Accountability and Oversight for the Data-Driven Web with The Hubble Privacy Telescope

Paper #XX

Abstract

Privacy has all but disappeared from today's data-driven world. Users are eager to share their data online, and web services are eager to collect and monetize that data. As conventional confidentiality-based privacy becomes a distant dream, accountability and oversight become increasingly important to attain a balance between the data's commercial value and users' privacy and best interests. This paper puts forth *scalable web accountability*, a vision and research agenda whereby a new generation of web accountability tools – developed by the systems research community – are placed in the hands of privacy watchdogs, such as investigative journalists or the Federal Trade Commission, to let them audit companies' use of personal data.

To facilitate the building of web acountability tools, we have developed *Hubble*, the first scalable and extensible infrastructure system for answering questions about data use for targeting and personalization. Hubble makes two major contributions. First, it leverages state-of-the-art statistical methods in unique ways to accurately detect targeting in black-box services based on experiments with differentiated user profiles. Second, Hubble provides an extensible and dynamic architecture that lets auditors specify hypotheses related to a targeting question of

interest, which it then explores, validates, and refines automatically, in real time, and with solid statistical guarantees. Using Hubble, we have built two web accountability tools to ask specific questions about (1) how Google targets ads within and across its services and (2) how third-party web trackers use users' browsing histories to target ads on the web. Using these tools, we have found solid evidence contradicting two Google privacy statements and XXX for trackers.

1 Introduction

What do web trackers do with the data they collect about us, such as our browsing histories or mobile locations? Are our children's online activities specifically targeted by online advertisers? Do any sites out there use our browsing profiles, or data from our Facebook accounts, to personalize their contents? Do any shopping sites tailor their prices based on such information? How does Google use our account data to target ads, news, or recommendations within and across its services? Does it use data within our accounts to target ads outside its services? And why can we at best only give abstract, dated, or small-scale answers to these questions (e.g., "Shopping sites probably tailor prices" or "Staples was found in 2012 to tailor prices based on location"), but not concrete, current, and web-scale responses (e.g., "Here is a list of the shopping sites on the Internet that tailor prices based on personal data as of March 25, 2015")?

Many of us have perhaps asked ourselves one question or another about targeting, personalization, or discrimination on the web. Indeed, in a world of pervasive data collection by web services and unrestrained data sharing by the end users, posing questions about how these services use personal data has gained critical importance. While personal data – such as browsing histories, locations, emails, or heartbeat patterns – can enable useful features for the users, such as recommendations for new movies, smart personal assistants, and beneficial health advice, its unsupervised use can also lead to unfair or deceptive practices that can harm users economically, professionally, and socially. Examples of problematic data use cases have already been discovered: several consumer sites, such as Orbitz and Staples, were found to adjust their product

pricing based on user location []; loan companies are considering using public Facebook information to decide on loans []; and ads targeting pregnant teens have caused embarrasement when mis-delivered to their parents [].

The traditional approach to prevent such situations and increase users' privacy is to equip end-users with *protection tools*. These tools either prevent collection or access to their personal data by the web services, or mitigate the effects of the data's use on the users. Examples of end-user privacy tools include: encryption, anonymity networks, ad and tracker blockers, location fuzzying tools [], etc. Unfortunately, despire enormous research in developing such systems, preserving privacy remains a challenging arms-race against powerful economic drivers and human factors. Services view users' data as a commercial asset and devise increasingly creative ways to track and leverage that information. Users appear incapable of using protection tools correctly, or unwilling to use any tools that sacrifice their connectivity, convenience, or ability to share.

In this paper, we argue for a new approach to privacy for this data-driven world, which places accountability and oversight in the foreground of the privacy arms race. Our vision is to equip privacy watchdogs, such as investigative journalists, the Federal Trade Commission (FTC), or consumer protection agencies, with *web accountability tools* that help them pose critical questions about web services' use of personal data to expose or deter any unfair or deceptive practices. Our informal discussions with a number of investigative journalists and FTC officers confirm the great demand for such tools, which today are extremely difficult to build due to a lack of scientific methods and infrastructures to enable them.

To facilitate the building of web acountability tools, we have developed *Hubble*, the first scalable, reliable, and extensible infrastructure system for answering questions about data use for targeting and personalization. The first paragraph of this section includes examples of the kinds of questions Hubble aims to support. To investigate a question of interest (e.g., "How do web trackers use the data they collect?"), a privacy

watchdog (a.k.a., auditor) develops a set of targeting hypotheses (e.g., web trackers use the data to target ads on the web, or to personalize web page contents). The auditor then implements an API to specify these hypotheses in terms of putative personal data inputs (such as visited pages) that are thought to be used to target putative outputs (such as ads). Hubble then runs the experiments from the vantage points of multiple user profiles with differentiated inputs, and uses statistical correlation of the inputs and the outputs to predict the targeting. We find this methodology surprisingly flexible and supportive of many questions about targeting, personalization, and discrimination on the web.

Hubble's design brings two major research innovations. First, Hubble leverages state-of-the-art statistical methods to accurately detect targeting in black-box services. We show that these methods are well fit to the targeting problem, are accurate, precise, and flexible, and scale well with large numbers of hypotheses. Second, Hubble provides a dynamic, self-driven architecture that assists web auditors throughout their examination process, including at-scale exploration of many potential hypotheses about targeting on the web, followed by validation and detailed investigation of those hypotheses that pan out; the result is statistically significant evidence for these hypotheses.

Atop Hubble, we have built two web accountability tools to investigate (1) how Google targets ads within and across its services and (2) how third-party web trackers use users' browsing histories to target ads on the web. Running relatively large-scale experiments with these tools, we demonstrate that Hubble is accurate, scalable, and flexible. Moreover, using these tools we found solid statistical evidence that contradicts two of Google's privacy statements related to ad targeting in their services. We believe that both of these stem from abuses of their advertising infrastructure, and recommend improved filtering in some cases, along with a more careful formulation of these statements to avoid confusion.

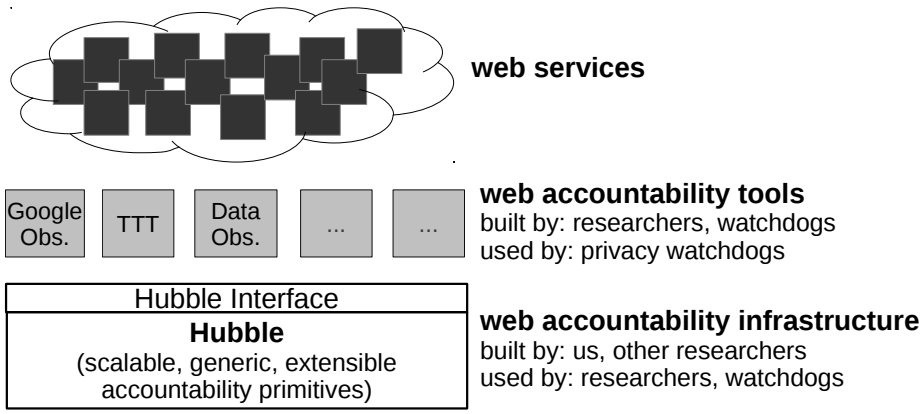

Fig. 1: **Web Accountability with Hubble.** With a few core primitives, we hope to make it easy to build transparency tools to answer at scale many important questions about the data-driven Web.

