Mixed Election Simulation

Paul Johnson and Erik Herron[1]

[1]University of Kansas

Swarmfest, Torino Italy

Availability

Code: email me pauljohn@ku.edu
Technical Writeup:
modelDescription_2005_06_06.pdf
This Presentation: elections_swarmfest_2005.pdf

Ordinary Political Model

- Voters have "ideal points" on the real number line
- Candidates and/or parties compete by making promises
- Voters choose most desirable (closest) candidate / party

Ordinary Political Model

- Voters have "ideal points" on the real number line
- Candidates and/or parties compete by making promises
- Voters choose most desirable (closest) candidate / party

Ordinary Political Model

- Voters have "ideal points" on the real number line
- Candidates and/or parties compete by making promises
- Voters choose most desirable (closest) candidate / party

We Think We know that:

- "Single-Member Plurality Elections" lead to 2 party systems
  - Voters select among candidates within "single member district"
  - Usually 2 center parties emerge
- "Proportional Elections" lead to multi-party systems
  - Voters select party
  - Algorithm used to calculate seats from votes (various)
  - More representative, less stable

We Think We know that:

- "Single-Member Plurality Elections" lead to 2 party systems
  - Voters select among candidates within "single member district"
  - Usually 2 center parties emerge
- "Proportional Elections" lead to multi-party systems
  - Voters select party
  - Algorithm used to calculate seats from votes (various)
  - More representative, less stable

We Think We know that:

- "Single-Member Plurality Elections" lead to 2 party systems
  - Voters select among candidates within "single member district"
  - Usually 2 center parties emerge
- "Proportional Elections" lead to multi-party systems
  - Voters select party
  - Algorithm used to calculate seats from votes (various)
  - More representative, less stable

We Think We know that:

- "Single-Member Plurality Elections" lead to 2 party systems
  - Voters select among candidates within "single member district"
  - Usually 2 center parties emerge
- "Proportional Elections" lead to multi-party systems
  - Voters select party
  - Algorithm used to calculate seats from votes (various)
  - More representative, less stable

We Think We know that:

- "Single-Member Plurality Elections" lead to 2 party systems
  - Voters select among candidates within "single member district"
  - Usually 2 center parties emerge
- "Proportional Elections" lead to multi-party systems
  - Voters select party
  - Algorithm used to calculate seats from votes (various)
  - More representative, less stable

We Think We know that:

- "Single-Member Plurality Elections" lead to 2 party systems
  - Voters select among candidates within "single member district"
  - Usually 2 center parties emerge
- "Proportional Elections" lead to multi-party systems
  - Voters select party
  - Algorithm used to calculate seats from votes (various)
  - More representative, less stable

Research Problem

- Mixed Electoral System
  - Some seats allocated by proportional representation
  - Some seats allocated by single-member district contests
- Are these systems the
  - Best of "both worlds"
  - "Bastard Hybrids" without the strengths of either

Research Problem

- Mixed Electoral System
  - Some seats allocated by proportional representation
  - Some seats allocated by single-member district contests
- Are these systems the
  - Best of both worlds
  - "Best and Worst" worlds

Research Problem

- Mixed Electoral System
  - Some seats allocated by proportional representation
  - Some seats allocated by single-member district contests
- Are these systems the
  - Best of "Both Worlds"
  - "Bastard Hybrids" without the strengths of either!

Research Problem

- Mixed Electoral System
  - Some seats allocated by proportional representation
  - Some seats allocated by single-member district contests
- Are these systems the
  - Best of "Both Worlds"
  - "Bastard Hybrids" without the strengths of either!

Research Problem

- Mixed Electoral System
  - Some seats allocated by proportional representation
  - Some seats allocated by single-member district contests
- Are these systems the
  - Best of "Both Worlds"
  - "Bastard Hybrids" without the strengths of either!

Model Voters

- Create districts with ideological tendencies (parameter=mode & diversity).
- Voters within districts assigned preferences from tendencies of the districts.

Model Voters

- Create districts with ideological tendencies (parameter=mode & diversity).
- Voters within districts assigned preferences from tendencies of the districts.

Voter Behavior Types

- Choose Party, Vote for its candidate
- Choose Candidate, Choose SMD candidate separate
- Choose Candidate, Vote for her party

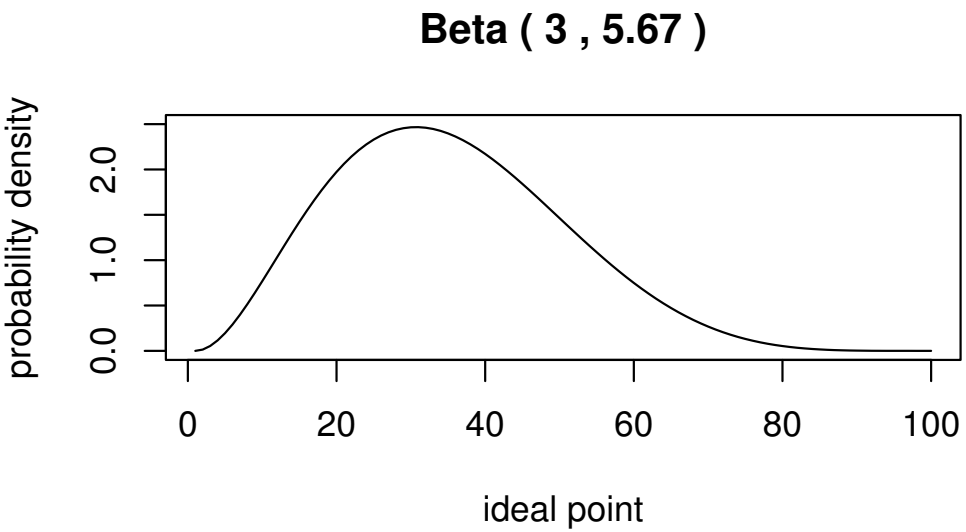
Voter Behavior Types

- Choose Party, Vote for its candidate
- Choose Candidate, Choose SMD candidate separate
- Choose Candidate, Vote for her party

