

# Electoral Engineering through Simulation

October 16, 2012

*Author:*

Michael FOWLIE

*Supervisor:*

Mark C. WILSON

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

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Conclusion</b>	<b>4</b>
<b>3</b>	<b>Methodology</b>	<b>4</b>
3.1	Hypothesis . . . . .	4
3.2	Procedure . . . . .	5
3.2.1	Generating Data . . . . .	5
3.2.2	Statistical Analysis . . . . .	5
3.3	Tools . . . . .	6
<b>4</b>	<b>Measures</b>	<b>7</b>
4.1	Measures of Unfairness . . . . .	8
4.2	Measures of Instability . . . . .	9
4.2.1	Effective Number of Parties . . . . .	9
4.2.2	ENP using Shapley Shubik Power Index . . . . .	9
4.2.3	Entropy of Shapley Shubik Power . . . . .	11
<b>5</b>	<b>Artificial Societies</b>	<b>11</b>
5.1	Polya Eggenberger (Urn) Model . . . . .	12
5.1.1	Urn Process . . . . .	14
5.1.2	Parameter selection . . . . .	14
5.1.3	Sampling . . . . .	14
5.2	Spatial Model . . . . .	14
5.3	Preference Swapping Model . . . . .	15
<b>6</b>	<b>District Magnitude</b>	<b>15</b>
<b>7</b>	<b>Voting Rules</b>	<b>16</b>
7.1	Single Transferable Vote . . . . .	16
7.2	Proportional Systems . . . . .	16
7.2.1	District Proportional . . . . .	16
7.2.2	Supplementary Member . . . . .	16
<b>8</b>	<b>Apportionment Methods</b>	<b>17</b>
8.1	St Lague . . . . .	17
8.2	Hill . . . . .	17
8.3	Jefferson . . . . .	17
8.4	Hamilton . . . . .	17
<b>9</b>	<b>Technical Notes on Statistical Analysis</b>	<b>17</b>
<b>10</b>	<b>Efficiently Sampling from a Discrete Probability Distribution</b>	<b>17</b>
<b>11</b>	<b>Fitting real world data to the Spatial Model</b>	<b>18</b>

<b>A FSharp Sourcecode</b>	<b>20</b>
<b>B R Sourcecode</b>	<b>21</b>

# 1 Introduction

Election outcomes have two properties that we are interested in, fairness/proportionality and stability/governability. Under different voting rules (and different parameters for these rules), election outcomes have different tendencies for values for these properties. Both of these properties are desirable however it is well understood that there is a trade-off between the two.

It is claimed that the trade-off between these two properties is non linear [Carey and Hix, 2011].

If there is non linearity in the trade-off, it follows that there must be an arguably ideal point.

[Carey and Hix, 2011] uses data from 610 elections in 81 countries. It categorised voting rules based on District Magnitude (See Section 6). We investigate this trade-off in the context of artificially generated societies and in particular look for non linearity.

# 2 Conclusion

As can be seen from the below plots, we found curvature in our results. Of particular interest is the shape of the curvature in Figure 2.

We find that the choice of measurements has a great effect on our results. For example in Figure. 2, it is debatable whether Green (DM 2-10) or Blue (DM>10) is optimal; however by plotting the same simulation using ENP of the Shapley Shubik Power Index (See Section 4.2.2) (Figure 3) the Blue results seem to do much better than the Green results with significant gains in fairness for almost no loss in stability.

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 lower values of District Magnitude. In our simulations this applies to  $DM > 30$ . This applies to both the Spatial Model and the Polya Eggenberger model.

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 slightly lower values with gains in fairness.

# 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 7 and various parameters for these voting systems. So that our results to be comparable with [Carey and Hix, 2011], 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. We also consider the case using the stability index discussed in Section 4.2.2.

## 3.2 Procedure

### 3.2.1 Generating Data

### 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 6) as the explanatory variable.

See Section 9 for technical notes.

