Presidential Election Simulator

Gary Gregg
University of Washington
garygr@uw.edu
INFO 474 – Interactive Information Visualization, Winter 2018

ABSTRACT

The Electoral College has been a feature of the American republic since its inception in the late 18th century. The college is an indirect means of electing the President and Vice President of the United States. Its creation reflected a general mistrust of direct democracy at the time of its creation. The operation of the college can be confusing, and several times in more than two centuries it has produced a winner who did not receive at least a plurality of the popular vote. Simulation of Presidential elections can help students and foreign observers understand how the Electoral College can either amplify, or invert the popular vote in an election. We present here an interactive Presidential election simulator that has a high degree of statistical control over its results. Our goal is increase student education about this unique feature of American politics. Throughout this paper, we use a plural first person pronoun, although at this time there is only one author.

INTRODUCTION

At the Constitution Convention in 1787, the founding fathers of the American republic devised the Electoral College as a means of indirect election of the President and Vice President of the United States. The reasons for its creation are varied. According to Joe Miller at *FactCheck.org*, the founding fathers created the Electoral College because they were afraid of direct democracy. James Madison worried about "factions," which he defined as citizens who had a common interest in a proposal that would violate the rights of others, or harm the nation as a whole. In *The Federalist Papers*, Alexander Hamilton wrote that this feature of the Constitution was designed to ensure "that the office of the President will never fall to the lot of any man who is not in an eminent degree endowed with the requisite qualifications." Events in the 21st century may lead observers to doubt whether Hamilton's goal has been achieved. Nevertheless, the Electoral College has persisted since the inception of the republic. Since the existence of the college is specified by the Constitution, it would take an amendment to the Constitution to abolish the college, and provide for direct election of the chief executive of the United States.

A review of the workings of the college are thus: Each state in the United States is assigned a number of electors

based on its population, and is the same as that state's number of representatives in the U.S. House of Representatives, plus two. The additional electors are assigned based on the state's representation in the U.S. Senate. Because of the additional two electors that the Constitution specifies, states with smaller populations have an outsize representation in the college. In Wyoming, for example, each elector represents the will of less than 200,000 voters, but in Texas each elector represents more than 700,000 voters (Reference: Wikipedia). Because the assent of ¾ of the states are required to amend the Constitution, it is unlikely that the less populated states of the United States would agree to abolish their additional influence in the selection of the President and Vice President. The Constitution also specifies that every ten years a census shall be conducted. The results of the census are used, in part, to reapportion electors among the various states. Thus every ten years – due to shifting U.S. population – a state may gain or lose representation both in the U.S. House of Representatives, and in the Electoral College.

Each state is relatively free to determine the process by which they select electors. The electors themselves are citizens selected by the political party of the Presidential candidates. The Constitution specifies that electors meet on the first Monday after the second Wednesday in December after a Presidential general election. There, they cast separate ballots for the President and the Vice President of the United States. Currently, 48 of U.S. states, select a slate of electors who are all pledged to support the winner of the popular vote *in that one state*. Thus, the winner of the popular vote wins all the electors in these states. Maine and Nebraska use a different formula. These states select an elector pledged to the winner of the popular vote in each of their Congressional districts, and two electors pledged to the winner of the state at-large.

The Constitution species that a candidate must win support of a majority of electors in the college in order to be elected President. Currently, that number is 270 of 538 electors. If no Presidential candidate wins support of the majority of electors, then the incoming U.S. House of Representatives will select a President from among the top three competitors in the college. The U.S. Senate will select a Vice President from among the top two competitors in that race. Congressional selection of these two chief executives has happened only once, in 1824. In this election, both the popular vote and the electoral vote were split between four candidates, with Andrew Jackson having a plurality count of both. But he did not enjoy a majority of either. Despite his

predominance, the U.S. House of Representatives selected John Quincy Adams instead of Jackson. The two major party system is, in part, a result of a desire for Presidential elections not to be decided by Congress in this way. With only two candidates participating, it would take a tie for one of them not to receive a majority of electoral votes. A tie is an unlikely outcome, and has not before occurred.

