A New Generation of Tools for Scalable Web Accountability Featuring the Hubble Extensible Infrastructure

Paper #XX

Abstract

The concept of personal privacy as a precious and fragile commodity worthy of protection has come under siege in 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 notions of privacy become a fading 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 presents our concept of *scalable web accountability*, a vision and research agenda whereby a new generation of web accountability tools are placed in the hands of privacy watchdogs, such as investigative journalists or the Federal Trade Commission, to let them monitor services' use and potential abuse of personal data.

To facilitate the building of web accountability tools, we developed *Hubble*, the first scalable and extensible infrastructure that detects data use for targeting and personalization. Hubble's two main contributions are: (1) leveraging *state-of-the-art statistical methods in unique ways* to accurately detect targeting in black-box services based on experiments with differentiated user profiles, and (2) providing an *extensible and dynamic architecture that assists web auditors* throughout their examination process, from hypothesis specification to hypothesis exploration, validation, and refinement, all in real time and with solid statistical guarantees. We used Hubble to build two web accountability tools to pose specific questions about (1) Google's ad targeting within and outside its services and (2) third-party trackers' targeting of personal browsing histories. We discovered solid evidence contradicting one of Google privacy statements [**xxx and cases of third-party tracker targeting of children's websites**].

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? Which sites 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? Do any insurance sites do so? How do Google or Facebook use our account data to target ads, news, or recommendations within, across, or outside their services? And why can we find no responses to these questions that are concrete, current, and web-scale?

Indeed, the ability to pose critical questions about personal data use on the web has become increasingly important in today's world, where the concept of personal privacy has come under siege. To defend themselves, those users aware of the risks they assume by entrusting private data to web services can choose from an array of end-user privacy protection tools, such as encryption, anonymity networks, location fuzzers, and ad/tracker blockers; however, many users do not use such tools correctly or are unwilling to use them at all if they sacrifice connectivity, convenience, application functionality, or the ability to share data. Further, the tools themselves become rapidly obsolete as web services devise increasingly creative ways to track and leverage private data for their commercial ends [**? ? ?**].

While the conventional approach to privacy protection, then, arms concerned end-users, this paper argues for the urgent development and adoption of a new and complementary paradigm, one that places accountability and oversight at the foreground of the privacy arms race. *Scalable web accountability*, our vision and research agenda, posits the creation of a new generation of web accountability tools for use by privacy watchdogs, be they investigative journalists, the Federal Trade Commission, other public agencies, or private entities that collect personal data and wish to themselves safeguard that data. Using these tools, public or private web auditors can pose and examine critical questions and reach evidence-based conclusions about how web services are using or abusing (intentionally or not) private data. Our early discussions with several investigative journalists and FTC officers confirm great demand and significant potential for impact from such tools, which today are difficult to build due to a lack of scientific methods and infrastructures to enable them.

To facilitate the building of web accountability tools, we developed *Hubble*, the first scalable, reliable, and extensible infrastructure for answering questions about data use for targeting and personalization. To investigate a question of interest (e.g., "How do web trackers use the data they collect?"), a privacy watchdog (or auditor) develops a set of targeting hypotheses (e.g., web trackers use the data to target ads or to personalize web pages). 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 statistically correlates the inputs and outputs to uncover targeting. We find this methodology surprisingly flexible and supportive of many questions about web targeting, personalization, and discrimination.

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

Atop Hubble, we built two web transparency tools to investigate, which we will release opensource along with Hubble: (1) how Google targets ads within and across its services, and (2) how third-party web trackers use personal browsing histories to target ads on the web. Running relatively large-scale experiments with these tools, we demonstrate Hubble's accuracy, scalability, and flexibility. Moreover, using these tools we uncovered solid statistical evidence that contradicts one of Google's privacy statements related to ad targeting. We believe that the violation stems from advertisers abusing Google's ad infrastructure.

On a broader scale, our research contributions include:

1. The **first formulation of scalable web accountability** and its requirements. (§2)
2. The **first infrastructure for web targeting experiments** that assists web auditors throughout their examination process. It is scalable, robust, and extensible for both auditors and researchers. (§3)
3. A **flexible API and experiment design methodology** to model many questions about web targeting, personalization, and discrimination, including some posed by prior studies [13, 14, 26, 27, 32, 36], which had to devise special-purpose tools and run at small scale. (§3.2)
4. The **first accurate targeting detection mechanism** that uses state-of-the-art statistical methods with well-known accuracy properties at scale. The key innovation is the tight interplay between system design and statistical methods that enhance each other. (§4)
5. **Two web accountability tools** that answer important questions about ad targeting on the web. (§5)
6. The **first at scale evaluation and study of targeting** by a first-party service and third-party trackers. (§6, §7)

2. Scalable Web Accountability

This paper provides initial support for our vision of *scalable web accountability*. We aim to promote the building of a new generation of web accountability and oversight tools that monitor the use of personal data on the web. We envision these tools being built by privacy watchdogs, researchers, and the open-source community using infrastructural support and primitives built by the research commmunity. Our hypothesis is that *a few core primitives can greatly facilitate the construction of a variety of tools that can answer at scale many important questions about data use that now fly under the radar of detection*. We test this hypothesis by designing and building *Hubble*, a scalable and extensible infrastructure that provides a first set of core primitives. We next walk through a specific example scenario to motivate the need for Hubble and derive its requirements, after which we show that many other use cases share these requirements.

2.1 Example Scenario

Ann, an investigative journalist, wishes to investigate whether and how third-party trackers or advertisers target children and adolescents. She hypothesizes that advertisers might leverage information amassed by web trackers to bid for users with browsing histories characteristic of children or adolescents. Ann wants to run a study to both quantify the amount of children-oriented targeting and find specific instances of what might be deemed as inappropriate or illegal targeting (e.g., targeting pornographic movies or recreational drugs at teenagers or promoting unhealthy eating habits to young children). Unfortunately, the number of websites dedicated to children is large, and there are even more neutral websites frequented by both children and adults on which ads targeted at children might appear.

Thus, Ann would like to run a *large-scale survey experiment* that tries out a lot of children's websites, looks for ads on even more websites, and finds those ads that target children's websites in particular (likely a small fraction of all ads she collects). For any case of inappropriate or illegal targeting, Ann plans to investigate through journalistic means (e.g., interview the advertiser or tracker potentially responsible for the targeting) whether the targeting was intentional. Such investigations are expensive, so Ann requires *high confidence* in a result to justify her effort.

Ann involves her technical staff, Bob, in her experiment. Bob is a decent developer with minimal background in statistics, but he is certainly no systems or statistics expert. Discussing the project, Ann and Bob quickly realize that running a rigorous experiment at scale for this purpose would be very challenging. Their initial idea is to take each pair of sites – one children's site (W_c) and some other site, W_o – and execute two sub-experiments for each pair (W_c, W_o): in one, they would launch a browser and visit first W_c and then W_o to collect the ads; in the other, they would launch a browser and visit only W_o. If they found a particular ad consistently appear in the first experiment but not in the second, they could deem it as targeted against W_c. But how should they define "consistently appear" so as to be confident of the prediction? And what if advertisers only targeted profiles with sufficient history of children's website visits? They would need to try all combinations of children's websites, which raises further scaling and confidence questions. For example, how could they obtain high confidence in their results with all the noise that arises from visiting many websites from one profile, which many trackers observe and target in different ways? The study seems unamenable.

Enter Hubble. Bob decides to build their experiment on Hubble, which provides all the primitives needed to run experiments at scale and with solid statistical guarantees. First, Hubble provides a *survey experiment primitive*, which lets an auditor efficiently explore a wide range of potential targeting hypotheses (such as childrens' website W_c is targeted by ads on website W_o); it combines those hypotheses to use minimal resources and relies on state-of-the-art statistical methods to weed out the noise and provide statistically significant correlations. Second, Hubble associates a well-understood *statistical significance metric* with each targeting result so an auditor can interpret the significance of his results. Third, Hubble provides a *validation experiment primitive*, which automatically triggers a new, laser-focused experiment to obtain further evidence for results from survey experiments whose confidence is below acceptable level for an auditor (e.g., increase from 0.9% to 0.99% confidence). Finally, if Ann and Bob wanted to investigate further aspects about each targeting hypothesis found by the survey (e.g., which tracker was responsible for targeting an ad to childrens websites), they could use Hubble's *refinement experiment primitive*, which allows the launch of additional experiments that leverage hypotheses previously discovered to pose new questions. A key property of both validation and refinement experiments is that they *trigger in real-time* as interesting hypotheses are found. Because the web changes rapidly, it is crucial to capture as much evidence as possible on the spot to allow subsequent interpretation of a result.

