Redistricting Simulation through Markov Chain Monte Carlo

Kosuke Imai

Department of Government Department of Statistics
Institute for Quantitative Social Science
Harvard University

SAMSI Quantifying Gerrymandering Workshop
October 8, 2018

Joint work with Benjamin Fifield, Michael Higgins, Jun Kawahara, and Alexander Tarr

Motivation and Progress of Our Team's Efforts

- Redistricting simulation:
  - detect gerrymandering
  - assess impact of constraints (e.g., population, compactness, race)
- In 2013 when our team started working on the project,
  - many optimization methods existed but there were surprisingly few simulation methods
  - no theoretical justification for standard "random seed and grow" algorithms
- Need a simulation method that:
  1. samples uniformly from a target population of redistricting maps
  2. incorporates common constraints
  3. scales to redistricting problems of moderate and large size
- Paper presented at the 2014 Political Methodology Summer Meeting
- Open-source R package redist first published at CRAN in May 2015

Overview of the Talk

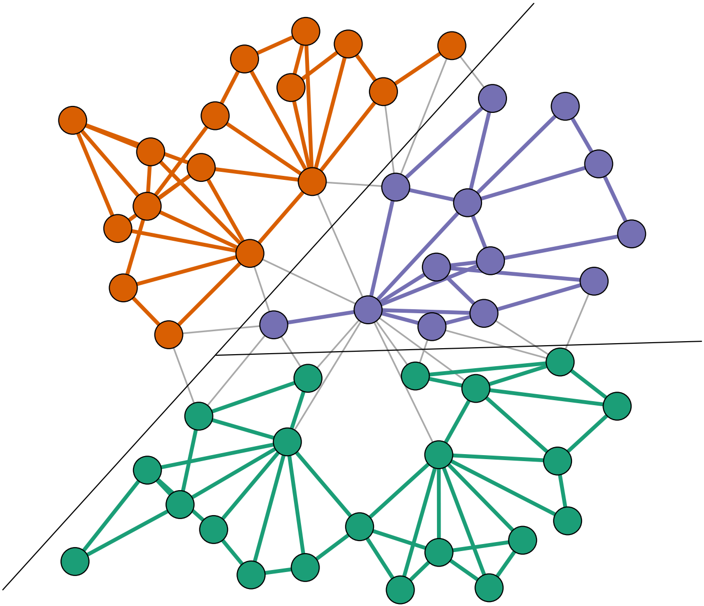
1. Markov chain Monte Carlo algorithms
2. Validation studies
3. Empirical studies

The Random Seed-and-Grow Algorithm

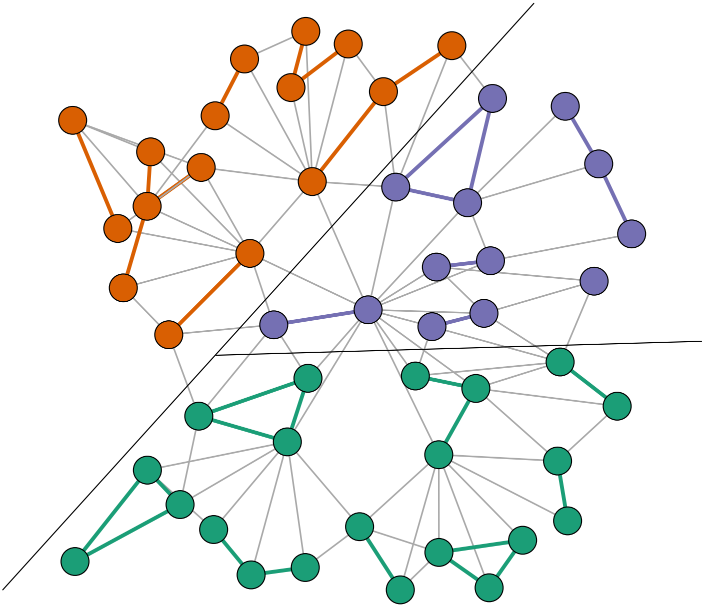
- Cirincione *et. al* (2000), Altman & McDonald (2011), Chen & Rodden (2013):
  1. Randomly choose a precinct as a "seed" for each district
  2. Identify precincts adjacent to each seed
  3. Randomly select adjacent precinct to merge with the seed
  4. Repeat steps 2 & 3 until all precincts are assigned
  5. Swap precincts around borders to achieve population parity
- Modify Step 3 to incorporate compactness

