David Bass

2 November 2020

Constructing and Simulating Critical Values for the Plurality Test

Abstract

If m candidates run in an election in which n voters vote for 1 candidate, how can we be confident that the leading candidate has a statistically significant plurality of the votes? In a first-past-the-post election with multiple candidates competing for one position, there is often no single candidate who obtains a majority of the votes. Thus, it is of interest to determine which candidate in an election wins the most votes, i.e. a plurality. In this paper, I construct, describe, and simulate critical values for a hypothesis test, which I will call the plurality test, to determine whether the leading candidate in an election has a statistically significant plurality of the vote.

Introduction

When performing a hypothesis test, it is necessary to establish a rejection region for the relevant test statistic that corresponds to p-values less than or equal to the chosen significance level α. Often, this rejection region is derived from the underlying distribution of the test statistic under the null hypothesis. For example, let the test statistic $x \sim \mathcal{N}(0, 1)$ under the assumption of the null hypothesis, let hypotheses $H_0 : x = 0$ and $H_a > 0$, and let significance level $\alpha = 0.05$. Because the integral of the standard normal distribution from 1.6449 to ∞ is approximately equal to 0.05, a rejection region of $x > 1.6449$ can be established for the hypothesis test because, under the assumption of the null hypothesis, the probability of the test statistic assuming a value greater than 1.6449 is less than α. 1.6449, the finite bound of the rejection region, is called the critical value.

However, the underlying probability distribution function of the test statistic under the null hypothesis is not always known. Without a distribution to consult, it is impossible to establish a rejection region for the hypothesis test with calculus, as done in the example above. Fortunately, simulations can alternatively be used to estimate critical values for a rejection region.

When performing a hypothesis test whose alternative hypothesis is a one-sided inequality, there will only be one well-defined critical value because the rejection region will extend either to ∞, in the case of a greater-than alternative hypothesis, or to $-\infty$, in the case of a lesser-than alternative hypothesis. To calculate the critical value via simulation, a large number of trials B (B is generally between 10^4 and 10^6, with greater values giving greater accuracy, but with diminishing returns) are performed via random reproduction of an experimental scenario under the assumption of the null hypothesis. Thus, for a greater-than alternative hypothesis, the critical value can be estimated to be equal to the $100(1-\alpha)$th percentile test statistic from the B test statistics calculated from the B simulated experiments. In the case of a lesser-than alternative hypothesis, the critical value would be estimated as the 100αth percentile test statistic. This method works because the proportion of simulated experiments, which are performed under the assumption of the null hypothesis, that take on value at least as extreme as the simulated critical value is equal to α. Thus, test statistic values at least as extreme as the simulated critical values fall into the hypothesis test's rejection region, meaning that the null hypothesis can be rejected at a $100\%(1-\alpha)$ confidence level.

This method of simulating critical values is an essential tool for non-parametric statistics because it allows for hypothesis testing to be performed on test statistics whose underlying distributions are unknown. However, it has numerous downsides. The most significant downside of all statistical simulation methods is the time and computing resources that they take. As aforementioned, simulating critical values should be done with at least 10^4 trials; it can take a decent personal computer many minutes to run this many simulations for a single experimental scenario. Additionally, each experimental scenario must be simulated

separately. When a parametrized distribution of the test statistic is available, calculating critical values for different experimental scenarios (e.g. normally-distributed test statistics with different means and variances) simply requires tweaking some values in a probability density function or a line of code. However, when simulating critical values, a new simulation, with B simulated experiments, must be performed for each new choice of experimental parameters or significance level α. Finally, simulations are not perfect; they are the result of random number generation, so they are subject to random error. While the accuracy of simulated estimates can be trusted with great confidence for high values of B, rigorous scientific guidelines may require that B be too great to make performing the simulation feasible on a normal computer.

Experimental

In a first-past-the-post election in which n voters each vote for one of m candidates, it is possible that all candidates receive roughly equal amounts of votes. In order to claim that the candidate with the most votes has a statistically significant lead over his or her competitors, a hypothesis test can be constructed.

Let x_1 be the number of votes received by the candidate with the greatest number of votes and let x_2 be the number of votes received by the candidate with the second-greatest number of votes. Because we are only concerned with determining the winner of the election, a sufficient test statistic for the hypothesis test is $d = x_1 - x_2$. I will call this hypothesis test the plurality test. Its hypotheses are as follows:

$H_0 : x_1 = x_2$.

