A Privacy-Preserving Validation Server Version 2.0

Technical White Paper for an Automated Validation Server Prototype

Erika Tyagi, Silke Taylor, Graham MacDonald, Deena Tamaroff, Josh Miller, Aaron R. Williams, and Claire McKay Bowen

September 2024

This technical white paper provides an overview of our privacy-preserving validation server version 2.0 prototype. The goal of the prototype is to provide a testable solution to government agencies looking to improve and automate statistical disclosure control processes. We hope that testing with researchers and agencies, continuous improvement, and dissemination of our learnings will lead to significantly increased access to valuable data and insights used to craft better public policy.

In the first section, we discuss the purpose of the project, key concepts, improvements from our previous version 1.0 prototype, and how we envision a researcher might use the tool. In the second section, we describe our process for building the tool and how the system could be adopted by other organizations. We provide technical details about the front end, application programming interface (API), and back end; how the prototype can be set up and configured; and how the system might be modified and improved in the future.

Introduction

The Data Governance and Privacy Practice Area at the Urban Institute aims to ensure all people are responsibly represented in data. One of the projects within this body of work is the Safe Data Technologies Project, which develops innovative and practical tools to safely expand access to confidential administrative tax data that advance evidence-based policymaking.[1] A validation server is one such tool that can allow many more researchers to analyze confidential data while protecting the privacy of the records represented in the data.

The project team created a version 2.0 validation server prototype and describes the process and development in this white paper. Improving on an earlier prototype developed by the Urban Institute in 2021, the enhanced prototype is more flexible, powerful, and user-friendly than its predecessor (Taylor et al. 2021). We envision that the continuous development and improvement our validation server will contribute to the expanding set of data access tools available to government agencies seeking to adopt modern data privacy methods to support high-quality research and evidence based on sensitive data.

Background: Traditional Data Access

Currently, nongovernment researchers interested in using data analytics to inform the public debate are typically limited to two modes of data access: (1) accessing public data and statistics, and (2) secure access to confidential data. Though public data are extremely useful for certain purposes, such as tax modeling, they are necessarily redacted, blurred, and edited to a degree to protect the privacy of the data. However, such protections make many important analyses impossible.

For those seeking to secure data access, such as administrative tax data, they must first verify if the government agency allows external researchers access. If so, those researchers are often restricted to working in collaboration with analysts in government agencies or must apply to highly selective research programs run by these agencies, where seats may not be available due to ongoing work by incumbents. Additionally, some data access programs require having US citizenship or going through a rigorous clearance and review process to analyze the confidential data, which can take months or years to obtain. Once a researcher gains access, they then face other hurdles. Data access is often restricted to a computer at a secure enclave or through a government-issued laptop. For those without the laptop, they may need to travel hundreds of miles away to the closest secure enclave. Another common hurdle is releasing any outputs from the secure environment may take many more weeks or months for a disclosure review staff member to review and approve. These reviews, though thorough and done by experts, may be based on human judgment and rules of thumb, which can introduce inconsistencies and delays into the process.

Contribution: Another Tier of Data Access

These data access challenges informed the bipartisan Commission on Evidence-Based Policymaking's Final Report in 2017 (Commission on Evidence-Based Policymaking 2017), Foundations for Evidence-Based Policymaking Act of 2018,[2] and the Advisory Committee on Data for Evidence Building in 2022 (Advisory Committee on Data for Evidence Building 2022), leading to the proposal of a National Secure Data Service (NSDS). The proposed NSDS was envisioned as a shared services model that would streamline and innovate data sharing and linking to enable decisionmaking at all levels of government and in all sectors.

Our project is developing one tool, an automated validation server, that would create a new tier of data access to government agencies within the NSDS data sharing model. Specifically, we envision that the validation server, paired with a public file, such as synthetic data, would reduce or eliminate the lengthy clearance process to analyze confidential data, enabling analysts and agencies to produce

valuable public insights more rapidly. The tool also has the potential to replace manual review of requests for disclosure of confidential information, which naturally leads to staff time bottlenecks, with automated privacy-preserving noise addition to the outputs. To our knowledge, other versions of research-focused validation servers in the government and private sector still require manual review, lacking this automated component. Under our system's current design, researchers can submit a set of analyses and receive results that are slightly altered to automatically preserve privacy— meaning that researchers would not have direct access to the confidential data in any form throughout the data access process.

An automated validation server is still early technology and will initially require statistical disclosure offices at government agencies to manually review many of these submissions—although these reviews should be more standardized and less time-consuming than current manual review processes because noise would automatically be added to output and disclosure risk can more easily be measured. Our hope is that as trust grows and the technology develops, government staff will manually review fewer submissions. The automation of disclosure review could significantly increase the number of valuable projects that contribute to better understanding of our society and the creation of more policies based on that research evidence.

Our validation server 2.0 is currently built and tested using public-use data. The goal is to further vet the system to minimize disclosure risks, allow external reviewers to provide feedback to help us further develop and improve the technology, and contribute to the development of automated privacy preserving technologies in government. A user-based authentication system protects the system, and it is only accessible by invitation at this time as we test with trusted users to improve the tool's usefulness in anticipation of a future NSDS.

Key Concepts

We acknowledge that the data privacy literature often has conflicting definitions, so we define several key concepts in this section:

Validation server: A validation server is a digital tool that creates an intermediate layer between a researcher and the original, confidential data. With this intermediate layer, a researcher can analyze the confidential data without seeing them. We make the same delineation between validation server and verification server as Williams and colleagues (2024) to avoid confusion. A validation server provides the output from an analysis that has been slightly altered to ensure privacy protection. A verification server provides information about the quality of statistical inference derived from the associated public data estimates against confidential data without returning confidential estimates, such as ensuring the signs and significance of estimates from a regression model match (Barrientos et al. 2023). For our validation server prototype, we consider an automated validation server without manual submission by statistical agency or program experts with appropriate access to the confidential data. Past examples of a manual process include the US Census Bureau's SIPP Synthetic Beta (SSB) and the synthetic

Longitudinal Business Database (SynLBD) developed in collaboration with the Internal Revenue Service (IRS) and Cornell University (Vilhuber and Abowd 2016).

Differential privacy: This is a strong, formal definition of privacy used in the previous version 1.0 validation server prototype. In the data privacy and confidentiality community, formally private methods mean researchers can mathematically prove and quantify the worst-case scenario of privacy being leaked when releasing a public data publication or statistic. Though several different methods can be used to implement differential privacy in practice, they all add noise to data analysis results proportionate to a specified privacy budget and the sensitivity of the data or statistic requested. This allows data owners or curators, such as government statistical agencies, to quantify the maximum amount of privacy loss incurred with any single or multiple release of information.

Global and local sensitivity: Differential privacy experts often use "sensitivity" to describe how robust a given data analysis request, or query, is to outliers. Experts quantify global sensitivity by measuring how much the result of an analysis changes given the absence or presence of the most extreme possible record that could be in the data population but might not be observed in the data (i.e., the universe of all possible versions of the data). In other words, if privacy experts do not know how a bad actor will attack the data, the privacy expert should assume the worst-case scenario: that the attacker has information on every observation of the data but one and has unlimited computational power. This means differential privacy tries to account for any possible version of the data that could exist, protecting against future data releases and new technologies. If the result from an analysis is too sensitive to outliers for any possible version of the data, then a differentially private method will add more noise to protect the records currently present in the data. "Local sensitivity" is a different measure of robustness from an output, such as a statistic, that quantifies the sensitivity by the absence or presence of the observed records in the confidential data. If a data privacy researcher uses local sensitivity instead of global sensitivity, the formal privacy guarantee no longer holds for the particular method.