Voter Behavior Types

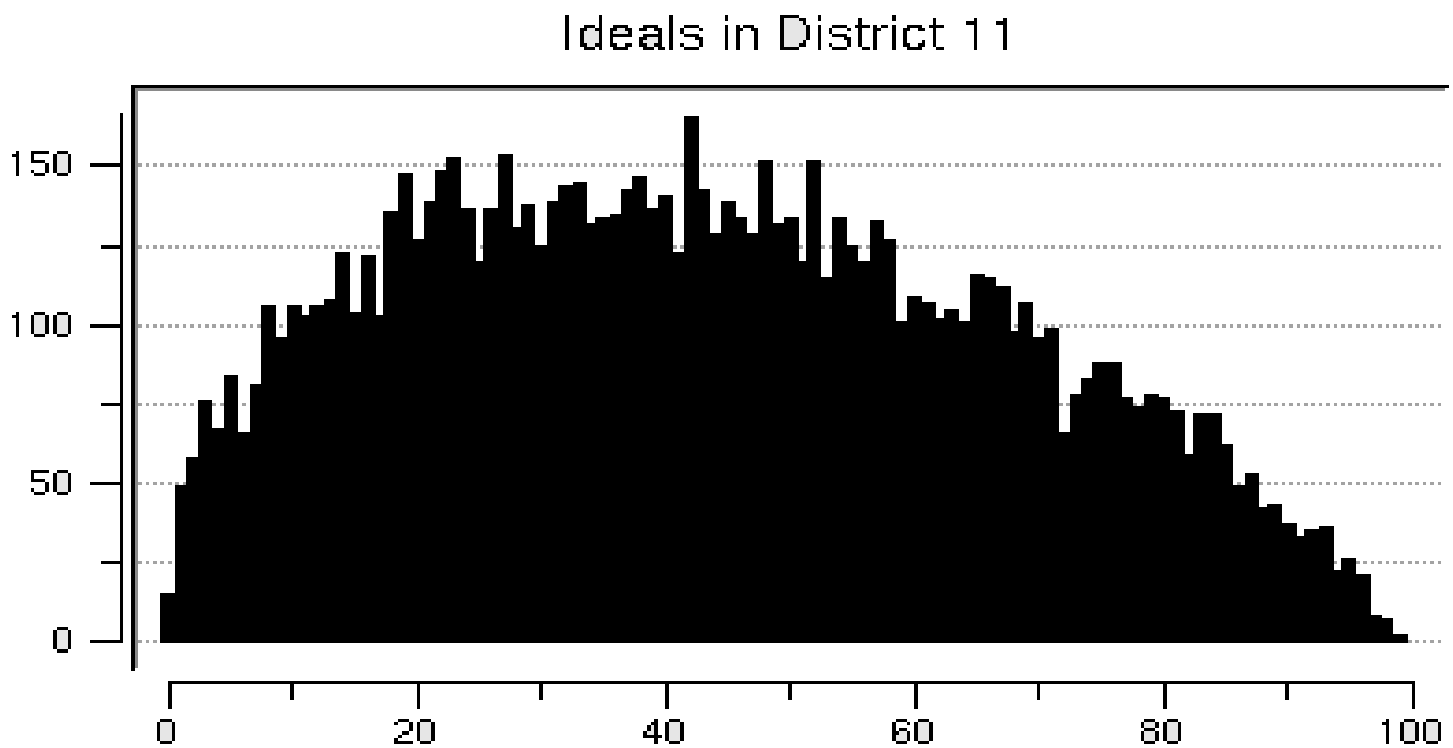
- Choose Party, Vote for its candidate
- Choose Candidate, Choose SMD candidate separate
- Choose Candidate, Vote for her party

