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

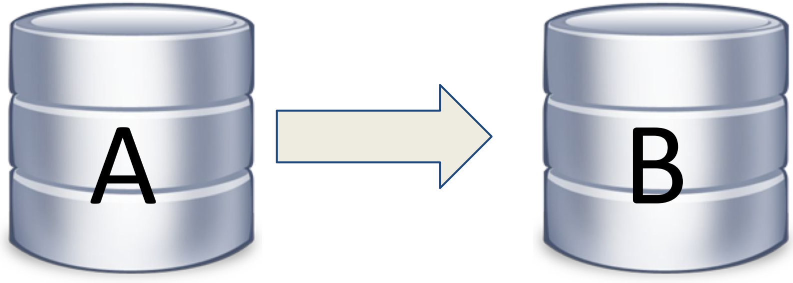
School of Engineering & Applied Sciences
Harvard University

March 24, 2022

Transformations and Measurements

Transformations:

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

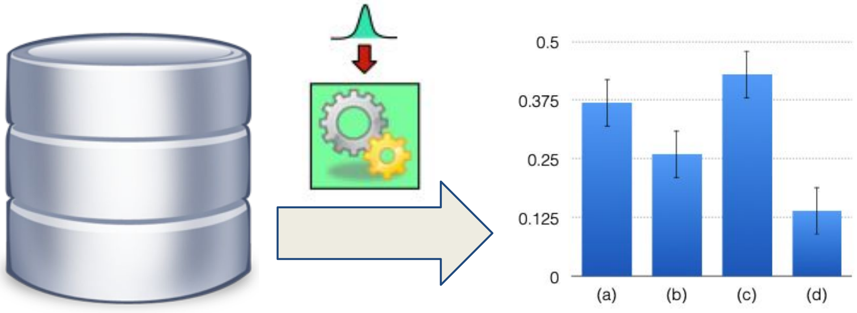


Transformation Attributes

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

Measurements:

Randomized functions from data(sets) to outputs.



