# Automation and Collaboration in Complex Epidemiological Workflows with OSPREY

Jonathan Ozik
Nicholson Collier
Arindam Fadikar
Justin M. Wozniak
jozik@anl.gov
ncollier@anl.gov
afadikar@anl.gov
woz@anl.gov
Argonne National Laboratory
Lemont, Illinois, USA

Valérie Hayot-Sasson
Kyle Conroy
Kyle Chard
valeriehayot@gmail.com
kyleconroy@bsd.uchicago.edu
chard@uchicago.edu
University of Chicago
Chicago, Illinois, USA

Jacqueline M. Wentz
Erin Acquesta
Jaideep Ray
jwentz@sandia.gov
eacques@sandia.gov
jairay@sandia.gov
Sandia National Laboratory
Livermore, California, USA

## Abstract

The Open Science Platform for Robust Epidemic Analysis (OSPREY) was introduced to address critical gaps in applying high-performance computing (HPC) to epidemiologic modeling. While the initial implementation focused on integrated, algorithm-driven HPC workflows, this paper explores the second and third goals of OSPREY: data ingestion, curation, and management, and the development of a Shared Development Environment (SDE) for rapid response and collaboration. We present two use cases to demonstrate the implementation and impact of each of these goals. The first use case highlights real-time data ingestion and curation for epidemiologic modeling, while the second focuses on developing efficient global sensitivity analyses for epidemic models. These advancements aim to enhance OSPREY's capabilities for supporting public health decision-making and reproducible science.

## CCS Concepts

• **Information systems** → **Decision support systems**; • **Computing methodologies** → **Modeling and simulation**.

## Keywords

HPC workflows, epidemiological modeling, automation, model exploration

**ACM Reference Format:**
Jonathan Ozik, Nicholson Collier, Arindam Fadikar, Justin M. Wozniak, Valérie Hayot-Sasson, Kyle Conroy, Kyle Chard, Jacqueline M. Wentz, Erin Acquesta, and Jaideep Ray. 2018. Automation and Collaboration in Complex Epidemiological Workflows with OSPREY. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX


*Conference acronym 'XX, Woodstock, NY*

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

## 1 Introduction

The COVID-19 pandemic revealed critical gaps in the ability of researchers to leverage advanced computational systems for epidemic analysis. While epidemiological modeling has long been a cornerstone in supporting public health decision-making, the unprecedented scale and complexity of the pandemic underscored the need for more robust, scalable, and collaborative approaches to modeling and analysis. The Open Science Platform for Robust Epidemic Analysis (OSPREY) [8] was introduced to address these challenges by lowering barriers to high-performance computing (HPC) resources, automating workflows, and enabling rapid response capabilities.

An integral part of OSPREY design was the development of central goals for enabling timely and robust public health decision support:

(1) Integrated, algorithm-driven HPC workflows: OSPREY aims to streamline access to HPC resources for developers of epidemiological models and model exploration algorithms [19]. It needs to integrate data, simulations, and algorithms while allocating heterogeneous resources (CPU, GPU, and accelerators) based on task needs.
(2) Data ingestion, curation, and management: OSPREY needs to support continuous data assimilation by integrating real-time data streams with models. It needs to ingest, curate, store, and index data while managing models and outputs, ensuring data quality and provenance.
(3) Shared Development Environment for rapid response and collaboration: OSPREY needs to enable rapid, collaborative development and efficient porting of modeling and model exploration codes to HPC, offering a shared, automated, and scalable framework for model exploration in a Shared Development Environment (SDE).

These goals were informed and refined by our team's work in supporting public health stakeholders during the COVID-19 pandemic [17, 20] and beyond [12, 16]. The initial implementation of OSPREY focused on the first goal, prototyping integrated, algorithm-driven multi-facility HPC workflow capabilities [8].

This paper explores the second and third goals of OSPREY: data ingestion, curation, and management, and the development of a SDE for rapid response and collaboration. These goals are motivated

by two aspects of epidemiological modeling and public health decision support that could benefit from automation and time-sensitive workflows, metadata and provenance management, and fast turnaround prototyping on heterogeneous computing resources.

First, the dynamic nature of public health data, characterized by heterogeneous, incomplete, and rapidly changing data streams, poses significant challenges for epidemiological modeling. Researchers often struggle to integrate diverse data sources, quantify uncertainty, and ensure data provenance, all while maintaining the computational efficiency required for real-time decision support.

Second, during the COVID-19 pandemic the collaborative nature of epidemic analysis became increasingly apparent as researchers across disciplines worked together to refine models, validate results, and share insights. Despite this collaboration, differences in HPC environments, programming languages, workflow structures, and data formats hindered the reproducibility and scalability of shared modeling and collaboration.

Here we focus on advancements in OSPREY's capabilities in these two areas. Specifically, we present two use cases that demonstrate implementations of OSPREY's second and third goals. The first use case highlights automated, event-driven data ingestion, curation, and analysis for local epidemic modeling, showcasing how OSPREY facilitates the integration of diverse data streams to support rapid, always-on analyses. This use case relies on the AERO event-driven automation framework [16]. The second use case demonstrates the flexibility of OSPREY to enable rapid integration of epidemiological models and complex algorithms for analyses. This use case builds on the latest capabilities of the EMEWS multi-language model exploration framework [7]. Both AERO and EMEWS were developed and have evolved in response to use-inspired requirements gathered through existing public health collaborations and stakeholder workshops [10, 21, 22].

Through these use cases we show how OSPREY aims to enhance the ability for epidemiological modeling to support public health decision-making during crises and improve the reproducibility and scalability of epidemic analyses. The advancements presented in this paper represent a significant step toward realizing OSPREY's vision of an open science platform that empowers researchers to respond to public health emergencies with speed, accuracy, and collaboration.

