CS 591: Formal Methods in Security and Privacy

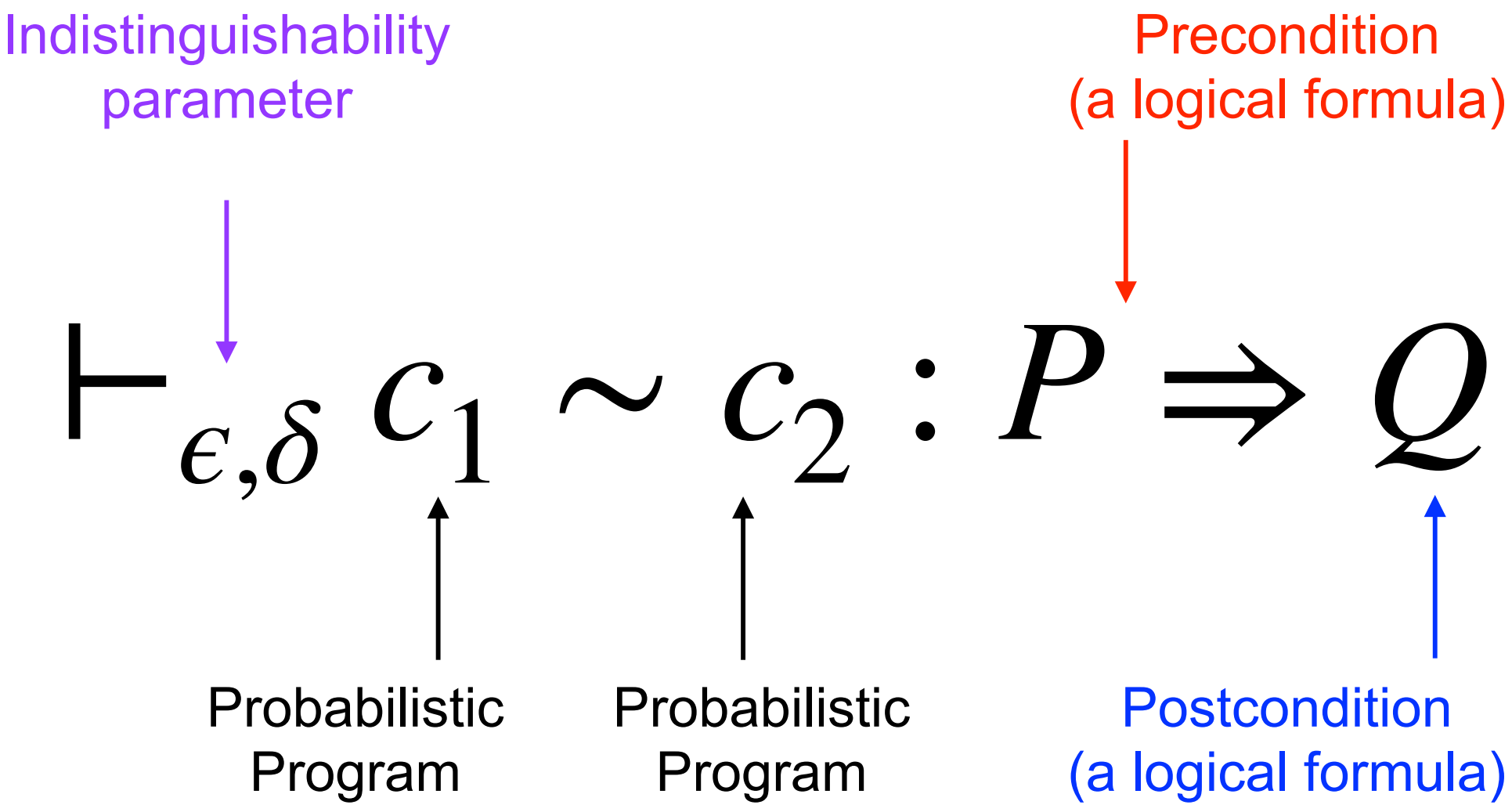
Differential Privacy examples

Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu

Where we were…

apRHL


Indistinguishability parameter
Precondition (a logical formula)
$\vdash_{\epsilon,\delta} c_1 \sim c_2 : P \Rightarrow Q$
Probabilistic Program
Probabilistic Program
Postcondition (a logical formula)

apRHL: More general Lap rule (still restricted)

$$\frac{}{\vdash_{k*\varepsilon,0} x_1 :=\$\ \mathrm{Lap}(1/\varepsilon, y_1) \sim x_2 :=\$\ \mathrm{Lap}(1/\varepsilon, y_2) : |y_1 - y_2| \leq k \Rightarrow =}$$

Probabilistic Relational Hoare Logic

Composition

$$\frac{\vdash_{\varepsilon_1,\delta_1} c_1 \sim c_2 : P \Rightarrow R \qquad \vdash_{\varepsilon_2,\delta_2} c_1' \sim c_2' : R \Rightarrow S}{\vdash_{\varepsilon_1+\varepsilon_2,\delta_1+\delta_2} c_1; c_1' \sim c_2; c_2' : P \Rightarrow S}$$

apRHL
awhile

$$P /\backslash\ e\langle 1\rangle \leq 0 \Rightarrow \neg b1\langle 1\rangle$$

$$\vdash_{\varepsilon_k,\delta_k} c1 \sim c2 : P /\backslash b1\langle 1\rangle /\backslash b2\langle 2\rangle /\backslash k = e\langle 1\rangle /\backslash e\langle 1\rangle \leq n \Longrightarrow P /\backslash b1\langle 1\rangle = b2\langle 2\rangle /\backslash k < e\langle 1\rangle$$

$$\vdash_{\sum\varepsilon_k,\sum\delta_k} \text{while } b1 \text{ do } c1 \sim \text{while } b2 \text{ do } c2 : P /\backslash b1\langle 1\rangle = b2\langle 2\rangle /\backslash e\langle 1\rangle \leq n \Longrightarrow P /\backslash \neg b1\langle 1\rangle /\backslash \neg b2\langle 2\rangle$$

Releasing partial sums

```
DummySum(d : {0,1} list) : real list
   i:= 0;
   s:= 0;
   r:= [];
   while (i<size d)
      s:= s + d[i]
      z:=$ Lap(eps,s)
      r:= r ++ [z];
      i:= i+1;
   return r
```

I am using the easycrypt notation here where `Lap(eps,a)` corresponds to adding to the value `a` noise from the Laplace distribution with `b=1/eps` and mean `mu=0`.