- No theoretical properties known
- The resulting maps may not be representative of the population
- "Local" exploration is difficult

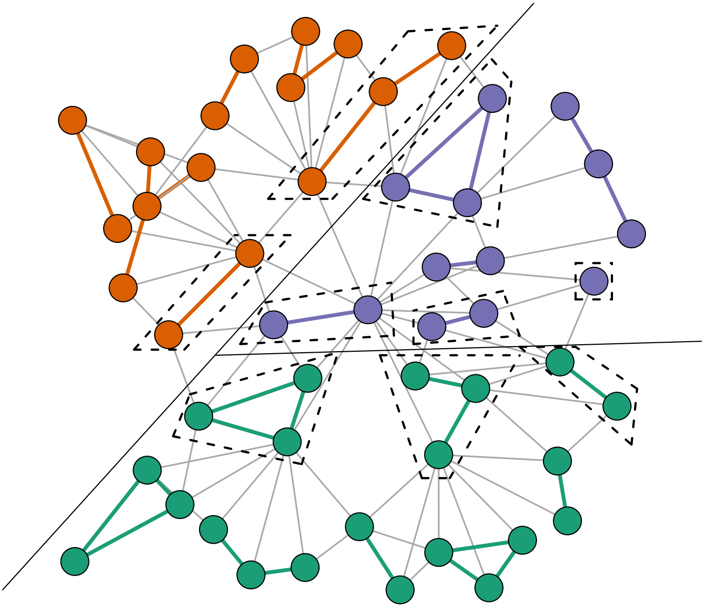
Redistricting as a **Graph-Cut** Problem



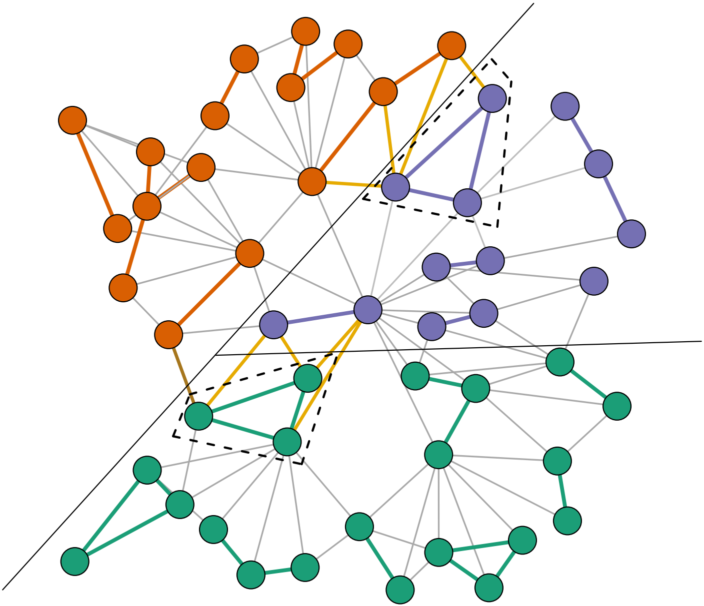
Step 1: Independently "Turn On" Each Edge with Prob. q



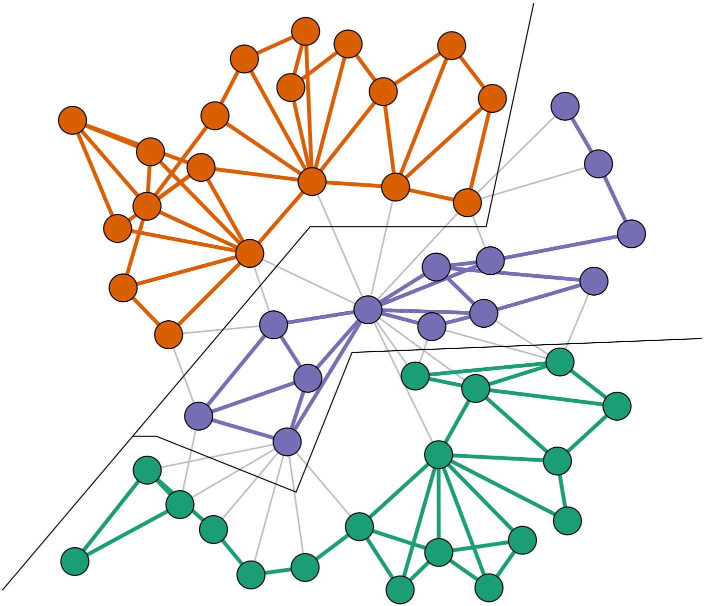
Step 2: Gather Connected Components on Boundaries