To build their experiment on Hubble, Bob will have to write code to combine these primitives into an end-to-end experiment design. Hubble will run the experiment, analyze the data, validate the results, and output a set of ads that, with high confidence, target children's sites.

2.2 Other Examples

The preceding example illustrates how a set of primitives in Hubble can be used to support investigations of a specific web targeting question while meeting the requirements of a specific auditor. We find that Hubble and its primitives are applicable to many other use cases involving different questions and auditors. First, Hubble is designed to support investigation of any question that can be formulated as a set of hypothesis of the form *"Personal data input X is being used to target outputs of a particular kind.* We believe that all the questions listed at the beginning of §1 are applicable. Moreover, we have surveyed several prior studies on price discrimination [] and news and search results personalization [], all of which relied on in-house, purpose-specific investigations and were run at small scales. We show those studies could have been run on Hubble for the same or improved results. Second, our anecdotal discussions with several investigative journalists and FTC officers have revealed the importance of (1) being able to run experiments that try many different hypotheses (often times auditors may not need what know what they are looking for upfront) and (2) having solid statistical justification for any result (if not a direct causal link), along with the associated evidence. We have formulated Hubble's design goals around these requirements.

2.3 Design Goals

• *Extensibility.* Hubble must be extensible in two dimensions. First, it lets auditors extend it to implement the tools necessary for their investigations. Second, it lets researchers develop new core primitives to support use cases that Hubble cannot currently support.

• *Statistical justification for inferences.* Hubble must provide statistical justification for its correlations. Any validations necessary to provide such guarantees must be run in real time, right after Hubble detects the correlation to ensure that the evidence is still available.

• *Support technical but non-expert auditors.* Hubble's direct users must have basic understanding of statistics to interpret our confidence levels and understand the difference between correlation and causation.

• *Use established algorithms.* An important decision we made in Hubble was to leverage well-established statistical methods, which are well understood and provide a rich repository of algorithms to tap into for future extensions. For example, while Hubble does not provide causal guarantees, a significant body of work exists in statistics surrounding algorithms for causal inference [**?**].

We know of no prior system that achieves these goals. The closest contenders – XRay [**?**] and AdFisher [**?**], which aim to support targeting and personalization questions – are singular building blocks rather than real infrastructures designed to provide a comprehensive set of primitives to web auditors. Moreover, we find that these systems do not scale well in practice. XRay additionally provides no statistical guarantees for its results and uses in-house algorithms that in our experience lose accuracy at scale (§6.6).

3. The Hubble Infrastructure

To explore targeting questions with Hubble, a developer (auditor, researcher): (1) designs a series of *experiments* that

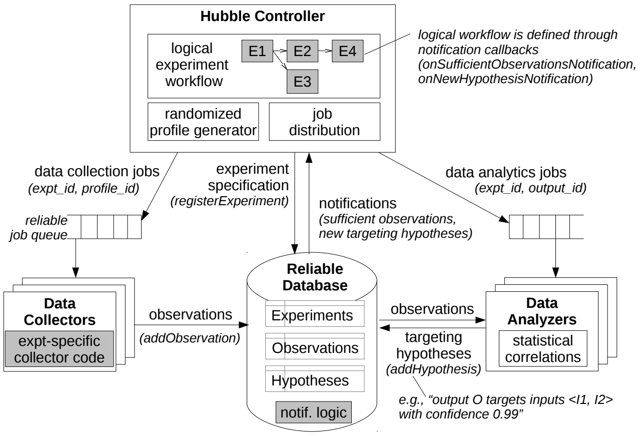


Fig. 1: **The Hubble Architecture.** Grey boxes are experiment-specific; white boxes are Hubble components.

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.1 illustrates Hubble's architecture and developer APIs.

3.1 Architecture

Hubble's architecture (Fig.1(a)) consists of three core components: (1) *data collection workers* gather data outputs (such as ads and recommendations) from the web based on profiles with differentiated inputs (such as browsing histories); (2) *data analytics workers* process the collected outputs to identify targeting through correlation; and (3) the *Controller* coordinates the experiments and their analyses by receiving notifications about new output observations and hypotheses. All components are stateless and the state is persisted to a scalable, reliable database (DB). Hubble creates 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 the targeting hypotheses that Hubble's analytics workers have produced.

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 information 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 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.5 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: $E1$ → E2. In reality, a more complex design is needed for this experiment, which involves further experiments and is described in §3.3.

[**xxx Naming scheme doesn't match API.**] The Hubble API requires the developer to 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. `onSufficientObservationNotificat` contains the functionality to respond to new observations. At it's simplest, `onSufficientObservationNotiviation` 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.

[**xxx 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).**]

3.3 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.

3.4 Testing and Evaluation

[**xxx I'll extract some of the systemsy stuff related to testing from the stats and will bring it here. I need this component as part of the infrastructure, it's an absolutely key component.**]

```
Controller API:
registerExperiment (expt_id, input_ids, profile_ids)

onInit () {
    // To implement: register the initial experiment in the
    // workflow.  Use registerExperiment.
}
onSufficientObservationsNotification (expt_id, output_id) {
    // (Optional) To  implement:  launch an analytics job
    // for the given output.
}
onNewHypothesisNotification (expt_id, output_id) {
    // To implement: register a new experiment to
    // validate or refine hypothesis for the output.
}
onDestroy () { ... }
```

```
Data Collector API:
addObservation (expt_id, profile_id, context_id,
                        uncontrolled_vars, output_id, count =1)

runDataCollection (expt_id, profile_id) {
   // To implement: write code to (1) populate profile w/
   // the assigned inputs (from Experiments table) and  (2)
   // collect outputs. Use addObservation to report outputs.
}
```

```
Data Analyzer API:
launchAnalytics (expt_id, output_ids, [params])
addHypothesis (expt_id, output_id, input_ids, confidence)
```

(a) API

Experiments table:
| expt_id* | profile_id* | input_ids |

Observations table:
| expt_id* | output_id* | profile_id* | context_id* | \
| uncontrolled_vars* | count |

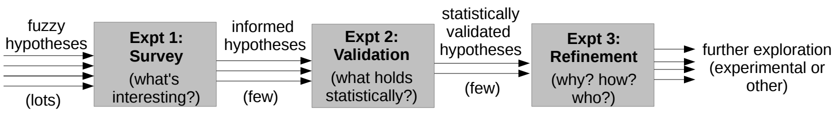
Hypotheses table:
| expt_id* | output_id* | input_ids | confidence |

Other experiment-specific tables may exist, and Hubble doesn't constrain their schemas. Examples of typical tables include:

Inputs table:
| input_id* | content |

Outputs table:
| output_id* | content |

(b) Databases



(c) Experiment Design

Fig. 2: The Hubble API (a), Databases (b) and Experiment Design (c).

(a) Shows the API that a developer (or researcher) uses to build on Hubble.

(b) Shows the schema of Hubble's DBs; * identifies the fields that form the primary key in each table.

(c) Shows a typical experiment design in Hubble; it consists of three phases: survey (large scale), validation (small, focused experiments for hypotheses found by the survey), and a refinement (further explores).

4. Statistical Correlation

A core contribution in this paper is to show how out-of-the-box tools for statistical inference can be combined to accurately detect targeting and personalization in black-box services. Previous work have proposed new but often non-statistically grounded algorithms for identifying targeting [36**?**]; 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 robust 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 inference methods to develop increasingly robust and expressive targeting detection mechanisms.