More often that not, the result of Electoral College vote tends to magnify the result of the popular vote. However, this outcome is far from certain. In four elections since 1824, a major party candidate who did not have a least a plurality of the popular vote became President. In 1876, Rutherford Hayes prevailed over Samuel Tilden, despite Tilden enjoying a *majority* of the popular vote. In 1888, Benjamin Harrison prevailed over Grover Cleveland, who had a popular vote plurality. In 2000, George W. Bush prevailed over Albert Gore, Jr. in this way, and most recently, in 2016, Donald Trump bested Hillary Clinton without a popular plurality. In this most recent election, Mrs. Clinton enjoyed an almost 3 million popular vote margin, an advantage of more than 2%. In the Electoral College, Mr. Trump won the election with 304 electoral votes to 227 for Clinton. Due primarily to the winner-take-all approach to elector selection, we can see that the results in the college can be powerfully different than the general election popular vote.

It is difficult for new students of civics and politics to understand how a President can be selected when another candidate enjoyed more popular support. This is also true of foreign observers who are new to the American election system. Using simulation, we can demonstrate quickly, and in real time, how results like this can occur. And we can also demonstrate how the Electoral College can easily magnify the result of a popular vote win. This was our goal in creating an effective Presidential election simulator, along with providing a high degree of mathematical control over the variability of the results.

RELATED WORK

Other researchers, and interactive game creators have done similar work. None that we have found seek the degree of mathematical control over their simulations that we seek. The following is a short list of other efforts:

2016 Presidential Election Simulation (Clinton vs. Trump): https://www.270towin.com/2016-simulation/

Presidential Election Game: https://270soft.com/us-election-games/president-election-game-2016-infinity/

METHODS

In order to simplify the process of creating an effective simulation, we decided to include the following limitations:

- That simulations would be limited to two major party candidates
- That the winner-take-all elector selection method would also apply to Maine and Nebraska
- That only the most recent five Presidential elections would be available for simulation, 2000 to 2016

The statistical distribution that most accurately models the result of a two candidate election is the *beta* distribution. The beta distribution is specified by two parameters, α and β. When α and β are both equal, and greater than 1, the PDF of the distribution takes the form of a symmetric bell curve limited to values between 0 and 1. The mean of such a distribution is 0.5. In contrast, the PDF of a Gaussian, or normal distribution has a similar shape, but the tails of the distribution have positive values extending to both positive and negative infinity. For the beta distribution, if α is less than β, then the curve of the distribution skews left, with more mass occurring toward 0 than 1, and the mean is less than 0.5. If α is greater than β, then curve of the distribution skews right, with more mass occurring toward 1 than 0, and the mean is greater than 0.5. See figure 1 for representative samples of the beta distribution

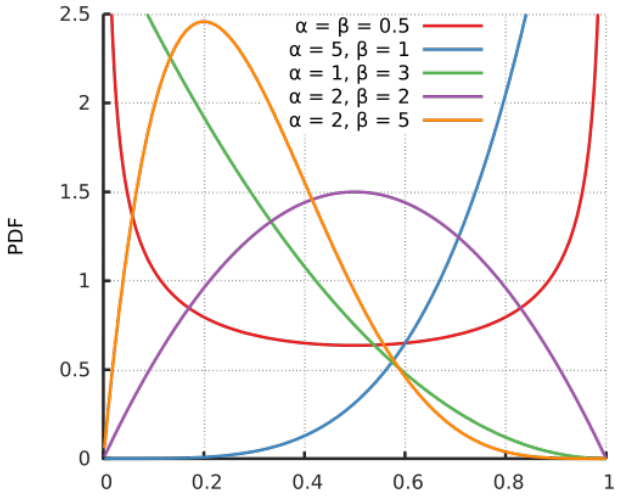

Figure 1: The beta distribution for various values of α and β (source: Wikipedia)

The beta distribution can also be uniquely specified by a mean and a variance, and if these two quantities are specified then the α and β can easily be calculated. The reverse is also true.

For our simulation, we chose to create a beta distribution for each state, for each of the five modeled Presidential elections. For the mean of the distribution in each case, we chose to use the *actual result* of the Presidential election of that state in that year. We chose to calculate the variance for each distribution as the variance of actual election results for each state in question for each of the years examined. Thus, for example, for the state of Hawaii, we determined the variance using the election results from

all five years. For all states, therefore, the variance of the sampling distribution was the same for all five election years. Examination of the variance across states showed a not-quite statistically significant inverse correlation between the population of the state, and the variance of its general election returns. The confidence interval for this correlation was at about the 90th percentile. Nevertheless, we decided not to make variance a function of state population, and stuck just with a function using the existing election returns.

