Computational Game Theory to Study Empirical Elections

Fabricio Vasselai
Political Science & Scientific Computing (vasselai@umich.edu)

Motivation

Simulations of electoral results are regularly used to study actual elections in many ways. For example: to predict electoral outcomes, to help identify frauds, to serve as synthetic data for Supervised Machine Learning.

How simulations of electoral results are usually done:

- extrapolation from the past (e.g. using regression);
- sampling from assumed distributions.

Problems:

- volatility - past not always a good predictor in democracies;
- reliance on assumption of a parametric DGP for elections;
- inability to model strategic behavior.

Solution:

- simulate individual voters, with their possibly strategic behavior, to more directly **approximate** the DGP of an election.

Method

The technique we propose resorts to a sub-field of Artificial Intelligence known as Multi-Agent Systems, as a framework to translate canonical Game Theory models of voting into computer simulations. Specifically:

- it implements SMD (Myerson and Weber, 1993), SNTV (Cox, 1994) and Run-off (Bouton, 2013) models as iterative discrete-time algorithms.
- it extends those to include strategic abstention - following Palfrey and Rosenthal (1985) and Demichelis and Dhillon (2010) - and sincere voters
- it both generalizes Myerson's (1998) two-candidate Poisson pivotal probabilities to multi-candidates, with multi-way ties, and to SNTV and runoff and proposes heuristics for those probability calculations.

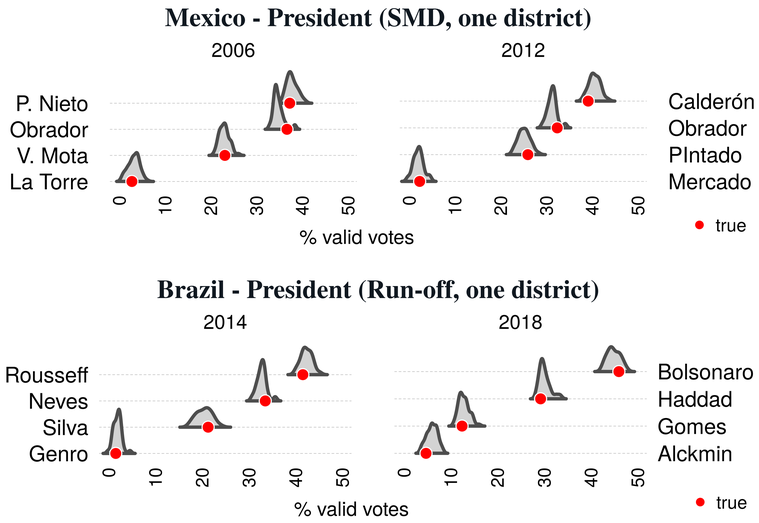
Simulation Overview

In a simulation, there are Nautonomous agents (electors), indexed by $i = 1 \cdots N$, choosing among alternatives (candidates) available in the set $\mathcal{J}$. Vector $\mathbf{u}^i \in \mathbb{Q}_{[0,1]}^{|\mathcal{J}|}$ holds individual utilities that i gets in case each candidate wins. π^i is i's current candidate choice; c^i and q^i are i's cost and current probability of voting. Each iteration, every elector i simultaneously update their current vote choice π^i by maximizing the utility they expect from voting for candidates, i.e. E_j^i, versus from abstaining, i.e. $E_\varnothing^i$, s.t. $\pi^i = \text{argmax}_{j \in \mathcal{J}'}(E_j^i - E_\varnothing^i)$, where and $\mathcal{J}' = \mathcal{J} \setminus \text{argmin}_{h \in \mathcal{J}}(\mathbf{u}_h^i)$. Agents are aware of neither other individuals' preferences or choices, nor of the distribution preferences come from. They know only of candidates' current expected vote shares and with that calculate $(E_j^i - E_\varnothing^i)$ as a function of E_j^i weighted by the probability that voting for each would alter the final outcome. Then, they also update their probability of turning out to vote, by: $q^i = q^i + \text{sign}(\mathbf{u}_{\pi^i}^i - c^i) \cdot \epsilon$, where $\epsilon \in \mathbb{Q}_{[0,1]}$ is any small value (as a learning rate; see Mebane et.al., 2019). Simulation runs until approximate convergence detected.

