A Geometric Approach to Voting

by

Lee Fisher

Honors Thesis

Appalachian State University

Submitted to the Department of Mathematical Sciences
in partial fulfillment of the requirements for the degree of

Bachelor of Science

May 2016

Approved by:

Vicky Klima, Ph.D., Thesis Director,
Mathematical Sciences Honors Director

Mark Ginn, Ph.D., Chair, Mathematical Sciences

Abstract

This paper proposes an intuitive extension of positional weighted voting systems. A positional weighted voting system requires voters to submit fully ranked ballots and assigns point values to a voters preferences for each rank. The voters' rankings are sorted into the entries of a matrix called the preference matrix which when multiplied by a vector called the weights vector returns the number of points each candidate receives. Choosing different reasonable weights can lead to different outcomes in an election. The paper proposes a voting system in which the winner of an election is the candidate who would win over the greatest proportion of distinct reasonable weights vector choices. In the paper we conclude by applying the new method to the 2010 San Francisco district representative election.

1 Introduction

In 1998 Donald Saari and Fabrice Valognes [2] pioneered the formal study of the mathematics of voting theory. Donald Saari's work has continued through a series of other papers into the present day. In [2] they introduce the idea of the positional ranked voting system, restricting their attention to three of six possible profiles for the three candidate case. They point out many symmetries and nice properties of the Borda Count election system. Donald Saari continues his work in [1], he extends results from [2] to consider all six profiles for the three candidate case. He points out flaws and paradoxes involved in picking the Condorcet winner extending results about the Borda Count naturally to all six profiles. Later, Daughtery et al. [3] use representation theory to make statements about three candidate elections and positional ranked voting systems. In [3] they also point out the importance of using the sum-zero subspace to simplify calculations. Kent Vashaw [7] in his senior honors thesis translates many of the results in [3] into results in linear algebra. In this paper we formalize, extend, and apply a system proposed at the end of [7].

2 Positonal Weighted Voting

Anna Anderson (A), Bart Baxter (B) , and Claire Clemmons (C) are each running for mayor of the town of Smallsbury. Smallsbury has a tiny but active voter population, and this year the election seems close. The Smallsbury administration is worried about the possibility of a three way tie, so half as a precaution and half out of pure curiosity they ask the voters to list their candidates in order of preference as opposed to simply

selecting their first choice when they go to vote. On the election day 260 people cast ballots. Each voter picks one out of $3! = 6$ possible rankings of candidates. The votes are tallied and sorted as follows:

$$A > B > C : 50$$

$$A > C > B : 60$$

$$B > A > C : 20$$

$$B > C > A : 70$$

$$C > A > B : 20$$

$$C > B > A : 40.$$

The first entry tells us that 50 voters had the preference ranking: Anna first, Bart second, and Claire third. The second entry in the list tells us that 60 voters had the preference Anna first, Claire second, and Bart third. The fully ranked profiles allow us to run the classical voting method just as easily as always. The first and second entries tell us how many voters select Anna as their first choice, the third and fourth entries tell us how many voters chose Bart as their first choice, and the last two tell us how many voters chose Claire as their first choice. When we add the respective entries together we get: Anna with 110 votes, Bart with 90 votes, and Claire with 60 votes. In the end, Anna seizes victory just barely from Bart and with little contest from Claire. The story does not end here.

The curious tallier for the Smallsbury administration studies the fully ranked profiles

and notices that although Anna had won with 110 votes, in the end an equal number of people chose Anna as their last place choice. The tallier decides to rank the candidates by the number of people who chose them for last place. In this way, Anna gets 110 last place votes, Bart gets 80 last place votes, and Claire gets 70 last place votes. If the Smallsbury administration had opted to choose the least disliked person instead of the most liked person for office, Claire, who came in last, would have won!

The curious tallier grows more curious. The tallier wonders: "If I subtract the number of last place votes from the number of first place votes, then who will win?" The idea made sense to the tallier; a last place vote should count against you as much as a first place vote counts for you. In this way Anna gets $110 - 110 = 0$ votes, Bart gets $90 - 80 = 10$, and Claire gets $60 - 70 = -10$ votes. If we run the election this way then Bart will win.

So in the end, who really wins? The first method, the plurality method, will choose the candidate liked by the greatest proportion of voters, the second method, the antiplurality method, will choose the candidate disliked by the smallest proportion of people, and the third method, the Borda Count, will, intuitively, choose the candidate most liked on "average". All three of the voting systems seem reasonable, but could it be that one follows the spirit of democracy better than the others? The voting systems are all actually just special cases of positional weighted voting systems.

We start our explanation of positional weighted voting systems by reorganizing the data. An election with many more than three candidates will have an immensely long list of possible rankings. For example, in a six candidate election the list would be 720 entries long. The preference matrix presents the data in a more managable way.