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

4.1 Basic Tools and Models

The core statistical method we leverage 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 variables* $x_1, x_2, \ldots, x_p$, plus a random noise term ε with mean zero. (Categorical variables are typically realized using "dummy variables": if a categorical variable takes K possible values, it is expanded to K mutually exclusive $\{0,1\}$-valued variables.) 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 $\mathbf{w}$.

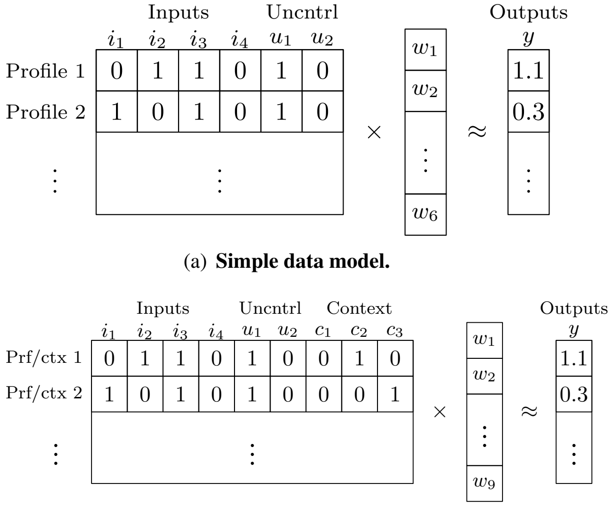
In a basic Hubble application for a particular service (e.g., ads targeting), each of the vectors of input values corresponds to a profile, and the variables $x_1, x_2, \ldots, x_p$ correspond to $\{0,1\}$-valued indicators for the possible inputs, such as the websites that the user has visited that might be used to target ads. The output variable y encodes a particular measured output of service—the number of displays for a particular ad, for instance.[2] See Fig.3(a) for an illustration (for a somewhat more elaborate data model which we discuss later).

Our use of the regression coefficients in Hubble is for screening the inputs and generating possible targeting hypotheses. Therefore, it is important to estimate these coefficients effectively. The statistical literature offers many different possible estimators, each with its own special properties. However, not all are suited for our problem: 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 $p \gg n$,

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

[2] While Hubble can measure several different outputs for any given input vector, we limit this discussion to a single output variable y.



$$\begin{array}{c|cccccc|} & \multicolumn{4}{c}{\text{Inputs}} & \multicolumn{2}{c}{\text{Uncntrl}} \\ & i_1 & i_2 & i_3 & i_4 & u_1 & u_2 \\ \hline \text{Profile 1} & 0 & 1 & 1 & 0 & 1 & 0 \\ \text{Profile 2} & 1 & 0 & 1 & 0 & 1 & 0 \\ \vdots & & & & \vdots & & \\ \hline \end{array} \times \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_6 \end{bmatrix} \approx \begin{array}{c} \text{Outputs} \\ y \\ \hline 1.1 \\ 0.3 \\ \vdots \end{array}$$

(a) **Simple data model.**

$$\begin{array}{c|ccccccccc|} & \multicolumn{4}{c}{\text{Inputs}} & \multicolumn{2}{c}{\text{Uncntrl}} & \multicolumn{3}{c}{\text{Context}} \\ & i_1 & i_2 & i_3 & i_4 & u_1 & u_2 & c_1 & c_2 & c_3 \\ \hline \text{Prf/ctx 1} & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ \text{Prf/ctx 2} & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ \vdots & & & & & \vdots & & & & \\ \hline \end{array} \times \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_9 \end{bmatrix} \approx \begin{array}{c} \text{Outputs} \\ y \\ \hline 1.1 \\ 0.3 \\ \vdots \end{array}$$

(b) **Full data model**.

Fig. 3: **Hubble's data models.** The simple data model shows values for inputs, uncontrolled variables, and outputs for two profiles. (An even simpler model would omit the uncontrolled variables u_1 and u_2.) The full data model shows these values along with the context values for two profile/context pairs.

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 well-known algorithm for sparse linear regression is Lasso [31]. Under certain conditions on the n input vectors (which we may ensure are likely to be satisfied *by construction* of our profiles), 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 $\mathbf{w}$ [5]—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 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 [28]. While the theoretical guarantees for this method are not as established as those for Lasso, empirically studies across multiple domains (e.g., [6, 34]) 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.

4.2 Hubble's Models

Hubble's correlation algorithms leverage both linear and logistic regression, as well as two different data models (Fig.3) to perform regressions at different granularities of observation—either the profile level or at a context level.

Simple data model. In the simple data model (Fig.3(a)), we measure occurrences of an output (e.g., displays of an ad) at a profile granularity. As mentioned before, each profile is treated as a vector of input values, together with an output value that is either the (logarithm of the) number of occurrences of an output in a profile (in the case of linear regression), or a binary indicator of the occurrence of the output (in the case of logistic regression). The input values are either indicators for the experimental inputs (e.g., whether a particular website was visited), or uncontrolled variables (e.g., the machine IP address, the time-of-day). We include these uncontrolled variables in the regression model to guard against erroneously correlations that could otherwise simply be explained by other factors that we do not directly control. For instance, some ads could be shown more during work hours to target office workers, but the time-of-day in which data is collected for certain profiles could inadvertantly be weakly correlated with certain inputs. Including a time-of-day variable in the regression would help suppress these erroneous correlations.

Full data model. In many cases, the context surrounding an output (e.g., the webpage on which an ad is displayed) is meaningful to the study at hand, and hence an analysis at a context-specific granularity may be desirable. The full data model (Fig.3(b)) leverages this context information to enable this detailed analysis and possibly improve detection of correlations between inputs and outputs. In this data model, output occurrences are counted at the level of a profile/context pair. Furthermore, we include additional input variables that specify or describe the context; these are often categorical, and hence expanded as a dummy variables as previously discussed. This allows context-specific hypotheses to be formed on the basis of the estimated regression coefficients for the context variables.

We have found empirically that the logistic regression method is generally more effective for the simple data model (with binary outputs), while the linear regression method better handles the full data model (with context and log count outputs), and hence use these as defaults. **[xxx check results]** Developers can choose other combinations of regression types and data models via the `LaunchAnalytics` function in the API (see Fig. 2(a) XXX).

4.3 R Implementation

Hubble is written in Ruby, but uses statistical routines (e.g., Lasso) from R, a programming environment designed for statistical computing. Although R is not primarily designed to be a library tool, its core routines we require can be readily replaced with implementations from scalable frameworks such as Spark [37] and GraphLab [23]. Fig.4(a) shows the code called to run the regressions for each pair of regression type and data model. The input and output data values are first converted into the appropriate format based on the desired data model, and then the R `cv.glmnet` procedure is invoked to estimate the regression coefficients us-

```
# reg_type is :logistic or :linear
# model is :full or :simple
def self.regression(expt_id, output_id,
    training_set, reg_type, model)
  R.data = matrix(expt_id, output_id,
      training_set, model)
  R.reg_type = reg_type
  R.displays = displays_vector(expt_id,
    output_id, reg_type, model)
  R.eval("""
    fit <− cv.glmnet(data, displays,
        reg_type)
    1se <− coef(fit, fit$lambda.1se)
  """)
  return select_params(expt_id,
    output_id, profiles, R.1se)
end
```

(a)

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

(c)

```
def self.pvalue(mappings, observations
  guesses, testing_set)
  # mappings is {profile: inputs},
  # observations {profile: #displays}
  R.n_profiles_w_ad =
    observations.keys.count
  R.p_profiles_wo_ad =
    testing_set.count − R.n_profiles_w_ad
  R.right_guesses, R.total_guesses =
    evaluate_guesses(testing_set,
    mappings, observations)
  # the −1 is to compute Pr(X >= x)
  # and not Pr(X > x)
  R.eval("""
    pval <−phyper(right_guesses−1,
      profiles_w_ad, profiles_wo_ad,
      total_guesses, lower.tail=FALSE,
      log.p=FALSE)
  """)
  return R.pval
end
```

(b)

Fig. 4: **Correlation code** to (a) run regressions, (b) compute p-values and (c) correct for multiple inferences.

ing the selected regression type. (We use the `glmnet` cross-validation procedure to automatically determine algorithm hyperparameters.) The resulting regression coefficients are then returned.