We used the traditional definition of left-versus-right shift in the beta distribution of each state in each year. Democratic leaning states had a left shift in their distribution (mean less than 0.5), and Republican leaning states had a right-shift (mean greater than 0.5). Values ranged from 0.0430 for the District of Columbia in 2016, to 0.7571 for the State of Wyoming that same year. From means and variances, we were able to calculate the α and β for each distribution. Using a mean that was the same as the actual general election result produces results that have the same general “flavor” as the actual election that year.

The act of “conducting an election” involves taking a random draw from the beta distribution for any state in any year. To do this, the process is to take a random draw from a uniform distribution on [0, 1]. Using that value, and the inverse-CDF function of the Beta distribution, we create a random draw on the distribution itself. To increase precision, we initially decided to write the simulation engine for the simulation in Java, and apply a open-source software package from Apache to simulate random draws (see Class Beta Distribution).

Using election data from Wikipedia from 2000 through 2016, we painstakingly copied election results for each state into a comma-separated (CSV) file, indexed by year, state abbreviation and state name. We only included numbers for the major party candidates. For example, in 2000, we excluded results for independent candidate Ralph Nader. Using the R programming language, and the Rstudio IDE, we read the resulting CSV file, and verified our data entry by comparing totals from our file against the known totals. Then we calculated percentage Democratic versus Republican to be used as the means of our statistical distributions. At this point, we could also calculate variances. We calculated the same variance for each state in each of the elections (as discussed above). We then added the electoral votes for each state in each year. Since two censuses had been conducted, in 2000 and 2010, we had to account for changes due to reapportionment. These changes occurred in 2004 and 2012 due to censuses in 2000 and 2010,

respectively. We also calculated the α and β parameters for each resulting beta distribution, and added these to our CSV file.

With data in hand, we were then able to move on to creating the user interface for the simulator. We chose to use HTML and JavaScript so that we might create a web application. We used common HTML labels to inform users, and spinner widgets so that users might have fine tuning capability over the means of the beta distributions at the state level. Initial values for the spinners were the actual election results for that year for the state in question. We allowed the user to also apply a factor between 0.9 to 1.1 to the distribution variances for all states in an election. Lower values would produce results closer to the actual election results, higher values would produce more variant and surprising results. We presented distribution mean selection widgets together with state labels in a tabular format, with six columns of seventeen. This accounted for label and widget for each state, plus the District of Columbia.

For results, we used a D3, color-coded state map (or Choropleth). Example code from Mike Bostock at D3 jump-started us in that regard (see Choropleth). For the map, we needed to replace county-level color coding with a statewide coding. We used the traditional blue coloring for Democratic siding states, and red coloring for Republican siding states.

In addition to the map, we generated tabular results for each state giving state name, Democratic totals, Republican totals, and electoral votes to be awarded by a win by either side. The last table row gave nationwide totals for Democratic and Republican votes, and Democratic and Republican electoral votes. We provided a radio button selection of the year currently being displayed and simulated. We provided five options 2000, 2004, 2008, 2012 and 2016, with the initial value being 2000. We read the previously generated CSV file using JavaScript/D3, and initially presented only actual election results for the years in question. We then added buttons with the options “simulate election” and “restore actual results.”

Peer review of our work at this point yielded the suggestion that we might have a button, or buttons, to left-shift or right-shift the distribution means for all states at once. We took this into account, added these buttons, and hooked them up. One press of either button results in a 0.01 left or right shift of the distribution means of all states. With another added button, the user is able to reset one or more shifts in either direction.

At this point, we ready to begin the process of trying to integrate a simulation engine with our user

interface. We had had some initial doubts regarding our plan to use a Java-based simulation engine that was called from JavaScript. Our plan fell apart at this point. Research indicates that a JavaScript implementation is browser specific, and the ability to call out into a Java class is not generally possible. There are JavaScript interpreters that are written in Java, and have the ability to call into Java. However, these technologies are not generally integrated into browser scripting capability.

What to do? After some reflection, we opted to put the entire simulation into JavaScript, but that led to this problem: The only effective JavaScript modeling of the beta distribution exists in a package that requires a considerable amount of setup to integrate with a server-side application. Our aim with this simulation was to allow a user to start the web-based simulation at their own local host. We then looked at using an inverse-CDF function of the Kumaraswamy distribution (see Kumaraswamy distribution), which is quite similar to the beta distribution. Using the Kumaraswamy, a closed-form of the inverse-CDF is quite easy to calculate. The PDF of the Kumaraswamy appears in figure 2, and as the reader can see, it is quite similar to the Beta distribution.

