Sampling Plackett-Luce Distributions to Study Condorcet Paradoxes in Ranked-Choice Voting

Ben Heskin

July 1, 2025

1 Abstract

This report models the incidence of Condorcet paradoxes in elections that use ranked-choice voting. It is known that the occurrence of such paradoxes becomes more likely as the number of voters increases, assuming each candidate is equally preferred by the voters. However, such a scenario is not plausible in the real-world, as some candidates will inevitably be preferred more or less by the electorate than others. In the absence of real data, ballots were generated using a function that samples the Plackett-Luce distribution, which in this context considers the voters propensity to vote for a candidate using weights. Then, an algorithm was run that tests the frequency of Condorcet paradoxes for various amounts of voters and candidates, inferring the probability of its occurrence for various sets of these parameters. The programming language used to generate the ballots and implement the algorithm was R. Interestingly, the probability of a Condorcet paradox tends to zero very quickly for a set of candidates with randomised weights.

Contents

2 Introduction

Social choice theory is the study of collective decision procedures and mechanisms, and it can be studied from both a mathematical and an economic perspective. A ranked-choice voting system is a collective decision-making procedure in which voters can rank a list of n candidates in order of preference, up to n preferences. In theory, such systems can expose voting paradoxes within a society, with one example being the Condorcet paradox, where individual preferences do not aggregate into a transitive majority societal preference. The frequency of such a paradox occuring for different amounts of candidates and voters has been modelled, and is known to stay the same as the number of voters increases. That said, such models assume that the probability of each candidate being given a certain preference by voters is equal, which is obviously not plausible.

The main goal of this project is to study the occurrence of such a paradox in a model that tries to mirror more closely what occurs in real world elections. This project conjecutures, that if we simulate elections using a Plackett-Luce model, the probability of the occurance of Condorcet paradoxes in a ranked choice environment diminishes as the number of voters increases.

3 Definitions

For the sake of clarity, some definitions are included below that will be taken as understood by the reader henceforth:

- Let us define a set S, where $a, b, c \in S$, and a binary relation $\prec$ that is:

  1. Connected: $\forall\ a, b \in S$ at least one of $a \prec b$ or $b \prec a$ holds.
  2. Transitive: $\forall\ a, b, c \in S$, if $a \prec b\ \ \&\ \ b \prec c$ then $a \prec c$
  3. Reflexive: $\forall\ a \in S, a \prec a$.

  Hence $\prec$ is a **ranking** of set S.

- The **majority ranking** ($\prec_M$) is defined such that for any two candidates a and b in S, $a \prec_M b$ holds $\iff$ the number of voters who prefer a over b is at least as many as the number of voters who prefer b over a.

- A **preference profile** is a set consisting of individual preference orders / ballots of each voter.

4 Theory

4.1 Condorcet Paradox

4.1.1 Definition

Given a set of candidates S , there exists a cyclic preference relation among some candidates A, B, and $C \in S$ such that:

$$A \prec_M B, \;\; B \prec_M C, \;\; C \prec_M A,$$

where $\prec_M$ denotes the majority preference relation. This preference profile implies that:

A majority of voters prefer candidate A over candidate B,

A majority of voters prefer candidate B over candidate C,

A majority of voters prefer candidate C over candidate A.

Hence, we have an occurrence of intransitive preferences. Such an occurrence is called a **Condorcet paradox.** [1]

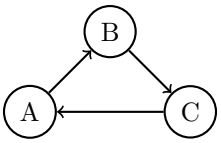

Figure 1
Graphical Representation of the Condorcet Paradox

The probability of a Condorcet paradox in a scenario with three candidates has been modeled to be approximately 8.77%. Even in larger constituencies, the probability of such paradoxes remains and increases to a near certainty when the number of voters increases.

4.1.2 Examples

We now consider an arbitrary example of a Condorcet paradox in an election with six candidates and six voters. In this example, candidate A is preferred by the majority over candidate B, candidate B is preferred by the majority over candidate C, candidate C is preferred by the majority over candidate D, candidate D is preferred by the majority over candidate E, and candidate E is preferred by the majority over candidate F. However, candidate F is preferred by the majority over candidate A. Hence, the majority preferences of this electorate are cyclic, and there is no clear overall winner in this election. This scenario is presented in a table and a graph below, where the vertices of the graph represent the candidates and the directed edges indicate the candidate that was preferred in each pairing.

