CS208: Applied Privacy for Data Science
The Local Model: Implementations

James Honaker & Salil Vadhan

School of Engineering & Applied Sciences
Harvard University

April 5, 2019

CRCS Center for Research on Computation and Society
at Harvard John A. Paulson School of Engineering and Applied Sciences

Example: Randomized response

- Each person has data $x_i \in \mathcal{X}$
  - Analyst wants to know average of $f: \mathcal{X} \to \{-1,1\}$ over $\boldsymbol{x}$
  - E.g. "what is the fraction of diabetics"?
- Randomization operator takes $y \in \{-1,1\}$:

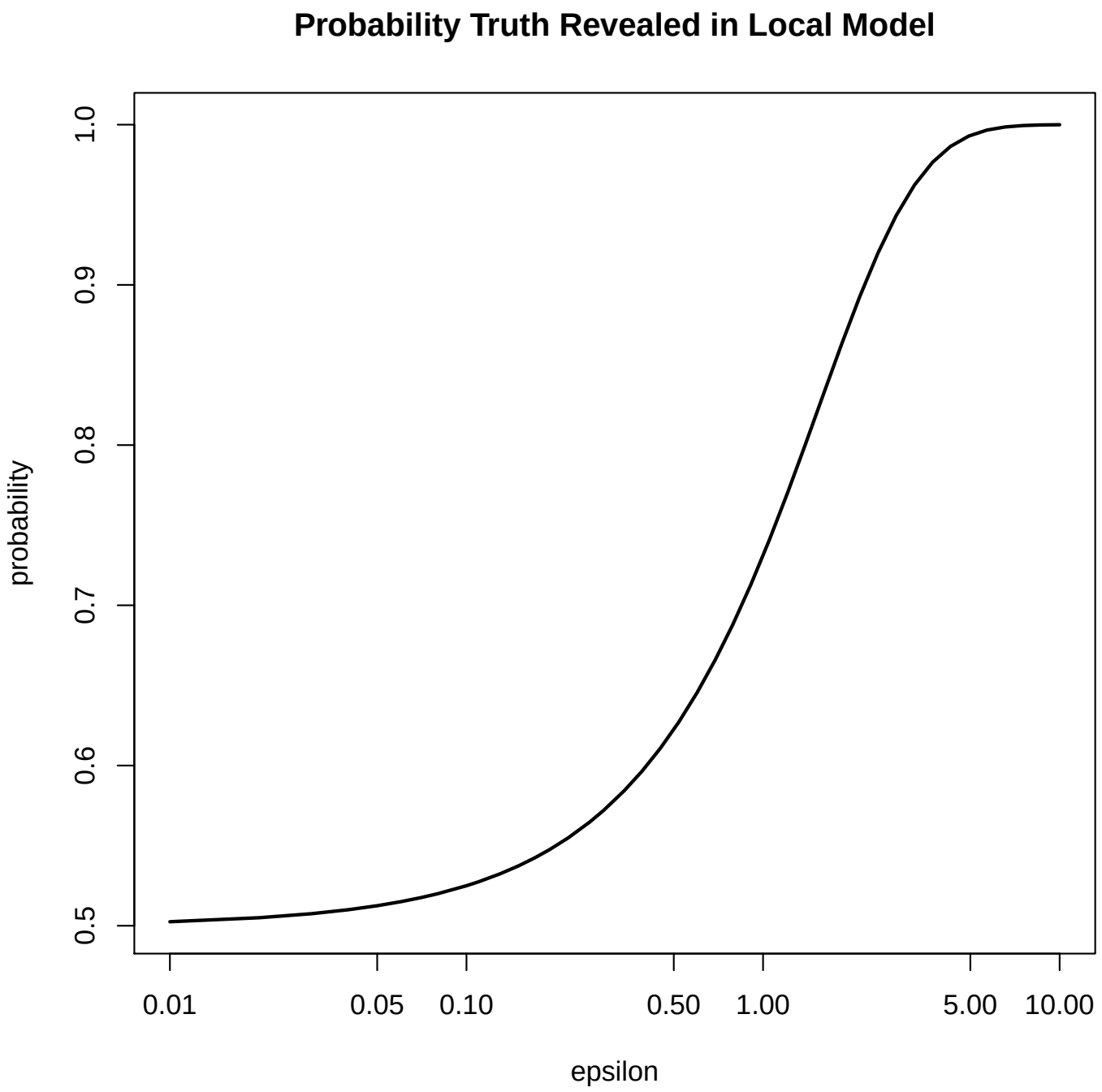
$$Q(y) = \begin{cases} +y & w.p. \frac{e^{\epsilon}}{e^{\epsilon}+1} \\ -y & w.p. \frac{1}{e^{\epsilon}+1} \end{cases}$$