The remainder of this paper is organized as follows. In §2, we describe the data ingestion, curation, and management use case, using aggregation of wastewater-based effective reproduction number $R(t)$ as the application. In §3, we describe a surrogate-based approach to improve the sample efficiency of Global Sensitivity Analysis (GSA) of a meta-population epidemiological model, including the analysis of uncertainties arising from both model parameters and stochastic model processes. We conclude in §4 and identify future directions.

## 2 Automation for Data Ingestion, Curation, and Management Use Case

The first use case addresses the need for increased automation to streamline model-based analysis support for public health decision making. In order to facilitate the integrating of diverse data sources, quantify their uncertainty, ensure data provenance, and make the data directly usable by potentially complex epidemiological analyses we present an open-source hybrid and asynchronous data research automation platform called Automated Event-based Research Orchestration (AERO). AERO is implemented as a distributed platform, storing metadata centrally and integrating distributed user-owned and -managed resources for data storage and workflow execution. We apply AERO to inferring epidemic trends via noisy passive surveillance signals, namely wastewater data from Chicago-area water reclamation plants.

### 2.1 Monitoring the effective reproduction number, $R(t)$, via concentrations across wastewater treatment plants

$R(t)$ is a time-varying quantity that represents, on average, the number of new cases caused by an already-infected individual throughout the span of their illness. It is a useful epidemic quantity for detecting trends in community disease transmission and informing policy interventions, and it is closely monitored by public health officials throughout an epidemic. $R(t)$ can be estimated using a variety of methods and data sources [14]. However, the mandates that ensured consistent access to updated COVID-19 surveillance datasets have ended, and many of the datasets that had previously been used for inputs into the estimation of $R(t)$, such as COVID-19 cases and hospitalizations, are no longer actively maintained. Nonetheless, COVID-19 and other respiratory diseases remain a public health risk and monitoring $R(t)$ in the absence of reporting mandates is essential. Passive surveillance indicators, such as pathogen concentrations in wastewater, do not require populations to opt-in and can always be monitored. However, the signal from such data streams is noisy and subject to complicated dynamics, which increases the difficulty in obtaining clear estimations of $R(t)$. Here we address this difficulty in two ways. First, we leverage a compartmental model-based $R(t)$ estimation framework [13] (Goldstein method). This method combines a mechanistic epidemiological model and a separate statistical model of the observed pathogen genome concentrations in wastewater. $R(t)$ is estimated as a posterior distribution using a semi-parametric Bayesian sampling framework. This estimation procedure is significantly more computationally expensive than more standard $R(t)$ estimation methods (e.g., [9]) and, therefore, can benefit from HPC resources. Second, we pool estimates across multiple wastewater sources and use a population-weighted ensemble average to improve the $R(t)$ signal to noise. Automating this approach leverages the event-based triggers and metadata management provided by the AERO framework [16].

### 2.2 Combined WW workflow implementation

In previous work, we adapted the Golstein method's manual process [13] into an automated workflow using the Illinois Wastewater Surveillance System wastewater data [1] from the O'Brien water reclamation plant [16]. Our current work incorporates three additional wastewater sources: the Calumet, Stickney South, and Stickney North water reclamation plants, and performs an additional aggregate analysis. The workflow consists of three steps: 1) a data ingestion and preprocessing step, 2) an execution step where the $R(t)$ analysis model is run, and 3) an aggregate analysis

step that aggregates the four water reclamation plant analyses. The first two steps are run as four individual workflows for each of the wastewater sources, and the final step when all of those four individual $R(t)$ analyses have produced new data (see Figure 1).

The first step is implemented as an AERO ingestion flow using the AREO Python API. AERO will poll the wastewater data source at a user specifiable frequency, in this case daily. If there is a data update, the new data is uploaded to a user-specifiable Globus collection, where we used the Argonne Leadership Computing Facility (ALCF) Eagle Globus endpoint. The data is also temporarily sent to a user-specifiable Globus Compute [4] endpoint, in our case the Argonne Laboratory Computing Resource Center (LCRC) Bebop cluster, where the validation and transformation function is run with the data as input. The transformed data file is then uploaded to the Globus endpoint. Versioning metadata, such as a checksum, a timestamp, and version number is stored in the AERO metadata database both for the input and transformed data. Note that the data itself never passes through the AERO server, only the metadata.

When registering an ingestion flow using the AERO API, a user specifies the polling frequency, a URL from which to retrieve the data, a function to run on the data, any other arguments to that function, and a Globus Computer [4] endpoint where the function will run. The AERO API wraps the function call with additional code that 1) performs the data retrieval from the Globus endpoint, staging that data as input; 2) calls the user-specified function, in this case the data transformation function; 3) uploads any outputs of the function to a specified Globus endpoint; and 4) updates the AERO database with the relevant metadata for the input and output data. The registration returns one or more UUIDs that uniquely identify the output data. These UUIDs can then be used to specify that data as input to an AERO analysis flow. When the data identified by that UUID is updated, then any analysis flows that have registered that UUID as input are triggered by AERO.

For the second step, the $R(t)$ analysis model is implemented as an AERO analysis flow using the UUID of the wastewater data transformation output as its input. Consequently, when a wastewater source is updated, triggering a validation and transformation, it produces an updated output. This, in turn, triggers the analysis flow that depends on the data source. Registering an analysis flow is much like registering an ingestion flow, but rather than a URL, data UUIDs are specified as inputs. If there are multiple input UUIDs, the user can specify that the analysis function should be run when either one or all of the inputs are updated.

Similar to the ingestion flow, a user-specified analysis function is wrapped by AERO code so that the input data is downloaded and any output is uploaded to a Globus endpoint, and the metadata is entered and updated in the AERO database. In our use case, when the $R(t)$ analysis flow is triggered, the updated data is downloaded from the ALCF Eagle Globus endpoint to a temporary location on LCRC Bebop. There, a Python code harness function, previously registered with the analysis flow, executes a Julia code $R(t)$ estimation and then executes R code to create the $R(t)$ plots and R data objects from the tabular data produced by the estimation. AERO then stores the model's tabular data, binary R datatable objects, and plots to the ALCF Eagle Globus endpoint.