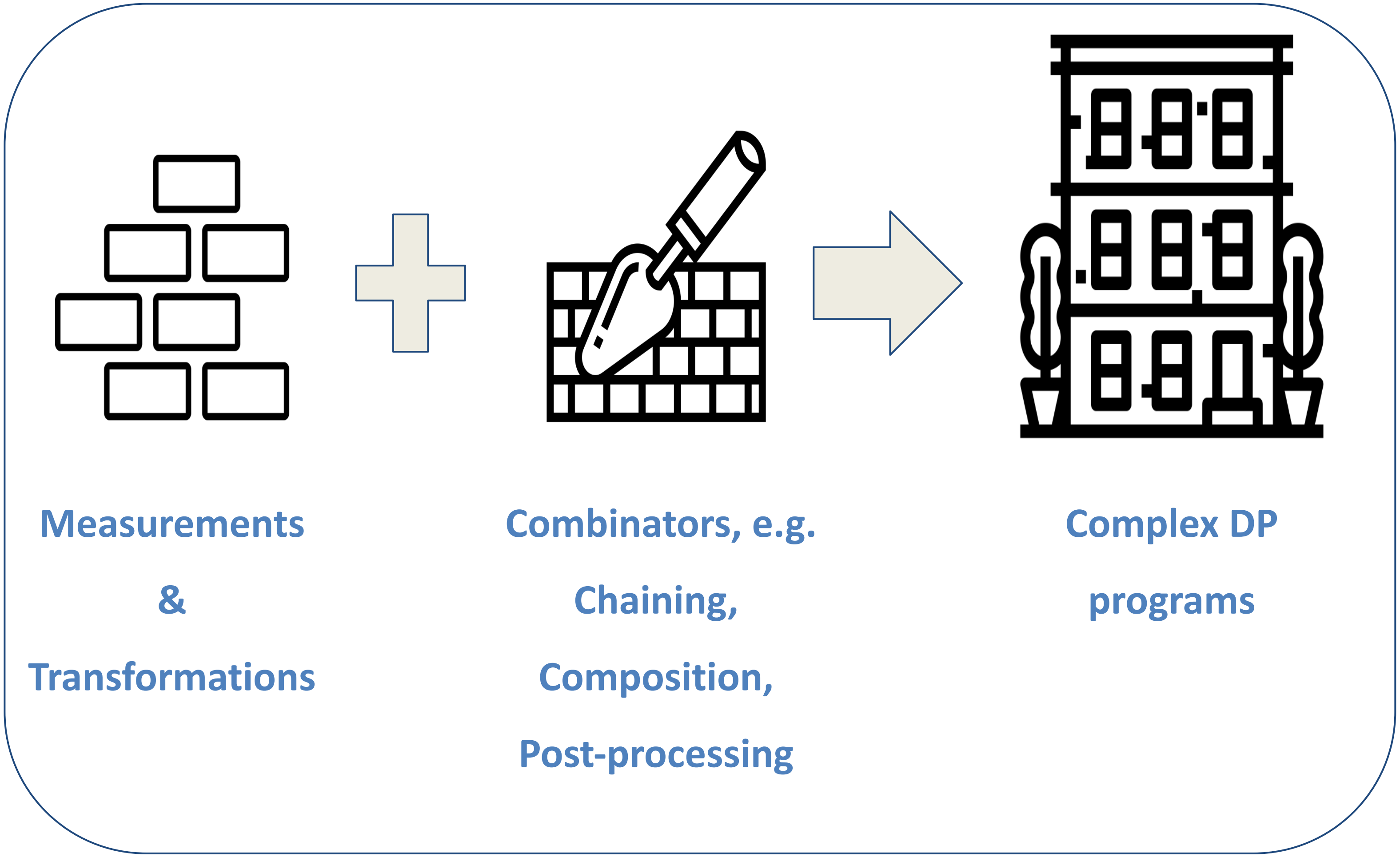
Measurement Attributes

- 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				
c-stable transformation				
global sensitivity				
Base Multidim Gaussian				
Restricted Sensitivity				
Histograms				

Combinators: Chaining, Composition and Post-processing



Measurements

&

Transformations

Combinators, e.g.