Maximum observed sensitivity: Chetty and Friedman (2019) initially proposed Maximum Observed Sensitivity (MOS) algorithm through their work with the US Census Bureau's Opportunity Atlas. This algorithm uses an approach that is similar to calculating the local sensitivity. MOS defines the maximum change in a statistic from adding or removing a single record within partitions of the observed data and some addition possible observations that could be in the universe of possible datasets, rather than considering the full universe of possible datasets. In the version 2.0 prototype described in this paper, we implement a generalized version of a local sensitivity approach that is inspired by the MOS algorithm to add noise to a range of statistics returned from tabular and regression analyses. We will refer to this as MOS-inspired for the remainder of the paper.

Privacy budget: Often described by the Greek letter epsilon in the privacy literature, a privacy budget allows the data owner a knob to better understand and quantitatively limit the worst-case amount of information being released. Paired with information on typical errors for an analysis, the privacy budget also allows the researcher or analyst to better understand the trade-offs of their analysis, so they only request the data they need to be publicly released at an appropriate accuracy

level. In a sense, a privacy budget is a numerical expression of the desire to achieve a balance between the optimal protection of individual privacy and the public value of data. We understand that a system using a local sensitivity approach instead of global sensitivity, such as the MOS-inspired method, means the privacy budget does not hold the same formal privacy guarantees. However, local sensitivity-based methods still provide the data owner a sense of accounting and adds noise to outputs, providing some privacy protection. We discuss this further in the last section about the limitations and future directions.

Review and refinement budgets versus public release budgets: In our prototype, we present two sets of privacy budgets. The first is a "review and refinement budget," which we initially imagine to be fairly generous, to try and mimic exploratory data analysis that researchers are used to. But to minimize the disclosure risk, researchers and analysts would be prohibited from removing any of results produced from this budget from the secure environment. In other words, the results returned from any analyses using the review and refinement budget would only be for experimentation, review, and refinement of any final analyses. Despite the fact that no results may be released publicly, the review and refinement budget should have a finite cap, to enable data owners to meet stringent government requirements about the confidentiality and security of confidential data. The "public release budget" is a separate privacy budget that we envision would be less generous. However, any analyses using the public release budget would be cleared for full public release. Though managing two budgets adds complexity, we felt the trade-off was valuable enough to users, many of whom rated data experimentation in a secure environment as key to any analysis. More discussion on this is provided in conclusion of the paper.

Public-use data: For this prototype, we use the publicly available 2022 and 2023 Annual Social and Economic Supplement (ASEC) of the Current Population Survey (CPS) as our example dataset.[3]

Confidential data: In the future, our goal will be for the automated validation server system to use the administrative tax data. However, for the prototype system, we use the public-use data as a stand-in for confidential data to allow an array of users to test and improve the system while our team conducts various security tests to harden the system.

Synthetic data: Synthetic data replace confidential observations in a dataset with pseudorecords that can statistically represent the confidential observations (Pickens, Andre, and Morrison 2023). The goal of most data synthesis is to closely mimic the underlying distributional and statistical properties of the original, confidential data. In our prototype system, we assume that researchers will have access to synthetic data, which will be the public file of the administrative tax data, with the same variables and structure as the confidential data that will be in the validation server, allowing them to develop their analyses in a familiar environment before submitting their code to the validation server.

Secure environment: In the future, we imagine this system could exist within one of two environments, depending on the implementation needs of the data owners. In the first environment, researchers would apply for access. They would then enter a facility, perhaps one established by the NSDS or the agency holding the data, to perform analyses and request public release of results. The facility would consist of hardened machines and other necessary physical and software safeguards. An

alternative is to allow for remote submission in a secure cloud enclave, using remote access, a VPN, or similar technology. All processing in the system would happen on a secure, remote server under control of the data owner, and the environment on the user's end could be controlled via remote access protocols. Thus, users could potentially submit analyses over this connection and receive results in a secure manner. Decisions as to how to approach this environment will, of course, be up to individual agencies and the NSDS.

User: We define users as individuals who seek to analyze the confidential data, such as analysts, researchers, planners, or decisionmakers. We refer to users and researchers interchangeably in this white paper.

Data steward: A data steward or data maintainer is an individual or agency who possesses the confidential dataset and is responsible for its safekeeping.

Key Improvements from the Previous Prototype

In 2021, the Urban Institute developed a first-of-its-kind privacy-preserving validation server prototype. This prototype allowed researchers to submit analyses and receive noise-infused summary statistics based on the underlying data, preserving the privacy of the individuals represented by perturbing those estimates using differential privacy. This initial version allowed researchers to submit SQL code on public data standing in for confidential data and receive results with noise added using a differentially private mechanism via the SmartNoise platform for tabular statistics.[4] This automated system represented a significant achievement as the first researcher-focused system, to our knowledge, capable of allowing this type of automated and interactive privacy-preserving analyses without allowing direct access to the underlying data.

Through developing and testing the initial prototype, we gained invaluable knowledge for how to make such a system feasible and useful for a statistical agency or future NSDS. Building on the success of this initial prototype and soliciting feedback from privacy experts, economists, and government agency staff, we then incorporated the following key improvements into the version 2.0 prototype:

More types of statistical analyses are supported. The version 2.0 prototype allows for both table generation and a range of outputs based on regression-based analyses. This allows researchers to ask more questions of the underlying data relative to the previous prototype, which only allowed for a much narrower set of tabular output.

The workflow is more friendly to researchers. The version 2.0 prototype accepts as input code written in the R programming language, which many researchers are already familiar with. The previous prototype required researchers to submit code using SQL, which is much less commonly used among social science researchers. Within R scripts that leverage an R package we developed for our system, researchers can also submit a much broader range of preprocessing code prior to defining regression or tabular analyses (e.g., to filter to a subset of observations, create new variables, and apply other

transformations), which allows them to submit much more complex analyses than the previous prototype supported.

Researchers have more informed and granular control over their privacy budgets. The version 2.0 prototype gives researchers more granular control over their privacy budgets by allowing them to specify epsilon values for individual statistics in an analysis. This allows researchers to receive more accurate outputs, such as statistics, of particular importance to their research and better manage their limited privacy budget relative to the previous prototype, which required users to specify an epsilon value for an entire analysis. In version 2.0, researchers can also see more intuitive graphs showing the trade-off between epsilon and accuracy, defined as the bias, for each statistic to help them determine appropriate epsilon values.

Results for common analyses are more accurate. The version 2.0 prototype implements an approach to defining privacy that uses local sensitivity, which generally provides more accurate results for most outputs. We suspect it will provide more accurate results for regression methods relative to formally private methods. Although the system does not provide a formal privacy guarantee, using our MOS-inspired approach still requires accounting of what outputs are permitted out the system, unlike most frameworks within the federal statistical system, providing risk-based privacy protection and a mechanism by which such protection can be consistently managed. The previous prototype (version 1.0) used differential privacy, which resulted in infusing too much noise for many types of analyses to be useful to researchers and practitioners (Barrientos et al. 2021).

The system is more flexible, scalable, and secure. The underlying infrastructure in the version 2.0 prototype decouples key steps in the system to allow for increased flexibility. For example, the updated prototype can more easily accommodate different privacy algorithms, an optional manual review step, and other features to meet the needs of different data stewards. This means that future versions of the prototype could add noise using different privacy enhancing techniques for regression analyses versus tabular analyses, or that an agency could implement different privacy enhancing techniques for different underlying datasets. The ability to implement different privacy enhancing techniques means that if new methods are discovered, our prototype can be updated with the latest, state-of-the-art methods.

The version 2.0 prototype also implements a scalable, parallelized back-end architecture to compute sensitivities across hundreds or even thousands of nodes simultaneously, meaning time-consuming or compute-intensive privacy-preserving algorithms can more easily be implemented. The updated tool also integrates additional security measures including stricter authentication, tighter networking restrictions, and additional layers of encryption.