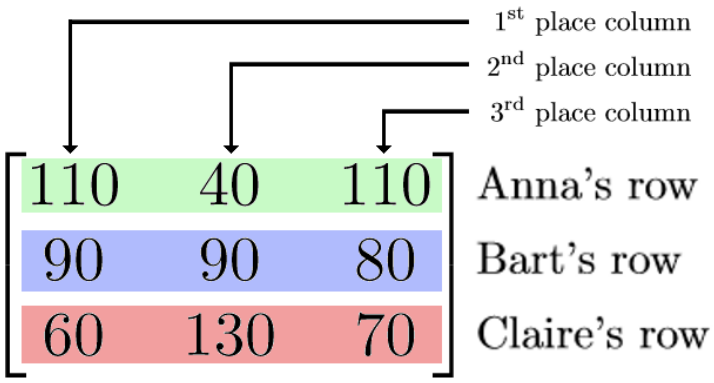

Figure 1: The preference matrix for the Smallsbury election

Figure 1 gives the preference matrix for the Smallsbury election. The top left corner of the matrix contains the sum of the first two numbers in the earlier list. That entry is the number of voters who picked Anna as their first choice. Likewise, the top middle entry is the number of voters who picked Anna as their second place choice. So all together the columns from left to right of the preference matrix denote the rank of the preference and the rows denote the candidate of preference. One more example: the bottom middle entry would be the number of people who chose Claire as their second choice.

We can revisit the results of the curious tallier by selecting different weights vectors. The entries of a weights vector tell us the value of a first, second, or third place vote. When we multiply the preference matrix by the weight $[1, 0, 0]^T$ we are using the plurality method. This weight tells us that first place choices are worth one point and the other places are worth nothing. The product of the preference matrix with the weight yields the results vector. The largest entry in the results vector tells us the winner:

$$\begin{bmatrix} 110 & 40 & 110 \\ 90 & 90 & 80 \\ 60 & 130 & 70 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = (1) \begin{bmatrix} 110 \\ 90 \\ 60 \end{bmatrix} + (0) \begin{bmatrix} 40 \\ 90 \\ 130 \end{bmatrix} + (0) \begin{bmatrix} 110 \\ 80 \\ 70 \end{bmatrix} = \begin{bmatrix} 110 \\ 90 \\ 60 \end{bmatrix}. \quad (1)$$

Anna won by the plurality method before, this confirms that the weight choice is no different from the more casual context.

Next we will consider the weights $[0, 0, -1]^T$ and $[1, 0, -1]^T$:

$$\begin{bmatrix} 110 & 40 & 110 \\ 90 & 90 & 80 \\ 60 & 130 & 70 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = (0) \begin{bmatrix} 110 \\ 90 \\ 60 \end{bmatrix} + (0) \begin{bmatrix} 40 \\ 90 \\ 130 \end{bmatrix} + (-1) \begin{bmatrix} 110 \\ 80 \\ 70 \end{bmatrix} = \begin{bmatrix} -110 \\ -80 \\ -70 \end{bmatrix} \quad (2)$$

and

$$\begin{bmatrix} 110 & 40 & 110 \\ 90 & 90 & 80 \\ 60 & 130 & 70 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = (1) \begin{bmatrix} 110 \\ 90 \\ 60 \end{bmatrix} + (0) \begin{bmatrix} 40 \\ 90 \\ 130 \end{bmatrix} + (-1) \begin{bmatrix} 110 \\ 80 \\ 70 \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \\ -10 \end{bmatrix}. \quad (3)$$

Equation 2 shows the antiplurality results from before. In the same manner Equation 3 shows the Borda count results from before.

Any three real numbers will give us a weight, although that weight may not necessarily be reasonable. We say that all reasonable weights have non-decreasing entries. For instance, a weight such as $[0, 0, 1]^T$, which gives one point to a last place vote and no points to the other places, is unreasonable. Such a vector will elect a candidate that the

greatest number of people dislike. However, the plurality, antiplurality, and Borda count weights are all reasonable weights. At this point we take a detour to formalize much of the terminology.

3 The Linear Algebra of Voting

Consider an election with n candidates and V voters. The *preference matrix* is an $n \times n$ matrix with natural number entries; the entry in the i^{th} row and j^{th} column is the number of people who selected the i^{th} candidate for j^{th} place. Any vector in $\mathbb{R}^n$ can be a *weight*; however a weight is *reasonable* if it has non-increasing entries. We call the set of all reasonable weights the reasonable region. The *results vector* is the product of the preference matrix with the weight. If the largest value of the results vector occurs its j^{th} entry, then the j^{th} candidate wins the election with that particular weight. A candidate's *winning region* is the set of all reasonable weights that will elect that particular candidate. We can quickly observe that the rowsums and columnsums of a preference matrix are all equal to V. This is because the total number of people who choose candidate i for any place must be the total number of people who voted, and the total number of people who pick any candidate for j^{th} place must also be the total number of people who voted. Equation 4 gives the setup for the n-candidate case;

$$\begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix}. \tag{4}$$

Multiplying a weight, $\vec{w}$, by a positive scalar will not change the winner or the election results as a ranking. A fixed preference matrix, M and a weight $\vec{w}$ yield the result $M\vec{w} = [P_1, P_2, \cdots, P_n]^T$. If we multiply $\vec{w}$ by the positive scalar a we obtain the new results vector $M(a\vec{w}) = [aP_1, aP_2, \ldots aP_n]$. If $P_1 \geq P_2 \geq \cdots \geq P_n$ then likewise $aP_1 \geq aP_2 \geq \cdots \geq aP_n$. Since any two weights that are positive multiples of each other will give the same outcome, we can limit ourselves to only picking weights of the same length. We propose one as the most convenient length for our purposes.