ratio is e^{ϵ}

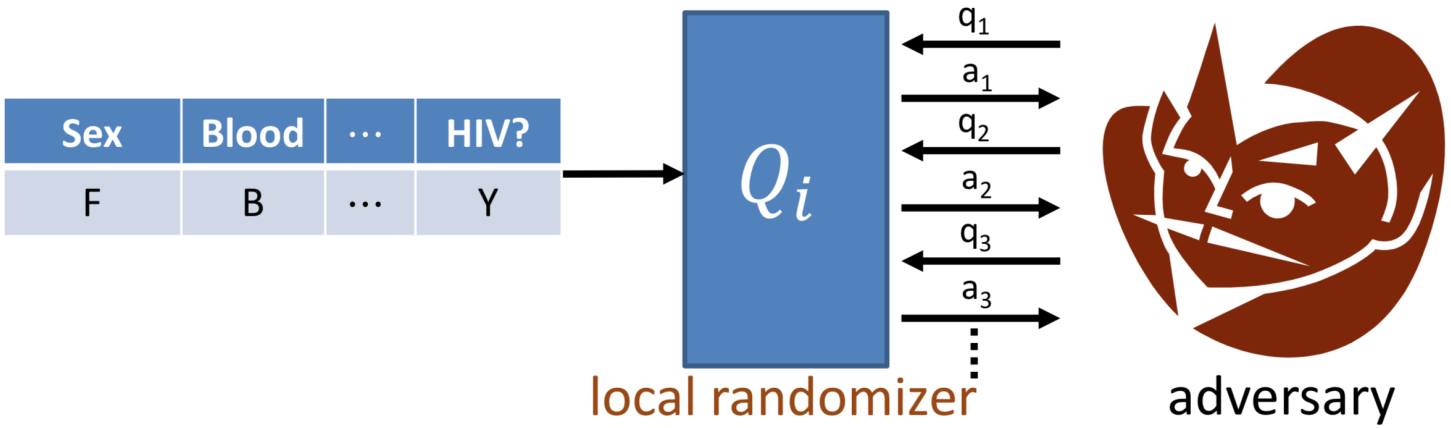
- Observe:
  - If $c_{\epsilon} = \frac{e^{\epsilon}+1}{e^{\epsilon}-1}$, then $\boldsymbol{E}(c_{\epsilon} \cdot Q(y)) = y$
- How can we estimate a proportion?
  - $A(x_1, \ldots, x_n) = \frac{1}{n}\sum_i c_{\epsilon} \cdot Q(f(x_i))$
- **Proposition:** $\boldsymbol{E}\left|A(\boldsymbol{x}) - \frac{1}{n}\sum_i f(x_i)\right| \leq \frac{c_{\epsilon}}{2\sqrt{n}} \approx \frac{1}{\epsilon\sqrt{n}}$.

Contrast with $\frac{1}{n\epsilon}$ in central model (via Laplace noise)

Slide from Adam Smith


Probability Truth Revealed in Local Model
probability
1.0
0.9
0.8
0.7
0.6
0.5
0.01
0.05
0.10
0.50
1.00
5.00
10.00
epsilon

