Electoral Engineering through Simulation

October 22, 2012

Author:
Michael Fowlie

Supervisor:
Mark C. Wilson

COMPSCI 380 - Undergraduate Project in Computer Science
Semester 2, 2012
University of Auckland

Contents

1 Abstract **4**

2 Introduction **5**

3 Methodology **5**
3.1 Hypothesis . . . 5
3.2 Procedure . . . 5
3.2.1 Generating Data . . . 5
3.2.2 Statistical Analysis . . . 6
3.3 Tools . . . 6

4 Measures **6**
4.1 Measures of Unfairness . . . 7
4.2 Measures of Instability . . . 7
4.2.1 Effective Number of Parties . . . 7
4.2.2 ENP using Shapley Shubik Power Index . . . 7
4.2.3 Entropy of Shapley Shubik Power . . . 8

5 Artificial Societies **8**
5.1 Polya Eggenberger (Urn) Model . . . 8
5.1.1 Urn Process . . . 8
5.1.2 Parameter selection . . . 8
5.1.3 Sampling . . . 8
5.2 Spatial Model . . . 9
5.3 Preference Swapping Model . . . 9
5.3.1 Redistricting . . . 10

6 Apportionment Methods **11**
6.1 St Lague Method . . . 11
6.2 D'Hondt/Jefferson Method . . . 11
6.3 Hill Method . . . 11
6.4 Hamilton Method . . . 12

7 Fitting real world data to the Spatial Model **12**

8 District Magnitude **14**

9 Voting Rules **14**
9.1 Single Transferable Vote . . . 14
9.2 Proportional Systems . . . 14
9.2.1 District Proportional . . . 14
9.2.2 Supplementary Member . . . 15

10 Efficiently Sampling from a Discrete Probability Distribution **15**

11 Undesirability **16**

12 Results **16**
12.1 Simulation Results 16
12.2 Minimum Undesirability 30
12.3 Analysis of Different Apportionment Methods 30

13 Discussion **36**

14 Conclusion **38**

15 Possible Further Research **38**

References **40**

A FSharp Sourcecode **41**

B R Sourcecode **46**

1 Abstract

It is claimed in (Carey and Hix, 2011) that the trade-off between expected fairness and stability in election systems is non-linear and that small medium sized District Magnitudes are optimal, based on outcomes of elections in the real world. Does this apply in general or can this effect be explained by other factors? We test the hypothesis that the trade-off is non-linear against artificial societies and across a range of voting systems. We use the Spatial Model, Polya Eggenberger Model and a preference swapping model and the STV, District Proportional and SM rules. Our results show clearly that there is non-linearity in this trade-off however the shape of the curve differs greatly depending on the system we use. We compare these different outcomes under a range of different undesirability functions that value the properties differently. We find that a District Magnitude of $DM = 3$, $DM = 8$ or $DM = 20$ is optimal, depending on how the properties are valued with respect to each other.

2 Introduction

In relation to election outcomes, there are two properties that we are interested in. These are fairness/proportionality and stability/governability. Under different voting rules (and different parameters for these rules), election outcomes have different tendencies for values of these properties. A high score for each of these properties[1] is desirable, however it is well understood that there is a trade-off between the two.

(Carey and Hix, 2011) uses data from 610 elections in 81 countries. It categorised voting rules based on District Magnitude (see Section 8). We investigate this trade-off in the context of artificially generated societies and in particular look for non linearity.

We use the STV (see Section 9.1) and DP (see Section 9.2.1) methods and compare results by varying the District Magnitude (see Section 8).

It is claimed that the trade-off between these two properties is non linear (Carey and Hix, 2011). As District Magnitude increases, fairness usually increases at the expense of stability; however as we demonstrate this is not always the case. Furthermore, in some cases the opposite effect occurs.

If there is non linearity in the trade-off, it follows that there must be an arguably optimal value of District Magnitude. However there is no clear definition of a desirability function as some people may value one property more than the other.

We compare the results under a number of different undesirability functions which vary the relative contributions of fairness and stability to the result. In each case we calculate the District Magnitudes that optimizes the function in question and thus calculate the District Magnitudes that tend to optimise most of these functions. We find that the optimal case is $DM = 3$, 8 or 20.

3 Methodology

3.1 Hypothesis

Our hypothesis H_1: There is a non-linear relationship between mean scores for fairness and stability under the different voting systems discussed in Section 9 and various parameters for these voting systems. We use the Gallagher Index (see Section 4.1) as a measure of fairness and the ENP Index (see Section 4.2.1) as a measure of stability, so that our results are comparable with (Carey and Hix, 2011). We also use one other measure of stability and one other measure of fairness as discussed in Section 4.2.2 and Section 4.1 respectively. All permutations of these measures are considered.

3.2 Procedure

3.2.1 Generating Data

To evaluate the different artificial societies and different voting systems, we consider these in separate runs of the simulator.

In each run of the simulator, we generate a large number of elections for each value of the property that we vary [2]. This property is usually District Magnitude (see Section 8). Each election is generated with a fixed number of seats S and, as the District Magnitude changes, the number of Districts D also changes so that $S = DM \cdot D$. This also forces us to only consider District

[1] Although as we measure the opposite of these properties, a low score of the opposite is desirable.)

[2] This is typically 2000 although we vary this depending on the computational cost of a simulation.

Magnitudes that are an integer divisor of the number of seats. We chose to consider the case with $S = 120$ because it has a large number of divisors and is within an order of magnitude of typical values. This is also the value used in New Zealand. We also considered the case with a larger value $S = (5) \cdot (120) = 600$ which is near to the number used in the United Kingdom. We chose 600 as it is a multiple of 120 and thus shares all the divisors of 120.

Each election is generated independently, for example with the Spatial Model we define points for each party and each electorate in the Space of Issues in each election.

3.2.2 Statistical Analysis

We fit a regression curve to the measure of fairness and a regression curve to the measure of stability, using DM (see Section 8) as the explanatory variable.

see Section 12.3 for technical notes.

3.3 Tools

The simulator itself consists of a library written in the Visual C# ®[3] 5.0 programming language along with scripting code written in the Visual C# and Visual F# 2.0 programming languages (F# is a variant of the ML functional programming language). Code was written using the Visual Studio ® 2010 and Visual Studio 2012 IDEs and the RGui and Notepad++ editors. Simulations are performed on Amazon EC2™[4] c1.xlarge instances with Intel ®[5] Xeon ® E5410 processors and a workstation with an Intel Pentium ® G620 processor.

Data from (Vowles et al., 2008) was imported and processed with scripts using SAS ®[6] software and Visual C#.

The resulting dataset is then analysed using R 2.15.1.

4 Measures

We actually measure the *opposite* of the desirable properties, that is, we measure unfairness and instability.

Unfairness and Instability are measured in a variety of ways as described below.

Note that in our plots we normalize measures so the minimum value for each property is optimal. For the measures 4.2.1 and 4.2.2 this involves subtracting 1 from the values so that 0 is optimal. For the measure 4.2.3 we multiply the values by -1 so that lower plotted values are better. The Gallagher Index and Loosemorehanby Index are already optimal with values at 0.

[3]Visual C# and Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

[4]Amazon Web Services, the "Powered by Amazon Web Services" logo, EC2 are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries.

[5]Intel, Xeon and Pentium are trademarks of Intel Corporation in the U.S. and/or other countries.

[6]SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

4.1 Measures of Unfairness

We define δ_i such that $\delta_i = v_i - s_i$

where v_i is the proportion of votes of party i and

s_i is the proportion of seats of party i.

We use the Loosemorehanby Index $D = \frac{1}{2}\sum_{i=1}^{m} |\delta_i|$

and the Gallagher Index $D = \sqrt{\frac{1}{2}\sum_{i=1}^{m} \delta_i^2}$. (Benoit, 2000)

4.2 Measures of Instability

4.2.1 Effective Number of Parties

As a measure of ungovernability, we use the Effective Number of Parties. $ENP = \frac{1}{\sum_{i=1}^{m} p_i^2}$ (Laakso and Taagepera, 1979) where p_i is the proportion of seats that party i has and m is the number of parties.

4.2.2 ENP using Shapley Shubik Power Index

As for the purposes of governability, only power is relevant (as opposed to the proportion of seats), it seems reasonable to modify the ENP formula to use political power instead of the proportion of seats, by party. $ENP = \frac{1}{\sum_{i=1}^{m} SSPI_i^2}$

$SSPI_i$ denotes the Shapley Shubik Power of party i.

We consider an arbitrary order over all parties. We consider all parties ranked before party i to be part of one coalition and all parties ranked after party i to be part of another coalition. We repeat the analysis for every possible permutation over all parties and every possible value of i in the case of each permutation.

Party i is not considered to be part of either coalition. Party i is pivotal if and only if party i joining one of the two possible coalitions would cause either to become the majority depending on which it joins[7]. We define $pivotal_i$ to be the number of times party i is pivotal over all possible orders.

$SSPI_i = \frac{pivotal_i}{\sum pivotal}$ (Shapley and Shubik, 1954)

i.e. $SSPI_i$ denotes the proportion of orders in which party i is pivotal.

For example a parliament with a party having 0.51 of the seats has the same amount of governability as one with a party with 0.99 of the seats. In both of these cases the party in question has all of the power as defined by the Shapley Shubik power index, however the ENP formula distinguishes between these two cases.

This measure has, in some cases, the interesting effect of measuring an increase of stability when reducing the threshold under MMP. Consider the following election (See Figure 1) where v_i denotes the number of votes of party i, p_i denotes the Shapley Shubik power of party i with a 10% threshold, and q_i denotes the Shapley Shubik power of party i with a zero threshold. Note that a *lower* value for ENP is considered more stable.

[7] i.e. the coalition has more than 0.5 of the total vote share.

4.2.3 Entropy of Shapley Shubik Power

We also consider the entropy of the Shapley Shubik index. $\sum_{i=1}^{m} SSPI_i \cdot \ln(SSPI_i); SSPI_i \neq 0$

5 Artificial Societies

In each society we define m parties and $M = m!$ preference orders, where a preference order is an order over all parties. We define n as the total number of voters in a given district.

5.1 Polya Eggenberger (Urn) Model

In each district we sample n preference orders, each representing a single voter, from the Polya Eggenberger distribution with parameter a where a represents the homogeneity of preferences in a given district. We assume independence of districts and there is no bias in favour of any particular party, as the number of districts approaches ∞ the distribution of votes to parties will approximate a uniform distribution - thus this model is not very realistic.

5.1.1 Urn Process

The Polya Eggenberger distribution can be understood via intuition from the urn process. The urn contains multiple balls, each of a different colour (where a colour represents preference order). To sample using the urn process we select a ball from the urn uniformly at random, record the colour, and place the ball and a copies of the ball into the urn. We define the weight of a preference order (colour of ball) to be equivalent to the number of balls in the urn, and the weight of the urn to be equal to the total weight of all colours of balls in the urn, however, it is important to note that a (and thus weight) need not be an integer.

5.1.2 Parameter selection

We use both the case with a constant a for all districts, and the case for a random a, where we first define $b = U(0, 1)$ and then let $a = b/(1 - b)$.

5.1.3 Sampling

A naive approach to sampling a value x from the Polya Eggenberger distribution would be to store the weights of each preference order in a vector w then select a value uniformly at random

Votes	p_i	q_i
47	0.3333	0.6667
26	0.3333	0.1667
26	0.3333	0.1667
01	0.0000	0.0000
$\sum_{i=1}^{m} SSPI_i^2$	0.3333	0.5000
$\frac{1}{\sum_{i=1}^{m} SSPI_i^2}$	3	2

Figure 1: Under some elections, increasing the number of parties increases stability when measuring as described in section 4.2.2.

$y = U(0, \sum_{i=1}^{M} w_i)$ then select the minimum value of x such that $\sum_{i=1}^{x}(w_i) \geq y; x \leq M$. We would then adjust the weight for the selected preference order w_x by adding a.

This has time complexity of $O(M)$ per sample, or $O(n \cdot M)$ to sample a single district, and space complexity $O(M)$. Note that $M = m!$ can be quite large. We use a more efficient method to sample, by replacing x with a compressed data structure (see Section 10) with worst case space complexity $O(min(n \cdot \ln(M), M))$ and worst case time complexity $O(n \cdot \ln M)$ (per district) which is less than $O(n \cdot m \ln m)$

5.2 Spatial Model

Our model has a parameter z which is a vector representing the standard deviation in each dimension in the space of issues where $z_j \geq z_{j+1}$

We have a parameter d which represents the standard deviation of population means of districts as a portion of the standard deviation of voters.

We define a dimensional space $|z|$ called the space of issues. Each voter and each party are represented by a point in this space. Voters prefer parties with a lower distance between a voter's point and a given party's point. In each party i we select $p_{i,j} = N(0, z_j)$ where p_i represents the point in the space of issues of party i. We also consider the case in which two of the parties represent the traditional major left and right parties[8]. Population means for voters in a district k is selected $\mu_{k,j} = N(0, z_j \cdot d)$ We assume districts are independent. In each district k, a voter n is selected $v_{n,j} = N(\mu_{k,j}, z_j)$

Parameter selection

An obvious class of parameter values to select are all non-zero variances set to be equal, with 1, 2 or 3 non-zero dimensions. In particular two dimensions is interesting as the dimensions can refer to economic liberty and social liberty.

We fit parameters to this model, obtained from real world data, via the method discussed in Section 7.

5.3 Preference Swapping Model

The Preference Swapping Model extends upon the model used in the 2011 referendum simulator (Pritchard and Wilson, 2011). It takes a set of real world elections as a parameter and creates a cluster of similar elections around each real world election. The model works by redistributing real election results, evolving an input election by inferring the preference orders for each vote and switching preferences by some transformation function.

The transformation function for preference orders works by switching the order of the preferences with some probability. The probability of switching between 1st place and 2nd place may be different from switching between 1st place and 3rd place, for example. As we are only concerned with 1st place preferences, we do not consider the case where another pair of preferences also switches. We also use the case where we apply the output of our model as the input to another stage in our model (e.g. so that we model 2 or more steps into the future). Each stage is calculated in the same way; with the exception that we infer preference orders only from the first preferences in the first step.

[8] In this case both the left and right parties have points almost at the origin. e.g. $(+0.0001, 0)$ and $(-0.0001, 0)$.

We define s_i to be the probability that a voter switches the 1st and ith preference, where s_i is selected uniformly at random between 0 and $\max(s_i)$ where $\max(s_i)$ is a parameter to our model.

It doesn't make much sense for $s_i < s_{i+1}$ however, even with $max(s_i) > max(s_{i+1})$ (as we would expect), our model can generate values of s_i and s_{i+1} such that $s_i < s_{i+1}$. We consider both the case where we apply no additional restrictions on the model and the case where $s_i \geq s_{i+1}$

5.3.1 Redistricting

It was not required to generate D_{new} [9] new districts that each were strictly a combination of the original districts and that each original district was used once. It was only required that after redistricting we would get similar results than if we had done this.

Redistricting is done by the following algorithm:

- Calculate the principal components of districts [10] and the distribution of the deviations for each principal component[11].
- Generate modified distribution for each principal component by selecting a new standard deviation for each $SD_{new} = SD_{old} \cdot \sqrt{\frac{D_{new}}{D_{old}}}$ [12].
- In the case of each new district, we sample from the modified distributions of principal component deviations and back-transform these deviations into district information. In the case that the generated information is invalid [13] we try this step again, otherwise we keep the information and continue.

The resulting districts will have total vote proportions by party that is approximately equal to the total votes proportions by party of the original data.

Parameter selection

We use the 2011 election in New Zealand as a start point and infer preference orders and the probability of switching preference orders from the NZES 2008 (Vowles et al., 2008).

We infer preference orders from the survey by assuming that the first preference of a voter is the party that the voter actually voted for and that the remaining parties are ordered in descending order based on the scores that voters were asked to rate parties by.

We found that 18% of voters switched between their first and second preference, and 12% switched between their first and third preferences. We do not consider switching between preferences lower than 3rd as our preference data is not reliable enough. We select $\max s_2 = 0.18$ and $\max s_3 = 0.12$.

In the case where voters gave the same score to more than one party, we consider all possible orders of parties with the same score with the weight for each score reduced accordingly.

[9] Recall that D represents the number of districts.

[10] We calculate the principal components of a dataset with rows representing districts and columns representing parties. A cell represents the proportion of votes in a given district that were for a given party.

[11] Assuming the deviations are normally distributed.

