# README for "Empirical Bayes When Estimation Precision Predicts Parameters"

Jiafeng Chen, Stanford University

jiafeng@stanford.edu

MS-22935 for *Econometrica*

June 2, 2025

## Overview

The code in this replication package constructs simulated Monte Carlos from the Opportunity Atlas (Chetty et al., 2022) using Python. The Monte Carlo is time-consuming but extremely parallelizable (a few minutes per draw on a 2022 Apple M1 Max Mac Studio, $1000 \times 15$ draws). All code is in Python, but NPMLE estimation relies on the package rpy2. The generated Monte Carlo data is included. Without re-generating the Monte Carlo data, the rest of the code should run in a short time. Three main analysis files generate all 9 figures (5 in the main text, 4 in the online appendix) and 1 table (online appendix).

## Data Availability and Provenance Statements

☐ This paper does not involve analysis of external data (i.e., no data are used or the only data are generated by the authors via simulation in their code).

The data are taken from the published datasets by Chetty et al. (forthcoming) in Chetty et al. (2022) under CC-BY-4.0. They are available at `https://opportunityinsights.org/data/?geographic_level=0&topic=0&paper_id=1652#resource-listing` (Accessed 2024-09-16).

### Statement about Rights

✓ I certify that the author(s) of the manuscript have legitimate access to and permission to use the data used in this manuscript.
✓ I certify that the author(s) of the manuscript have documented permission to redistribute/publish the data contained within this replication package. Appropriate permission are documented in the LICENSE.txt file.

### License for Data

The data are licensed under a CC-BY-4.0 license. See LICENSE.txt for details.

### Summary of Availability

✓ All data **are** publicly available.
☐ Some data **cannot be made** publicly available.
☐ **No data can be made** publicly available.

☐ Confidential data used in this paper and not provided as part of the public replication package will be preserved for _____ years after publication, in accordance with journal policies.

### Details on each dataset and data source

This package builds from the following raw data files, all from Chetty et al. (2022). They are downloaded at `https://opportunityinsights.org/data/?geographic_level=0&topic=0&paper_id=1652#resource-listing` (Accessed 2024-09-16).

1. `data/raw/tract_covariates.dta` ("Neighborhood Characteristics by Census Tract"): `https://opportunityinsights.org/wp-content/uploads/2018/10/tract_outcomes_dta.zip` (Accessed 2024-09-16)

2. `data/raw/tract_outcomes_early.csv` ("All Outcomes by Census Tract, Race, Gender and Parental Income Percentile"): `https://opportunityinsights.org/wp-content/uploads/2018/10/tract_outcomes.zip` (Accessed 2024-09-16)

## Computational requirements

INSTRUCTIONS: In general, the specific computer code used to generate the results in the article will be within the repository that also contains this README. However, other computational requirements - shared libraries or code packages, required software, specific computing hardware - may be important, and is always useful, for the goal of replication. Some example text follows.

INSTRUCTIONS: We strongly suggest providing setup scripts that install/set up the environment. Sample scripts for Stata, R, Julia are easy to set up and implement. Specific software may have more sophisticated tools: Python, Julia.

### Software Requirements

- Python (last run on 3.9.13)

  See `environment.yml` for list of packages.

  - RPy2 3.5.16

  RPy2 status:

```
> python -m rpy2.situation
rpy2 version:
3.5.16
Python version:
3.9.13 | packaged by conda-forge | (main, May 27 2022, 17:01:00)
[Clang 13.0.1 ]
Looking for R's HOME:
    Environment variable R_HOME: None
    Calling `R RHOME`: /Library/Frameworks/R.framework/Resources
    Environment variable R_LIBS_USER: None
R's value for LD_LIBRARY_PATH:

R version:
    In the PATH: R version 4.2.1 (2022-06-23) -- "Funny-Looking Kid"
    Loading R library from rpy2: OK
Additional directories to load R packages from:
None
C extension compilation:
  include:
  ['/Library/Frameworks/R.framework/Resources/include']
  libraries:
  ['pcre2-8', 'lzma', 'bz2', 'z', 'icucore', 'dl', 'm', 'iconv']
  library_dirs:
  ['/opt/R/arm64/lib', '/opt/R/arm64/lib']
  extra_compile_args:
  ['-std=c99']
  extra_link_args:
  ['-F/Library/Frameworks/R.framework/..', '-framework', 'R']
Directory for the R shared library:
lib
CFFI extension type
  Environment variable: RPY2_CFFI_MODE
  Value: CFFI_MODE.ANY
  ABI: PRESENT
  API: PRESENT
```