Local DP



Require: for all i, x_i, x_i' ~~differing on one row,~~ all strategies A

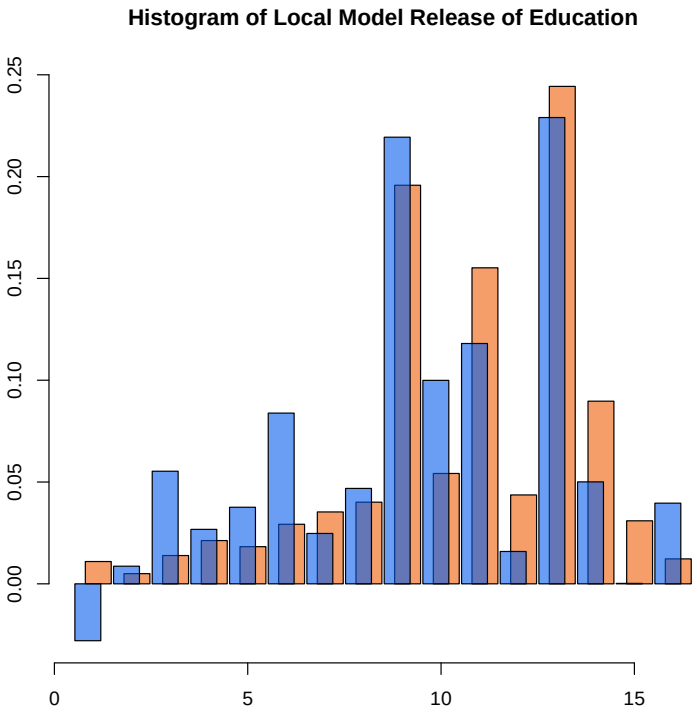
$$\Pr[A \text{ outputs YES after interacting w}/Q_i(x_i)]$$
$$\leq e^{\varepsilon} \cdot \Pr\left[A \text{ outputs YES after interacting w}/Q_i(x_i')\right] + \delta$$

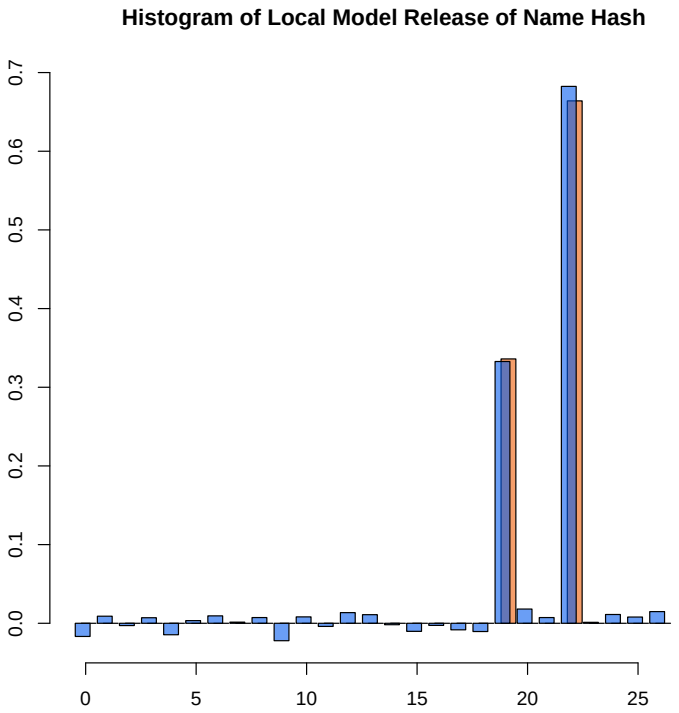
Recap: DP Histograms

- Local randomizer $Q(x_i)$ for $x_i \in \{1, \dots, D\}$
  1. Construct "1-hot" vector $e_{x_i} = (0,0, \dots, 0,1,0, \dots, 0) \in \{0,1\}^D$.
  2. Apply $(\varepsilon/2)$-DP RR to each coordinate to get $y_i \in \{0,1\}^D$:

$$y_i[j] = \begin{cases} e_{x_i}[j] & \text{w.p. } e^{\varepsilon/2}/(1+e^{\varepsilon/2}) \\ 1 - e_{x_i}[j] & \text{o.w.} \end{cases}$$

  3. Send y_i to server.
- Server uses $(y_1, \dots, y_n)$ to estimate histogram $f = \sum_i e_{x_i}$.
- Error per bin $O(\sqrt{n}/\varepsilon)$.