As in the first step, the registration of the second step returns UUIDs for the outputs of the $R(t)$ analysis, and these can be used as inputs to additional flows. In the third step, an analysis flow that performs aggregation over the four $R(t)$ analysis flow outputs takes the UUIDs associated with the binary datatable objects produced in the second step as inputs. When all of these data sources have been updated, a simple Python harness calls an R function which performs the aggregation, producing an aggregate plot of population-weighted $R(t)$ as output (see Figure 2). As in the previous steps, the AERO API wraps this Python harness function in code that both retrieves the input from and stores the output to the ALCF Eagle Globus endpoint.

All three of the flows use Globus Compute [4] to execute their respective functions. As mentioned above, when registering the flow, both the function and the endpoint on which to run it are specified. The data transformation function and the aggregating function executed in the first and third steps respectively were run on a Globus Compute endpoint configured on a login node on the Bebop cluster. The computation expense of the transformation and aggregation steps are low, both tasks running in under a minute, and so a shared login node can be used. The analysis function run in the second step is computationally expensive and it is run using a Globus Compute endpoint configured for a compute node using the GlobusComputeEngine. When AERO triggers the analysis flow, Globus Compute will queue a job on Bebop's PBS scheduler to run the function on one node.

The process described in this use case and depicted in Figure 1 is fully automated, and the outputs are directly shareable with public health stakeholders through standard Globus Collection permissions. This removes unnecessary manual intervention and provides data update-driven and timely model-based epidemiological analyses for the assessment of epidemic trends to directly support public health decision making. It also leverages the "bring your own storage and compute" design of AERO, enabling the use of existing storage and compute infrastructure access that research groups already possess. The approach relies on the security and robustness of Globus technologies such as Globus Auth [23], Flows [5], and Timers [2].

## 3 Shared Development Environment Use Case

OSPREY's third goal focuses on the need for a Shared Development Environment (SDE). The SDE needs to enable rapid, collaborative development and efficient porting of modeling and model exploration (ME) codes to HPC. It needs to consider differences in HPC environments, programming languages, and workflow structures to support reproducibility and scalability of epidemiological analyses. Some aspects of the SDE were covered in the previous section, namely the portability of AERO workflows on HPC systems to which the user already has access. In this use case we further expand on how we have been supporting the OSPREY SDE goal and demonstrate a complex, multi-component workflow. More specifically, this workflow faced: 1) the need for a large-scale and efficient implementation of a many-task Global Sensitivity Analysis (GSA) described below; 2) rapid prototyping in R-based algorithms across multiple variations; and 3) the integration of a previously developed epidemiological model. The use case thus illustrates three implicit drivers of SDE capabilities: flexibility, collaboration, and asynchronicity.

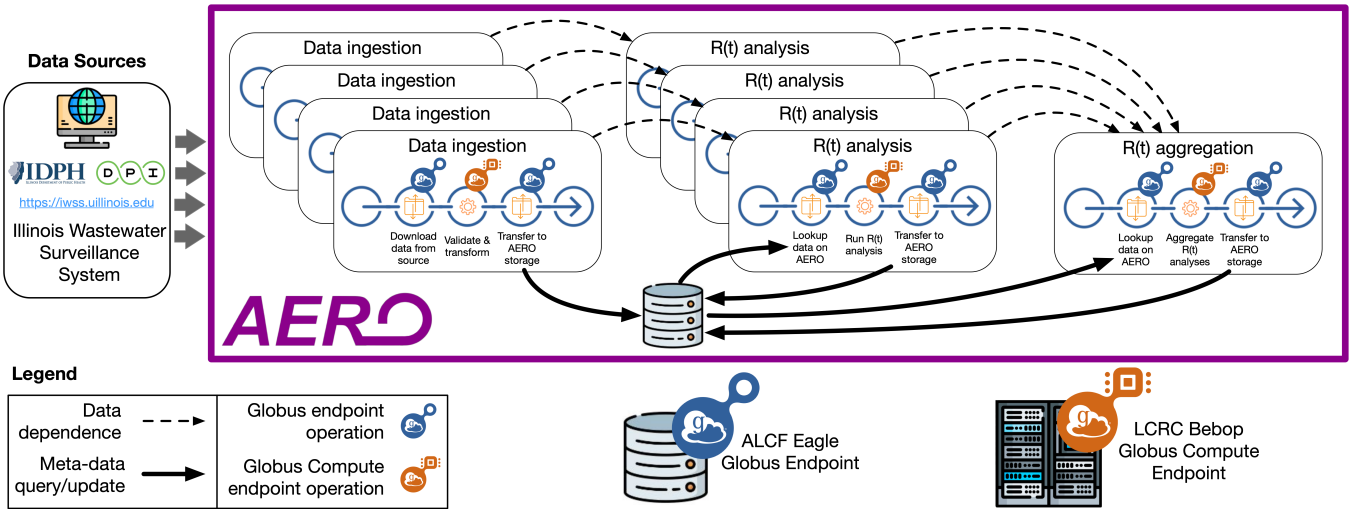


**Figure 1: Automated multi-source wastewater $R(t)$ estimation workflow. The data dependencies between the data ingestion, $R(t)$ analysis, and $R(t)$ aggregation flows (i.e., collection of tasks) are depicted with dashed arrows. Meta-data queries and updates to the AERO server are shown as solid arrows. Tasks within each flow are adorned with badges indicating operations that utilize "bring your own storage and compute" resources, in our case the ALCF Eagle Globus endpoint for storage and the LCRC Bebop Globus Compute endpoint for compute. Updates in the data from the Illinois Wastewater Surveillance System trigger the data ingestion and subsequent analysis flows.**

The flexibility requirement in this workflow is exemplified by the need for integrating existing multi-language libraries and models not only as workflow tasks, which is handled by most workflow systems, but also to drive the logic of the workflow itself. For the latter, existing solutions in this space either have limited capabilities for defining complex and iterative workflow logic, or are focused on single language environments, typically Python.