Releasing partial sums

```
DummySum(d : {0,1} list) : real list
   i:=0;
   s:=0;
   r:=[];
   while (i<size d)
      z:=$ Lap(eps,d[i])
      s:= s + z
      r:= r ++ [s];
      i:= i+1;
   return r
```

Today: more examples of differentially private programs

Report Noisy Max

(a)

(b)

(c)

(d)

Suppose that each one of us can vote for one star, and we want to say who is the star that receives most votes.

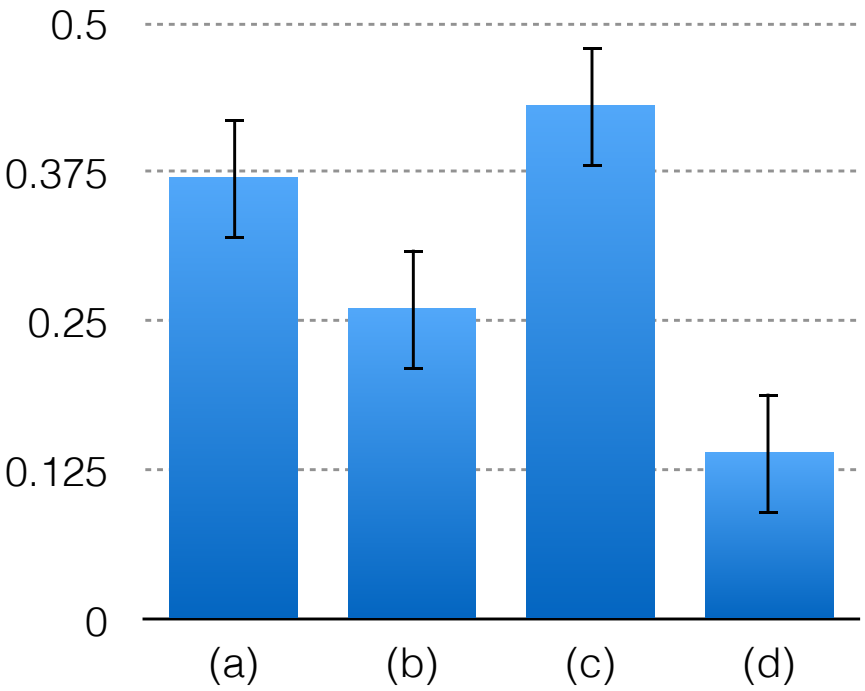
Report Noisy Max

(a)

(b)

(c)

(d)



Intuition:
We can compute the histogram add Laplace noise to each score and then select the maximal noised score.

Report Noisy Max - intuition

We can prove this algorithm
ε-differentially private

Report Noisy Max - intuition

We can prove this algorithm
ε-differentially private

Databases differing
in one individual

Report Noisy Max - intuition

$q_1(D)$+noise $q_1(D')$+noise

$q_2(D)$+noise $q_2(D')$+noise

$q_3(D)$+noise $q_3(D')$+noise

....

$q_k(D)$+noise $q_k(D')$+noise

Databases differing in one individual

We can prove this algorithm ε-differentially private

Report Noisy Max - intuition

l sensitive queries

$q_1(D)$+noise	$q_1(D')$+noise
$q_2(D)$+noise	$q_2(D')$+noise
$q_3(D)$+noise	$q_3(D')$+noise
.....	.....
$q_k(D)$+noise	$q_k(D')$+noise

Databases differing in one individual

We can prove this algorithm ε-differentially private

Report Noisy Max - intuition

We need to coordinate Noises

l sensitive queries

$q_1(D)$+noise $q_1(D')$+noise

$q_2(D)$+noise $q_2(D')$+noise

$q_3(D)$+noise $q_3(D')$+noise

....

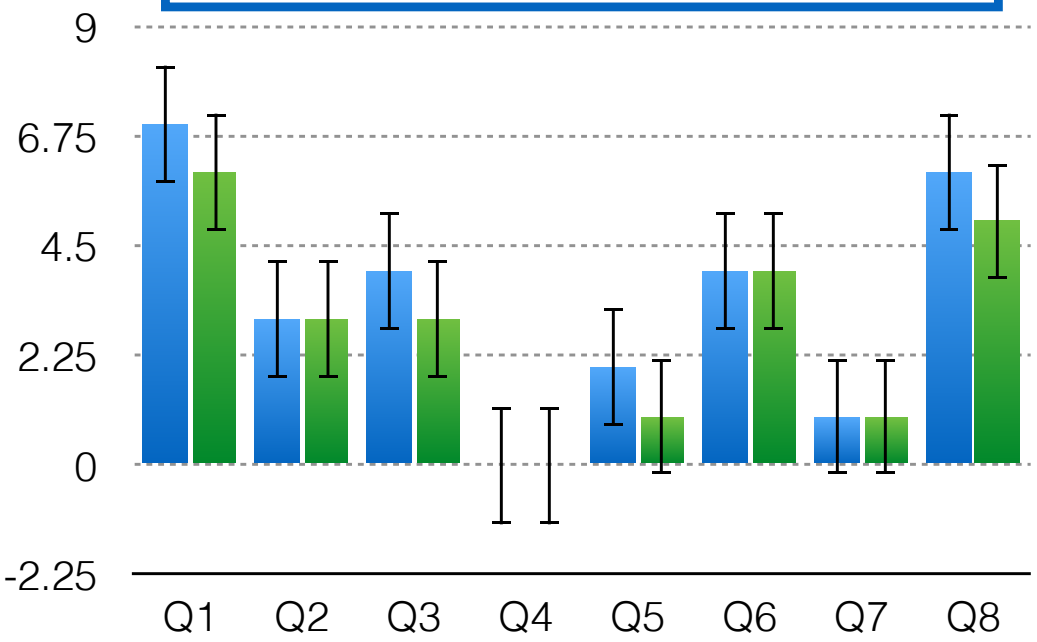
$q_k(D)$+noise $q_k(D')$+noise

Databases differing in one individual

We can prove this algorithm ε-differentially private

Report Noisy Max - intuition

We need to coordinate Noises



We can prove this algorithm ε-differentially private

1 sensitive queries

$q_1(D)$+noise $q_1(D')$+noise

$q_2(D)$+noise $q_2(D')$+noise

$q_3(D)$+noise $q_3(D')$+noise

....

