Journal of Humanistic Mathematics

Volume 8 | Issue 1 January 2018

Predicting the Next US President by Simulating the Electoral College

Boyan Kostadinov
New York City College of Technology, CUNY

Recommended Citation

Kostadinov, B. "Predicting the Next US President by Simulating the Electoral College," *Journal of Humanistic Mathematics*, Volume 8 Issue 1 (January 2018), pages 64-93. DOI: 10.5642/jhummath.201801.05 . Available at: https://scholarship.claremont.edu/jhm/vol8/iss1/5

Predicting the Next US President by Simulating the Electoral College

Boyan Kostadinov

Mathematics Department, New York City College of Technology, CUNY, New York, USA
bkostadinov@citytech.cuny.edu

Abstract

We develop a simulation model for predicting the outcome of the US Presidential election based on simulating the distribution of the Electoral College. The simulation model has two parts: (a) estimating the probabilities for a given candidate to win each state and DC, based on state polls, and (b) estimating the probability that a given candidate will win at least 270 electoral votes, and thus win the White House. All simulations are coded using the high-level, open-source programming language R. One of the goals of this paper is to promote computational thinking in any STEM field by illustrating how probabilistic modeling and computer simulations can solve real-world problems for which analytical solutions may be difficult to find.

Keywords: *presidential election forecast, simulation of electoral votes distribution, bootstrap simulation, Bayesian analysis, probability to win the White House*

1. Introduction

I developed the Electoral College simulation model described in this paper in an undergraduate course on Monte Carlo simulations, a couple of weeks before the presidential election on November 6, 2012. In fact, I fully implemented this simulation model in a single lecture on November 6, and the simulation results, based on the final state polls, convinced students and instructor that Barack Obama would win the White House, with about 90% likelihood of success.

I implemented the same simulation model again for the 2016 presidential election. When I did the first simulations in mid-August 2016, the state polls implied more than 99% probability that Clinton would win. However, when I repeated the simulations at the end of September 2016, I got about 57% chance that Clinton would win. Clinton's win probability kept oscillating up and down depending on the incoming state polls, which provide the input data for the simulation model. Having such a great volatility in the win probability was a sign that the state polls, at least for some key battleground states, were probably not very reliable, which turned out to be one of the main problems with pretty much all presidential forecasting models in 2016. In addition, some of the battleground states turned out to be too close to predict, and Trump won them unexpectedly by very small margins. For example, in Michigan (16 electoral votes), Trump won by almost 11,000 votes (1.0704×10^4) votes, in Wisconsin (10 electoral votes), Trump won by a bit more than 22,000 (2.2748×10^4) votes, and in Pennsylvania (20 electoral votes), Trump won by a bit more than 44,000 (4.4292×10^4) votes. Just these three states account for 46 electoral votes. Given that Trump won 304 electoral votes, without these 46 votes, he would have had 258 electoral votes, 12 short of the 270 needed to win the White House. In fact, the total number of these extra Trump votes from the three states, which made all the difference, is curiously 33 short of 77,777.[1]

Perhaps the most important consequence of this simulation model was the realization among my students that computational problem-solving and computer simulations can be applied to any field of science to solve difficult problems for which no exact mathematical solution can be easily found. I strongly believe that computer simulations can empower students of any discipline, from STEM to social science, to find solutions to important, real-world problems of any complexity.

One of the goals of this paper is to substantiate this belief and to hopefully convince the reader that it is worthwhile to invest time and effort into mastering high-level coding and simulation techniques that can be applied to difficult real-world problems that often have no analytical solutions, and where computer simulation is the only means of solving the problem.

[1] All these numbers are based on the official 2016 election results from the Federal Election Commission [10].

I hope that this simulation project would be a small step towards achieving this goal. There is no doubt in my mind that in an increasingly quantitative world, computational problem-solving is already playing a much more prominent role than ever before.

In this paper, I take a leisurely approach to presenting a simulation-based election forecasting model, hoping to make it more accessible to the diverse readership of this journal. Simulating the Electoral College distribution can be divided into two parts:

(a) Estimating the probabilities to win each state and DC, based on the state polls' sizes and percentages, using simulation as well as Bayesian models.
(b) Estimating the probability for a given candidate to win the Electoral College, and thus the White House, based on the previously computed state win probabilities.

2. Simulation Software

The problem of estimating the probability that a presidential candidate will win the Electoral College, and thus the White House, is not an easy problem to solve by simulation, especially from the low-level programming point of view. However, by vectorizing the code and using a functional programming approach, we can implement the simulation that estimates the probability to win the Electoral College in just a few lines of code.

Modern scientific computing allows for quick development and high-level, vectorized, functional programming of simulation models, which can solve very difficult problems in just a few lines of code, unlike the traditional low-level, procedural programming that typically requires hundreds of lines of code to accomplish the same task. The functional programming approach is therefore a natural choice for both educational and research-inspired simulations. One excellent software platform for this purpose is the R language, an open-source computational framework, which combines the best of both worlds, procedural and functional programming. R can be downloaded for free from the R Project [20]. A short introduction to creating simulations and visualizations using R can be found in the papers [3, 18]. A more comprehensive introduction to scientific programming and simulation using R can be found in the books [16, 14, 17], while [30] is a useful R reference.

A more general introduction to probabilistic thinking can be found in [31]. At the more popular level, I refer the interested reader to [22], where one can find an illuminating discussion of public opinion polls, as well as [24], where Nate Silver offers his insight into the art and science of forecasting the future.

3. A Bootstrap simulation for the state win probability

The simulation method that we develop in this section for estimating the state win probabilities is an example of the *bootstrap* method based on resampling of the given poll data. Resampling refers to using the observed poll data to set up a probability distribution representing all possible responses with the corresponding relative frequencies, based on the observed data. We then use this distribution to simulate a large number of hypothetical poll samples, which can be used to make any statistical inferences we desire. This approach is based on using computer simulations and we use R as our statistical programming environment of choice.

Brad Efron is usually credited with inventing the bootstrap method, but Julian Simon was promoting and teaching the bootstrap at least a decade before Efron wrote his famous 1979 paper [6] on the subject. In the preface of *Resampling: The New Statistics*, [26], Julian Simon writes:

> Beneath the logic of a statistical inference necessarily lies a physical process. The resampling methods allow us to work directly with the underlying physical model by simulating it, rather than describing it with formulae. This general insight is also the heart of the specific technique Bradley Efron felicitously labeled 'the bootstrap' ([7]), a device I introduced in 1969 that is now the most commonly used, and best known, resampling method. [...] The method was first presented at some length in the 1969 edition of my book *Basic Research Methods in Social Science* [27].

The bootstrap method has become an increasingly popular alternative to classical statistical inference in both research and education, and in recent years, the interest among researchers and educators has dramatically increased, given the availability of modern statistical programming languages such as R, which makes it easy to quickly implement computer simulations

with pretty much any degree of complexity. For a more pedagogical introduction to the bootstrap method, see [26, 28, 29, 8], and for a more comprehensive mathematical introduction, see [7, 9].

