. J. Wei and S. Durham. The randomized play-the-winner rule in medical trials. *Journal of the American Statistical Association*, 73(364):pp. 840–843, 1978.

[119] D. Whitney, J. Haberman, and T. D. Sweeny. From textures to crowds: Multiple levels of summary statistical perception. In John Simon Werner and Leo M Chalupa, editors, *The New Visual Neurosciences*. MIT Press, 2014.

[120] Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[121] Ben James Winer. *Statistical principles in experimental design.* McGraw-Hill Book Company, 1962.

[122] H. Woltman, A. Feldstain, J.C. MacKay, and M. Rocchi. An introduction to hierarchical linear modeling. *Tutorials in Quantitative Methods for Psychology*, 8(1):52–69, 2012.

[123] M. Zelen. Play the winner rule and the controlled clinical trial. *Journal of the American Statistical Association*, 64(325):pp. 131–146, 1969.

[124] Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, Language Technologies Institute, School of Computer Science, 2005.