[12] This reduces or increases the variance of the deviations in accordance with the Central Limit Theorem. With a smaller number of districts, one would expect each district to have a closer distribution of votes to the mean than the case with a larger number of districts.

[13] We consider a generated district to be invalid if any party receives a negative number of votes.

We normalise the weights for all preference orders where voters voted for a given party to 1, then treat the normalised weights as a probability that a voter for a given party has a particular preference order.

We assume that given a voter voted for a particular party, the probability it has a particular preference order is independent of its district and from election to election.

We select s_i to be the proportion of voters whose vote in 2005 (the previous election) was equal to their ith current preference.

6 Apportionment Methods

Apportionment methods ensure that when rounding to integers, in this case vote shares to seat shares, we can minimise error (for some definition of error), while rounding to a constant total.

We let p_i be the proportion of votes that party i receives and we define apportionment functions that transform p to s where s_i is the seat share of party i.

The true number of seats a party i deserves is $t_i = |s| \cdot p_i$, however t_i may not be an integer.

In the case of each method we define $l_i = floor(t_i)$ and $s_i = l_i + k_i$ where k_i is either 0 (we round down) or 1 (we round up).

It follows that $\sum s = \sum l + \sum k$ and thus $\sum k = \sum s - \sum l$

The methods differ on how k is defined.

To resolve the case of ties, we add a small vector of additional random votes totalling less than 1 real vote.

6.1 St Lague Method

We do not calculate k directly. In this method we iteratively allocate $|s|$ seats.

In each iteration we allocate the next seat to the party with the highest quotient where $quot_i = \frac{v_i}{2s_i+1}$

6.2 D'Hondt/Jefferson Method

The D'Hondt and Jefferson methods are mathematically equivalent in that they both return the same results, though they are calculated differently.

We generate an $m \cdot |s|$ matrix [14]. Values in the cells are set such that $(i, j) = \frac{v_i}{j}$

We then select a cut-off threshold such that there are only $|s|$ values in the matrix that are above the cut-off.

Each party i is allocated as many seats as there are values of j such that the value in cell (i, j) is greater than the threshold.

6.3 Hill Method

We iteratively allocate each seat. In each iteration we allocate the next seat to the party with the highest quota, where $quot_i = \frac{v_i}{\sqrt{floor(s_i)^2+(floor(s_i)+1)^2}}$

[14] Recall that m is the number of parties.

6.4 Hamilton Method

We already know $\sum k$, we allocate $k_i = 1$ to the values of i in descending order of $t_i - l_i$ for $\sum k$ values of i.

7 Fitting real world data to the Spatial Model

It is of interest to calculate parameters for the Spatial Model (see Section 5.2) that are reasonably like the real world. To do so we attempt to fit real world data to the spatial model.

We assume that there is a relationship between the parameter z and the proportion of variance explained by each principal component of the results per party by district.

We attempt to validate this hypothesis by simulating a large number of elections under our model, several for each of many different values of z and compare it to the values obtained when applying the formula to the variance from principal components of the election results. See Figure 2.

We fitted a regression line and found extremely strong evidence $p - value < 2 \times 10^{-}16$ against H_0 that there was no relationship between the two variables. The fit had high error $R^2 = 0.49$ but this error can be mitigated by taking a large number of samples from artificial societies for a given z value and comparing distances to a reasonably large set of real world elections.

We select z as to minimise the error of the model in describing a set of real world elections. We recursively select smaller and smaller ranges that we predict contain the target value of z to minimise the error as follows:

We define $y = \sum_i {z_i}^2$

- 1. Generate several values uniformly at random within the range of y values that we believe the target y to be in.
- 2. Find a value of z with y close to our target and update the values of y with the actual values from our values of z
- 3. For every value of z we simulate a number of elections and compute the mean error.
- 4. Select all the results (by y value) that have the minimum mean absolute error, and discard the rest.
- 5. Calculate the underlying distribution of y values that we retained in step 4. (assuming a Normal distribution)
- 6. Calculate new upper and lower bounds for the target y value. If these bounds are within our target error margin we select mean, otherwise we loop with these new bounds.
- 7. Select a z value from a number of generated candidates with the target y that minimises the error.

We found that this method can, with reasonable accuracy, calculate z values from election outcomes generated from the spatial model, however it had higher error from real world election outcomes and the results varied each time we ran the algorithm. We found that the values of z the algorithm found from real world data was not unlike the parameters we were using in the simulation.

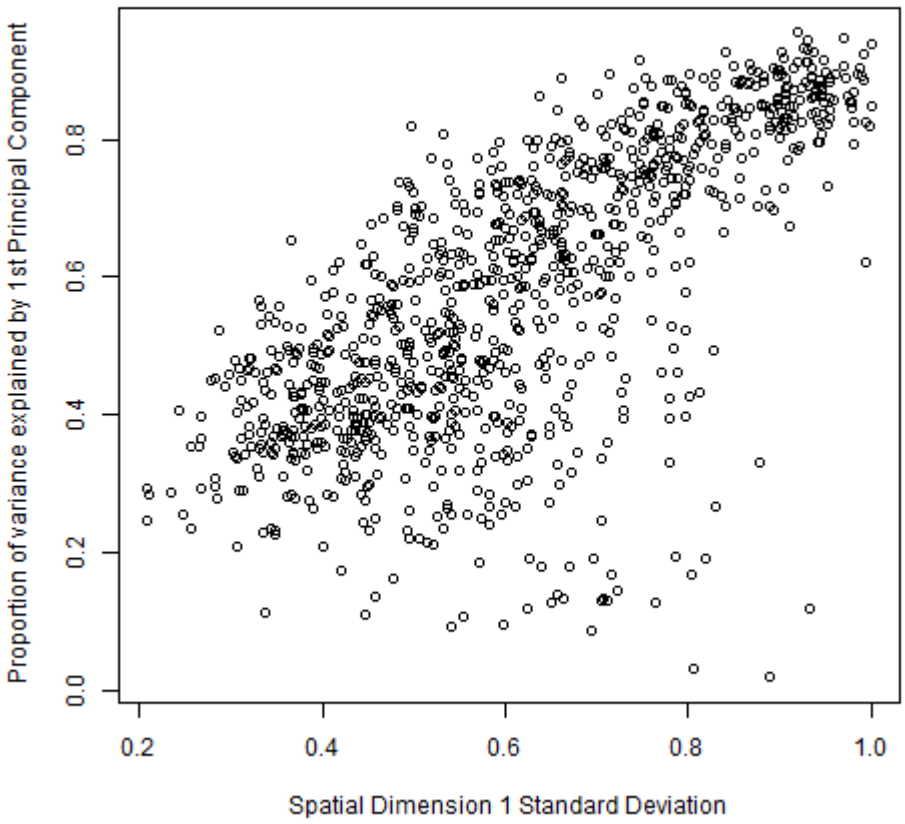

Figure 2: Values of z_0 vs proportion of variance explained by the first principal component

8 District Magnitude

District Magnitude represents the number of seats in a particular district. (Carey and Hix, 2011) uses the median district magnitude to group collections of voting rules and their parameters for analysis. In some systems (e.g. SM, see Section 9.2.2) the district magnitude varies from district to district. Using the median would group SM in the same category as FPP which is undesirable. We resolve this problem by analysing voting systems independently instead of grouping the data together. We use the median in all cases except the case of SM, in which case we use the district magnitude of the largest district.

9 Voting Rules

9.1 Single Transferable Vote

In this system we take into account the complete preference orders of voters. We define a number of equal sized districts and define a quota, such that $quota = (\frac{n}{DM+1}) + 1$. (where n is the number of voters in a given district)

All parties need to obtain *quota* votes to obtain a seat. The process of allocating votes to parties, in each district, works as follows:

- 1. Assign a weight w of 1.0 to all votes. The system can be optimised by combining identical votes and using the sum of their weight.
- 2. Calculate the sum of weights $\omega = \sum_i w_i; p_{i,1} = j$, by party j, for all votes. Select the party to assign the weight by the highest non-excluded and party in a voter's preference order[15].
- 3. Each party that obtains a weight greater than or equal to the quota is elected. We reduce the weights of all votes that have the highest non-excluded preference equal to a given elected party by multiplying each by $m = \frac{\omega - quota}{\omega}$, thus there is *quota* weight removed.
- 4. If no party was elected in Step 3, we exclude the party that obtained the least weight in Step 2.
- 5. We loop back to Step 2. until enough parties are elected.

9.2 Proportional Systems

We use the following two voting systems which are special cases of a more general system which is proportional in each district. At one extreme of both of these systems is FPP (First Past the Post), and the other extreme, pure proportional. In each district under these systems, we apply some apportionment method (see Section 6) to select the number of seats by party, and aggregate the results. These systems take into account only first preference votes.

9.2.1 District Proportional

In this special case, we define multiple districts of equal size. Each voter votes under only one district.

[15] $p_{i,k}$ denotes the kth non excluded preference of voter i.

9.2.2 Supplementary Member

In this special case, we define a number of districts of size 1 and a single large district of some larger size. Voters are assumed to each vote in one of the smaller districts and to all also vote in the larger district. We assume voters vote the same way for both the small district and the large district.

10 Efficiently Sampling from a Discrete Probability Distribution

We define w_i to be the weight of element i in the probability distribution, where $Pr(X = i) = \frac{w_i}{\sum w}$ We take advantage of the fact that most elements in the probability distribution have a weight of 1. However this structure does not actually store w directly.

The structure is similar to a binary search tree, however instead of containing elements it contains *ranges* where leaf nodes (i.e. nodes with a range length of 1) represent the elements of w. Every element in the tree contains a precomputed parameter representing the weight of the element, where we define the weight of a non-leaf node (that is, a node that doesn't represent a single element) to be the sum of the weights of its left and right child. Much of the tree is not ever written to, or read from, and so we can lazily load nodes on demand. This lazy loading saves us a significant amount of space for a large $|w|$.

When sampling from the distribution, we select a value y uniformly at random $y = U(0, \sum w)$ and call a recursive search function on the root node, as defined by the below pseudo-code:

```
Node RecursiveSearch(Node n, double y)
{
        if (n.length == 1)
                return n;

        double skipAmount = 0.0;

        if (getLeftChild(n) != null)
        {
                if (getLeftChild(n).weight>=y)
                {
                        return RecursiveSearch(getLeftChild(), y);
                }
                else
                {
                        skipAmount = getLeftChild(n).weight;
                }
        }

        return RecursiveSearch(getRightChild(n), y - skipAmount);
}
```

Once the node in question is found, we return the sampled value from the distribution and update the weight of the node by the requested amount. When increasing the weight of an element, we need to increase the weight of the parent node [16] which takes, in the worst case, $\ln |w|$ time.

11 Undesirability

We define an undesirability function to be a function of both unfairness and instability which returns a real number which is optimal when minimised. Different functions have different levels of sensitivity to each of these properties. For the purposes of this analysis we consider only linear functions so as to minimise the number of arbitrary parameters we use.

It seems unreasonable to define a linear undesirability function that has a coefficient for either unfairness or instability that is less than zero [17], thus only functions with coefficients equal to or greater than zero are considered. We transform unfairness and instability by first dividing each by its corresponding standard deviation across our entire dataset [18]. We do not need to bother about constant errors [19]. We construct an undesirability function by first creating a line L_1 from the optimal point [20] to a point along the line segment L_2 $(1, 0)$ to $(0, 1)$. We consider L_2 to be an axis that we measure the plotted transformed point of unfairness and instability against.

We subdivide L_2 into a large number of equal length parts and consider the undesirability functions that are defined when L_1 intersects L_2 at each different boundary of these different subsections of L_2.

As we are interested only in minimising undesirability across multiple values of District Magnitude, we compare only the undesirability of average points for each District Magnitude.

In reality, only undesirability functions that intersect close to the midpoint of L_2 are worth considering. Extreme values correspond to only caring about one property.

12 Results

12.1 Simulation Results

The outcomes from a series of simulation runs are plotted below. For each run, four plots are provided. The bottom left corner of each plot is optimal. Points in the top left plot represent individual simulations in a given run. Points in the top right plot represent means predicted, in accordance with our regression model, for a given District Magnitude. The bottom left plot may be slightly misleading as points plotted here are not necessarily near results from actual simulations. Each point represents a single District Magnitude and the position along each axis refers to the 95th percentile value for simulations using this District Magnitude. The purpose of this plot is to

[16] By updating the parent's weight, the weight of the parent's parent must also be updated, etc.

[17] In the case of a linear undesirability function with coefficients less than zero, we would consider a less unfair or less unstable result to be more undesirable, that is, a more fair or more stable result to be undesirable.

[18] This is in an attempt to neutralise any bias in our choice of random undesirability functions.

[19] Subtracting a constant from either as we only ever consider minimising undesirability across the same undesirability function and a constant added to the function would clearly be constant across all values from that function. Likewise, multiplying by a positive constant would have the same effect across all values of the function and thus we only need to consider one undesirability function of any set of undesirability functions that can be transformed by a linear function from one to the other.

[20] The origin $(0, 0)$ after transforming the properties as described above.

demonstrate the worst expected values for each of the two properties plotted. Points in the bottom right plot represent means, from actual results, per District Magnitude.

There is a cluster of results in the bottom centre-right of Figure 10. This represents the case when a single party has all the Shapley Shubik Power.

Supplementary Member (See Figure 4) has a much different curve than for the other rules [21].

The results in Figure 14 are generated using the Preference Swapping Model (Section 5.3) using spatial information obtained from (NZEC, 2011) and preference information obtained from (Vowles et al., 2008).

Unless otherwise stated, results are calculated with 120 districts and 8 parties and results calculated with the spatial model include major left and right parties.

[21] Recall that District Magnitude for Supplementary Member is defined differently. By the definition we use for the other systems, SM would have $DM = 1$ in all cases

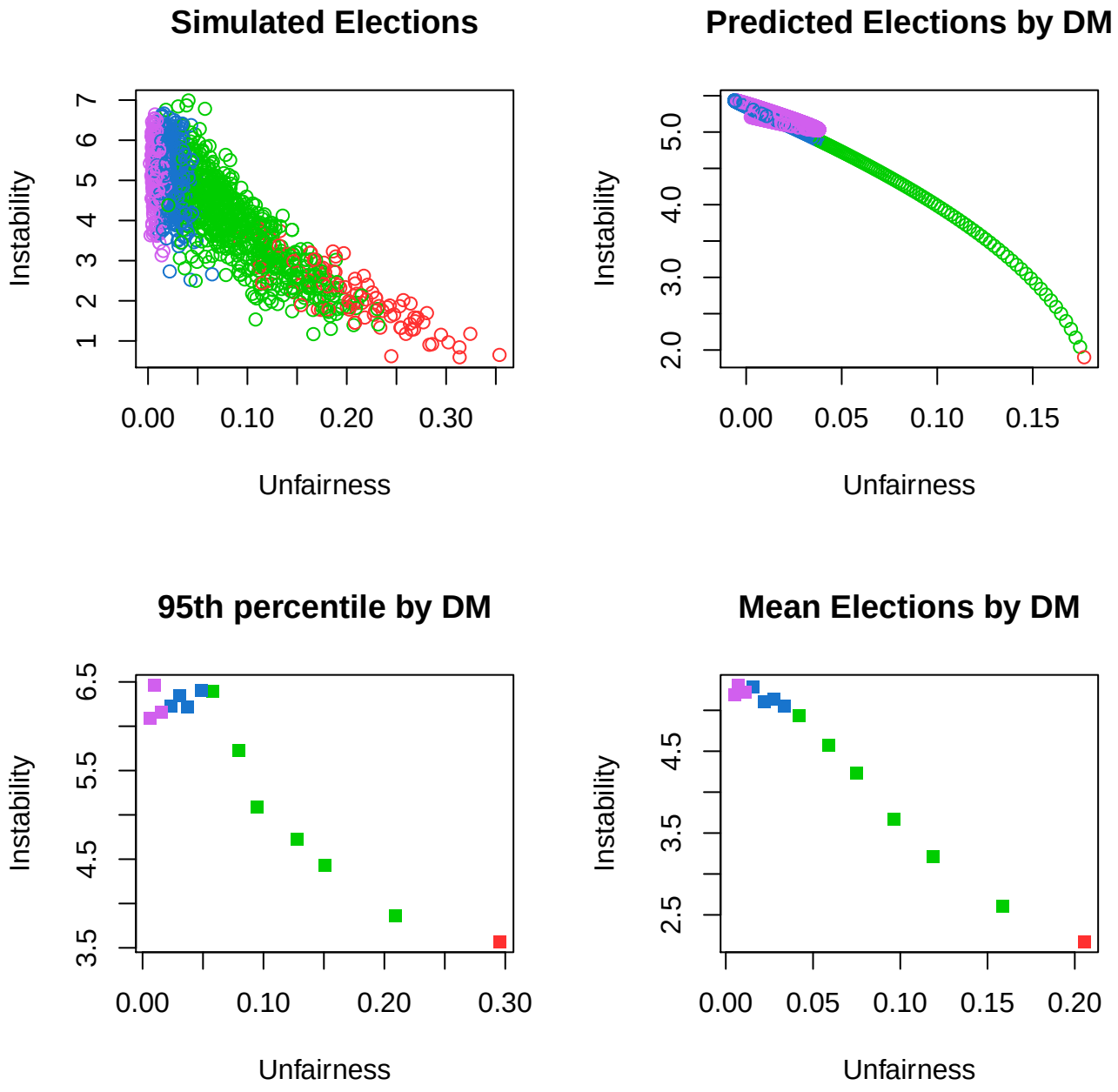

Figure 3: Results from simulated elections with the *Spatial Model* with 2 *dimensions* and the *District Proportional* rule. Plotted using the *Gallagher* Index vs. *ENP*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