Voter	1st	2nd	3rd	4th	5th	6th
1	A	B	C	D	E	F
2	B	C	D	E	F	A
3	C	D	E	F	A	B
4	D	E	F	A	B	C
5	E	F	A	B	C	D
6	F	A	B	C	D	E

Figure 2
Table for Six Voters and Six Candidates

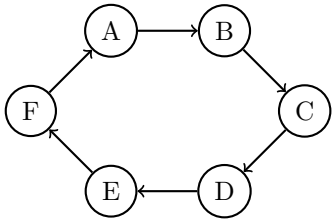

Figure 3
Condorcet Paradox Graph for Six Voters and Six Candidates

Interestingly, real-world examples of Condorcet paradoxes in elections that use a ranked-choice voting system have been found, meaning that the occurrence of the paradox is common and it is more than a theoretical concept. One such occurrence was in the jurors' preferences in the 2023 Grand Final of the Eurovision Song Contest. There were five distinct Condorcet cycles involving Austria and Lithuania [5]:

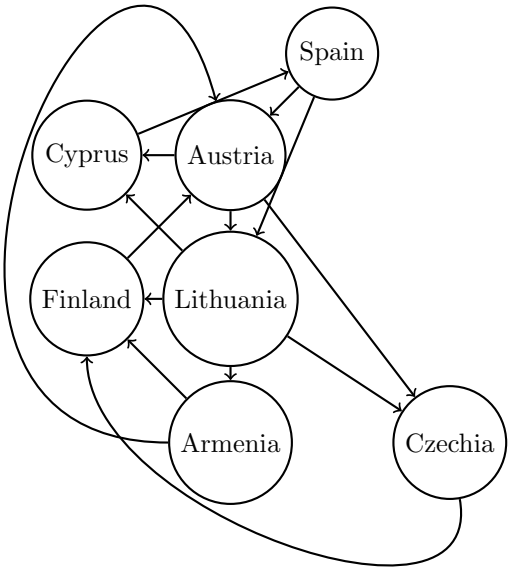

Figure 4
Occurrences of Condorcet Paradoxes in the 2023 Grand Final of the Eurovision Song Contest

4.2 Plackett-Luce Distribution

4.2.1 Definition

The Plackett-Luce distribution is a family of distributions on strict rankings of a finite set. Considering a set of J items $\{i_1, i_2, \ldots, i_J\}$, the probability of ranking the j'th item first is:

$$\frac{\lambda_j}{\sum_{i \in S} \lambda_i}$$

The probability of ranking the j'th item second given the k'th item was ranked first is:

$$\frac{\lambda_j}{\sum_{i \in S, i \neq i_k} \lambda_i}$$

where λ_i is the worth or weighting of candidate i [3] [4] .

4.2.2 Properties

There are several important and relevant properties of the Plackett-Luce distribution should be considered.

First, if all of the weights are equal, then the Plackett-Luce distribution assigns equal probability to all rankings.

Second, multiplying the weights by a common factor does not alter the relative probabilities assigned by the Plackett-Luce distribution.

Third, and perhaps most importantly, if we take a Plackett-Luce distribution and look at the relative rankings within a subset of the original set, then the distribution of those rankings is again a Plackett-Luce distribution. Hence, if we consider a set of weights where some of them are the same, an equal probability is assigned to each of these weights, and the remaining unequal weights will follow a Plackett-Luce distribution. This last property will be considered in the results section of this report.

5 Algorithm

This section discusses different parts of the code used in this project. The code is written in R, and all of its described in this section are included in the appendix of this report. [7]

To summarize, the code generates ballots for each voter (there are n), with every candidate (there are m) getting a preference from 1 to m from each voter. The ballots are collated into an $n \times m$ matrix. Then, the occurrence of any Condorcet paradoxes is determined for the ballot matrix.

The code has a few parameters, for example, the amount of voters is defined. In this case, a for loop is run over a vector that contains increasing powers of 2. Also, the number of times the process is repeated for each number of voters is defined. This ensures that a total probability of a Condorcet paradox can be derived for each step. Also, the respective weights are arbitrarily pre-chosen and inputted into the model.