Adding multiples of the all ones vector to a weight will not change an elections result either. Since all rows and columns of the preference matrix sum to the number of voters, shifting a weight by a constant will shift the results vector by a constant without changing the actual ranking. In symbols the justification is straightforward:

$$\text{If } M\vec{w} = \vec{P} \text{ then } M(\vec{w} + a\vec{1}) = \vec{P} + aM\vec{1} = \vec{P} + av\vec{1}.$$

This observation leads us to focus our attention on a collection of vectors orthogonal to the all ones vector. The subspace of vectors that is orthogonal to the all ones vector and also contains the zero vector is called the sum-zero subspace. The name is well fitting because all vectors in this subspace have entries that sum to zero.

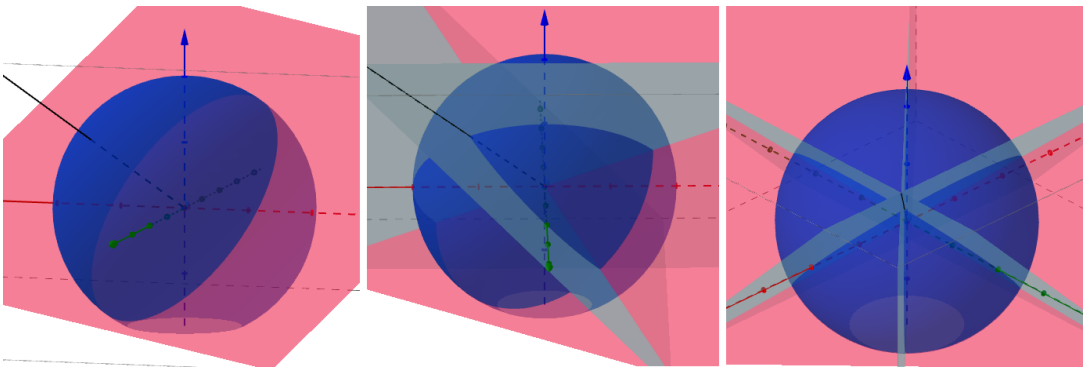

Figure 2: The unit sphere, all ones vector, sum-zero subspace, and reasonable regions

Figure 2 gives three different views of the terms we defined for the three candidate case. The image on the left shows a picture of the unit sphere in blue, the all ones vector in black, and the sum zero subspace in red. The images on the right and middle also show the planes $y = x$, $y = z$, and $z = x$ all in grey. The choice of axes corresponding to first, second, and third place weights determines one of the six symmetric regions to be the reasonable region. The image in the middle shows one of the regions and the image on the right highlights the symmetry of the different possible reasonable regions. A preference matrix induces a partition of $\mathbb{R}^n$ into winning regions for each of the candidates, thus it also induces a partition of the reasonable region, sum-zero subspace, unit sphere, and the intersection of all three.

We require one more important result for this election machinery. Because two candidates may tie for first place, the winning regions do not completely partition $\mathbb{R}^n$. However, they almost partition the entire space. The set of all points where any two candidates tie for first place is either a subset of dimension strictly less than n or it a piece of $\mathbb{R}^n$ of dimension equal to n. We can prove that if the preference matrix is invertible then the second possibility will not happen.

Theorem 1. *If the preference matrix in an n-candidate election is invertible then the region in which the candidates tie for first and second place is of dimension less than n.*

Proof. We will prove the contrapositive. Take $\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_n$ to be a set of n linearly independent weights for which the same two candidates tie for first place. Without loss of generality we say candidates one and two tie for first place for each choice of $\vec{w}_i$. We call the preference matrix M, and we call the matrix whose columns are the weights $\vec{w}_i$, W. The product MW is an $n \times n$ matrix whose top two rows are exactly the same. Since we assumed the $\vec{w}_i$ were linearly independent, W is invertible. Therefore M must not be invertible and we are done. □

Although we cannot prove our conjecture, we think that even in the non-invertible case tieing regions are not likely to have dimension n.

4 The Total Reasonable Region Winner

With some new terminology in hand, we can now describe a new voting procedure. We use the previous observations to ensure that the winner will be the candidate who would win with the greatest proportion of distinct reasonable weights choices. The *Total Reasonable Region winner*, or the TRR winner, is the candidate whose winning region intersected with the surface of the unit $(n-1)$-sphere and the reasonable region has the greatest volume. Intuitively, the TRR winner is the winner most likely to win given a random, reasonable weight. From this point on we propose two methods for computing the TRR winner.