The first component of our simulation model is to estimate the probabilities to win each one of the 50 states and DC, based on state polls' percentages and sample sizes, using the bootstrap method. We take the results of a state poll and estimate through resampling the probability that each candidate will win the majority of votes and collect the state's electoral votes on his/her way to winning to Electoral College, and the White House.

We use the state polling data provided by FiveThirtyEight [11]. Consider a Florida poll conducted by Siena College on September 10-14, 2016. Florida had 29 electoral votes for the 2016 presidential election. The results are based on polling 867 likely voters: 41% for Clinton, 40% for Trump, and 19% for Other. We want to answer the following question:

- Given a state poll, what are the chances that Clinton will get more votes in the election than Trump?

In other words, we want to calculate how likely it is that the poll's predicted winner will be the actual election winner, who gets the electoral votes of that state. Based on the given Florida poll, there is about 62% chance Clinton would win Florida's 29 electoral votes. The sample size is important to the calculations as it impacts the margin of error. Larger sample size gives a smaller margin of error. The margin of error impacts the probability that the poll's predicted winner is correct. For example, Clinton's 41% and Trump's 40% poll percentages, with sample size of 400 yields about 58% probability that Clinton would win Florida.

The key idea behind our simulation model is to use the poll's percentages to define a discrete distribution for the random variable S, which has three possible outcomes: A, B, O, since each voter can vote for either candidate A, B or Other. Thus, we can define the distribution of S as follows:

$$S = \begin{cases} \text{A}, & \text{with probability } p_1 \\ \text{B}, & \text{with probability } p_2 \\ \text{O}, & \text{with probability } p_3, \end{cases} \tag{1}$$

where $\sum_{k=1}^{3} p_k = 1$, and (p_1, p_2, p_3) are the poll's percentages for A, B and Other. Strictly speaking, S represents a random experiment with three pos-

sible outcomes rather than a random variable, because a random variable must have numerical values. Of course, instead of using the names A, B and O, one can also assign any numbers they wish to represent the three possible values. For example, 1,2,3 could be one numerical choice for the possible values of S. For the Florida poll at hand, the distribution of S is specified by the probability vector $(0.41, 0.40, 0.19)$. To create a simulation model, we assume that each voter, in a sample of size $N = 867$, has the same distribution S of voting for A, B or O, and all voters vote independently of each other. Based on these assumptions, we can simulate a single scenario of how every voter in a sample of size N will vote. Simulating from finite, discrete random variables can be coded in R using the function `sample()`, whose signature is the following:

```
sample(states, size=867, replace = TRUE, prob = perc)
```

This function generates a sample of size 867 from the 3 possible values in the vector `states<-c("A","B","O")`, and the sampling is done with replacement. The probabilities for the three possible values come from the poll's percentages `perc<-c(0.41,0.40,0.19)` for A, B and O. Of course, simulating a single scenario is not very useful. In order to get any statistically useful information we must repeat this random experiment thousands of times. This can be accomplished in several ways. If we were using a low-level programming language, we would then simply create a `for loop` and run this random experiment as many times as we wish. Using a high-level programming language like R allows for two other alternative approaches. The first approach is to bundle the code that generates a single random experiment into an R function and then call this function many times, using the `replicate()` R function. An alternative approach is to simulate a matrix of size $n \times N$ from the distribution of S, representing n scenarios for the votes in a sample of size N. Here is a compact implementation of this idea:

```
N<-867 # the sample size
n<-2e4 # number of scenarios
states<-c("A","B","O") # the 3 possible states for every voter
perc<-c(0.41,0.40,0.19) # poll's percentages
sim.polls<-sample(states,size=n*N,replace=TRUE,prob=perc)
dim(sim.polls)<-c(n,N) # reshape into n by N matrix
```

In this approach, we use 20,000 simulations to generate the matrix `sim.polls` with $n = 20,000$ rows and $N = 867$ columns, where each row represents a single simulated poll of size N. Next, we count how many people vote for candidates A and B in every simulated poll, that is, we are counting A's and B's across the rows of the matrix `sim.polls`. This way, the kth row produces a pair of numbers (a_k, b_k) representing the counts for the A's and B', respectively:

$$\text{row}_k \to (a_k, b_k), \quad k = 1, \ldots, n \tag{2}$$

So, we get two vectors `nA` and `nB` containing these counts for all n rows:

$$\texttt{nA} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad \text{and} \quad \texttt{nB} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}. \tag{3}$$

We can generate the vector `nA` in R by applying the `rowSums()` function to the matrix of logical values, obtained from the matrix `sim.polls` of simulated polls, where a value of `TRUE` corresponds to a vote for candidate A; such a matrix of logical values can be obtained from the logical expression `sim.polls == "A"`. The function call `rowSums(sim.polls == "A")` then returns the vector `nA` of numbers representing the vote counts for candidate A across each row of the matrix `sim.polls`. In a similar way, the function call `rowSums(sim.polls == "B")` returns the vector `nB` of vote counts for candidate B from each simulated poll. Finally, we want to compare the counts for candidates A and B in each simulated poll. The comparison operation can be vectorized:

$$\texttt{nA} > \texttt{nB} = \begin{bmatrix} a_1 > b_1 \\ a_2 > b_2 \\ \vdots \\ a_n > b_n \end{bmatrix}. \tag{4}$$

The reason we are vectorizing our math this way is simply because in R all arithmetic and logical operators act element-wise on vectors and matrices. Thus, writing vectorized code is almost the same as doing the corresponding mathematics on paper, which makes it a natural choice for doing numerical mathematics.

The final piece of the simulation model is estimating the probability that more people would vote for A than B during the election. We can estimate this probability from the samples `nA` and `nB` (based on a pool of size N) of the corresponding random variables nA and nB. The single line of vectorized code `mean(nA>nB)`, which takes the mean of the vector of logical values `nA>nB`, estimates the desired probability $\mathbb{P}(nA > nB)$, by computing the relative frequency of the event $nA > nB$. Applying the logical operator $>$ (element-wise) on the two samples `nA` and `nB` results in a vector of logical values, `TRUE`, if the corresponding count for A is indeed greater than the corresponding count for B, or `FALSE` otherwise. It is important to understand what taking the arithmetic average (mean) of a vector of logical values really does. R has the ability to *coerce logical values into the binary digits*, any time an arithmetic operation is performed on logical values. When logical values appear in any algebraic expression, we have: `TRUE` $\to 1$ and `FALSE` $\to 0$. Thus, we have the following:

$$\texttt{mean(nA>nB)} = \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}(a_k > b_k) = \frac{\#(nA > nB)}{n} \approx \mathbb{P}(nA > nB), \tag{5}$$

where we define the indicator function:

$$\mathbb{1}(a_k > b_k) = \begin{cases} 0, & \text{if } a_k > b_k \text{ is } \texttt{FALSE} \\ 1, & \text{if } a_k > b_k \text{ is } \texttt{TRUE}. \end{cases} \tag{6}$$

Here $\sum_{k=1}^{n} \mathbb{1}(a_k > b_k) = \#(nA > nB)$, and $\#(nA > nB)$ is the total number of poll scenarios, out of n, for which more people vote for A than B. Since $\frac{\#(nA>nB)}{n}$ is nothing but the relative frequency of the event that more people vote for A than B, by *the law of large numbers*, we can justify the claim that we are estimating the probability $\mathbb{P}(nA > nB)$.

```
mean(rowSums(sim.polls == "A") > rowSums(sim.polls == "B"))
```

For this Florida poll, the code above gives 62.39% chance that Clinton would win Florida.