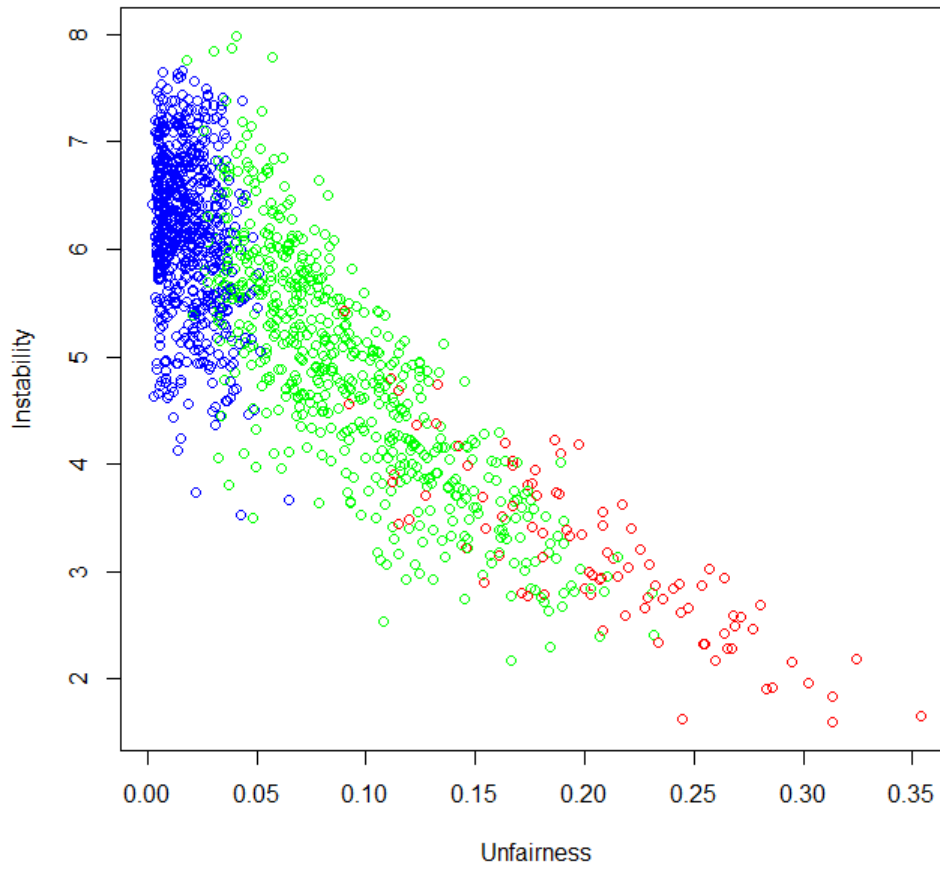


Figure 1: Results from simulated elections with the Spatial Model and District Proportional rule. DM=1 (Red) DM=2-10 (Green) DM>10 (Blue)

### 3.3 Tools

The simulator itself consists of a library written in C# 5.0 along with scripting code written in C# and F# 2.0. Simulations are performed on Amazon EC2 c1.xlarge instances with Intel® Xeon® E5410 processors and a workstation with an Intel® Pentium® G620.

Data from [Vowles et al., ] was imported and processed with scripts written in both SAS and C#.

The resulting dataset is then analysed using R 2.15.1.

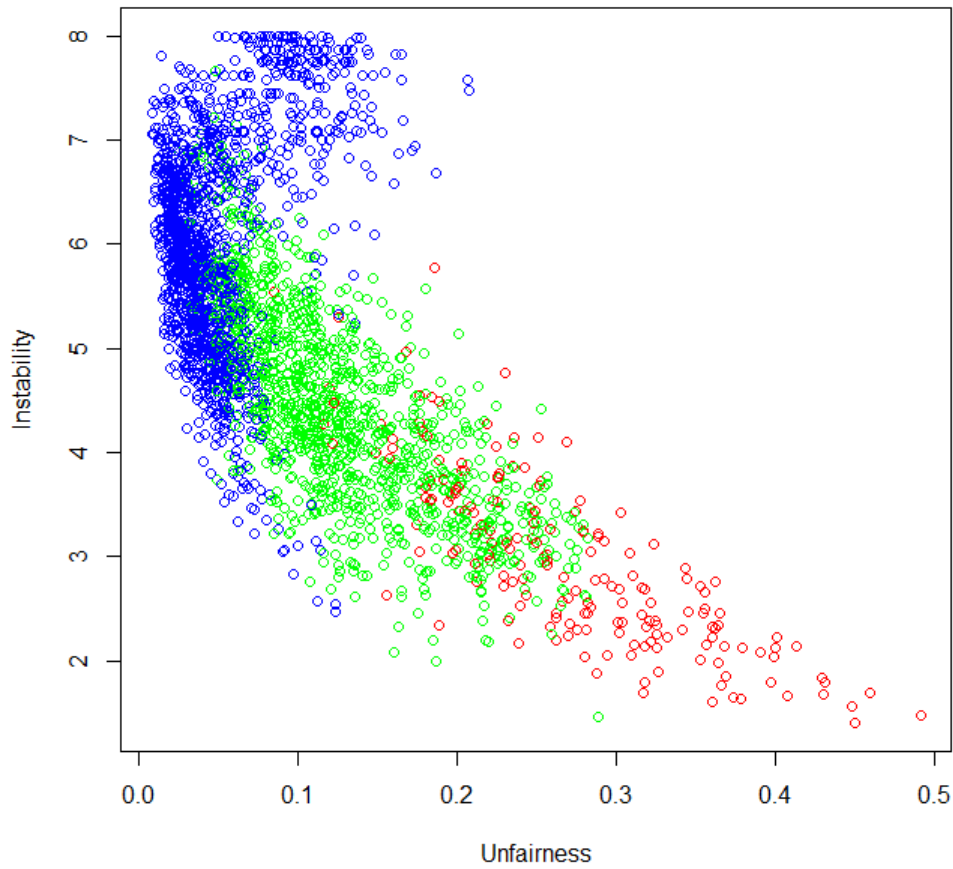


Figure 2: Results from simulated elections with the Spatial Model and STV rule. DM=1 (Red) DM=2-10 (Green) DM>10 (Blue)

## 4 Measures

Fairness and Stability are measured in a variety of ways, described below.

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

Note in our plots we normalize the point (0,0) to be optimal. For the measures 4.2.1 and 4.2.2 this involves subtracting 1 from the values.