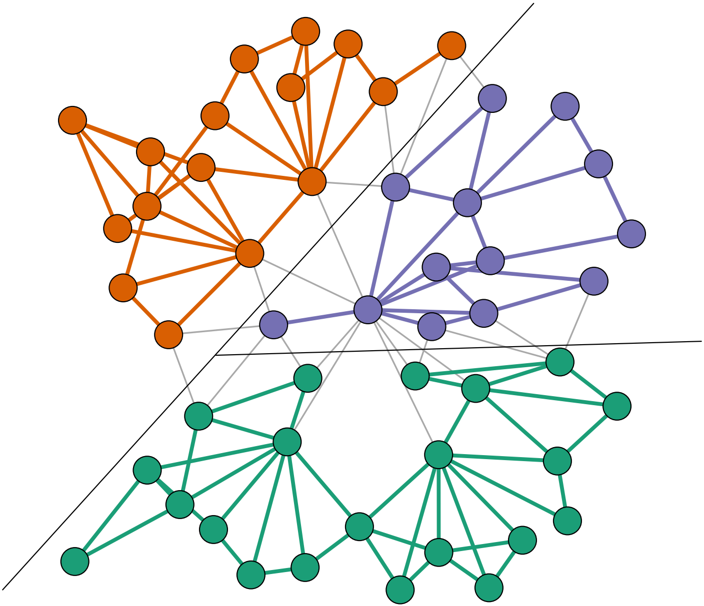
Step 3: Select Subsets of Components and Propose Swaps



Step 4: **Accept** or Reject the Proposal



Step 4: Accept or **Reject** the Proposal



The Theoretical Property of the Algorithm

- We prove that the algorithm samples *uniformly* from the population of all valid redistricting plans
- An extension of the Swendsen-Wang algorithm (Barbu & Zhu, 2005)
- Metropolis-Hastings move from plan $\mathbf{v}_{t-1} \rightarrow \mathbf{v}_t^*$:

$$\alpha(\mathbf{v}_{t-1} \rightarrow \mathbf{v}_t^*) = \min\left(1, \frac{\pi(\mathbf{v}_t^* \rightarrow \mathbf{v}_{t-1})}{\pi(\mathbf{v}_{t-1} \rightarrow \mathbf{v}_t^*)}\right) \approx \min\left(1, \ (1-q)^{|B(C^*,\mathbf{v})|-|B(C^*,\mathbf{v}^*)|}\right)$$

  where q is the edge cut probability and $|B(C^*, \mathbf{v})|$ is # of edges between connected component and its assigned district in redistricting plan $\mathbf{v}$ $\rightsquigarrow$ Easy to calculate
- Exact Metropolis ratio is too costly to evaluate, but approximation appears to work well

Incorporating a Population Constraint

- Want to sample plans where
$$\psi(V_k) \; = \; \left| \frac{p_k}{\bar{p}} - 1 \right| \leq \epsilon$$
where p_k is population of district k, $\bar{p}$ is average district population, ϵ is strength of constraint
- Strategy 1: Only propose "valid" swaps $\rightsquigarrow$ slow mixing
- Strategy 2: Oversample certain plans and then reweight
  1. Sample from target distribution f rather than the uniform distribution:
  $$f(\mathbf{v}) \; \propto \; g(\mathbf{v}) \; = \; \exp\left(-\beta \sum_{V_k \in \mathbf{v}} \psi(V_k)\right)$$
  where $\beta \geq 0$ and $\psi(V_k)$ is deviation from parity for district V_k
  2. (Approximate) Acceptance probability is still easy to calculate,
  $$\alpha(\mathbf{v} \rightarrow \mathbf{v}^*) \; \approx \; \min\left(1, \; \frac{g(\mathbf{v}^*)}{g(\mathbf{v})} \cdot (1-q)^{|B(C^*,\mathbf{v})| - |B(C^*,\mathbf{v}^*)|}\right)$$
  3. Discard invalid plans and reweight the rest by $1/g(\mathbf{v})$