User Interface and Perspective

A researcher using the validation server would first develop their analyses using a public file, such as a synthetic dataset, from their local computer. Using synthetic data outside of the secure environment lets them investigate the structure of the data, form an analysis plan, and develop their R code through a

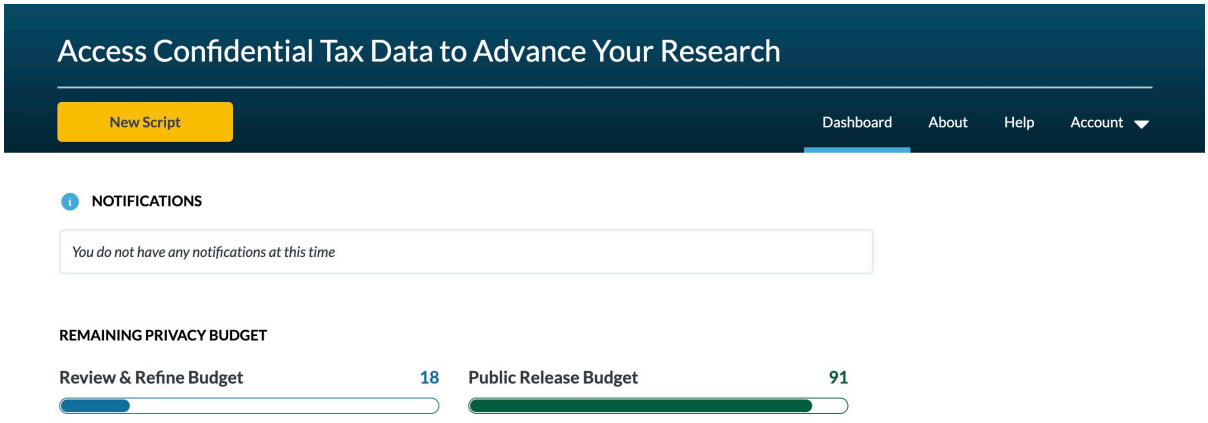
familiar programming environment. Their R code can include nearly any transformation and preprocessing steps (e.g., to filter observations, create new variables, aggregate and reshape data, and write custom helper functions, etc.). Their code would then use functions from our R package to specify analyses for submission to the validation server. These R functions format analyses into the structure required to properly add noise to the output using the MOS-inspired algorithm. Currently, a range of regression and tabular analyses are supported, including linear and general linear models (i.e., lm or glm R objects) along with a standard set of summary statistics (i.e., mean, median, sum, standard deviation, variation, minimum, maximum, counts, and percentiles) optionally computed by groups. A researcher can include multiple analyses within a single R script. This means we assume that researchers will conduct their exploratory data analysis on the public data and use the validation server to gain more accurate outputs using R code. Once the researcher ensures their code runs successfully on the synthetic data, they would then interact with the validation server through the secure web interface.

After entering a valid set of login credentials to access the system, the user would be able to view a personalized dashboard. For a first-time user, the dashboard will show a visualization of two undiminished privacy budgets. For a returning user, the dashboard will show a summary of and links to any previous or in-progress submissions and notifications about any in-progress analyses.

FIGURE 1

Validation Server Version 2.0 Dashboard Interface

Landing page and dashboard with the user's notifications, remaining privacy budgets, and previous submissions



URBAN INSTITUTE

Source: Authors' screenshot of the user interface.
Notes: Upon logging into the validation server interface, the user can access a personalized dashboard with notifications of any activity since their last visit to the site, their remaining privacy budgets, and links to their previous submissions. The landing page also includes a button allowing the user to upload a new submission. This example shows the dashboard for a returning user.

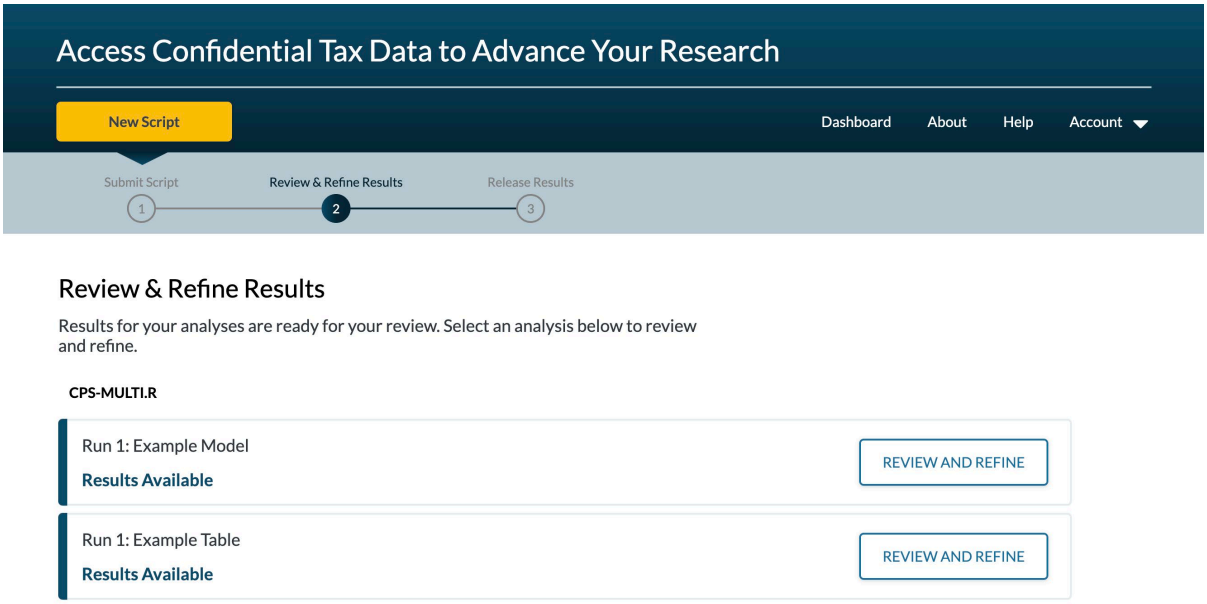
Both first-time and returning users will be able to click a button on the dashboard labeled "New Script" to upload a new analysis. The process of submitting an analysis consists of three steps. In the first step, the user will be asked to upload a script in the form of an R script containing one or more analyses. This step is labeled "1. Submit Script" in the interface. While waiting for the system to compute results

for the uploaded script, the user will have the option to remain on the page until the results display or exit and be notified when the results are available. Notifications will be sent to both the user's dashboard and email address.

FIGURE 2

Validation Server Version 2.0 "Review and Refine Results" Interface

User interface to view review and refine results from previously submitted analyses