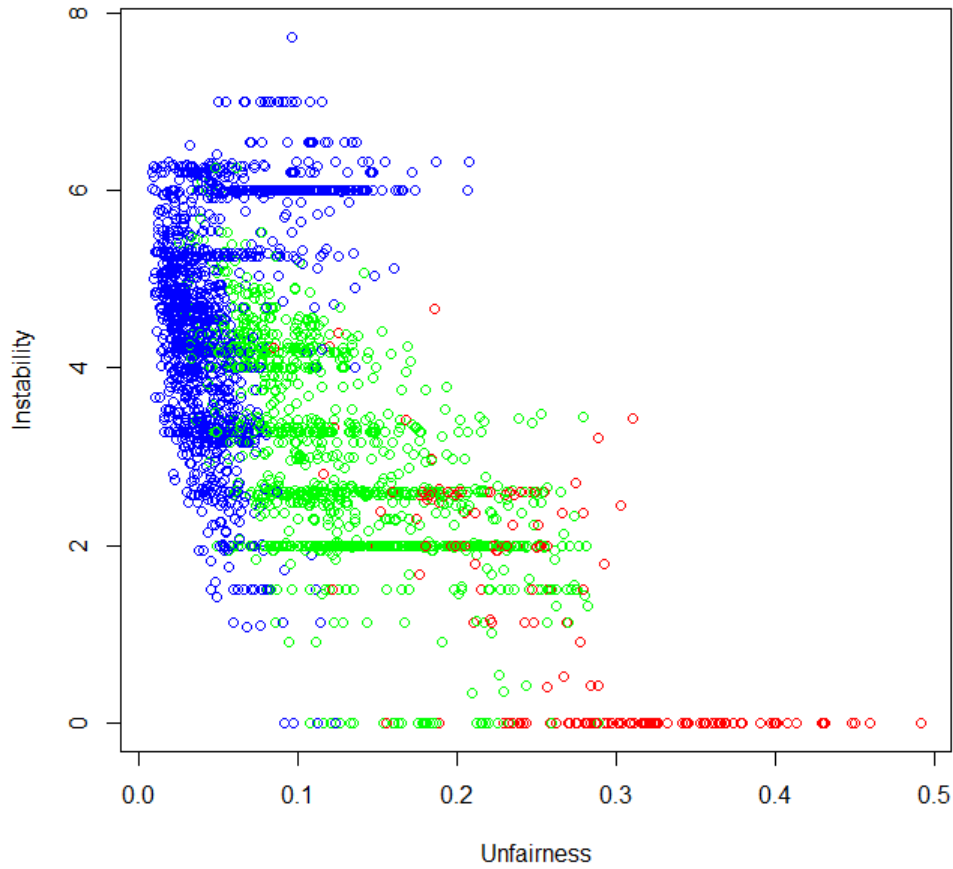


Figure 3: Results from Figure 2 using ENP with SSPI. DM=1 (Red) DM=2-10 (Green) DM>10 (Blue)

## 4.1 Measures of Unfairness

We define  $\delta_i = v_i - s_i$  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  $D = \sqrt{\frac{1}{2} \sum_{i=1}^m \delta_i^2}$ .

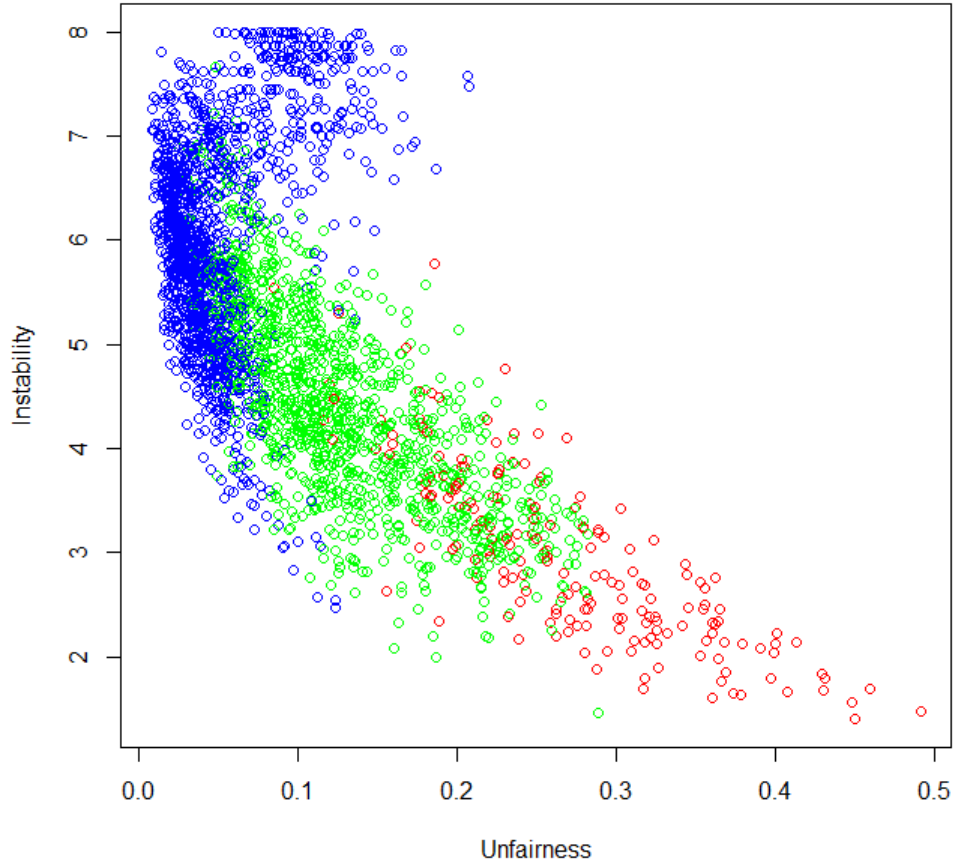


Figure 4: Results from simulated elections with the Polya Eggenberger Model and STV rule. DM=1 (Red) DM=2-10 (Green) DM>10 (Blue)