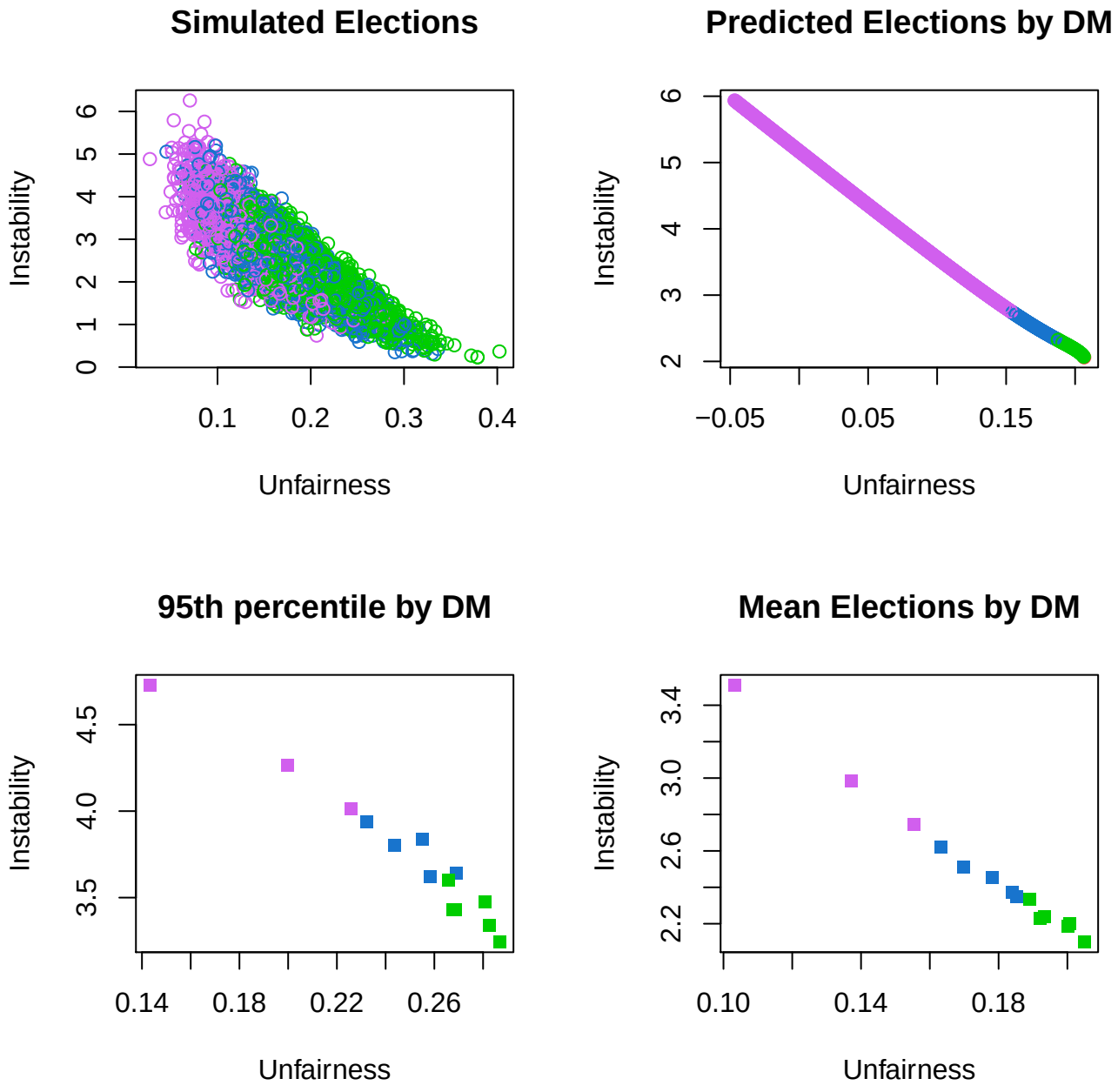

Figure 4: Results from simulated elections with the *Spatial Model* with 2 *dimensions* and the *Supplementary Member* rule. Plotted using the *Gallagher* Index vs. *ENP*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

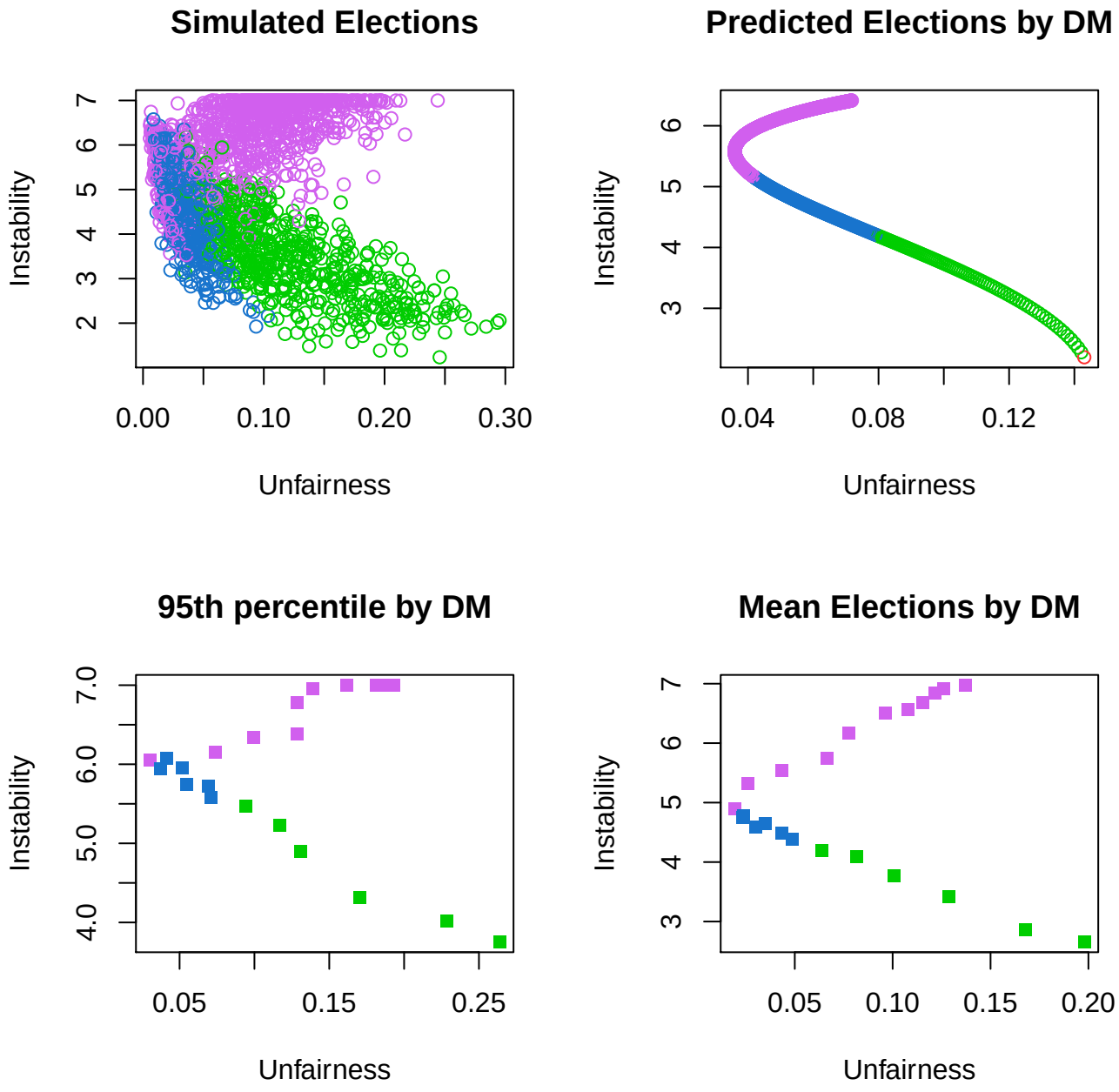

Figure 5: Results from simulated elections with the *Spatial Model* with 2 *dimensions* and the *STV* rule. Plotted using the *Gallagher* Index vs. *ENP*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

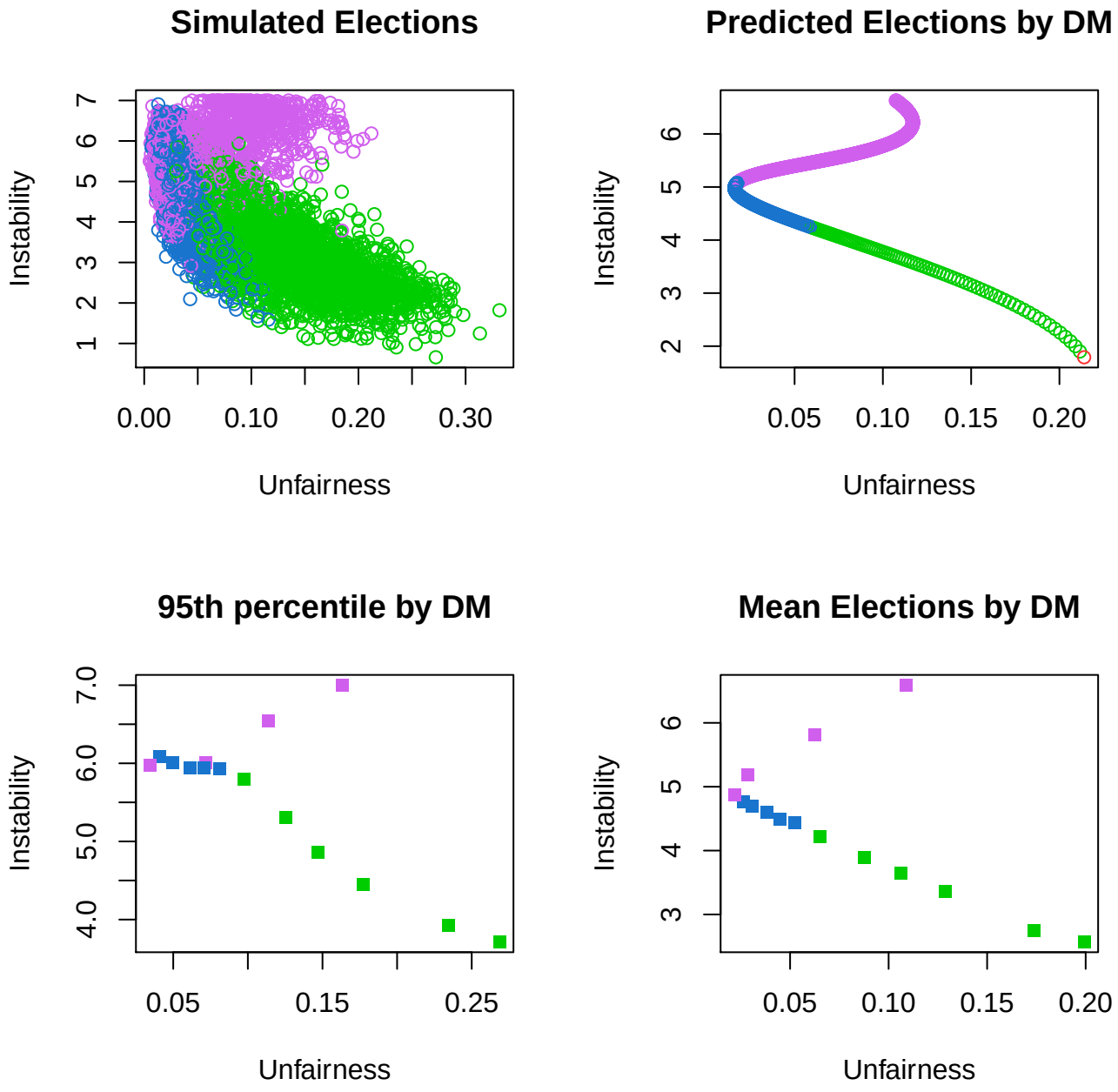

Figure 6: Results from simulated elections with the *Spatial Model* with 2 *dimensions* and the *STV* rule with *600 districts*. Plotted using the *Gallagher* Index vs. *ENP*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

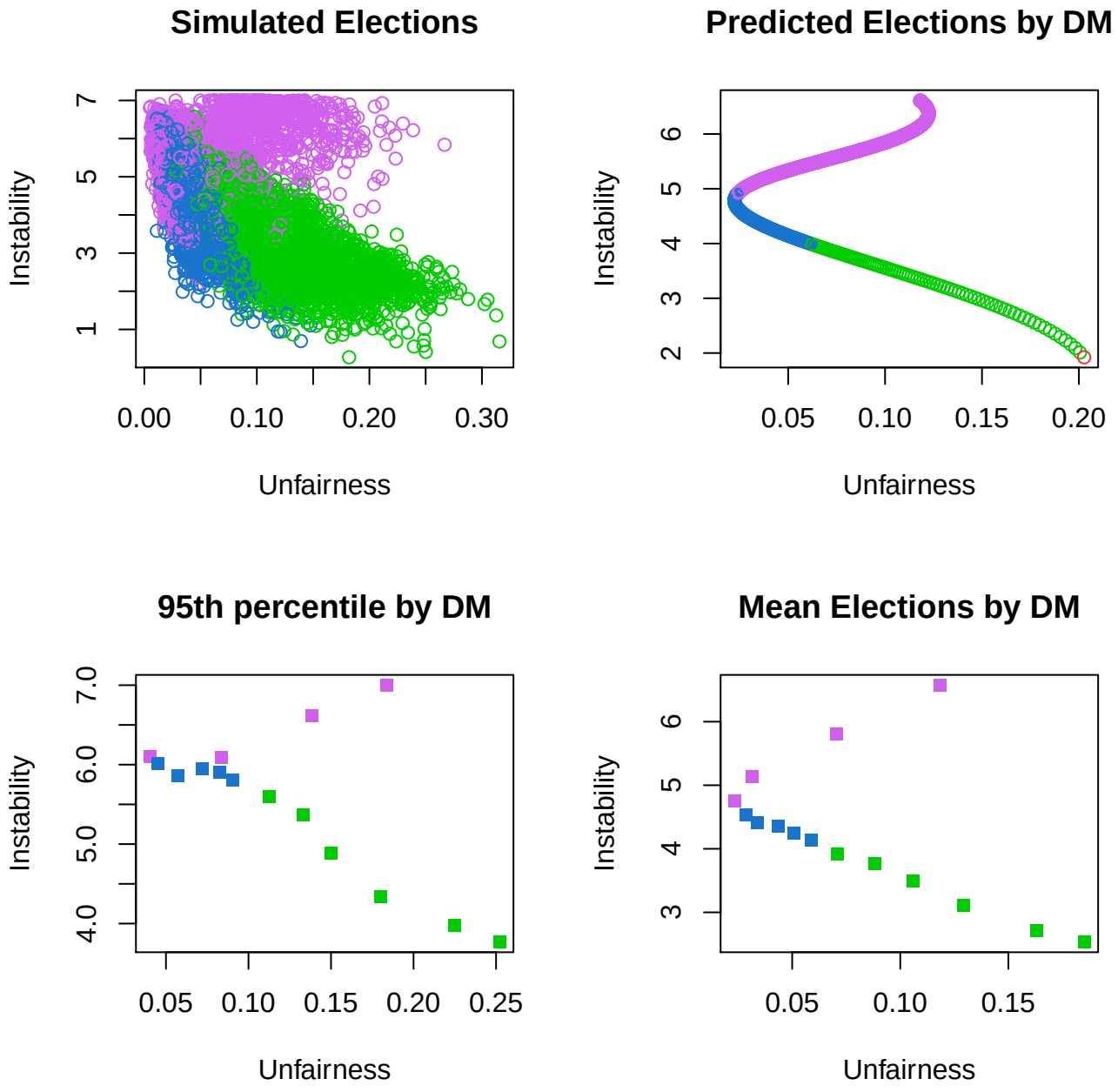

Figure 7: Results from simulated elections with the *Spatial Model* with 2 *dimensions* and the *STV* rule. Does not include major left and right parties. Plotted using the *Gallagher* Index vs. *ENP*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