4. A Bayesian model for the state win probability

Bayesian inference is the process of fitting a probability model to given data, resulting in fully specified probability distributions for the model parameters.

Thus, the Bayesian approach differs fundamentally from classical statistics, where the unknown population parameters are treated as fixed constants, rather than random variables. However, treating the unknown population parameter as a random variable does not mean we believe it is inherently random, rather it reflects the state of our knowledge about the parameter. In the Bayesian approach, before we have any data, we assign to the unknown model parameter some initial probability distribution, called the *prior distribution*, which is completely subjective and reflects our beliefs about the true value of the unknown parameter. We then update the prior distribution using Bayes' rule to obtain the so called *posterior distribution*, by learning from the data related to our model. The posterior distribution reflects our new, revised beliefs about the unknown parameter, given the collected data. There is a number of excellent texts introducing Bayesian data analysis. We recommend [19] as a hands-on tutorial with R, which builds from introductory examples to advanced applications, while [13, 1] are excellent texts at the more advanced level that also discuss election forecasts.

We can use any discrete prior distribution as a way to capture our initial beliefs. A discrete prior maybe less accurate for representing a continuous parameter but it is straightforward computationally and the posterior distribution is easily derived from Bayes' formula. Thus, a discrete prior could serve as a good introductory example of Bayesian analysis. See [13, 1, 19] for such examples.

Given that the proportion of Clinton voters in the Florida population is a continuous parameter, we now consider a continuous density $g(\theta)$ defined on the interval $(0, 1)$, as the prior distribution, which represents our initial beliefs about the unknown proportion Θ. We introduce a Bayesian model for estimating the probability that a given candidate will win the state, given a state poll data. We consider the same Florida presidential election poll, conducted by Siena College on September 10-14, 2016. A random sample of 867 likely voters is taken, and 41% of the polled voters would vote for Clinton, 40% for Trump, and 19% for a third party candidate or undecided. We assume that the fight for the presidency will be between Clinton and Trump, thus we ignore the votes for the third party candidate. Hence, we reduce the data to 355 potential votes for Clinton, and 347 potential votes for Trump, for a total of 702 votes among the two major party candidates. If we are interested in whether Clinton would win the Florida

election, we can consider a vote for Clinton to be a "success", and a vote for Trump to be a "failure", within that context.

We model the unknown fraction of the voting Florida population that favors Clinton, by the continuous random variable Θ. We assume that our initial state of knowledge about this unknown parameter is expressed by a continuous *beta prior distribution*, which is a convenient family of densities for a proportion. The beta density $g(\theta)$ has a kernel proportional to:

$$g(\theta) \propto \theta^{a-1}(1-\theta)^{b-1},\ 0 < \theta < 1, \tag{7}$$

where the parameters a and b are chosen to reflect the user's prior beliefs about the unknown proportion Θ. We can find estimates for a and b based on statements about the percentiles of the distribution. If we have little or no prior information, we can set $a = 1$ and $b = 1$, a choice that results in the continuous uniform distribution on $(0, 1)$. On the other hand, suppose we believe that the proportion of Clinton voters is equally likely to be smaller or larger than 50%. In addition, we are 90% confident that the proportion is less than 65% for the state of Florida. Statistically speaking, our beliefs are equivalent to saying that the median (50th percentile) and 90th percentiles of the proportion Θ are given by 0.5 and 0.65, respectively. The function `beta.select` in the `LearnBayes` package, developed by Jim Albert [2], is useful for finding the shape parameters of the beta density implied by our prior beliefs. The arguments of 'beta.select' are two lists that specify the two prior percentiles, according to our beliefs. The function returns the values of the implied beta parameters. Here is the R code that implements this procedure:

```
library(LearnBayes) # load LearnBayes library
(shape<-beta.select(list(p=.5,x=.5),list(p=.90,x=.65)))


## [1] 8.95 8.95
```

Thus, our prior beliefs are matched by a beta density with parameters $a = 8.95$ and $b = 8.95$. The beta distribution becomes skewed if $a \neq b$, and large parameters imply more prior information, which is reflected by a narrow prior beta distribution. Using the implied prior beta distribution,

we can compute the prior probability that Clinton would win Florida's popular vote during the election:

$$\mathbb{P}(\Theta > 0.5) = \int_{0.5}^{1} g(\theta)\, d\theta, \tag{8}$$

where we use the normalized beta density $g(\theta)$ with the implied shape parameters a and b. In fact, this probability is already specified by our prior beliefs, given that we believe the median to be 0.5. In general, it is easy to compute this probability in R, given any set of beliefs that specify certain prior percentiles. More specifically, below is the R code that computes the desired probability, using the cdf `pbeta`:

```
1 - pbeta(0.5,shape1=shape[1],shape2=shape[2])

## [1] 0.5
```

Alternatively, we can use the upper-tail of the cdf `pbeta`:

```
pbeta(0.5,shape1=shape[1],shape2=shape[2],lower.tail=FALSE)

## [1] 0.5
```

In order to update the prior distribution, given a state poll data, we need the *likelihood* function $L(\text{data}|\theta)$, which we define as the probability of getting the observed number of $s = 355$ potential votes for Clinton, as the number of "successes", and $f = 347$ potential votes for Trump, as the number of "failures". This is a Binomial model for the given data specified by a total of $N = 702$ independent Bernoulli trials, each having a probability of "success" given by θ. More precisely:

$$L(\text{data}|\theta) \propto \theta^s (1-\theta)^f,\ 0 < \theta < 1. \tag{9}$$

The likelihood function can be identified as the beta density with shape parameters $s+1$ and $f+1$. We can find the *posterior distribution* $g(\theta|\text{data}) = \mathbb{P}(\Theta = \theta|\text{data})$ that the fraction of the voting Florida population in favor of Clinton equals θ, given the data, by applying Bayes' rule.

Note that the prior and the likelihood need to be specified only up to a multiplicative constant. We have the key Bayesian observation:

$$\text{posterior} \propto \text{likelihood} \times \text{prior} \implies g(\theta|\text{data}) \propto L(\text{data}|\theta) \times g(\theta) \tag{10}$$

The posterior distribution is then given by a beta density with parameters $a+s$ and $b+f$, obtained from the product of a beta prior with parameters a and b, and a beta likelihood with parameters $s+1$ and $f+1$:

$$g(\theta|\text{data}) \propto \theta^{a+s-1}(1-\theta)^{b+f-1},\ 0<\theta<1, \tag{11}$$

where $a+s = 363.95$ and $b+f = 355.95$. We can use the R function `dbeta` that implements the beta density to compute the prior, likelihood, and posterior, given that all three are beta distributions. In order to compare the prior and the posterior distributions, we superimpose them on the same graph shown in Figure 1 below.