$H_a : x_1 > x_2$.

For the remainder of this paper, I will use the significance level $\alpha = 0.05$. Thus, the rejection region for the test statistic d corresponds to the values for d that are so extreme (note that, by definition, $x_1 \geq x_2$, so $d \geq 0$) that, under the null hypothesis that $x_1 = x_2$,

they would only occur with probability less than α.

Because I was not able to calculate an underlying distribution for d, I developed a simulation to calculate critical values for such rejection regions given different combinations of m and n. The code simulates 10^4 (i.e., $B = 10^4$) elections for each pairwise combination of values in the sets of voter counts $m = \{2, 3, 5, 10, 20\}$ and candidate counts $n = \{10^1, 10^2, 10^4, 10^6\}$ under the null hypothesis's assumption that each voter is equally likely to vote for any candidate. Note that this is a convenient simplification from the aforementioned null hypothesis, which only claims that the 2 leading candidates get the same number of votes. The test statistic $d = x_1 - x_2$ values were sorted (in ascending order) for each simulation and the respective critical value was estimated as the 95th percentile value of d from this sorted list. The full, documented Java code used to run these simulations can be found in Appendix 1.

Results

The simulations yielded the following results:

	Critical values (number of votes that the leading candidate needs to lead by for a significant plurality)					
		Number of candidates				
		2	3	5	10	20
Number of	10	6	4	3	2	2
voters	100	20	13	9	5	4
	10000	198	126	81	48	30
	1000000	1954	1234	788	416	254

Figure 1.

The significant increases in critical value with increased number of voters n, but constant number of candidates m is due to the need for more votes to show a significant advantage of the leading candidate when there are more overall voters. However, in every column of Figure 1, the rate of increase of the critical values is significantly less than that of n; note that n increases by a factor of 100 from the penultimate row to the last row, but the critical values all increase by less than a factor of 10. This is likely a result of how when there are

more voters, the variance decreases, so a statistically significant difference can be determined from a lesser difference d.

Within each row, the critical values decrease logarithmically as the number of candidates m increases. This is likely due to the fact that large values of d are uncommon when the votes are distributed among many candidates.

The critical values can be more intuitively displayed as proportions of n, as they are in the following graphs, which were generated by the R code found in Appendix 2:

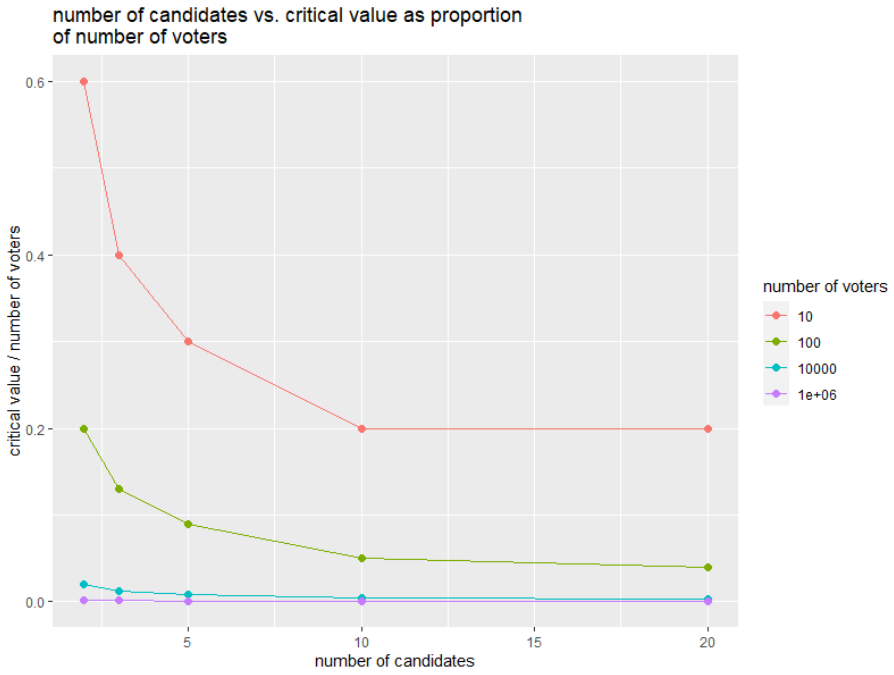

Figure 2.