The collaboration requirement in this workflow is exemplified by the use of disparate software systems that work together. These include components in Python and R, as well as complex systems such as Globus and system schedulers. The models and algorithms were originally developed previously, although development continued as a part of this effort. The combination of independent development with collaborative interaction is an aspect of software engineering in general, but is especially emphasized in workflows.

The asynchronous requirement in this workflow is exemplified by the need for ensuring high compute resource utilization through complex interleaving of different sized sequential tasks. This is enabled by the decoupled design of the underlying workflow system.

## 3.1 Global Sensitivity Analysis of a Stochastic Disease Model

*3.1.1 MetaRVM.* The SEIR (Susceptible Exposed Infectious Recovered) model is a widely used compartmental framework in epidemiology that captures the basic progression of infectious diseases. It classifies individuals into four states: Susceptible (S), who can contract the disease; Exposed (E), who have been infected but are not yet infectious; Infectious (I), who can spread the disease; and Recovered (R), who have gained immunity. Despite its simplicity and utility, the basic SEIR model lacks the granularity needed to represent complex disease dynamics such as varying symptom severity, healthcare interactions, and population heterogeneity. The MetaRVM [12] model extends the SEIR framework by introducing additional compartments to capture more detailed disease progression and heterogeneous mixing across demographic subgroups. It also accounts for vaccination, hospitalization, and fatalities (see Figure 3 for the MetaRVM compartments). Typically, nearly all individuals are classified as Susceptible (S) at the beginning of a simulation, except for a usually small number of the initial infections. Vaccinated individuals enter the Vaccinated (V) compartment, with immunity determined by the vaccine efficacy parameter ($ve$), and face a reduced probability of infection. Vaccine-conferred immunity wanes at a rate of $1/dv$. Upon exposure, Susceptible and Vaccinated individuals transition to the Exposed (E) state, remaining there for an average of $de$ days before becoming infectious. A proportion ($pea$) of exposed individuals become Asymptomatic (Ia), while the remainder enter the Presymptomatic (Ip) state, lasting for $da$ and $dp$ days, respectively. Asymptomatic individuals recover directly, whereas Presymptomatic individuals progress to the Symptomatic (Is) state. Symptomatic individuals either recover (R) or require hospitalization (H), with transitions occurring at rates of $1/dp$ and $1/ds$, and the probability of direct recovery given by $psr$. Hospitalized individuals remain in the H compartment for an average duration of $dh$ days, after which they either recover or die (D), with the probability of death determined by $phd$. For diseases where reinfection is possible, Recovered individuals may return to the Susceptible (S) state after an average of $dr$ days.

Before applying MetaRVM to real-world data, it is essential to understand how uncertainty in model parameters influences the model outputs. This is accomplished through Global Sensitivity Analysis (GSA), a systematic approach for quantifying the relative

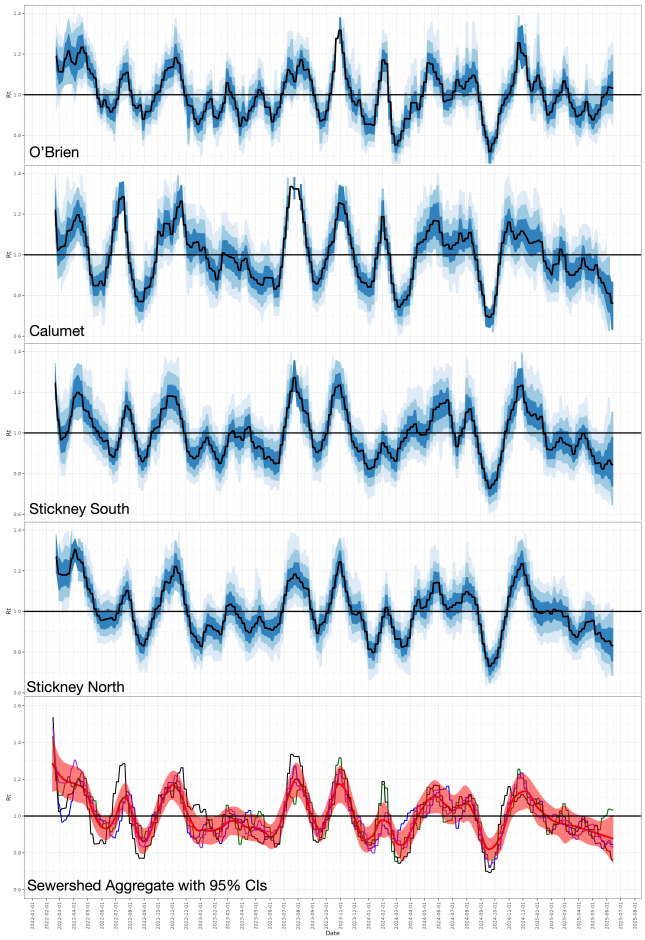


**Figure 2: Automatically generated estimates of $R(t)$ using the Goldstein method [13] for four Chicago-area water reclamation plants: O'Brien, Calumet, Stickney South, and Stickney North. The bottom panel is the output of the population-weighted ensemble $R(t)$ that is triggered when the four individual $R(t)$ estimates are completed. The red band is the 95% confidence interval.**

importance of input parameters on the variability of key model outputs. Unlike local sensitivity analysis, which examines small perturbations around fixed values, GSA explores the full range of plausible parameter values, accounting for inter-parameter interactions and nonlinearities. Given the complex structure and high-dimensional parameter space of MetaRVM, GSA helps identify the most influential parameters, facilitates dimensional reduction to aid in model calibration efforts, and can also inform data collection priorities. Sobol sensitivity analysis [18] is a variance-based GSA method that decomposes the total variance of the model output into contributions from individual input parameters and their higher-order interactions. The Sobol indices provide a quantitative measure of each parameter's effect: the first-order index reflects the main effect of a single parameter, while total-order indices capture both main and interaction effects.