5.1 Generation of Ballots

First, m ballots must be generated for n candidates, each with a corresponding worth/weight. This is done by defining a function that selects the candidate who receives the first preference on a specific ballot, based on the pre-determined weights. Then, the function re-normalises the weights after removing the candidate that was chosen. This process is continued iteratively until the last candidate is given a preference on the ballot. The function is laid out in pseudocode below:

```
function GENERATE_SINGLE_BALLOT(candidates, weights)
    m ← number of candidates
    ballot ← vector of length m
    remaining_candidates ← candidates
    remaining_weights ← weights
    for i ← 1 to m do
        selected_candidate ← sample(remaining_candidates, weights)
        ballot[index of selected_candidate in candidates] ← i
        Update and normalize remaining_weights
    end for
    return ballot
end function
```

This process is repeated n times, and each generated ballot is sequentially placed into a corresponding row of the ballot matrix, resulting in a matrix that captures the preferences of all the voters.

5.2 Measuring the Occurrence of Condorcet Paradoxes

Next, given the ballot matrix was generated, an approach that leverages preferences matrices is employed to quantify the occurrence of Condorcet paradoxes.

First, an empty $m \times m$ matrix is created, namely the preference matrix (P). A loop iterates through the ballot matrix, and if candidate i is preferred over candidate j in a particular ballot, the entry in p_{ij} is incremented by 1.

```
1: function CreatePreferenceMatrix(ballots_matrix, m, n)
2:     Initialize preference_matrix as an empty m × m matrix
3:     for i ← 1 to n do
4:         b ← cast to numeric(ballots_matrix[i, :])
5:         for i ← 1 to n do
6:             for j ← 1 to n do
7:                 if b[i] < b[j] then
8:                     preference_matrix[i, j] ← preference_matrix[i, j] + 1
9:                 end if
10:            end for
11:        end for
12:    end for
13:    return preference_matrix
14: end function
```

Second, the preference matrix has its transpose taken away from it, creating an augmented preference matrix. If entry p_{ij} is positive, then candidate i had a majority preference over candidate j and if entry p_{ij} is negative, then candidate j had a majority preference over candidate i .

This augmented preference matrix can then be used to define a matrix C where $c_{ij} = 1$ if candidate i is preferred by the majority over candidate j (ie if p_{ij} is positive), or $c_{ij} = 0$ otherwise.

Inductively, the value of the entry c_{ij} in C^k is the number of chains that start at i and end at j of length k. Intuitively, the entries on the trace of C^k must represent the chains that begin at candidate i and end at candidate i, ie. a Condorcet paradox. This project focuses on the occurrence of a Condorcet paradox, therefore if the trace of any of the matrices C^k, for $3 \leq k \leq m$ has a non zero entry, the corresponding position of a vector for this loop is incremented by 1.

```
1: power_matrix ← matrix_C
2: cp ← 0
3: for i ← 3 to n do
4:     power_matrix ← power_matrix × matrix_C
5:     for j ← 1 to n do
6:         if power_matrix[j, j] > 0 then
7:             cp ← cp + power_matrix[j, j]
8:         end if
9:     end for
10: end for
11: if cp > 0 then
12:     cp_vector[z] ← 1
13: else
14:     cp_vector[z] ← 0
15: end if
```

This process is repeated 1000 times for each number of voters to extrapolate probability of a Condorcet paradox for that number of voters.

6 Results

The first sample set of results, as shown in Figure 5, illustrates the frequency of Condorcet paradoxes for four different sets of weights: five same weights, five random weights, three equal weights, and two equal weights. As a code-check test, the scenario with identical weights was run to confirm that the probability of a paradox increases as the number of voters increases. Second, a set of random weights was run, and, as expected, the number of Condorcet paradoxes decreased quickly as the number of voters increased. Then, scenarios with two equal weights and three equal weights were run. For the scenario where there were two equal weights, the three remaining weights formed their own Plackett-Luce distribution and the probability had a more gradual decrease as the number of voters increased. An opposite analysis can be performed for the scenario with three equal weights.