## 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 portion of seats that party  $i$  has.

### 4.2.2 ENP using Shapley Shubik Power Index

As for the purposes of governability, only power [Shapley and Shubik, 1954] is relevant (as opposed to the portion of seats), it seems reasonable to modify the ENP formula to use political power

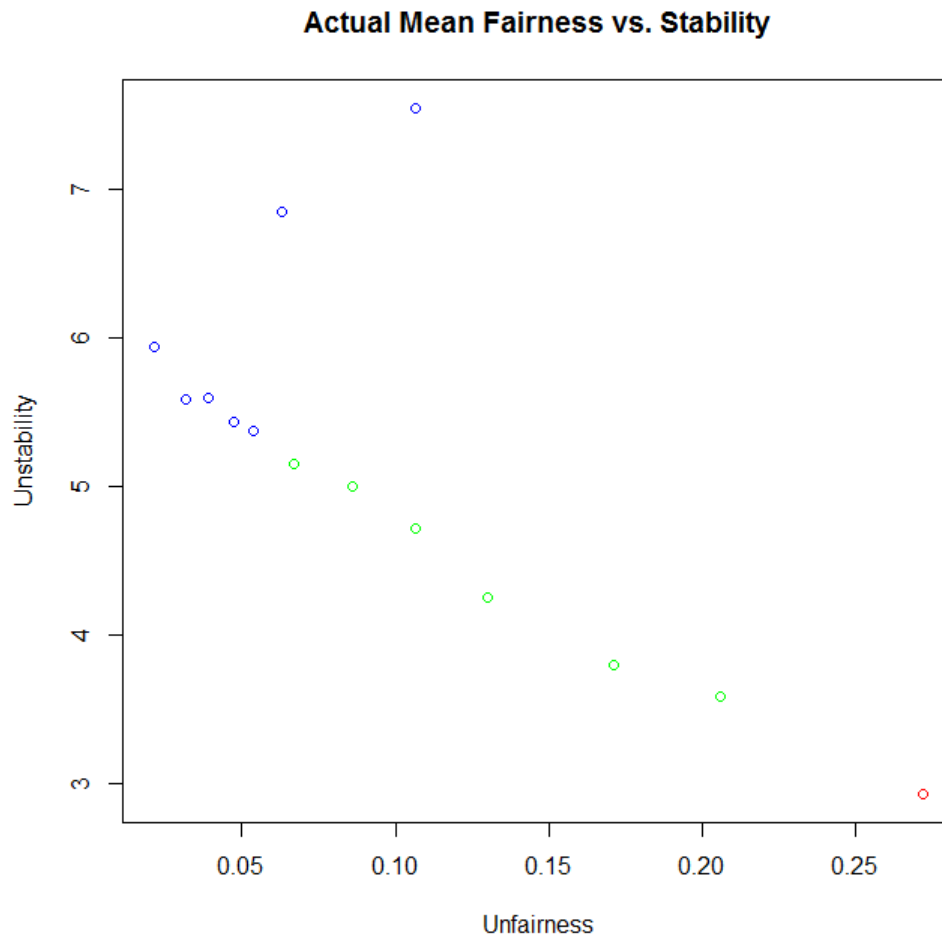


Figure 5: Mean results by District Magnitude. Same results as Figure 2

instead of the proportion of seats, by party.  $ENP = \frac{1}{\sum_{i=1}^m SSPI_i^2}$

For example a parliament with a party with 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.

However 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 10) 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.