Chaining,

Composition,

Post-processing

Complex DP

programs

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***

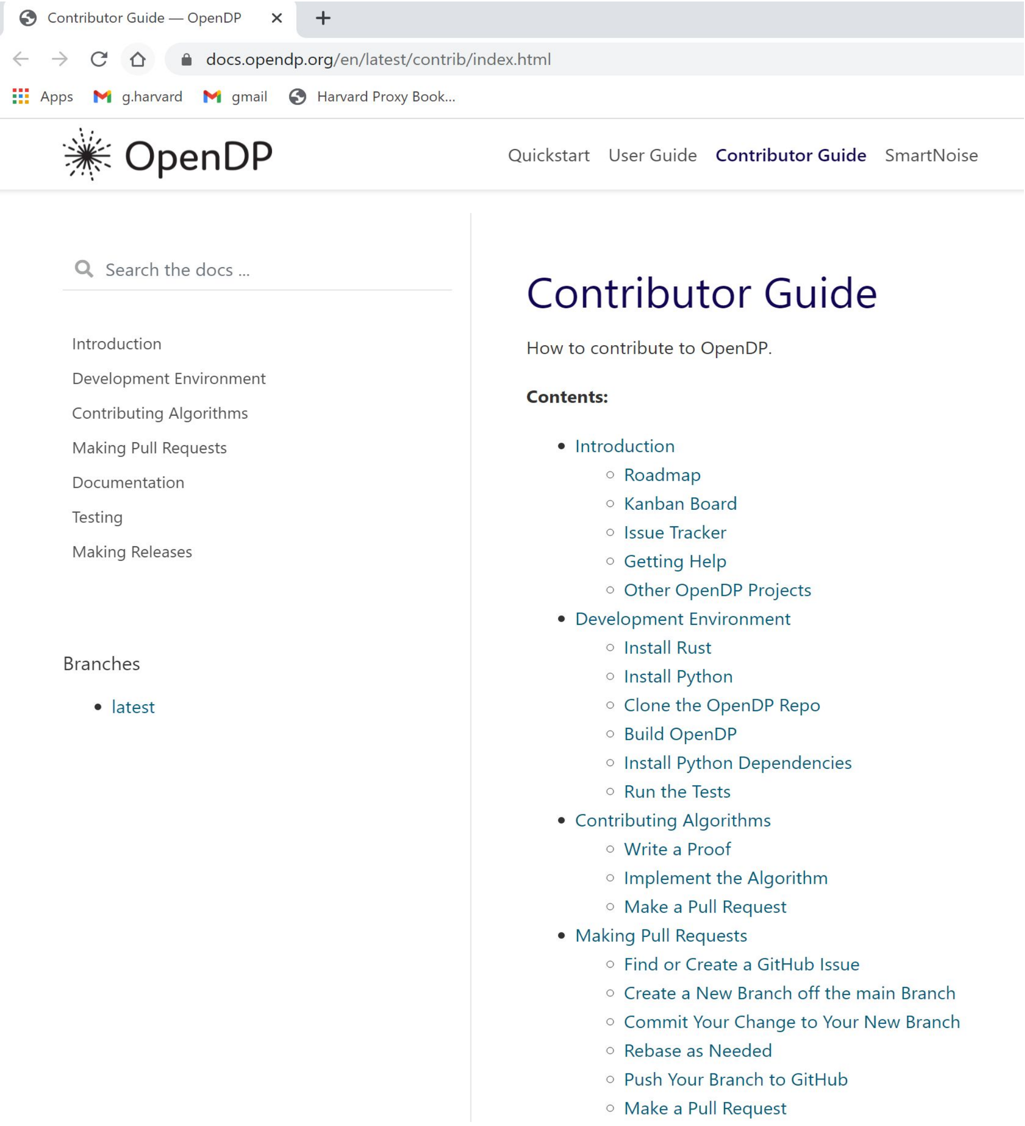
Transformation attributes

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

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

Contributing to OpenDP



Contributor Guide

How to contribute to OpenDP.

Contents:

- Introduction
  - Roadmap
  - Kanban Board
  - Issue Tracker
  - Getting Help
  - Other OpenDP Projects
- Development Environment
  - Install Rust
  - Install Python
  - Clone the OpenDP Repo
  - Build OpenDP
  - Install Python Dependencies
  - Run the Tests
- Contributing Algorithms
  - Write a Proof
  - Implement the Algorithm
  - Make a Pull Request
- Making Pull Requests
  - Find or Create a GitHub Issue
  - Create a New Branch off the main Branch
  - Commit Your Change to Your New Branch
  - Rebase as Needed
  - Push Your Branch to GitHub
  - Make a Pull Request

- Standard GitHub process (pull requests, review, etc.)

- Core Library code gets extra vetting:
  - Proofs for algorithms
  - Prototypes will go in “contrib” or “playground”

- Many types of contributions
  - Bugs
  - Feature requests
  - Documentation
  - Tools

Correctness: Contributing to the Library

Any new code contribution requires a **proof** that for all inputs,

- The contributed code either raises an exception or produces a valid (interactive) measurement or transformation.
- The contributed code does not modify already constructed operators or other library code.

The proof can be:

- derived automatically if the new contribution only uses components of the library.
- supplied by the contributor in documentation form and validated by the OpenDP team (and the user community).
- supplied by the contributor with the assistance of verification tools and validated by the OpenDP team (and the user community).

While privacy is important, vetting should also evaluate accuracy/utility, efficiency, and code quality.

Elements of a Library Contribution

- Implementation in Rust or Python
    - With good code style, documentation, unit tests, etc.

- A pseudocode description of each component
    - Must be sufficient for a member of the dev team (a “code committer”) to verify that the implementation is correct, and
    - Must be sufficient for a DP researcher (on the “editorial board”) to verify that privacy/stability properties
    - Interns will be working on the first examples of these!