Example District



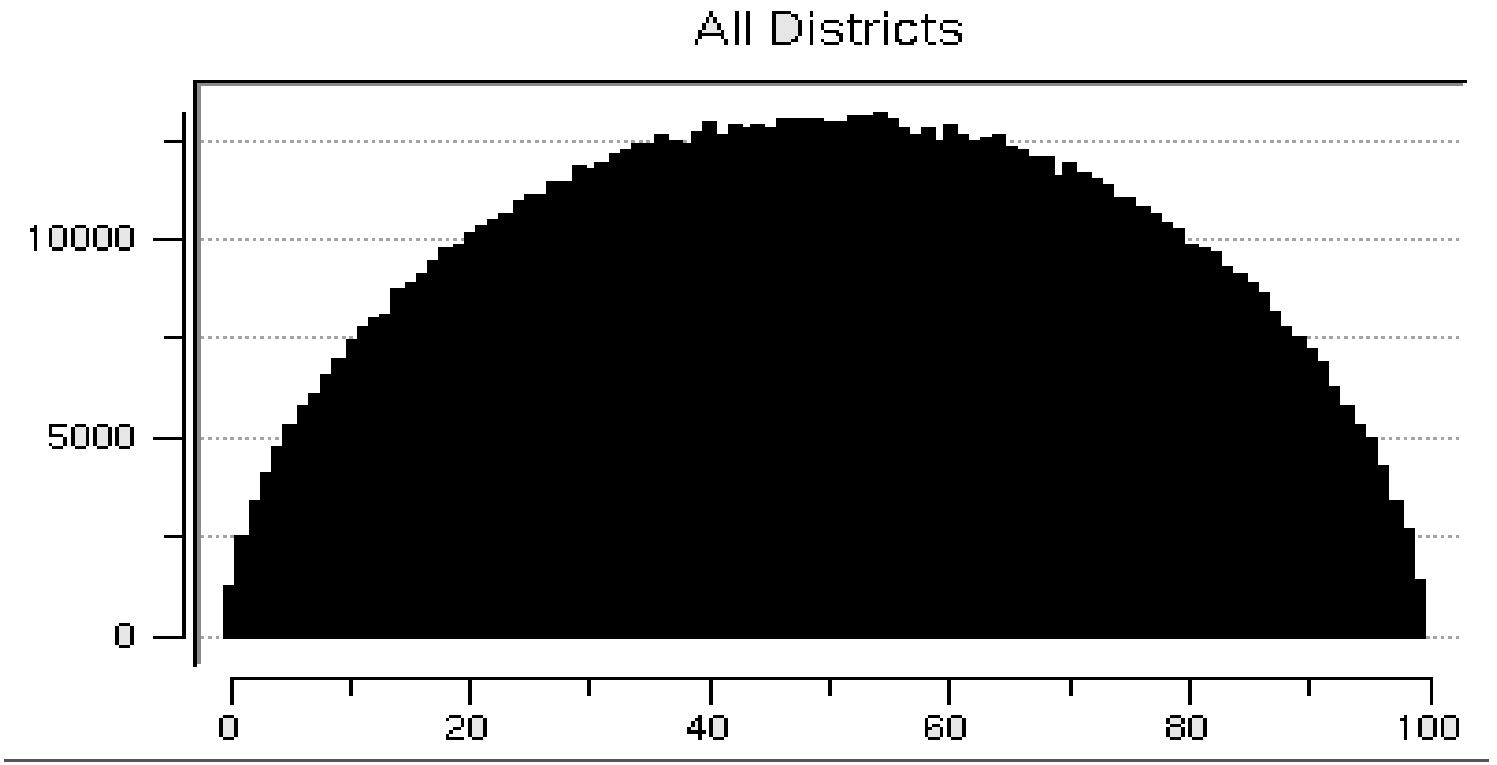
- Beta probability distribution parameterized by the mode & diversity

Realization



National electorate

District Modes Uniform on (30,70)



Bootstrapping a Political System

- Draw "**party founders**" from national electorate
- District level "**electioneers**" register party members within districts
- Parties choose SMD candidates as median of members within district

Bootstrapping a Political System

- Draw "**party founders**" from national electorate
- District level "**electioneers**" register party members within districts
- Parties choose SMD candidates as median of members within district

Bootstrapping a Political System

- Draw "**party founders**" from national electorate
- District level "**electioneers**" register party members within districts
- Parties choose SMD candidates as median of members within district

Do Parties Run Candidates In All Districts?

- If Yes: Simple
- If No: Need algorithm to allocate SMD candidates
- We chose
  - Resource constraint model (membership = scarce resource)
  - Parties allocate SMD candidates to states where they have the most members

Do Parties Run Candidates In All Districts?

- If Yes: Simple
- If No: Need algorithm to allocate SMD candidates
- We chose
  - Resource constraint model (membership = scarce resource)
  - Parties allocate SMD candidates to states where they have the most members

Do Parties Run Candidates In All Districts?

- If Yes: Simple
- If No: Need algorithm to allocate SMD candidates
- We chose
  - Resource constraint model (membership = scarce resource)
  - Parties allocate SMD candidates in districts where they have the most members

Do Parties Run Candidates In All Districts?

- If Yes: Simple
- If No: Need algorithm to allocate SMD candidates
- We chose
  - Resource constraint model (membership = scarce resource)
  - Parties allocate SMD candidates in districts where they have the most members

Do Parties Run Candidates In All Districts?

- If Yes: Simple
- If No: Need algorithm to allocate SMD candidates
- We chose
  - Resource constraint model (membership = scarce resource)
  - Parties allocate SMD candidates in districts where they have the most members

Rubber Meets The Road

- Voters cast votes for both PR and SMD contests
- Electioneers report PR totals and SMD winners to Parliament
- **Parliament** allocates PR seats
  - LR-Hare and Compensated
  - Threshold 0 5 10

Rubber Meets The Road

- Voters cast votes for both PR and SMD contests
- Electioneers report PR totals and SMD winners to Parliament
- **Parliament** allocates PR seats
  - LR-Hare and Compensated
  - Threshold 0 5 10

Rubber Meets The Road

- Voters cast votes for both PR and SMD contests
- Electioneers report PR totals and SMD winners to Parliament
- **Parliament** allocates PR seats
  - LR-Hare and Compensated
  - Threshold 0 5 10