The first and most straightforward way to compute the TRR winner is by a Monte-Carlo method. We wrote a program which picks a large number of vectors whose entries are normally distributed with mean zero and variance one. Then the program divides all the vectors by their Euclidean norm. The resulting vectors are uniformly distributed on the surface of an $(n-1)$-sphere. This procedure is a consequence of a result in [6](page 24) that was formalized in [4] and [5]. For a justification, consider a collection of n indenpendent normally distributed random variables each with mean zero and variance one. This probability distribution in Equation 5 depends only on the length of the vector and not on any $(n-1)$-spherical angles;

$$f(\vec{x}) = \frac{1}{\sqrt{(2\pi)^n}} e^{-\frac{1}{2}(x_1^2+x_2^2+\cdots+x_n^2)} = \frac{1}{\sqrt{(2\pi)^n}} e^{-\frac{1}{2}|\vec{x}|^2}. \tag{5}$$

Thus, if vectors with idependent identically distributed normal variable entries with mean zero are scaled to lie on the unit sphere then they will lie uniformly on the sphere. We then sort the entries of the vectors to lie in the reasonable region. This does not change their distribution. In this way we pick an arbitrary number of points on the surface of the unit $(n-1)$-sphere intersect the reasonable region.

In the three candidate case, we can find the TRR winner analytically. We begin by reconsidering our observation about the sum-zero subspace. When we restrict our attention to this subspace, the TRR winner and proportions do not change at all. The sum zero subspace for three candidates is only a plane and then measuring and graphing the candidates winning regions is feasible to do by hand. However, in larger dimensions reducing to the sum-zero subspace only marginally reduces the computation complexity.

In the observation we only gave an informal justification before proceding we now provide a formal proof.

Theorem 2. *Projection onto the sum-zero subspace preserves rankings in* $\mathbb{R}^3$.

Proof. The projection matrix for the sum-zero vector subspace is

$$T \equiv \begin{bmatrix} 2/3 & -1/3 & -1/3 \\ -1/3 & 2/3 & -1/3 \\ -1/3 & -1/3 & 2/3 \end{bmatrix}.$$

If $\vec{w}$ is a choice of weights in $\mathbb{R}^3$ then $T\vec{w}$ is a choice of weights whose vectors sum to 0. Consider a ranking $A\vec{w} = R$ our goal is to show that going to $A(T\vec{w}) = R'$ preserves the ordering on R. Call V the number of voters. The points awarded to a candidate in a positionally weighted voting system are:

$$P_A = a_{11}w_1 + a_{12}w_2 + a_{13}w_3 = a_{11}(w_1 - w_3) + a_{12}(w_2 - w_3) + w_3V,$$

$$P_B = a_{21}w_1 + a_{22}w_2 + a_{23}w_3 = a_{21}(w_1 - w_3) + a_{22}(w_2 - w_3) + w_3V,$$

$$P_C = a_{31}w_1 + a_{32}w_2 + a_{33}w_3 = a_{31}(w_1 - w_3) + a_{32}(w_2 - w_3) + w_3V.$$

Observe the effect of sending w_1 to $(T\vec{w})_1$, w_2 to $(T\vec{w})_2$, and w_3 to $(T\vec{w})_3$. This

means

$$w_1 \to {}^{2w_1}/_3 - {}^{w_2}/_3 - {}^{w_3}/_3,$$

$$w_2 \to {}^{2w_2}/_3 - {}^{w_1}/_3 - {}^{w_3}/_3,$$

$$w_3 \to {}^{2w_3}/_3 - {}^{w_1}/_3 - {}^{w_2}/_3.$$

When we make this substitution into the equations for points each candidate we need only observe that $T(\vec{w})_1 - T(\vec{w})_3 = w_1 - w_3$ and that $T(\vec{w})_2 - T(\vec{w})_3 = w_2 - w_3$. To see that the linear projection of the weights will only change the coefficient on V. However since the coefficients on V in all three equations are all equal, the linear projection is only a shift in the rankings of each candidate, so we have proven the theorem.

□

Theorem 3. *A weight is reasonable if and only if its projection onto the sum zero subspace is reasonable.*

Proof. We will show that a weight $\vec{w}$ is reasonable if and only if $T(\vec{w})$ is reasonable.