URBAN INSTITUTE

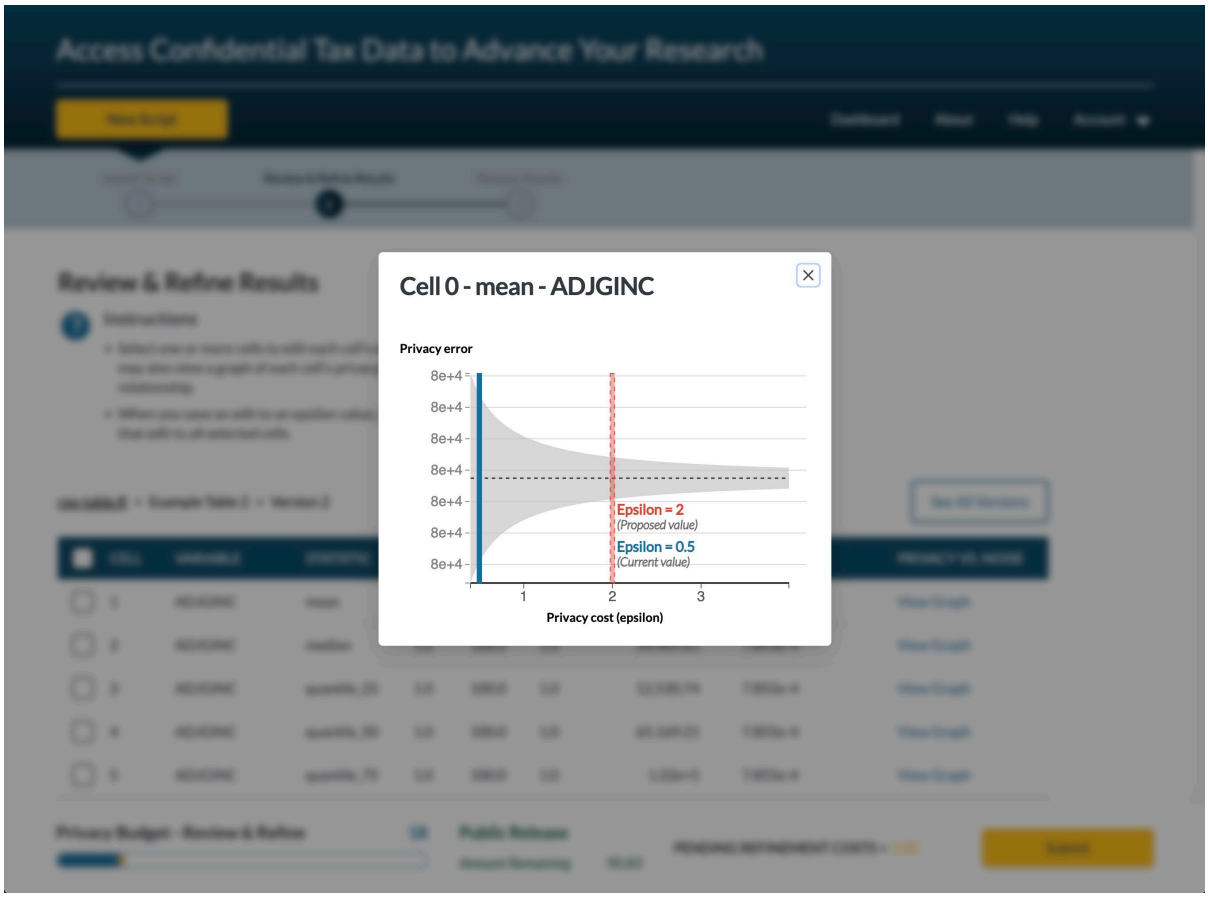
Source: Authors' screenshot of the user interface.
Notes: The user can review and refine results from a successful submission. Users can include multiple analyses within an R script, which are displayed as separate output tables in the validation server interface. This example includes two analyses that the user named "Example Model" and "Example Table" within their R script.

In the second step of the process of submitting a new analysis, the user will be able to select a specific regression or tabular analysis contained within their script. This step is labeled "2. Review & Refine Results" in the interface. For each analysis, the user will see a table containing the noise-infused estimate and epsilon value for each statistic. The user will also see a button linking to any previous versions of the same analysis. The user will be able to select one or more statistics and update its epsilon value. For each statistic, the user will be able to open a modal window showing a graph of the relationship between privacy error and privacy cost, including a comparison between that of the initial epsilon value and any user-proposed change in epsilon. Our goal is that this visualization helps users determine an optimal trade-off of privacy and usefulness for their specific research and choose an appropriate epsilon value. As the user proposes refinements by editing epsilon values, a running total of "pending refinement costs" will display on the bottom of the page. Upon clicking "submit," a modal window will prompt the user to confirm the proposed changes to their review and refinement budget.

FIGURE 3

Validation Server Version 2.0 Privacy Error vs. Cost Tradeoff Graph

User interface displaying the relationship between privacy error and cost for a given statistic



URBAN INSTITUTE

Source: Authors' screenshot of the user interface.

Notes: For each statistic, the user can view a graph showing the trade-off between privacy error (along the y-axis) and privacy cost or epsilon (along the x-axis). The goal of the visualization is to help users specify an appropriate level of epsilon based on their specific research needs. The horizontal dashed line indicates the noise-infused value returned to the user. The shaded area indicates the 90th percentile of noise based on the MOS-inspired algorithm. The blue vertical line indicates the current value of epsilon for the statistic, while the red vertical line indicates the user-proposed value. This example shows this trade-off for the mean of the ADJGINC variable if the user previously specified an epsilon value of 0.5 and is considering updating the value to 2.0.

Once the new results for the proposed privacy cost refinements are ready, the user will receive a notification to both their dashboard and their email address to view them. At this point, the user has the option to either propose additional refinements, or to click "Release" to proceed to the final step in the process. This step is labeled "3. Release Results" in the interface. The user will be able to select one or more analyses for which they have proposed privacy cost refinements. As the user selects analyses to release, a running total of "pending release costs" will display on the bottom of the page. Upon clicking "submit," a modal window will prompt the user to confirm the proposed changes to their public release budget. Released results will be available to download as a CSV file and remove from the system. We

imagine that a data steward may initially include a manual review step by a statistical disclosure expert before letting users release results from the secure environment.

The interface also includes "About" and "Help" pages with detailed information about the data in the system including a downloadable codebook, example R code to help users develop regression and tabular analyses, and additional background on the project. In the future, we plan to develop a more robust set of resources to help users learn about the key concepts described in the previous section to ensure they are able to use the validation server system effectively. Effective communication materials will be key for potential users to adopt such a system.

Building the Tool

To prioritize improvements from the version 1.0 prototype, the Urban team conducted extensive user testing with a range of researchers, analysts, and staff at federal agencies. Overwhelmingly, potential users wanted to submit a broader range of analyses (more than tabular statistics, particularly regression analyses), use a more familiar statistical programming language (such as R), and have more granular control over their privacy budgets—all of which are implemented in the version 2.0 prototype.