Jargon

1. **Run**: start the model, repeatedly hold elections
2. **Timestep**=one election
3. Between elections:
   - 3.1 Voter ideals can be "re-randomized"
   - 3.2 Party positions can be "re-randomized"
   - 3.3 Party positions might be "adapted" to observed membership tendencies

Jargon

1. **Run**: start the model, repeatedly hold elections
2. **Timestep**=one election
3. Between elections:
   - 3.1 Voter ideals can be "re-randomized"
   - 3.2 Party positions can be "re-randomized"
   - 3.3 Party positions might be "adapted" to observed membership tendencies

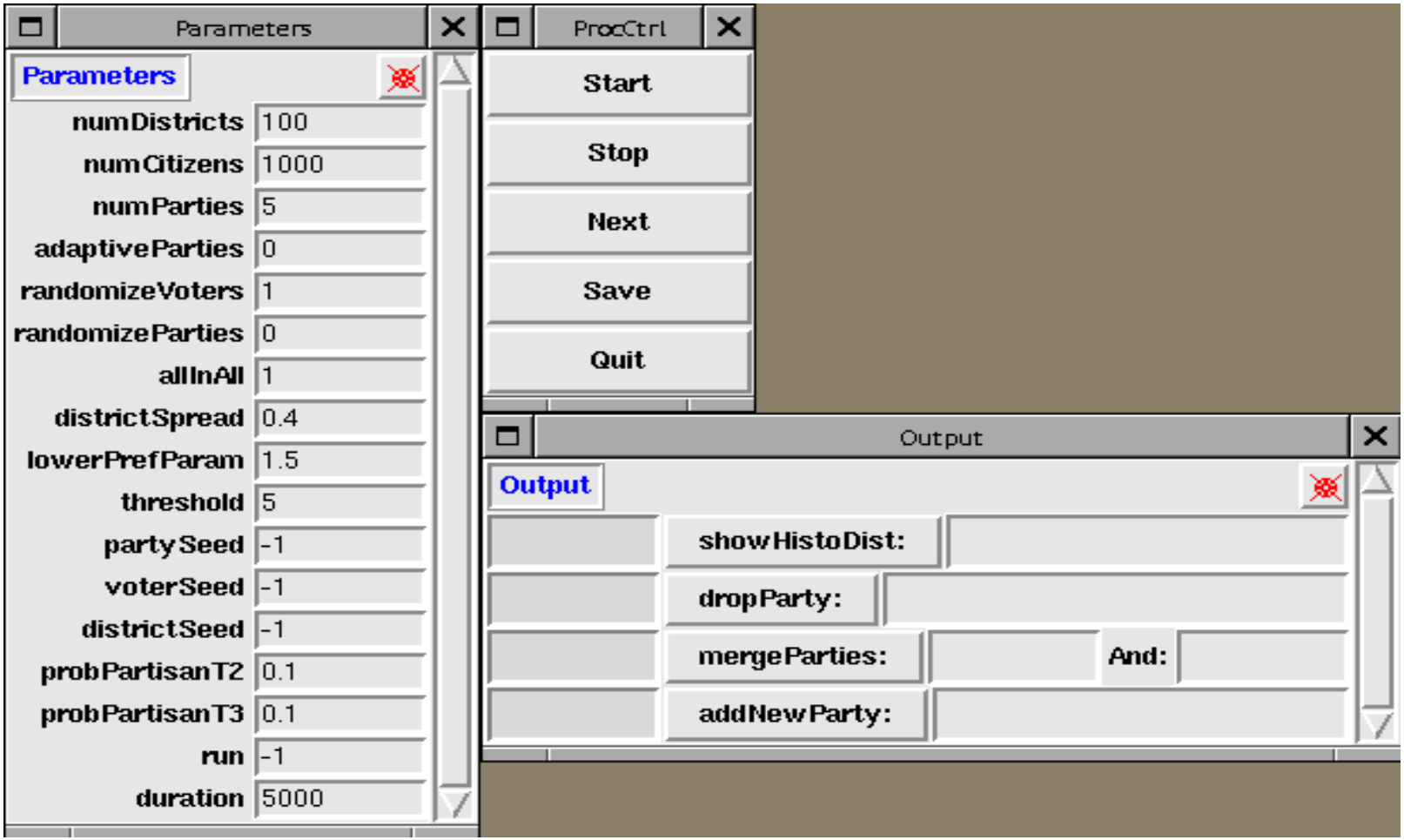
Jargon

1. **Run**: start the model, repeatedly hold elections
2. **Timestep**=one election
3. Between elections:
    - 3.1 Voter ideals can be "re-randomized"
    - 3.2 Party positions can be "re-randomized"
    - 3.3 Party positions might be "adapted" to observed membership tendencies