Histogram of Local Model Release of Education
0.00
0.05
0.10
0.15
0.20
0.25
0
5
10
15


Histogram of Local Model Release of Name Hash
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
5
10
15
20
25

Discovering Unknown Values (Apple)

- Method described so far requires server to decide on a small set values $v_1, \ldots, v_s$ to estimate frequencies of.

- Goal: find unanticipated frequent values v

- Idea: reconstruct v one symbol at a time
  - do RR on $h(x_i)||x_i[j]$ for each bitposition $j = 1, \ldots, \log D$.
  - $\hat{f}[w||\sigma_j]$ is large $\Rightarrow$ there probably is a frequently occurring value $v \in \{1, \ldots, D\}$ such that $h(v) = w$ and $v[j] = \sigma_j$.
  - $\hat{f}[w||\sigma_1], \ldots, \hat{f}[w||\sigma_{\log D}]$ large $\Rightarrow v = \sigma_1 \cdots \sigma_{\log D}$

Discovering Unknown Values (Apple)

Algorithm 2 Client-Side $\mathcal{A}_{\mathsf{client-CMS}}$

Require: Data element: $d \in \mathcal{D}$; ϵ, $\mathcal{H}$.

1. Sample j uniformly at random from $[k]$.
2. Initialize a vector $\boldsymbol{v} \leftarrow -\mathbf{1} \in \mathbb{R}^m$.
3. Set $v_{h_j(d)} \leftarrow 1$.
4. Sample $\boldsymbol{b} \in \{-1, +1\}^m$, where each b_ℓ is i.i.d. where $\Pr[b_\ell = +1] = \frac{e^{\epsilon/2}}{e^{\epsilon/2}+1}$.
5. $\tilde{\boldsymbol{v}} \leftarrow (v_1 b_1, \cdots, v_m b_m)$.
6. **return** $\tilde{\boldsymbol{v}}$, index j.

Algorithm 9 Client-Side $\mathcal{A}_{\mathsf{client-SFP}}$

Require: String: $\boldsymbol{s} \in \mathcal{D}$; privacy parameters: (ϵ, ϵ'), hash functions $(\mathcal{H}, \mathcal{H}')$, and h with output size 256.

1. Sample ℓ uniformly at random from $\{1, 3, 5, 7, 9\}$.
2. Set $\boldsymbol{r} \leftarrow h(\boldsymbol{s}) \| \boldsymbol{s}[\ell : \ell + 1]$.
3. **return** $(\mathcal{A}_{\mathsf{client-CMS}}(\boldsymbol{r}, \epsilon', \mathcal{H}'), \mathcal{A}_{\mathsf{client-CMS}}(\boldsymbol{s}, \epsilon, \mathcal{H}), \ell)$.

Discovering Unknown Values (Apple)

```
Actual Names:
setosa
 versicolor
 virginica
      temp.l
temp.b   1   2   3   4   5   6   7   8   9  10
         0   0   0   0   0   0 264 236 220 316
     a   0   0   0   0   0 247   0   0  89   0
     c   0   0   0   0   0  79   0  98   0   0
     e   0 354   0   0   0   0   0   0   0   0
     g   0   0   0  99   0   0   0   0   0   0
     i   0  72   0   0 188   0  80   0   0   0
     l   0   0   0   0   0   0   0 105   0   0
     n   0   0   0   0   0  82   0   0   0   0
     o   0   0   0 219   0   0 101   0  87   0
     r   0   0 190   0   0   0   0   0   0  98
     s 244   0   0  81 250   0   0   0   0   0
     t   0   0 243   0   0   0   0   0   0   0
     v 185   0   0   0   0   0   0   0   0   0
 [1] "s" "e" "t" "o" "s" "a" ""  ""  ""  ""
      temp.l
temp.b   1   2   3   4   5   6   7   8   9  10
         0   0   0   0   0   0 104  94  89 324
     a   0   0   0   0   0 125   0   0 228   0
     c   0   0   0   0   0 251   0 254   0   0
     e   0 335   0   0   0   0   0   0   0   0
     g   0   0   0 238   0   0   0   0   0   0
     i   0 243   0   0 493   0 231   0   0   0
     l   0   0   0   0   0   0   0 260   0   0
     n   0   0   0   0   0 234   0   0   0   0
     o   0   0   0  86   0   0 258   0 260   0
     r   0   0 488   0   0   0   0   0   0 234
     s 100   0   0 222  92   0   0   0   0   0
     t   0   0  81   0   0   0   0   0   0   0
     v 519   0   0   0   0   0   0   0   0   0
 [1] "v" "e" "r" "g" "i" "c" "o" "l" "o" ""
```