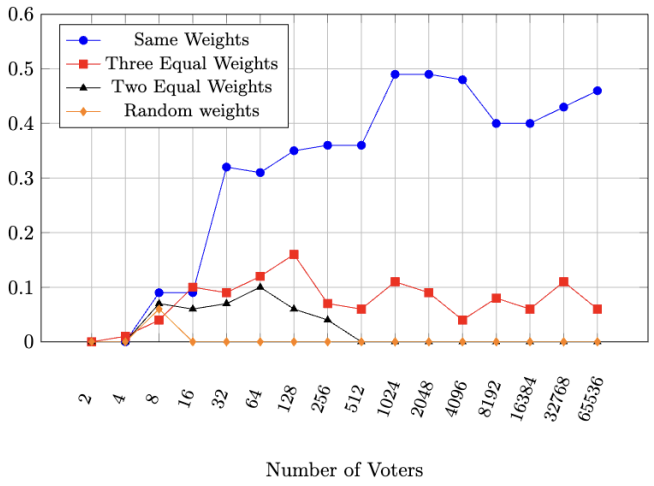

Figure 5

Next, to understand how different numbers of candidates affect the probability of Condorcet paradoxes, different and equal sets of weights were run for five and thirteen candidates, as seen in Figure 6. As previously modelled, the probability of the occurrence of a Condorcet paradox does indeed increase as the number of candidates increases in scenarios with equal weights. As expected, the probability of Condorcet paradoxes increases slower and has a higher peak as the number of candidates increases.

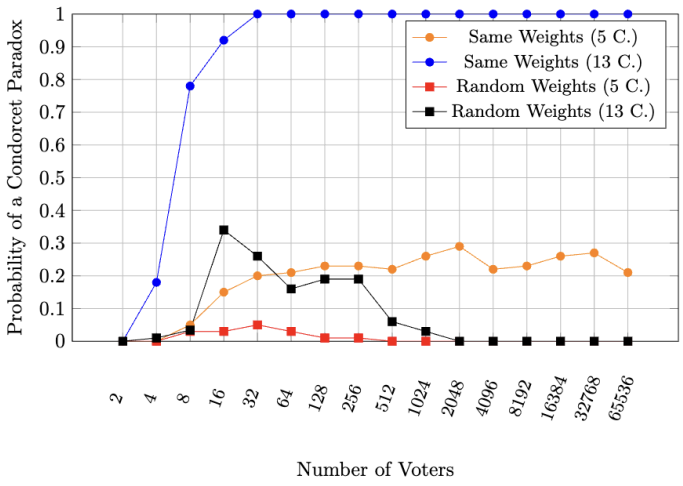

Figure 6

Finally, scenarios involving twenty-six candidates with different sets of weights were analyzed to further understand the impact of a large candidate pool on the probability of a Condorcet paradox. The number of candidates was set to twenty-six to closely model the Eurovision Song Contest Final, which featured the same number of participating countries in 2023. It is important to note that there were 183 jurors for the 2023 final:

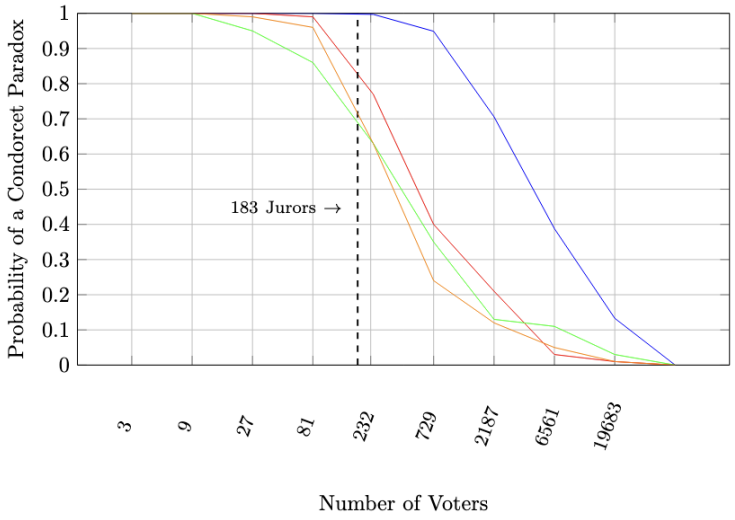

Figure 7

It is clear that for cases such as the final of the Eurovision Song Contest, it is plausible that a scenario with a high number of candidates has a very high probability for a relatively low number of voters under a Plackett-Luce model. This probability still diminishes quickly as the amount of voters increases, as seen with other various numbers of candidates.

7 Conclusion

It is clear that under a Plackett-Luce model, the probability of a Condorcet paradox in a ranked-choice voting system diminishes pretty quickly as the amount of voters increases. Condorcet paradoxes do occur in real-world votes such as the Eurovision Song Contest, these cases need to be considered in their specific contexts. The Eurovision Song Contest has a relatively low number of jurors (voters), and as seen in the Plackett-Luce model presented in this report, the probability of the occurrence of paradoxes remains relatively high. This high probability is especially relevant in the grand final of the Eurovision, where there are twenty-six countries / candidates. This study models the probability of a Condorcet paradox increasing with the number of candidates, indicating that such a large number of participants increases the likelihood of a Condorcet paradox.