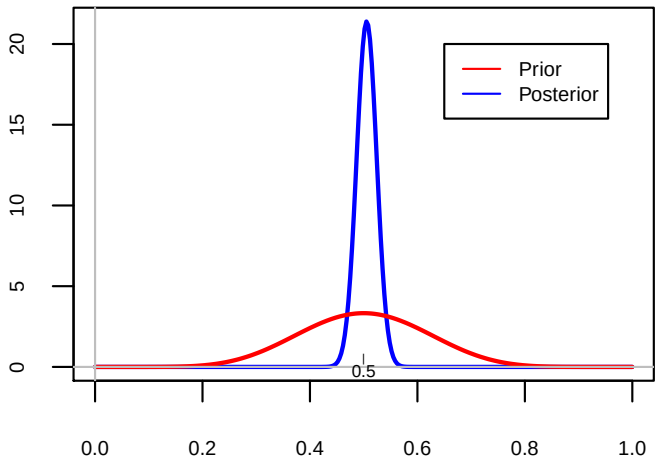

Figure 1: The prior density $g(\theta)$ and the posterior density $g(\theta|data)$ for the proportion of Florida voters who favor Clinton, based on a state poll from September 14, 2016.

This graph was obtained with the following R code:

```
curve(dbeta(x,a+s,b+f),from=0,to=1,n=300,col="blue",xlab="theta")
curve(dbeta(x,a,b),n=300,col="red",add=TRUE)
legend(.8,25,c("Prior","Posterior"),col=c("red","blue"))
```

Using the posterior distribution, we can now compute the posterior probability that Clinton would win Florida's popular vote during the election by answering the question:

- How likely is it that the proportion of Clinton voters during the election would be greater than 50%?

This question is answered by computing the posterior probability $\mathbb{P}(\Theta > 0.5|\text{data})$:

$$\mathbb{P}(\Theta > 0.5|\text{data}) = \int_{0.5}^{1} g(\theta|\text{data})\, d\theta \approx 61.73\%, \tag{12}$$

where $g(\theta|\text{data})$ is the normalized posterior beta with shape parameters $a+s$ and $b+f$. Similar to the corresponding prior probability, we can compute the posterior probability using the beta cdf `pbeta`:

```
pbeta(0.5, a + s, b + f,lower.tail = FALSE)

## [1] 0.6172777
```

The inverse cdf function `qbeta` (quantile function) can be used to construct interval estimates for θ. For example, a 90% interval estimate for θ is obtained by computing the 5th and 95th percentiles of the posterior beta density:

```
(interval<-qbeta(c(0.05, 0.95), a + s, b + f))

## [1] 0.4749154 0.5361797
```

Therefore, a 90% posterior interval estimate for the proportion of Florida voters who favor Clinton is given by $(0.475, 0.536)$.

We can also consider the last eleven Florida polls before the election, collected by FiveThirtyEight [12], which were conducted by various pollsters and organizations between October 20 and November 7, 2016. The polling data show no signs of trending across time for the candidates' proportions, given their margins of error. This suggests that we can consider these polls as repeated random samples from the same underlying population.

If we treat the aggregated results from these additional polling data as additional prior knowledge about the Florida election, then curiously our prior information consists of 7008 votes for Clinton and the same number of 7008 votes for Trump. With this new prior information, we can update the posterior beta distribution, and obtain a probability of 50% that Clinton would win Florida's popular vote.

Finally, we make an adjustment to our polling data by dropping one of the Florida polls conducted by SurveyMonkey between November 1-7, 2016, because it has the lowest grade of C- among the 11 polls we considered earlier. All polls were graded by FiveThirtyEight in terms of the quality of the pollster. In this case, we have a total of 5085 votes for Clinton and 5167 votes for Trump. After updating the posterior beta distribution, we obtain a much smaller probability of 20.92% that Clinton would win Florida's popular vote, which suggests that Clinton did not have a favorable chance of winning Florida, given the aggregated polling data. Similar Bayesian analysis on aggregated state polling data can be done for all key battleground states. More sophisticated Bayesian models for forecasting the US Presidential election are discussed in [13, 1].

In fact, Clinton did not win Florida. She received 49.38% of the votes cast for either Clinton or Trump, while Trump received 50.62% of the Clinton-Trump votes, [32]. This suggests that the Florida election was really too close to predict, and the questionable quality of the state polls, in general, made forecasting even more difficult.

5. Simulating the toss of a biased coin

Flipping a coin is an example of the simplest discrete random experiment, having only two possible outcomes: heads ('success') or tails ('failure'). If we have a fair coin, the two possible outcomes are equally likely and this determines their probabilities of 0.5. Of course, a coin may be biased in which case we can have an arbitrary probability of heads $p = \text{Prob}(\text{'success'})$, which fixes $q = \text{Prob}(\text{'failure'}) = 1 - p$. However, we cannot compute with the names 'success' and 'failure', and that is why we introduce real numbers to represent the possible values; one natural choice is the binary one, that is, we use 1 to represent 'success' and 0 to represent 'failure', thus defining a Bernoulli random variable.

We are interested in simulating a realization of the random variable X on the computer. Being able to simulate the simplest discrete random variable is the first step to simulating much more complex probabilistic models. Simulating X on the computer means to have the ability to generate numbers from the set of possible X values: 0 or 1. However, getting either 0 or 1 at random on the computer is not enough to guarantee a correct simulation of X, since X is characterized not only by its possible values but also by the probabilities of these values to occur. The idea is to think of probabilities as the long run relative frequencies of these events occurring. Therefore, if we simulate X on the computer one million times, we want the fraction of times 1 occurs to be approximately equal to $p = P(X = 1)$ and the fraction of times 0 occurs to be approximately equal to $q = P(X = 0)$. Only then, we can claim that we have correctly simulated the random variable X on the computer.

The R code that simulates a single flip of the biased coin, having a probability of success $p = 0.75$, is given by `(runif(1) < 0.75)`. We get a logical value by comparing the fixed number 0.75, representing the probability of success, with the random number, generated by `runif(1)`, sampled from the standard Uniform distribution $U(0, 1)$. We claim that if `TRUE` is simulated, we can say that we have simulated 'success', otherwise we have simulated 'failure'. Thus, the simulation generates the correct possible values of 1 (`TRUE`) or 0 (`FALSE`), but we still need to make sure that in the long run the relative frequency of the 1's is approximately equal to the theoretical probability $p = 0.75$ of 1 occurring.