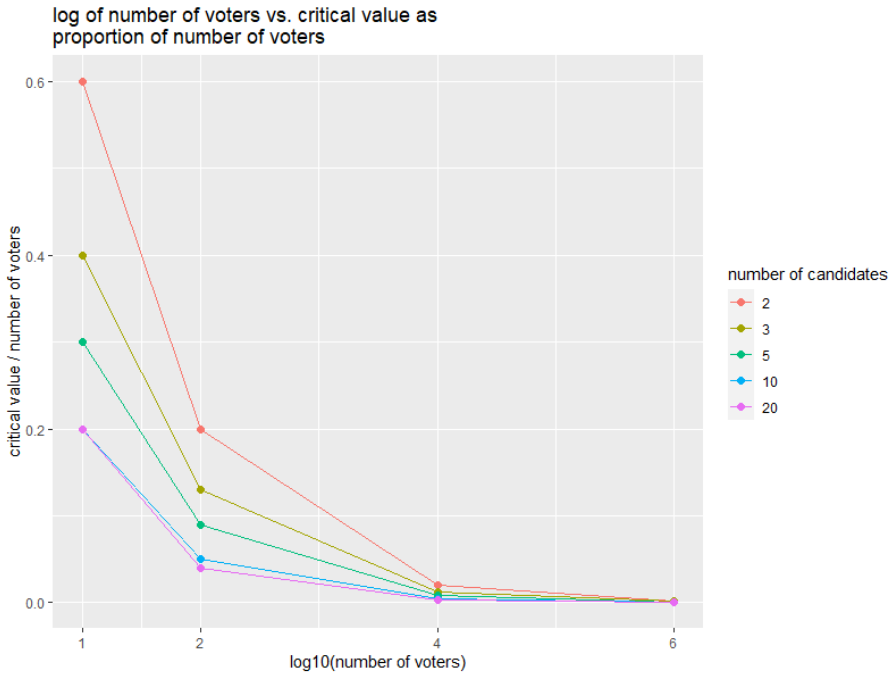

Figure 3.

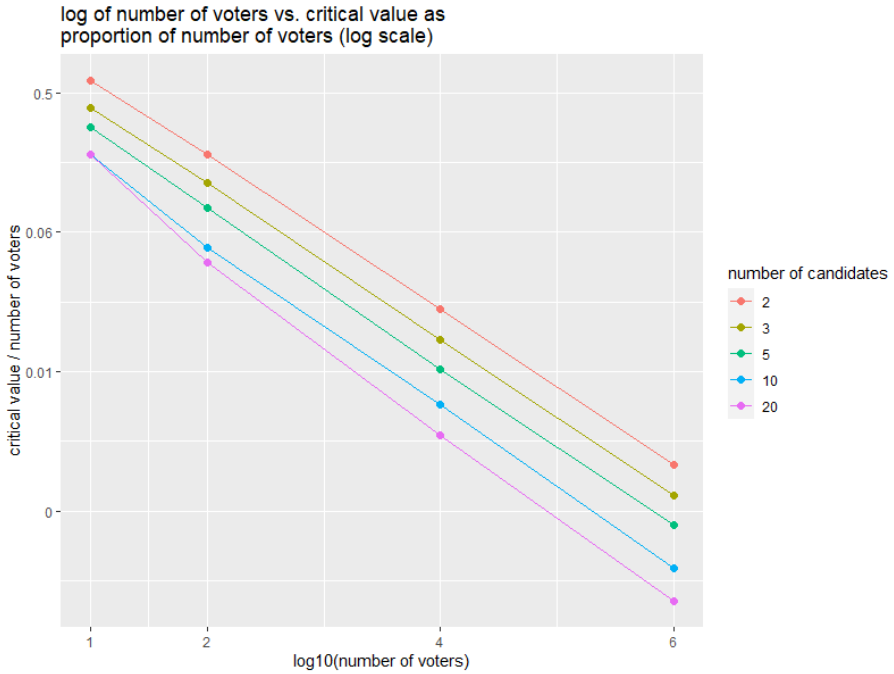

Figure 4.

Figure 2 and Figure 3 show the aforementioned negative correlations between number of candidates m and critical values and number of voters n and critical values, respectively. Figure 4 differs from Figure 3 only in that the y-axis is transformed to a $\log_2$ scale. This

scaling reveals the trend that there is a strongly linear relationship between $\log_{10} n$ and $\log_2$(critical value / n), i.e. for each increase in n by a factor of 10, the proportion of n that the critical value represents decreases by a factor of 2 times some constant k (k equals to the slope of the lines in the graph).