Additional Constraints

1. Compactness (Fryer and Holden 2011):

$$\psi(V_k) \propto \sum_{i,j \in V_k, i<j} p_i p_j d_{ij}^2$$

   where d_{ij} is the distance between precincts i, j

2. Similarity to the adapted plan:

$$\psi(V_k) = \left| \frac{r_k}{r_k^*} - 1 \right|$$

   where r_k (r_k^*) is the # of precincts in V_k (V_k of the adapted plan)

3. Any criteria where constraint can be evaluated at each district

$$g(\mathbf{v}) = \exp\left\{ -\beta \sum_{V_k \in \mathbf{v}} (w_1 \cdot \psi_1(V_k) + w_2 \cdot \psi_2(V_k) + \cdots + w_L \cdot \psi_L(V_k)) \right\}$$

Improving the Mixing of the Algorithm

- Single iteration of the proposed algorithm runs very quickly
- But, like any MCMC algorithm, convergence may take a long time

1. Swapping multiple connected components
   - more effective than increasing q
   - but still leads to low acceptance ratio
2. Simulated tempering (Geyer and Thompson, 1995)
   - Lower and raise the "temperature" parameter β as part of MCMC
   - Explores low temperature space before visiting high temperature space
3. Parallel tempering (Geyer 1991)
   - Run multiple chains of the algorithm with different temperatures
   - Use the Metropolis criterion to swap temperatures with adjacent chains

Validation Studies based on Florida Data

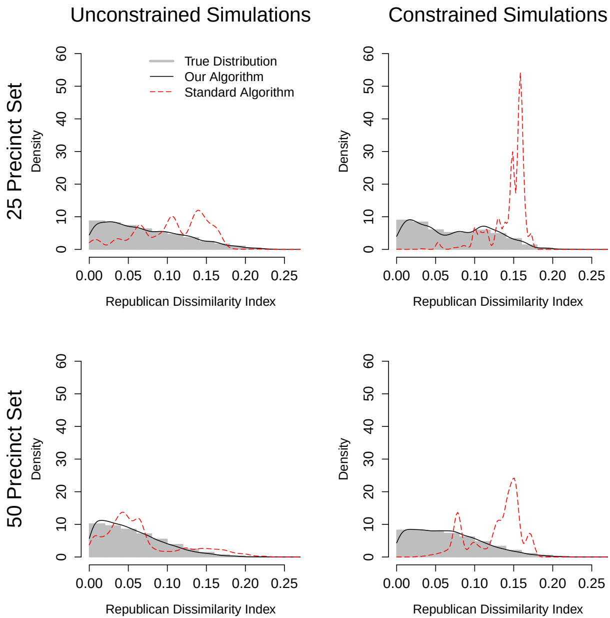
- Evaluate algorithms when all valid plans can be enumerated

- # of precincts: 25 and 50
- # of districts: 2 and 3 for the 25 set, and 2 for the 50 set
- With and without a population constraint of 20% within parity
- Also, consider simulated and parallel tempering

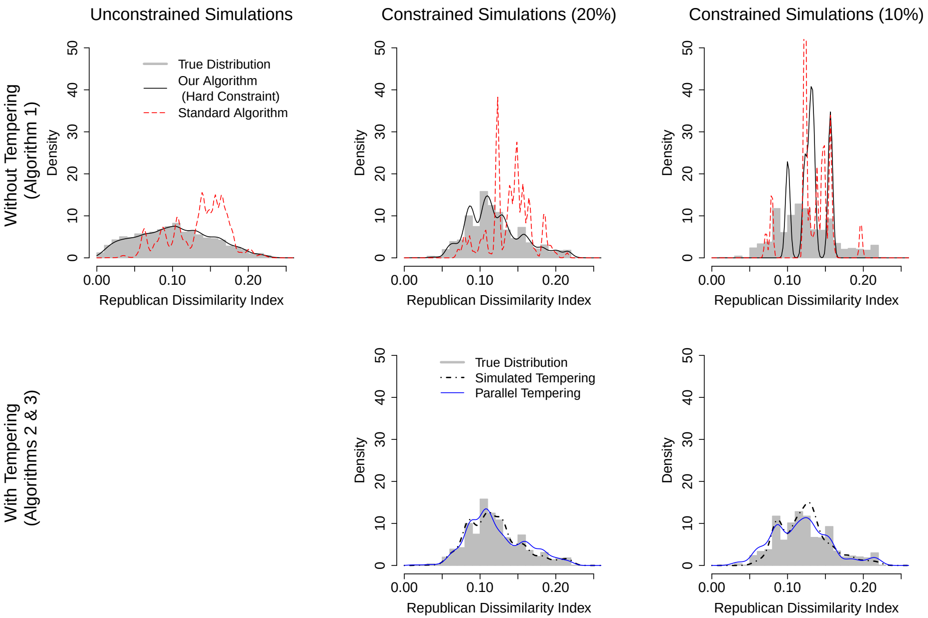
- Comparison with the standard "random seed-and-grow" algorithm via the `BARD` package (Altman & McDonald 2011)
- 10,000 draws for each algorithm
- Republican Dissimilarity Index for each simulated plan:

$$D = \frac{1}{2}\sum_{i=1}^{n} \frac{w_i|R_i - \overline{R}|}{\overline{R}(1-\overline{R})}$$

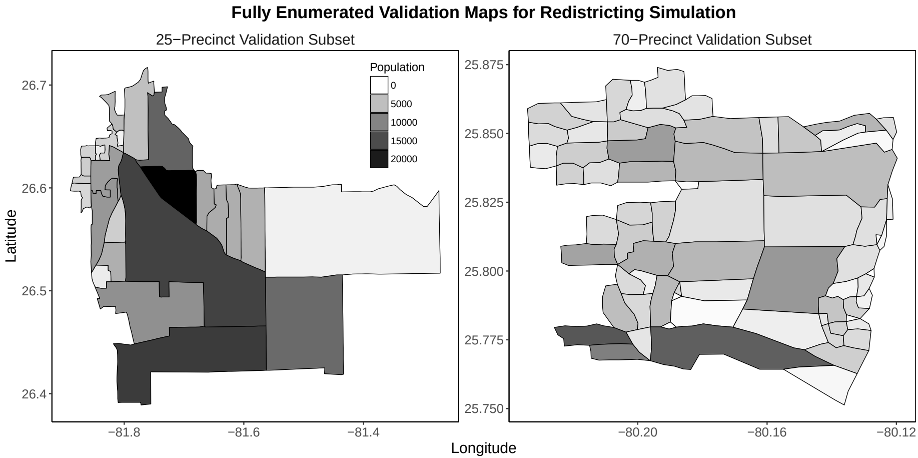
Our Algorithm vs. Standard Algorithm



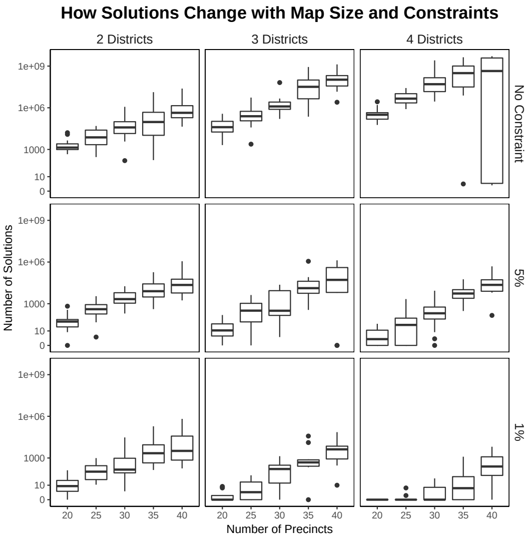
Simulated and Parallel Tempering



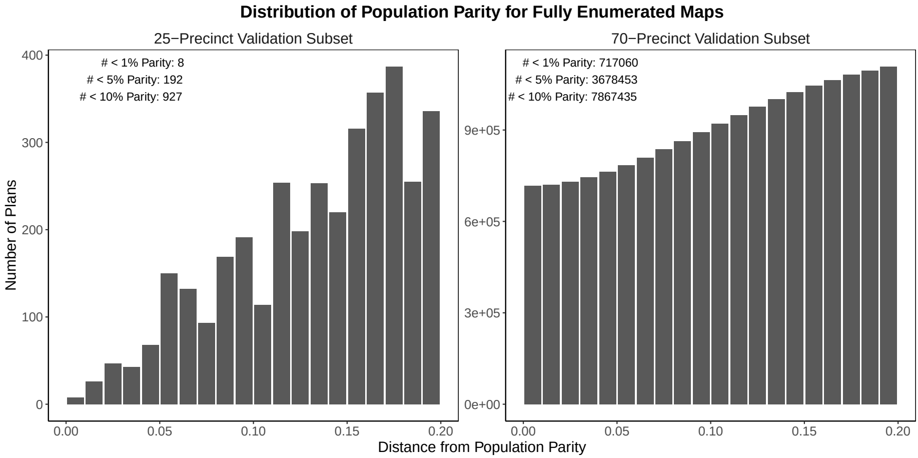
More Validation Studies based on Florida Data