Overall, our contributions are:

1. The first accurate targeting detection mechanism shown to work well at scale. It leverages state-of-the-art statistical methods with well-known scaling properties. In contrast, we show that prior work relying on in-house algorithms and scaling proofs, does not actually scale well in real scenarios [**?**].
2. The first infrastructure system (Hubble) for web targeting experiments that assists web auditors throughout their examination process.
3. A new, flexible API and experiment design methodology that can be used to answer at scale many different questions about targeting on the web, including some questions asked by prior small-scale, limited discrimination studies.
4. Two web accountability tools that answer interesting questions about ad targeting on the web. The tools reveal two new privacy statement violations within Google and interesting targeting facts about third-party trackers.
5. Open-source releases of all components (Hubble and its tools) will be made upon publication.

2 Motivation and Vision

2.1 Example Tools

To motivate our design, we present three examples of web accountability tools (or simply tools) that we have either built, started building, or hope to build in the future atop Hubble. The examples show-case questions of increasing scope and size. All examples are driven by informal discussions we have had with several potential future users of the Hubble technology and tools: investigative journalists and FTC officers.

Example 1: Observatories for Massive Services. Google, Facebook, and other web services with giant user bases are amassing immense amounts of information about their users. Any detrimental use of personal data – intentional or not – in these services will result in massive impact on the end-users. It is therefore important for privacy watchdogs to keep these services under close scrutiny. Several journalists have already expressed interest in tools to help them investigate questions related to major companies. Examples of questions we have received: "What [are] Google and other companies doing [with] information collected on K-12 users of their enterprise platform. [Are they] using that for advertising or commercial purposes on non-enterprise platforms?" and "Is Google XXX some other question?" Answering such questions today with some degree of scale and certainty would be difficult for a knowledgeable researcher and very likely .

Using Hubble, we have started to build GObservatory, a tool that monitors Google's use of account data to target ads, recommendations, and other aspects within and outside its services. Using it, we have .

Example 2:

Example 3:

2.2 Design Goals

Goals: - scalability - statistically sound evidence (validations) - extensibility – ability to support many questions. Yet, the kind of question is restricted. We only help

answer questions that can be formulated as a set of hypotheses of the form "Personal data X is being used to target outputs of a particular kind."

2.3 Closely Related Works

A number of prior works have investigated various aspects about targeting, personalization, and discrimination on the data-driven web [**?**, 30, 12, 11, 23, 7, 28]. Until recently, the approach has been small-scale and purpose-specific experiments, where the tools needed to answer a specific question are developed for the purpose of that specific question. For example, Wall Street Journal's investigations of online price discrimination were done in the specific context of Orbitz and Staples [7, 28]. To study a new travel site, one would have to build that infrastructure from scratch. This approach results in redundancy between investigations and small-scale, one-off experiments. Given that the web is a large, ever-changing system with many different services, this one-off-experiment approach cannot possibly fullfill the great needs for accountability and oversight in today's data-driven web.

Two recent papers share our view of the great need for robust building blocks for transparency and accountability on the web: XRay [19] and AdFisher [**?**]. Like Hubble, they rely upon experiments with differentiated inputs to predict targeting. However, neither of them meet our requirements. XRay aims to support large-scale experiments (though our experiments contradict) and develops in-house targeting prediction algorithms that provide no guarantees on the correctness of any individual result [19]; AdFisher provides statistical guarantees but does not scale well [**?**]. **[Look again at AdFisher, not sure it offers any guarantees.]** Moreover, we deem both of these systems more as toys rather than real infrastructure systems designed to meet a comprehensive list of needs for web auditors. xxx

Finally, OpenWPM [8, **?**] is a complementary infrastructure system for automating data collection for web transparency and accountability. It focuses on streamlining the *data collection* procedure through browser automation, whereas we focus on develop-

ing the primitives for *data analysis*, as well as the end-to-end system infrastructure for designing and running these the experiments. We plan to integrate OpenWPM into Hubble in future releases to ease the implementation of data collection procedures for our users.

3 The Hubble Design

[Terminology questions: - transparency vs. accountability. What should we use? The title, abstract, intro use accountability. I feel that systems people will not react well to the vague notion of transparency. Accountability, on the other hand, can lead people to thinking about proofs of computation and other stuff like that, which are obviously out of scope. Mike Walfish will surely think of that.] xxx

The *Hubble Privacy Telescope* (or *Hubble*) is a scalable, extensible, and robust infrastructure system for data targeting experiments. Developers and auditors use Hubble to build the tools necesary to answer high-level questions about personal data targeting on the web. §2 already gave examples of such questions. This section uses the following question for illustration: "How do web trackers use the data they collect to target ads?"

Briefly, to explore targeting questions with Hubble, a developer (1) designs a series of *experiments* that test, refine, and validate *targeting hypotheses* about how specific inputs are used to target web service outputs. (2) implements *the Hubble interface* to model those experiments based on well-defined rules that ensure Hubble's effectiveness, and (3) launches the experiments with Hubble, which uses user profiles with differentiated inputs to collect outputs and analyzes the results to identify any statistically significant evidence for the targeting hypotheses. Fig.6 illustrates Hubble's architecture and developer APIs.

3.1 Architecture

Hubble's architecture (Fig.6(a)) consists of three core components: (1) *data collection workers* that gather data outputs such as ads and recommendations from the web

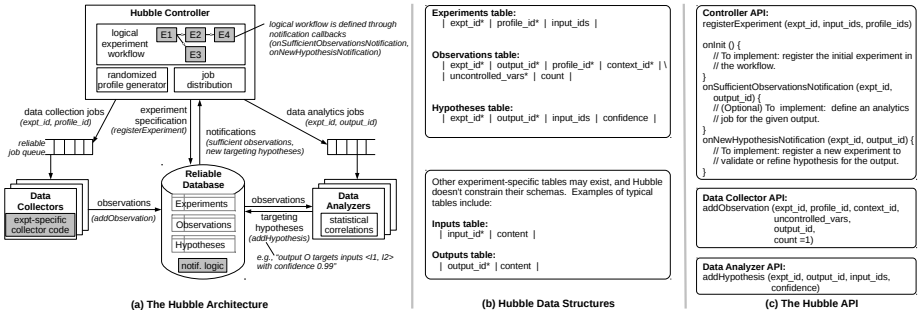

Fig. 2: **The Hubble Design.** (a) shows the Hubble architecture; grey boxes are experiment-specific, white boxes are Hubble components. Hubble has three main components: (1) the *Controller*, which coordinates the experiments and their analyses; (2) *data collection workers*, which run experiments and collect the data; and (3) *data analysis workers*, which analyze data from the experiments to detect targeting. These components are stateless; all state is maintained in a reliable, scalable database (DB). (b) shows the schema of Hubble's persistent state; * identifies the field(s) that form the primary key in each table. (c) shows the API that a developer implements and/or uses to build a data observatory tool on Hubble.

based on the user specified or Hubble generated profiles; (2) *data analytics workers* process the collected outputs to identify targeting through correlation; (3) the *Controller* handles notifications of new observations or hypotheses and starts new data collection or analytics workers as needed. All components are stateless and the state is persisted to a scalable, reliable database (DB). The controller can also generate profiles from sets of inputs. Hubble create and manages three tables that developers should not modify: `Experiments` maintains Hubble metadata about the user-defined experiments; `Observations` amasses the data obtained by the data collectors for use by the data collectors; `Hypotheses` contains all of the targeting hypotheses that Hubble validated.