*3.1.2 MUSIC-GSA.* In this illustration, we adopt the active learning–based GSA algorithm introduced by Chauhan et. al. [6], which uses a Gaussian process (GP) surrogate model [15] trained on a limited number of simulations to efficiently estimate first order Sobol sensitivity indices. Unlike conventional sampling strategies that may require a large number of simulations to achieve accurate variance decomposition, this method actively selects new input locations to improve the surrogate model where it matters most for estimating sensitivity indices.

Central to the method is the MUSIC (Minimize Uncertainty in Sobol Index Convergence) acquisition function, which specifically targets the reduction of uncertainty in the variance of the estimate in main-effects. In particular, the EIGF–Expected Improvement in Global Fit–acquisition function is used in this particular illustration. This contrasts with more common acquisition functions like EI (Expected Improvement) and UCB (upper confidence bound) [15], which focus on minimizing prediction error in global surrogate prediction. By refining the surrogate in a goal-directed manner, MUSIC improves convergence rates in estimating main effects, and has the potential to offer computational savings by reducing the number of simulations needed compared to other approaches for GSA. In our use case we apply the MUSIC active learning–based GSA framework to the MetaRVM model and compare it to a polynomial chaos expansion (PCE) approach for sample efficiency.

In stochastic simulation models, GSA is often performed on the mean response, calculated across multiple replicates (i.e., stochastic variations) for a given parameter setting. However, stochastic processes can embody important information beyond just noise, e.g., a particular distribution of initial infections in a population, and can provide meaningful insights into a system's behavior. As a result, we seek to distinguish between two types of uncertainties: aleatoric uncertainty, which arises from the stochastic processes in the system, and epistemic uncertainty, which stems from incomplete knowledge or assumptions about the model parameters. Additionally, understanding how model parameters interact with the model randomness can be informative. To address these uncertainties, we conduct separate GSAs on individual replicates of the simulation, with each replicate generated using a unique random stream seed value.

The setup for the GSA experiment is as follows. Five of the MetaRVM model parameters are treated as uncertain within their specified ranges (listed in Table 1), while the remaining parameters are fixed at nominal values. The quantity of interest for the GSA is the total number of hospitalizations at the end of the simulation period, which is set to 90 days. We perform the GSA independently on 10 simulation replicates to assess the variability in parameter influences across model stochasticity. The MUSIC algorithm, implemented via the activeSens R package [11], is used to perform the GSA. It relies on a GP surrogate model constructed using the hetGP package [3]. For the acquisition strategy, we use the EIGF criterion, with the D1 formulation as the D-function, following the method described in Chauhan et al. [6].

## 3.2 Workflow implementation

The workflow is implemented using the EMEWS framework [7] and is driven by an R-based model exploration (ME) code, leveraging

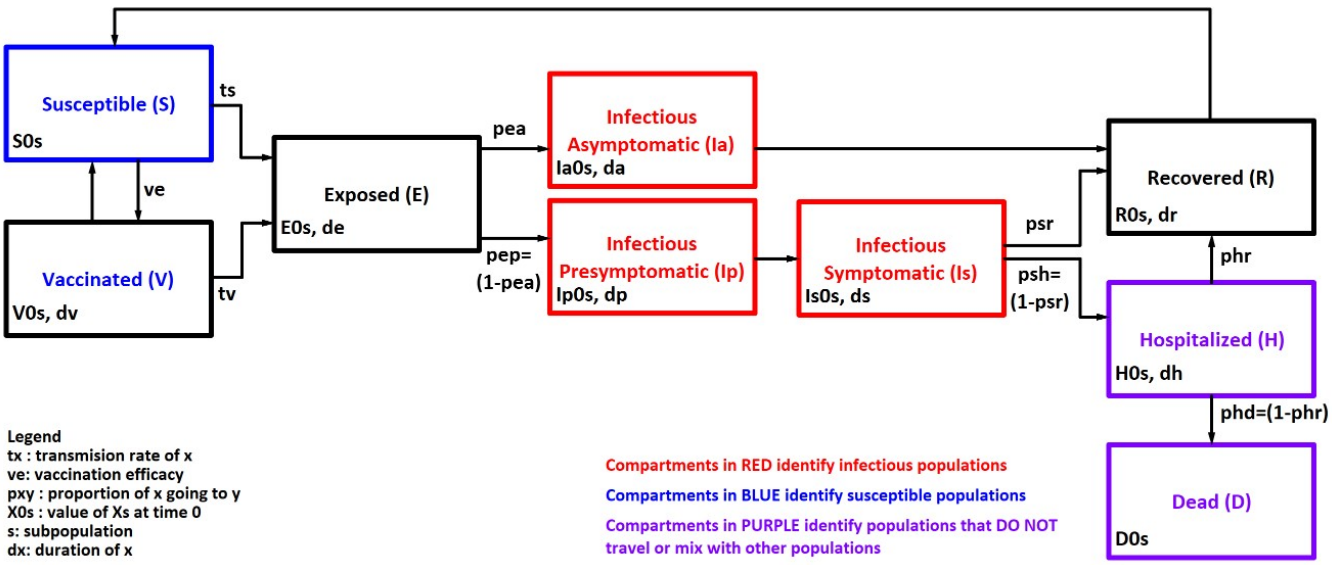


**Figure 3: The MetaRVM model [12] compartments, transitions between compartments, and associated parameters.**

**Table 1: MetaRVM model parameters and ranges for GSA**

| Parameter | Description | Range |
|---|---|---|
| *ts* | Transmission rate for susceptible | (0.1, 0.9) |
| *tv* | Transmission rate for vaccinated | (0.01, 0.5) |
| *pea* | Proportion of asymptomatic cases | (0.4, 0.9) |
| *psh* | Proportion of hospitalized | (0.1, 0.4) |
| *phd* | Proportion of dead | (0, 0.3) |

the multi-language capabilities of EMEWS and the framework's ability to flexibly utilize the 3rd party ME libraries described above. EMEWS is a HPC ME framework, developed for large-scale analyses (e.g., calibration, optimization) of computational models. EMEWS is based on a decoupled architecture consisting of a task database, and a task API, with both Python and R implementations, for distributing tasks on heterogeneous compute resources. EMEWS worker pools running on those compute resources retrieve and evaluate tasks submitted to the task database, e.g., the worker pools run models where the tasks' data are model input parameters.