- MOSEK 10.2.5 (Mosek and RMosek installation: `https://docs.mosek.com/latest/rmosek/install-interface.html`)
- R version 4.2.1
  - REBayes (2.51)
  - nprobust (0.4.0)

```
> sessionInfo()
R version 4.2.1 (2022-06-23)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS 15.1.1

Matrix products: default
LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] REBayes_2.51   Matrix_1.6-4   nprobust_0.4.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.11    rstudioapi_0.14 magrittr_2.0.3  splines_4.2.1   tidyselect_1.2.0 munsell_0.5.0   colorspace_2.1-0 lattice_0.20-45  R6_2.5.1
[10] rlang_1.1.2.9000 fansi_1.0.5    dplyr_1.1.3    tools_4.2.1    grid_4.2.1     gtable_0.3.4    utf8_1.2.4     cli_3.6.1      tibble_3.2.1
[19] lifecycle_1.0.4 ggplot2_3.4.4   vctrs_0.6.4    glue_1.6.2     compiler_4.2.1  pillar_1.9.0    generics_0.1.3  scales_1.2.1    Rmosek_10.2.0
[28] pkgconfig_2.0.3
```

Portions of the code use bash scripting, which may require Linux.

### Controlled Randomness

- ✓ Random seed is set at:
  - Line 181 of `covariate_additive_model.py`
  - Line 216 of `empirical_exercise.py`
- ☐ No Pseudo random generator is used in the analysis described here.

### Memory, Runtime, Storage Requirements

**Summary** Approximate time needed to reproduce the analyses on a standard (2025) desktop machine:

- ✓ <10 minutes (Reproducing from scored Monte Carlo outputs)
- ✓ 10-60 minutes (Reproducing from Monte Carlo outputs)
- ☐ 1-2 hours
- ☐ 2-8 hours
- ☐ 8-24 hours
- ✓ 1-3 days (Full reproduction)
- ☐ 3-14 days
- ☐ > 14 days

Approximate storage space needed:

- ☐ < 25 MBytes
- ☐ 25 MB - 250 MB
- ☐ 250 MB - 2 GB
- ☐ 2 GB - 25 GB
- ✓ 25 GB - 250 GB
- ☐ > 250 GB
- ☐ Not feasible to run on a desktop machine, as described below.

**Details** The code was last run on a **2022 Mac Studio, 32GB RAM, Apple M1 Max**.

## Description of programs/code

The following are files needed for replicating tables and figures:

1. `build_data.py` performs basic cleaning on the raw Opportunity Atlas data and saves the processed data in `data/processed/oa_data_used.feather`. The rest of the analysis proceeds only with `data/processed/oa_data_used.feather`.
2. Code for generating the Monte Carlo data:
   (a) `monte_carlo.sh` runs the calibrated simulation exercise
   (b) `coupled_bootstrap.sh` runs the coupled bootstrap exercise
   (c) `weibull_model.sh` runs the Weibull distribution exercise in online appendix
   (d) `additive_model.sh` runs an additional model exercise in online appendix

   These save data in `data/simulated_posterior_means/`
3. `generate_scores.py` takes Monte Carlo results in `data/simulated_posterior_means/` and computes various metrics. It saves results in `results/[SimulatorName]/*.csv`.
4. Code for generating tables and figures:
   (a) `assets_introduction.py` generates Figures 1–3 and data in footnote 6
   (b) `assets_empirical.py` generates Figures 4–5
   (c) `assets_appendix.py` generates Table OA5.1, Figures OA5.1–5.4 in the online appendix.

These depend on the following lower-level files:

1. `empirical_exercise.py` samples either calibrated Monte Carlo simluation or coupled bootstrap simulation. It then computes various posterior mean estimates using various empirical Bayes or non-empirical Bayes methods.
2. `covariate_additive_model.py` runs CLOSE-NPMLE with a flexible additive model for covariates.
3. Helpers:
   (a) `residualize.py` implements linear residualization by covariates
   (b) `conditional_means/` contains methods for estimating conditional means
   (c) `empirical_bayes/` contains methods for implementing empirical Bayes methods
   (d) `postprocessing/` contains methods for computing various metrics and visualization
   (e) `simulator/` contains code for implementing various methods for simulation synthetic data from raw data