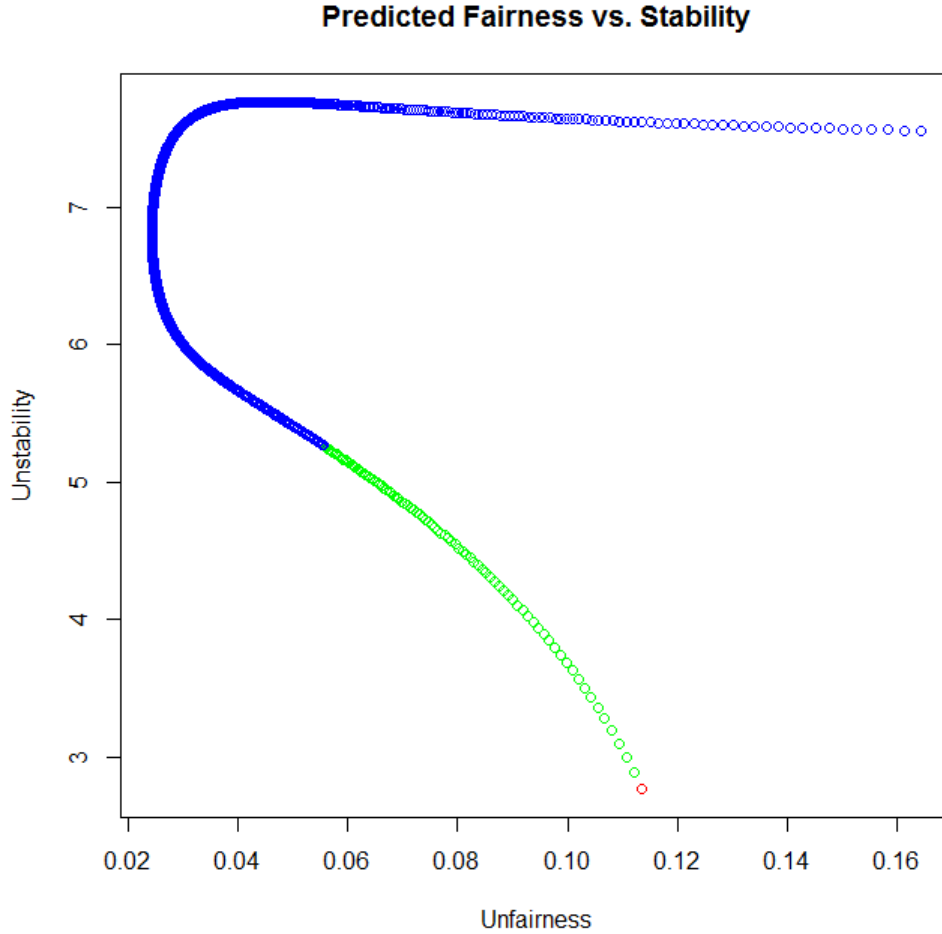


Figure 6: Fitted Regression Curve by District Magnitude. Same results as Figure 2

### 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.

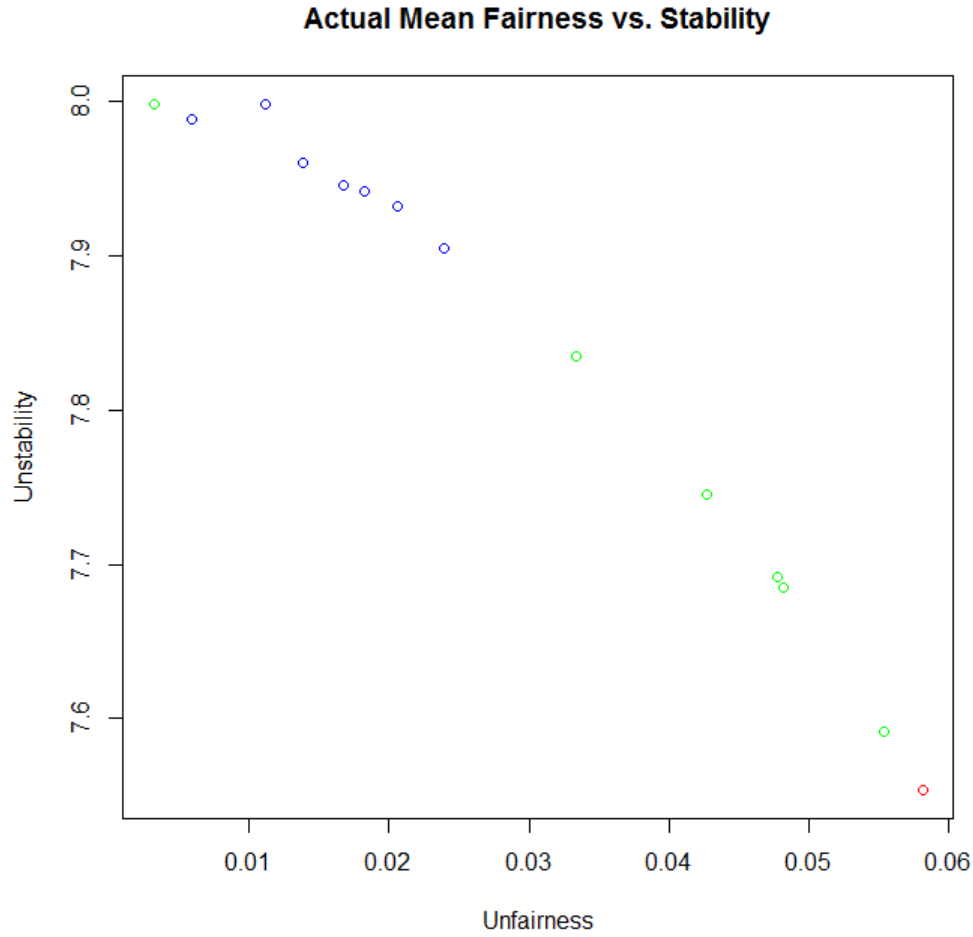


Figure 7: Mean results by District Magnitude. Same results as Figure 4

### 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 districts are independent. As we assume independence of districts and there is no bias in favour of any particular party, as the number of districts approaches  $\infty$  the distribution of votes to parties will approximate a uniform distribution - thus this model is not very realistic.