The proposed way to simulate the flip of a biased coin guarantees the correct relative frequencies in the long run because of the fact that $\mathbb{P}(U < p) = p$ for $U \sim U(0, 1)$, given that $0 \leq p \leq 1$. We can demonstrate this by generating a sample from X of size 10^6, using the code `sample<-(runif(1e6) < 0.75)`. Here, `sample` is a vector of logical values (of size one million), which represents simulating the flip of the biased coin one million times. More precisely, the R call `runif(1e6)` generates a vector of size one million with random samples from $U(0, 1)$. R compares a vector with a number element-wise, by replicating the number into a vector of the same size. Thus, the resulting `sample` is a vector of logical values of size one million, where `TRUE` represents 'success' and `FALSE` represents 'failure'. Once we have a large sample, it is very easy to compute the relative frequency of 'success' by calling `mean(sample)`, which returns 0.7493.

Simulating the correct possible values and the correct relative frequencies of these values is a general simulation principle, which applies to generating good samples from any random variable. For a comprehensive introduction to simulation techniques, we refer the reader to [23].

6. Simulating the Electoral College by tossing 51 biased coins

In the US presidential election, each state and DC have an assigned number of electoral votes. The candidate who gets the largest number of votes in a given state, receives the corresponding number of electoral votes. A candidate wins the election if he or she receives at least 270 electoral votes, that is, the majority of the total number of 538 electoral votes.

Having the state win probabilities allows us to generate a large sample from the distribution of Clinton's electoral votes by tossing 51 biased coins, for which the 'success' probabilities correspond to Clinton's state win probabilities, as computed in Sections 3 and 4. A single random experiment would be flipping all 51 biased coins at once and adding the electoral votes for those states where the coin flips resulted in Clinton's win. Simulating this random experiment thousands of times then gives a large sample from the distribution of Clinton's electoral votes, which in turn allows us to estimate any statistics of interest, such as the average number of electoral votes won by Clinton, or the probability that Clinton would get at least 270 electoral votes and win the presidency.

When we simulate the outcome of the Presidential Election in 10 key states, considered Clinton's firewall, by simply tossing a fair coin for each one of these states to determine whether Trump or Clinton would win that state, then Clinton's chances of winning the White House turn out to be less than 22%. However, when we use the state polls conducted by the professional pollsters to estimate the state win probabilities, then the projected probability that Clinton would win the White House varies between 50% and 100% depending on the pollsters and the timing of the state polls being conducted. Of course, after the election, it became clear that the professional pollsters got the pre-election state polls spectacularly wrong, and predicting the election results by simply flipping a fair coin in 10 firewall states would have resulted in a much more realistic prediction than using the state polls for estimating the state win probabilities.

I should mention that one of the reviewers suggested that while Monte Carlo simulations are easy to implement in this context, it is possible, given the 51 state win probabilities, to *exactly* compute the probability that a given candidate wins at least 270 electoral votes, by using either polynomial multiplication in a computer algebra system or dynamic programming.

6.1. Electoral predictions from state polls

The simulated electoral map in Figure 2 is based on state polls from September 26, 2016. The map shows one particular realization of the likely Republican states in red and the likely Democratic states in blue. The simulation that generated Figure 2 is based on Clinton's state win probabilities, estimated by the Bayesian model; these are all summarized in Appendix A.

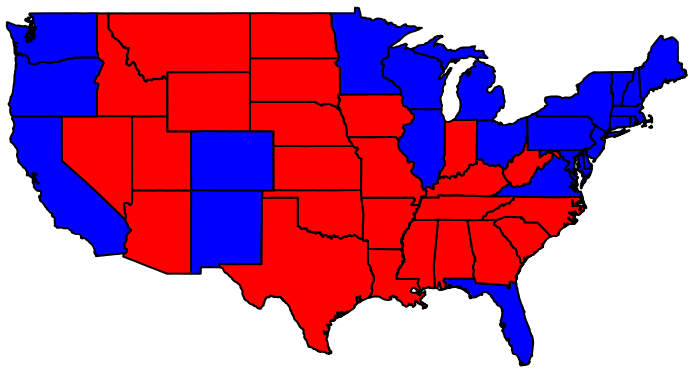

Figure 2: One possible realization of the 2016 electoral map, based on pre-election polls.

We are now ready to implement the final part of our simulation model. For a given state, let p_k be the probability that Clinton wins the kth state. We can think of whether Clinton wins the kth state or not as flipping a biased coin having a probability of 'success' p_k, where 'success' means getting the majority of votes in that state's election. We assume that all 51 such random Bernoulli experiments are independent of each other, and have a different probability of 'success'. Let $p = (p_1, \ldots, p_{51})$ be the vector of state win probabilities, computed using the techniques in Sections 3 and 4. Imagine now, flipping 51 different, biased coins, having probabilities of 'success' specified by the probability vector p. The outcome of this experiment is the essence of the presidential election.

In order to get the statistics for Clinton winning the election, we are most interested in understanding the Electoral College distribution. We can simulate each one of the biased coin flips as $(u_k < p_k)$, where u_k is a single sample from the standard Uniform distribution $U(0, 1)$, and p_k is the probability of 'success' for the kth flip, that is the probability Clinton wins the kth state. If 'success' is simulated then the state is won and its electoral votes v_k go to Clinton. Therefore, the total number of electoral votes (eVotes) that go to Clinton for any given realization of this random experiment is given by:

$$\text{eVotes} = \sum_{k=1}^{51} v_k \mathbb{1}(u_k < p_k), \tag{13}$$

where $\mathbb{1}(u_k < p_k)$ is the indicator function, which returns 1 if Clinton wins the kth state, and 0 otherwise. In order to simulate a large sample of eVotes, we define the R function `eVotes()`, which implements (13) by simulating a single election experiment and returning the total number of electoral votes won by Clinton. Of course, a single realization of the total number of electoral votes won does not carry any statistically useful information. However, we can replicate this random experiment many times by using the R function `replicate()` to generate a large sample `eVotesSample` from the electoral votes for Clinton. In Figure 3, we show the density histogram of this distribution, along with a summary of sample statistics. Here is the R code needed for this purpose:

```
eVotes<-function() sum(Votes*(runif(51)<WinProbBayes))
eVotesSample<-replicate(1e4,eVotes())
# density histogram of electoral votes
hist(eVotesSample,breaks=50,freq=F,col="blue")
summary(eVotesSample) # sample statistics
```

```
##    Min. 1st Qu.  Median    Mean   3rd Qu.    Max.
##   181.0   258.0   274.0   273.9   291.0   361.0
```

In the code above, `Votes` is the vector of electoral votes for each state and DC, and `WinProbBayes` is the vector of state win probabilities, based on the Bayesian model, see Appendix A. Once we have a large sample from the