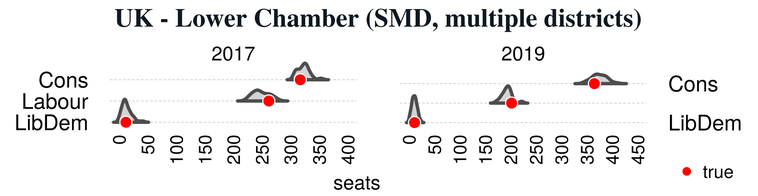
Formulae for $E_j^i - E_\varnothing^i$ and for pivotal probabilities (exact or heuristics) vary per electoral rule-set. Model inputs $\mathbf{u}^i$, c^i and q^i are flexible: they may come from specified distributions, former elections data, surveys, etc.

Application 1: replicating electoral outcomes' DGP

We use surveys to sample elector types and prob. of abstention. Thousands of elections are simulated, with % of sincere voters varying (sampled from $\text{U}(0.5, 0.9)$). Examples of simulated posteriors vs. true actual results:

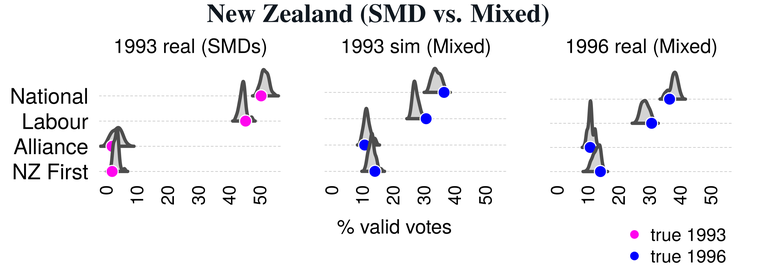


In cases with multiple districts, I use a novel GIS-based imputation to assign voter types to actual districts (more in my APSA 2021 paper):



App. 2: investigating electoral system changes

- Once simulations are validated against actual results, they can be used to explore how electoral system changes could affect the party system.
- For example, again using GIS imputation from surveys, we can use an approximation of Cox and Shugart's (1996) PR model to simulate New Zealand's 1993 under the mixed system that was adopted in 1996:



For more thorough sims. on departures from SMD, see Baltz's poster.

App 3: validating election forensic tools

Problem:

- Because fraud is designed to be hard to spot, false negatives are hard to assess. And as Mebane (2016) suggests, false positives can arise due to buzz caused by valid strategic behavior on the part of electors.
- Our simulations can help, by generating synthetic ground truths (with frauds 'manually' applied).

Synthetic Validation Data:

- Over 3,000 simulations (of SMD, SNTV and Run-off elections), each with between 100-1000 simulated "precints", what is the % of false positives/negatives of fraud detection approaches:

	False +	False -
2-digit mean	7.2%	5.1%
Last dig. mean	2.0%	5.8%
Finite Mix. Model	14.4%	1.3%

More details about this type of application in our Polmeth 2019 paper.

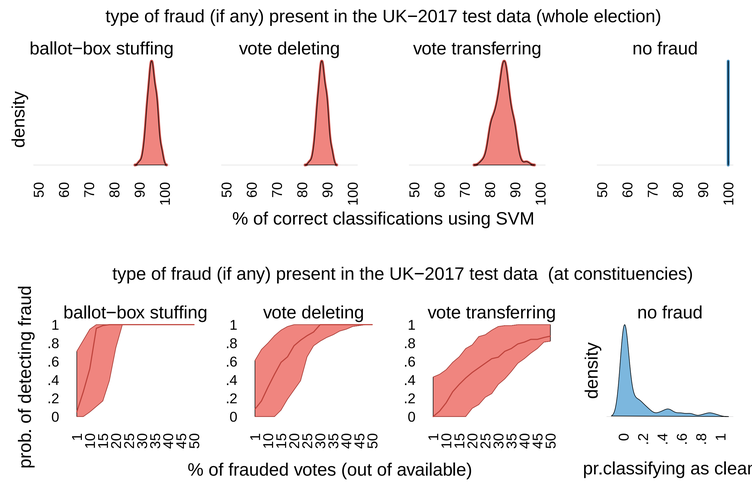
App. 4: synthetic train data for SML fraud detection

Synthetic Training Data:

- 10K plurality contests simulated for each UK constituency, with model inputs from British Electoral Studies. Each time, applying no fraud, ballot-box stuffing, vote deletion or vote transferring (25% prob. each)

Test data:

- many instances of "manually" manipulated vs. clean UK-2017 results



More details about this application in my Polmeth 2020 poster.