enumpart (Kawahara et al. 2017) Quickly Computes the Total Number of Solutions



Total Number of Solutions and Population Parity

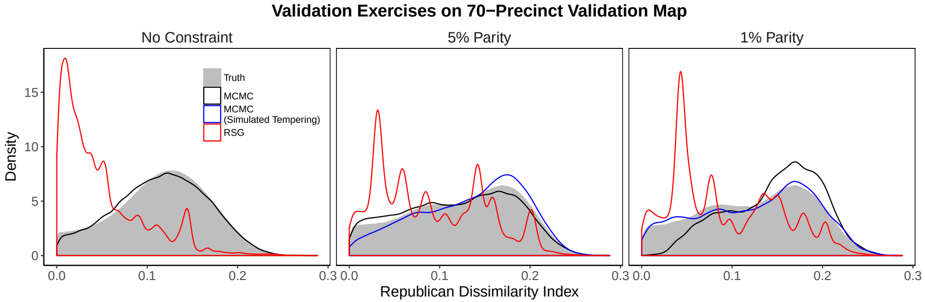


Total number of solutions without population constraint

1. 25 precinct (3 districts) example: 117,688
2. 70 precinct (2 districts) example: 44,082,156

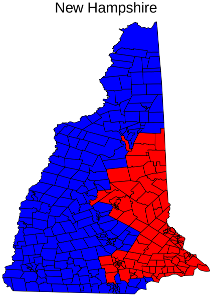
Validation Results

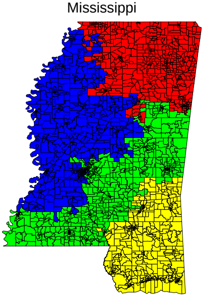
1. Divide 70-precincts into 2 districts: no constraint, 5%, 1%
2. 4 MCMC chains for 50,000 iterations each: with and without simulated tempering
3. Run 200,000 iterations of random seed-and-grow algorithm



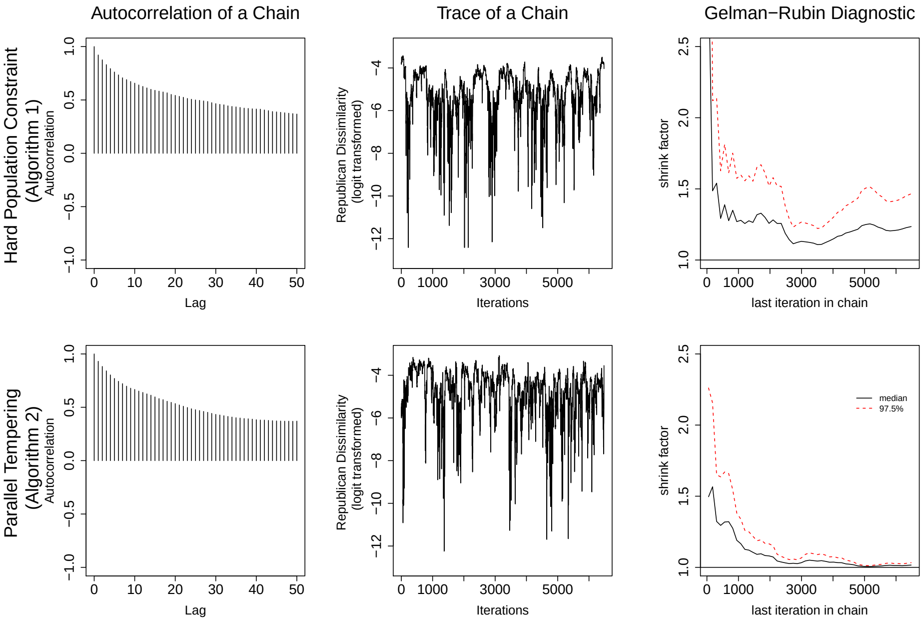
Empirical Studies

- Apply algorithm to state election data:
  1. New Hampshire: 2 congressional districts, 327 precincts
  2. Mississippi: 4 congressional districts, 1,969 precincts
  3. 1% (NH) and 5% (MS) deviation from population parity
- Convergence diagnostics:
  1. Autocorrelation
  2. Trace plot
  3. Gelman-Rubin multiple chain diagnostic