4.4 Statistical Significance with p-values

An important contribution of Hubble's correlation detection is that it provides confidence levels for its inferences (e.g., targeting guesses) via p-values. A p-value is a metric of confidence for statistical inferences that measures how likely observed 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 better than random guessing. The null hypothesis is that our predictions are the same as those provided by fair coin tosses; we use as our test statistic the number of correct predictions of ad displays across a set of accounts, and the null distribution is the binomial distribution with success probability equal to $1/2$. Hubble splits the dataset into a training set and a testing set (usually 70% and 30% of the total number of accounts, 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—i.e., run the hypothesis test. We predict that each profile containing a targeted input will be shown the ad. If these predictions are significantly more accurate than what could have been achieved by random guessing (as predicted by the null bionomial distribution), then the p-value will be small (e.g., ≤ 0.05, by convention), and thus will lend confidence that our inference detected a real correlation.

We also test the precision of our predictions, which may be a more relevant quantity than accuracy, for instance, when there is substantial imbalance between the number of accounts that are shown an ad and the number of accounts that are not. In this case, a more suitable the null hypothesis is that the accounts on which we predict that an ad was shown are chosen uniformly at random from among the test set. Here, Hubble again splits the dataset as above, and uses the appropriate testing distribution (which in this case is the hypergeometric distribution) to compute the p-value.

Fig.4(b) shows the code that performs this prediction testing to compute the p-values (for the hypergeometric test). We leverage R implementations of distribution functions to compare our ad presence predictions to many random guesses with a probability that corresponds to the fraction of profiles that have the output. The appropriate numbers of profiles with and without an output are tabulated, and then the appropriate upper-tail probability under the hypergeometric distribution is computed using an R function call. This tail probability is the desired p-value, which is returned.

4.5 Accountability Tool Testing

[xxx (from djh): i don't understand this subsection from the point-of-view of talking about benchmarking. how exactly are the p-values used to benchmark? what is the yardstick? number of discoveries? i don't think i am the right person to write about these issues.] [xxx (from rg): much of this content will be moved onto the infrastructure section. I'll fiddle with it and then will ask you to confirm that stats-wise it's correct.]

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 §**??**. 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 tests, as we do in Hubble to assess the inferences about the many ads collected during our study: that it is quite possible for some p-values to appear small purely by random chance. Indeed, after 20 independent tests, we expect to find at least one p-value that is less than 0.05, even when there are no real correlations! To address this multiple testing problem, we apply a standard correction to the p-values from all the tests from a large experiment so that we can guarantee a small expected proportion of false discoveries. The correction procedure, called the Benjamini-Hochberg-Yekutieli procedure [4], adjusts the p-values (possibly making them slightly larger) so that among the adjusted p-values that are less than (say) 0.05, the expected fraction that correspond to cases where the null hypothesis is true (i.e., false discoveries) is at most 0.05. Such a correction causes Hubble to be stricter about declaring findings, but ensures that overall, most findings can be trusted. We use an implementation of this procedure in R as shown in Fig.4(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.

5. Hubble-based Tools

This section show-cases the kinds of web accountability and oversight tools that can be built atop Hubble, and what it takes to build them. To date, we have built two such tools: (1) *TrackTheTracker*, a tool that answers questions about how web trackers use web histories to target ads on the web, and (2) *GObservatory*, a tool that incorporates support for a variety of questions about targeting within and across Google services. To further test our system's flexibility 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 implemented tools and designs of prior experiments. Fig.**??** shows the lines of code and experiment designs for our implemented tools.

[xxx Riley and Francis: Please replace Fig. ?? with the figure like the one I sent you. Riley: add your expt design tiles. Please make everything fit into a very compact, preferably single-row figure at the top of the page. Include a LoC table for each tool. Please try to figure out a way to fit everything as compactly as possible. Take a look at how compact the architecture figure is. That's how I want yours to be. Minimize white spaces, they are very costly!]

5.1 TrackTheTracker (TTT)

TTT builds upon Hubble to investigaate the following question: "How do web trackers target ads on a user's browsing history?" Fig.**??** shows TTT's workflow, which consists of three Hubble experiments: E_1 is a broad survey experiment to find targeted ads on interesting websites; E_2 is a smaller experiment to validate the ads hypothesized as targeted in the survey and prune out untargeted ad and websites; E_3 is a refinement experiment to determine which trackers contributed to the targeting of ads cross-domain that the survey discovered and the validation confirmed. We describe each in turn.

E_1 aims to find promising ads and targeted websites on which to conduct the tracker targeting data collection. In E_1 each input is a website to be surveyed, and there can be hundreds or thousands of these websites. Each website is assigned to a profile with probability 0.5 using the default Hubble profile generation capability; different profiles will have different sets of websites assigned, encoding different browsing histories. The `data collection worker` assigned to a profile drives a headless browser using Selenium [**?**] to visit the pages on each input site assigned to it; it records all display ads observed on each site as Hubble outputs. To detect ads on arbitrary pages, we leverage a version of AdBlock that reports any identified ad but does not disable it. 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. To detect trackers, we leverage TrackingObserver, a tracking blocker plugin []. TTT uses Hubble's default statistical correlation methods (§4) to obtain hypotheses about which output display ads target which input sites.

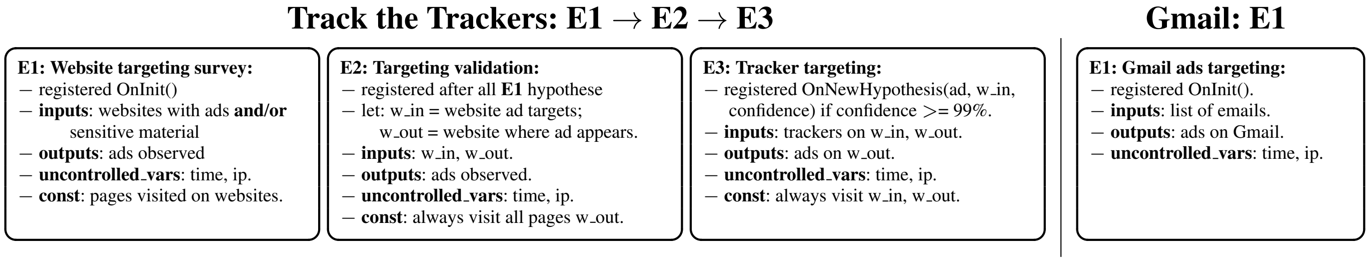
E_2 aims to validate a single targeting hypothesis from E_1 in a more controlled environment. E_2 creates a group of n profiles, half of which get assigned the targeted website. In addition, all of the profiles get assigned the websites on which the ad appeared; excluding the targeted website if the ad appeared there too. Since, our input is the presence of the targeted website in a profile, E_2 is restricted to ads that appear on at least one website other than the targeted. 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 then drives a headless browser to all sites in that group and collects all observed ads as outputs. TTT uses the Hubble's default statistical methods to determine which ads target which trackers.

§**??** shows results of using TTT in measurements of tracker-based ad targeting with 100 input websites selected based on Alexa popularity, 10 pages per website, and hundreds of trackers.

5.2 GObservatory

A number of investigative journalists have asked us a number of questions related to how various massive services, such as Google and Facebook, are using the huge piles of in-



Track the Trackers: E1 → E2 → E3

E1: Website targeting survey:
— registered OnInit()
— **inputs**: websites with ads **and/or** sensitive material
— **outputs**: ads observed
— **uncontrolled_vars**: time, ip.
— **const**: pages visited on websites.

E2: Targeting validation:
— registered after all **E1** hypothese
— let: w_in = website ad targets; w_out = website where ad appears.
— **inputs**: w_in, w_out.
— **outputs**: ads observed.
— **uncontrolled_vars**: time, ip.
— **const**: always visit all pages w_out.

E3: Tracker targeting:
— registered OnNewHypothesis(ad, w_in, confidence) if confidence $>= 99\%$.
— **inputs**: trackers on w_in, w_out.
— **outputs**: ads on w_out.
— **uncontrolled_vars**: time, ip.
— **const**: always visit w_in, w_out.