$q_k(D)$+noise $q_k(D')$+noise

Databases differing in one individual

Report Noisy Max

Algorithm 8 Pseudo-code for Report Noisy Max

```
1: function RNM(D, q_1, ..., q_m, ε)
2:     for i ← 1, ..., m do
3:         c_i ← LapMech(D, q_i, ε)
4:     end for
5:     return argmax_i c_i
6: end function
```

Theorem: The algorithm RNM is ε-differentially-private

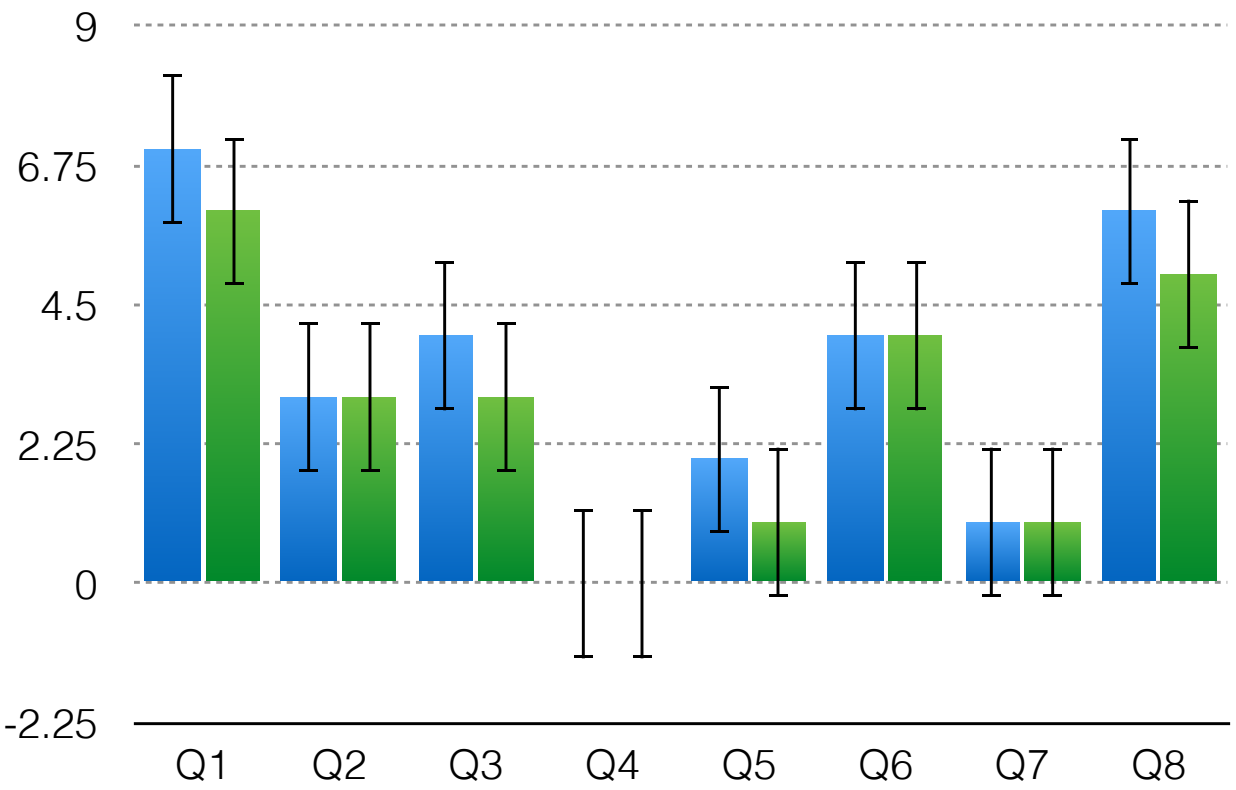
Report Noisy Max

Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Report Noisy Max



Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Report Noisy Max

Simplifying assumptions

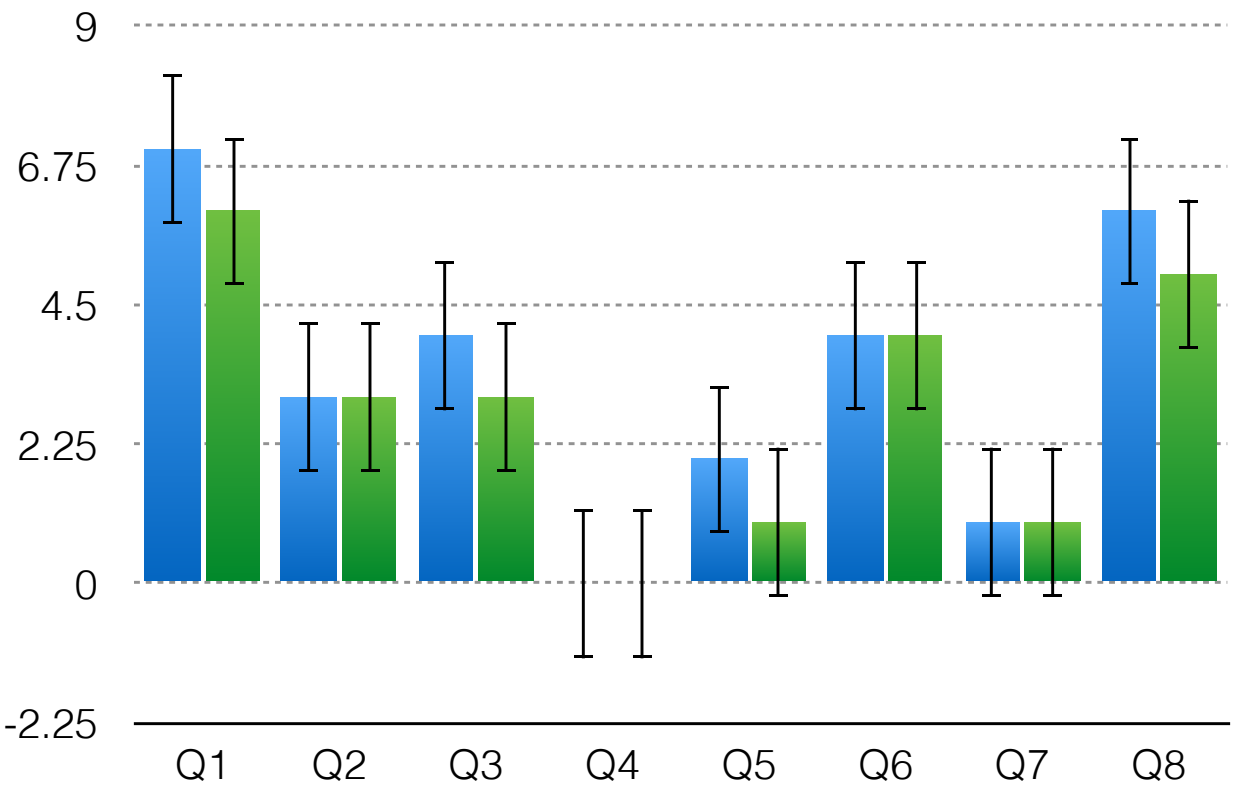
$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k'

noise added at round i.