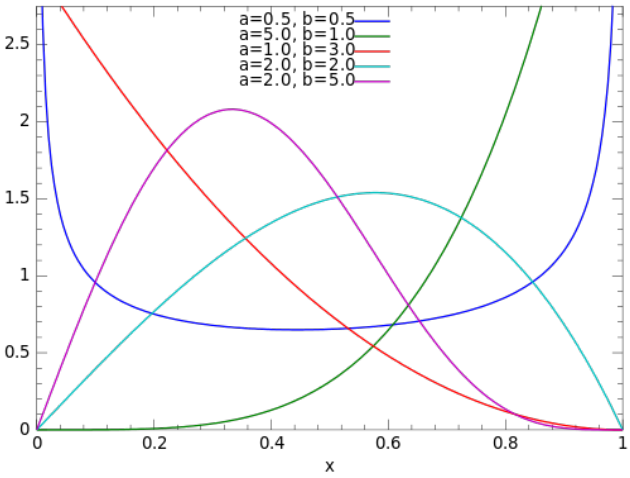

Figure 2: The Kumaraswamy distribution for various values of α and β (source: Wikipedia)

Ultimately, even this effort proved unsuccessful. We decided, eventually, to use the Gaussian, or normal distribution with the means and variances we had chosen. The mass of any of these distributions – for the variances we had chosen – lie tightly gathered around the mean. So substituting the Gaussian distribution for the beta distribution results in insignificant error. We judged one situation worth accounting for, although its occurrence is practically impossible: This is the possibility that a random draw from a normal distribution will result in a value less than zero, or greater than one. To account for this possibility, we set any thus chosen values to zero, or one, respectively. Considering the application, the reader will agree it unlikely that a candidate will receive zero votes in a statewide Presidential election, or receive all the votes.

We borrowed our algorithm for performing the reverse-CDF transform of the Gaussian distribution using

JavaScript code obtained on the Internet (see Filip Zembowicz). This code uses the Ziggurat method. Thus armed, we completed coding of the entire simulation engine in JavaScript.

RESULTS

We were quite excited to see our simulator in action, but quickly noted that the simulation could produce improbable results for the variance levels we had chosen. For example, states of the deep south could easily flip red to blue, and this seemed unlikely given the current political climate in the USA. After several trials, we determined that a constant factor of 0.4, along with the factor applied by the user, resulted in more believable results. With parameters of this nature, for example, it was possible to flip the state of Tennessee from red to blue in the year 2000. This seems plausible, since the Democratic candidate, Alfred Gore Jr., was from Tennessee. He actually lost his home state that year in a close election that saw the White House change hands politically after eight years of Democratic control

The 2000 election was particularly instructive for how the Electoral College results can be in contrast to the popular vote. We were able to obtain scenarios were Gore beat Bush in the popular vote and lost the electoral vote. This happened in the actual result. However, we were also able to produce scenarios where either Bush or Gore won outright in both results, and even scenarios where Bush won the popular vote, and Gore the electoral vote.

The peer suggestion of providing a button to move the whole country left or right for any given election was actually a nice touch. One can play what-if games on election day, and left or right shift the entire electorate on in response to some imagined news event. Additionally, it is possible to see the results of shifting only one state either left or right in any election. In fact, no Republican has ever been elected without winning Ohio (Coffey et al. 2011). By shifting Ohio left by some number of points, the user can determine whether it is possible.

2004 was an election in which George W. Bush won a small majority of the popular vote. His electoral vote win was amplified, and he won the reelection that year. However, his victory was not so large that our simulation was unable to produce a different result in certain trials. It was possible for candidate John Kerry to win a victory in the Electoral College in some scenarios. Such a scenario involved squeaking out a win in Ohio.

Neither of Barack Obama’s wins in 2008 or 2012 were in serious jeopardy of being overturned in any scenario our default simulation could construct. To overturn these results even once in limited trials required us to hit the

whole-country right shift button three times before performing the simulation.

The 2016 has another flavor entirely. Due to the magnitude of her popular vote win, it is difficult to conduct a simulation that does not have Hillary Clinton besting Donald Trump in the popular result. It is difficult, although not as difficult, to construct such a scenario where Clinton bests Trump in the Electoral College. For her to do so requires that our simulation overturn results in all of Michigan, Wisconsin and Pennsylvania. Even then, it must maintain the result for Virginia: No small task overall.