Suppose $\vec{w}$ is reasonable, this means $w_1 \geq w_2 \geq w_3$. Now consider

$$T(\vec{w}) = \begin{bmatrix} {}^{2w_1}/_3 - {}^{w_2}/_3 - {}^{w_3}/_3 \\ -{}^{w_1}/_3 + {}^{2w_2}/_3 - {}^{w_3}/_3 \\ -{}^{w_1}/_3 - {}^{w_2}/_3 + {}^{2w_3}/_3 \end{bmatrix}.$$

Clearly $w_1 \geq w_2 \geq w_3$ if and only if $w_1 \geq w_2$ and $w_2 \geq w_3$.

$$
\begin{aligned}
\text{And, } w_1 \geq w_2 &\iff \frac{2w_1}{3} - \frac{w_3}{3} \geq \frac{2w_2}{3} - \frac{w_3}{3} \\
&\iff \frac{2w_1}{3} - \frac{w_3}{3} - \frac{w_2}{3} \geq \frac{2w_2}{3} - \frac{w_3}{3} - \frac{w_1}{3} \\
&\iff T(\vec{w})_1 \geq T(\vec{w})_2. \\
\text{Also, } w_2 \geq w_3 &\iff \frac{2w_2}{3} - \frac{w_1}{3} \geq \frac{2w_3}{3} - \frac{w_1}{3} \\
&\iff \frac{2w_2}{3} - \frac{w_1}{3} - \frac{w_3}{3} \geq \frac{2w_3}{3} - \frac{w_1}{3} - \frac{w_2}{3} \\
&\iff T(\vec{w})_2 \geq T(\vec{w})_3.
\end{aligned}
$$

Therefore, $w_1 \geq w_2 \geq w_3$ if and only if $T(\vec{w})_1 \geq T(\vec{w})_2 \geq T(\vec{w})_3$.

□

The sum zero subspace passes through the origin. Winning regions extend orthogonally from the sum-zero subspace in both directions. If we measure the candidates winning regions on the reasonable sum-zero subspace intersect the unit sphere then the winning regions will lie in the same proportions as they would in the higher-dimensional space.

5 Back to Smallsbury

Using the result about the sum zero subspace, we can find an analytical solution to the Smallsbury election paradox. Recall the equation governing the Smallsbury election:

$$\begin{bmatrix} 110 & 40 & 110 \\ 90 & 90 & 80 \\ 60 & 130 & 70 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} P_A \\ P_B \\ P_C \end{bmatrix}. \tag{6}$$

We will start by picking an orthonormal basis for the sum zero subspace in $\mathbb{R}^3$. We pick the basis $\vec{u} = [\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}, -\frac{2}{\sqrt{6}}]^T$ and $\vec{v} = [\frac{2}{\sqrt{2}}, -\frac{2}{\sqrt{2}}, 0]$. So any vector in the subspace can be written as a linear combination and $\vec{u}$ and $\vec{v}$. That is, $\vec{w} = a\vec{u} + b\vec{v}$ for some $a, b \in \mathbb{R}$. A particular choice will be reasonable if $\frac{a}{\sqrt{6}} + \frac{b}{\sqrt{2}} \geq \frac{a}{\sqrt{6}} - \frac{b}{\sqrt{2}} \geq \frac{-2a}{\sqrt{6}}$ which is true if and only if $\sqrt{3}a \geq b \geq 0$. Since we have made the suitable change of basis, we can restrict to the unit sphere by saying: $\sqrt{a^2 + b^2} \leq 1$ Then using our sum zero weights with Equation 6 yields

$$\begin{bmatrix} 110 & 40 & 110 \\ 90 & 90 & 80 \\ 60 & 130 & 70 \end{bmatrix} \begin{bmatrix} \frac{a}{\sqrt{6}} + \frac{b}{\sqrt{2}} \\ \frac{a}{\sqrt{6}} - \frac{b}{\sqrt{2}} \\ \frac{-2a}{\sqrt{6}} \end{bmatrix} = \begin{bmatrix} \frac{-35\sqrt{6}a}{3} + 35\sqrt{2}b \\ \frac{10\sqrt{6}a}{3} \\ \frac{25\sqrt{6}a}{3} - 35\sqrt{2}b \end{bmatrix}. \tag{7}$$

If we want to determine the winning regions we can start by determining the tieing regions. Setting any two entries of the results vector from Equation 7 equal to each other will give us a tieing line. The tieing lines form the boundaries for each candidate's winning region. Graphs of those lines in terms of a and b will tell the rest of the story.