There are limitations to this this model. Each voter must give each candidate a preference, which is not always the case in real-world elections. Additionally, it is also difficult to list cycles and to see how their characteristics change as the number of voters changes and if different sets of weights have different effects on the rate at which the probability of a Condorcet paradox decreases. Moreover, there is no consideration of voting blocs in this model. Voting blocs are groups of voters that have similar voting preferences who often such transfer their preferences to candidates that represent the views of the bloc [2].

There are issues with the implementation of this model. A lot of the computation time was spent generating the ballots. In particular, for iterations with low numbers of candidates, it is clear that calculations such as the re-normalisation of weights were often repeated. Perhaps a package that uses memoisation could have been beneficial. Also, most of the rest of the time was spent creating the preference matrix. Perhaps a function that considered Smith Sets would have been more efficient [6].

References

[1] Marquis de Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. 1785. PNG.

[2] Isobel Claire Gormley and Thomas Brendan Murphy. A mixture of experts model for rank data with applications in election studies. *The Annals of Applied Statistics*, 8(2):1145–1181, 2014.

[3] R. Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, 1959.

[4] R. L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society*, 1975.

[5] Mya Prendergast. Capstone project on voting paradoxes in the eurovision song contest, 2024.

[6] John H. Smith. Aggregation of preferences with variable electorate. *Econometrica*, 41(6):1027–1041, 1973.

A Code

A.1 Function used to Generate the Ballots

```
c <- c("A", "B", "C", "D", "E" )
# Function to generate a single ballot
generate_ballot <- function(candidates, weights) {
    m <- length(candidates)
    ballot <- integer(m)  # a vector to store rankings which will be added to the
    ballots matrix
    remaining_c <- c
    remaining_w <- w
    for (i in seq_len(m)) {
        # Select a candidate based on the current weights
        selected_c <- sample(remaining_c, 1, prob = remaining_w)
        # Store the rank of the selected candidate in the ballot
        ballot[match(selected_c, c)] <- i
        # Remove the selected candidate from the remaining candidates
        to_remove <- match(selected_c, remaining_c)
        remaining_c <- remaining_c[-to_remove]
        # Remove the corresponding weight and re-normalize the remaining weights
        remaining_w <- remaining_w[-to_remove]
        if (length(remaining_w) > 0) {
            remaining_w <- remaining_w / sum(remaining_w)
        }
    }
    return(ballot)
```

A.2 Function used to create the Preference Matrix

```
    # Define the ballot_to_preference function
ballot_to_preference <- function(ballots_matrix, c) {
    num_voters <- n
    m <- length(candidates)
    preference_matrix <- matrix(0, m, m)

    # Iterate through each ballot
    for (i in 1:num_voters) {
        # Get the current ballot
        bal <- as.numeric(ballots_matrix[i, ])
        # Compare each candidate's rank with others in the ballot
        for (a in 1:n) {
            for (b in 1:n) {
                if (bal[a] < bal[b]) {
                    preference_matrix[a, b] <- preference_matrix[a, b] + 1
                }
            }
        }
    }
    return(preference_matrix)
```

A.3 Code used to Detect the Occurrence of a Condorcet Paradox

```
        preference_matrix <- ballot_to_preference(ballots_matrix, c)
         Preference_matrix_transpose <- t(preference_matrix)
         Winner_matrix <- preference_matrix - Preference_matrix_transpose
         modified_matrix <- matrix(0, n, n)
         for (c in 1:n) {
             for (d in 1:n) {
                 if (Winner_matrix[c, d] > 0) {
                     modified_matrix[c, d] <- 1
                 } else {
                     modified_matrix[c, d] <- 0
                 }
             }
         }

         power_matrix <- modified_matrix
         cp <- 0
         # Multiply the matrix n times
         for (i in 3:n) {
             power_matrix <- power_matrix %*% modified_matrix
             for (j in 1:n) {
                 if (power_matrix[j, j] > 0) {
                     cp <- cp + power_matrix[j, j]
                 }
             }
         }
         if (cp > 0) {
             cp_vector[z] <- 1
         } else {
             cp_vector[z] <- 0
         }
     }
     # Count the number of ones in the cp_vector
     number_of_ones <- sum(cp_vector)
     percentage_ones <- number_of_ones / r
print ("The probability of a Condorcet paradox for" m "Candidates, with the weights" w "
    is" percentage_ones )
```