Gmail: E1

E1: Gmail ads targeting:
— registered OnInit().
— **inputs**: list of emails.
— **outputs**: ads on Gmail.
— **uncontrolled_vars**: time, ip.

Fig. 5: **Experiment Design.** TTT contains three stage experiments It conducts a large scale survey for cross website ads, a validation experiment to improve confidence on the targeted ads, and a tracker stage to attribute ads to specific trackers. Gmails is a one stage experiment.

formation they have about us to target various classes of populations. We believe that Hubble can be leveraged to investigate such questions more easily and more reliably than ever before. More broadly, we believe that there is great need for comprehensive oversight tools to monitor data flows within and across major services. As a show-case for Hubble, we have begun to build a "data observatory" for Google, *GObservatory*, which integrates experiment designs for investigating a variety of questions about Google targeting: How are Gmail ads targeted on users' inbox contents? How does Youtube target video recommendations based on a user's histories? How does Google Search target the ads? And is Youtube data used to target ads outside of Google services (e.g., for ads on the web)? Many more questions could be potentially supported, and we hope that we and others can integrate more in the future.

To answer these questions, we have built data collectors for each service, which scrape ads from Gmail, Google Search, and YouTube, and recommendations from YouTube. We also leverage TTT's scraping of ads on the web (based on AdBlock) to collect Google ads on the web that might use information from Google accounts to target (based on AdBlock). Scraping takes some engineering effort to encode (especially when no standard tool is applicable, such as AdBlock), but is conceptually simple. For each question, we have defined a Hubble experiment workflow; support for different questions was implemented at different stges of the Hubble design, hence the kinds of experiment designs we implemented for each varies. Fig.**??** and Fig.**??** show the experiment designs for the Gmail and YouTube questions. We discuss here only Gmail, the simplest (and oldest) of the designs, for which we include results in §**??**.

For Gmail, we implemented a one-level experiment (Fig.**??**) to determine which emails are used to target ads in Gmail. We wished to extend it to multiple stages, but half-way during our development, Gmail shut down its ad service. We did get to monitor Gmail ads targeted and their targeting for 30 days before it was suddenly shut down, and present surprising post-mortem results in §**??**. This (now obsolete) feature of GObservatory was inspired by XRay [**?**], however, as §**??** shows, our Hubble-based implementation is much more precise at scale and offers solid statistical guarantees of its predictions.

5.3 Enabling more accountability tools

As highlighted by a recent series of works, personalization is quickly becoming ubiquitous. Multiple transparency tools – all of those designed with a specific application in mind – have recently been proposed to reveal what affects results returned by search engine [13, 36], news and product recommendation [14], and online pricing [14, 26, 27, 32]. We now show how Hubble allows one to reproduce and often improve those tools.

Personalization of News & Search. Hubble could enhance the study of search engine personalization as done previously in [13, 36]. Since it leaves full freedom to build the data collector, one can leverage previous effort and code already available to build corpus of queries, automate browser configuration and attributes set in account, and collect search results using either PhantomJS [13] or a Chrome extension [36]. The experiment design specifies how inputs and outputs are defined for various purposes:

Inputs are typically attributes of interest to study for personalization. Here, we need to distinguish between two kinds. On the one hand, *non-exclusive* inputs (*e.g.,* the content of your search history with or without clicking on some results, web-browsing history) are collection of binary choice whether or not to include an item and those can chosen arbitrarily and randomly as before. On the other hand, while we also can formally define an input for each possible location of interest, those inputs are *exclusive* of each other (for one profile, only at most one location can be set). Hubble can handle that by specifying exclusivity between inputs and ensuring that no two exclusive inputs get placed together. The analytics using a regression model for each value naturally extends to reveal correlation even with exclusive inputs.

To define *Outputs* As done in those previous works, one can measure how personalization affect results in comparison to a gold standard, sometimes called control or organic. The data collector in that case can translate collected search results into a set of difference output triples $(r, w, +)$, $(r, w', -)$, respectively denoting that a website w entered the top-r result, and/or that w' left the top-r results. Note that those outputs are correlated (i.e. observing $(1, w, +)$ implies that $(2, w, +)$ is observed too, that is unless w has rank 2 in the gold standard). Hubble handles that gracefully as it runs correlation and analytics on different outputs entirely separately; each of these search results differences will be not only noticed and validated with statistical significance, but they will also be attributed to the input(s) responsible for the differentiated treatment. In contrast with previous studies, a tool using Hubble does not require a gold standard: alternatively, one can define output (r, w) as denoting that w is observed among the top-r results, Hubble will return as "targeted output" pairs (r, w) which can be attributed to particular inputs, denoting personalization, while those that appear broadly will be classified as "non-targeted".

So far we explained how a single experiment on search personalization can be run within Hubble. Personalization can leverage multiple experiments run subsequently as hypotheses from an early stage are formed. For instance, geographic location is considered one dominant factor [13, 36], but it occurs at different scales: country scale, which includes in particular government censorship, but also state, city, neighborhood, or even immediate proximity to landmarks and venues. Once a particular personalized search results is identified and attributed to a region, that observation can automatically trigger a higher resolution and more focused experiment to determine within that region how personalization behaves. Other exclusive inputs can also be refined in further experiment, e.g. by launching a new refined experiment to determine how personalization depends on the year of birth, once it has been established that a particular website appears more prominently to people under 35. For a large part, previous studies on search personalization did not agree even on the rough proportion of personalized results (28.6% in [36] vs. 11.7% in [13]), and it has been primarily attributed to difference in measurement methods using either synthetically generated profile or user's participation/crowdsourcing. Given that the former is cheap and relatively neutral, while the latter is more realistic and personalized but also much more expensive to run, one could imagine a specific experiment being launched to study personalization in more realistic workload once a proper hypothesis is identified.

Price Discrimination Studies. Building tools on top of Hubble to study price discrimination, as previously done in [14, 26, 27, 32] can benefit from the same techniques. While one can possibly interpret prices as a continuous variable and build specific analysis to model its variation with inputs, this is not necessary as one can also discretize the price range into $0 < p_1 < p_2 < \cdots < p_m$ and introduce outputs as tuples of the form (j, p_i) to denote that product j was offered at price above p_i. For the same reason as above, one could easily observed non-targeted inputs, which are price seen by most people, and differentiated treatment that are less frequent and can be attributed to specific inputs.

E2 ← E1 → E3:

E1: Exploratory Search expt:
— registered OnInit().
— non—exclusive **inputs**: search terms, click history, browsing history.
— exclusive **inputs**: {l_1,...,l_m} (i.e. locations),
{f,m,o} (i.e. gender), {15—35;36—50;51—75} (i.e. age)
— **outputs**: (r,w,+), (r,w,—) (**or** (r,w) **if** no gold standard)
where r is rank **in** [1;10] **and** w is a URL
— **uncontrolled_vars**: time, ip.
— **const**: data center reached

E2: zoom experiment:
— registered OnNewHypothesis(ad, s_in,
confidence) if confidence >= 99%.
— let: s_in = terms with personalized results;
l_i = relevant region.
— **inputs**: {l'_1,...,l'_m} (partition of l_i)
— **outputs**: (r,w).
— **uncontrolled_vars**: time, ip.
— **const**: search s_in **and** all other **inputs**.

E3: crowd experiment:
— registered OnNewHypothesis(ad, s_in,
confidence) if confidence >= 99%.
— let: s_in = term personalized;
— **inputs**: all above.
— **outputs**: all above.
— **uncontrolled_vars**: time, ip, AMT history profile.
— **const**: search s_in, result collected through crowd.

Fig. 6: **Possible Experiment Design for search.** E1 is a large-scale survey experiment to find which search are targeted. E2 brings a more refined picture of the impact of location once a term and a region is identified. E3 validates and compares for specific search term with additional more realistic profiles.

Workload	Profiles	Inputs	Outputs	Observations
gmail-small	101	19	2359	784644
gmail-large	119	299	5574	1016323
gmail-redun	101	45	3396	1405786
youtube-tiny	41	32	186	672
youtube-small	45	64	308	1408
amzn-tiny	21	30	1456	352
amzn-small	51	61	2593	1664
website-large-a	200	84	46237	908938
website-large-b	200	83	140701	2402416

Table 1: **Workloads used to evaluate Hubble** Shows metrics about the workloads with which we evaluated Hubble.