In the architecture figure, white boxes denote tool-agnostic Hubble components; grey boxes denote tool-specific components that a developer implements. The most valuable generic component is the statistical correlation engine that identifies input/output targeting with minimal assumptions. The most time consuming component is the experiment specific data collection code. The data colleciton module emulates user inputs such as web browsing to create differentiated profiles and then collects outputs in the form of ads, recommendations, or other personalized facet. This typically involves

a web crawler or browser automation, a conceptually simple but fairly engineering-heavy process. The most intellectually challenging tool-specific component is the experiment design.

In the simplest form, an experiment is specified as a set of *inputs* that the developer hypothesizes might be used for targeting (e.g., the websites in a user's history might be used to target ads on the web). In practice, experiment designs are often more complex. They are specified as workflows of simple experiments that will survey an ecosystem, then validate and refine targeting hyptoheses. For example, a developer may create an initial experiment that collects information about a large number of sites that are suspected of having targeted ads and then a series of refinement experiments that determin what sites and which specific trackers ads target. Hubble's streamlined architecture supports this kind of experiment chaining and ensures collection in real time of sufficient evidence to both validate and refine targeting hypotheses.

To launch experiments in Hubble, a developer registers the first experiment in her workflow with the Controller by calling `registerExperiment` in the Hubble API. She specifies a unique ID for that experiment, the set of inputs on which to identify targeting (e.g., the set of webpages to visit), a set of profiles to exercise the inputs, and the data collection procedure to invoke for that experiment. Profiles can be either soft profiles (represented by cookies and other browser state) or accounts (such as Google accounts); soft profiles need no a priory set-up but accounts do. The Controller then assigns the inputs randomly to the different profiles and creates a data collection job for each profile. The jobs are distributed to data collection workers through a reliable job queue. One profile will be exercised by one data collection worker, which first populates its profile with the specified inputs (e.g., visits those pages) and then collects the service outputs offered to its profile (e.g., the ads shown on the visited pages). Whenever a data collector observes an output, it will report it to Hubble by calling the `addObservation` function in the Hubble API. The function persists informa-

tion about the context of the observation (such as which profile and others) into the `Observations` table in the reliable database for later analysis.

As motivated in §2, timeliness is vital for effective investigations of the ever-changing web. A key feature in Hubble is to both identify plausible targeting hypotheses and validate and refine them in as close to real time as possible. To this end, Hubble monitors the `Observations` table using a trigger-like mechanism installed in the DB. When sufficient data is available for a particular output O (e.g., when an ad was observed in the context of sufficient differentiated profiles), the DB triggers a notification to the Controller, which launches a data analytics job for that particular output O in an attempt to determine the inputs that it is targeting. The analytics job is picked up by an analytics worker, which leverages known statistical methods in unique ways to identify whether any subset of the inputs strongly correlate with the output, and if so which. In addition, the statistical methods also yield a metric of *confidence* that measures the statistical significance of their guess. All data needed to do the correlation is in the `Observations` and `Experiment` tables.

For example, using the information about the profiles in which ads were seen, statistical correlation may find that an ad O is often seen in profiles that include websites $I1$ and/or $I2$ in their histories, and never in profiles missing one or both of these websites. In such a situation, statistical correlation will conclude that O targets $\{I1, I2\}$ with high confidence (e.g., .99). This association, along with its confidence, will be added to the `Hypotheses` table in the DB. Other outcomes exist for the analytics job. First, the statistics may find that there is sufficient evidence in the observations to flag the ad as *untargeted* against any of the explanatory inputs, in which case the correlation engine will add O, $\{\}$, $confidence$ into the `Hypotheses` table. Note that the ad could still be targeted against aspects that the experiment did not model as inputs to vary in the experiments (e.g., the ad is targeting the city in which the data collectors were run). Second, the statistics may find that the evidence to make a targeting conclusion either

Fig. 3: **TTT Experiment Design.** E1 is a large-scale exploratory experiment to find cross-website targeting of ads. E2 is a focused refinement experiment to find which trackers target ads between particular pairs of websites.

way may be insufficient. In such situations, no targeting hypothesis is added to the database. If later on, more observations of the ad are amassed through data collection, then the correlation job will run again, which may enable a more precise outcome.

Like the `Observations` table, the `Hypotheses` table also has a trigger installed, which notifies the Controller whenever a new targeting hypothesis with some minimal confidence is added to it. Upon receiving the notification for a new targeting hypothesis (O, $\{I1, I2\}$, $confidence$), the Controller invokes a developer-provided callback, `onNewHypothesisNotification`, which determines the next steps. This is where a developer can register any validation and/or refinement experiments in her workflow, which focuses on the newly discovered targeting hypothesis and either gathers more data to further increase the confidence or asks a different question (e.g., which specific tracker was responsible for targeting ads against webmd.com). To register a new experiment, the developer will use again the `registerExperiment` method in the Hubble API, and Hubble will launch that new experiment (or experiments if there are multiple) similarly to the starting experiment.

3.2 API

The Hubble API is best explained with an example. Fig.3 shows a simplified version of the experiment design in TTT, which we will use as an example throghout the rest of this section. The design has two phases: (1) a first exploration phase, which

discovers cross-website targeting and (2) a second refinement phase, which for each targeting hypothesis for an ad A, it determines which tracker(s) were responsible for the targeting. In the simplest case (shown in the figure), each phase runs one experiment, chained one after the other: $E1 \rightarrow$ E2. In reality, a more complex design is needed for this experiment, which involves further experiments and is described in §3.4.

[Naming scheme doesn't match API.] The Hubble API requires the developer to xxx implement 4 experiment specific components along with an optional 5th component. First, the developer must populate the experiment `Experiment` with 3 callbacks: `init`, `onSufficientObservationNotification`, and `onNewHypothesisNotification`. The `init` funtion is responsible for initializing the experiment including, populating the database with all inputs (pages to visit or search queries), creating all profiles that will be used to collect data, and optionally assigning inputs to profiles. By default, Hubble will assign each input to a profile with probability 0.5 but this can be overriden if an experiment requires more complex assignment. `onSufficientObservationNotification` contains the functionality to respond to new observations. At it's simplest, `onSufficientObservationNotiviat` will create a hyptothesis job the `Experiment` but may implement more complex funcionality such as updating experiment specific analytics. `onNewHypothesisNotification` is called the analytics worker generates new hypotheses with sufficient confidence. It's main responsibility is to start new experiments in the workflow if needed and publish results of the current experiment.

The developer must also implement the `startCollection` function on the `CollectionWorker` object. `startCollection` implements the functionality to collect all data for a specific profile and calling `addObservation` function for each output observed. For example, in *TrackTheTrackers* `startCollection` will drive a web browser to visit all of the web pages associated with a browsing profile and collected all of the trackers and display ads observed on those pages.

Last, the developer may override the `startAnalysis` function on the Hubble`AnalysisWorker`. Hubble provides robust statistical methods to detect targeting but these methods may not be appropriate for all circumstances. For these cases, Hubble allows the developer to implement their own analysis mechanisms.