## Instructions to Replicators

### Replicate the Monte Carlo data

The following bash commands generates Monte Carlo data. This is time consuming to produce, and so the data is transmitted directly.

```
# Builds the raw analysis dataset from raw data
python build_data.py
```

```
# Run the Monte Carlo
# The following generates results/[simulator-name]
# where [simulator-name] is one of "coupled_bootstrap-0.9",
# "covariate_additive_model", "npmle_by_bins", "weibull"

# Calibrated simulation exercise
sh monte_carlo.sh # see note at the end

# Validation exercise using coupled bootstrap
sh coupled_bootstrap.sh

# Weibull exercise in OA5.3
```

```
sh weibull_model.sh

# Additive model exercise in OA5.4
# Saves results directly in results/covariate_additive_model/*.csv
sh additive_model.sh
```

These use the library GNU Parallel

O. Tange (2018): GNU Parallel 2018, March 2018, https://doi.org/10.5281/zenodo.1146014.

### Note on replicating Monte Carlo data

For some upstream reason having to do with MOSEK or REBayes, running `monte_carlo.sh` for many iterations might silently fail, due to memory leak. When it has failed, the code would appear to run but resource consumption is low and no new output is generated. Interrupting the code prints `Segmentation fault`. I find it quite difficult to reproduce the issue, as there's no fixed data seed causing a problem. When this happens, interrupting and restarting resolves the issue. This has only happened when the I repeatedly apply NPMLE to sample new data and to estimate various methods.

Each Monte Carlo draw results in a file of the form

`data/simulated_posterior_means/[SimulatorName]/[VariableName]/[Seed].feather`.

It is not time-consuming to verify a small subset of the results. For instance, we may run the following to generate results from a particular seed ("94999" in this case) in `data/simulated_posterior_means_sample`.

```
python empirical_exercise.py --nsim 1 --starting-seed 94999 --data-dir "data/simulated_posterior_means_sample"
    --simulator-name npmle_by_bins --methods all --est_var kfr_black_pooled_p25

python empirical_exercise.py --nsim 1 --starting-seed 94999 --data-dir "data/simulated_posterior_means_sample"
    --simulator-name coupled_bootstrap-0.9 --methods all --est_var kfr_black_pooled_p25

python empirical_exercise.py --nsim 1 --starting-seed 94399 --data-dir
"data/simulated_posterior_means_sample" --simulator-name weibull --methods
    indep_gauss,close_npmle,close_gauss,close_gauss_parametric --est_var kfr_black_pooled_p25
# The Weibull exercise only has 100 seeds from 94301 to 94400
```

We can then run the following Python code to verify that the data generated agrees with the data provided

```
import pandas as pd

# Reproduces as of 2025-06-01
orig = pd.read_feather("data/simulated_posterior_means/coupled_bootstrap-0.9/kfr_black_pooled_p25/94999.feather")
new =
    pd.read_feather("data/simulated_posterior_means_sample/coupled_bootstrap-0.9/kfr_black_pooled_p25/94999.feather")
print((orig.drop("czname", axis=1) - new.drop("czname", axis=1)).values.std())

orig = pd.read_feather("data/simulated_posterior_means/npmle_by_bins/kfr_black_pooled_p25/94999.feather")
new = pd.read_feather("data/simulated_posterior_means_sample/npmle_by_bins/kfr_black_pooled_p25/94999.feather")
print((orig - new).values.std())

orig = pd.read_feather("data/simulated_posterior_means/weibull/kfr_black_pooled_p25/94399.feather")
new = pd.read_feather("data/simulated_posterior_means_sample/weibull/kfr_black_pooled_p25/94399.feather")
print((orig - new).values.std())

## Output
# 0.0
# 0.0
# 0.0
```

### Replication of results given the Monte Carlo data

Score the Monte Carlo results and generate csv files in `results/`

```
### Given Monte Carlo results in results/[simulator-name]/

# Clean up the generated raw Monte Carlo results
python generate_scores.py --simulator-name coupled_bootstrap-0.9 --nsim 1000
python generate_scores.py --simulator-name npmle_by_bins --nsim 1000
python generate_scores.py --simulator-name weibull --nsim 100
# The additive model results are processed directly
```

Given the csv files in `results/`, creating tables and figures is ready momentarily.

```
# Generate figures and tables in assets/
# Fig 1, 2, 3 and footnote 6
python assets_introduction.py

# Fig 4, 5
python assets_empirical.py

# Table OA5.1, Fig OA5.1, OA5.2, OA5.3, OA5.4
python assets_appendix.py
```

# References


**Chetty, Raj, John Friedman, Nathaniel Hendren, Maggie R. Jones, and Sonya R. Porter**, "Replication Data for: The Opportunity Atlas: Mapping the Childhood Roots of Social Mobility," 2022.

**_ , John N Friedman, Nathaniel Hendren, Maggie R Jones, and Sonya R Porter**, "The opportunity atlas: Mapping the childhood roots of social mobility," *American Economic Review*, forthcoming.