Report Noisy Max



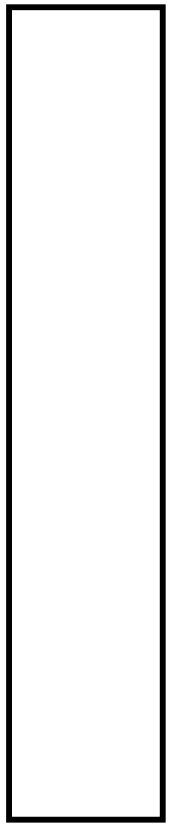
Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round i.

Report Noisy Max



Simplifying assumptions

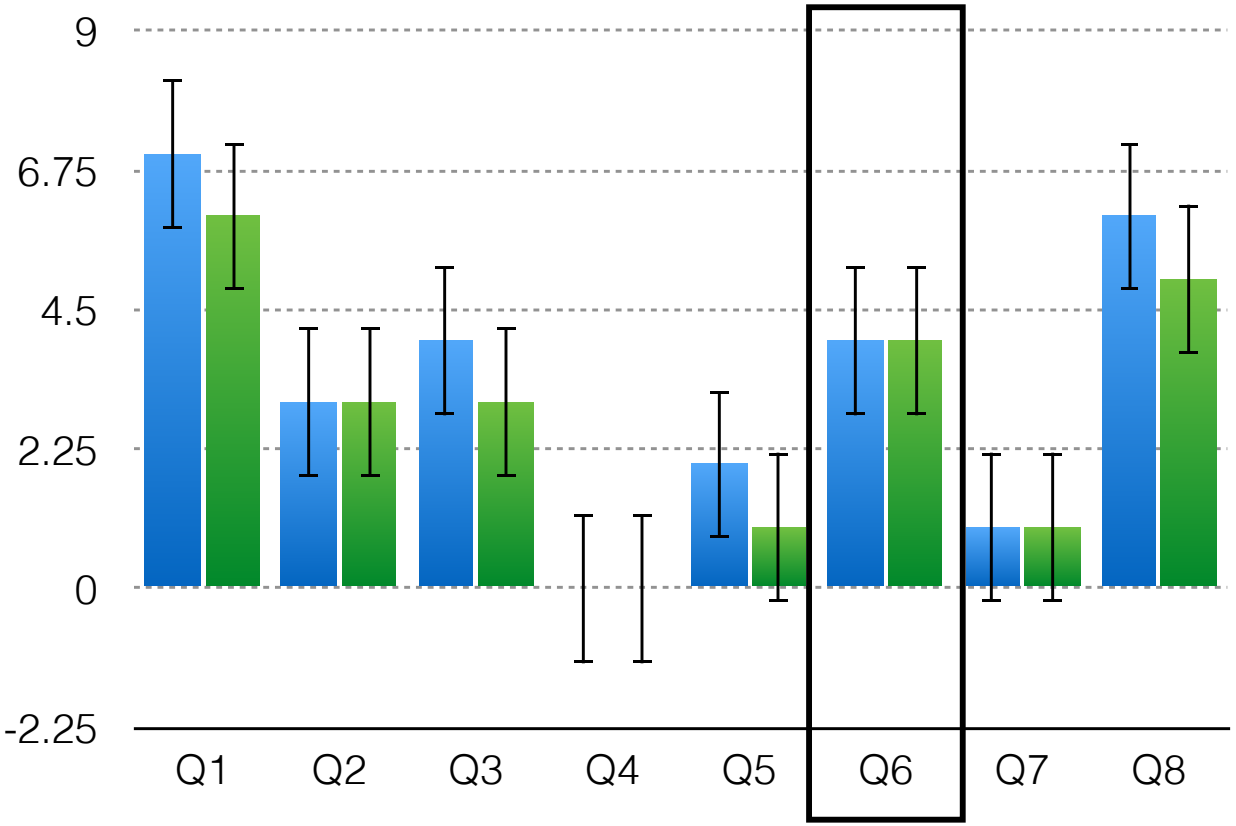
$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round i.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max



Simplifying assumptions

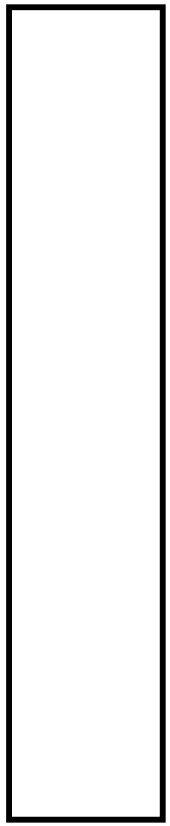
$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round i.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max



Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

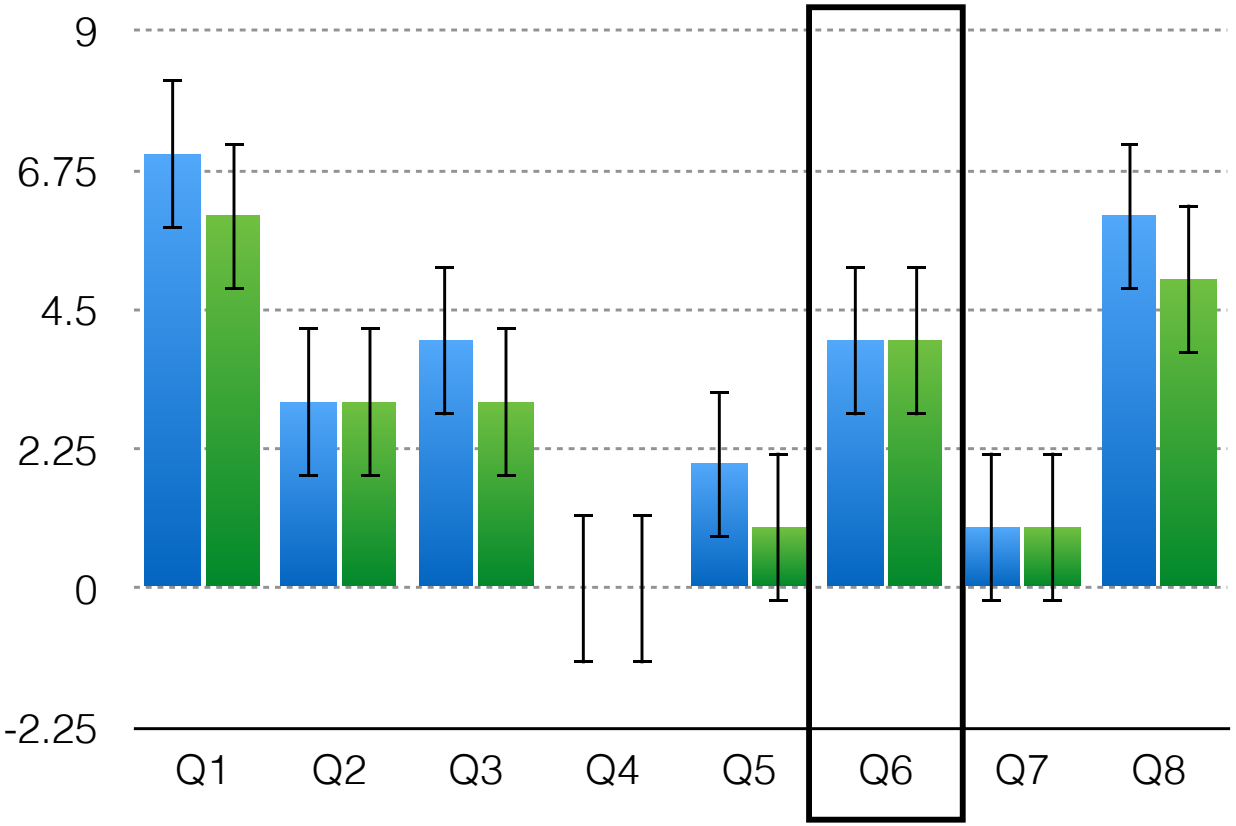
Notation r_k, r_k' noise added at round i.

We want to show:

$$\Pr_{x \sim RNM(D)}[x = i \,|\, r_{-i}] \leq e^{\epsilon} \Pr_{x \sim RNM(D')}[x = i \,|\, r_{-i}]$$

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max



Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

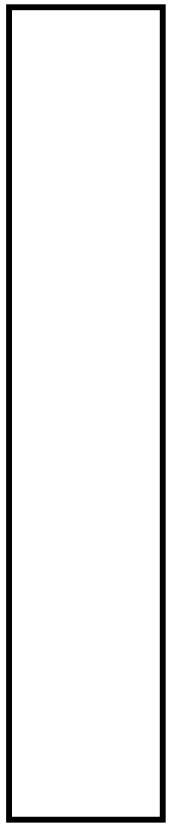
Notation r_k, r_k' noise added at round i.

We want to show:

$$\Pr_{x \sim RNM(D)}[x = i \mid r_{-i}] \leq e^{\epsilon} \Pr_{x \sim RNM(D')}[x = i \mid r_{-i}]$$

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max



Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

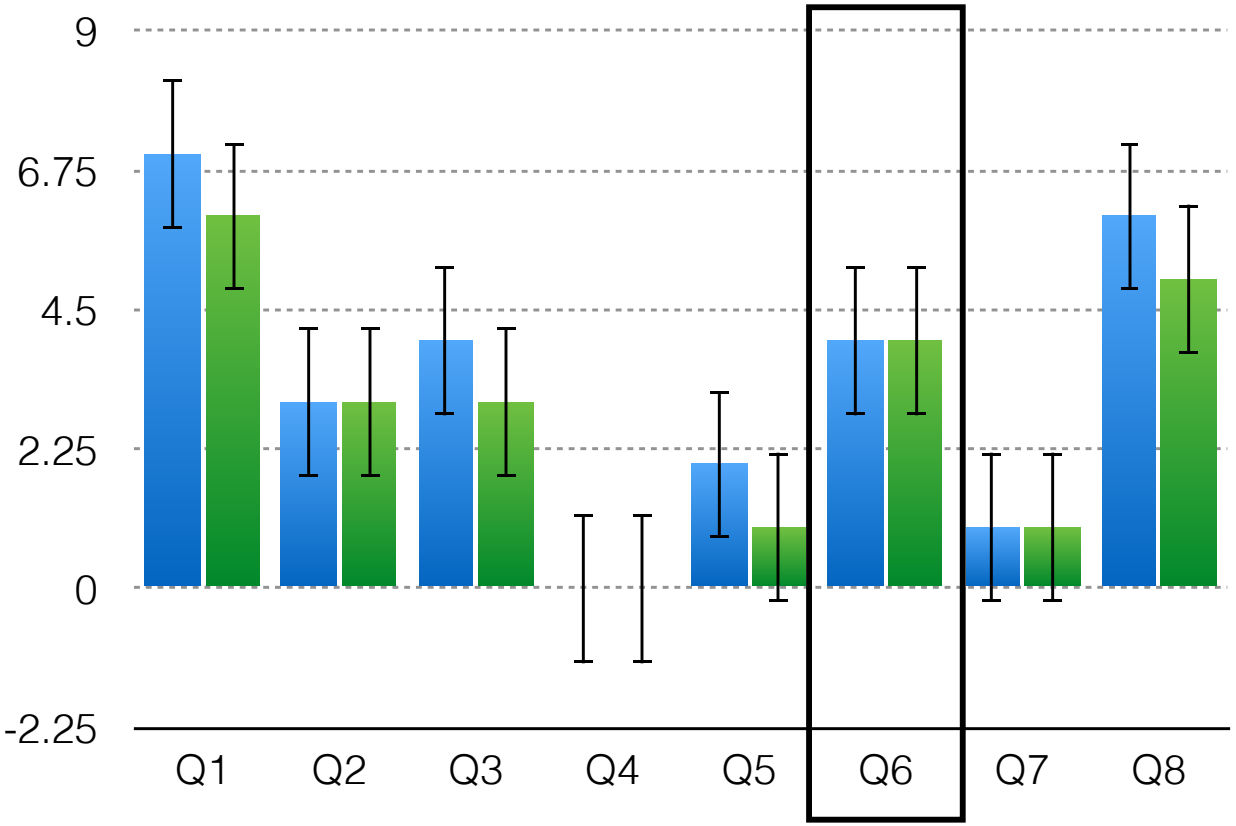
Notation r_k, r_k' noise added at round i.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

By fixing the noises r_j for all $j \neq i$ we can compute the following

$r^* = \min_r \; c_i + r \geq c_j + r_j$ for all j

Report Noisy Max



By fixing the noises r_j for all $j \neq i$ we can compute the following

$r^* = \min_r \; c_i + r \geq c_j + r_j$ for all j

Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round i.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max