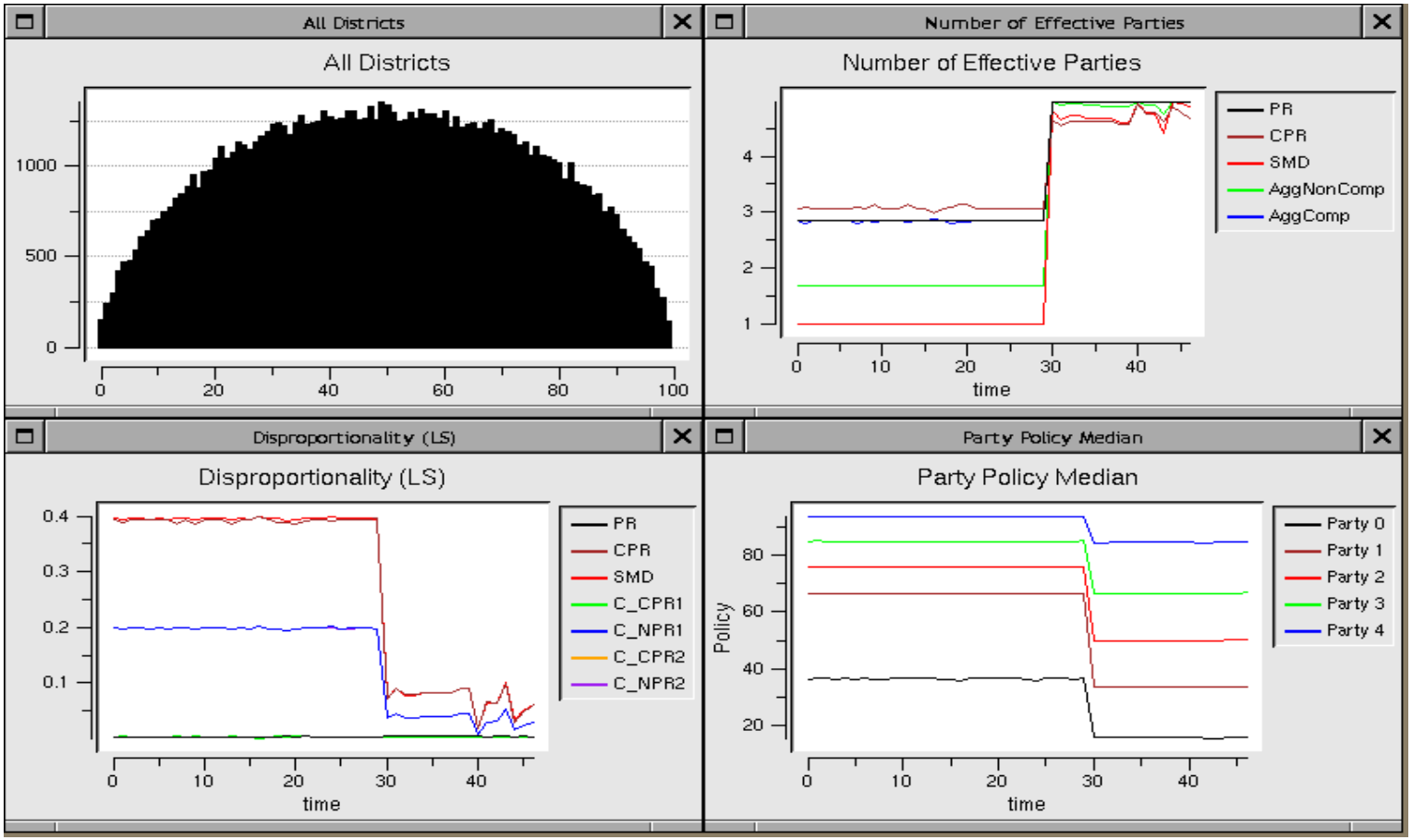
Batch Or Graphical Interface

- Histogram of ideal points
- Median of party SMD candidates
- Party seats
- Indicators of representation

Control Panel



Graphs


All Districts
All Districts
1000
500
0
0
20
40
60
80
100
Number of Effective Parties
Number of Effective Parties
4
3
2
1
0
10
20
30
40
time
PR
CPR
SMD
AggNonComp
AggComp
Disproportionality (LS)
Disproportionality (LS)
0.4
0.3
0.2
0.1
0
10
20
30
40
time
PR
CPR
SMD
C_CPR1
C_NPR1
C_CPR2
C_NPR2
Party Policy Median
Party Policy Median
Policy
80
60
40
20
0
10
20
30
40
time
Party 0
Party 1
Party 2
Party 3
Party 4

Batch Design

100 runs with each of

- Number of Parties = 5 , 10, 15
- Voter Type Distributions
  $(1, 0, 0), (0, 1, 0), (0, 0, 1), (1/3, 1/3, 1/3)$
- PR thresholds 0, 5, 10
- All Parties Run SMD in All Districts 0, 1

Condition Sequence of Batch Simulation

- 30 steps: Randomly chosen party founders "stuck" in position between elections
  - Voter ideal points re-drawn
- ?? steps: Parties adapt dynamically, founder stands at "median of candidate positions" from previous election
  - Voter ideal points are a "fixed target"
- 30 steps (after equilibration): Parties adapt dynamically with randomly re-draw voters
- 100 steps: Fix voters, Randomly Choose party founders.

Condition Sequence of Batch Simulation

- 30 steps: Randomly chosen party founders "stuck" in position between elections
  - Voter ideal points re-drawn
- ?? steps: Parties adapt dynamically, founder stands at "median of candidate positions" from previous election
  - Voter ideal points are a "fixed target"
- 30 steps (after equilibration): Parties adapt dynamically with randomly re-draw voters
- 100 steps: Fix voters, Randomly Choose party founders.

Condition Sequence of Batch Simulation

- 30 steps: Randomly chosen party founders "stuck" in position between elections
  - Voter ideal points re-drawn
- ?? steps: Parties adapt dynamically, founder stands at "median of candidate positions" from previous election
  - Voter ideal points are a "fixed target"
- 30 steps (after equilibration): Parties adapt dynamically with randomly re-draw voters
- 100 steps: Fix voters, Randomly Choose party founders.