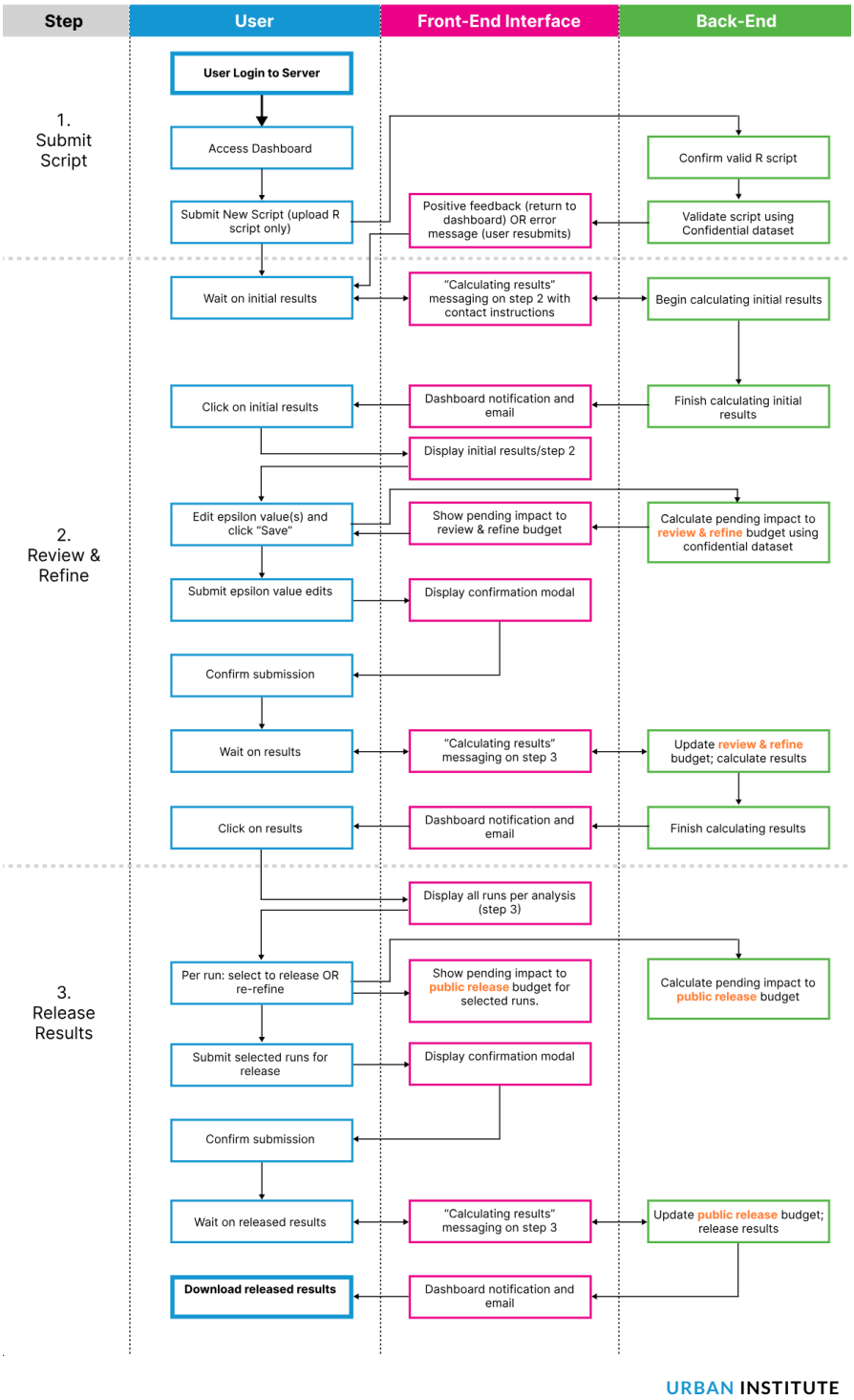
To develop the version 2.0 prototype, the Urban team used an iterative process to design and build the tool in collaboration with the digital agency Graphicacy.[5] The team refined the design for the tool through a series of more than 10 sets of wireframes. Throughout the wireframe design process, the Urban team solicited and incorporated feedback from researchers, privacy experts, and technologists at all career levels to center the user's perspective. The level of detail of the wireframes increased over time as the feedback sessions revealed which particular interactions required additional thought and complexity. The wireframes were developed concurrently to the back end and API.

The Urban team also created a detailed process map to visualize the coordination between user input, the response of the front-end interface, and the back-end activity of the system. This process map was helpful in understanding and improving the user experience of the tool as well as identifying outstanding questions about its functionality and outstanding privacy concerns.

FIGURE 4

Validation Server Version 2.0 Process Map

Concurrent interactions between user inputs, feedback from the front-end interface, and the back-end engine



Source: Authors' depiction of the validation server process.

The team then worked closely with Graphicacy to translate the final set of wireframes first to a clickable prototype consisting of high-fidelity mockups, and then to a live prototype through an iterative build process involving continuous feedback. Allowing for a continuous feedback loop helped us make real-time decisions to identify and improve edge cases with error management and user workflows. The decoupled nature of processes running outside of a page load on the website presented a number of user-experience challenges that were mitigated with loading icons and a notifications system that announces to the user when results are ready for review. Visualizing the remaining privacy budgets also proved tricky because we needed to keep the overview present but small enough to not overwhelm the other elements on the screen.

Throughout the development process, Urban's engineering team conducted research on security requirements for the system with input from government and Urban security experts. Like with the version 1.0 prototype, we explicitly decided to use opensource technology to allow the system to be adaptable to future use and adopted an "infrastructure as code" development approach to ensure that the underlying infrastructure could be reproduced and version controlled. We also only used cloud services that comply with FedRAMP High Baseline standards, ensuring that federal government agencies could adopt the system in the future.[6]

Technical Infrastructure

In this section, we describe the technical infrastructure at a high level and discuss how others can set up, configure, and modify this system themselves.

How the Technical Infrastructure Works

The version 2.0 validation server prototype is composed of four parts: a front-end user interface for submitting analyses and viewing results, a back-end system for running analyses in a privacy-preserving manner, an API to coordinate communication between the front and back ends, and an R package to let users define regression and tabular analyses to submit to the system.

FRONT END

Researchers interact with the validation server via a front-end user interface built using TypeScript, an extension of JavaScript. The front-end interface facilitates the following via direct calls to the API: authenticating users, viewing a dashboard with previously submitted analyses, uploading new analyses, refining and releasing analyses using data privacy budgets, and displaying the trade-off between accuracy and privacy for each statistic in an analysis. The interface also displays notifications to indicate changes to the status of a submitted analysis and additional information to help orient users to the system.

APPLICATION PROGRAMMING INTERFACE

Interactions between the front-end interface and the back end are handled by an API built using the Django REST framework. The Django model objects are stored in a MySQL database hosted on an

Amazon Web Services (AWS) Relational Database Service (RDS) server. The API and front-end application are hosted on an AWS Elastic Compute Cloud (EC2) server.

At the moment, an administrator adds new users manually to screen potential users. In the future, we anticipate connecting the system to a multifactor authentication single sign-on system using Security Assertion Markup Language (SAML). User authentication is handled via the django-rest-knox library, which provides features such as encryption and token expiry. To authenticate, a user enters their email and password through the front end and is then posted to the API login endpoint. If the authentication is successful, the API returns a token back to the front end. This token is then included in the header of all subsequent requests to API endpoints to verify the user's authorization.

Upon authentication, the user can upload an R script through the web interface. After verifying that that uploaded file is the correct file type, the front end posts the file to the jobs endpoint along with an optional name that the user can assign to the submission through the interface. The API then triggers the creation of a new Job object and an initial Run object associated with the job. Each job can have multiple runs, reflecting multiple sets of epsilon values specified by the user for a given R script. After creating the new job, the API uploads the R script to an encrypted AWS Simple Storage Service (S3) bucket and triggers an AWS Step Function execution to start the run on the back-end engine using a default epsilon value of 1.0 divided equally across all statistics in the run. The engine posts status updates back to the API indicating if the run is running, completed, or failed. If the run successfully completed, the engine also posts the results to the runs endpoint, where the front end can request results to display through the web interface.

If the user decides to refine their results, the front end posts the updated epsilon values to the refine endpoint, which creates a new run for the job and invokes an AWS Lambda function to trigger the back-end engine. Once completed, the engine posts the results back to the API, as described in the previous paragraph.

If the user decides to release one or more analyses in the submission, the front end posts the runs and analyses that the user selected to the release endpoint. The API then generates a CSV file with the output that the user can download and emails the user indicating that their results have been released. In the future, we imagine that this could also trigger a manual review step if requested by agency data stewards before allowing users to remove results from the secure environment.

The API also implements a budget endpoint that allows the front end to query the user's available privacy budgets. The API calculates the cost of a run at the time of submission and will deny a run submission of the cost exceeds the user's available budget.

BACK END

The back-end system is built using the AWS Serverless Application Model (SAM). The system is centrally managed by an AWS Step Function, a serverless orchestration service that coordinates distributed and multistep application workflows in the cloud. When a user submits a new R script, the API copies the R script to an S3 bucket that is encrypted using AWS Key Management Service (KMS).

