Reproducibility of A Conditional Ordinal Stereotype Model to Estimate Police Officers' Propensity to Escalate Force

true

August 02, 2025

This form documents the artifacts associated with the article (i.e., the data and code supporting the computational findings) and describes how to reproduce the findings.

Part 1: Data

- ☐ 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).
- ☒ I certify that the author(s) of the manuscript have legitimate access to and permission to use the data used in this manuscript.

Abstract

The data provided here are the Seattle Police Department (SPD) use-of-force data, which illustrate the conditional ordinal stereotype model in practice. `dataSPD.RData` contains the Seattle PD use-of-force data already formatted for analysis, a single data frame `d` with three columns, an incident ID, a scrambled officer ID, and the force type (an ordinal value from 1 to 4).

Availability

- ☒ Data **are** publicly available.
- ☐ Data **cannot be made** publicly available.

If the data are publicly available, see the *Publicly available data* section. Otherwise, see the *Non-publicly available data* section, below.

Publicly available data

- ☒ Data are available online at: https://github.com/gregridgeway/COS-police-use-of-force/tree/main/data
- ☐ Data are available as part of the paper's supplementary material.
- ☐ Data are publicly available by request, following the process described here:
- ☐ Data are or will be made available through some other mechanism, described here:

Description

`dataSPD.RData` contains the Seattle PD use-of-force data already formatted for analysis, a single data frame `d`. The data frame contains the following columns:

`id`: use-of-force incident identifier, sequential integers from 1 to 4,821 `idOff`: officer identifier, integers from 1 to 1,503. `idOff` is a scrambled identifier and does not link to any Seattle PD officer information

`y`: ordinal use-of-force level, the most serious level of force that the officer used at this incident. `y` contains integers from 1 to 4, where 1 is the lowest severity and 4 is the highest severity. These map onto the SPD force types with 0 representing no force or de minimus force up to 3 representing lethal or potentially lethal force.

Incidents with no variation in `y` (e.g. single officer incidents, incidents where no officer uses force) have been removed from the data since they have no information for the parameters of interest through the conditional likelihood.

File format(s)

- ☐ CSV or other plain text.
- ☒ Software-specific binary format (.Rda, Python pickle, etc.): **RData**
- ☐ Standardized binary format (e.g., netCDF, HDF5, etc.):
- ☐ Other (please specify):

Data dictionary

- ☐ Provided by authors in the following file(s):
- ☐ Data file(s) is(are) self-describing (e.g., netCDF files)
- ☒ Available at the following URL: https://github.com/gregridgeway/COS-police-use-of-force/tree/main/data

Part 2: Code

Abstract

The code provided here implements the conditional ordinal stereotype model and reproduces the main results of the article, including three simulations and the analysis of Seattle Police Department use-of-force data. The provided code is mostly written in R with the most computationally intensive conditional likelihood calculation written and parallelized in C++.

Description

Code format(s)

- ☒ Script files
  - ☒ R
  - ☐ Python
  - ☐ Matlab
  - ☐ Other:
- ☐ Package
  - ☐ R
  - ☐ Python
  - ☐ MATLAB toolbox
  - ☐ Other:
- ☐ Reproducible report
  - ☐ R Markdown
  - ☐ Jupyter notebook
  - ☐ Other:
- ☐ Shell script
- ☒ Other (please specify): C++ source code

Supporting software requirements

Version of primary software used R version 4.5.1

Libraries and dependencies used by the code R packages

- For implementation of the conditional ordinal stereotype model
  - Rcpp 1.0.14
  - RcppParallel 5.1.10
- For testing analytic calculations
  - arrangements 1.1.9
  - PoissonMultinomial 1.1
- For tables and figures
  - ggplot2 3.5.2
  - xtable 1.8-4
  - viridis 0.6.5
  - RColorBrewer 1.1-3
  - ggbeeswarm 0.7.2
  - igraph 2.1.4
- For processing MCMC draws
  - doParallel 1.0.17
  - parallel 4.5.1
  - foreach 1.5.2
  - future 1.58.0
  - progressr 0.15.1
  - doFuture 1.1.1
- For general data manipulation
  - dplyr 1.1.4
  - tidyr 1.3.1
  - purrr 1.0.4

Supporting system/hardware requirements (optional)

No specific system or hardware requirements are needed to run the code. Multi-core processing is used to speed up the calculations, but the code can run on a single core.

Parallelization used

☐ No parallel code used
☒ Multi-core parallelization on a single machine/node
  - Number of cores used: 8-40

☐ Multi-machine/multi-node parallelization
  - Number of nodes and cores used:

Machines used for the article

- 8 core Intel i7-6820HQ CPU @ 2.70GHz 16Gb RAM (laptop)
- 16 core 12th Gen Intel Core i7-1260P @ 2.10 GHz 16Gb RAM (laptop)
- 40 core CentOS Linux 7 Intel Xeon @ 2.60GHz (general processing cluster)

License

☒ MIT License (default)
☐ BSD
☐ GPL v3.0
☐ Creative Commons
☐ Other: (please specify)

Part 3: Reproducibility workflow

Scope

The provided workflow reproduces:

☒ Any numbers provided in text in the paper
☒ The computational method(s) presented in the paper (i.e., code is provided that implements the method(s))
☒ All tables and figures in the paper
☐ Selected tables and figures in the paper, as explained and justified below:

Workflow

Location

The workflow is available:

☐ As part of the paper's supplementary material.
☒ In this Git repository: https://github.com/gregridgeway/COS-police-use-of-force
☐ Other (please specify):

Format(s)

☐ Single master code file
☐ Wrapper (shell) script(s)
☐ Self-contained R Markdown file, Jupyter notebook, or other literate programming approach
☒ Text file (e.g., a readme-style file) that documents workflow
☐ Makefile
☐ Other (more detail in *Instructions* below)

Instructions

The main README file describes all the scripts used to produce the results in the article. All of the scripts are self-contained. That is, inside each script there is code that will `source()` the needed conditional ordinal stereotype code, which in turn compiles the C++ code to make it available for use. Every table and figure in the article has an associated line in one of the R scripts that produces it. Comments in the R script will indicate which Section, Table, or Figure, or quoted sentence connects with that line of code.

The computationally intensive scripts save their output to the `output/` folder, which is included in the Git repository. Those prefixed with "mcmc" contain the MCMC draws from the conditional ordinal stereotype model. Those prefixed with "results" contain results from the processed MCMC draws. The computationally intensive scripts, particularly `mcmcSPD.R` and `mcmcSPDpermute.R`, could be avoided if only the MCMC draws or results are of interest.

Expected run-time

Approximate time needed to reproduce the analyses on a standard desktop machine:

☐ < 1 minute
☐ 1-10 minutes
☐ 10-60 minutes
☐ 1-8 hours
☒ > 8 hours
☐ Not feasible to run on a desktop machine, as described here:

Approximate timing results on a basic 8 core desktop machine (more cores = shorter run time)

- Main article results

  - Simulated example 1 (Section 4.1): 2 minutes
  - Simulated example 2 (Section 4.2): 4 minutes
  - Simulated example 3 (Section 4.3): 4 minutes
  - Main analysis of SPD data (Section 5): 3 days
- Supplementary material results
  - Appendix E1: 40 minutes
  - Appendix E2: 11 hours
  - Appendix E3: 4 hours
  - Appendix F, Five placebo tests on SPD data: 15 days