Condition Sequence of Batch Simulation

- 30 steps: Randomly chosen party founders "stuck" in position between elections
  - Voter ideal points re-drawn
- ?? steps: Parties adapt dynamically, founder stands at "median of candidate positions" from previous election
  - Voter ideal points are a "fixed target"
- 30 steps (after equilibration): Parties adapt dynamically with randomly re-draw voters
- 100 steps: Fix voters, Randomly Choose party founders.

Condition Sequence of Batch Simulation

- 30 steps: Randomly chosen party founders "stuck" in position between elections
  - Voter ideal points re-drawn
- ?? steps: Parties adapt dynamically, founder stands at "median of candidate positions" from previous election
  - Voter ideal points are a "fixed target"
- 30 steps (after equilibration): Parties adapt dynamically with randomly re-draw voters
- 100 steps: Fix voters, Randomly Choose party founders.

Condition Sequence of Batch Simulation

- 30 steps: Randomly chosen party founders "stuck" in position between elections
  - Voter ideal points re-drawn
- ?? steps: Parties adapt dynamically, founder stands at "median of candidate positions" from previous election
  - Voter ideal points are a "fixed target"
- 30 steps (after equilibration): Parties adapt dynamically with randomly re-draw voters
- 100 steps: Fix voters, Randomly Choose party founders.

Actual demonstration!

No extra charge!

Interesting Bug Invented

- Caution about "local variables" and references
- OK to do

```
- someMethod {
    id anAgent = [AClass create: self];
    [otherClass setAgent: anAgent]; }
```

- **anAgent** is a LOCAL VARIABLE.
- Is it safe to allow otherClass to "use" that agent after code exits this method?
- That is OK because create grabs memory for **anAgent** and it is held until **anAgent** is dropped.

Interesting Bug Invented

- Caution about "local variables" and references
- OK to do

```
- someMethod {
id anAgent = [AClass create: self];
[otherClass setAgent: anAgent]; }
```

- **anAgent** is a LOCAL VARIABLE.
- Is it safe to allow otherClass to "use" that agent after code exits this method?
- That is OK because create grabs memory for **anAgent** and it is held until **anAgent** is dropped.

Interesting Bug Invented

- Caution about "local variables" and references
- OK to do

```
- someMethod {
id anAgent = [AClass create: self];
[otherClass setAgent: anAgent]; }
```

- **anAgent** is a LOCAL VARIABLE.
- Is it safe to allow otherClass to "use" that agent after code exits this method?
- That is OK because create grabs memory for **anAgent** and it is held until **anAgent** is dropped.

Interesting Bug Invented

- Caution about "local variables" and references
- OK to do

```
- someMethod {
id anAgent = [AClass create: self];
[otherClass setAgent: anAgent]; }
```

- **anAgent** is a LOCAL VARIABLE.
- Is it safe to allow otherClass to "use" that agent after code exits this method?
- That is OK because create grabs memory for **anAgent** and it is held until **anAgent** is dropped.

Interesting Bug Invented

- Caution about "local variables" and references
- OK to do

```
- someMethod {
id anAgent = [AClass create: self];
[otherClass setAgent: anAgent]; }
```

- **anAgent** is a LOCAL VARIABLE.
- Is it safe to allow otherClass to "use" that agent after code exits this method?
- That is OK because create grabs memory for **anAgent** and it is held until **anAgent** is dropped.

But...

- Suppose instead you try

```
char * colors = {"white","green","blue"}
prSeatGraph = [EZGraph createBegin: self];
[prSeatGraph setColors: colors count: 3];
prSeatGraph = [prSeatGraph createEnd];
```

- That's headed for a **big crash**
- EZGraph does not copy the array into a data structure, it just makes a "note" of where the data is.
- The next time EZGraph tries to find the colors, they are not there

But...

- Suppose instead you try

```
char * colors = {"white","green","blue"}
prSeatGraph = [EZGraph createBegin: self];
[prSeatGraph setColors: colors count: 3];
prSeatGraph = [prSeatGraph createEnd];
```

- That's headed for a **big crash**
- EZGraph does not copy the array into a data structure, it just makes a "note" of where the data is.
- The next time EZGraph tries to find the colors, they are not there

But...

- Suppose instead you try

```
char * colors = {"white","green","blue"}
prSeatGraph = [EZGraph createBegin: self];
[prSeatGraph setColors: colors count: 3];
prSeatGraph = [prSeatGraph createEnd];
```

- That's headed for a **big crash**
- EZGraph does not copy the array into a data structure, it just makes a "note" of where the data is.
- The next time EZGraph tries to find the colors, they are not there

But...