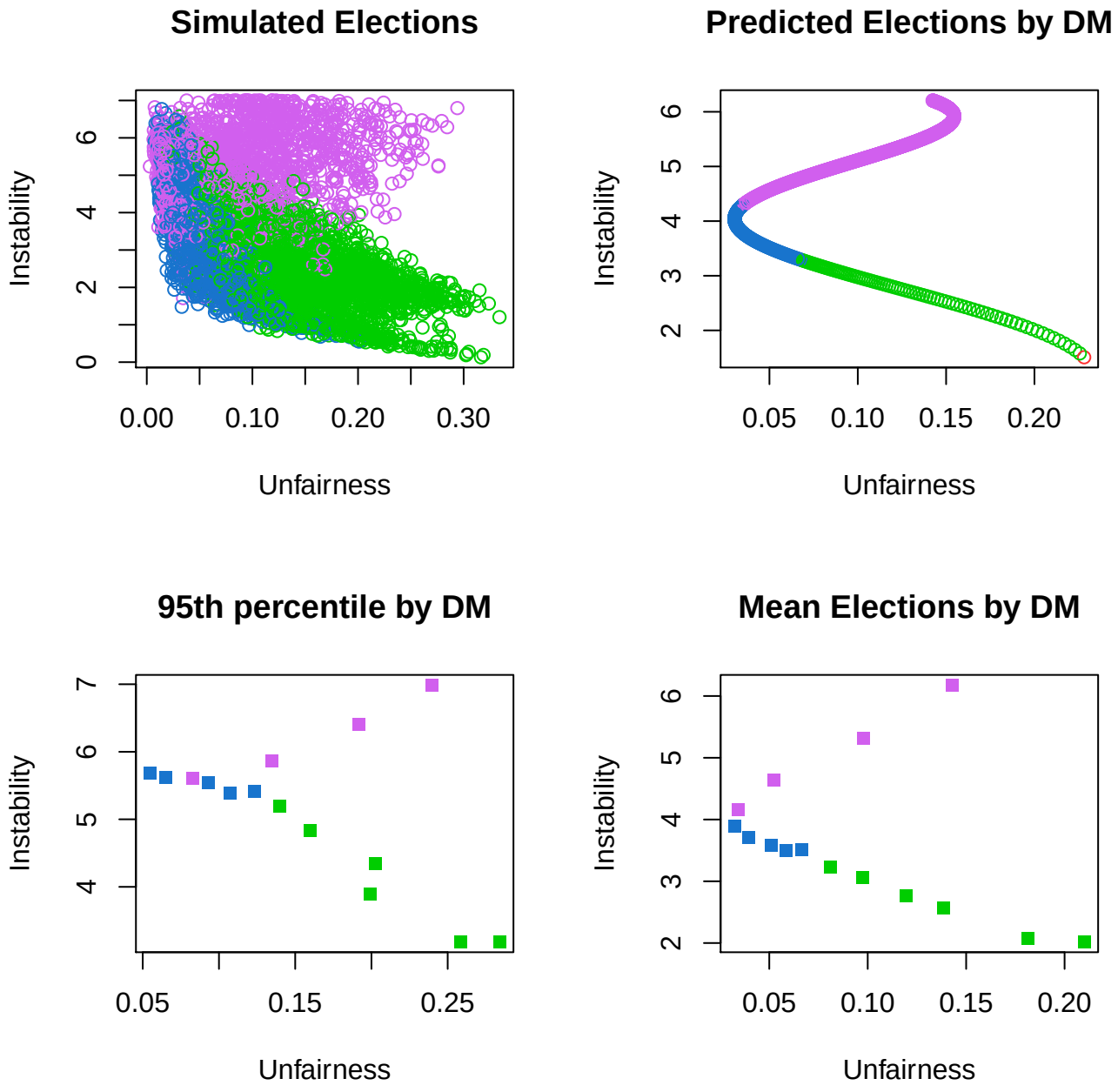

Figure 8: Results from simulated elections with the *Spatial Model* with 1 *dimension* and the *STV* rule. Plotted using the *Gallagher* Index vs. *ENP*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

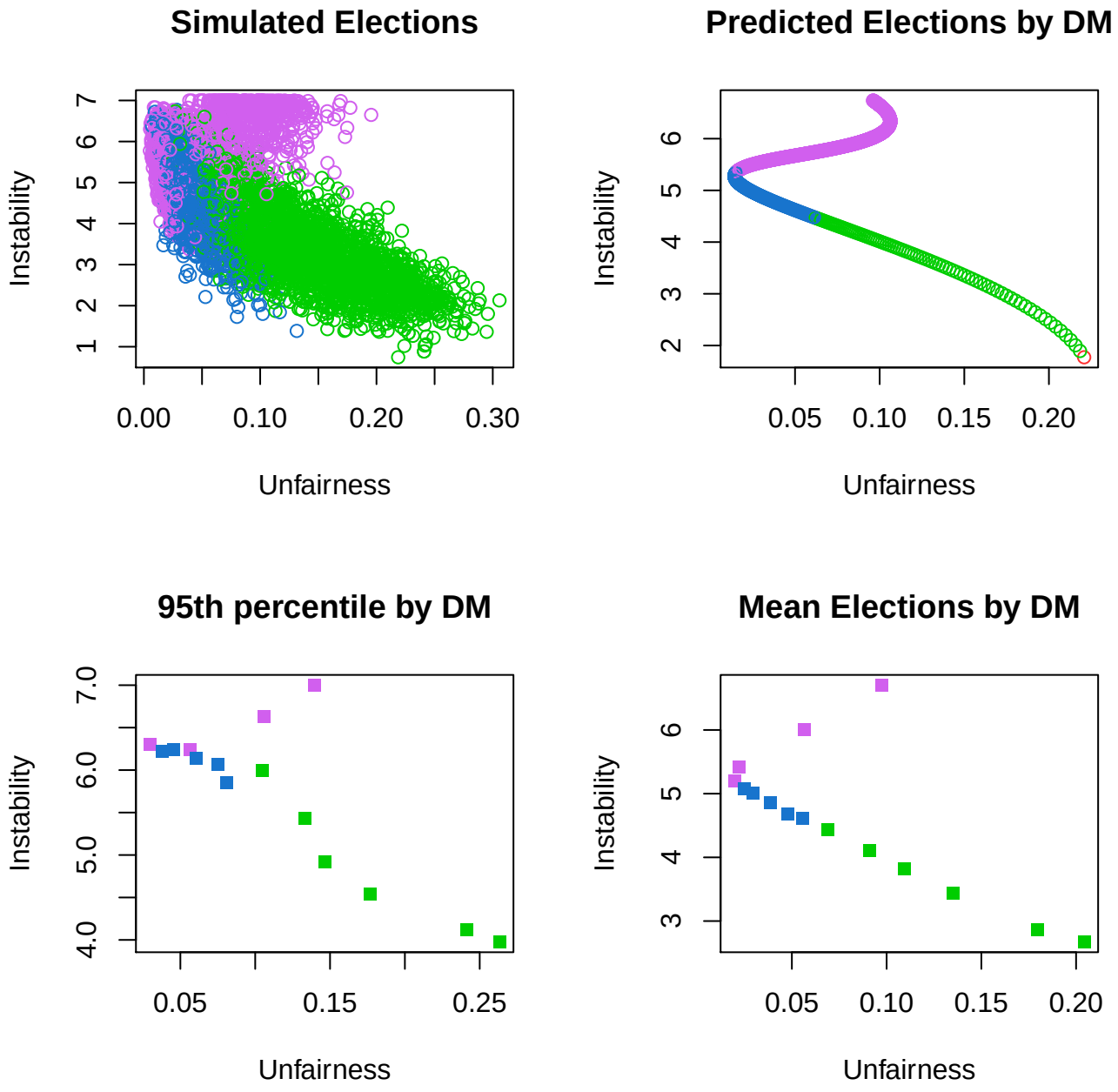

Figure 9: Results from simulated elections with the *Spatial Model* with 3 *dimensions* and the *STV* rule. Plotted using the *Gallagher* Index vs. *ENP*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

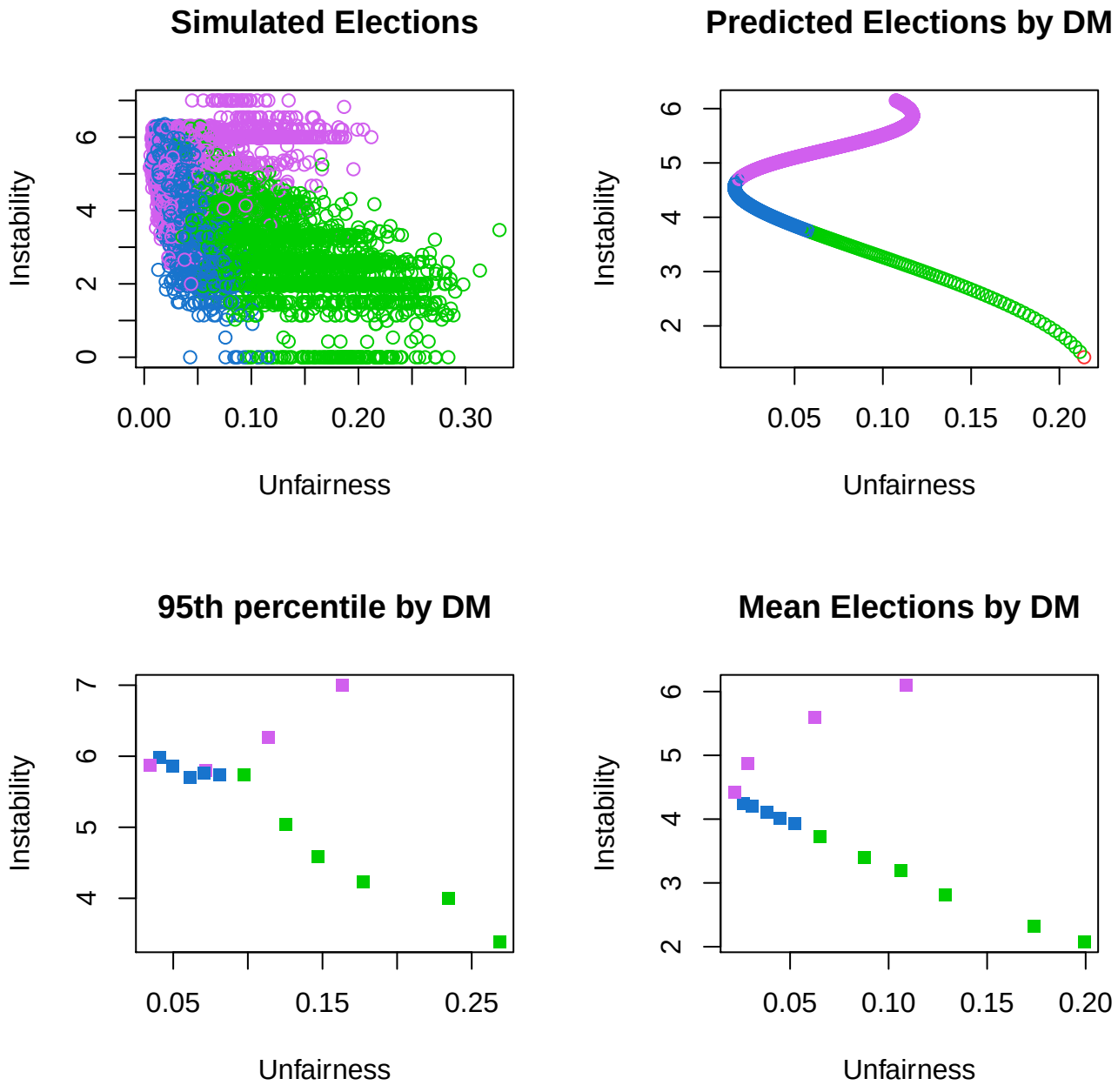

Figure 10: Results from simulated elections with the *Spatial Model* with 2 *dimensions* and the *STV* rule. Plotted using the *Gallagher* Index vs. *ENP using SSPI*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

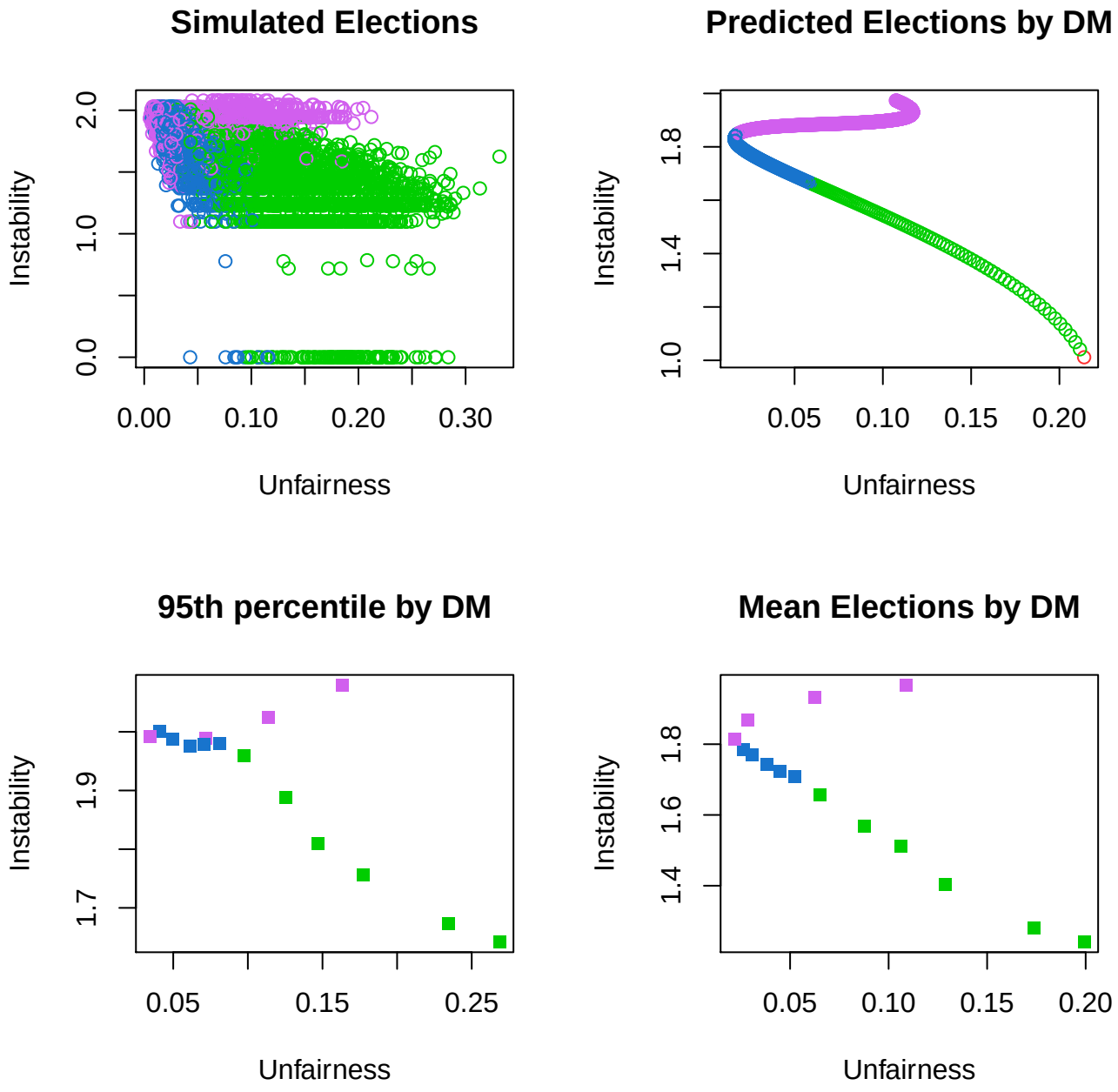

Figure 11: Results from simulated elections with the *Spatial Model* with 2 *dimensions* and the *STV* rule. Plotted using the *Gallagher* Index vs. *Entropy of SSPI*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