- A written proof that every contributed component is valid (produces only valid transformations and measurements)
    - Interns will be working on the first examples of these!

- Some evidence of utility
    - Can be analytic or empirical

Tips for Writing Contributions

- Prototype using the Python bindings when possible
- Break your contribution and its proof into as modular pieces (measurements, transformations, combinators) as possible
  - E.g. Laplace = Clamp >> Sum >> BaseLap
- Get feedback on components as you go
- Think carefully about the choices of metrics and data domains to use throughout
  - You may need to define new ones - make sure they are defined precisely!
- Be explicit about arithmetic
  - Pseudocode & proofs must specify data types, exception/overflow handling
  - Avoid floating point arithmetic; use preferred fixed-point library (MPFR)
  - Can use idealized real-number arithmetic assumptions for prototyping, if stated explicitly

Other Issues in Programming DP

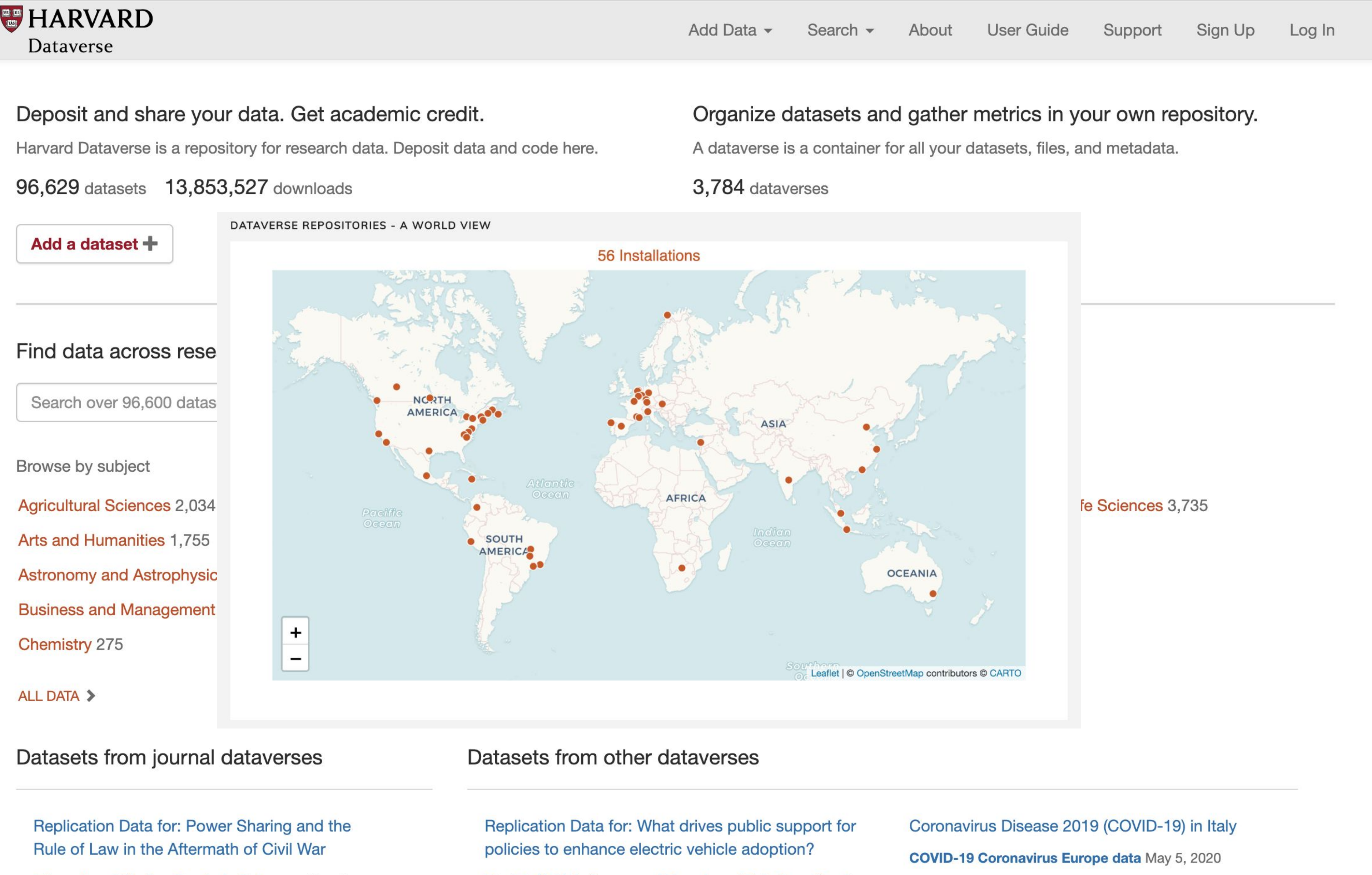
- Multi-relational databases
  - Need to define input metric/adjacency carefully
  - Standard joins have unbounded stability constant, so need to truncate results or use “local sensitivity” approximations.
- Side-channel attacks
  - Info can be leaked through timing, approx. of real numbers, global state, 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
  - Cf. DP Creator
- Choice of Programming Model (e.g. SQL vs. MapReduce