[Need to explain uncontrolled variables and other API. The section is too oriented toward how the system works. This is an API section, which needs to focus on the API (what not how).] xxx

3.3 Statistical Correlation

A core contribution in this paper is to show how out-of-the-box, traditional statistical tools can be combined to accurately detect targeting and personalization in black-box services. Previous work aiming to discover targeting invented new algorithms and proved properties about them [**?**, **?**]; in our experience, these algorithms are fragile, inflexible, and do not scale with large numbers of hypotheses (contrary to claims). For Hubble, we opted to leverage established statistical methods with well understood properties and solid implementations to detect targeting; we find our mechanisms precise, scalable, and flexible. Moreover, we foresee great potential to tap into the enormous and highly active body of work on statistical correlation methods to develop increasingly robust and expressive targeting detection mechanisms.

This section describes how we combine known statistics to detect targeting from experiments with differentiated-input profiles.[1]

[Daniel: I replaced response variables with output variables to match our terminology from the rest of the paper. I'm sure response is a more general term, but for the purposes of this paper, let's stick with output.] xxx

Linear and Logistic Regression. The core statistical method we build upon in Hubble to generate targeting hypotheses is *linear regression*. The basic linear regression model posits that an *output variable* y is determined by a linear combination of p *input*

[1] While the core system developers are systems researchers, our use of statistical methods was supervised by a co-author expert in statistics.

variables $x_1, x_2, \ldots, x_p$, plus a random noise term ε with mean zero. For example, the input variable x_7 could be a $\{0, 1\}$ indicator for whether a particular profile visits website W_7; the output could be the number of times the ad is observed in that profile. Using vector notations $\mathbf{x} := (x_1, x_2, \ldots, x_p)$ and $\mathbf{w} := (w_1, w_2, \ldots, w_p)$, the linear model is written as $y = \langle \mathbf{x}, \mathbf{w} \rangle + \varepsilon$. Here, $\mathbf{w}$ are the *regression coefficients*, and $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_i a_i b_i$ is the dot product between vectors.

Given n vectors of input values $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ together with their corresponding output values $y^{(1)}, \ldots, y^{(n)}$, the goal of linear regression algorithms is to estimate the regression coefficients $\mathbf{w}$. Many options exist for estimating the coefficients, each with different properties. However, not all are suited for our problem, so choice of algorithm requires care. For example, many traditional statistical regression algorithms (such as ordinary least squares) are only applicable when the number of input vectors is at least the number of input variables (i.e., $p \leq n$). In contrast, in many of our problems, we have many input variables (e.g., the list of websites to try) and significantly fewer input vectors (i.e., $p \gg n$).

Sparse linear regression is a better fit for our problem. It is designed to estimate $\mathbf{w}$ specifically for cases where $p \gg n$, but the regression coefficients $\mathbf{w}$ are assumed to be sparse—i.e., have only a few non-zero entries. This last condition makes the assumption that only a few input values will, in combination, be correlated with the output. One known algorithm for sparse linear regression is Lasso [27]. Under certain conditions on the n input vectors (which we discuss XXX where XXX), Lasso has be proven to estimate $\mathbf{w}$ accurately (with bounded error) as long as $n \geq O(k \log p)$, where k is the number of non-zero entries in $\mathbf{w}$ [4]—i.e., the number of input variables potentially correlated with the output.

The linear regression model is not always a suitable model. In particular, when the output is binary-valued, a generalized linear model called *logistic regression* is often a better fit. For instance, the output could be an indicator variable of whether a particular

Profiles	Displays		IPs	Input1	Input2	Input3
	#	bool				
profile1	2	1	ip1	1	1	0
profile2	9	1	ip2	1	0	1
profile3	0	0	ip2	0	1	1

(a) **Simple data model matrix**.

Profiles	Context	Displays		IPs	Input1		Input2		Input3	
		#	bool		C	P	C	P	C	P
profile1	input1	2	1	ip1	1	1	0	1	0	0
profile1	input2	0	0	ip1	0	1	1	1	0	0
profile2	input1	7	1	ip2	1	1	0	0	0	1
profile2	input3	2	1	ip2	0	1	0	0	1	1
profile3	input2	0	0	ip2	0	0	1	1	0	1
profile3	input3	0	0	ip2	0	0	0	1	1	1

(b) **Full data model matrix** that includes context.

Fig. 4: **Hubble's data models.** Examples of matrices for Hubble's data models, on three accounts and three inputs, where IP adresses are followed as uncontrolled variables. Displays are used either as a number (#) in linear regressions, or as a boolean value (bool) in logistic regressions. For the full model, each input has two variables: one to measure the influence of the input's context (C) and one for its influence through the whole profile (P).

ad is displayed to the user on a website. This model posits $\Pr[y = 1] = g(\langle \mathbf{x}, \mathbf{w} \rangle)$, where $g(z) = 1/(1 + e^{-z})$. To estimate $\mathbf{w}$, a variant of Lasso has been developed, called L_1-regularized logistic regression [24]. While the theoretical guarantees for this method are not as established as those for Lasso, empirically studies across multiple domains (e.g., XXX, XXX) have demonstrated it to be effective at estimating sparse regression coefficients.

With these basic tools in place, we next describe how we apply them to the targeting problem in Hubble.

[(From Daniel): Is it necessary to use exact code in Fig.5? Seems like some details are unnecessary.] xxx

Modeling the Targeting Problem. Hubble correlation algorithms leverage both linear and logistic regressions, as well as two different data models (Fig.4) to perform regressions at different granularity of observation. Fig.5(a) shows the code called to run the regressions for each pair of regression type and data model. Hubble is written in Ruby, but uses an implementation of lasso in R (glmnet library), a programming language specialized in statistics and data analysis.

For linear regressions the output variable y is $log(\#displays/\#trials)$. We normalize with the number of trials during the whole experiment to avoid given too much weight to inputs we would scrap more often. Taking the log is a common practice from statistics when ...

```
# familly is :logistic or :linear
# model is :full or :simple
def self.regression(expt_id, output_id,
    training_set, familly, model)
  R.data = matrix(expt_id, output_id,
      training_set, model)
  R.familly = familly
  if familly == :linear
    R.displays = data[:log_displays]
  elsif familly == :logistic
    R.displays = data[:bool_displays]
  end

  R.eval <<EOF
    fit <- cv.glmnet(data, displays,
        familly)
    1se <- coef(fit, fit$lambda.1se)
  EOF
  return select_params(expt_id,
    output_id, profiles, R.1se)
end
```

(a)

```
def self.correct(pvalues)
  R.pvalues = pvalues
  R.eval <<EOF
    adjusted <- p.adjust(pvalues,
        method="BY")
  EOF
  return R.adjusted
end
```

(c)

```
def self.pvalue(data, guesses,
    testing_set)
  # mappings is {profile: inputs},
  # observations {profile: #displays}
  mappings = data[:mappings]
  observations = data[:observations]
  profiles_w_ad =
    observations.select { |_, n| n > 0 }.count
  R.p_random = profiles_w_ad / testing_set.count

  R.right = R.wrong = 0
  testing_set.each do |profile|
    if (mappings[profile] & guesses).count > 0
      # at least one input guessed present,
      # we predict ad presence
      observations[profile] > 0 ? R.right += 1 : R.wrong += 1
    end
  end

  R.eval <<EOF
    r <- binom.test(c(right, wrong), p_random)
    pval <- r$p.value
  EOF
  return R.pval
end
```

(b)

Fig. 5: **Correlation code** to (a) run regression, (b) compute p-values and (c) correct for multiple inferences.