RAPPOR Setup

1. Change directory to where you want to install the RAPPOR repository, then:
   `$ git clone https://github.com/google/rappor.git`

2. Install R packages:
   ```
   library.list <- c("glmnet", "limSolve", "ggplot2", "optparse", "shiny")
   install.packages(library.list, repos="https://cran.cnr.berkeley.edu/")
   ```

3. Change directory to `$ cd rappor/apps/rappor-sim`
   `$ ./run_app.sh`

4. Point browser to: `http://localhost:6788/`

5. Also worth looking at if you are exploring this codebase:
   `cd rappor/apps/rappor-analysis $ ./run_app.sh`

RAPPOR Simulation

RAPPOR | Privacy | Population | Decoding

Probability of reporting noise (p):

0.01 — 0.5 — 0.99

0.01 0.11 0.21 0.31 0.41 0.51 0.61 0.71 0.81 0.91 0.99

Probability of reporting signal (q):

0.01 — 0.75 — 0.99

0.01 0.11 0.21 0.31 0.41 0.51 0.61 0.71 0.81 0.91 0.99

Probability of lies (f):

0 — 0.5 — 0.99

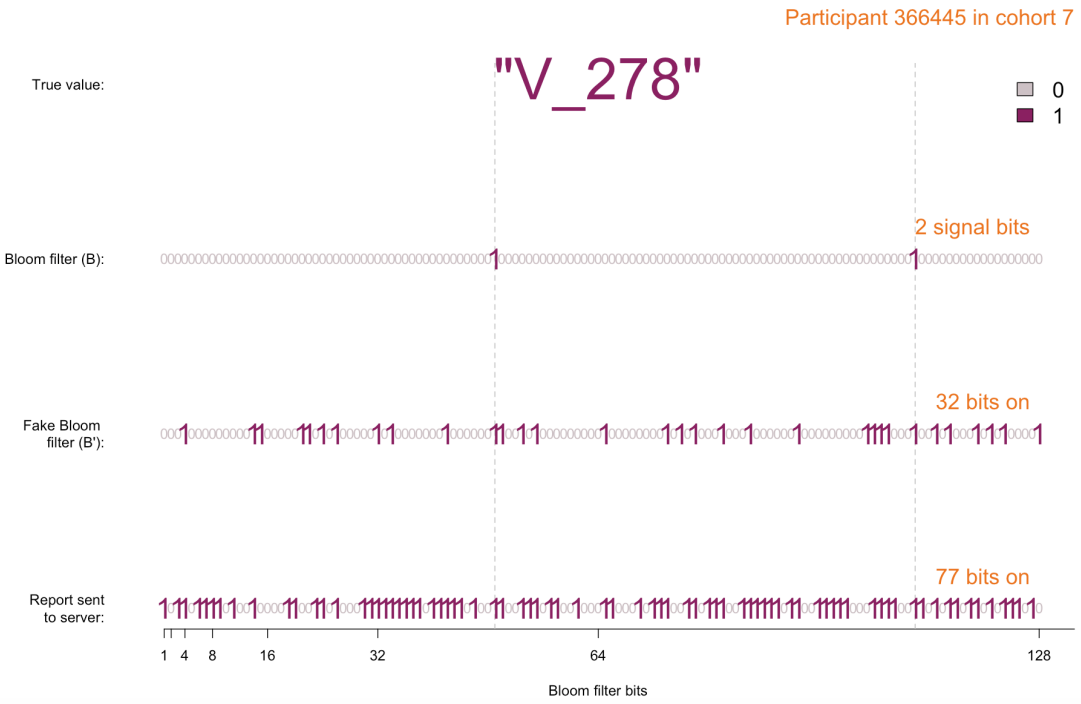
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 0.99