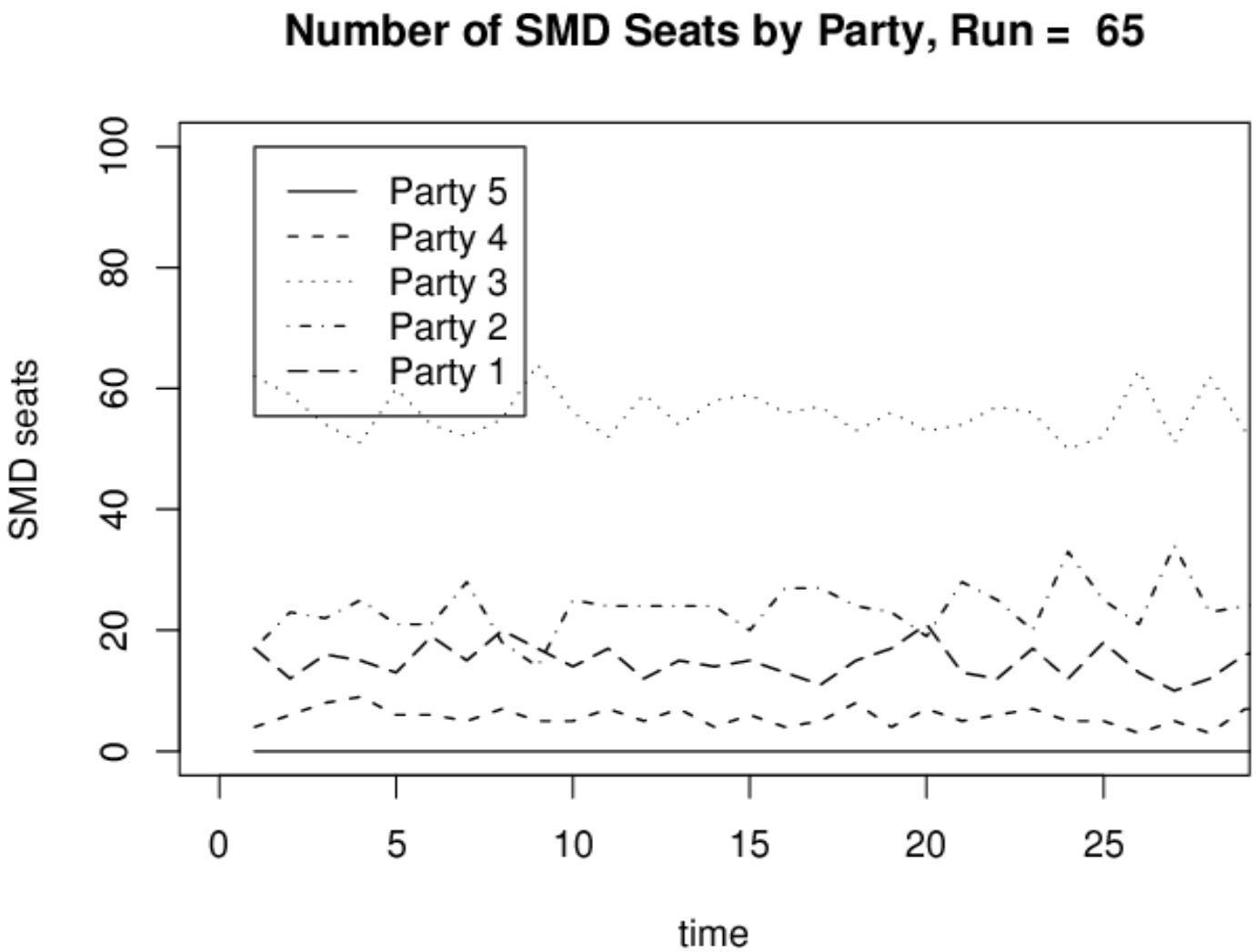
- Suppose instead you try

```
char * colors = {"white","green","blue"}
prSeatGraph = [EZGraph createBegin: self];
[prSeatGraph setColors: colors count: 3];
prSeatGraph = [prSeatGraph createEnd];
```