The two different data models allow Hubble to perform the analysis at a profile or a context granularity. First, the simple data model (Fig.4(a)) measures the number of occurrences of an output at a profile granularity. Each input is then modeled as a vector that indicates if the input was present in the profile. The regression will thus measure the correlation between displays of the output, and the presence of an input in a profile, allowing Hubble to detect targeting.

In some cases however, when the current context of an output (e.g., the currently displayed web page) is meaningful to the output (e.g., and ad), this context gives us finer grain information. The full data model (Fig.4(b)) leverages this information to better detect correlation between inputs and outputs. The displays are counted at the level of a tuple (profile, context input) and each input has two explaining variable. One

for context effects, which is a vector that indicates if the input was the current context, and one for profile effects that indicates if the input was present in the profile.

Implementation in R. Mention Spark, too, here, please. Mention that R is used at whichever company.

Statistical significance and p-value. An important contribution of Hubble's correlation detection is to provide generic confidence levels for targeting guesses using p-values. The p-value is a metric of confidence for statistical inferences that measures how likely results are under a null hypothesis. Formally the p-value is the proportion of the null hypothesis distribution that can give results equal or more extreme than what is observed. The smaller the p-value, the less likely observations are to be seen under the null hypothesis, and the more confident we are that the inference is statistically significant.

For example if Hubble's algorithms detect that ad_1 is targeted on a specific website, we should see this ad more often in profiles that visit this website. This means we should be able to predict in which profile this ad will appear. In this case, the null hypothesis is a random guess: how much better are we than a random guess to predict the presence of an ad? To assess targeting confidence, Hubble splits the dataset into a training set and a testing set (usually 70% and 30% respectively) before performing the analysis. The training set is used to run the algorithm and detect targeting. We then use the testing set to measure confidence. In each profile that contains a targeted input, we predict that the output will appear. If our predictions are so good that a random guess is very unlikely to produce them, the p-value will be small and we are confident that our inference detected a real correlation. Fig.5(b) shows the code that performs this prediction testing to compute the p-values. We leverage the R implementation of a binomial test to compare our ad presence predictions to many random guesses with a probability that corresponds to the fraction of profiles that have the output.

Correlation vs. Causation. Thus far, we have discussed concrete results and methods for inferring *correlation* between a set of input variables and an output. We now turn to the question of inferring *causal* relationships, a subject of intense study in statistics largely due to its importance in domains such as epidemiology and the social sciences. Generally speaking, when we have control over the input variables, it is indeed possible to determine their causal effects.

To illustrate this, suppose (for simplicity) that the input variables are binary-valued, and that we are interested in determining the causal effect of a particular variable x_7, which we are able to manipulate. First, what is the causal effect of x_7? If we set $x_7 = 1$ in a particular profile, then we observe the output y corresponding to the profile with $x_7 = 1$. However, we do not observe the potential outcome that would have been observed had we set $x_7 = 0$. Let us call these two potential outcomes y_1 and y_0, respectively: if $x_7 = 1$, then $y = y_1$; if $x_7 = 0$, then $y = y_0$. The *average causal effect* of x_7 on y is the average difference between y_1 and y_0 conditioned on $x_7 = 1$, which is written as $\mathbb{E}[y_1 - y_0 | x_7 = 1]$. However, because y_0 is not observed when $x_7 = 1$, it is not immediately obvious how to estimate this quantity.

A natural attempt at estimating average causal effect is to use an empirical estimate of $\mathbb{E}[y|x_7 = 1] - \mathbb{E}[y|x_7 = 0]$ based on $y^{(1)}, \ldots, y^{(n)}$ and $x_7^{(1)}, \ldots, x_7^{(n)}$:

$$\widehat{\text{ace}} := \big\{\text{avg. of } y^{(i)} \text{ where } x_7^{(i)} = 1\big\} - \big\{\text{avg. of } y^{(i)} \text{ where } x_7^{(i)} = 0\big\}.$$

This estimate, however, is generally compromised by selection bias, since

$$\begin{aligned}
&\mathbb{E}[y|x_7 = 1] - \mathbb{E}[y|x_7 = 0] \\
&= \underbrace{\mathbb{E}[y_1 - y_0 | x_7 = 1]}_{\text{average causal effect}} + \underbrace{\mathbb{E}[y_0|x_7 = 1] - \mathbb{E}[y_0|x_7 = 0]}_{\text{selection bias}}.
\end{aligned}$$

For example, if $x_7 \in \{0, 1\}$ indicates if a patient goes to a hospital, and $y \in \{0, 1\}$ indicates if the patient dies; then the selection bias could be large simply because very

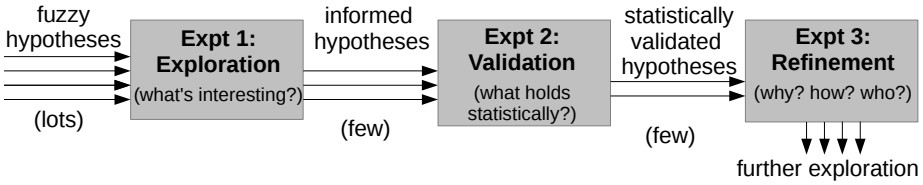

Fig. 6: **Control-Loop Experimental Design.** This is probably not quite te figure we want here – too general and probably not what we can show.

sick people are more likely to go to the hospital. The selection bias could result in $\widehat{\text{ace}}$ showing a large effect even when the true average casual effect is zero.

However, *if x_7 is determined independently of all other variables* (e.g., with an independent coin toss), then the selection bias vanishes! Indeed, in this case, $\mathbb{E}[y_0|x_7 = 1] = \mathbb{E}[y_0|x_7 = 0]$, so the estimate $\widehat{\text{ace}}$ truly estimates the average casual effect of x_7 on y. This means that we may evaluate casual effects of variables that we are able to independently control.

[The above paragraph is hard to understand. I would suggest revising it a bit. I like having the math in, but I would give a bit more information at the end about the implications of the independence assumption (maybe state clearly that the variables must be independently controllable). This would link very well into the experiment design subsection (the assumptions lead to rules for effective experiment design). Daniel: would you care to write a first draft of that?] xxx

3.4 Experiment Design

Talk about the various purposes of doing control loop design for a particular tool. Fig.**??** illustrates those varied purposes abstractly. One purpose is to study *why* some effect is happening (e.g., attribution) – e.g., the trackers. Another purpose is to validate (interesting) hypotheses whose confidence is weaker than desired by an auditor (though it is still high enough to believe that there is indeed an effect).

Rules for Experiment Design. Experiment design must fit certain critical rules to be effective with Hubble. First, Hubble assumes that all inputs in a particular experiment

can be independently controlled betweeen one another. I.e., the assignment of inputs to profiles must be allowed to be uniform at random. Any relationship that may cause a particular input to will result in violations of causality.

3.5 Accountability Tool Testing

Hubble is flexible enough to support new analysis algorithms, and to be applied to many different questions and hypotheses. It is thus crucial to develop benchmarking techniques to compare new algorithms to existing ones, and to assess the performances of Hubble's analysis building blocks applied to new transparency tools. Because most web services don't provide a ground truth for their targeting, and because manual labelling is tedious and unreliable, we need an automated, reliable benchmarking technique.