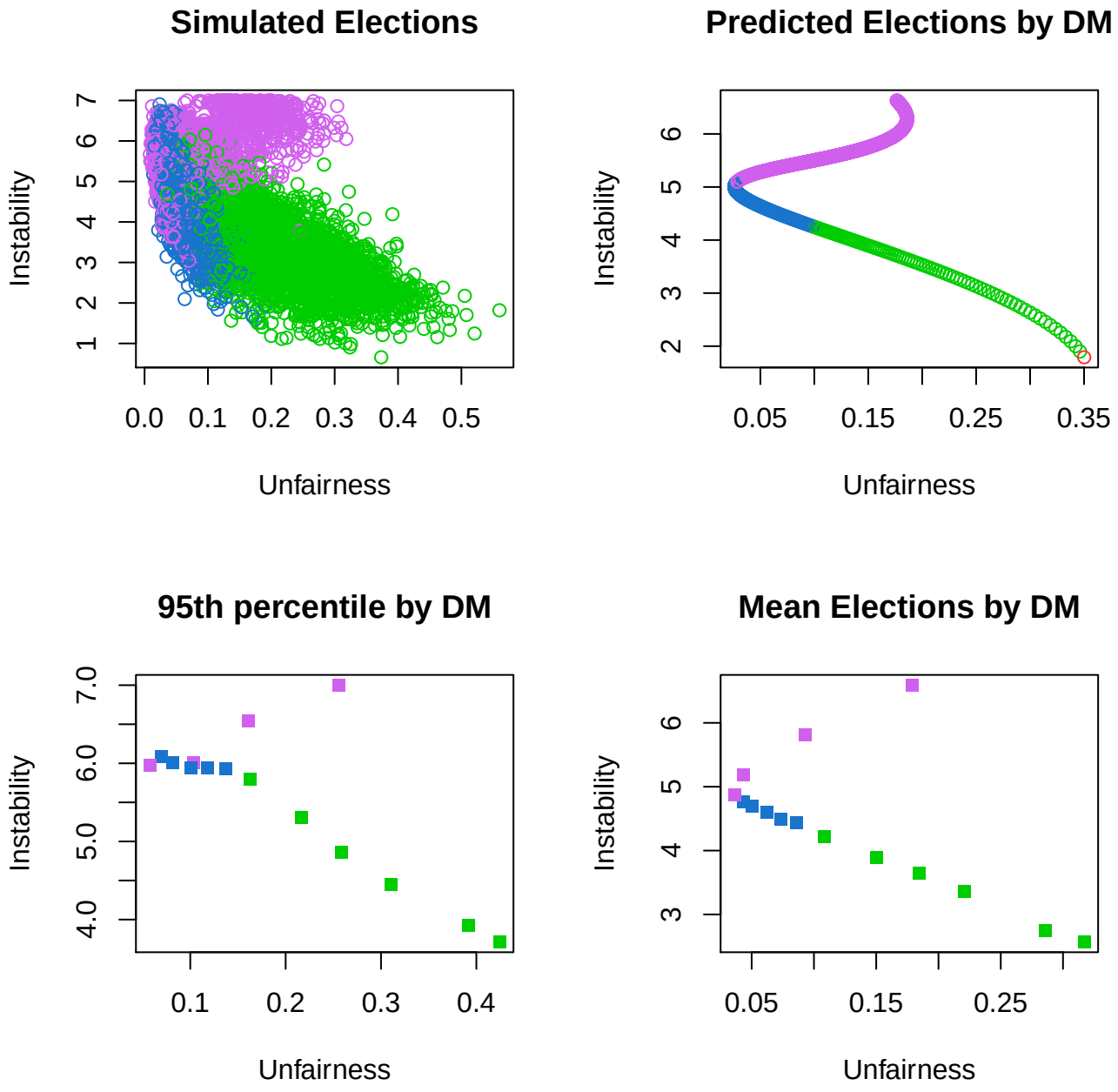

Figure 12: Results from simulated elections with the *Spatial Model* with 2 *dimensions* and the *STV* rule. Plotted using the *Loosemorehanby* Index vs. *ENP*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

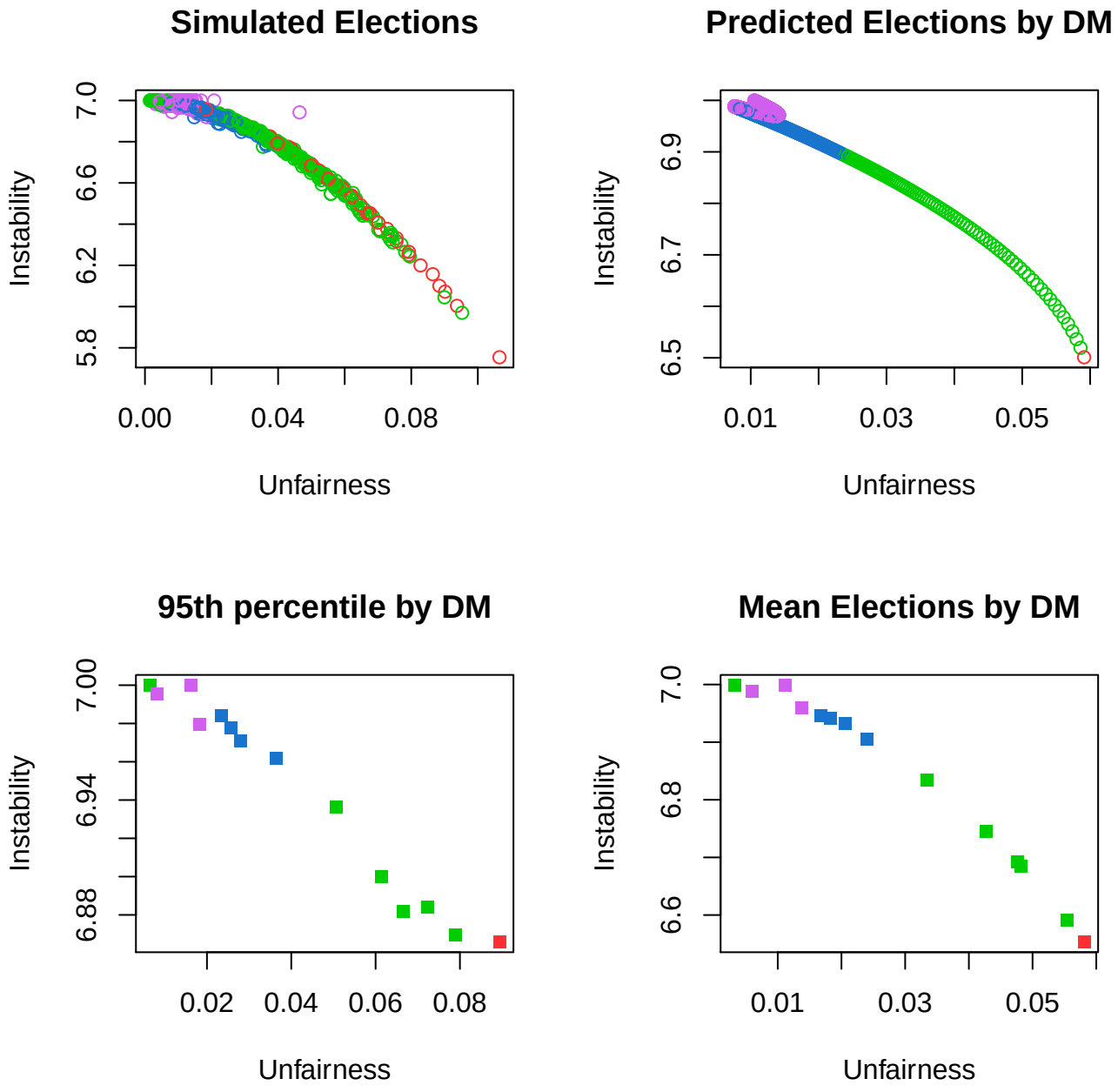

Figure 13: Results from simulated elections with the *Polya Eggenberger Model* and the *STV* rule. Plotted using the *Gallagher* Index vs. *ENP*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

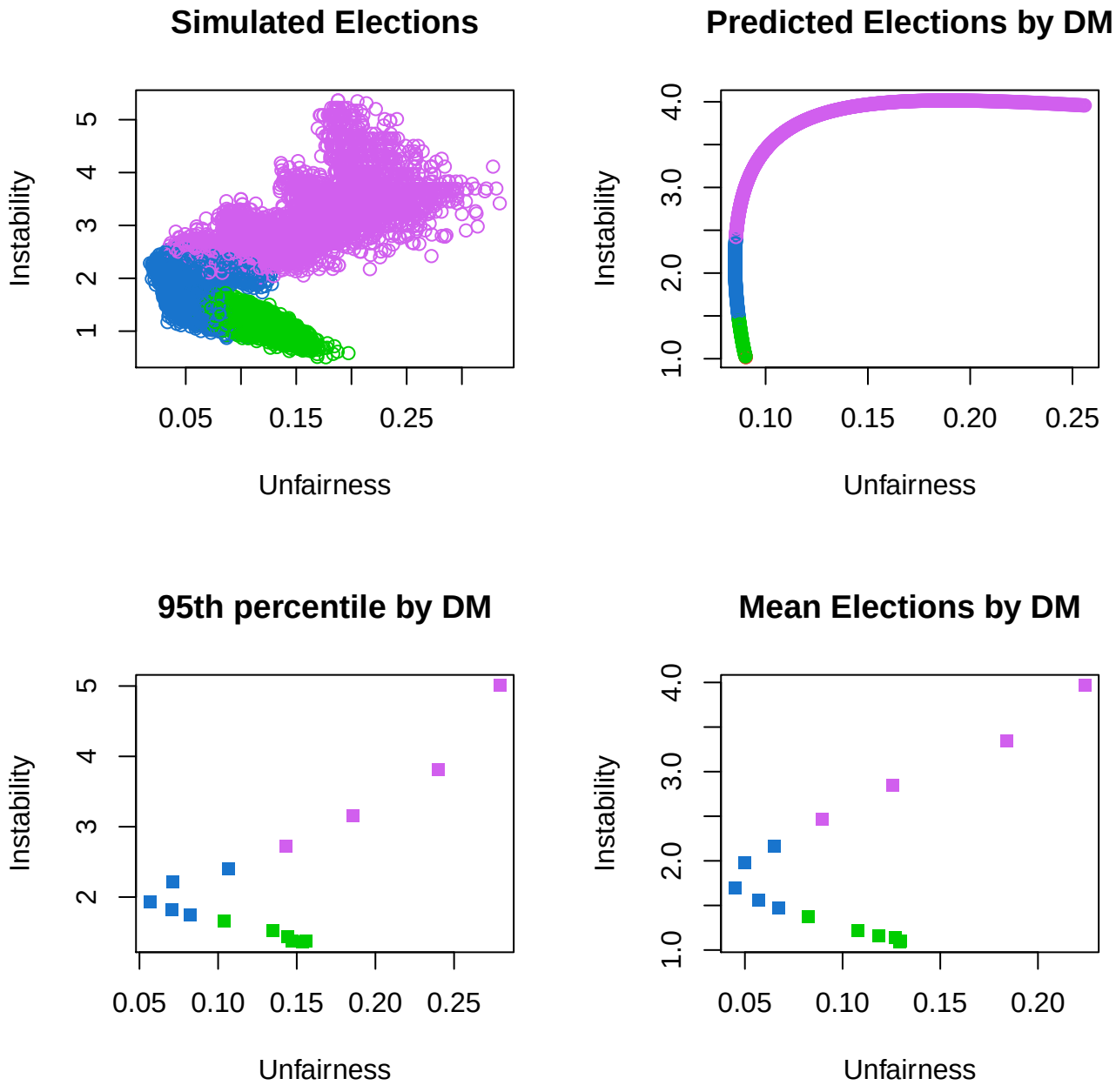

Figure 14: Results from simulated elections with the *Preference Swapping Model* using information from New Zealand and the *STV* rule. Plotted using the *Gallagher* Index vs. *ENP*. DM=1 (Red) DM=2-9 (Green) DM=10-29 (Blue) DM>30 (Purple)

12.2 Minimum Undesirability

The following plots represent the probability of a given District Magnitude having the minimum undesirability (as defined in Section 11). This assumes a uniform distribution of all possible and reasonable linear undesirability functions on one measure of each unfairness and instability.

We consider all reasonable linear undesirability functions and then disregard results for the most extreme District Magnitudes, so that we do not have to make an arbitrary decision about the cut-off for extreme undesirability functions.

12.3 Analysis of Different Apportionment Methods

We ran multiple simulations with the Spatial Model and the District Proportional rule, only differing by the apportionment method used in each case. We used the same District Magnitudes used in other simulation runs.

We analysed the results by creating regression fit using dummy variables for the apportionment method factors.

In the case of all our simulations, the assumption of independence between samples is satisfied by experiment design. The assumption of equal variance appears to be satisfied.

We found extremely strong evidence of a difference in fairness [22] between apportionment methods [23]. We found extremely strong evidence of a difference in stability [24] between apportionment methods [25].

We found extremely strong evidence [26] of a difference in fairness between St Lague and Hill methods. We estimate that on average fairness scores for the Hill method are between 0.35 and 1.1 percentage points higher.

We found extremely strong evidence [27] of a difference in fairness between St Lague and Jefferson methods. We estimate that on average fairness scores for the Jefferson method are between 1.6 and 2.4 percentage points higher.

We found extremely strong evidence [28] of a difference in stability between St Lague and Hill methods. We estimate that on average the ENP under the Hill method are between 0.33 and 0.17 lower.

We found extremely strong evidence [29] of a difference in stability between St Lague and Jefferson methods. We estimate that on average the ENP under the Jefferson method are between 0.62 and 0.46 lower.