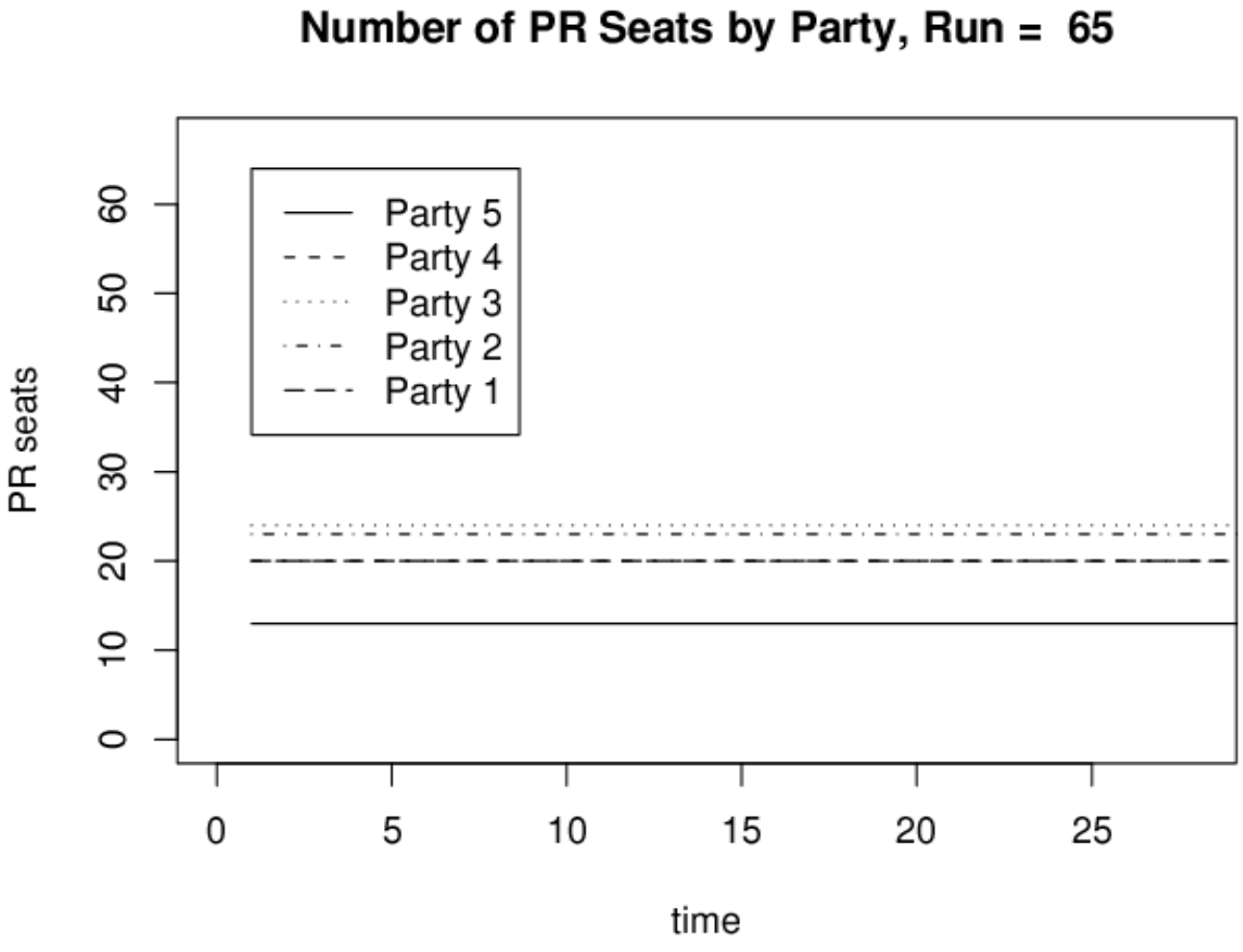
- That's headed for a **big crash**
- EZGraph does not copy the array into a data structure, it just makes a "note" of where the data is.
- The next time EZGraph tries to find the colors, they are not there

What do we find out?

SMD is a lot more exciting than PR



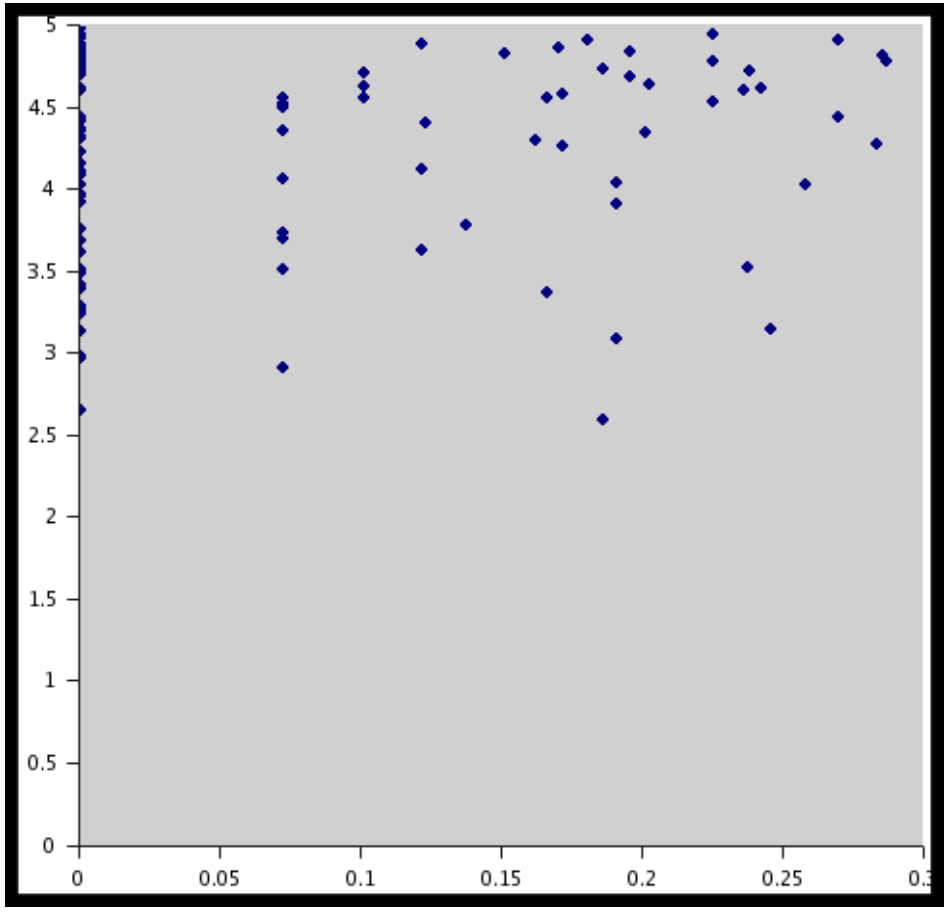
But With PR


Number of PR Seats by Party, Run = 65
Party 5
Party 4
Party 3
Party 2
Party 1
PR seats
0
10
20
30
40
50
60
0
5
10
15
20
25
time

Instability Index

Calculate standard deviation in Number of Seats for each party
Instability Index=Average of those standard deviations

PR



Vert=#N of Effective Parties Horiz=Instability index

But SMD is quite different