Open Science: Discovery, Replication, Reuse

> *"Accessible and reusable data are fundamental to science in order to continuously validate and build upon previous research. Progressive expansive scientific advance rests upon access to data accompanied with sufficient information for reproducible results, a scientific ethic to maximize the utility of data to the research community, and a foundational norm that scientific communication is built on attribution."*

- Crosas et al. 2015

Data Repositories Use Case



Deposit and share your data. Get academic credit.

Harvard Dataverse is a repository for research data. Deposit data and code here.

96,629 datasets 13,853,527 downloads

Organize datasets and gather metrics in your own repository.

A dataverse is a container for all your datasets, files, and metadata.

3,784 dataverses

Find data across rese

Browse by subject

- Agricultural Sciences 2,034
- Arts and Humanities 1,755
- Astronomy and Astrophysic
- Business and Management
- Chemistry 275

fe Sciences 3,735

ALL DATA >

DATAVERSE REPOSITORIES - A WORLD VIEW

56 Installations

Datasets from journal dataverses

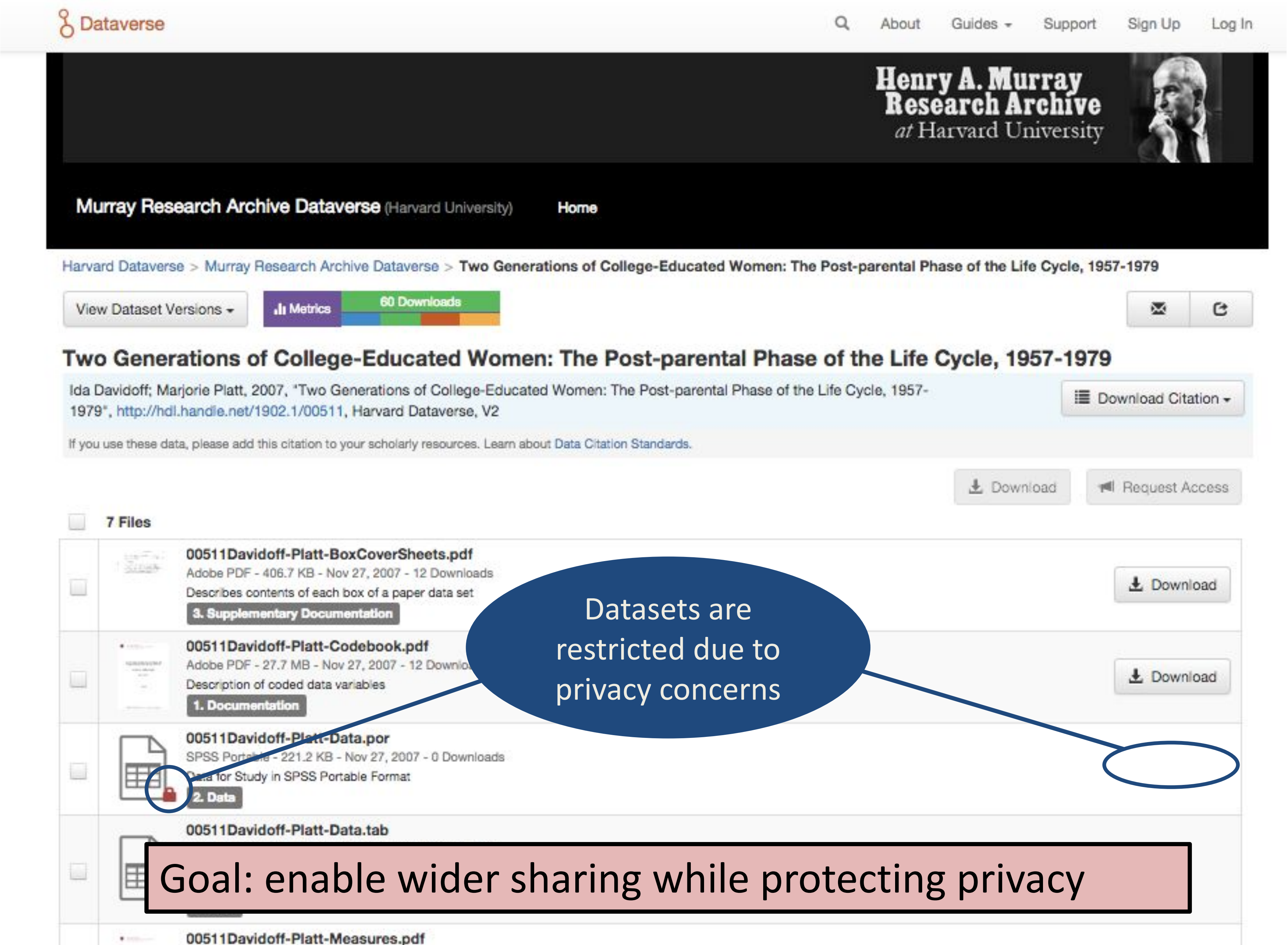
Replication Data for: Power Sharing and the Rule of Law in the Aftermath of Civil War

Datasets from other dataverses