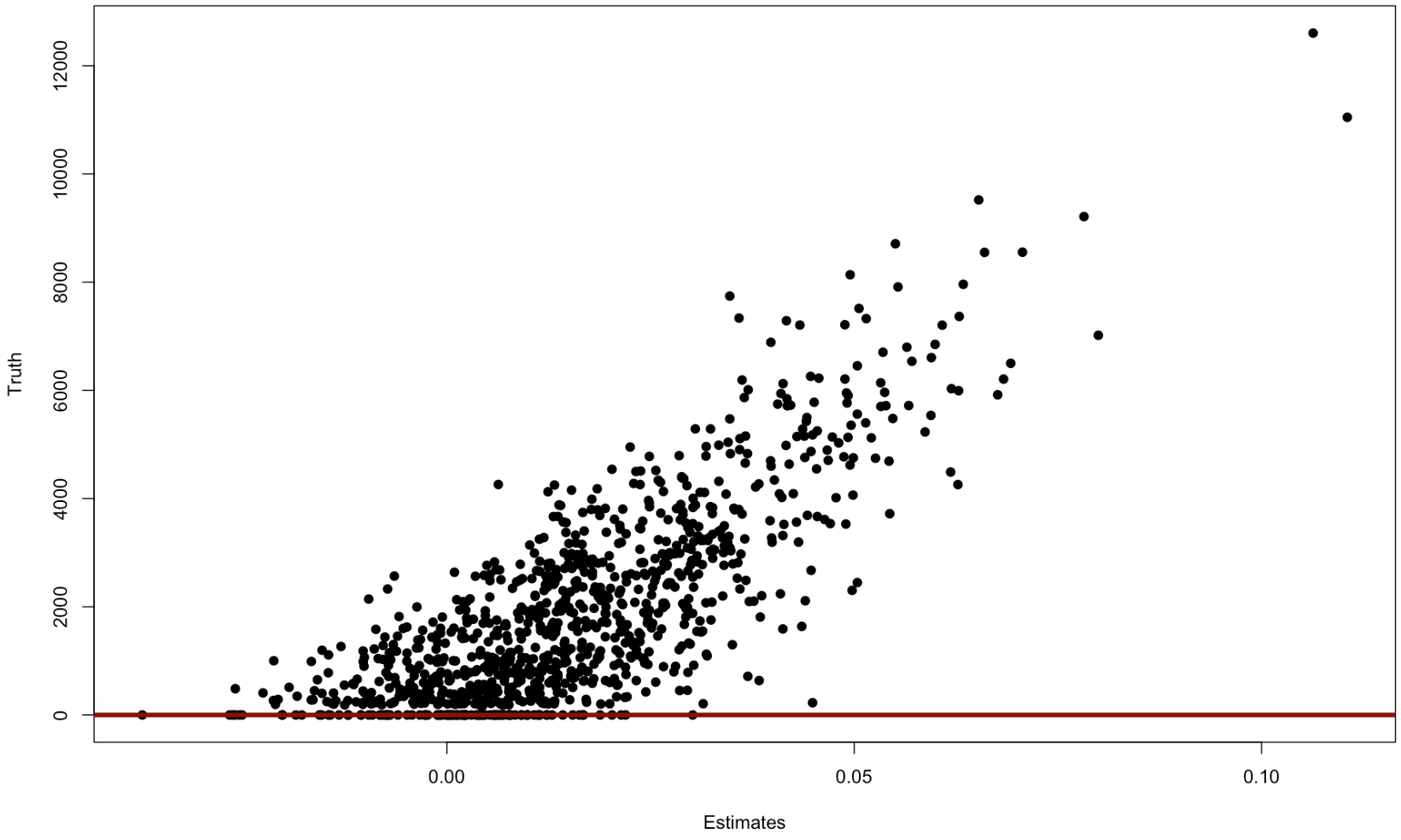
Effective p	0.5625
Effective q	0.6875
exp(e_1)	2.9279
e_1	1.0743
exp(e_inf)	81.0000
e_inf	4.3944
Detection frequency	0.0065