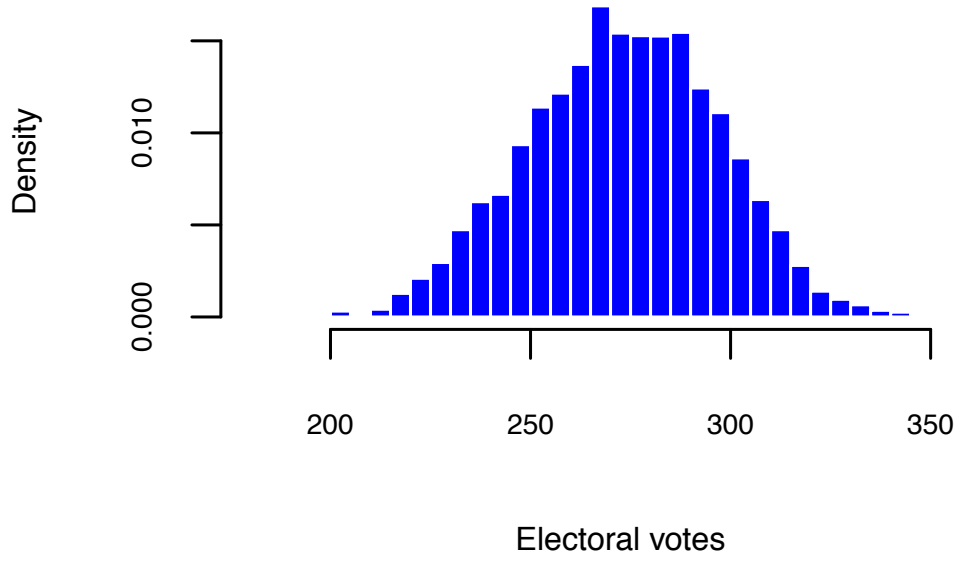

Figure 3: Distribution of electoral votes for Clinton.

Electoral College distribution, we can compute the relative frequency that Clinton will receive at least 270 electoral votes:

$$\mathbb{P}(\text{eVotesSample} \geq 270) \approx \texttt{mean(eVotesSample >= 270)} \approx 0.5713. \quad (14)$$

This is an example of a vectorized computing, where all operations are done with vectors and functions applied to them. In the code, `eVotesSample` is a vector of size ten thousand and the logical expression `eVotesSample >= 270` creates a vector of logical values, `TRUE`, if the inequality is satisfied, and `FALSE` otherwise. Taking the mean of a logical vector coerces `TRUE` to 1 and `FALSE` to 0. In (15), we show how the arithmetic mean of a vector of logical values is equivalent to computing the relative frequency that the condition `(eVotesSample >= 270)` is satisfied.

$$\texttt{mean(eVotesSample >= 270)} = \frac{1}{N}\sum 1's = \frac{\#1's}{N}, \quad (15)$$

where $N =$ `length(eVotesSample)`. This is a key insight for estimating probabilities based on computing relative frequencies, which is justified by the law of large numbers.

For the state polls, as of September 26, 2016, we computed a single number of 57.13% for the probability that Clinton would win the White House. However, this probability estimator is a random variable itself, so if we compute it again and again, we will keep getting different values, reflecting its random nature. Generating a large sample from the distribution of the probability estimator represents its *sampling distribution*. The spread of this sampling distribution, captured by its sample standard deviation, is a measure for the sample error of our estimate. Generating this sampling distribution can be implemented using again the `replicate()` function, but this goes beyond the scope of the article.

As of September 26, 2016, Nate Silver's estimate for the same probability was 54.8%. However, his methodology involves a number of poll adjustments, based on the quality of pollsters, in addition to using a more sophisticated simulation model. More details about his methodology can be found in [11].

Hillary Clinton's chances of winning the White House, estimated by FiveThirtyEight using pre-election state polls, were moving up and down in time like a big wave, oscillating between 50% and 90% during the last 3 months before the election: from 49.9% on July 30 up to 89.2% on August 14, then down to 54.8% on September 26, then up again to 88.1% on October 17, then down again to 64.5% on November 4. On Election Day, November 8, 2016, FiveThirtyEight predicted a 71.4% chance for Hillary Clinton winning the White House. See Figure 4 and [11]. Such a great volatility in the estimated chances of winning was later considered to be a sign that some state polls were not reliable and they did not capture very well the actual distributions of Trump and Clinton voters in some key states.

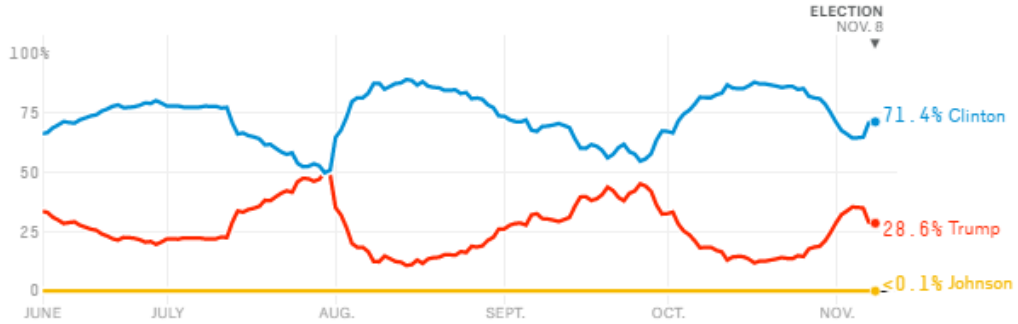

Figure 4: Winning probabilities for Trump and Clinton. Source: FiveThirtyEight [11]

6.2. Electoral Predictions from Flipping a Fair Coin

The simulated electoral map in Figure 5 is based on flipping a fair coin in 10 battleground states, and it shows one particular realization of the likely Republican states in red and the likely Democratic states in blue. The 10 states are as follows: *Colorado, Florida, Nevada, New Hampshire, North Carolina, Pennsylvania, Virginia, Wisconsin, Michigan and Minnesota.*

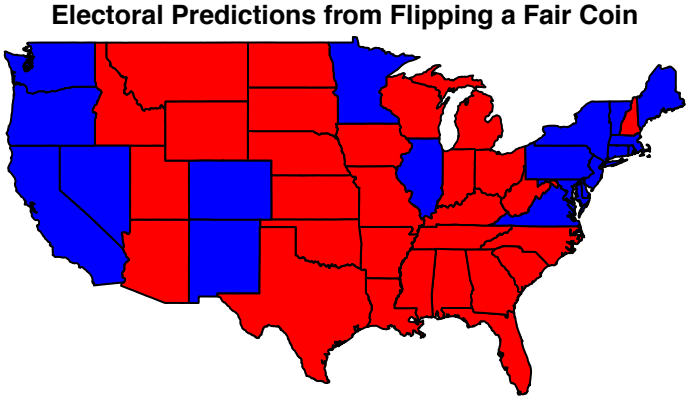

Figure 5: One realization of the electoral map based on tossing a fair coin in 10 key states.

We assume all remaining states and DC to be readily predictable whether they will vote blue or red, and we assign a probability of one for Clinton to win any historically blue state, and a probability of zero for her to win any historically red state. In the case of Ohio and Iowa, despite the fact that Obama won them in 2012, we assigned probabilities of zero for Clinton to win them because the pre-election state polls were consistently giving Clinton negligible chances of winning them.

