# CS208: Applied Privacy for Data Science
# Programming Frameworks & Query Interfaces

School of Engineering & Applied Sciences
Harvard University

March 22, 2022

# Programming Frameworks for DP

Goal: make it easier for a data custodian or analyst to write programs that are DP, and be confident that they actually are DP.

Common approach (starting with PinQ [McSherry `09]):

- (Small) set of trusted DP subroutines: (Lap, Geo, ExpMech, …) only channel for info to flow from dataset to rest of program.
- Track privacy budget consumption: using composition of DP, with either a runtime monitor or static analysis.
- Allow "stable" data transformations: (recursively) track impact on privacy consumption.

# Dataset Transformations

- Let $d(x, x')$ denote distance between datasets $x, x'$.
  - Number of rows on which they differ for public $n$ model.
  - $|x \Delta x'|$ for unknown $n$ model.

- Def: A mapping from datasets to datasets is $c$-stable (aka $c$-Lipschitz or $c$-stable) iff

$$\forall x, x' \; d\big(T(x), T(x')\big) \leq c \cdot d(x, x').$$

- Lemmas:
  - If $M$ is $\varepsilon$-DP and $T$ is $c$-stable, then $M \circ T$ is $c\varepsilon$-DP.
  - If $T_1$ is $c_1$-stable and $T_2$ is $c_2$-stable, then $T_2 \circ T_1$ is $c_1 c_2$-stable.

# Calculate the Stability Constants

- Per-row transforms (SELECT):
  $$T\big((x_1, \dots, x_n)\big) = \big(f(x_1), \dots, f(x_n)\big).$$

- Trimming: $T(x)$ = remove the bottom and top 20 elts
  (viewing $x$ and $T(x)$ as unordered)

- Subsetting (WHERE): $T(x) = \{r \in x : \pi(r) = \text{true}\}$ (multiset)
  (use unknown $n$ model)

- GROUP BY: $T(x) = (\{r \in x : r_i = c\})_{c \in \mathrm{dom}_i}$

# Partitioning

- "Parallel Composition" Lemma: Let $S_1, \ldots, S_k$ be disjoint subsets of $\mathcal{X}$ and let $M_1, \ldots, M_k$ be $\varepsilon$-DP algorithms (for the unknown $n$ model). Then $M(x) = \left(M_1(x|_{S_1}), \ldots, M_k(x|_{S_k})\right)$ is $\varepsilon$-DP.

- A "1-stable" 1-to-k transformation $T(x) = (x|_{S_1}, \ldots, x|_{S_k})$.

- Also have 2-to-1 transformations (Union, Intersection, Join).

# Tracking Sensitivity

| Transformation | Stability |
|---|---|
| Select($T$, $maper$) | (1) |
| Where($T$, $predicate$) | (1) |
| GroupBy($T_1$, $keyselector$) | (2) |
| Join*($T_1$,$T_2$, $n$, $m$, $keyselector_1$, $keyselector_2$) | (n,m) |
| Intersect($T_1$,$T_2$) | (1,1) |
| Union($T_1$,$T_2$) | (1,1) |
| Partition($T$, $keyselector$, $keysList$) | (1) |

**Table 1.** Transformation stability

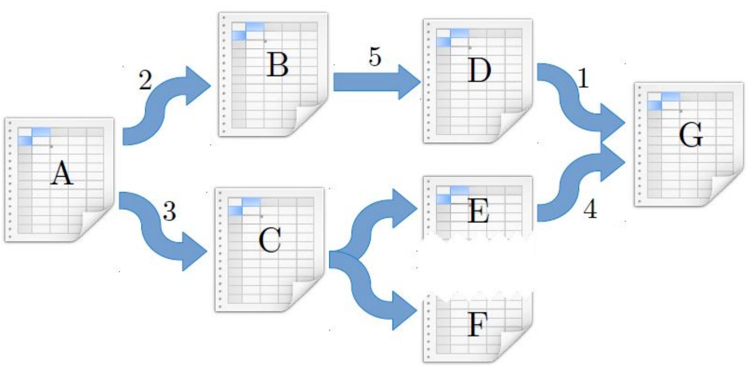



**Fig. 2.** Transformations

| | s | Calculation |
|---|---|---|
| A | 1 | Input table |
| B | 2 | $s(A) \times 2$ |
| C | 1 | $s(A) \times 3$ |
| D | 10 | $s(B) \times 5$ |
| E | 3 | $s(C)$ |
| F | 3 | $s(C)$ |
| G | 22 | $s(D) \times 1 + s(E) \times 4$ |

**Fig. 3.** Scaling factors (s)

[from Ebadi & Sands, "Featherweight PinQ", 2017]