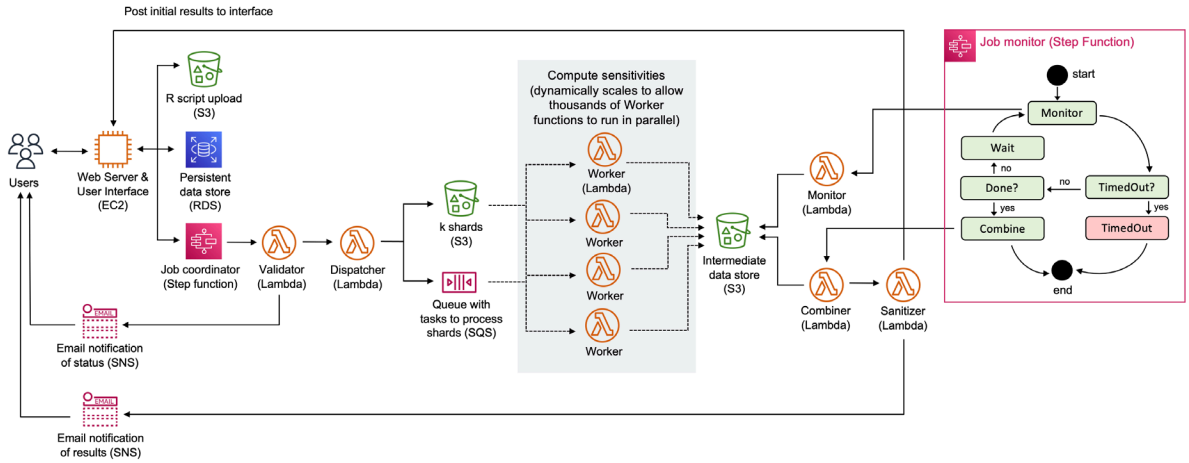
The API then triggers a new execution of the state machine (or Step Function workflow) with an event payload that includes the location of the R script on S3 and other relevant metadata to process the submission. The state machine then triggers a series of AWS Lambda functions based on the contents of the submission. The Lambda functions use a Python 3.9 runtime and a custom container image with R version 4.2.1, a predefined set of commonly used R packages, and rpy2 (a Python package that provides an interface to running R embedded within a Python process) already installed. This container image is hosted on Amazon Elastic Container Registry (ECR).

Upon a new job submission, the MOS-inspired algorithm is executed by the back end to compute sensitivities for all analyses contained in the R script. An architecture diagram of the back-end engine workflow when a user submits a new analysis is shown below (and described in detail in the following section).

FIGURE 5

Validation Server Version 2.0 Complete Architecture Diagram

System architecture for a new submission to compute initial results



URBAN INSTITUTE

Source: Authors' depiction of the validation server architecture.

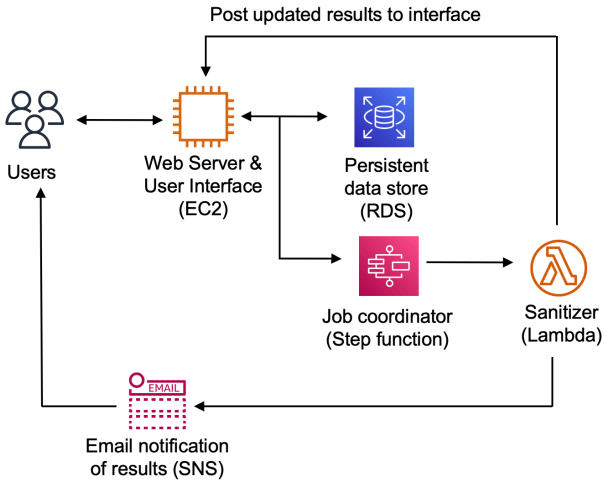
Notes: All AWS resources are hosted within a Virtual Private Cloud (VPC) and placed in a private subnet with restricted network access. The resources are encrypted using AWS Key Management Service (KMS). Access to the resources is managed through AWS Identify and Access Management (IAM). This diagram provides a simplified overview of the AWS resources in the validation server system. For detailed descriptions of the resources, complete source code, and the CloudFormation templates used to build the AWS infrastructure, please refer to the relevant GitHub repositories listed on page 17.

If the user is instead updating epsilon values to refine a previous submission, the "Sanitizer" Lambda function is called. The "Sanitizer" function computes updated results based on the previously computed sensitivities that are stored on S3 after initially computed for a given submission. This simplified workflow is represented in the diagram below.

FIGURE 6

Validation Server Version 2.0 Simplified Architecture Diagram

System architecture for an update to a previous submission to compute refined results



URBAN INSTITUTE

Source: Authors' depiction of the validation server architecture.

Notes: All AWS resources are hosted within a Virtual Private Cloud (VPC) and placed in a private subnet with restricted network access. The resources are encrypted using AWS Key Management Service (KMS). Access to the resources is managed through AWS Identify and Access Management (IAM). This diagram provides a simplified overview of the AWS resources in the validation server system. For detailed descriptions of the resources, complete source code, and the CloudFormation templates used to build the AWS infrastructure, please refer to the relevant GitHub repositories listed on page 17.

When processing a submission for the first time, the state machine will first trigger the "Validator" Lambda function to run the R script on the full confidential dataset. If the R script contained a syntax error or timed out, the engine would post the job status back to the API with the relevant error message and end the workflow execution. If the R script successfully ran on the full dataset, the true values of the estimates for the analyses would be stored on S3 and the "Dispatch" Lambda function would be triggered.

The "Dispatcher" Lambda function first splits the full dataset into non-overlapping randomly sampled subsets, drawing without replacement. These subsets are also written to S3. Then, the "Dispatcher" function dynamically computes the optimal number of tasks to calculate the sensitivities given the runtime of the user-submitted R script. Because Lambda functions have a maximum timeout of 900 seconds, this step ensures that analyses with longer runtimes are assigned more tasks, each processing fewer rows. Once the optimal number of tasks are computed, the "Dispatcher" function dispatches these tasks in the form of Simple Queue Service (SQS) messages, where each message assigns a subset and set of nonoverlapping rows in the subset for a "Worker" Lambda function to process. This provides a decoupled, scalable, and parallelized architecture where thousands of functions can launch within seconds, run simultaneously, and immediately scale back down once they have completed.

When a new task arrives in the SQS queue, a "Worker" Lambda function is automatically triggered to compute the maximum local sensitivity for each statistic based on the assigned rows and subset. When each "Worker" function finishes processing its assigned rows, it writes these sensitivities to S3. In the background, a "Monitor" Lambda function periodically determines if the "Worker" functions have all completed by comparing the number of SQS tasks that were initially dispatched to the number of results that have been written to S3 for a given submission.

Once the "Monitor" function determines that all tasks have completed, the "Combiner" function is triggered. This function combines the results from each "Worker" task to compute the overall maximum local sensitivity along with the other inputs required for the MOS-inspired algorithm to properly add noise to estimates.

Lastly, the state machine automatically triggers the "Sanitizer" function to add noise to the results using a default nominal epsilon value for the initial submission. As previously noted, when users refine an analysis, the state machine skips to this "Sanitizer" step and uses the MOS-inspired input values from the initial "Combine" function and the updated user-specified epsilon values. The "Sanitizer" function then posts the sanitized results back to the API in a JSON format, notifies the user that their results are available through an email notification, and updates the job status to indicate that the job has been completed. Similarly, if a job fails or times out, the user is notified, and the job status is updated through the API.

R PACKAGE

The R package, we call validationserver, lets users specify tabular and regression outputs to submit to the validation server system through two core functions:

- The get_table_output function lets users request one of more summary statistics (i.e., mean, median, sum, standard deviation, variance, minimum, maximum, number of observations, number of distinct observations, and quantiles), one or more variables to compute the summary statistic for, and one or more variables to optionally group by. Through this function, a user could request the mean and median earned income grouped by race and sex, for example.
- The get_regression_output function lets users request output from a regression analysis. Currently, users can pass any linear model or general linear model (i.e., lm or glm objects in R) into these functions, and the system will return all estimates included in the tidy and glance methods from the broom R package for the specified model.