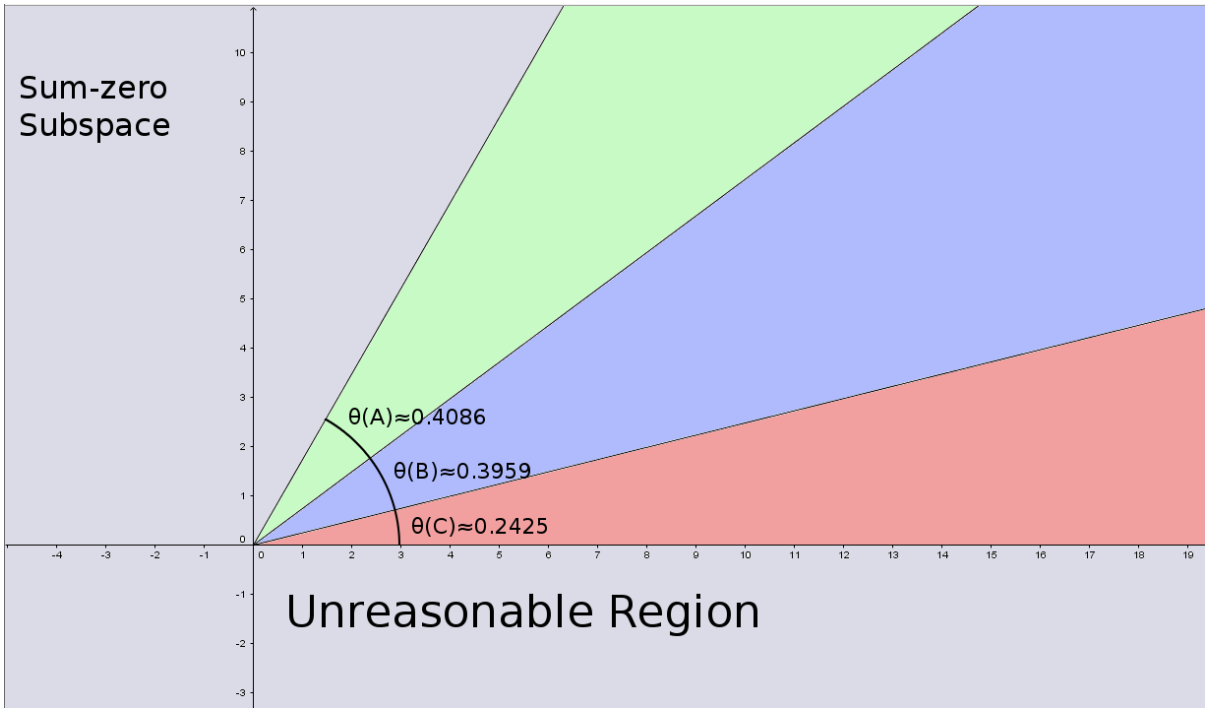

Figure 3: In this case $A \succ B \succ C$

Figure 3 shows the picture of the winning regions with choices of a on the horizontal axis and choices of b on the vertical axis.

In this case we find out that Anne is the TRR winner, but just by a hair. The reasonable regions encopasses an angle of $\frac{\pi}{3}$. Thus Anne gets 39.02% of the reasonable region, Bart gets 37.81% of the region and Clare gets 23.16% of the region. For comparison we run a Monte-Carlo method to randomly select points on the reasonable unit sphere, using $1,000,000$ points Anne gets $390,245$, Bart gets $377,881$ and Clare gets $231,1847$. The percentages for the Monte Carlo method are: Anne with 39.025%, Bart with 37.788%, and Clare with 23.185%, which is almost exactly what the analytic method gave as a result. We now procede to try and find the TRR winner from a real world vote.

6 San Francisco

The city of San Fracisco uses a choice ranked voting system for district representative elections. When residents vote for positions in the San Francisco administration they select and rank their top three choices. The data for the district 10 representative election in 2014 was available and easy to analyze [8]. The race was between five, arguably six, candidates: Shawn M. Richard, Malia Cohen, Ed Donaldson, Marlene Tran, Tony Kelly, and the enigmatic candidate zero: write-ins or no preference.

Handling write-ins was the first hurdle we needed to overcome to be able to find the TRR winner. The ballot data did not distinguish between different write-ins or reveal what the voters wrote. Since it was unfortunately impossible for the write-ins to win we had to figure out a way to handle the data. In our first approach, we aimed to preserve the structure that fully ranked ballots would give to a preference matrix. When we used this approach we just threw out all the votes that had listed write in or no preference somewhere on the ballot. Before scrubbing the data for partially ranked ballots there were 8,171 votes, and afterwards there were 7,085 votes. So 1,086 votes were lost to scrubbing. The incomplete preference matrix was as follows

$$\begin{bmatrix} 2203 & 2193 & 2291 & ? & ? \\ 1587 & 1599 & 1569 & ? & ? \\ 1360 & 1309 & 1329 & ? & ? \\ 1030 & 1089 & 1044 & ? & ? \\ 905 & 895 & 852 & ? & ? \end{bmatrix}.$$

The rows from top to bottom are for candidates: Malia Cohen, Tony Kelly, Marlene Tran, Shawn M. Richard, and Ed Donaldson. Since we only know the voters' top three choices and not their fully ranked preferences we do not directly know the entries of the last two columns of the preference matrix. We wanted to continue to enforce the restriction of having all row and column sums equal. However, this restriction alone is not enough to infer the missing entries. We assumed that the last two columns have equal entries. Intuitively, we made the assumption that voters on the aggregate are just as likely to pick any one candidate for 4^{th} place as they are to pick that candidate for 5^{th} place. Making this assumption allows us to fill in the rest of the matrix

$$\begin{bmatrix} 2203 & 2193 & 2291 & ? & ? \\ 1587 & 1599 & 1569 & ? & ? \\ 1360 & 1309 & 1329 & ? & ? \\ 1030 & 1089 & 1944 & ? & ? \\ 905 & 895 & 852 & ? & ? \end{bmatrix} \rightarrow \begin{bmatrix} 2203 & 2193 & 2291 & 199 & 199 \\ 1587 & 1599 & 1569 & 1165 & 1165 \\ 1360 & 1309 & 1329 & 1543 & 1543 \\ 1030 & 1089 & 1044 & 1961 & 1961 \\ 905 & 895 & 852 & 2216 & 2216 \end{bmatrix}.$$