Hubble offers transparency developers the ability to test the precision of their tools even in the absence of available ground truth. Once again, Hubble leverages common practices from statistics. To validate or invalidate a prediction, Hubble uses a threshold on the p-values described in §3.3. To be conservative when performing the evaluation we also set aside a bigger proportion of the data for testing compared to regular confidence computation. This gives less training data to make guesses, but ensures more trustworthy p-values.

A common pitfall with p-values arises when performing multiple statistical inferences on the same dataset, as we do in Hubble to study multiple ads we collect during our measurements. This causes a *multiple comparison problem*: the risk of false positive increases compared to p-values for a unique inference. Intuitively by looking at many different hypotheses we are more likely to find statistical significance by mistake. A common technique to take this effect into account is to do a *false discovery rate* procedure that penalizes p-values to control the expected proportion of false positives. In Hubble we use the Benjamini-Hochberg procedure [3] and report penalized p-values

to evaluate algorithms. Once again, we leverage an R implementation as shown in Fig.5(c).

To the user, Hubble provides an evaluation mechanism to test the correlation algorithms on new transparency tools, as well as a benchmarking tool on existing datasets to develop new correlation algorithms.

4 Hubble-based Observatories

We have begun to build two observatories on Hubble. While more work is needed before we can have comprehensive products, our observatories are posing a set of diverse and compelling questions about how Google and Web trackers in general target ads. To further test our system's support for other kinds of questions, we have additionally gone through the experiment design phases for several targeting questions posed by prior art. This section describes both our observatories' implementations and our designs of prior experiments.

I want this section to include full specs of experiment design (best with code that the programmer would write, e.g.: implementations of onInit, on...Notification, blah). We need to choose a format and stick with it across all examples.

4.1 GObservatory

[Francis. Describe what we support currently: - How does google target ads on Gmail using Gmail (obsolete)? - How does google search target ads? - How does youtube target recommendations? - How is youtube data used to target ads outside of Google? Say that we're limited by the collector sides, not by our analyses or interfaces. Show a couple of examples of how the experiments work.] xxx

Google Ad Network (AdSense, AdWords, (others?)) is one the most widespread ad system. But unlike others, Google also collects a huge amount of data from its users. Controlling both sides of the advertisement business could be very dangerous for users' privacy. Therefore we built a "Google Observatory", using the Hubble interface to gain a deeper understanding of Google's system, and try to answer the following questions:

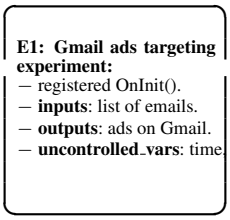

Fig. 7: **Gmail ads Experiment Design.**

Fig. 8: **Youtube → outside Google Experiment Design.** E1 is a large-scale exploratory experiment to find cross-sevice targeting of ads. E2 is a focused refinement experiment to find if video ads on Youtube are used (and if so, which) to target ads.

- How does Google targets ads using Gmail (obsolete)? - How does Youtube targets recommendations? - Is Youtube data used to target ads outside of Google websites?

We built data collectors for these specific services, but many other questions can be investigated providing suitable collectors. For instance, Fig.7 shows a one-level experiment using Hubble to determine which emails are used to target ads in Gmail. Fig.8 is a 2-level experiment to detect if Youtube's data is used to target ads outside of its context.

4.2 TrackTheTracker (TTT)

To exercise and test Hubble, we built TTT. TTT is a system built using the Hubble interface that attempts to answer the question "What ads in my browsing history are targeted on which trackers?" Give some background about trackers or something like that.

The TTT system is a workflow of three hubble experiments. The first experiment, E_1, in the workflow is a broad survey experiment to find targeted advertisements on promising websites. The second, E_2, is a smaller experiment to validate the potentially targeted ads from the survey findings and prune out unfavorable advertisements and websites. The final experiment, E_3 in the workflow very specific experiment to actually determine tracker targeting based on the results of the survey and validation experiments.

The purpose of E_1 in the workflow is to find promising ads and targeted websites on which to conduct the tracker targeting data collection. In E_1 an input is website to be surveyed. E_1 consists of 100 input websites with ten pages each that are assigned to each profile with probability 0.5 using the default Hubble profile generation capability. The `data collection worker` assigned to a profile drives a headless browser using Selenium [**?**] to 10 constant pages on each input site assigned to the profile and records all display ads observed on each site as Hubble outputs. In addition to collecting display ads, the `data collection worker` also records all trackers observed on each site for use in future experiments in the workflow. TTT uses the standard Hubble statistical methods to generate hypotheses about which output display ads target which input sites.

E_2 attempts to validate the targeting results from E_1 in a more controlled environment. E_2 creates groups of profiles where each group consists of the site targeted by an ad and all of the sites on which that ad appears. Each group will have the size of the group plus 20 profiles where each site is assigned to a profile with probability 0.5. The data collection and analysis follow the same procedure as E_1. The ads and their respective groups of site validated as targeted in E_2 will be used in E_3.

E_3 is similar to the first two experiments and uses the same groups of sites as E_2 but instead of using sites as inputs E_3 uses the trackers collected in E_1. For each group of sites TTT takes the union of all of the trackers observed on those sites and creates the

magnitude of that set plus twenty accounts for each group. Using the standard Hubble assignment mechanism each tracker is assigned to a profile with probability 0.5. The `data collection worker` used in E_3 drives blocks all trackers not assigned to that profile. It will then drive a headless browser to all sites in that group and collects all observed ads as outputs. TTT then uses the standard Hubble statistical methods to determine which ads target which trackers."

4.3 Modeling Prior Studies

[Augustin.] Write the followings. Please insert code that shows how you would model each. xxx

Personalization of News/Search Studies. Please look at the Bobble paper and try to model the questions they ask with our own API/models. Specify the differences in semantics between the results that they would achieve vs. what we would achieve.

Price Discrimination Studies. Please look at some price discrimination study and do the same. Specify the differences/llimitations of our study compared to theirs. One limitation is that we will not be able to model well continuous outputs like the price. Say that the stats do permit incorporating continuous variables, but right now we only support categorical variables (inputs and outputs).

5 Hubble Evaluation

5.1 Accuracy on Ground Truth

[Mathias.] Evaluate precision/recall of Hubble against ground truth. The results shown here are exclusively from Amazon, YouTube, and the labeled Gmail experiment. Clearly state that Gmail is manual and potentially faulty, and that all three are from simple-case and small-scale experiments. Conclude that Hubble does reasonably well. Remind that XRay does reasonably well on those workloads, too, and say that the differences come at scale and with complex input structures. But say that scale requires a different kind of evaluation methodology than ground truth. So we evaluate that next. xxx

5.2 Tool Testing Evaluation

[Mathias.] This section validates our proposed evaluation methodology by comparing its conclusions with those one would reach with a ground truth. Hopefully, at least for Amazon and Youtube, it should be the case that the conclusions one would reach by evaluating a tool against ground truth would be very similar to those conclusions one would reach by employing our test methodology. xxx

Now that the evaluation tool is validated, we next evaluate the Hubble at scale.