[22] As measured by the Gallagher Index
[23] p-value $= 2.2e-16$
[24] As measured by the Effective Number of Parties
[25] p-value $= 2.2e-16$
[26] p-value=0.000102
[27] p-value=2e-16
[28] p-value=5.51e-10
[29] p-value=2e-16

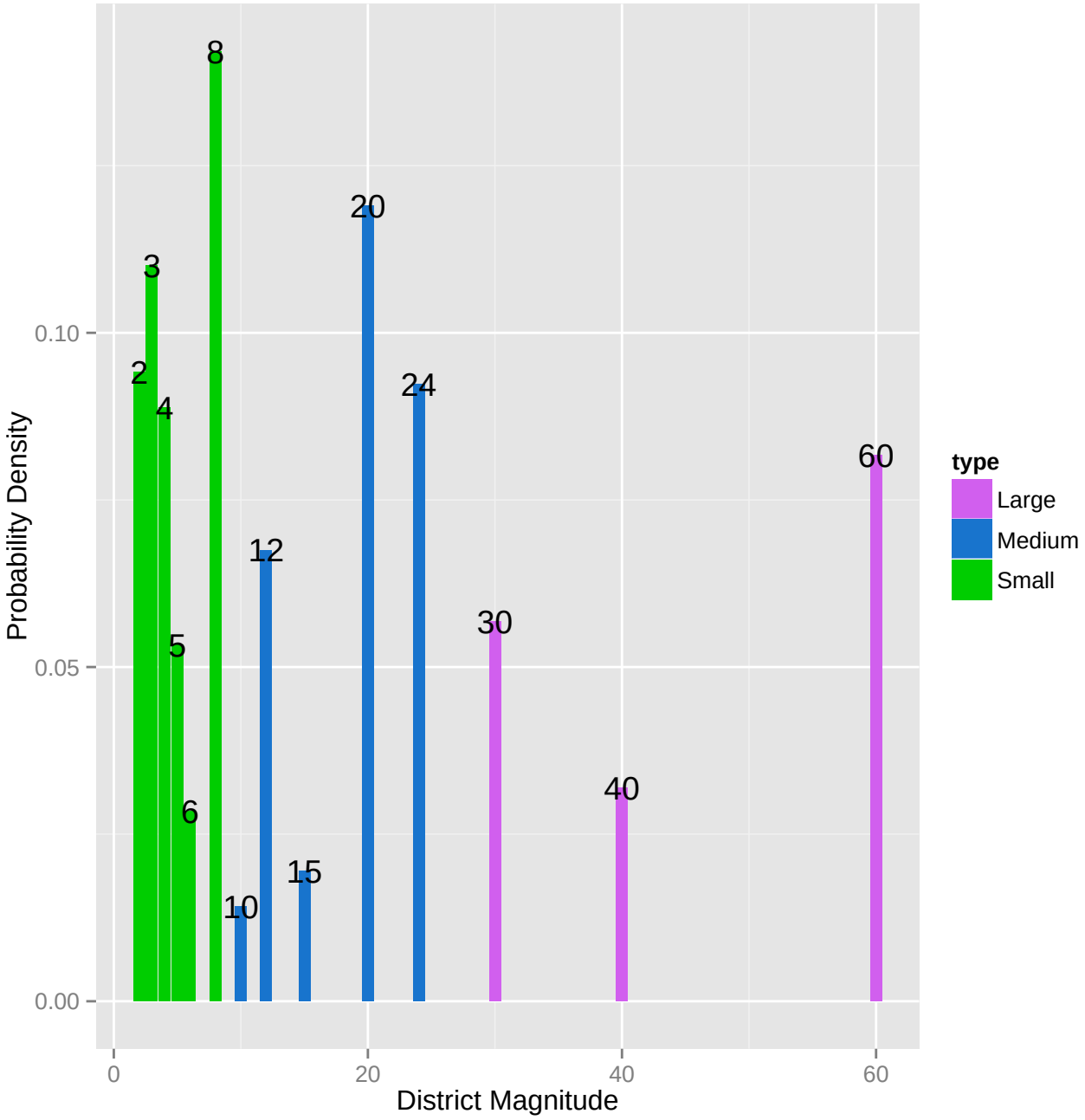

Figure 15: Probability Distribution of least undesirable District Magnitude (including extreme values). *Gallagher* and *ENP*

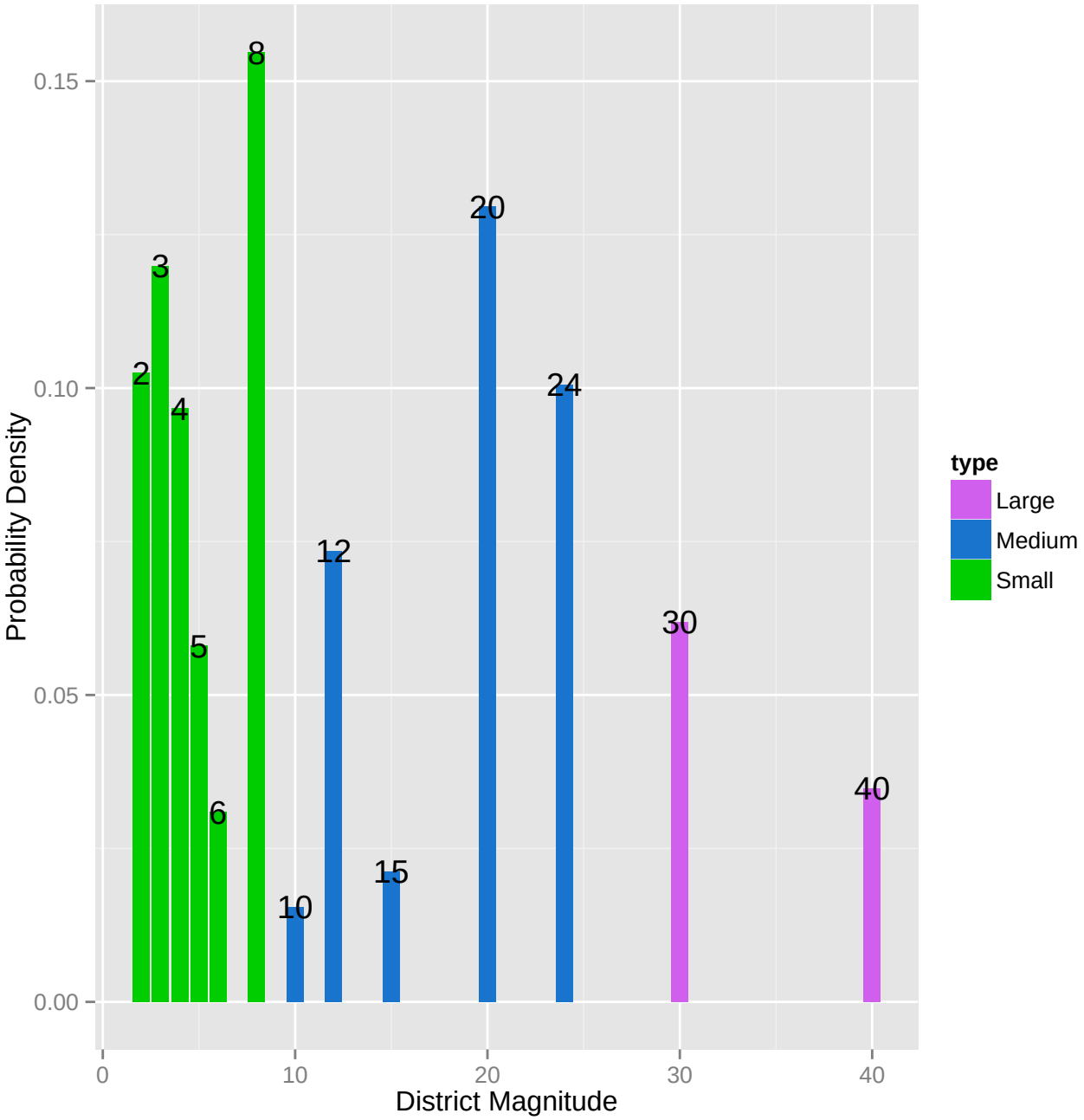

Figure 16: Probability Distribution of least undesirable District Magnitude. Same results as 15 excluding extreme values.

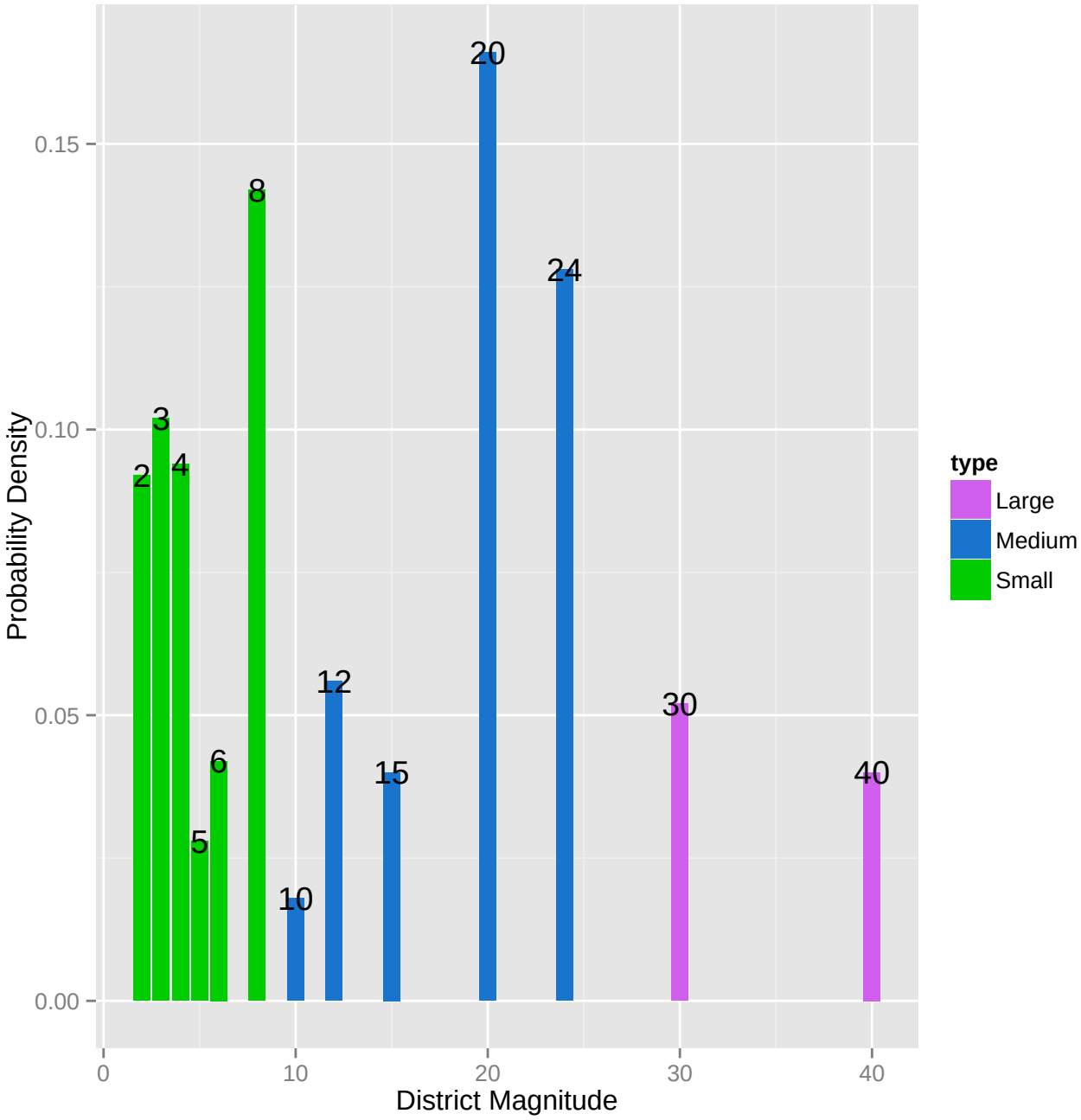

Figure 17: Probability Distribution of least undesirable District Magnitude. *Gallagher* and *ENP with SSPI*

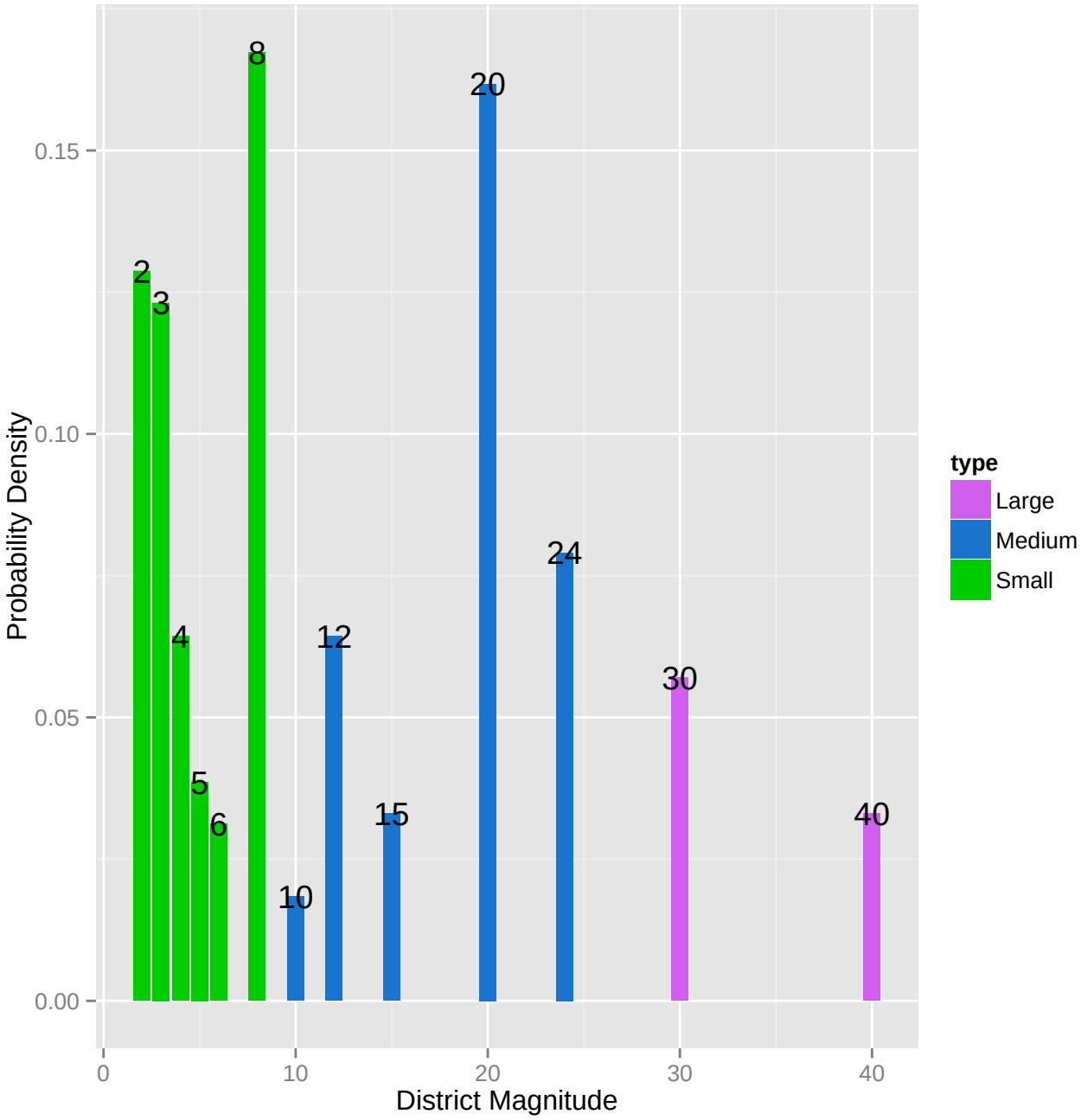

Figure 18: Probability Distribution of least undesirable District Magnitude. *Loosemorehanby* and *ENP with SSPI*

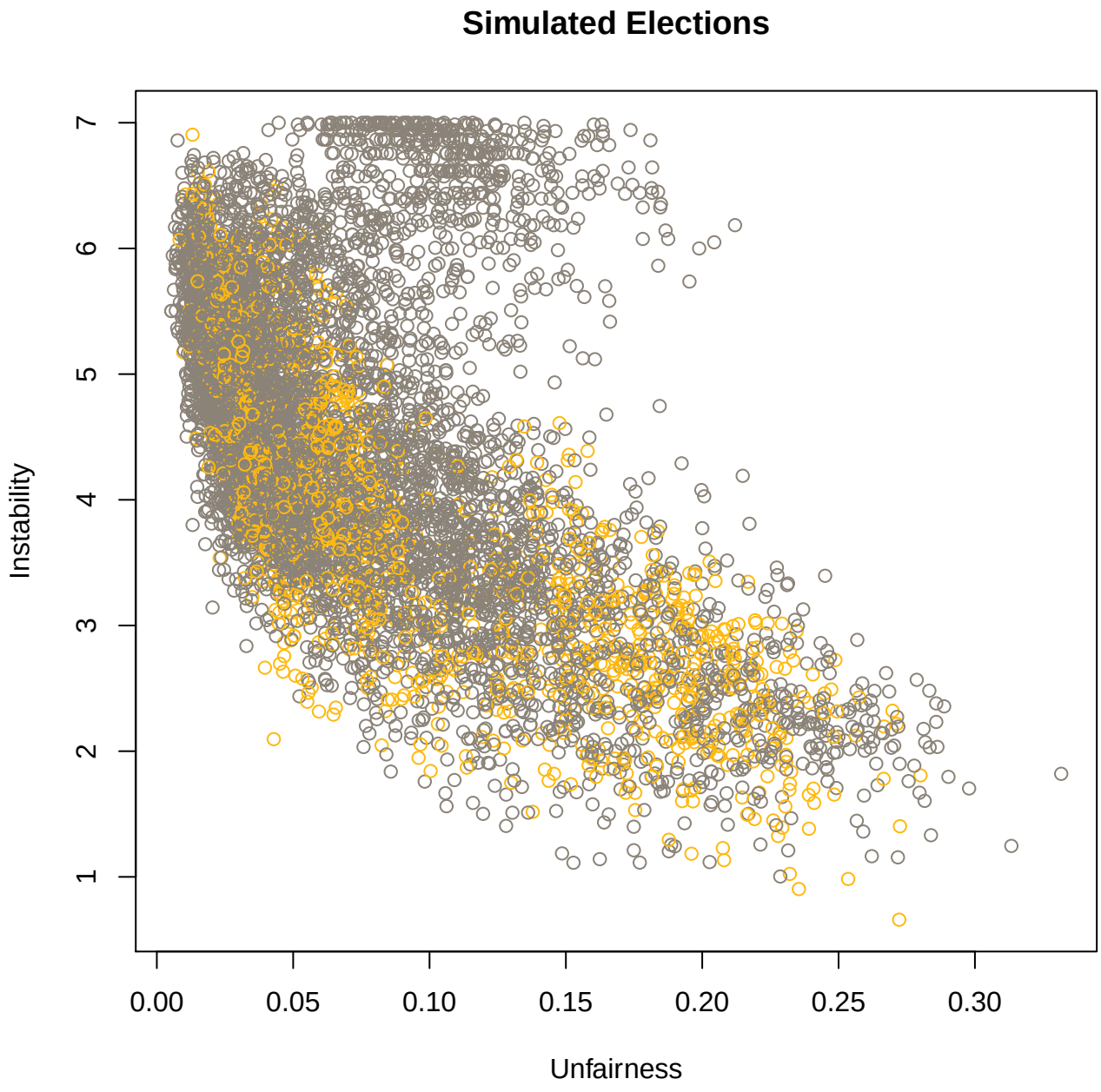

Figure 19: Results from simulated elections with the *Spatial Model* with 2 *dimensions* and the *STV* rule. Plotted using the *Gallagher* Index vs. *ENP*. "Optimal DM" (Yellow) All else (Gray)

13 Discussion

As can be seen from the plots, we found curvature in our results, however the shape of the curvature varies between models and between voting systems.