Replication Data for: What drives public support for policies to enhance electric vehicle adoption?

Coronavirus Disease 2019 (COVID-19) in Italy

COVID-19 Coronavirus Europe data May 5, 2020



Dataverse

Henry A. Murray Research Archive *at* Harvard University

Murray Research Archive Dataverse (Harvard University)

Harvard Dataverse > Murray Research Archive Dataverse > Two Generations of College-Educated Women: The Post-parental Phase of the Life Cycle, 1957-1979

Two Generations of College-Educated Women: The Post-parental Phase of the Life Cycle, 1957-1979

Ida Davidoff; Marjorie Platt, 2007, "Two Generations of College-Educated Women: The Post-parental Phase of the Life Cycle, 1957-1979", http://hdl.handle.net/1902.1/00511, Harvard Dataverse, V2

If you use these data, please add this citation to your scholarly resources. Learn about Data Citation Standards.

□ 7 Files

□ 00511Davidoff-Platt-BoxCoverSheets.pdf
Adobe PDF - 406.7 KB - Nov 27, 2007 - 12 Downloads
Describes contents of each box of a paper data set
3. Supplementary Documentation

□ 00511Davidoff-Platt-Codebook.pdf
Adobe PDF - 27.7 MB - Nov 27, 2007 - 12 Downlo
Description of coded data variables
1. Documentation

□ 00511Davidoff-Platt-Data.por
SPSS Portable - 221.2 KB - Nov 27, 2007 - 0 Downloads
Data for Study in SPSS Portable Format
2. Data