$r^* = \min_r \; c_i + r \geq c_j + r_j$ for all j

Notice that

$$\Pr_{x \sim RNM(D)}[x = i \mid r_{-i}] = \Pr_{r \sim Lap}[r \geq r^*]$$

Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round i.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max

$r^* = \min_r\ c_i + r \geq c_j + r_j$ for all j

Notice that we also have

$$c_i' + 1 + r^* \geq c_j' + r_j$$

Since

$$c_i + r^* \geq c_j + r_j$$

and this says

$$\Pr_{x \sim RNM(D')}[x = i \mid r_{-i}] \geq \Pr_{r \sim Lap}[r \geq 1 + r^*]$$

Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round i.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max

$r^* = \min_r \; c_i + r \geq c_j + r_j$ for all j

Summarizing we have:

$$\Pr_{x \sim RNM(D)}[x = i \,|\, r_{-i}] = \Pr_{r \sim Lap}[r \geq r^*]$$

And

$$\Pr_{x \sim RNM(D')}[x = i \,|\, r_{-i}] \geq \Pr_{r \sim Lap}[r \geq 1 + r^*]$$

Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round i.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max

$r^* = \min_r \; c_i + r \geq c_j + r_j$ for all j

Summarizing we have:

$$\Pr_{x \sim RNM(D)}[x = i \mid r_{-i}] = \Pr_{r \sim Lap}[r \geq r^*]$$

And

$$\Pr_{x \sim RNM(D')}[x = i \mid r_{-i}] \geq \Pr_{r \sim Lap}[r \geq 1 + r^*]$$

How can we connect them?

Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

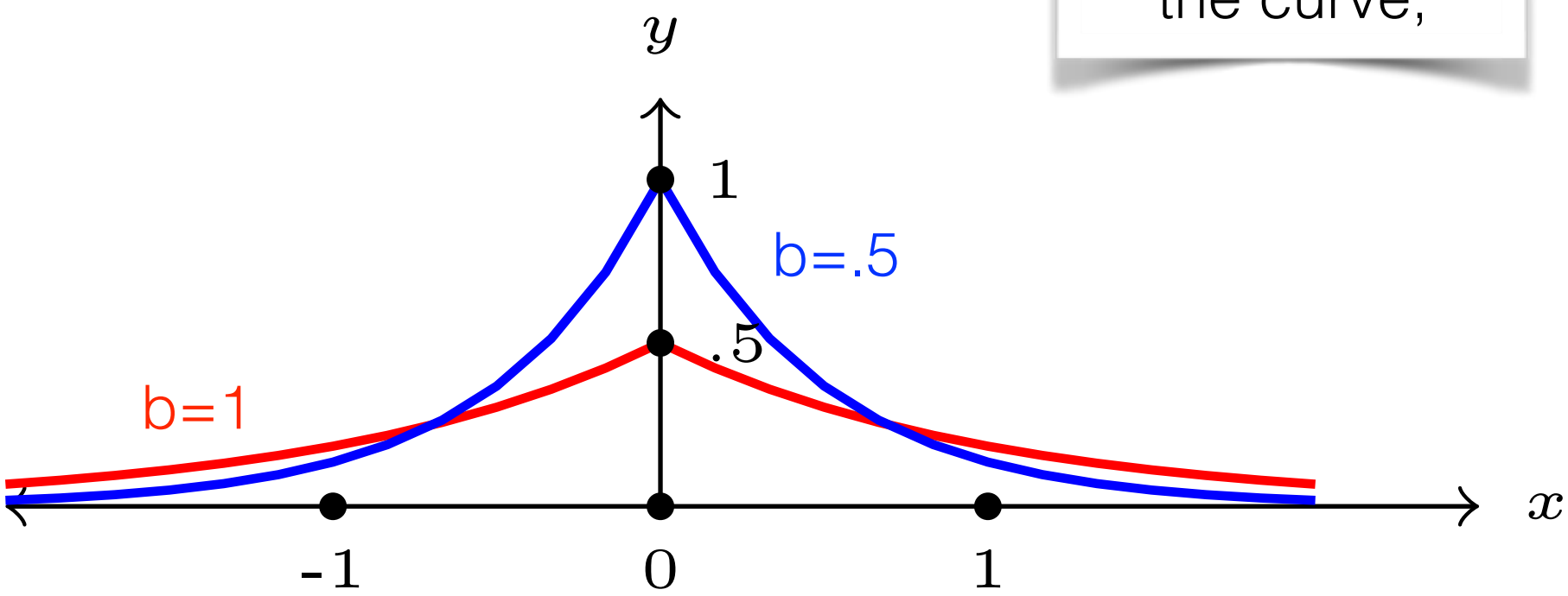
Notation r_k, r_k' noise added at round i.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Laplace Distribution

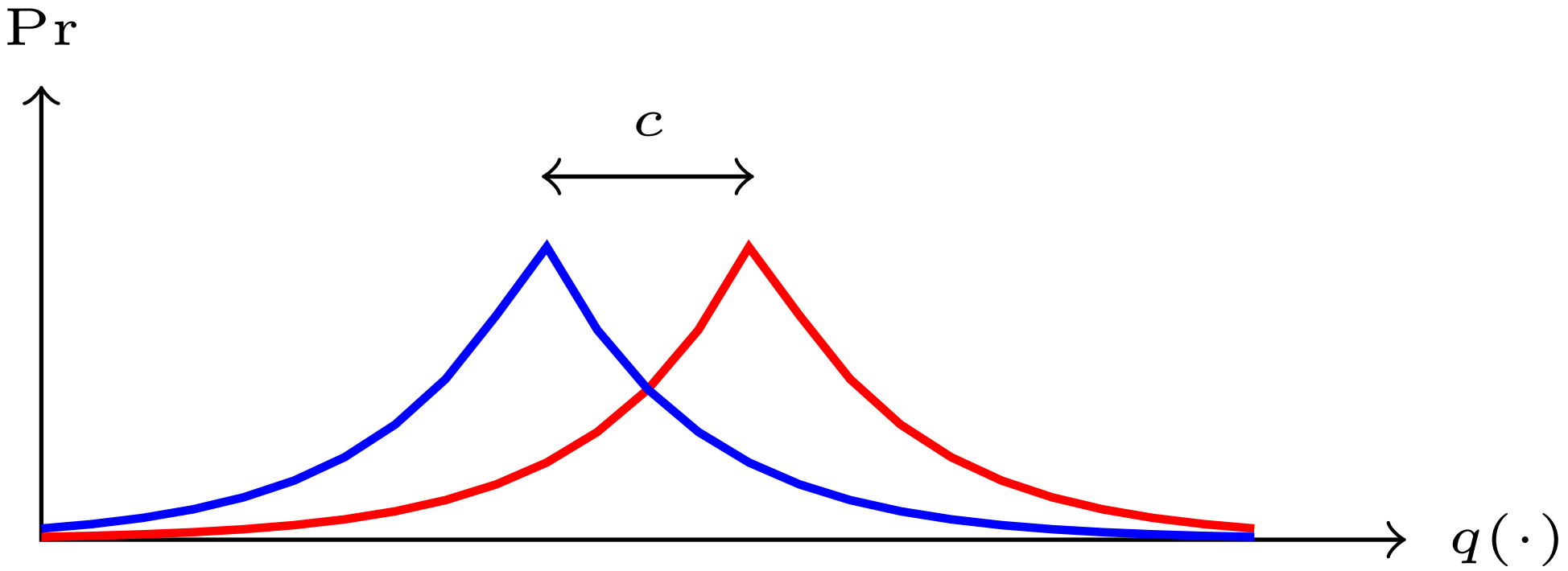
$$\text{Lap}(b, \mu)(X) = \frac{1}{2b} \exp\left(- \frac{|\mu - X|}{b} \right)$$