Of particular interest is the shape of the curvature in Figure 5 for very high district magnitude.

Although fairness tends to increase at the expense of stability with an increase in District Magnitude, we found that extremely high values of District Magnitude under STV have consistently worse results than for lower values of District Magnitude. In our simulations this applies to $DM > 30$. This applies to the Spatial Model, preference swapping model and the Polya Eggenberger Model. There is a possibility that this effect is due to problems with our choice of measure of unfairness, in particular that the measures of unfairness used take into account only first preferences.

This effect still applies when we increase the number of districts to the much higher level of 600. See Figure (6).

Interestingly, with the District Proportional rule, we found almost the opposite effect. Likewise increasing District Magnitude tends to trade-off an increase in fairness for a decrease in stability, but extremely high values of District Magnitude tends to have higher stability than for slightly lower values of District Magnitude while still having gains in fairness.

We find that the choice of measurements has a non trivial effect on our results. For example in Figure 5, it is debatable whether Green (DM 2-10) or Blue (DM>10) is optimal; as around $DM = 4 - 6$ the trade-off is very close to linear but, at the extremes, clearly the Blue values $DM = 10 - 29$ is optimal in the case that we favour fairness over stability and very small values $DM = 2 - 3$ is optimal if we favour stability over fairness.

By plotting the same simulation using ENP of the Shapley Shubik Power Index (see Section 4.2.2), Figure 10, we find an even stronger effect than we observed on each side of $DM = 4 - 6$. In particular $DM = 3$ seems to do very well and $DM = 10$ has much better 95th percentile performance than $DM = 8$, in both fairness and stability.

The Loosemorehanby Index doesn't seem to have much of an effect on our results. (See Figure 5 vs. Figure 12)

The results from the preference swapping model and STV are very similar to those from the Spatial Model and STV, in contrast to the Polya Eggenberger Model and STV. This is not surprising as we would expect the preference swapping model and Spatial Model to be the most realistic and so have similar results as opposed to the highly unrealistic Polya Eggenberger.

In Figure 15 it is clear that the extreme values are at $DM = 1$ and $DM \geq 60$, both of which we ignore in further plots.

When plotting using the Loosemorehanby Index, we get better results for $DM = 2$ than $DM = 4$, however, with the Gallagher Index, $DM = 3$ does much better than $DM = 2$ ($DM = 3$ also does well under the Gallagher Index).

There are relatively constant peaks at $DM = 8$ and $DM = 20$ across the different plots.

There are small values in $DM = 12$ and $DM = 24$. They are by comparison to other large values, much smaller, in particular when considering how close $DM = 10 - 12$ is to the nearby peak at $DM = 8$; so we ignore these.

Overall, $DM = 3$, $DM = 8$ and $DM = 20$ seem to be the best. See Figure 19 for a comparison runs of these optimal District Magnitudes vs. non-optimal District Magnitudes.

Supplementary Member has a much different curve than for the other systems. This system does very poorly in terms of both fairness and stability compared to other systems. It seems that medium to high district magnitudes around $DM = 30$ are the best parameter values under this

system.

The regression fits were poor due to the large amount of variation within a single class and due to the distribution of District Magnitudes we used. Unfortunately we are not able to, for example, use equally sized District Magnitudes of size 80 in an election with 120 seats. The regression fits showed strange behaviour around these values. We only used the regression fits for illustrative purposes so this will not have an effect on our conclusion.

We found that Hamilton and St Lague apportionment methods are comparable in both fairness and stability.

We found that Hill and Jefferson methods tend to trade-off fairness for stability, more so with the Jefferson method.

14 Conclusion

We found extremely strong evidence of non-linearity in the results but this non-linearity mostly applies to extreme values of District Magnitude. Clearly a District Magnitude of 1 (Red) tends to do very poorly. Performance of extremely high values compared to slightly lower values of District Magnitude varies from system to system and model to model. It appears that the trade-off in the context of small to medium sized District Magnitudes $DM = 2 - 7$ is roughly linear.

We agree with (Carey and Hix, 2011) that low magnitude systems tend to do better if we value each property equally, however slightly increasing or decreasing District Magnitude from this range has great effects.

We found the same general results applied when raising the number to an extremely high level.

We estimate that depending on how much fairness and stability are valued with respect to each other, optimal District Magnitude is $DM = 3$, $DM = 8$ or $DM = 20$. We find that Supplementary Member has poor performance, however optimal DM of the largest district under SM is around 30.

We found extremely strong evidence of a difference between the apportionment systems. We found a difference between the St Lague method and Jefferson method and a larger difference between the St Lague method and the Hill method. We found no difference between St Lague and Hamilton.

The Hill and Jefferson methods trade-off fairness for stability [30]. The Jefferson method does this more so. The Jefferson method seems to gain less stability for the increase in fairness that the Hill method gets.

15 Possible Further Research

The artificial societies generated in these simulations assumed independent districts, which does not accurately represent the real world. It is not clear how to generate districts that are not independent.

The Spatial Model could be expanded to include more parameters. For example each voter could have an extra vector which represents how important each issue is to the voter. Distances between the voter and a given party along a given axis would then be multiplied by the corresponding value in this vector to calculate the new distance along the axis.

All of the measures of unfairness used only take into account the first preferences of voters. It is not clear how to measure unfairness by taking into account complete preference orders. This may be the cause of extreme values of District Magnitude under STV having poor results for fairness under our simulations.

This simulation assumes all voters vote sincerely. In reality this assumption is simply not true due for example to tactical voting. It is possible that certain voting systems or parameters for these voting systems have different susceptibility to tactical voting. In order to accommodate this, we would need some method of transforming true preference orders to modified tactical preference orders. It may be possible to approximate this process iteratively. For example, we could loop some large number of times, with each iteration we would select a number of like voters and check whether switching their tactical vote results in these voters getting better representation. Voters that get better representation in this case would keep this new tactical vote with some probability.

[30] When compared to St Lague

It is unclear how we would measure whether or not a particular voter gets better representation, in particular taking into account their full preference orders.

References

Benoit, K. (2000). Which electoral formula is the most proportional? a new look with new evidence. *Political Analysis*, 8(4):381–388.

Carey, J. and Hix, S. (2011). The electoral sweet spot: Low-magnitude proportional electoral systems. *American Journal of Political Science*, 55(2):383–397.

Laakso, M. and Taagepera, R. (1979). Effective number of parties: A measure with application to west europe. *Comparative political studies*, 12(1):3–27.

NZEC (2011). General election 2011. http://www.electionresults.govt.nz/.

Pritchard, D. G. and Wilson, D. M. C. (2011). 2011 referendum simulator. http://www.stat.auckland.ac.nz/ geoff/voting/.

Shapley, L. and Shubik, M. (1954). A method for evaluating the distribution of power in a committee system. *American Political Science Review*, 48(03):787–792.

Vowles, J., Miller, R., Banducci, S., Sullivan, A., Karp, J., and Curtin, J. (2008). 2008 election study. http://www.nzes.org/exec/show/2008.

A FSharp Sourcecode

```
open CS380
open CS380.VotingRules

open System
open System.Threading.Tasks;

let constrain (x:double, lower:double, upper:double) =
    Math.Min(Math.Max(x, lower), upper);

let r() =
    Rand.NextDouble();

let numberOfParties = 8
let electorateCount = 120
let voteCount = 500
let simulatedElectionCount = 2000

let sampleRandVector() =
    let seq1 = Seq.toList(seq { for i in 1 .. numberOfParties do yield
        r() })
    seq { for x in seq1 do yield x }

let sampleUnitVector() =
    let seq1 = sampleRandVector()
    let sum = Seq.sum(seq1)
    seq { for x in seq1 do yield x / sum }

let sumOfSquares s =
    Seq.sum(seq { for x in s do yield x * x });

let calcEntropy s =
    Seq.sum(seq { for x in s do yield if x = 0.0 then 0.0 else x * Math
        .Log(x) });

let normalize s =
    let total = Seq.sum(s)
    seq { for x in s do yield x / total }

let normalizeDistance s =
    let dist = sumOfSquares(s)
    let result = seq { for x in s do yield x / sqrt(dist) } |> Seq.
        toArray
    let dist2 = sumOfSquares(result)
```

```
    result

let abs (x:float) =
    Math.Abs(x)

let simulateElectionsByModel2 (societyGen:unit->
    ArtificialSocietyGenerator) (ruleFactory:unit->VotingRule) (filename
    :string) (districtCount:int) =
    let strm = System.IO.File.Open(filename, System.IO.FileMode.
        OpenOrCreate)
    strm.SetLength((int64)0)
    let report = new CSVReportWriter<SimulationResults>(new CSVWriter(
        strm))

    report.AddColumn("lijphart", fun a -> a.LijphartIndex.ToString())
    report.AddColumn("raes", fun a -> a.RaesIndex.ToString())
    report.AddColumn("loosemorehanby", fun a -> a.LoosemoreHanbyIndex.
        ToString())
    report.AddColumn("gallagher", fun a -> a.GallagherIndex.ToString())
    report.AddColumn("enp", fun a -> a.EffectiveNumberOfParties.
        ToString())
    report.AddColumn("pca", fun a -> a.EffectiveNumberOfPCAVars.
        ToString())
    report.AddColumn("governability", fun a -> a.Governability.ToString
        ())
    report.AddColumn("entropy", fun a -> a.EntropyIndex.ToString())
    report.AddColumn("entropyseatprop", fun a -> a.EntropySeatPropIndex
        .ToString())

    let performSimulation() =
        let society = societyGen()
        society.PartyCount <- numberOfParties;
        society.ElectorateCount <- districtCount;

        society.SetupElection();
        let districts = seq { for index in 1..society.ElectorateCount
            do yield society.SampleElectorate(voteCount)}
        let situation = new VotingSituation();
        situation.PartyCount <- numberOfParties;

        let processDistrict d =
            let ev = new ElectorateVotes();
            ev.Magnitude <- society.DistrictMagnitude;
            ev.VoteCounts <- d;
            ev;
```

```
            situation.SetElectorates(seq { for district in districts do
                yield processDistrict(district) });

            let rule = ruleFactory()
            let simResults = SimulationResults.ComputeSimulation(situation,
                rule)
            simResults

        let results : SimulationResults array = Array.zeroCreate(
            simulatedElectionCount)

        ignore (Parallel.For (0, simulatedElectionCount, (fun (iteration:
            int) ->
            results.[iteration] <- performSimulation()
            System.Console.WriteLine(iteration)
            )))

        for result in results do
            report.WriteLine(result)

        report.Close()
        System.Console.Write("Test ")
        System.Console.Write(filename)
        System.Console.WriteLine(" Complete")

    let simulateElectionByModel (societyGen:unit->
        ArtificialSocietyGenerator) (ruleFactory:unit->VotingRule) (filename
        :string) =
        simulateElectionsByModel2 societyGen ruleFactory filename (
            electorateCount / societyGen().DistrictMagnitude)

    let isValidDistrictMagnitude(dm:int) =
        let districtCount:int = electorateCount / dm
        let districtCount2 = districtCount * dm;
        (districtCount2 = electorateCount)

    [<EntryPoint>]
    let main args =
        printfn "CS380 Hypothesis Testing"

        let stv() = new STVVotingRule() :> VotingRule
        let fpp() = new FPPVotingRule() :> VotingRule

        let pdHamilton() =
            let rule = new ProportionalByDistrictVotingRule()
            rule.Apportionment <- ApportionmentMethod.Hamilton
```

```
    rule :> VotingRule

let pdHill() =
    let rule = new ProportionalByDistrictVotingRule()
    rule.Apportionment <- ApportionmentMethod.Hill
    rule :> VotingRule

let pdJefferson() =
    let rule = new ProportionalByDistrictVotingRule()
    rule.Apportionment <- ApportionmentMethod.Jefferson
    rule :> VotingRule

let pd() =
    let rule = new ProportionalByDistrictVotingRule()
    rule.Apportionment <- ApportionmentMethod.StLague
    rule :> VotingRule


let sm() =
    let rule = new SupplementaryMemberVotingRule()
    rule.Apportionment <- ApportionmentMethod.StLague
    rule.TotalSeats <- electorateCount
    rule :> VotingRule

let districtMagnitudes = seq { 1..electorateCount } |> Seq.where(
   isValidDistrictMagnitude) |> Seq.toArray;

// Spatial Model

for dm in districtMagnitudes do
    let spatialSocietyFactory() :ArtificialSocietyGenerator =
        let society = new SpatialArtificialSociety()
        society.EnableMajorLeftRight <- true;
        society.Dimensions <- [| 1.0; 1.0; |]
        society.DistrictMagnitude <- 1;
        society :> ArtificialSocietyGenerator

    simulateElectionsByModel2 spatialSocietyFactory sm ("spatial2_"
        + dm.ToString() + "_sm.csv") (electorateCount-dm)

for dm in districtMagnitudes do
    let spatialSocietyFactory() :ArtificialSocietyGenerator =
        let society = new SpatialArtificialSociety()
        society.EnableMajorLeftRight <- true;
        society.Dimensions <- [| 1.0; 1.0; |]
        society.DistrictMagnitude <- 1;
```

```
        society :> ArtificialSocietyGenerator

    simulateElectionsByModel2 spatialSocietyFactory stv ("spatial2_
        " + dm.ToString() + "_stv.csv") (electorateCount/dm)

// Urn Model

for dm in districtMagnitudes do
    let urnSocietyFactory() : ArtificialSocietyGenerator =
        let society = new UrnArtificialSociety()
        society.AlphaGenerator <- fun () ->
            let beta = r()
            beta / (1.0 - beta)

        society.DistrictMagnitude <- dm;
        society :> ArtificialSocietyGenerator

    simulateElectionsByModel2 urnSocietyFactory stv ("urn_" + dm.
        ToString() + "_stv.csv") (electorateCount/dm)

// Preference Swapping Model
let nzSpatial = SpatialFile.FromStream("nzspatial.csv")
let nzPref = PreferenceFile.FromStream("nzpref2.csv");

for dm in districtMagnitudes do
    let prefSwapSocietyFactory() : ArtificialSocietyGenerator =
        let society = new PreferenceSwappingArtificialSociety()
        society.DistrictMagnitude <- dm;
        society.PreferenceInformation <- nzPref;
        society.SpatialInformation <- nzSpatial;
        society.MaxChanceSwap2 <- 0.18;
        society.MaxChanceSwap3 <- 0.12;
        society :> ArtificialSocietyGenerator

    simulateElectionsByModel2 prefSwapSocietyFactory stv ("
        nzprefswap_" + dm.ToString() + "_stv.csv") (electorateCount
        / dm)

Console.WriteLine("Simulation Complete")

0 // return an integer exit code
```