□ 00511Davidoff-Platt-Data.tab

00511Davidoff-Platt-Measures.pdf

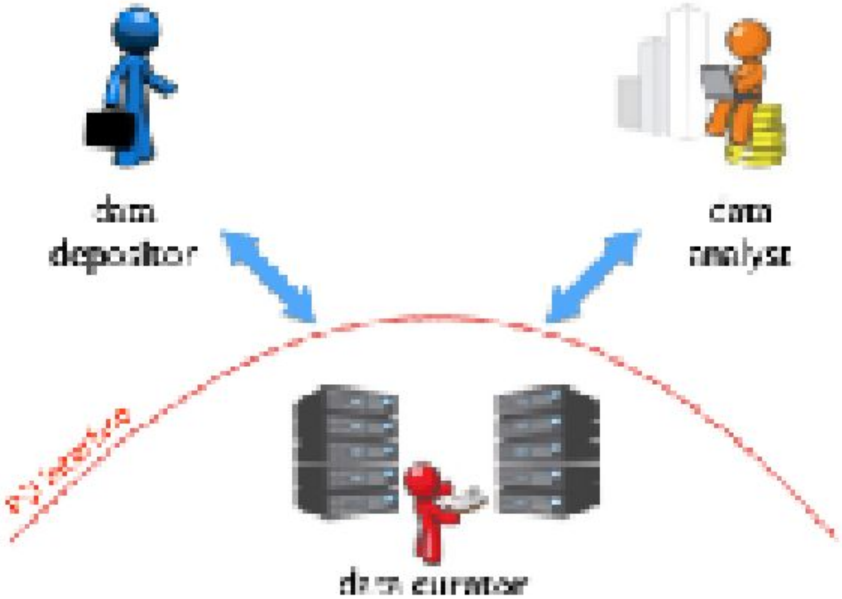
Datasets are restricted due to privacy concerns

Goal: enable wider sharing while protecting privacy

Goals of PSI/DPCreator

- **Generality:** applicable to datasets across social science.
- **Accessible:** no differential privacy expert optimizing algorithms for a particular dataset or application
- **Workflow-compatible:** fits into workflow of practicing social scientists, using familiar concepts & tools
- **Tiered access:** DP interface for wide access to rough statistical information; users can still apply for raw data (cf. Census PUMS vs RDCs)

Actors

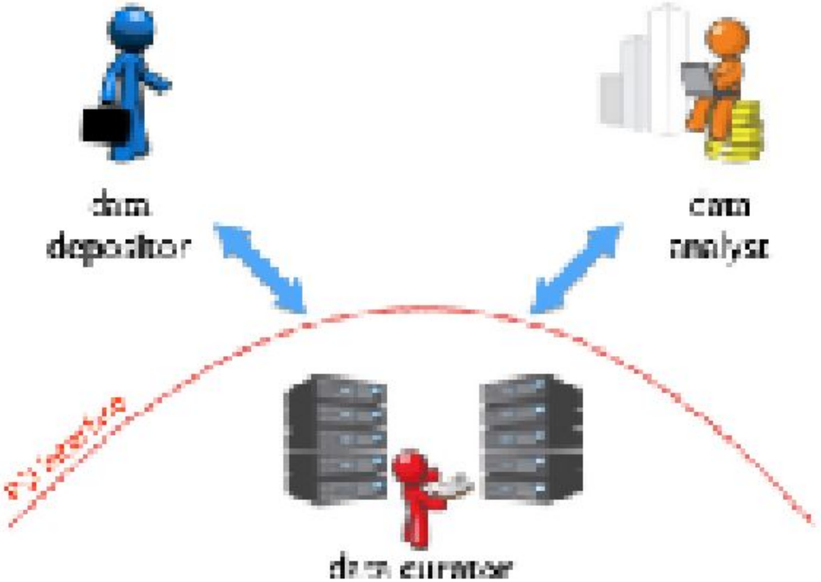


data depositors: Come to deposit their sensitive dataset in a repository, and may wish to make DP access available.

data curators: Maintain the hardware and software on which PSI runs and the accompanying repository infrastructure

data analysts: Come to access sensitive datasets in the repository, often with the goal of data exploration

Actors



	Level of Trust	DP Expertise
data depositors:	Trusted	None
data curators:	Trusted	Modest
data analysts:	Untrusted	None
	Semi-Trusted	None