The MUSIC workflow in this use case consists of 10 instances of the MUSIC algorithm, one for each of the 10 replicate experiments, although the workflow itself has separately been scaled to 100 replicate experiments as well. Each MUSIC algorithm begins by producing multiple parameter sets (i.e., an initial experiment design) containing the five MetaRVM inputs (Table 1) for evaluation from a latin hypercube sample (LHS). Subsequent iterations of each MUSIC algorithm produce a single parameter set for evaluation. These parameter sets are submitted to an EMEWS database using the EMEWS R task API. An EMEWS worker pool running on Improv, an HPC cluster managed by Argonne National Laboratory's LCRC, consumes these parameter sets and runs the MetaRVM model with the parameters as input. The results of the MetaRVM model, i.e., the number of hospitalizations, are submitted back to the EMEWS database where the MUSIC algorithm retrieves them for evaluation and analysis.

The difference between the initial number of evaluations and those in subsequent iterations in each of the MUSIC instances introduces difficulties in maintaining optimal use of computational resources. For example, if our MUSIC instances were run sequentially, the larger initial parameter evaluations may be able to fully utilize available cores, but the subsequent evaluations of individual parameters would not. This would results in poor compute utilization and longer runtimes to complete the full set of MUSIC instances. Our solution was to interleave the 10 MUSIC instances such that the compute resource is kept fully utilized. EMEWS's decoupled architecture that supports asynchronous evaluations allows us to easily do this (see [7] for further details). Submitting a task consists of inserting the task into a task database. Rather than wait for the task to complete, the submission returns a *Future*, which encapsulates the asynchronous execution of the task. This Future can then be queried later for the result of the task evaluation.

In the MUSIC algorithm the initial and subsequent steps of the algorithm have been implemented such that multiple instances of the algorithm can be interleaved using these asynchronous Futures. During each step, each algorithm performs a submission of tasks, and gets the Futures for those task evaluations back in return. Then, in turn, each algorithm checks for the completion of a single Future, ceding control to the next instance after this check. When all the Futures from an instance's submission have completed, that instance can continue to its next step. This continues from the initial sample set through the subsequent iterations such that submission and checking of results by each instance is interleaved with the other instances, resulting in better utilization of the computational resources than if a single instance was running to completion before another could start.

This interleaved workflow is wrapped by initialization and finalization code that demonstrates some additional EMEWS capabilities, namely the ability to programmatically start a worker pool on a compute node via an API call. The initialization code first sets up the EMEWS task queue used for the task submissions, and then starts an EMEWS worker pool. When this initialization code is run in production on a compute node (as opposed to locally when testing), the code starts a worker pool by submitting a job to the compute resource scheduler (e.g., SLURM or PBS). The job then starts the worker pool, running on a number of user-defined compute nodes. Once all of the MUSIC algorithms have finished, the finalization code closes the task queue, and stops the worker pool.

### 3.3 GSA Results

Figure 4 presents a comparative analysis of the MUSIC and PCE-based GSA algorithms for estimating first-order Sobol sensitivity indices across five MetaRVM parameters and fixing the random seed. The goal of this experiment is to evaluate the relative performance of the two approaches in approximating sensitivity indices and to assess how quickly each method converges to stable estimates as the sample size increases. The PCE-based method is included to highlight the limitations of one-shot approaches, as PCE uses a single experimental design to produce Sobol sensitivity indices, and to investigate how much improved sample efficiency can be expected from algorithms like MUSIC for GSA when considering limited computational budgets. We chose a degree 3 PCE as it performed the best among the PCE degrees we examined. In Figure 4 each facet corresponds to one parameter, with curves showing how the estimated indices evolve as additional samples are added one at a time. MUSIC demonstrates relatively quick (by 200 samples) stabilization compared to PCE, indicating better sampling efficiency across all the model parameters. While this sampling efficiency is less important with less computationally expensive compartmental epidemiological models, the potential for faster time-to-solution would greatly benefit more expensive agent-based epidemiological models [20]. To assess the impact of stochastic variability on sensitivity estimates, we ran the GSA independently on 10 replicates of the MetaRVM model. The results are summarized in Figure 5, where each facet corresponds to one of the five model parameters, and each line represents the evolution of the first-order Sobol index over increasing sample size for a given replicate.

## 4 Conclusion

Through two use cases we have demonstrated how OSPREY is supporting its original design goals of data ingestion, curation, and management, and the development of a SDE for rapid response and collaboration. The applications presented illustrate automation and time-sensitive workflows, metadata and provenance management, and flexible, fast turnaround prototyping on HPC resources. Future work includes continued co-design with our public health partners to further identify epidemiological analyses that can be directly integrated via OSPREY-enabled automation into their business processes, both between and during public health emergencies. The flexibility of OSPREY will also enable the development of novel, HPC-oriented model exploration algorithms to better utilize HPC and large cloud computing resources to produce timely responses to urgent questions. There is also a continued need to improve the ability to share scientific workflows, including making workflow artifacts such as models and model exploration algorithms more easily discoverable and shareable. This will help in creating a general readiness in enabling research groups and stakeholders to collaborate more effectively should the need arise.

## Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant 2200234, the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357 and the Bio-preparedness Research Virtual Environment (BRaVE) initiative. This research was completed with resources provided by the Argonne Computing, Environment, and Life Sciences Directorate General Computing Environment, the Laboratory Computing Resource Center at Argonne National Laboratory, and the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility.

## References

[1] 2025. Illinois Wastewater Surveillance System. https://iwss.uillinois.edu/.