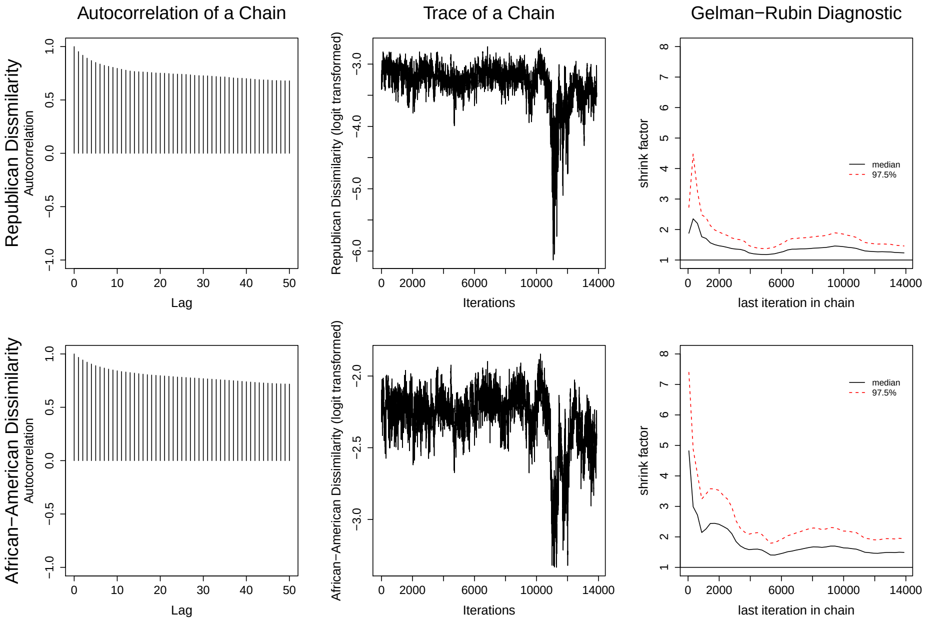




New Hampshire: Tempering Works Better

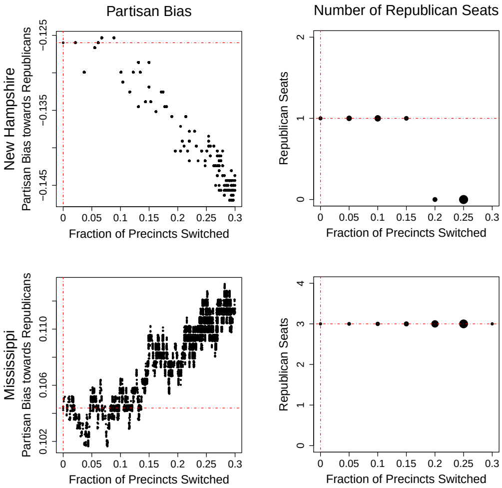


Mississippi: Parallel Tempering, More Challenging Case



Redistricting Plans that are Similar to the Adapted Plan

- Question: How does the partisan bias of the adapted plan compare with that of similar plans? $\rightsquigarrow$ Local exploration



Concluding Remarks

- Scholars use simulations to characterize the distribution of redistricting plans
- Commonly used algorithms lack theoretical properties and speed
- Our MCMC algorithm has:
  - better theoretical properties
  - superior speed
  - better performance in validation studies
  - can do global exploration for small states and local exploration for other states
- Future research:
  - more validation studies
  - more diffused starting maps
  - larger states with more districts and precincts
  - apply the method to historical redistricting data

References

1. Paper at https://imai.fas.harvard.edu/research/redist.html
2. R package at https://github.com/kosukeimai/redist
3. Comments and suggestions: Imai@Harvard.Edu