DISCUSSION

Together the most interesting, and the most tedious part of our work was the data entry task of entering election results from Wikipedia. To eliminate error caused by fatigue, we broke this task up into manageable time chunks. Afterward, we verified our work using R and Rstudio by ensuring that vote sum in all states matched the Wikipedia supplied U.S. total for the year in question. Careful attention to detail revealed an error in Wikipedia content, as a vote result from Colorado in 2012 did not match Wikipedia's federal election sources. Calculating Democratic and Republican percentages for each state and in each election was made easy with R, as was calculating election variances. Tedious data entry was again required for entering state electoral vote assignments. Sometime later, we learned we needed to account for an electoral reapportionment event for the 2004 election year.

Creating the HTML/JavaScript interface was an engaging task. Many thanks to Mike Bostock for his sample D3 code for displaying a U.S. map. Only minor modifications were required to turn the map from a U.S. county representation into a whole-state representation. We learned that we could easily display horizontally and vertically aligned HTML elements using an HTML table element. This proved quite useful for both the distribution mean controls, and for the tabular election results.

Despair set in when we realized we would be unable to easily implement a Java-based election simulator. After several hours of frustration, we were able to substitute a Gaussian (normal) distribution for our preferred beta distribution, and continue with a purely JavaScript based solution. There appears to be no discernible difference in the quality of election simulation due to this substitution.

FUTURE WORK

We foresee future possible work in the following areas:

- Integration of a true, JavaScript based beta distribution solution to our simulation random draws

- Integration of the ability to allow a state to select electors by Congressional district; this would currently be necessary for accurate representations of Maine and Nebraska elections
- Simulate election years from further back in history
- Enhance the D3 election result experience to have a result map “fade” in, giving a notional feel of results appearing over time on election night
- Enhance the D3 election result experience to have a result map appear from east to west, notional for the way they actually arrive on election night
- Enhance the D3 election result to have mouse-over capability for each state’s election results
- Supply the same the Electoral College background history in our web application that appears in this report

REFERENCES

Miller, J. (2008), The Reason for the Electoral College, https://www.factcheck.org/2008/02/the-reason-for-the-electoral-college/

Various Authors (2002 – 2018), The Electoral College (United States), https://en.wikipedia.org/wiki/Electoral_College_(United_States)

National Archives and Records Administration, What is the Electoral College?, https://www.archives.gov/federal-register/electoral-college/about.html

Various Authors, United States Presidential Election, 2000, https://en.wikipedia.org/wiki/United_States_presidential_election,_2000

Various Authors, United States Presidential Election 2004, https://en.wikipedia.org/wiki/United_States_presidential_election,_2004

Various Authors, United States Presidential Election 2008, https://en.wikipedia.org/wiki/United_States_presidential_election,_2008

Various Authors, Unites States Presidential Election 2012, https://en.wikipedia.org/wiki/United_States_presidential_election,_2012

Various Authors, United States Presidential Election 2016, https://en.wikipedia.org/wiki/United_States_presidential_election,_2016

2016 Presidential Election Simulation (Clinton vs. Trump): https://www.270towin.com/2016-simulation/

Presidential Election Game: https://270soft.com/us-election-games/president-election-game-2016-infinity/

Blitzstein, Joseph K., and Hwang, Jessica (2014), Introduction to Probability, CRC Press

Various Authors, Beta Distribution, https://en.wikipedia.org/wiki/Beta_distribution

Class Beta Distribution (2003 – 2016), http://commons.apache.org/proper/commons-math/apidocs/org/apache/commons/math4/distribution/BetaDistribution.html

Bostock, Mike (2018), Choropleth, https://bl.ocks.org/mbostock/4060606

Various Authors, Kumaraswamy Distribution, https://en.wikipedia.org/wiki/Kumaraswamy_distribution

Filip Zembowicz, Normal-Distributed Random Variables in JavaScript using the Ziggurat Algorithm, https://www.filosophy.org/post/35/normaldistributed_random_values_in_javascript_using_the_ziggurat_algorithm/

Various Authors, Politics of Ohio, https://en.wikipedia.org/wiki/Politics_of_Ohio

Coffey, Daniel J., John C. Green, David B. Cohen and Stephen C. Brooks. 2011. Buckeye Battleground: Ohio, Campaigns and Elections in the Twenty-First Century. Akron, OH: University of Akron Press