In Figure 6, we generated a sample of size 10^4 from the distribution of Clinton's electoral votes, using our simulation model, by assuming that the outcome of the election in the specified 10 battleground states is determined by tossing a fair coin, while in the other states the probability is fixed to be either zero or one, depending on whether the states are historically red or blue.

Below, we give the sample summary statistics of the simulated sample of Clinton's electoral votes:

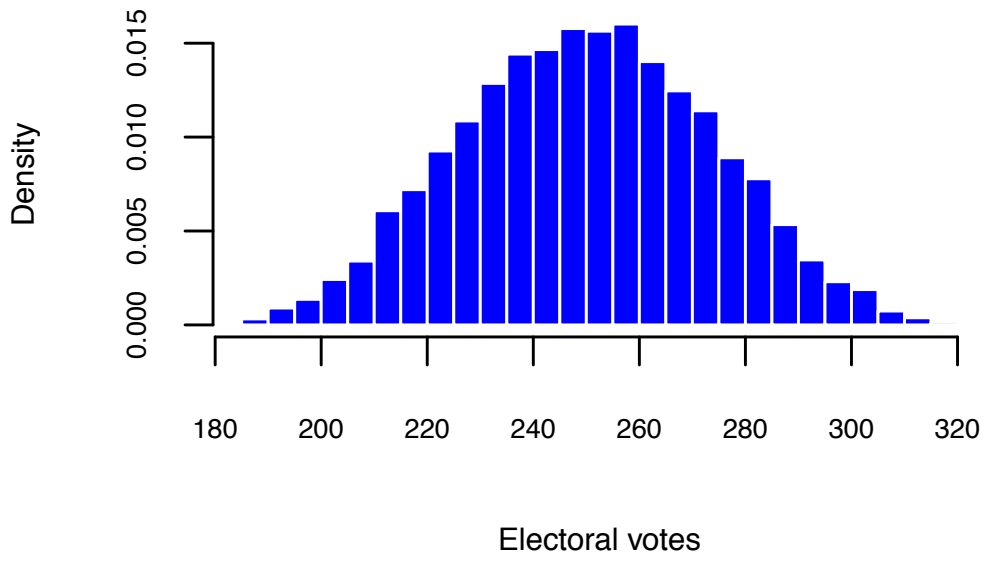

Figure 6: Clinton's electoral votes based on tossing a fair coin in 10 key states.

```
##    Min. 1st Qu.  Median   Mean   3rd Qu.    Max.
##   185.0   234.0   251.0   250.7   268.0   317.0
```

With these assumptions, the probability that Clinton would win the Electoral College is 21.83%.

7. United States Presidential Election Results 2016

From the 538 electoral votes, Trump won 304 and Clinton won 227 in the presidential election of 2016. In state-by-state tallies, Trump earned 306 pledged electors, Clinton 232. They lost respectively 2 and 5 votes to faithless electors. Trump carried 30 states plus Maine's 2nd congressional district, which brought in one extra electoral vote, while the remaining 3 electoral votes from Maine went to Clinton, who carried 20 states plus DC. Note that Maine and Nebraska award two electoral votes to the statewide winner and a single electoral vote to the winner of each congressional district. Figure 7 shows the official 2016 election map, where the red states are won by Trump and the blue states are won by Clinton.

By winning more than 270 electoral votes Donald Trump became the 45th President of the United States, despite the fact that he lost the popular vote by close to 2.9 million votes. More precisely, Trump's popular vote was 62,984,825, while Clinton's popular vote was 65,853,516, see [32].

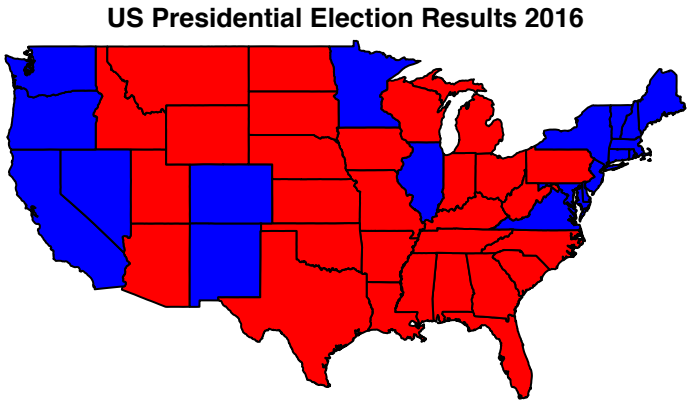

Figure 7: The 2016 presidential election map: blue states won by Clinton, red by Trump.

Journalists, academics, and other political forecasters all made mistakes, some bigger than others, in predicting Trump's chances of winning the presidency during the campaign. I myself was convinced that Clinton would win, given the state polls. I recommend the articles by Nate Silver [25], the Editor-in-Chief of *FiveThirtyEight*, Natalie Jackson [15], the Senior Polling Editor at *The Huffington Post*, and Nate Cohn [4], from *The New York Times*, reflecting on their predictive models, as well as the reasons, as they see them, why key state polls underestimated Trump's support in the decisive Rust-Belt region.

8. Conclusion

The main goal of this article is to illustrate the power of computational thinking and computer simulations in the context of the US Presidential Election by simulating the Electoral College distribution and computing the probability of winning the White House. We build the simulation model using actual state polling data from FiveThirtyEight. The state polls are used as an input for the simulation model and they are used to estimate the probability that a given candidate would win the state during the actual election.

Of course, if the state polls are wrong then the predictions will not be reliable, and this is exactly what happened in the 2016 presidential election. In particular, the state polls in the key firewall states of Clinton did not capture the correct distributions of Trump and Clinton voters, which led to a grossly overestimated probability for Clinton to win at least 270 electoral votes and the White House. Most of the key players in this prediction business had estimated between 85% and 99% probability that Clinton would win the election, based on the state polling data. A notable exception was FiveThirtyEight, which gave Clinton 71.4% chance of winning on election day.

After Trump's historic upset, it became clear that the state polls were not reliable. We used our simulation model to show that if we ignore the pre-election state polls in ten of Clinton's firewall states and flip a fair coin instead, in each one of them, in order to determine who wins the state, then Clinton's chances of winning the presidency drop below 22%. More sophisticated Bayesian models, such as Bayesian Kalman filters, along with poll adjustments and ranking of pollsters, similar to what FiveThirtyEight did, could further improve the predictive power of the election forecast.

It is important to emphasize that the election forecast based on our simulation model is as good as the quality of the state polls, which provide the input data for the model. In addition, the election forecast reflects the likely voter sentiments only at a particular moment in time when the state polls are conducted, and in order to get a more accurate picture one needs to keep updating their forecasts any time new state polling data becomes available, which fits well with the Bayesian point of view.

