Building A Web-based Simulation Application to Identify Optimal Study Design and Analysis Combinations For Health Impacts of Policies and Programs

Epidemiology in the Cloud and on Savio

Jennifer Ahern, MPH, PhD
Scott Zimmerman, MPH
Ellie Colson, MPH

Division of Epidemiology

School of Public Health
UNIVERSITY OF CALIFORNIA, BERKELEY
Outline

• Project Background, Overview and Current State

• Open questions
Background

Motivation

- Large-scale policies and programs have substantial influences on health, e.g. violent crime reduction initiatives and economic development projects.

- Remarkably, for most policies and programs, the health effects of are never assessed, particularly those not designed to influence health.

- In the rare instance that the post-implementation health effects of a program or policy are examined, current approaches to studying effects have major limitations.

- The choice of study design and analysis approach has been informed by general frameworks, but more rigorous quantitative assessment of which approaches are best to answer the scientific question of interest would greatly improve study quality.

- There is a critical need for a rigorous system that can be applied to improve assessment of the post-implementation effects of policies and programs on a wide range of health outcomes.
Grant Aims

- **Establish a new system** to determine the health and health disparities impacts of policies and programs.
  - Create a web-based simulation generator to identify the optimal study design and analysis combination for a specific program or policy.
  - Compile a database of over 10 years of population health data that can be utilized to estimate health effects of policies and programs in California.

- **Apply this system** to assess the health and disparities effects of a compelling set of current policies and programs related to criminal justice in California.
  - Specifically we will assess the impacts of a violence program implemented by Oakland, and the criminal justice realignment (Assembly Bill 109) that was mandated statewide due to severe prison overcrowding.
Timeline

• Work on system started April 2014
• Currently:
  – Concluding setup of the system architecture with current simulation scripts
  – [www.studysimulator.com](http://www.studysimulator.com) – sign up for e-mail notices
• Near future:
  – Testing and refinement of simulation algorithms
  – UI/UX
## Architecture Overview

<table>
<thead>
<tr>
<th>Component</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client</strong></td>
<td>AngularJS by Google</td>
</tr>
<tr>
<td><strong>Server</strong></td>
<td>Node.js express</td>
</tr>
<tr>
<td><strong>Cluster</strong></td>
<td>Microsoft Azure, Sav, Amazon Web Services</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>io, MySQL</td>
</tr>
</tbody>
</table>
Web-based simulation generator to identify the optimal study design and analysis combination for a specific program or policy.
Simulation Stage 1: Determine Distribution

Goal: Calculate a distribution given information about the population from the user

The distribution will be used to generate a population for our simulated studies and analyses.
Simulation Stage 1: Determine Distribution

Simple Parallelization:
- Independent units trying to calculate good distributions given the desired characteristics of the population.
- Units may use different algorithms.
- No sharing of information between units (for now).
Simulation Stage 2: Simulate Studies

Goal: Determine the performance of a set of study designs and analyses on the simulated population output from step 1

Returned to the user:
- Design and analysis performance metrics
  - bias, variance, mean squared error, confidence interval coverage
- Code to run the best metric(s) locally
- The simulated population
Simulation Stage 2: Simulate Studies

Parallelization:

- Loop 1: Sample from population
  - Loop 2: Bootstrap sampled data
  - Loop 3: Apply matching and analysis methods

Currently parallelized across outermost loop
Simulation Stage 2: Simulate Studies

Parallelization:

- Loop 1: Sample from population 1,000 iterations

- Loop 2: Bootstrap sampled data 500 times; will collapse across the bootstraps to estimate CI coverage

- Loop 3: Apply matching and analysis methods 7 matching, 12 analysis = 84 combinations

Estimating quantity of interest 96,000,000 times
Currently parallelized across outermost loop
Open Questions

• Cluster Computing
• Computation Time
• User Management/Limitations
• Database
• Testing
• Future Directions
Cluster Computing:

We have multiple cluster computing options

• Goal: Be able to transfer the computation part of the system from one platform to the others (or to future options) with minimal reconfiguration.