We also chose to round down any fractional entries. The adjusted preference matrix now has the property that all rows and columns sum to nearly 7,085. The matrix is clearly not invertible, but having a sigular preference matrix was not problematic. We used a python program (see the appendix) that employs a Monte-Carlo algorithm to compute the proportions of the winning regions. When we ran the program on the generated matrix we use one million points in the calculation. Malia Cohen won all one million randomly selected reasonable weights, which agrees with what happened in the San Francisco election system. Malia Cohen is the current district 10 representative for

the San Fracisco city council. Even though the preference matrix makes it seem like it is at least a little close, in terms of the TRR winner Malia Cohen wins by a landslide.

Using the partial data in our preference matrix would force the preference matrix to have differing row and column sums. However, we only used this fact when we restricted to the sum zero subpace. We did not use any of the machinery of the sum-zero subspace to compute our winner with the Monte-Carlo method. This time we will keep all the incomplete ballots. If a voter only picks a first place choice we will add that vote to that candidates first place entries in the preference matrix and not add anything to any other entries in the preference matrix. With this philosophy we also make the last two columns of the preference matrix $\vec{0}$. After reconsidering the data the new preference matrix is as follows;

$$\begin{bmatrix} 3190 & 3274 & 3229 & 0 & 0 \\ 2210 & 2204 & 2202 & 0 & 0 \\ 1727 & 1703 & 1721 & 0 & 0 \\ 1174 & 1214 & 1241 & 0 & 0 \\ 1067 & 1023 & 1045 & 0 & 0 \end{bmatrix}.$$

From top to bottom the candidate ordering for the rows is still the same: Malia Cohen, Tony Kelly, Marlene Tran, Shawn M. Richard, and Ed Donaldson. This time the results are different. With one million random weight choices Malia Cohen wins 864,479, Ed Donaldson wins 132,532, and Tony Kelly wins 2,989 of them. The other two candidates get no wins. It seems odd that when we choose to zero out the last two columns, Ed Donaldson, the most obscure candidate, comes in second place. This

seems unintuitive, but voting methods similar to the anti-plurality method will favor Ed Donaldson greatly because his obscurity implies that he is not strongly disliked.

The district 10 San Francisco city council election was not a very close election, Malia Cohen wins in pretty much any reasonable way of interpreting the ballots. While infrequent, closer elections can exhibit strange Smallsbury like paradoxes.

7 Future Work

In summary, we have explained the idea of a positional weighted voting system and what it means for such a system to be reasonable. We have introduced the idea of the TRR election system, based on the philosophy of trying all possible reasonable weights and selecting the winner as the one who would win the most. We used symmetry arguments and computer programs to make finding this winner a tractable problem. Yet many questions remain unanswered. Ealier in the paper we proved that the tieing regions would have dimension less than n if the preference matrix is invertible. We know a necessary condition to ensure trivial tieing regions, but we do not know the sufficient conditions and we do not know how likely it is that these conditions are met. We also struggled with ambiguity on how to extend the TRR system to partially ranked profiles. We do not know what the most natural extension of the TRR system is to this kind of data or if one possible extension is more natural than others.

References

[1] Donald G. Saari, *Explaining all three-alternative voting outcomes*, J. Econom. Theory **87** (1999), no. 2, 313–355.

[2] Donald G. Saari and Fabrice Valognes, *Geometry, voting, and paradoxes*, Math. Mag. **71** (1998), no. 4, 243–259.

[3] Zajj Daugherty, Alexander K. Eustis, Gregory Minton, and Michael E. Orrison, *Voting, the symmetric group, and representation theory*, Amer. Math. Monthly **116** (2009), no. 8, 667–687.

[4] George Marsaglia, *Choosing a Point from the Surface of a Sphere*, Ann. Math. Statist. **43** (1972), no. 2, 645–646.

[5] Mervin E. Muller, *A Note on a Method for Generating Points Uniformly on N-dimensional Spheres*, Commun. ACM **2** (1959), no. 4, 19–20.

[6] Harald Cramér, *Mathematical Methods in Statistics*, H. Princeton University Press, Princeton, N. J., 1946.

[7] Kent Vashaw, *Positional Weighted Voting and Linear Algebra*, `http://mathsci.appstate.edu/students/honors-program/completed-honors-theses/spring-2014-theses`. Accessed May 6, 2016.