B R Sourcecode

```
 electorateCount = 70;
electorateIds = 1:electorateCount;
parties = c(
        "Labour",
        "New Zealand First Party",
        "Conservative Party",
        "National Party",
        "Green Party",
        "ACT New Zealand",
        "Alliance",
        "Aotearoa Legalise Cannabis Party",
        "Democrats for Social Credit",
        "Libertarianz",
        "Mana",
        "M   ori Party",
        "United Future"
);

nzspatial.df = data.frame()
districtNames = vector()
for (id in electorateIds)
{
        htmlSource = paste("http://www.electionresults.govt.nz/
            electionresults_2011/electorate-",id,".html",sep="");
        html = readLines(htmlSource)

        titleIndex = grep("Official Count Results -- ", html)
        districtName = strsplit(strsplit(html[titleIndex]," -- ")
            [[1]][2],"</title>")[[1]][1];
        districtNames[id] = districtName;

        for (pId in 1:length(parties))
        {
                p = parties[pId];
                index = grep(p, html)
                voteCount = as.numeric(sub(",","",sub("^ +", "",
                    strsplit(strsplit(html[index],"<td align=\"right\">"
                    )[[1]][2],"</td")[[1]][1])))
                nzspatial.df[id,pId] = voteCount;
        }
}

names(nzspatial.df) = parties;
nzspatial.df$name = districtNames;
```

```
library(ggplot2)

tinyColor = "firebrick1";
smallColor = "green3";
mediumColor = "dodgerblue3";
largeColor = "mediumorchid2";

map = function(v, f)
{
        result = NULL

        for (i in 1:length(v))
        {
                result[i] = f(v[i]);
        }

        return (result);
}

datafiles.df = data.frame(
        datasource = c(
                "C:/Users/Michael/Documents/Visual Studio 11/Projects/
                    CS380/HypothesisTesting/bin/Debug/results2/pd_
                    spatial2_results.csv",
                "C:/Users/Michael/Documents/Visual Studio 11/Projects/
                    CS380/HypothesisTesting/bin/Debug/spatial_1/spatial1
                    .csv",
                "C:/Users/Michael/Documents/Visual Studio 11/Projects/
                    CS380/HypothesisTesting/bin/Debug/spatial_2/spatial2
                    .csv",
                "C:/Users/Michael/Documents/Visual Studio 11/Projects/
                    CS380/HypothesisTesting/bin/Debug/spatial_3/spatial3
                    .csv",
                "C:/Users/Michael/Documents/Visual Studio 11/Projects/
                    CS380/HypothesisTesting/bin/Debug/results5/urnstv.
                    csv",
                "C:/Users/Michael/Documents/Visual Studio 11/Projects/
                    CS380/HypothesisTesting/bin/Debug/spatial_2pure/
                    spatial2pure.csv",
                "C:/Users/Michael/Documents/Visual Studio 11/Projects/
                    CS380/HypothesisTesting/bin/Debug/spatial_2large/
                    spatial2large.csv",
                "C:/Users/Michael/Documents/Visual Studio 11/Projects/
                    CS380/HypothesisTesting/bin/Debug/spatial2_sm/
                    spatial2sm.csv"
        ),
```

```
        name=c(
                "pd_spatial2",
                "stv_spatial1",
                "stv_spatial2",
                "stv_spatial3",
                "stv_urn",
                "stv_spatial2pure",
                "spatial2large",
                "sm_spatial2"
        ),stringsAsFactors=FALSE
)

for (fairnessId in c(0,1))
{
        for (stabilityId in c(0,1,2))
        {
                bestdmoverall.df = data.frame()

                for(datasetId in 1:nrow(datafiles.df))
                {
                        results.df = read.table(datafiles.df$datasource
                            [datasetId], sep = ",", skip = 0, header=T)
                        datasetName = datafiles.df$name[datasetId]

                        results.df$X <- NULL
                        results.df[1:5,]

                        fairnessName = "UNKNOWN"
                        if (fairnessId==0)
                        {
                                results.df$fairness = results.df$
                                    gallagher;
                                fairnessName = "gallagher";
                        }

                        if (fairnessId==1)
                        {
                                results.df$fairness = results.df$
                                    loosemorehanby;
                                fairnessName = "loosemorehanby";
                        }

                        stabilityName = "UNKNOWN"
                        if (stabilityId==0)
                        {
```

```
        results.df$stability = results.df$enp
            -1;
        stabilityName = "enp";
}

if (stabilityId==1)
{
        results.df$stability = (1/results.df$
            governability)-1;
        stabilityName = "enpsspi";
}

if (stabilityId==2)
{
        results.df$stability = -results.df$
            entropy;
        stabilityName = "entropysspi";
}

#plot.new()
pdf(paste("C:/Users/Michael/Documents/Visual␣
    Studio␣11/Projects/CS380/Report/images/",
    datasetName,"␣",fairnessName,"␣",
    stabilityName, ".pdf",sep=""))

split.screen(c(2,2))

screen(1)
results.df$col1 = largeColor
results.df$col1[results.df$dm<30] = mediumColor
results.df$col1[results.df$dm<10] = smallColor
results.df$col1[results.df$dm==1] = tinyColor
rOrder = order(rnorm(nrow(results.df)))
plot(results.df$fairness[rOrder], results.df$
    stability[rOrder], xlab="Unfairness", ylab="
    Instability",col=results.df$col1[rOrder],
    main="Simulated␣Elections")
results.df$col1 <- NULL

#Fit fairness
fairness.fit = lm(fairness~I(dm)+I(dm^2)+I(log(
    dm)+I(log(dm)^2)), data=results.df)
summary(fairness.fit)

#Fit stability
stability.fit = lm(stability~I(dm)+I(log(dm))+I
```

```
    (dm^2), data=results.df)
summary(stability.fit)

dummy.df = data.frame(dm=(10:1200)/10)
dummy.df$fairness = (predict(fairness.fit,
    dummy.df))
dummy.df$stability = predict(stability.fit,
    dummy.df)
dummy.df$isPredicted = TRUE;
dummy.df$type=1

dummy.df$col1 = largeColor
dummy.df$col1[dummy.df$dm<30] = mediumColor
dummy.df$col1[dummy.df$dm<10] = smallColor
dummy.df$col1[dummy.df$dm==1] = tinyColor

screen(2)
randOrder = order(rnorm(nrow(dummy.df)))
plot(dummy.df$fairness[randOrder], dummy.df$
    stability[randOrder], xlab="Unfairness",
    ylab="Instability", main="Predicted␣
    Elections␣by␣DM", col=dummy.df$col1[
    randOrder])

#Calculate extremes
#Note this plot may be misleading. Points on
    this plot do not necessarily correspond to
    actual predicted elections.
extreme.df = data.frame(dm=unique(results.df$dm
    ))
for(dm in extreme.df$dm)
{
        extreme.df$fairness05[extreme.df$dm==dm
            ] = quantile(results.df$fairness[
            results.df$dm==dm], 0.05)
        extreme.df$fairness95[extreme.df$dm==dm
            ] = quantile(results.df$fairness[
            results.df$dm==dm], 0.95)
        extreme.df$stability05[extreme.df$dm==
            dm] = quantile(results.df$stability[
            results.df$dm==dm], 0.05)
        extreme.df$stability95[extreme.df$dm==
            dm] = quantile(results.df$stability[
            results.df$dm==dm], 0.95)
}
```

```
screen(3)

extreme.df$col1 = largeColor
extreme.df$col1[extreme.df$dm<30] = mediumColor
extreme.df$col1[extreme.df$dm<10] = smallColor
extreme.df$col1[extreme.df$dm==1] = tinyColor
plot(extreme.df$fairness95, extreme.df$
    stability95, xlab="Unfairness", ylab="
    Instability", main="95th percentile by DM",
    pch=15,col=extreme.df$col1)

unknown = rep(NA,length(unique(results.df$dm)))
actualmeans.df = data.frame(dm=unknown,
    fairness=unknown, stability=unknown)
actualmeans.df$dm = unique(results.df$dm)
actualmeans.df$type=0
actualmeans.df$isPredicted=FALSE

for (dm in actualmeans.df$dm)
{
        actualmeans.df$fairness[actualmeans.df$
            dm == dm] = mean(results.df$fairness
            [results.df$dm==dm])
        actualmeans.df$stability[actualmeans.df
            $dm == dm] = mean(results.df$
            stability[results.df$dm==dm])
}

actualmeans.df$col1 = largeColor
actualmeans.df$col1[actualmeans.df$dm<30] =
    mediumColor
actualmeans.df$col1[actualmeans.df$dm<10] =
    smallColor
actualmeans.df$col1[actualmeans.df$dm==1] =
    tinyColor

screen(4)
plot(actualmeans.df$fairness, actualmeans.df$
    stability, xlab="Unfairness", ylab="
    Instability", main="Mean Elections by DM",
    pch=15, col=actualmeans.df$col1)

dev.off()

pdf(paste("C:/Users/Michael/Documents/Visual
    Studio 11/Projects/CS380/Report/images/",
```

```
    datasetName,"_",fairnessName,"_",
    stabilityName,"_optimal", ".pdf",sep=""))
results.df$col2 = "antiquewhite4";
results.df$col2[results.df$dm==3] = "
    darkgoldenrod1"
results.df$col2[results.df$dm==8] = "
    darkgoldenrod1"
results.df$col2[results.df$dm==20] = "
    darkgoldenrod1"
plot(results.df$fairness[rOrder], results.df$
    stability[rOrder], xlab="Unfairness", ylab="
    Instability",col=results.df$col2[rOrder],
    main="Simulated Elections")
dev.off()

undesirability.df = actualmeans.df;
undesirability.df$zFairness = (undesirability.
    df$fairness - min(undesirability.df$fairness
    ))/sd(undesirability.df$fairness)
undesirability.df$zStability = (undesirability.
    df$stability - min(undesirability.df$
    stability))/sd(undesirability.df$stability)

tradeoff.df = data.frame(tCoef=0:100)
for (tradeoffCoef in tradeoff.df$tCoef)
{
        tradeoffCoef2 = tradeoffCoef / 100.0;

        #Calculate position on line.
        ptX = tradeoffCoef2;
        ptY = 1.0 - tradeoffCoef2;
        distLine = sqrt(ptX * ptX + ptY * ptY);
        xCoef = ptX / distLine;
        yCoef = ptY / distLine;

        undesirability.df$undesirability =
                sqrt(
                        (undesirability.df$
                            zFairness * xCoef)^2
                            +
                        (undesirability.df$
                            zStability * yCoef)
                            ^2);

        tradeoff.df$bestDM[tradeoff.df$tCoef==
            tradeoffCoef] = undesirability.df$dm
```

```
            [order(undesirability.df$
            undesirability)][1];

        print(undesirability.df)
        print(undesirability.df$dm[order(
            undesirability.df$undesirability)])
}

print(tradeoff.df)

bestdm.df = data.frame(dm=unique(results.df$dm)
    )
bestdm.df$count = map(bestdm.df$dm, function(dm
    ) { sum(tradeoff.df$bestDM == dm) })
bestdm.df$prdensity = bestdm.df$count / sum(
    bestdm.df$count);

bestdm.df$type = "Large"
bestdm.df$type[bestdm.df$dm<30] = "Medium"
bestdm.df$type[bestdm.df$dm<10] = "Small"
bestdm.df$type[bestdm.df$dm==1] = "Tiny"

bestdm.df$col1 = largeColor
bestdm.df$col1[bestdm.df$dm<30] = mediumColor
bestdm.df$col1[bestdm.df$dm<10] = smallColor
bestdm.df$col1[bestdm.df$dm==1] = tinyColor

bestdmoverall.df = merge(bestdm.df,
    bestdmoverall.df, all=TRUE)

plot.new()

pdf(paste("C:/Users/Michael/Documents/Visual␣
    Studio␣11/Projects/CS380/Report/images/",
    datasetName,"_",fairnessName,"_",
    stabilityName,"_tradeoff",".pdf",sep=""))
print(ggplot(bestdm.df, aes(dm, prdensity, fill=
    type)) + geom_bar(stat="identity")+scale_x_
    continuous(name="District␣Magnitude")+scale_
    y_continuous(name="Probability␣Density")+
    scale_fill_manual(values = c("Large" =
    largeColor, "Medium" = mediumColor, "Small"
    = smallColor, "Tiny" = tinyColor)) )
dev.off()

tradeoff2.df = tradeoff.df[(tradeoff.df$bestDM
```

```
            > 1) & (tradeoff.df$bestDM < 120),]
}


bestdmoverall2.df = data.frame(dm=unique(results.df$dm)
    )
bestdmoverall2.df$count = map(bestdmoverall2.df$dm,
    function(dm) { sum(bestdmoverall.df$prdensity[
    bestdmoverall.df$dm == dm]) })
bestdmoverall2.df$prdensity = bestdmoverall2.df$count /
     sum(bestdmoverall2.df$count);
bestdmoverall2.df$type = "Large"
bestdmoverall2.df$type[bestdmoverall2.df$dm<30] = "
    Medium"
bestdmoverall2.df$type[bestdmoverall2.df$dm<10] = "
    Small"
bestdmoverall2.df$type[bestdmoverall2.df$dm==1] = "Tiny
    "


bestdmoverall2.df$col1 = largeColor
bestdmoverall2.df$col1[bestdmoverall2.df$dm<30] =
    mediumColor
bestdmoverall2.df$col1[bestdmoverall2.df$dm<10] =
    smallColor
bestdmoverall2.df$col1[bestdmoverall2.df$dm==1] =
    tinyColor

pdf(paste("C:/Users/Michael/Documents/Visual Studio 11/
    Projects/CS380/Report/images/", fairnessName,"_",
    stabilityName,"_tradeoff_overall",".pdf",sep=""))
print(ggplot(bestdmoverall2.df, aes(dm, prdensity,fill=
    type)) + geom_bar(stat="identity")+scale_x_
    continuous(name="District Magnitude")+scale_y_
    continuous(name="Probability Density")+scale_fill_
    manual(values = c("Large" = largeColor, "Medium" =
    mediumColor, "Small" = smallColor, "Tiny" =
    tinyColor))+geom_text(aes(dm, prdensity,fill=type,
    label=dm,label=dm)))
dev.off()

bestdmoverall3.df = bestdmoverall2.df[bestdmoverall2.df
    $dm > 1 & bestdmoverall2.df$dm < 60,]
bestdmoverall3.df$prdensity = bestdmoverall3.df$count /
     sum(bestdmoverall3.df$count);
pdf(paste("C:/Users/Michael/Documents/Visual Studio 11/
    Projects/CS380/Report/images/", fairnessName,"_",
```

```
                    stabilityName,"_tradeoff_overall_small",".pdf",sep="
                    "))
                print(ggplot(bestdmoverall3.df, aes(dm, prdensity,fill=
                    type,label=dm)) + geom_bar(stat="identity")+scale_x_
                    continuous(name="District Magnitude")+scale_y_
                    continuous(name="Probability Density")+scale_fill_
                    manual(values = c("Large" = largeColor, "Medium" =
                    mediumColor, "Small" = smallColor, "Tiny" =
                    tinyColor))+geom_text(aes(dm, prdensity,fill=type,
                    label=dm,label=dm)))
                dev.off()

        }
}

path = "C:/Users/Michael/Documents/Visual Studio 11/Projects/CS380/
    HypothesisTesting/bin/Debug/apportionment/"

hamilton.df =   read.table(paste(path,"hamilton.csv",sep=""), sep = ","
    , skip = 0, header=T)
hill.df =               read.table(paste(path,"hill.csv",sep=""), sep =
    ",", skip = 0, header=T)
jefferson.df =  read.table(paste(path,"jefferson.csv",sep=""), sep = ",
    ", skip = 0, header=T)
stlague.df =    read.table(paste(path,"stlague.csv",sep=""), sep = ",",
    skip = 0, header=T)

hamilton.df$apportionment = "hamilton";
hill.df$apportionment = "hill";
jefferson.df$apportionment = "jefferson";
stlague.df$apportionment = "a_stlague";

data.df = hamilton.df;
data.df = merge(data.df, stlague.df, all=TRUE);
data.df = merge(data.df, jefferson.df, all=TRUE);
data.df = merge(data.df, hill.df, all=TRUE);
data.df$apportionment = factor(data.df$apportionment);

data.df$fairness = data.df$gallagher;
data.df$stability = data.df$enp-1;

fitFairness = lm(fairness~apportionment,data=data.df);
fitStability = lm(stability~apportionment,data=data.df);
anova(fitFairness)
anova(fitStability)
```

```
summary(fitFairness)
summary(fitStability)

confint(fitFairness)
confint(fitStability)
```