b regulates the skewness of the curve,



Sliding property of the Laplace Distribution

$$\Pr_{x \sim Lap(\frac{1}{\epsilon}, \mu)}[k \leq x] \leq e^{c\epsilon} \Pr_{x \sim Lap(\frac{1}{\epsilon}, \mu)}[k + c \leq x]$$



Report Noisy Max

Summarizing we have:

$$\Pr_{x \sim RNM(D)}[x = i \mid r\_i]$$

$$= \Pr_{r \sim Lap}[r \geq r^*] \leq e^{\epsilon} \Pr_{r \sim Lap}[r \geq 1 + r^*]$$

$$\leq e^{\epsilon} \Pr_{x \sim RNM(D')}[x = i \mid r\_i]$$

Report Noisy Max

In a similar way we can prove:

$$\Pr_{x \sim RNM(D')}[x = i \,|\, r\_{i}] \leq e^{\epsilon} \Pr_{x \sim RNM(D)}[x = i \,|\, r\_{i}]$$

Report Noisy Max

```
RNM (q_1,...,q_N : (data → R) list,
     b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
     cur = q_i(b) + Lap(1/ε)
     if (cur > max)
          max =  cur ;
          output = i;
  return output;
```

Report Noisy Max

$\vdash(\varepsilon,0)$

$[\mathrm{adj}\ b_1\ b_2, GS(q_i)\leq 1, \ldots]$

```
RNM (q_1,...,q_N : (data → R) list,
     b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
     cur = q_i(b) + Lap(1/ε)
     if (cur > max)
          max =  cur ;
          output = i;
  return output;
```

$[\mathrm{output}_1=\mathrm{output}_2]$

Point-wise reformulation of differential privacy

Given $\varepsilon,\delta \geq 0$, a mechanism $M: db \rightarrow O$ where O is discrete, is (ε,δ)-differentially private iff $\forall b_1 \sim 1\ b_2$ and $\forall s \in O$:

$$\Pr[M(b_1) = s] \leq \exp(\varepsilon) \cdot \Pr[M(b_2) = s] + \delta_s$$

with $\sum \delta_s \leq \delta$.

Can we turn this definition into a rule?

apRHL: pointwise DP rule

$$\frac{\forall r \in R \quad \vdash_{\varepsilon, \delta_r} c_1 \sim c_2 : P \Longrightarrow x\langle 1\rangle = r \Rightarrow x\langle 2\rangle = r \qquad \sum \delta_r \leq \delta}{\vdash_{\varepsilon, \delta} c_1 \sim c_2 : P \Longrightarrow x\langle 1\rangle = x\langle 2\rangle}$$

Report Noisy Max

forall s, |-(ε,0)
[adj b_1 b_2,GS(q_i)≤1,…]

```
RNM (q1,…,qN : (data → R) list,
     b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
           max =  cur ;
           output = i;
  return output;
```

[$output_1$=s => $output_2$=s]

By applying the pointwise rule we get a different post

Report Noisy Max

```
forall s, |-(ε,0)
[adj b1 b2,GS(qi)≤1,…]
RNM (q1,…,qN : (data → R) list,
     b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
           max =  cur ;
           output = i;
  return output;
[output1=s => output2=s]
```

By applying the pointwise rule we get a different post

Notice that we focus on a single general s.

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
     b : list data, ε: R) : nat
  i = 0;
  max = 0;
[adj b1 b2,GS(qi)≤1,…]
  while (i < N){
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
           max =  cur ;
           output = i;
   return output;
[output1=s => output2=s]
```

We can apply standard RHL

Report Noisy Max

forall s, $\vdash_{(\varepsilon,0)}$

RNM ($q_1,\ldots,q_N$: (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
```

[adj b_1 b_2, GS(q_i)≤1,..., invariant]

```
      cur = q_i(b) + Lap(1/ε)
      if (cur > max)
            max =  cur ;
            output = i;
   return output;
```

[$output_1$=s => $output_2$=s]

Invariant

$$\ldots\ (max_1 < cur_1 \Rightarrow output_1 = i_1)$$
$$\wedge\ (max_2 < cur_2 \Rightarrow output_2 = i_2)$$
$$\wedge\ i_1 = i_2 \wedge output_1 = output_2$$

Report Noisy Max

forall s, ⊢(ε,0)

RNM ($q_1,\ldots,q_N$: (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
```

[adj b_1 b_2,GS(q_i)≤1,…, inv]<fun k => if k=s then ε else 0>

```
    cur = q_i(b) + Lap(1/ε)
    if (cur > max)
        max =  cur ;
        output = i;
  return output;
```

[$output_1$=s => $output_2$=s]

Report Noisy Max

forall s, |-(ε,0)

RNM (q_1,…,q_N : (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
```

[adj b_1 b_2,GS(q_i)≤1,…, inv,(i_1=s \/ i_1<>s)] <fun k … >

```
      cur = q_i(b) + Lap(1/ε)
      if (cur > max)
           max =  cur ;
           output = i;
   return output;
```

[$output_1$=s => $output_2$=s]

We can now proceed by cases

Report Noisy Max

forall s, |-(ε,0)

RNM (q_1,…,q_N : (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
```

[adj b_1 b_2,GS(q_i)≤1,…, inv, i_1=s] <fun k => if k=s then ε else 0>