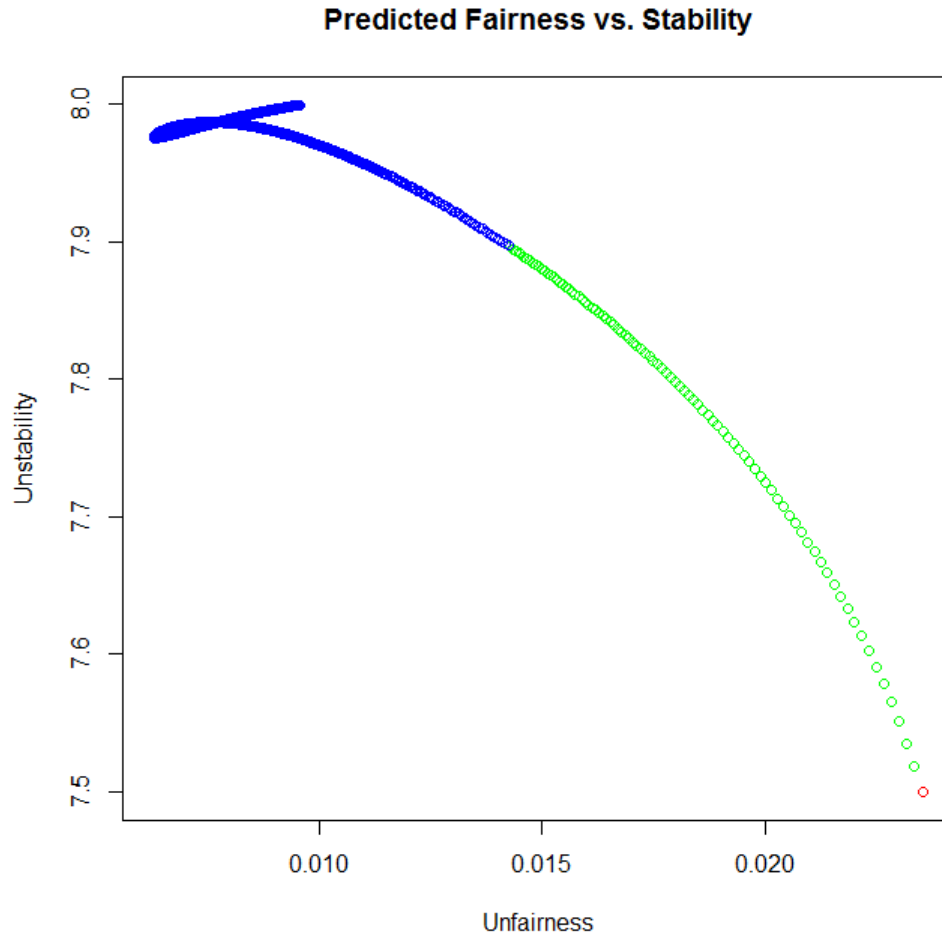


Figure 8: Fitted Regression Curve by District Magnitude. Same results as Figure 4

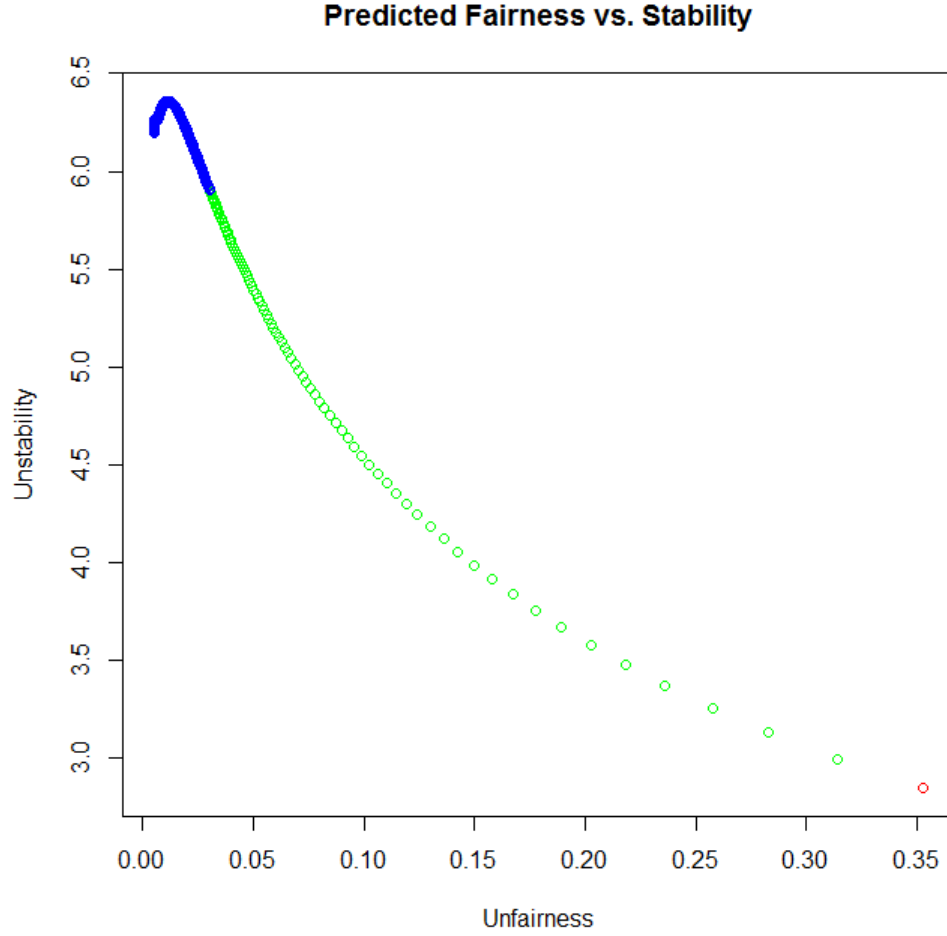


Figure 9: Fitted Regression Curve by District Magnitude. Same results as Figure 1

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
$\frac{sum_{i=1}^m SSPI_i^2}{1}$	0.3333	0.5000
$\frac{sum_{i=1}^m SSPI_i^2}{3}$	3	2

Figure 10: Under some elections, increasing the number of parties increases stability as measured as described in section 4.2.2

### 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 a preference order). To sample from the Urn we select a ball from the Urn uniformly at random, record it, and place the ball and  $a$  copies of the ball back 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.  $y = U(0, \sum_{i=1}^M w_i)$  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 better 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  $|z|$  dimensional space 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 voters 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 are instead represent the traditional major left and right parties. 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 11.

### 5.3 Preference Swapping Model

The Preference Swapping Model 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 evolving an input election by first inferring the preference orders for each vote and switching 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 (i.e. so that we model 2 or more steps into the future). Each stage is calculated in the same way with the exception that we only infer preference orders from the first preferences in the first step.