These functions are necessary to properly compute local sensitivities and apply the correct amount of expected noise for the analyses; however, users can include virtually any preprocessing code in their R scripts before calling these functions from the validationserver package. For example, users could subset the data, create new variables, or perform any other data transformations within their R scripts. Users can also include multiple analyses within a single R script to closely resemble typical programming workflows.

Right now, a limited set of external R packages are preinstalled in the validation server system. In the future, we imagine that data stewards could establish processes to allow users to request additional packages be installed in the system that would go through a vetting process similar to other secure analytical environments.

How Another Organization Might Set up and Configure the System

SOURCE CODE AND INFRASTRUCTURE

Like its predecessor, the validation server version 2.0 prototype is built and deployed in a cloud-native manner, hosted on the AWS cloud compute platform following an "infrastructure as code" development approach. The AWS services used in the system comply with FedRAMP High Baseline standards, ensuring the system could be easily adopted by government agencies and their partners in the future. The system is composed of five GitHub repositories:

Infrastructure: The repository for the AWS cloud infrastructure can be used as a guide to create a set of cloud resources for the overall validation server system. This repository includes an AWS CloudFormation template and a deploy script. This CloudFormation template creates the following AWS resources: a MySQL RDS to hold the API model objects, an EC2 instance to host the API and front end, a KMS key to facilitate encryption, and an Identity and Access Management (IAM) role to allow the API to securely interact with the back end.

Front end: The codebase for the front-end interface is hosted in another repository. This repository contains a Dockerfile and docker-compose file for creating a containerized version of the application that can be run locally for testing or deployed on a server. To manage the deployment of the front-end application, we use CircleCI, a continuous integration and continuous deployment (CI/CD) platform. When changes to the codebase are committed and pushed to this repository, the CI/CD system will rebuild and redeploy the application container on the server that hosts the site.

API: The codebase for the Django REST API similarly contains a Dockerfile and docker-compose files for creating a containerized version of the API that can be run locally for testing or deployed to a server. The docker-compose files configure additional containers that make up the full API service. The deploy.sh script contains the steps for deploying this system using the docker-compose tool. This repository is also set up with CircleCI for CI/CD, so when code changes are committed and pushed, the docker-compose stack is rebuilt and redeployed.

Back end: The codebase for the back-end engine is built using the AWS SAM framework. This repository contains the application source code along with a CloudFormation template to deploy the application including the following AWS resources: Lambda functions that implement the back-end engine, a Step Function to coordinate the Lambda functions, an SQS queue to assign tasks to the Lambda functions, an ECR repository with the Docker images used by the Lambda functions, an encrypted S3 bucket to store data files and R scripts, and IAM policies and roles to assign appropriate permissions to each of these services. When changes to the codebase are committed and pushed up to

the repository, a GitHub Actions workflow packages the SAM stack and deploys the serverless back-end engine to AWS.

R package: The codebase for the validationserver R package is built using a standard R package structure. This repository contains the source code for the R package along with a vignette and additional documentation describing how to use the package to specify tabular and regression analyses to submit to the validation server prototype.

HOW ANOTHER ORGANIZATION MIGHT MODIFY THE SYSTEM AND FUNCTIONALITY

We purposefully separated the privacy-preserving validation server technology stack into separate containerized pieces so each could be modified with minimal disruption to the overall system.

The front-end interface interacts with the rest of the validation server system via the Django REST API. This means the interface could potentially be modified, and as long as the API endpoints retain their general format, the rest of the system should continue to operate as intended.

Right now, the privacy-preserving back end runs the user-submitted code against a public-use file hosted in an encrypted S3 bucket. This could be supplemented or replaced with any other dataset with virtually no impact on the rest of the system.

The system could also accommodate expanded functionality, such as additional programming languages for user-uploaded analysis code (such as Python or Stata) or additional types of analyses beyond the set of regression and tabular analyses currently supported with minimal changes to the API or front end. Similarly, the back end could be updated to implement alternative privacy algorithms (or multiple privacy algorithms) without requiring substantive updates to the front end or API due to the flexibility and scalability of the system's architecture.

Limitations and Future Directions

The privacy-preserving validation server version 2.0 prototype is currently built to facilitate testing and improvement and has several limitations. It does not do the following:

- Allow researchers to more naturally incorporate survey weights into their analyses. Work with weighted datasets is possible for certain simpler calculations, but even these calculations are neither efficient nor ideal.
- Allow researchers to join external data to the validation server data to incorporate into their analyses.
- Accept input other than a single R script (i.e., researchers may not upload multiple R scripts in a single submission or upload code in other programming languages, such as Python or Stata).
- Allow researchers to define analyses using fully arbitrary R code (i.e., analyses must be defined using functions in the validationserver R package to ensure the appropriate amount of noise can be added to estimates and limit the security risks).

- Have robust learning libraries for privacy budgets, differential privacy or other formally private definitions, and other key concepts that researchers may not be familiar with.
- Include an interface for data stewards, such as federal statistical agency representatives, to more easily monitor activity, review output, and interact with users (e.g., to refresh privacy budgets, review requests to release output, view a user's submission history, and more).

In the future, plan to address several of these shortcomings and conduct additional red teaming to further harden the security of the system. There are also several challenges that the privacy experts on the Safe Data Technologies team will continue to explore as the privacy field advances. The following are some of the many that have been identified and plan to prioritize in our next update of the prototype:

Appropriately displaying errors in user-submitted code: Error messages are critical for helping users diagnose and resolve syntax errors in their code, but they can also reveal sensitive information about the underlying data (e.g., the presence or absence of particular observations). One potential solution is ensuring the synthetic datasets that users can access mirror the structure of confidential datasets as well as the range of possible values to develop and debug their code in a familiar environment outside of the validation server interface. Therefore, we assume that very few, if any, new errors will be identified after scripts are uploaded to the validation server. In other words, users should not be using the validation server for exploratory data analysis.

Ensuring the correct amount of noise is added for complex analyses: In the version 2.0 prototype, we implement a generalized version of a local sensitivity approach that is inspired by MOS to add noise to estimates from a range of regression and tabular analyses. In the future, we plan to conduct further research to better understand the privacy and statistical implications of this approach. Insufficient research has been done on local sensitivity-based privacy methods and whether the improved accuracy is sufficient for public policy decisionmaking.

We also plan to explore the implications of the interim budget, which allows users to run an analysis on a dataset multiple times by specifying different levels of epsilon and receive results with different amounts of noise. Although users should conduct most of their exploratory data analysis on the synthetic data, we initially proposed an interim budget to provide some flexibility for users to refine their analyses within the validation server. We need to conduct further tests to see if creating this interim budget provides the benefits we believe or creates further confusion for the user at the cost of increased privacy risk.

Speeding up time-intensive analyses on big datasets: Many privacy-preserving algorithms are time- or compute-intensive. The current prototype uses a distributed approach to compute sensitivities that can quickly and cost-effectively accommodate many of these algorithms. However, the current system has yet to be tested with multiple users running large workloads, so further testing is needed to understand the limits of scaling within our current architecture.

Conclusion

The goal of our work is to continue to provide improvements and steps forward for practitioners looking to improve on manual disclosure review and rules of thumb and reliably automate steps in that process. We use a mix of different privacy enhancing technologies, such as synthetic data and an MOS-inspired approach, to build our current validation server system and take the next step forward.

We have found that formally private methods do not achieve what researchers might consider reasonable levels of utility for many real-world analyses even with very generous privacy loss budgets. In these cases, it is not clear that disclosure risks are unacceptably large. We look forward to including improved formally private methods as privacy experts bridge the gap between theory and practice.