[2] Rachana Ananthakrishnan, Josh Bryan, Kyle Chard, Ryan Chard, Kurt Mckee, Ada Nikolaidis, Jim Pruyne, Stephen Rosen, and Ian Foster. 2023. Globus Timers: Scheduling Periodic Data Management Actions on Distributed Research Infrastructure. In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good (PEARC '23)*. Association for Computing Machinery, New York, NY, USA, 283–287. doi:10.1145/3569951.3597571

[3] Mickaël Binois and Robert B. Gramacy. 2021. hetGP: Heteroskedastic Gaussian Process Modeling and Sequential Design in R. *Journal of Statistical Software* 98, 13 (2021), 1–44. doi:10.18637/jss.v098.i13

[4] Ryan Chard, Yadu Babuji, Zhuozhao Li, Tyler Skluzacek, Anna Woodard, Ben Blaiszik, Ian Foster, and Kyle Chard. 2020. funcX: A Federated Function Serving Fabric for Science. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*. ACM. doi:10.1145/3369583.3392683

[5] Ryan Chard, View Profile, Jim Pruyne, View Profile, Kurt McKee, View Profile, Josh Bryan, View Profile, Brigitte Raumann, View Profile, Rachana Ananthakrishnan, View Profile, Kyle Chard, View Profile, Ian T. Foster, and View Profile. 2023. Globus Automation Services. *Future Generation Computer Systems* 142, C (May 2023), 393–409. doi:10.1016/j.future.2023.01.010

[6] Mohit S Chauhan, Mariel Ojeda-Tuz, Ryan A Catarelli, Kurtis R Gurley, Dimitrios Tsapetis, and Michael D Shields. 2024. On active learning for Gaussian process-based global sensitivity analysis. *Reliability Engineering & System Safety* 245 (2024), 109945.

[7] Nicholson Collier, Justin M. Wozniak, Arindam Fadikar, Abby Stevens, and Jonathan Ozik. 2024. Distributed Model Exploration with EMEWS. In *2024 Winter Simulation Conference (WSC)*. 72–86. doi:10.1109/WSC63780.2024.10838848

[8] Nicholson Collier, Justin M. Wozniak, Abby Stevens, Yadu Babuji, Mickaël Binois, Arindam Fadikar, Alexandra Würth, Kyle Chard, and Jonathan Ozik. 2023. Developing Distributed High-performance Computing Capabilities of an Open Science Platform for Robust Epidemic Analysis. In *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, St. Petersburg, FL, USA, 868–877. doi:10.1109/IPDPSW59300.2023.00143

[9] Anne Cori, Neil M. Ferguson, Christophe Fraser, and Simon Cauchemez. 2013. A New Framework and Software to Estimate Time-Varying Reproduction Numbers During Epidemics. *American Journal of Epidemiology* 178, 9 (Nov. 2013), 1505–1512. doi:10.1093/aje/kwt133

[10] Pedro Nascimento de Lima, Abby Stevens, Raffaele Vardavas, Jonathan Ozik, and Robert J. Lempert. 2023. *Co-Designing Capabilities for a Robust Pandemic Response: Stakeholder Engagement for Visioning, Backcasting, and Evaluating New Decision-Support Capabilities*. RAND Corporation, Santa Monica, CA. doi:10.7249/WRA3085-1

[11] Arindam Fadikar. 2024. *activeSens R package*. https://github.com/NSF-RESUME/activeSens

[12] Arindam Fadikar, Abby Stevens, Sara Rimer, Ignacio Martinez-Moyano, Nicholson Collier, Jonathan Ozik, and Charles Macal. 2025. Developing and Deploying a Use-Inspired Metapopulation Modeling Framework for Detailed Tracking of Stratified Health Outcomes. 2025.05.05.25327021 pages. doi:10.1101/2025.05.05.25327021

[13] Isaac H. Goldstein, Daniel M. Parker, Sunny Jiang, and Volodymyr M. Minin. 2024. Semiparametric inference of effective reproduction number dynamics from

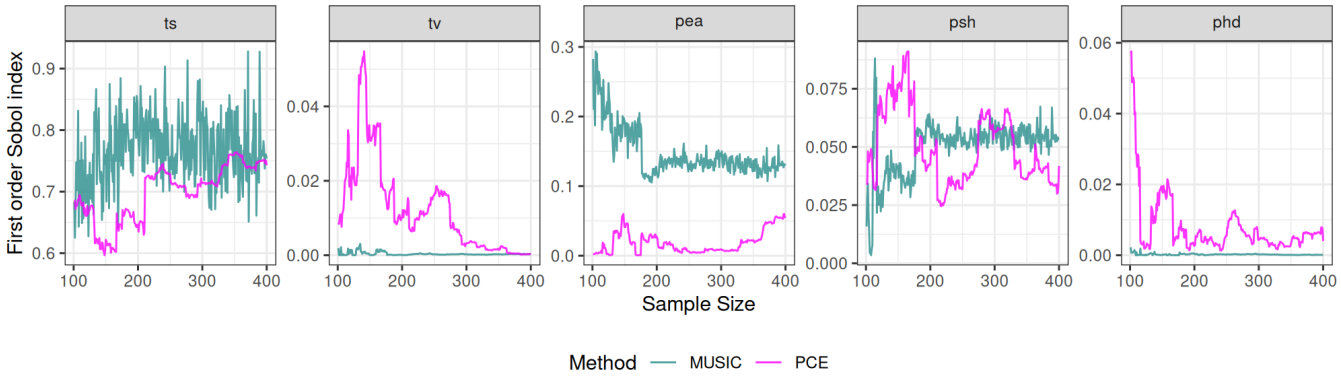


**Figure 4: Comparison of first-order Sobol sensitivity index estimates for five MetaRVM parameters using the MUSIC (teal) and PCE-based (magenta) algorithms as a function of sample size. Each panel corresponds to one model parameter.**