We believe that, as shown here for search and price discrimination, transparency tools previously proposed and Hubble are natural complement. Hubble can leverage them directly as data collector, but also enhance them by (1) orchestrating an experiment in which multiple inputs vary, without requiring that a separate comparison is run for all possible values of each attribute, (2) greatly simplify the design of adaptive multi-resolution tools, which provides a level of details that is otherwise prohibitive to obtain. Without going in the same amount of details, those two advantages also allows to enhance previous tools to study personalization of ads in different contexts [3, 7, 9, 22, 25, 33].

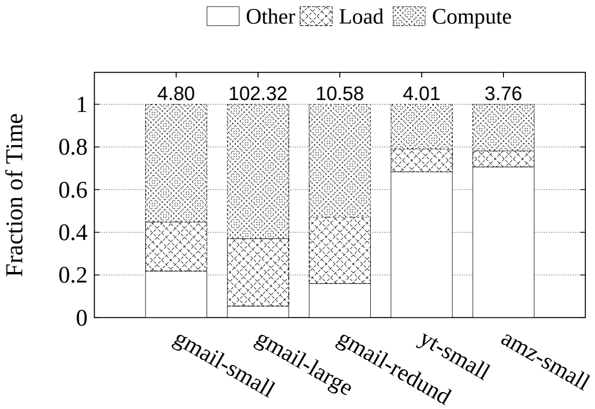


Fig. 7: **Analysis time breakdown** Displays what fraction of the total analysis the system spends computing the actual statistics, loading the data to be used, and other overhead. The average analysis time in seconds is displayed above each bar.

Workload	Mean	Median	Min	Max
gmail-small	4.80	4.93	0.56	11.06
gmail-large	102.32	108.57	21.00	233.83
gmail-redund	10.58	10.59	1.56	36.77
yt-small	4.01	4.11	1.01	9.13
amz-small	3.76	3.86	0.75	13.05

Table 2: **Workload Analysis Time.** Displays the mean, median, minimum, and maximum time in seconds to prepare a hypothesis for an ouput.

6. Hubble Evaluation

6.1 Methodology

We evaluated Hubble on nine datasets from Gmail, YouTube, and Amazon that range in both size and diversity and are presented in Table 1. The Gmail, YouTube and Amazon data sets are shared with the previous XRay project and are used as a point of common comparison with previous work and both of the website data sets were collected as part of TTT. In all of the experiments except for gmail-redun, each input is distinct from all other inputs. The gmail-redund experiment has groupings of 2, 3, or 4 inputs that were likely to attract similar outputs.

6.2 Performance

We evaluated two facets of Hubble performance, analysis time and scalability. Both will be relevant to Hubble name as scalability will limit the deployment size of any Hubble application and we expect users will take advantage of the provided statistical analysis functionality.

Analysis Time The measured the amount of time required for the Hubble analytics module to provide hypotheses for the outputs in different data sets. In addition to the raw time required we also instrumented the analytics worker to determine where it spent analysis time. Table 2 presents metrics about analysis times for a subset of the experiments. The time to compute a hypothesis scales roughly with the number of inputs and is independent of the number of observations. Note that gmail-large and gmail-redund both have a large number of observations but gmail-large contains many more inputs and took much longer to return hypotheses. Figure 7 presents the average time breakdown for each of the experiments evaluated. The instrumentation on the analytics workers revealed that compute time dominates the larger experiments followed by load time which includes loading the data from the database and passing it to R. In the two experiments with the shortest analysis times the workers spend more than half of the time on other bookkeeping duties.

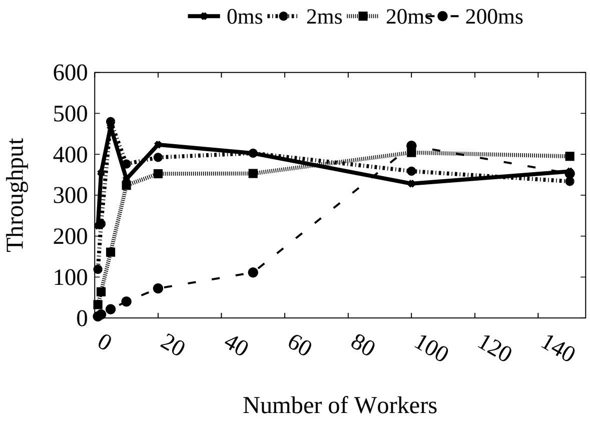
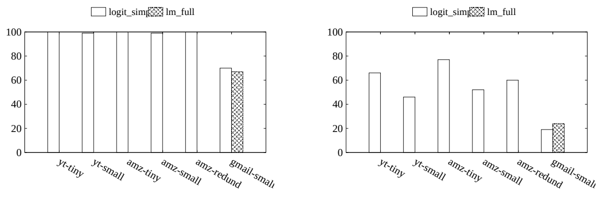


Fig. 8: **Hubble Hypothesis Throughput.** This figure is currently too small. I'll make it bigger later. Plots the number of workers

Scalabiblity 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 8 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. The *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 experiments with 0 and 2 ms latency achieved 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.

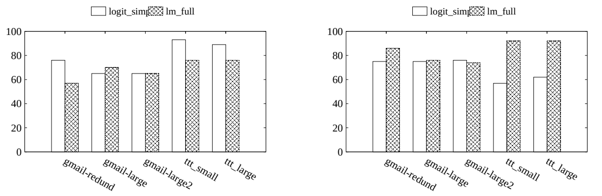
6.3 Accuracy on Ground Truth

The first question we seek to answer to evaluate Hubble's algorithms is their accuracy (precision and recall) on datasets from small scale experiments on services that are fairly simple and deterministic. Even though these datasets are not the most interesting, they are interesting because the services provide ground truth for the targeting results. Amazon and YouTube recommendations display a link that explains why



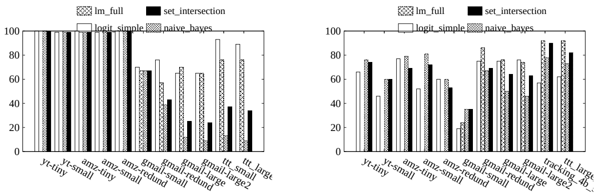
(a) **Precision with ground truth** (b) **Recall with ground truth**

Fig. 9: **Precision and Recall for datasets with ground truth.** For datasets from YouTube and Amazon the ground truth is provided by the service. The last dataset comes from Gmail with ground truth from manual labelling.



(a) **Precision from p-values** (b) **Recall from p-values**

Fig. 10: **Conservative precision and recall for datasets without ground truth.** The datasets come from Gmail and TTT experiments, and include a large number of inputs with redundancy.



(a) **Precision** (b) **Recall**

Fig. 11: **Comparison with XRay [?].** Comparison of XRay's and Hubble's algorithms on datasets from 4 different experiments. XXX Mathias: please organize the bars as follows: amzn-small gmail-small amzn-redund gmail-redund gmail-large1 gmail-large2. Separate the small-redund-large classes with some space. For bars, please show one single bar for Hubble (the "default" algo/model for each dataset, i.e., logit_simple for Amzn, lm_full for Gmail), plus bayes and set intersection. Drop tiny datasets datasets. These should add space to add the TTT datasets.

this recommendation is shown. The Gmail dataset is from an experiment with only 20 emails, with well differentiated topic. As Gmail does not provide ground truth, we had to manually label the ads. Even on this simple case, the labelling is very hard, and most probably contains many mistakes.

Fig. 9 shows precision and recall of the two main algorithms on multiple datasets. We can make two main observations. First, on the datasets from YouTube and Amazon with service provided ground truth, Hubble is very close to 100% precision. Almost all guesses returned by Hubble is confirmed by the ground truth. This is an important feature: since we explore so many hypothesis, guesses must be accurate. The recall is decent for the small number of profiles of these experiments, from 40% to 80%. Recall is also higher on experiments with less inputs, which confirms the role of the small number of profiles.