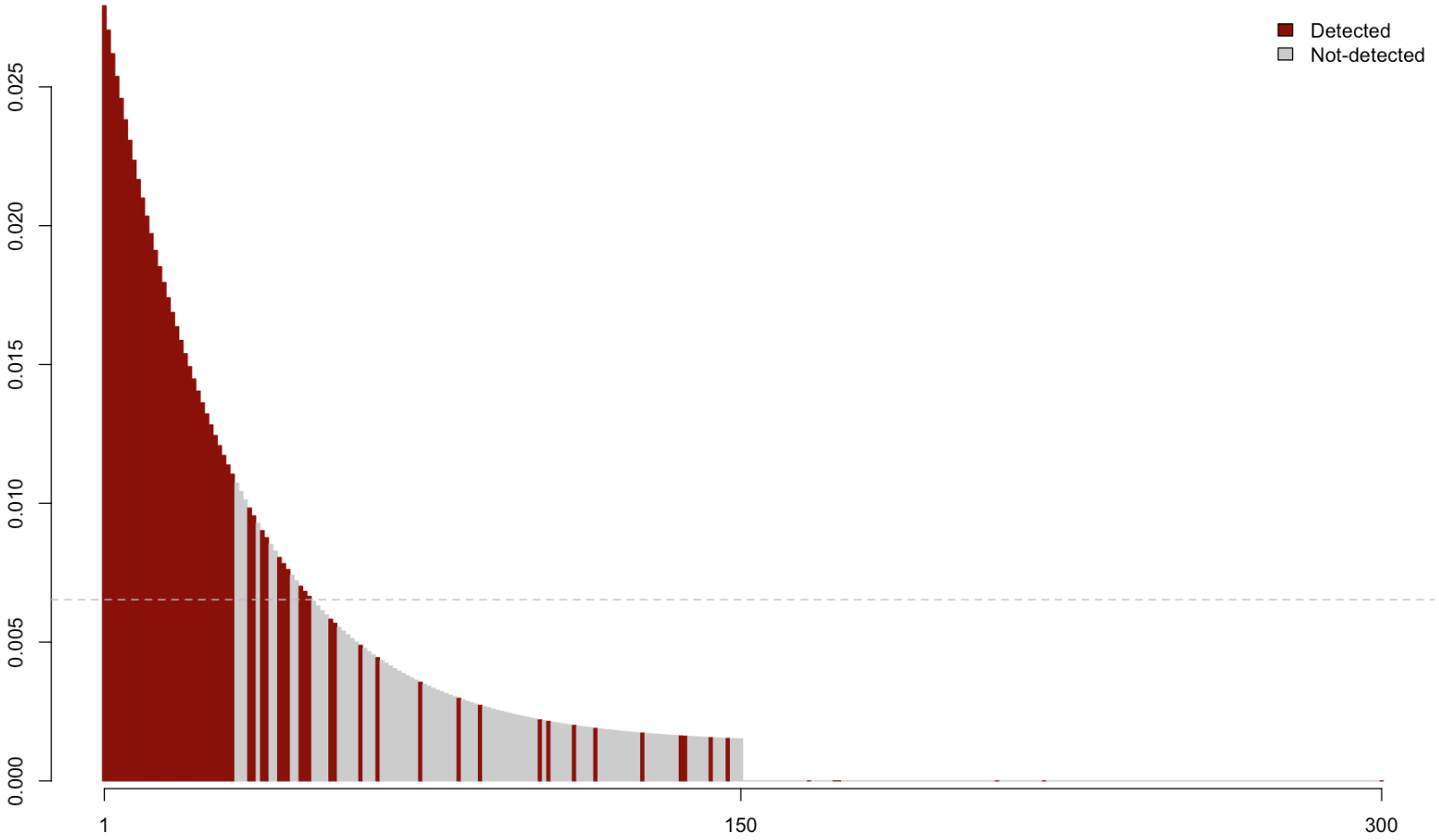
* In addition to p, q and f, the number of hash functions (set in the RAPPOR tab) also effects privacy guarantees.

Life of a Report | Population | Results | True Bits | Estimated Bits | Estimates vs Truth | Residuals

New Participant | New Value | New Report




Truth
0
2000
4000
6000
8000
10000
12000
0.00
0.05
0.10
Estimates


Detected
Not-detected
0.025
0.020
0.015
0.010
0.005
0.000
1
150
300