We define  $s_i$  to be the probability that a voter switches a it's 1st and  $i$ th 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  $s_i \geq s_{i+1}$

#### Parameter selection

We use the 2002, 2005, 2008 and 2011 elections in New Zealand as starting points and infer preference orders and the probability of switching preference orders from the NZES 2008 [Vowles et al., ].

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

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.

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 it's district and from election to election.

We select  $s_i$  to be the proportion of voters who's vote in 2005 (the previous election) was equal to their  $i$ th current preference.

## 6 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 7.2.2) the district magnitude varies from district to district. Using the median would group SM in the same category as FPP which is undesirable.

## 7 Voting Rules

### 7.1 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  $quota = (\frac{n}{DistrictMagnitude+1}) + 1$ . (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 of 1.0 to all votes.
- 2. Calculate the sum of weights, by party, for all votes. Select the party to assign the weight to by the highest non-excluded and non-elected party in a voter's preference order.
- 3. Each party that obtains a weight greater than or equal to the quota is elected. We reduce the weights of all votes that voted for a given elected party by multiplying each by  $m = \frac{Weight\ for\ votes\ for\ this\ party - quota}{Weight\ for\ votes\ for\ this\ party}$ , 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.

### 7.2 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 8) to select the number of seats by party, and aggregate the results. These systems only take into account first preference votes.

#### 7.2.1 District Proportional

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

#### 7.2.2 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.



## 8 Apportionment Methods

### 8.1 St Lague

### 8.2 Hill

### 8.3 Jefferson

### 8.4 Hamilton

## 9 Technical Notes on Statistical Analysis

## 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 it's 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 psuedocode:

```
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 (and the parent's parent, up to the root etc.) which takes in the worst case  $\ln |w|$  worst case time.

## 11 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 11.

We fitted a regression line and found extremely strong evidence  $p\text{-value} < 2 \times 10^{-16}$  against  $H_0$  that there was no relationship between the two variables. The fit had high error  $R^2 = 0.49$  however 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 contains 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 what the underlying distribution of  $y$  values that the  $y$  values that generated our current results is (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.

## References

- [Carey and Hix, 2011] Carey, J. and Hix, S. (2011). The electoral sweet spot: Low-magnitude proportional electoral systems. *American Journal of Political Science*.
- [Laakso and Taagepera, 1979] Laakso, M. and Taagepera, R. (1979). Effective number of parties: A measure with application to west europe. *Comparative Political Studies*.
- [Shapley and Shubik, 1954] Shapley, L. and Shubik, M. (1954). A method for evaluating the distribution of power in a committee system. *American Political Science Review*.
- [Vowles et al., ] Vowles, J., Miller, R., Banducci, S., Sullivan, A., Karp, J., and Curtin, J. 2008 election study. <http://www.nzes.org/exec/show/2008>.

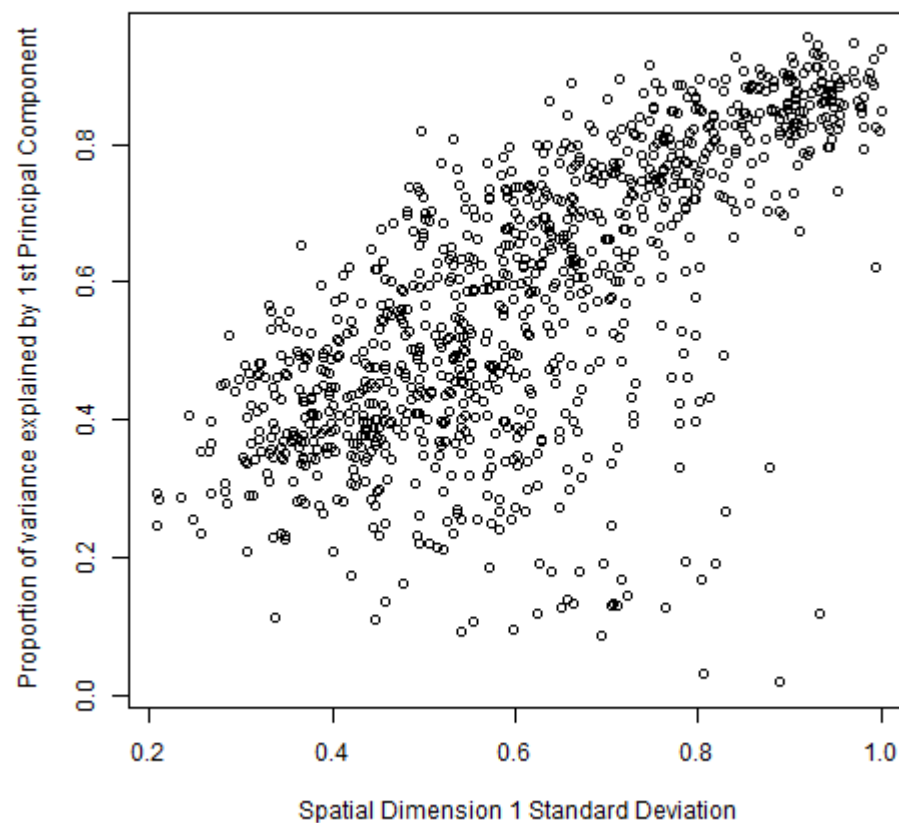


Figure 11: Values of  $z_0$  vs proportion of variance explained by the first principal component

## A FSharp Sourcecode

## B R Sourcecode