Having a fully operational system allows us to work with researchers and data stewards on implementation issues, conduct further user testing to address outstanding privacy questions, and work on other challenges. For example, what if two users submit the same analysis and obtain different results that are both valid? How might an agency manage a privacy budget? How might alternative privacy enhancing techniques and software architectures better serve researcher and data steward needs? Creating this version 2.0 of our validation server allows us to continue to test these challenges on a real system, build trust with practitioners, and ensure the validation server can scale as the data privacy field continues to evolve.

Notes

1 "Safe Data Technologies," Urban Institute, accessed September 4, 2024, https://www.urban.org/projects/safe-data-technologies.

2 Foundations for Evidence-Based Policymaking Act of 2018, Pub. L. No. 115-435, 132 Stat. 5529 (2019).

3 "Annual Social and Economic Supplements," US Census Bureau, accessed September 4, 2024, https://www.census.gov/data/datasets/time-series/demo/cps/cps-asec.html.

4 "SmartNoise," accessed September 4, 2024, https://smartnoise.org.

5 "Graphicacy," accessed September 4, 2024, https://graphicacy.com.

6 "AWS Services in Scope by Compliance Program – Federal Risk and Authorization Management Program (FedRAMP)," Amazon Web Services, accessed September 4, 2024, https://aws.amazon.com/compliance/services-in-scope/FedRAMP.

References

Advisory Committee on Data for Evidence Building. 2022. *Advisory Committee on Data for Evidence Building: Year 2 Report*. Suitland, MD: Bureau of Economic Analysis.

Barrientos, Andrés F., Aaron R. Williams, Joshua Snoke, and Claire McKay Bowen. 2023. "A Feasibility Study of Differentially Private Summary Statistics and Regression Analyses with Evaluations on Administrative and Survey Data." *Journal of the American Statistical Association* 119 (545): 52–65. https://doi.org/10.1080/01621459.2023.2270795.

Chetty, Raj, and John N. Friedman. 2019. "A Practical Method to Reduce Privacy Loss when Disclosing Statistics Based on Small Samples." Working Paper 25626. Cambridge, MA: National Bureau of Economic Research.

Commission on Evidence-Based Policymaking. 2017. *The Promise of Evidence-Based Policymaking*. Washington, DC: Commission on Evidence-Based Policymaking.

Pickens, Madeline, Jennifer Andre, and Gabriel Morrison. 2023. "Synthetic Data User Guide and Infographic." Washington, DC: Urban Institute.

Taylor, Silke, Graham MacDonald, Kyle Ueyama, and Claire Bowen. 2021. "A Privacy-Preserving Validation Server Prototype." Washington, DC: Urban Institute.

Vilhuber, Lars, and John M. Abowd. 2016. "Presentation: SOLE 2016: Usage and outcomes of the Synthetic Data Server." Presented at the Society of Labor Economists Annual Meeting in Seattle, WA on May 6.

Williams, Aaron R., Joshua Snoke, Claire McKay Bowen, and Andrés Felipe Barrientos. 2024. "Disclosing Economists' Privacy Perspectives: A Survey of American Economic Association Members' Views on Differential Privacy and the Usability of Noise-Infused Data." *Harvard Data Science Review*. https://doi.org/10.1162/99608f92.a8fb0371.

About the Authors

Erika Tyagi is a lead data engineer at the Urban Institute, where she contributes to a variety of projects aimed at democratizing access to data. This includes building open data portals, platforms to safely expand access to confidential data, data-intensive applications, and more. She has particular expertise leveraging cloud technology to develop cost-effective, secure, and scalable data systems.

Silke Taylor is the associate director of software engineering in the Technology and Data Science office at the Urban Institute. In her work, she focuses on providing advanced technological solutions in cloud computing, API development, and parallel computing. Her extensive experience also includes developing applications that meet federal security standards, such as FedRAMP.

Graham MacDonald is the chief information officer and vice president of technology and data science at the Urban Institute, where he leads the strategic planning and implementation of research, operations, and communications technology and works with staff across Urban to improve access to data, analytics tools, and innovative research methods.

Deena Tamaroff is a lead user experience designer at the Urban Institute. Her work involves the application of user research insights and user-centered design principles to the development of Urban Institute tools and products.

Josh Miller, director of web development at the Urban Institute, is a highly skilled and experienced web developer who has worked on complex web projects for over two decades. He served as a lead programmer and product owner on various web development projects for educational institutions and economic development organizations. Miller has extensive experience handling personal identifiable information disclosures, which pose a significant security risk for online retailers. He successfully led efforts to reduce personal information and ensure that they are blurred or removed as appropriate. He also implemented several accessibility improvement projects, many of which were 508 compliant.

Aaron R. Williams is a lead data scientist for statistical computing in the Income and Benefits Policy Center at the Urban Institute. He works on data privacy, data imputation, microsimulation modeling, and survey analysis with a focus on income, wealth, tax, and retirement policies. Williams has developed systems for safely releasing administrative data for research, including the Urban-Brookings Tax Policy Center's synthesis of individual tax records and the US Department of Labor's Safe Transfer, Restricted-Use Data Lake. He is a member of the Bureau of Labor Statistics Technical Advisory Committee, an adjunct professor in the McCourt School of Public Policy at Georgetown University, and the leader of Urban's R Users Group.

Claire McKay Bowen is a senior fellow at the Urban Institute. Her research focuses on developing technical and policy solutions aimed at safely expanding access to confidential data for advancing evidence-based policymaking. She also has interest in improving science communication and integrating data equity into the data privacy process. In 2024, she became an American Statistical Association Fellow "for her significant contributions in the field of statistical data privacy, leadership activities in support of the profession, and commitment to mentoring the next generation of statisticians and data scientists." Further, she is a member of the Census Scientific Advisory Committee and several other data governance and data privacy committees as well as an adjunct professor at Stonehill College.

Acknowledgments

The validation server version 2.0 prototype is a product of the Urban Institute's Safe Data Technologies Project. More information on the project is available at urban.org/projects/safe-data-technologies. This tool was funded by the National Science Foundation National Center for Science and Engineering Statistics (NCSES) [49100422C0008]. Original funding and support for the initial research and version 1.0 prototype was provided by the Alfred P. Sloan Foundation [G-2022-17149], Arnold Ventures, the Microsoft SmartNoise Early Adopter Accelerator Program, and NCSES. We are grateful to all our funders, who made it possible for the Urban Institute to advance its mission.

We are also thankful to our dedicated partners at the Statistics of Income Division of the IRS; our group of testers who provided early feedback; Graphicacy, for the design and front-end build; the entire Safe Data Technology project team; and our advisory board of key representatives and top experts across industry, academia, and government for their input in this work.

The version 2.0 validation server prototype would not exist without the foundation laid by the team that developed the version 1.0 prototype: Silke Taylor, Graham MacDonald, Kyle Ueyama, and Claire McKay Bowen; the Harvard OpenDP and Microsoft SmartNoise teams who provided technical assistance and opensource libraries; and Forum One who led design and development for the version 1.0 front end build. Several sections of this white paper were initially developed for the version 1.0 working paper and repurposed here.

The views expressed are those of the authors and should not be attributed to the Urban Institute, its trustees, or its funders. Funders do not determine research findings or the insights and recommendations of Urban experts. Further information on the Urban Institute's funding principles is available at urban.org/fundingprinciples.

The authors welcome feedback on this paper. Please send all inquiries to validationserver@urban.org.

500 L'Enfant Plaza SW
Washington, DC 20024
www.urban.org

ABOUT THE URBAN INSTITUTE

The Urban Institute is a nonprofit research organization that provides data and evidence to help advance upward mobility and equity. We are a trusted source for changemakers who seek to strengthen decisionmaking, create inclusive economic growth, and improve the well-being of families and communities. For more than 50 years, Urban has delivered facts that inspire solutions—and this remains our charge today.