5.3 Accuracy at Scale

[Mathias.] Precision/Recall of Hubble using our testing methodology. Use the larger and the redundant datasets from Gmail, as well as TTT datasets. The results shown here should reflect the scaling properties of Hubble, and how precision/recall varies with scale. xxx

5.4 Comparison with XRay

[Mathias.] Compare precision/recall of Hubble/stats with XRay's three algorithms. Show the tradeoffs that appear at small scale, and show that Hubble does better at scale and with complex inputs (redundancy). It would be good to include both ground truth results for Amazon and Youtube and results with our testing methodology for Gmail, TTT, and others. xxx

5.5 Performance

[In retrospect the throughput presented in this figure should be normalized to the rate at which we generate notifications. This also needs to be way shorter but I wanted to put something down.] We evaluated Hubble scalability by measuring the number of hypothesis notifications per second that Hubble could produce given a constant stream of inputs and an increasing number of *data analytics workers*. Figure 9 shows the results of this microbenchmark with the number of *data analytics workers* on the x axis and the throughput achieved on the y axis. We implemented this microbenchmark using Hubble's own API. We created a *data collection worker* that xxx

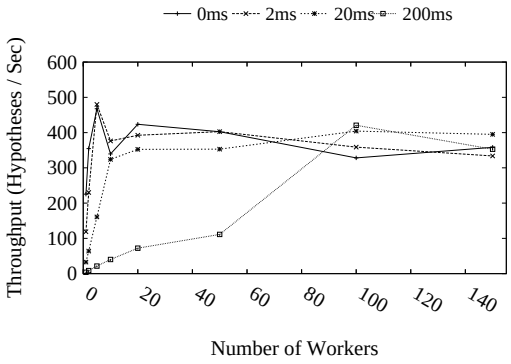

Fig. 9: **Hubble Hypothesis Throughput.** This figure is currently too small. I'll make it bigger later. Plots the number of workers

calls `addObservation` in a tight loop. We set the observation limit to one so that a notification would be produced on each `addObservation` call and each notification will create a job for a *data analytics worker*. Our *data analytics worker* paused for between 0ms and 200ms to simulate different amounts of analysis time before calling `addHypothesis`. We measured the throughput by the number of hypothesis the system produced per second as we added more hypothesis workers.

The *data collection worker* produced observations at a rate of approximately 500 notifications / second. The experiments with 0 and 2 ms latency achieve their maximum throughput with 5 workers while the experiments with 20 ms and 200 ms latency achieve maximum throughput with 10 and 100 workers respectively. Beyond 100 workers we observed contention on the master for all latencies. We deem this to be acceptable as the observation rate and hypothesis latency will be much lower during an actual deployment. We have observed that most hypothesis generation takes approximately one second and data collection notifications can be tuned by setting the notification limits.

6 Targeting Studies

We used Hubble to run a few example studies of targeting in and across various services. Our goal was not to study one service or question exhaustively, but rather to test our system's ability to answer very diverse questions about various kinds of services. We leave thorough measurement studies of targeting for future work. We used implemented pieces of our GObservatory and TTT tools to pose these questions.

6.1 Targeting of Gmail Ads

[Augustin/Francis.] xxx

Describe the setup (the last 30 days of Gmail ads on a 300-keyword dictionary). State the questions we posed: (1) what kind of targeting is prevalent (contextual vs. behavioral, on which topics in our dictionary, etc)? (2) is there targeting on sensitive topics? Use Days 14-17 to answer.

Main results:

- Show a bar graph of the top most targeted keywords (accourding to our measurements). Include always only "high-confidence" results only.

- Show a graph of contextual vs. behavioral targeting across the five days. Use this metric to differentiate: contextual = # of ad impressions within the context of email E, if we know that the ad targets E; behavioral = # of ad impressions outside the context of email E, if we know that the ad targets E. Because there are many caveats to this graph, let's be careful about the way we formulate the statements, e.g., use statements like: Hubble confirmed X contextually targeted ad impressions and Y behaviorally targeted impressions.

- We have found evidence of targeting on sensitive topics in violation of google's own targeting statements. Give examples for each of health-, financial-, race-, etc. targeting, which Google claimed it wasn't doing. Show a count of ads that we detected as targeted for each of these keywords.

- But violations are needles in a haystack. We didn't find anything for many targeting hypotheses.

6.2 Targeting in Google Search Ads

[Francis.] We posed similar questions for Google Search ads, with very different results. xxx

Main results:

- We have found no evidence of behavioral targeting on sensitive topics.

- In fact, we found evidence of only short-term historybehavioral targeting. Show a graph of how pairs are associated over time. Of course, it is conceivable that there is missed longer-term targeting on topics that we didn't look at, but mostly its very contextual.

- However, we did find another violation of Google's intentions: used guns. See others (perhaps counterfeit goods). We found a number of these examples, which suggests that Google might need a bit of revision on their filtering.

6.3 Targeting of Google Ads on the Web

[Francis.] We also wanted to see whether Google takes data from within our acounts and uses it to target ads outside its services' context. So we looked forany evidence of contamination from Google services to ads, and how it happened. We found that Youtube searches are used to target ads outside. The way it happens is quite interesting – through the Youtube ads themselves. xxx

6.4 Tracker-Based Ad Targeting

[Riley/Yannis.] [IT'S VERY URGENT THAT WE GET SOME HYPOTHESES HERE AND TEST THEM.] xxx xxx

7 Related Work

Although the topic of web transparency and accountability has garnered significant attention in several communities (e.g., security, networking, and machine learning), hardly any systems work exists, that focuses on building generic, scalable, and princi-

pled systems to make accountability and oversight more amenable on the giant, ever-changing web. Hubble is the first infrastructure system that meets a comprehensive list of practical requirements, including scale, extensibility, sound metrics for confidence levels in the results, and real-time investigations and validations of interesting hypotheses.

Targeting Measurement Studies. [Augustin.] Price discrimination studies, news/search bubble studies, ad targeting studies, etc. xxx

Web Transparency and Accountability. [Augustin.] XRay, OpenWPM, Anupam's stuff, Bobble. HTTPA (Tim Berners Lee). xxx

Further afield, much prior work has focused on *data collection transparency*, which seeks to reveal what information services *collect* about the users [26, 22, 1, 25, 15, 18, 17, 16]. The work is complementary to ours, which seeks to reveal what web services *do* with the data they collect.

Algorithmic Transparency. [Daniel.] People in ML community seem to have started to talk about algo transparency (algos that are built with transparency in mind). Talk about conceptual approaches in which the "world" can be made inherently more transparent, which might in the end help the building of tools like Hubble. HTTPA is also related to this. Same observation as for Learning theory (don't go into deep theory – OS folks will get lost). xxx

Learning Theory. [Augustin.] Whetever Rocco is doing. Present the basic goals of this field, major known results. Please don't go into details of the theory, remember you're talking to OS folks. xxx

Related Statistical Methods. Our methods for statistical experimental design and analysis draw from the subjects of *compressed sensing* [6, 5], *sparse linear regression* [27, 4], and *sparse signal recovery* [10, 9]. The experimental setups we consider correspond to sensing matrices that satisfy certain analytic properties that permit robust recovery of sparse signals. In Hubble, these signals correspond to the hypoth-

esized targeting effects we subsequently test and validate, and they are sparse when the targeting effects only depend on a small number of variables (e.g., **[e-mail types]**). xxx Our work has so far only considered simple and non-adaptive methods for generating these sparse hypotheses; there is substantial room for further exploration. First, we may support different analysis methods that improve over standard methods (like Lasso) in either computational efficiency [29, 20] or in statistical power [32]. These alternative methods may provide different computational/statistical trade-offs that can be assessed by the application developer. Many of these alternative methods also naturally fit in frameworks for distributed computation such as Spark [31] and GraphLab [21]. Second, we may use *adaptive methods* [13, 14] to potentially reduce the data collection requirements. Such methods use the outcomes of initial experiments to decide which experiments are best to conduct next. Finally, we may also explore (adaptive) recovery of *non-linear hypotheses* that are commonly used in machine learning [2]. Indeed, the theoretical framework of *learning with membership queries* may provide useful insights into experimental designs for identifying very rich classes of targeting mechanisms.