```
#results.df <- read.table("C:/Users/Michael/Documents/Visual Studio 11/
  Projects/CS380/HypothesisTesting/bin/Debug/results2/pd_spatial2_
  results.csv", sep = ",", skip = 0, header=T)
results.df <- read.table("C:/Users/Michael/Documents/Visual Studio 11/
  Projects/CS380/HypothesisTesting/bin/Debug/results4/spatial2stv.csv"
  , sep = ",", skip = 0, header=T)
#results.df <- read.table("C:/Users/Michael/Documents/Visual Studio 11/
  Projects/CS380/HypothesisTesting/bin/Debug/results5/urnstv.csv", sep
  = ",", skip = 0, header=T)
results.df$X <- NULL
results.df[1:5,]

results.df$fairness = results.df$gallagher;
#results.df$stability = results.df$enp-1;
results.df$stability = (1/results.df$governability)-1;

windows()
plot.new()
results.df$col1 = "blue"
results.df$col1[results.df$dm<10] = "green"
results.df$col1[results.df$dm==1] = "red"
rOrder = order(rnorm(nrow(results.df)))
plot(results.df$fairness[rOrder], results.df$stability[rOrder], xlab="
  Unfairness", ylab="Instability", col=results.df$col1[rOrder])
results.df$col1 <- NULL

#Fit fairness
windows()
plot.new()
plot(results.df$dm, results.df$fairness)

windows()
plot.new()
plot(results.df$dm, log(1/results.df$fairness))

fairness.fit = lm((1/fairness)~I(dm)+I(dm^2), data=results.df)

summary(fairness.fit)

windows()
plot.new()
plot(predict(fairness.fit, data=results.df), resid(fairness.fit, data=
  results.df))
```

```

#Fit stability
windows()
plot.new()
plot(results.df$dm, results.df$stability)

stability.fit = lm(stability~I(dm)+I(log(dm))+I(dm^2)+I(dm^3), data=
  results.df)

summary(stability.fit)

windows()
plot.new()
plot(predict(stability.fit, data=results.df), resid(stability.fit, data=
  results.df))

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

combined.df = dummy.df;
combined.df = rbind(dummy.df, data.frame(fairness=results.df$fairness,
  stability=results.df$stability, dm=results.df$dm, isPredicted = FALSE
), all.x=T);

combined2.df = combined.df[combined.df$dm>=1 & combined.df$dm<=8,];
combined2.df = combined2.df[order(combined2.df$isPredicted),]

windows()
plot.new()
plot(combined2.df$dm, combined2.df$stability, xlab="District_Magnitude",
  , ylab="Unstability", col=(c("red", "blue"))[combined2.df$
  isPredicted + 1])

windows()
plot.new()
plot(combined2.df$dm, combined2.df$fairness, xlab="District_Magnitude",
  ylab="Unfairness", col=(c("red", "blue"))[combined2.df$isPredicted
  + 1])

windows()
plot.new()
plot(combined2.df$fairness, combined2.df$stability, xlab="Unfairness",

```

```

      ylab="Unstability", main="Predicted_Fairness_vs._Stability", col=(c(
"red", "blue"))[combined2.df$isPredicted + 1])

dummy.df$col1 = "blue"
dummy.df$col1[dummy.df$dm<10] = "green"
dummy.df$col1[dummy.df$dm==1] = "red"

windows()
plot.new()
plot(dummy.df$fairness, dummy.df$stability, xlab="Unfairness", ylab="
Unstability", main="Predicted_Fairness_vs._Stability", col=dummy.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$col1 = "blue"
actualmeans.df$col1[actualmeans.df$dm<10] = "green"
actualmeans.df$col1[actualmeans.df$dm==1] = "red"

windows()
plot.new()
plot(actualmeans.df$fairness, actualmeans.df$stability, xlab="
Unfairness", ylab="Unstability", main="Actual_Mean_Fairness_vs._
Stability", col=actualmeans.df$col1)

# 2nd regression attempt. Based on means per DM.
fairness2.fit = lm((1/fairness)~I(dm)+I(dm^2)+I(log(dm)), data=
actualmeans.df)
stability2.fit = lm(stability~I(dm)+I(log(dm))+I(dm^2), data=
actualmeans.df)

summary(fairness2.fit)

```

```

summary(stability2.fit)

dummy2.df = data.frame(dm=(10:1200)/10)
dummy2.df$fairness = 1/(predict(fairness2.fit, dummy2.df))
dummy2.df$stability = predict(stability2.fit, dummy2.df)
dummy2.df$type=2
dummy2.df$isPredicted=TRUE

windows()
plot.new()
plot(dummy2.df$fairness, dummy2.df$stability, xlab="Unfairness", ylab="
  Unstability", main="Predicted [2] _Fairness_vs._Stability")

combined3.df = dummy2.df
#combined3.df = rbind(combined3.df, dummy.df, all.x=T)
combined3.df = combined3.df[order(rnorm(nrow(combined3.df))),]
combined3.df = rbind(combined3.df, actualmeans.df, all.x=T)
combined3.df$color = (c("black", "red", "green"))[combined3.df$type+1]

#reorder
#combined3.df = combined3.df[order(rnorm(nrow(combined3.df))),]
plot(combined3.df$fairness, combined3.df$stability, xlab="Unfairness",
  ylab="Unstability", main="Combined_Fairness_vs._Stability", col=
  combined3.df$color)

plot(combined3.df$dm, combined3.df$stability, xlab="District_Magnitude"
  , ylab="Unstability", main="District_Magnitude_vs._Stability", col=
  combined3.df$color)

```