• Working with Aaron Cluich to build a Savio-compatible image that we can use on Azure
Cluster Computing:
A Note on Savio

• Can’t currently submit jobs to Savio from a web server
  – Possible future direction
• Currently only using Savio for testing
Cluster Computing:
Architecture
Cluster Computing:
Architecture e.g.
Cluster Computing:
Open Questions

• Can we use one image and procedure across the various providers?
• How to scale up and down to meet demand?
• How to store current state mid-simulation in a valid way (i.e. replicable)?
Computation Time:
Main causes of high computational time

Study Simulation
• Number of iterations, number of bootstraps, number of matching methods, number of analysis methods
• SuperLearner, ensemble machine learning function
• Genetic matching
Computation Time:
Locations where SuperLearner and Genetic matching enter

- Without bootstrapping:
  - Loop 1: Sample from population data—1,000 iterations
    - Loop 2: Apply matching method—7 versions
      - SuperLearner x 1
      - Genetic matching x 1
    - Loop 3: Apply analysis method—12 versions
      - SuperLearner x 4 (if subclassification, this is SL x 40)

- Implies:
  - Running SuperLearner 64,000 times
  - Running Genetic matching 7,000 times

- If we include bootstrapping, these values are:
  - SuperLearner: 32,000,000 times
  - Genetic matching: 3,500,000 times
  - Parallelized across 32 cores on GrizzlyBear; took 1 month to run all iterations, **failed when compiling results of iterations**
Computation Time:
Methods for speeding up computation

- Fewer iterations and/or bootstraps; using convergence criteria
  - Risk introducing bugs related to convergence criteria
  - Risk making conclusions based on results that have not yet converged
- SuperLearner: fewer algorithms, alternative forms of SuperLearner, don’t repeat estimation of treatment or outcome models when used in multiple analysis methods
  - May result in poorer performance
- Genetic matching: smaller “population size”, other alternative implementations
  - May result in poorer performance
- Batch and/or parallelize in different ways?
User Management:

• Key user activities
  – Run Simulations
  – View results
  – Account management

• Currently managing user activities ourselves
  – Storing user activity data in DB
  – Any frameworks/Common solutions?
User Management:
Limiting User Computation

• Current Approach:
  – Record user name, email, IP address and then limit
  – Registration process: Force users to input some real information about themselves

• How to stop people from registering a bunch of e-mail addresses?

• Possible future direction: Allow people to use their own computational resources for simulations if they go beyond our allocation (We could provide instructions, scripts, images, etc.)

• How to decide on cutoffs?
Database:
Tables for a variety of purposes

• Simulation
  — DAGs
  — User Inputs
  — Intermediate data (designs, analyses, matching)
  — Temporary Tables
    • Used in Simulation Step 1
    • Potentially used for storing populations (if this reduction in allocated memory can increase computation speed)

• User Data
  — Email addresses, session data, visits, etc.
Database:
Open Questions

• Backup?
• Cost
• Security:
  – Currently using stored procedures/functions for all queries
  – DB credentials on server separately from the rest of the code
• Best location?
Future Directions

• How much does the user have to tell us so that we can give consistent recommendations from the design and analysis stage?

• Can we represent uncertainty in user inputs?

• Can we tell users how sensitive our recommendations are to their inputs?

• Extension of our applications beyond criminal justice umbrella
Acknowledgements

Thank you to everyone who has contributed to this project!

Kara Rudolph
Dana Goin
Jess Galin
Beth Anderson
Mark Van Der Laan
Will Dow
Laura Balzer
Erin Ledell
Sam Lendle
Jas Sekhon
Aaron Culich
Alex Luedtke
Josh Schwab

Funding for this research was provided in part by NICHD/NIH Office of the Director DP2HD080350, the Robert Wood Johnson Health and Society Scholars Program, and the University of California, Berkeley Committee on Research.
Cluster Computing: Potential Alternative Architecture

No communication between worker nodes and database, all through master node
User Management:
Maximizing Compute Resources

• Avoiding redundant computation:
  – Use DB to return results if already calculated for a different user
    • Make sure we always get the same results from the same inputs (i.e. convergence is achieved)
  – Some computation we could run cheaply and have results available?
    • Informed by what users use most, some of the more common DAGs

• Choice of pricing options:
  – Use spot pricing where possible
    • Implications for design: Must store intermediate results e.g. between iterations of loop 1