```
      cur = q_i(b) + Lap(1/ε)
      if (cur > max)
            max =  cur ;
            output = i;
   return output;
```

[$output_1$=s => $output_2$=s]

Case 1

Report Noisy Max

forall s, |-(ε,0)

RNM ($q_1,\ldots,q_N$: (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
```

[adj b_1 b_2,GS(q_i)≤1,…, inv, i_1=s] <ε>

```
      cur = q_i(b) + Lap(1/ε)
      if (cur > max)
            max =  cur ;
            output = i;
   return output;
```

[$output_1$=s => $output_2$=s]

We can simplify

Report Noisy Max

forall s, $\vdash_{(\varepsilon,0)}$

RNM ($q_1,\ldots,q_N$: (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
```

[adj b_1 b_2, $GS(q_i)\leq 1$,…, inv, i_1=s] <ε>

```
      cur = q_i(b) + Lap(1/ε)
      if (cur > max)
            max =  cur ;
            output = i;
   return output;
```

[$output_1$=s => $output_2$=s]

What rule shall we apply now?

apRHL: More general Lap rule (still restricted)

$$\frac{}{\vdash_{k*\varepsilon,0} x_1 :=\$\ \mathrm{Lap}(1/\varepsilon, y_1) \sim x_2 :=\$\ \mathrm{Lap}(1/\varepsilon, y_2) : |y_1 - y_2| \leq k \Rightarrow =}$$

Invariant

$$\ldots\ (max_1 < cur_1 \Rightarrow output_1 = i_1)$$
$$\wedge\ (max_2 < cur_2 \Rightarrow output_2 = i_2)$$
$$\wedge\ i_1 = i_2 \wedge output_1 = output_2$$

Report Noisy Max

forall s, $\vdash_{(\varepsilon,0)}$

RNM ($q_1,\ldots,q_N$: (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
    cur = q_i(b) + Lap(1/ε)
```

[adj b_1 b_2, GS(q_i)≤1,…, inv, i_1=s, cur_1=cur_2] <0>

```
    if (cur > max)
        max =  cur ;
        output = i;
  return output;
```

[$output_1$=s => $output_2$=s]

Report Noisy Max

forall s, $\vdash_{(\varepsilon,0)}$

RNM ($q_1,\ldots,q_N$: (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
    cur = q_i(b) + Lap(1/ε)
```

[adj b_1 b_2, GS(q_i)≤1,..., inv, i_1=s, cur_1=cur_2] <0>

```
    if (cur > max)
        max =  cur ;
        output = i;
  return output;
```

[$output_1$=s => $output_2$=s]

We can conclude this case by the asynchronous if rule

Report Noisy Max

forall s, |-(ε,0)

RNM (q_1,…,q_N : (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
```

[adj b_1 b_2,GS(q_i)≤1,…, inv, i_1<>s] <fun k => if k=s then ε else 0>

```
      cur = q_i(b) + Lap(1/ε)
      if (cur > max)
            max =  cur ;
            output = i;
   return output;
```

[$output_1$=s => $output_2$=s]

Case 2

Report Noisy Max

forall s, $\vdash_{(\varepsilon,0)}$

RNM ($q_1,\ldots,q_N$: (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
```

[adj b_1 b_2, GS(q_i)≤1,…, inv, i_1<>s] <0>

```
      cur = q_i(b) + Lap(1/ε)
      if (cur > max)
            max =  cur ;
            output = i;
   return output;
```

[$output_1$=s => $output_2$=s]

We can simplify

Report Noisy Max

forall s, $\vdash_{(\varepsilon,0)}$

RNM ($q_1,\ldots,q_N$: (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
```

[adj b_1 b_2, GS(q_i)≤1,…, inv, i_1<>s] <0>

```
      cur = q_i(b) + Lap(1/ε)
      if (cur > max)
            max =  cur ;
            output = i;
   return output;
```

[$output_1$=s => $output_2$=s]

What rule shall we apply now?

apRHL: More general Lap rule (still restricted)

$$\vdash k*\varepsilon, 0 \sim \frac{}{x_1 :=\$\ \mathrm{Lap}(1/\varepsilon, y_1) \;\; x_2 :=\$\ \mathrm{Lap}(1/\varepsilon, y_2) \;:\; |y_1 - y_2| \le k \Rightarrow =}$$

Can we use this rule here?

apRHL
Generalized Laplace

$x_1 :=\$\ \mathrm{Lap}(\varepsilon, e_1)$

$\vdash_{k_2 * \varepsilon, 0}\ \sim$

$x_2 :=\$\ \mathrm{Lap}(\varepsilon, e_2)$

$: \ |k_1 + e_1\langle 1\rangle - e_2\langle 2\rangle| \leq k_2$

$\Longrightarrow\ x1\langle 1\rangle + k_1 = x\langle 2\rangle$

Report Noisy Max

forall s, |-(ε,0)

RNM (q_1,…,q_N : (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
```

[adj b_1 b_2,GS(q_i)≤1,…, inv, i_1<>s] <0>

```
    cur = q_i(b) + Lap(1/ε)
    if (cur > max)
        max =  cur ;
        output = i;
  return output;
```

[$output_1$=s => $output_2$=s]

We can apply this rule with

$k_2 = q_{i\langle 2\rangle} - q_{i\langle 1\rangle}$

Morally

$\vdash_{(0,0)}$ Pre: true

output = input + Lap(ε)

Post: [output1−output2=input1−input2]

Report Noisy Max

forall s, |-(ε,0)

RNM ($q_1,\ldots,q_N$: (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
      cur = q_i(b) + Lap(1/ε)
```

[adj b_1 b_2,GS(q_i)≤1,…, inv, i_1<>s, cur_1+qi<2>-qi<1>=cur_2] <0>

```
      if (cur > max)
          max =  cur ;
          output = i;
  return output;
```

[$output_1$=s => $output_2$=s]

We can apply this rule with

$k_2 = q_{i\langle 2\rangle} - q_{i\langle 1\rangle}$

Report Noisy Max

forall s, |-(ε,0)

RNM (q_1,…,q_N : (data → R) list,
b : list data, ε: R) : nat

```
  i = 0;
  max = 0;
  while (i < N){
      cur = q_i(b) + Lap(1/ε)
```

[adj b_1 b_2,GS(q_i)≤1,…, inv, i_1<>s, cur_1+qi<2>-qi<1>=cur_2] <0>

```
      if (cur > max)
          max =  cur ;
          output = i;
  return output;
```

[$output_1$=s => $output_2$=s]

We can conclude this case by the asynchronous if rule and conclude