Second, on Gmail, which is less deterministic, precision and recall is not as good. However the results are not as reliable due to manual labelling which is very difficult and error prone. After running the algorithms, we could see multiple cases that seem to be real targeting but that we labelled untargeted, and vice versa. As relabelling after the fact is probe to bias we left our first impression, hence a lower precision and recall.

Hubble has a pretty good precision and recall on small datasets with ground truth, on different services. On more complex experiments, the algorithm's are very hard to evaluate: even on very simple Gmail experiments, labelling is complex and error prone. We thus require a different evaluation methodology that we discuss next.

6.4 Tool Testing Evaluation

In order to evaluate and benchmark algorithms on more complex datasets, with many inputs and topic redundancy between these inputs, we developed a technique based on common practices from statistics and described in § 4.5. We next describe the characteristics of this method and its performances.

The first thing to note is that this method does not measure recall. If we had a method capable of finding the targeting we missed, then we would use this method as our main algorithm. We still display a recall metrics for benchmarks, which is the proportion of targeting detected by a given algorithm, out of all the targeting detected by any algorithm in the benchmark. This information is useful to compare different algorithm: if an algorithm has a great precision but finds only a tiny proportion of what other algorithms find, it may not be valuable.

Another important feature described in § 4.5 is that this evaluation method is conservative. We could verify that on the datasets where ground truth is available. Our confidence metrics returned a precision around 85% for YouTube datasets, and 80% for the Amazon datasets. Compared to the almost 99% or 100% precision of the ground truth, it shows that our technique tends to underestimate precision. For on Gmail experiment, it actually predicts a better precision (about 10% higher). After looking at cases where both disagreed, we concluded that p-values seemed much better than our manual labels.

We also noted qualitatively that low p-values are a very good indicator of targeting. We looked at 100 ads with low p-value (< 0.05) and found only one instance where we could not explain the targeting. We also informally looked at hundreds of ads when searching for interesting targeting, and our experience is that very low p-values are almost always associated to meaningful targeting.

This evaluation method thus gives a solid lower bound on precision, and is most useful to compare different algorithms on complex datasets with many inputs and topic redundancy. Now that we validated the evaluation tool, we evaluate the Hubble at scale.

6.5 Accuracy at Scale

We now use our statistical analysis on larger, more complex datasets from Gmail and TTT experiments. We can see on Fig. 10 that despite our conservative evaluation techniques, our algorithms reach a good precision of more that 60%. Both algorithms also find 80% of the targeting. We can also see that on TTT experiments the Linear regression on the full data model has a lower precision than the Logistic simple (75% instead of 90%) but finds a bigger fraction of all targeting found (92% instead of 60%).

Note that Hubble always return the p-value to the user, who is then able to consider hypothesis only below a given threshold. Hence the absolute numbers returned by the evaluation are a bit hard to interpret. This is why this evaluation technique is best suited for comparison, as we show next.

6.6 Comparison with XRay

The closest related work we are aware of is XRay [**?**], which uses two types of algorithms to detect targeting: a Naive Bayes algorithm, and a Set Intersection algorithm. However the evaluation lacks large, complex experiments, due to lack of ground truth and labelling difficulty. We thus implemented their algorithms to use our benchmarking technique and compare with our statistics based approach. In order to make the fairest possible comparison, we also run their algorithms through our filtering mechanism that makes sure guesses are good predictors of ads on the training set and greatly increases precision.

Fig. 11 shows the results of the benchmark. We can distinguish two main trends. First, XRay's algorithms are a little better on simple datasets without redundancy in the inputs. The precisions are similar for all algorithms, but the Bayesian and Set Intersection have a better recall: +20% on the Amazon dataset and +12% on a simple Gmail dataset.

Second, when experiments become more complicated, with redundancy in inputs that trigger targeting on multiple email, Hubble's statistical algorithms become much better. For instance, Hubble's Logistic algorithm on with the Simple data model described in § 4.1 has +40% precision compared to XRay's algorithms on a medium size Gmail dataset with redundant emails, and up to +50% on larger inputs sizes with redundancy. Hubble's algorithms also find a larger portion of all the ads found by any algorithm. Both find almost 80% of them, while XRay's algorithms find 50% and 60% respectively, despite their lower precision.

At large scale, it is virtually impossible to construct datasets of inputs without redundancy. Hubble statistical approach is thus more adapted to answer complex questions in real studies.

7. 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.

7.1 Targeting of Gmail Ads

As a first example of personal data use, we turn to Gmail which, until November last year, offered personalized advertisments tailored to a user's email content. We selectively placed 300 emails containing single keywords or short phrases to encode a variety of topics, including commercial products (e.g. TV, cars, clothes) and sensitive topics (e.g., religion, sexual orientation, health). Our goal was to study (1) various aspects related to targeted advertisements, such as how frequent they are and how often they appear in the context of the email being targeted (a more obvious form of targeting) versus in the context of another email (a more obscure form of targeting) and (2) whether advertisers are able to target their ads to sensitive situations or special groups defined by race, religion etc. We ran the targeted ad collection for **[xxx 30]** days, from Oct. XXX to Nov. XXX, 2014 when Google shut down the Gmail ad service.

We collected more than 9.1M impressions created collectively by more than 44K unique ads. As expected, the distribution of impressions per ad is skewed: the median ads were observed 9 times in the experiment, while the top 25/5/1% of ads were observed 62/853/2,739 times. We classify an ad as *targeted* if its statistical confidence is high (p-value$<$ 0.05). In our experiment, 5,513 unique ads (12% of all) were classified as targeted, and collectively they are responsible from 43% of all impressions. While we observe that ads classified as targeted are seen more often (352 impressions for the median targeted ads), this could be an artefact of our measurement as most ads seen only occasionally present insufficient evidence and are by default classified as "non-targeted."

[xxx (RG) I don't get this paragraph at all. I removed the statistical method sentence b/c I thought it came out of the blue. My guess is that by now no one is questioning this, why point this out now with such weak evidence? In any case, please try to fix entire paragraph, it's very confusing.] Second, we observe that a non-negligible but small fraction of impressions (7%) for targeted ads appear outside the scope we infer. Interestingly, of the 94% targeted impressions that we correctly predict within scope, a little less than half (49%) appear in the context of an email in scope. This proves that behavioral targeting - where the advertising message is personalized but not related to the current content you are shown - is frequent enough to be the most common form of targeted advertising in this case.

[xxx The abstract mentions a violation of privacy statement. I'd like to keep that as it's an attention drawer (maybe we'll replace the violation word with something weaker. But please add a quote to the privacy statement here.] Finally, we were able to statistically confirm that various personal information, including those related to sensitive topics, can be used by advertiser to deliver their message. A brief summary of ads and associated scope infered can be found in Table **??**. Information about a user's health, race, religious affiliation or religious interest, sexual orientation, or difficult financial situation, all generate targeted advertisment. As Google explicitly proscribed such practice in its term of service for gmail ads, this proves the importance of transparency tools to remain vigilant against ill-intentioned advertisers. One example illustrates how opaque targeting can subtly circumvent current regulation: most ad systems forbids commercial message explicitly encouraging the consumption of recreational drugs, especially if that message is addressed at teenagers, but should an ad promoting equipment for indoor growth of plants be allowed to target email mentioning "cannabis", while its name suggests that it caters to a student audience? While we think those examples are proof of necessary investigation, we would like to point out that those violations are needles in a haystack (about 50 unique ads with debatable scope, less than 0.02% of the ones we observed). Several topics we have included (e.g., fatal diseases and loss, other religious affiliation) generated not a single ad classified as targeted.