Conclusion

By constructing and simulating the plurality test, I found strong graphical and numerical evidence that, as the number of candidates in an election increases, or as the number of voters in an election increases, a lesser vote difference d is required to claim, at a 95% confidence level that the leading candidate has a statistically significant plurality of the votes. Additionally, I estimated exact critical values for d in order to establish respective rejection regions of $d \in [\text{critical value}, \infty)$ for which the null hypothesis that no candidate has a significant plurality of the votes. I failed to construct a probability distribution function for the distribution of the test statistic $d = x_1 - x_2$ with parameters m and n, but the findings from my examination of Figure 4 give hope that there exists such a distribution that can be described mathematically.

The obvious application of my results is to real-life elections. The get_critical_val() function of my Java code could be used to generate a critical value of the test statistic d for any combination of m, n, and α, and B. While the values that I chose for these variables in my recorded simulations were arbitrary, they provide a good reference frame for the ranges of values of m and n that might exist in real-life elections.

In addition to elections, the plurality test is applicable to tree-based classification. In a tree-based classification model, observations in the training data are separated into some number of boxes based on their predictor values. For prediction of the response variable of an observation in the test data, the observation is placed into a box based on its predictor values and then estimated to have the categorical response variable value that is shared by

the greatest number of training data observations in that box. Essentially, the classification is determined based on a plurality vote of the observations in the box. In the context of the plurality test, n is the number of the training data observations in the box and m is the number of response variable values represented by observations in the box. While tree-based classification models normally don't consider the significance of the plurality in each box when classifying, it would be helpful to consider the significance of a given box's purity as assessed by the plurality test. If many boxes in the model fail to reject the null hypothesis of the plurality test, then the model's boxes likely lack the necessary response variable purity necessary to perform adequately. This is indicative of underfitting in tree-based models. In the case of underfitting, it would be optimal to subdivide extant boxes in order to increase average response variable purity in each box, allowing for a greater proportion of boxes to feature a significant plurality.

That being said, there were potential sources of error in my methodology. The null hypothesis of the plurality test states that $x_1 = x_2$. However, my simulated elections used the null hypothesis assumption that all candidates, not just the two leading ones, were equally favored. This was done to avoid introducing new variables to the plurality test, whose critical values are only dependent on m, n, and α, but may have contributed to a deflation of estimated critical values; it is possible, if the candidates besides the top 2 receive significantly fewer votes than the top 2 candidates under the null hypothesis, that the critical values would be greater because the top 2 candidates could get a large proportion of the total votes, making the election behave more like one with fewer candidates. As aforementioned, there exists a significant negative correlation between number of candidates and critical values of the plurality test.

Additionally, exact 95th percentile values of d under the null hypothesis were chosen as the estimated critical values in the simulation. Thus, rejection regions were established, for some critical value c, as $d \in [c, \infty)$, while rejection regions are normally established as open intervals in order to allow for the statement that the test statistic must be strictly more

extreme than the critical value c. This was done because d has a discrete distribution; only integer numbers of votes can exist. It is possible that more accurate critical values would have been selected by a method similar to the quantile() function in R, which can return decimal values even from integer-list outputs, as opposed to the simpler indexing method that I used to obtain 95th percentile values for d.

Thanks for reading!

Further Questions

- If the simulation's simplifying assumption that all candidates are equally likely to receive any given voter's vote has a deflationary impact on its critical value estimates, then what can be done to prevent that unwanted impact?

- Imagine that the goal of the election were instead to choose multiple candidates for a run-off election. How could the plurality test be modified to allow for a leading group of candidates, not just one leading candidate, to be identified?

- How could the plurality test be modified to describe rank-choice voting, in which each voter votes for multiple candidates?

- Can a probability density function with parameters m and n be constructed to describe the distribution of critical values of the plurality test, or must critical values always be simulated?

Acknowledgements

Thanks to Professor Gretchen Martinet for help with formatting and to Leo Murphy for posing the question that inspired this project.