[8] City and County of San Francisco Department of Elections, *November 4, 2014 Official Elections Results*, `http://www.sfelections.org/results/20141104/data/D10.zip`. Accessed May 8, 2016.

Appendix

```
import numpy as np

import numpy.ma as ma


#This takes a text file from the interpreter and computes a good estimate of

#the projected complete preference matrix.

def createPreferenceMatrix(f):


   #The Rankings vector simply puts the data from the interpreter output into

   #a 2N by 2N matrix, where N is the number of candidates.

   Rankings = [[], [], [], [], []]

   for line in f:

      Rankings[0] = f.readline().replace(':',' ').split()

      Rankings[1] = f.readline().replace(':',' ').split()

      Rankings[2] = f.readline().replace(':',' ').split()

      Rankings[3] = f.readline().replace(':',' ').split()

      Rankings[4] = f.readline().replace(':',' ').split()



   #This computes the number of candidates as half the number of entries in

   #one column of Rankings Vector

   CandidateNumber = int(len(Rankings[0])/2)
```

```
#This creates a square matrix that will be the completed preference
#matrix.at the end of the upcoming loops
Matrix = np.zeros((CandidateNumber,CandidateNumber), dtype=np.int)

#Lists the first place vote numbers according to how they're ordered to
#make everything fit right
for i in range(CandidateNumber):
   Matrix[0][i] = Rankings[0][2*i+1]

#This fills in the next two columns in an order matching the order for the
#first place column of the preference matrix
for i in range(CandidateNumber):
   j = 0
   while(Rankings[0][2*i] != Rankings[1][2*j]):
      j = j + 1
   Matrix[1][i] = Rankings[1][2*j+1]

   k = 0
   while(Rankings[0][2*i] != Rankings[2][2*k]):
      k = k + 1
   Matrix[2][i] = Rankings[2][2*k+1]

#This computes the number of voters; the sum of one column of the entries
#of the candidate matrix.
```

```
    VotingNumber = 0
    for i in range(CandidateNumber):
        VotingNumber += Matrix[0][i]



    #This fills in the rest of the preference matrix under the assumption that
    #voters are equally likely to swap lower ranked choices. That all votes
    #for ranks less than three are equal. It also trunucates fractional
    #entries by casting as an int. The estimated preference matrix will almost
    #obey the rule that column and rowsums are equal to the number of voters.
    #It's commented out right now, uncomment it to make the adjustment.

    #for i in range(CandidateNumber - 3):
    #  for j in range(CandidateNumber):
    #     Matrix[i+3][j] = int((VotingNumber - Matrix[0][j]-Matrix[1][j]-
    #     Matrix[2][j])/(CandidateNumber - 3))

    print(Matrix)
    return Matrix


#This will take the preference matrix, as well as a user given input on how
#many points to use in the procedure. It will pick that many points within the
#unit n-cube, and then mask by the restricting to lie in the unit n-sphere,
#and restricting them to have reasonable entries. With the remaining points it
```

```
#will run the election with each choice of weight and tally the number of wins
#for each candidate. The function will return this tally.
def RunElection(preferences, PointNum):

   #makes the dimension of the n-cube
   Dimension = len(preferences)

   #Makes tons of data points normally distributed with mean 0 and variance 1
   Data = np.random.normal(0.0,1.0,(PointNum,Dimension))

   #Scales everything to lie on the unit n-sphere and then sorts each weight
   #to be in the reasonable region.
   for i in range(PointNum):
      k = np.sqrt(np.sum(np.square(Data[i])))
      for j in range(Dimension):
         Data[i,j] = Data[i,j]/k
      Data[i] = sorted(Data[i], reverse = True)
   print("\n")

   #Multiplies the preference matrix with the random data and records
   #the results. Don't mess with this!! Science says it is right.
   Results = np.zeros((PointNum,Dimension), dtype=np.float)
   for i in range(PointNum):
      for k in range(Dimension):
```

```
            for j in range(Dimension):
                Results[i,k] += preferences[j,k] * Data[i,j]


    #Tallies how many elections each candidate wins
    Tally = np.zeros(Dimension, dtype = np.int)
    for i in range(PointNum):
        if(Results[i][0] != 0):
            j = np.argmax(Results[i])
            Tally[j] += 1


    print(Tally)
    return Tally


#This is the main function, it runs when the code is executed
if __name__ == "__main__":
    # prompts the user to select a Interpretation data file
    data = input("Name the file you want to analyze: ")
    p = int(input("How many points should I use in the calculation?..."))
    f = open(data,'r')


    #Creates the more readable results file
    g = open('Outcome'+ data, 'w')


    Preferences = createPreferenceMatrix(f)
```

```
FinalOutcome = RunElection(Preferences, p)


#This writes the data into the interpretation file.
for x in range(len(FinalOutcome)):
   g.write(str(FinalOutcome[x]) + " \n")


g.close
f.close
```