7.2 Targeting in Google Search Ads

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

Main results:

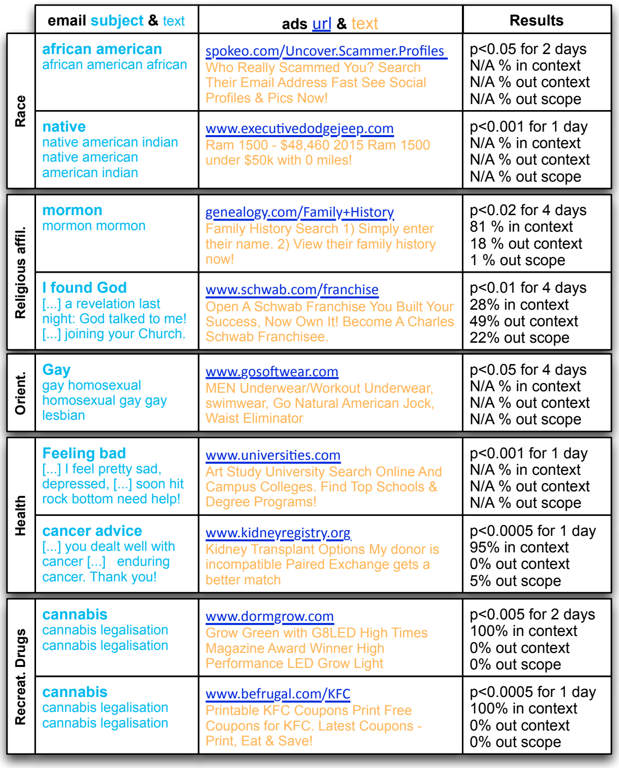
- We have found no evidence of behavioral targeting on sensitive topics.

- 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.

7.3 Targeting of Google Ads on the Web

[xxx 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.



	email subject & text	ads url & text	Results
Race	**african american** african american african	spokeo.com/Uncover.Scammer.Profiles Who Really Scammed You? Search Their Email Address Fast See Social Profiles & Pics Now!	p<0.05 for 2 days N/A % in context N/A % out context N/A % out scope
	native native american indian native american american indian	www.executivedodgejeep.com Ram 1500 - $48,460 2015 Ram 1500 under $50k with 0 miles!	p<0.001 for 1 day N/A % in context N/A % out context N/A % out scope
Religious affil.	**mormon** mormon mormon	genealogy.com/Family+History Family History Search 1) Simply enter their name. 2) View their family history now!	p<0.02 for 4 days 81 % in context 18 % out context 1 % out scope
	I found God [...] a revelation last night: God talked to me! [...] joining your Church.	www.schwab.com/franchise Open A Schwab Franchise You Built Your Success, Now Own It! Become A Charles Schwab Franchisee.	p<0.01 for 4 days 28% in context 49% out context 22% out scope
Orient.	**Gay** gay homosexual homosexual gay gay lesbian	www.gosoftwear.com MEN Underwear/Workout Underwear, swimwear, Go Natural American Jock, Waist Eliminator	p<0.05 for 4 days N/A % in context N/A % out context N/A % out scope
Health	**Feeling bad** [...] I feel pretty sad, depressed, [...] soon hit rock bottom need help!	www.universities.com Art Study University Search Online And Campus Colleges. Find Top Schools & Degree Programs!	p<0.001 for 1 day N/A % in context N/A % out context N/A % out scope
	cancer advice [...] you dealt well with cancer [...] enduring cancer. Thank you!	www.kidneyregistry.org Kidney Transplant Options My donor is incompatible Paired Exchange gets a better match	p<0.0005 for 1 day 95% in context 0% out context 5% out scope
Recreat. Drugs	**cannabis** cannabis legalisation cannabis legalisation	www.dormgrow.com Grow Green with G8LED High Times Magazine Award Winner High Performance LED Grow Light	p<0.005 for 2 days 100% in context 0% out context 0% out scope
	cannabis cannabis legalisation cannabis legalisation	www.befrugal.com/KFC Printable KFC Coupons Print Free Coupons for KFC. Latest Coupons - Print, Eat & Save!	p<0.0005 for 1 day 100% in context 0% out context 0% out scope

Fig. 12: Personalization of ads within the Gmail service

7.4 Tracker-Based Ad Targeting

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

8. 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 principled 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. [xxx Augustin.] Price discrimination studies, news/search bubble studies, ad targeting studies, etc.

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

Further afield, much prior work has focused on *data collection transparency*, which seeks to reveal what information services *collect* about the users [1, 17–20, 24, 29, 30]. The work is complementary to ours, which seeks to reveal what web services *do* with the data they collect.

Algorithmic Transparency. [**xxx 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).

Learning Theory. [**xxx Augustin.**] Whatever 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.

Related Statistical Methods. Our methods for statistical experimental design and analysis draw from the subjects of *compressed sensing* [8, 10], *sparse linear regression* [5, 31], and *sparse signal recovery* [11, 12]. 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 hypothesized 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., [**xxx e-mail types**]). 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 [21, 35] or in statistical power [38]. 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 [37] and GraphLab [23]. Second, we may use *adaptive methods* [15, 16] 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. [**xxx Augustin?**]

9. Conclusions

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] Barford, P., Canadi, I., Krushevskaja, D., Ma, Q., and Muthukrishnan, S. Adscape: Harvesting and Analyzing Online Display Ads. *WWW '14: Proceedings of the 23nd international conference on World Wide Web* (Apr. 2014).

[4] Benjamini, Y., and Yekutieli, D. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics* (2001), 1165–1188.

[5] Bickel, P. J., Ritov, Y., and Tsybakov, A. B. Simultaneous analysis of lasso and dantzig selector. *Ann. Statist. 37*, 4 (08 2009), 1705–1732.

[6] Bodik, P., Goldszmidt, M., Fox, A., Woodard, D. B., and Andersen, H. Fingerprinting the datacenter: Automated classification of performance crises. In *Proceedings of the 5th European Conference on Computer Systems* (New York, NY, USA, 2010), EuroSys '10, ACM, pp. 111–124.

[7] Book, T., and Wallach, D. S. An Empirical Study of Mobile Ad Targeting. *arXiv.org* (2015).

[8] 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.

[9] Datta, A., Tschantz, M. C., and Datta, A. Automated Experiments on Ad Privacy Settings: A Tale of Opacity, Choice, and Discrimination. Tech. rep., Aug. 2014.

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

[11] Gilbert, A., and Indyk, P. Sparse recovery using sparse matrices. *Proceedings of the IEEE 98*, 6 (June 2010), 937–947.

[12] 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.

[13] 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.

[14] Hannak, A., Soeller, G., Lazer, D., Mislove, A., and Wilson, C. Measuring Price Discrimination and Steering on E-commerce Web Sites. In *IMC '14: Proceedings of the 2014 Conference on Internet Measurement Conference* (Nov. 2014), ACM Request Permissions.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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).

[21] 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.

[22] Liu, B., Sheth, A., Weinsberg, U., Chandrashekar, J., and Govindan, R. AdReveal: improving transparency into online targeted advertising. In *HotNets-XII: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks* (Nov. 2013), ACM Request Permissions.

[23] 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.

[24] 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.

[25] Metwalley, H., Traverso, S., Mellia, M., Miskovic, S., and Baldi, M. CrowdSurf: Empowering Informed Choices in the Web. *12th Italian Networking Workshop* (Feb. 2015).

[26] 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.

[27] Mikians, J., Gyarmati, L., Erramilli, V., and Laoutaris, N. Crowd-assisted Search for Price Discrimination in E-Commerce: First results. *arXiv.org* (July 2013).

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

[29] 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).

[30] 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.

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

[32] Vissers, T., Nikiforakis, N., Bielova, N., and Joosen, W. Crying Wolf?On the Price Discrimination of Online Airline Tickets. *Proceedings of the 7th Hot Topics in Privacy Enhancing Technologies (HotPETs 2014)* (June 2014), 1–12.

[33] Wills, C. E., and Tatar, C. Understanding What They Do with What They Know. *WPES '12: Proceedings of the 12th annual ACM workshop on Privacy in the electronic society* (2012).

[34] Wu, T. T., Chen, Y. F., Hastie, T., Sobel, E., and Lange, K. Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics 25*, 6 (2009), 714–721.

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

[36] Xing, X., Meng, W., Doozan, D., Feamster, N., Lee, W., and Snoeren, A. C. Exposing Inconsistent Web Search Results with Bobble. In *PAM '14: Proceedings of the Passive and Active Measurements Conference* (2014).

[37] 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.

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