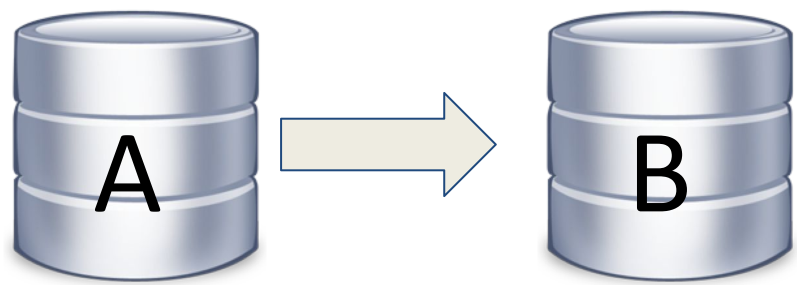
# OpenDP Programming Framework

- Generality in privacy definitions & algorithms
  - Pure DP, approximate DP, concentrated DP, f-DP, etc.
  - Node-level privacy in graphs, user-level privacy in streams, etc.
- Generality in privacy calculus
  - Composition, amplification by subsampling, group privacy, etc.
- Safe extensions of framework with vetted contributions
  - Clear spec for each component's privacy-relevant properties
- Interactive DP algorithms as first-class citizens
  - Adaptive composition, sparse vector, etc.
  - Still in implementation!
- Implementation in Rust w/Python bindings

# Transformations and Measurements

**Transformations**:

Function from data(sets) to data(sets).

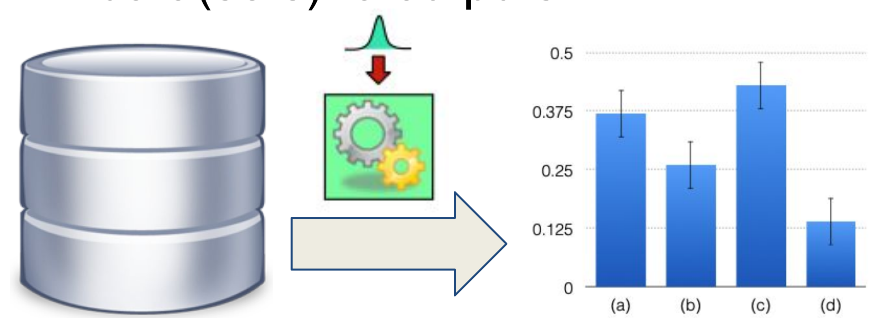



**Transformation Attributes**

- Input domain
- Input metric
- Output domain
- Output metric
- Function
- Stability relation

**Measurements**:

Randomized functions from data(sets) to outputs.




**Measurement Attributes**

- Input domain
- Input metric
- Output measure
- Function
- Privacy relation

# Example Transformations & Measurements

| | Input domain | Input closeness | Output domain | Output closeness |
|---|---|---|---|---|
| Clamp | | | | |
| Bounded Sum | | | | |
| Base Laplace | | | | |

# Example Transformations & Measurements

| | Input domain | Input closeness | Output domain | Output closeness |
|---|---|---|---|---|
| Clamp | | | | |
| Bounded Sum | | | | |
| Base Laplace | | | | |
| c-stable transformation | | | | |