Anything Else. [Augustin?] xxx

8 Conclusions

Preserving privacy XXX a challenging arms-race against powerful economic drivers and human factors. What remains is

References

[1] ACAR, G., EUBANK, C., ENGLEHARDT, S., JUAREZ, M., NARAYANAN, A., AND DIAZ, C. The Web never forgets: Persistent tracking mechanisms in the wild. *CCS '14: Proceedings of the 21st ACM conference on Computer and communications security* (2014).

[2] ANGLUIN, D. Queries revisited. In *Algorithmic Learning Theory*, N. Abe,

R. Khardon, and T. Zeugmann, Eds., vol. 2225 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 12–31.

[3] BENJAMINI, Y., AND YEKUTIELI, D. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics* (2001), 1165–1188.

[4] BICKEL, P. J., RITOV, Y., AND TSYBAKOV, A. B. Simultaneous analysis of lasso and dantzig selector. *Ann. Statist. 37*, 4 (08 2009), 1705–1732.

[5] CANDÈS, E. J., ROMBERG, J. K., AND TAO, T. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory 52*, 2 (2006), 489–509.

[6] DONOHO, D. L. Compressed sensing. *IEEE Transactions on Information Theory 52*, 4 (2006), 1289–1306.

[7] DWOSKIN, E. Why You Can't Trust You're Getting the Best Deal Online. *online.wsj.com* (Oct. 2014).

[8] ENGLEHARDT, S., EUBANK, C., ZIMMERMAN, P., REISMAN, D., AND NARAYANAN, A. Web Privacy Measurement: Scientific principles, engineering platform, and new results. *Princeton University* (June 2014).

[9] GILBERT, A., AND INDYK, P. Sparse recovery using sparse matrices. *Proceedings of the IEEE 98*, 6 (June 2010), 937–947.

[10] GILBERT, A. C., STRAUSS, M. J., TROPP, J. A., AND VERSHYNIN, R. One sketch for all: Fast algorithms for compressed sensing. In *39th ACM Symposium on Theory of Computing* (2007), pp. 237–246.

[11] GUHA, S., CHENG, B., AND FRANCIS, P. Challenges in measuring online advertising systems. In *IMC '10: Proceedings of the 10th annual conference on Internet measurement* (Nov. 2010), ACM Request Permissions.

[12] HANNAK, A., SAPIEZYNSKI, P., KAKHKI, A. M., KRISHNAMURTHY, B., LAZER, D., MISLOVE, A., AND WILSON, C. Measuring personalization of web search. In *WWW '13: Proceedings of the 22nd international conference on World Wide Web* (May 2013), International World Wide Web Conferences Steering Committee.

[13] HAUPT, J. D., BARANIUK, R. G., CASTRO, R. M., AND NOWAK, R. D. Compressive distilled sensing: Sparse recovery using adaptivity in compressive measurements. In *Proceedings of the 43rd Asilomar Conference on Signals, Systems and Computers* (2009), pp. 1551–1555.

[14] INDYK, P., PRICE, E., AND WOODRUFF, D. P. On the power of adaptivity in sparse recovery. In *IEEE 54th Annual Symposium on Foundations of Computer Science* (2011), pp. 285–294.

[15] KRISHNAMURTHY, B. I know what you will do next summer. *ACM SIGCOMM Computer Communication Review* (2010).

[16] KRISHNAMURTHY, B., MALANDRINO, D., AND WILLS, C. E. Measuring privacy loss and the impact of privacy protection in web browsing. In *Proc. SOUPS* (New York, New York, USA, 2007), ACM Press, pp. 52–63.

[17] KRISHNAMURTHY, B., AND WILLS, C. Privacy Diffusion on the Web: A Longitudinal Perspective. In *Proc. ACM WWW* (New York, New York, USA, 2009), ACM Press, p. 541.

[18] KRISHNAMURTHY, B., AND WILLS, C. E. On the leakage of personally identifiable information via online social networks. *SIGCOMM Computer Communication Review 40*, 1 (Jan. 2010).

[19] LECUYER, M., DUCOFFE, G., LAN, F., PAPANCEA, A., PETSIOS, T., SPAHN, R., CHAINTREAU, A., AND GEAMBASU, R. XRay: Enhancing the Web's Trans-

parency with Differential Correlation . In *23rd USENIX Security Symposium (USENIX Security 14)* (San Diego, CA, 2014), USENIX Association.

[20] LIN, D., FOSTER, D. P., AND UNGAR, L. H. VIF regression: A fast regression algorithm for large data. *Journal of the American Statistical Association 106*, 493 (2011), 232–247.

[21] LOW, Y., BICKSON, D., GONZALEZ, J., GUESTRIN, C., KYROLA, A., AND HELLERSTEIN, J. M. Distributed GraphLab: A framework for machine learning and data mining in the cloud. *Proc. VLDB Endow. 5*, 8 (Apr. 2012), 716–727.

[22] MAYER, J. R., AND MITCHELL, J. C. Third-Party Web Tracking: Policy and Technology. *Security and Privacy (S&P), 2012 IEEE Symposium on* (2012), 413–427.

[23] MIKIANS, J., GYARMATI, L., ERRAMILLI, V., AND LAOUTARIS, N. Detecting price and search discrimination on the internet. In *HotNets-XI: Proceedings of the 11th ACM Workshop on Hot Topics in Networks* (Oct. 2012), ACM Request Permissions.

[24] NG, A. Y. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning* (2004).

[25] REISMAN, D., ENGLEHARDT, S., EUBANK, C., ZIMMERMAN, P., AND NARAYANAN, A. Cookies that give you away: Evaluating the surveillance implications of web tracking. *Princeton University* (Apr. 2014).

[26] ROESNER, F., KOHNO, T., AND WETHERALL, D. Detecting and defending against third-party tracking on the web. In *NSDI'12: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation* (Apr. 2012), USENIX Association.

[27] TIBSHIRANI, R. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B 58* (1994), 267–288.

[28] VALENTINO-DEVRIES, J., SINGER-VINE, J., AND SOLTANI, A. Websites Vary Prices, Deals Based on Users' Information. *online.wsj.com* (Dec. 2012), 1–6.

[29] XIAO, L. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research 11* (2010), 2543–2596.

[30] XING, X., MENG, W., DOOZAN, D., FEAMSTER, N., LEE, W., AND SNOEREN, A. C. Exposing Inconsistent Web Search Results with Bobble. *Passive and Active Measurements Conference* (2014).

[31] ZAHARIA, M., CHOWDHURY, M., FRANKLIN, M. J., SHENKER, S., AND STOICA, I. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing* (Berkeley, CA, USA, 2010), HotCloud'10, USENIX Association, pp. 10–10.

[32] ZHANG, T. Adaptive forward-backward greedy algorithm for learning sparse representations. *IEEE Transactions on Information Theory 57*, 7 (July 2011), 4689–4708.