Appendix 1: Java Code

The following Java code was used to simulate the critical values for the plurality test:

```
import java.util.Arrays;
import java.util.Random;

public class plurality_test {
        static Random r = new Random();

        public static void main(String[] args) {
                r.setSeed(1);
                // iterations is the number of times that each simulation will be run

                // iterations should be at least 10^4, and can be increased for greater simulation
                // accuracy at the cost of greater runtime
                int iterations = (int) Math.pow(10, 4);
                // elements of sizes are the number of voters in a given election
                int[] sizes = {(int) Math.pow(10, 1), (int) Math.pow(10, 2), (int) Math.pow(10, 4),
                                (int) Math.pow(10, 6)};
                // elements of cand_counts are the number of candidates in a given election
                int[] cand_counts = {2, 3, 5, 10, 20};

                // run simulations for every pairing of elements in sizes and cand_counts and
                // print output
                int critical_val;
                for (int size : sizes) {
                        for (int count : cand_counts) {
                                // call get_critical_val at a significance level of 0.05, i.e. 95%
                                // confidence
                                critical_val = get_critical_val(iterations, size, count, .05);
                                System.out.println(critical_val);
                        }
                        System.out.println();
                }
        }

        // simulate election, return critical value
        public static int get_critical_val(int iterations, int size, int cand_count, double alpha) {
                int[] diffs = new int[iterations];

                // run simulation
                int[] votes = new int[cand_count];
                for (int k=0; k<iterations; k++) {
                        // clear vote counts before simulating new election
                        for (int i=0; i<cand_count; i++) {
                                votes[i] = 0;
                        }
                        // simulate new election
                        for (int i=0; i<size; i++) {
                                votes[r.nextInt(cand_count)]++;
                        }
                        // record difference in votes between candidate with most and second-most
                        // votes
                        Arrays.sort(votes);
                        diffs[k] = votes[cand_count-1] - votes[cand_count-2];
                }

                // return
                Arrays.sort(diffs);
                int critical_val = diffs[(int) (iterations * (1.0 - alpha))];
                return critical_val;
        }
}
```

Appendix 2: R Code

The following R code was used to create Figures 2, 3, and 4:

```
require(ggplot2)

#data copied from Java code output
data <- data.frame("num_of_voters" = rep(c(10^1, 10^2, 10^4, 10^6),
                                         each=5),
                   "num_of_cands" = rep(c(2, 3, 5, 10, 20), 4),
                   "critical_val" = c(6, 4, 3, 2, 2, 20, 13, 9, 5, 4, 198, 126,
                                      81, 48, 30, 1954, 1234, 788, 416, 254))

#create column to show critical values as a proportion of number of voters
data[,"critical_proportion"] = data[,"critical_val"] / data[,"num_of_voters"]

#
data[,"voter_levels"] = as.factor(data[,"num_of_voters"])
data[,"cand_levels"] = as.factor(data[,"num_of_cands"])

plot1 <- ggplot(data, aes(x=num_of_cands, y=critical_proportion,
                          color=voter_levels)) + geom_line() + geom_point(size=2) +
         ggtitle("number_of_candidates_vs._critical_value_as_proportion_\nof_number_of_voters") +
         xlab("number_of_candidates") +
         ylab("critical_value_/_number_of_voters") +
         labs(color = "number_of_voters")
plot1

#functions for making axis labels more readable
f1 <- function(x) log(x, 10)
f2 <- function(x) round(x, 2)

plot2 <- ggplot(data, aes(x=num_of_voters, y=critical_proportion,
                          color=cand_levels)) + geom_line() + geom_point(size=2) +
         ggtitle("log_of_number_of_voters_vs._critical_value_as_\nproportion_of_number_of_voters") +
         scale_x_continuous(trans="log10", labels=f1, breaks=c(10, 100, 10000, 1000000)) +
         xlab("log10(number_of_voters)") +
         ylab("critical_value_/_number_of_voters") +
         labs(color = "number_of_candidates")
plot2

plot3 <- ggplot(data, aes(x=num_of_voters, y=critical_proportion,
                        color=cand_levels)) + geom_line() + geom_point(size=2) +
         ggtitle("log_of_number_of_voters_vs._critical_value_as_\nproportion_of_number_of_voters_(log_scale)") +
         scale_x_continuous(trans="log10", labels=f1, breaks=c(10, 100, 10000, 1000000)) +
         scale_y_continuous(trans="log2", labels=f2) +
         xlab("log10(number_of_voters)") +
         ylab("critical_value_/_number_of_voters") +
         labs(color = "number_of_candidates")
plot3
```