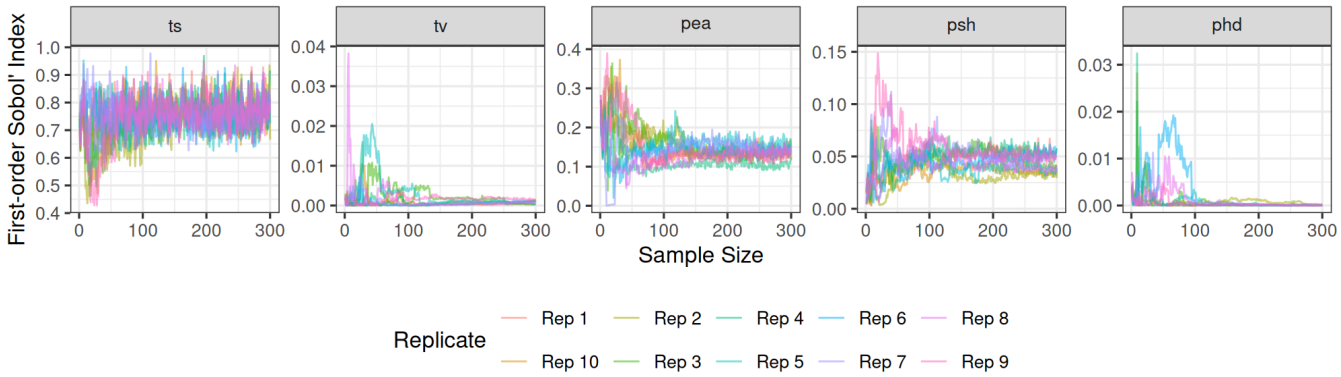


**Figure 5: First-order Sobol sensitivity indices estimated independently across 10 stochastic replicates of MetaRVM. Each panel corresponds to one model parameter.**


wastewater pathogen surveillance data. *Biometrics* 80, 3 (2024). doi:10.1093/biomtc/ujae074

[14] Katelyn M. Gostic, Lauren McGough, Edward B. Baskerville, Sam Abbott, Keya Joshi, Christine Tedijanto, Rebecca Kahn, Rene Niehus, James A. Hay, Pablo M. De Salazar, Joel Hellewell, Sophie Meakin, James D. Munday, Nikos I. Bosse, Katharine Sherrat, Robin N. Thompson, Laura F. White, Jana S. Huisman, Jérémie Scire, Sebastian Bonhoeffer, Tanja Stadler, Jacco Wallinga, Sebastian Funk, Marc Lipsitch, and Sarah Cobey. 2020. Practical Considerations for Measuring the Effective Reproductive Number, Rt. *PLOS Computational Biology* 16, 12 (Dec. 2020), e1008409. doi:10.1371/journal.pcbi.1008409

[15] Robert B Gramacy. 2020. *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences.* Chapman and Hall/CRC.

[16] Valérie Hayot-Sasson, Abby Stevens, Nicholson Collier, Sudershan Sridhar, Kyle Conroy, J. Gregory Pauloski, Yadu Babuji, Maxime Gonthier, Nathaniel Hudson, Dante D. Sanchez-Gallegos, Ian Foster, Jonathan Ozik, and Kyle Chard. 2025. AERO: An Autonomous Platform for Continuous Research. arXiv:2505.18408 [cs] doi:10.48550/arXiv.2505.18408

[17] Anna L. Hotton, Jonathan Ozik, Chaitanya Kaligotla, Nick Collier, Abby Stevens, Aditya S. Khanna, Margaret M. MacDonell, Cheng Wang, David J. LePoire, Young-Soo Chang, Ignacio J. Martinez-Moyano, Bogdan Mucenic, Harold A. Pollack, John A. Schneider, and Charles Macal. 2022. Impact of Changes in Protective Behaviors and Out-of-Household Activities by Age on COVID-19 Transmission and Hospitalization in Chicago, Illinois. *Annals of Epidemiology* (June 2022), S1047279722001053. doi:10.1016/j.annepidem.2022.06.005

[18] Art B. Owen. 2014. Sobol' Indices and Shapley Value. *SIAM/ASA Journal on Uncertainty Quantification* 2, 1 (2014), 245–251. arXiv:https://doi.org/10.1137/130936233 doi:10.1137/130936233

[19] Jonathan Ozik, Nicholson T. Collier, Justin M. Wozniak, and Carmine Spagnuolo. 2016. From Desktop to Large-Scale Model Exploration with Swift/T. In *2016 Winter Simulation Conference (WSC)*. 206–220. doi:10.1109/WSC.2016.7822090

[20] Jonathan Ozik, Justin M Wozniak, Nicholson Collier, Charles M Macal, and Mickaël Binois. 2021. A Population Data-Driven Workflow for COVID-19 Modeling and Learning. *The International Journal of High Performance Computing Applications* 35, 5 (Sept. 2021), 483–499. doi:10.1177/10943420211035164

[21] Abby Stevens, Jonathan Ozik, Kyle Chard, Jaline Gerardin, and Justin M. Wozniak. 2023. NSF RESUME HPC Workshop: High-Performance Computing and Large-Scale Data Management in Service of Epidemiological Modeling. arXiv:2308.04602 [cs]

[22] Abby Stevens, Jonathan Ozik, Junhong Chen, Rachel Poretsky, and Arvind Ramanathan. 2023. *NSF RESUME EcoEpi Workshop: One Health Surveillance and Predictive Intelligence for Eco-Epidemiological Modeling.* Preprint. Open Science Framework. doi:10.31219/osf.io/tazm2

[23] Steven Tuecke, Rachana Ananthakrishnan, Kyle Chard, Mattias Lidman, Brendan McCollam, Stephen Rosen, and Ian Foster. 2016. Globus Auth: A Research Identity and Access Management Platform. In *2016 IEEE 12th International Conference on E-Science (e-Science)*. 203–212. doi:10.1109/eScience.2016.7870901