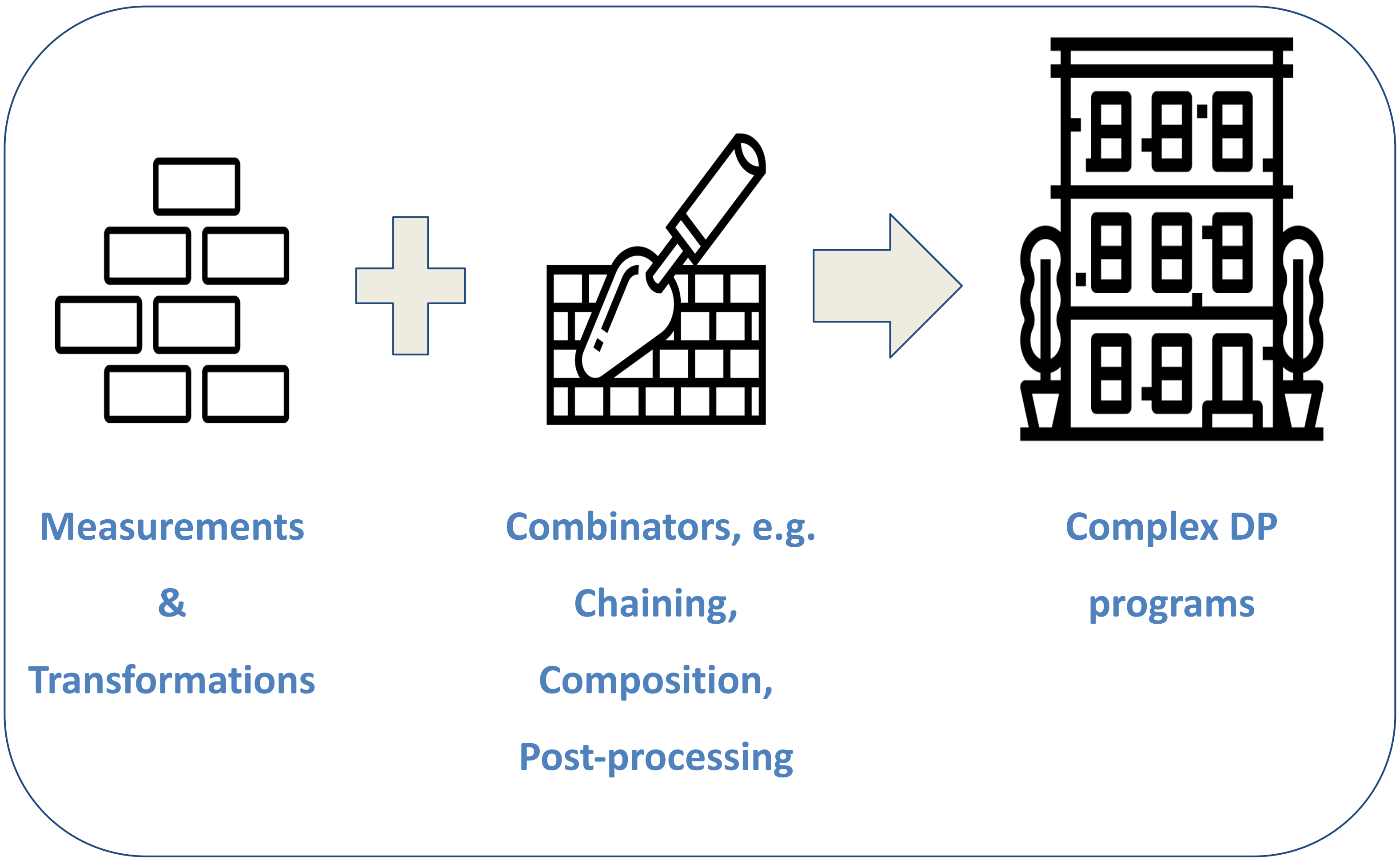
# Example Transformations & Measurements

| | Input domain | Input closeness | Output domain | Output closeness |
|---|---|---|---|---|
| Clamp | | | | |
| Bounded Sum | | | | |
| Base Laplace | | | | |
| c-stable transformation | | | | |
| global sensitivity | | | | |

# Example Transformations & Measurements

| | Input domain | Input closeness | Output domain | Output closeness |
|---|---|---|---|---|
| Clamp | | | | |
| Bounded Sum | | | | |
| Base Laplace | | | | |
| c-stable transformation | | | | |
| global sensitivity | | | | |
| | | | | |
| Base Multidim Gaussian | | | | |
| Restricted Sensitivity | | | | |

# Combinators: Chaining, Composition and Post-processing



# Privacy calculus: privacy and stability relations

To implement a privacy calculus based on the idea of stability we have:

- privacy relations in measurements to capture several notions of privacy. E.g. DP, approx. DP, Renyi DP, zCDP, f-DP.
- stability relation in transformations to capture general aggregate operations. E.g. bounded joins.
- combination of these relations by means of combinators such as chaining and composition.

relation($d_{in}$,$d_{out}$) should imply:
if two inputs are "$d_{in}$-close",
then the corresponding outputs (or distributions) are "$d_{out}$-close".

**Measurement attributes**

- Input domain
- Input metric
- ***Output measure***
- Function
- ***Privacy relation***

**Transformation attributes**

- Input domain
- Input metric
- Output domain
- Output metric
- Function
- ***Stability relation***

# Other Issues in Programming DP

- Multi-relational databases
  - Need to define input metric/adjacency carefully
  - Standard joins have unbounded stability constant, so need to truncate results or use “local sensitivity” approximations.
- Side-channel attacks
  - Info can be leaked through timing, approx. of real numbers, global state, exceptions, etc.
  - Constrain language & implementation to match model better.
- Verifying DP building blocks or more complex DP algs
  - Specialized programming languages.
  - Annotate programs with types to assist automated verification of DP.
  - Tradeoff between usability and expressiveness.
  - Now can even synthesize DP algorithms from examples!
- Guidance on Accuracy & Privacy Budgeting
  - Next time!
- Choice of Programming Model (e.g. SQL vs. MapReduce