A personal endnote

With this paper, I want to encourage instructors to invest time and effort into mastering high-level coding and simulation techniques. But I also hope to engage the readers with important matters of citizenship at a time of democracy in crisis. According to the Pew Research Center [5], only about 55.7% of the U.S. voting-age population (VAP) cast ballots in the 2016 presidential election. Based on data from the Census Bureau, there were 245.5 million Americans ages 18 and older in November 2016, about 157.6 million of whom reported being registered to vote. According to figures compiled by the Clerk of the U.S. House of Representatives, the number of votes tallied in the 2016 Presidential election was 136.8 million,

20.8 million fewer than the reported number of people registered to vote. **More importantly, about 100,000,000 eligible voters could not be bothered to vote in the 2016 election.** Opinion polls, and thus any forecasts based on them, would be worthless if citizens do not take their responsibilities seriously and cannot be bothered to participate in the democratic process and cast their votes on election day. The 55.7% VAP turnout in the 2016 election puts the U.S. behind most of its peers in the Organization for Economic Cooperation and Development (OECD), where most members are highly developed, democratic nations. In terms of the voter turnout rate, the U.S. placed 28th, given the most recent elections in each of the 35 OECD member states. The highest voter turnout rates among the OECD nations were reported in Belgium (87.2%), Sweden (82.6%) and Denmark (80.3%).

Register to vote at Rock the Vote [21].

Acknowledgements

I thank the editors for being so responsive and willing to offer help and support. I also thank the anonymous reviewers who promptly offered their constructive criticism and suggestions for improvement, which undoubtedly improved the quality of the paper.

A. Clinton's state win probabilities

In Sections 3 and 4, we offered two approaches to estimating the probability that a presidential candidate will win a given state, using state polls. We applied both approaches to polling data from all 50 States and DC. We used state polls data from FiveThirtyEight [11] to compute the state win probabilities `WinProbSim` and `WinProbBayes` (given as percentages) for Hillary Clinton to win the corresponding states, based on the bootstrap and Bayesian models, developed in Sections 3 and 4, respectively. In the table below, `Votes` represents the electoral votes for each state and DC, and in the columns `Clinton` and `Trump`, we have the polling percentages taken from FiveThirtyEight pre-election state polling data, as of September 26, 2016, where in column `Sample` we give the sample size of the poll. The last column `CV` contains the actual electoral votes that Clinton won in 2016.

	State	Votes	Sample	Clinton	Trump	WinProbSim	WinProbBayes	CV
1	Alabama	9	4100	33	57	0.0	0.0	0
2	Alaska	3	500	29	39	0.2	0.4	0
3	Arizona	11	649	38	40	26.6	28.4	0
4	Arkansas	6	831	34	55	0.0	0.0	0
5	California	55	1055	47	31	100.0	100.0	55
6	Colorado	9	784	41	42	36.8	37.8	9
7	Connecticut	7	1000	50	35	100.0	100.0	7
8	Delaware	3	637	43	33	99.9	99.8	3
9	DC	3	486	81	14	100.0	100.0	3
10	Florida	29	867	41	40	61.9	61.8	0
11	Georgia	16	638	40	47	2.5	2.9	0
12	Hawaii	3	460	64	33	100.0	100.0	4
13	Idaho	4	602	23	44	0.0	0.0	0
14	Illinois	20	700	45	39	96.0	95.7	20
15	Indiana	11	600	36	43	2.3	2.8	0
16	Iowa	6	612	37	44	2.3	2.8	0
17	Kansas	6	7769	37	44	0.0	0.0	0
18	Kentucky	8	732	29	52	0.0	0.0	0
19	Louisiana	8	905	35	45	0.0	0.0	0
20	Maine	3	779	42	39	82.0	81.8	3
21	Maryland	10	514	58	25	100.0	100.0	10
22	Massachusetts	11	700	47	34	100.0	100.0	11
23	Michigan	16	600	38	35	80.6	80.3	0
24	Minnesota	10	625	46	39	97.2	97.1	10
25	Mississippi	6	823	43	46	17.3	17.9	0
26	Missouri	10	1087	37	46	0.0	0.1	0
27	Montana	3	999	31	44	0.0	0.0	0
28	Nebraska	5	983	32	42	0.0	0.0	0
29	Nevada	6	704	40	43	18.2	19.4	6
30	New Hampshire	4	737	39	37	73.0	72.1	4
31	New Jersey	14	800	47	43	88.4	88.2	14
32	New Mexico	5	1103	40	31	100.0	100.0	5
33	New York	29	676	52	31	100.0	100.0	29
34	North Carolina	15	1024	43	45	23.8	24.3	0
35	North Dakota	3	400	32	43	0.4	0.6	0
36	Ohio	18	802	39	44	5.2	6.2	0
37	Oklahoma	7	515	36	51	0.0	0.0	0
38	Oregon	7	517	38	25	100.0	100.0	7
39	Pennsylvania	20	771	45	44	60.9	61.9	0
40	Rhode Island	4	800	44	41	81.9	82.0	4
41	South Carolina	9	1247	38	53	0.0	0.0	0
42	South Dakota	3	809	29	43	0.0	0.0	0
43	Tennessee	11	1021	31	51	0.0	0.0	0
44	Texas	36	700	36	42	2.9	3.7	0

45	Utah	6	820	25	34	0.1	0.0	0
46	Vermont	3	600	47	26	100.0	100.0	3
47	Virginia	13	1237	45	37	99.9	99.9	13
48	Washington	8	505	38	28	99.8	99.7	12
49	West Virginia	5	600	30	57	0.0	0.0	0
50	Wisconsin	10	677	41	38	80.8	81.6	0
51	Wyoming	3	402	19	54	0.0	0.0	0

References

[1] Jim Albert, *Bayesian Computation with R*, 2nd edition, Springer, New York, 2009.

[2] Jim Albert, *LearnBayes*, R package; available at https://cran.r-project.org/web/packages/LearnBayes/index.html, accessed on June 11, 2017.

[3] Nadia Benakli, Boyan Kostadinov, Ashwin Satyanarayana, and Satyanand Singh, "Introducing computational thinking through hands-on projects using R with applications to calculus, probability and data analysis", *International Journal of Mathematics Education in Science and Technology*, Volume **48**, issue 3 (2017), pages 393-427; doi:10.1080/0020739X.2016.1254296

[4] Nate Cohn, 'A 2016 Review: Why key state polls were wrong about Trump', *The New York Times*, May 31, 2017. Available at https://www.nytimes.com/2017/05/31/upshot/a-2016-review-why-key-state-polls-were-wrong-about-trump.html, accessed on June 11, 2017.

[5] Drew DeSilver, "U.S. trails most developed countries in voter turnout", Pew Research Center report, May 15, 2017. Available at http://www.pewresearch.org/fact-tank/2017/05/15/u-s-voter-turnout-trails-most-developed-countries, last accessed on January 23, 2018.

[6] Bradley Efron, "Bootstrap methods: Another look at the jackknife", *The Annals of Statistics*, Volume **7** (1979), pages 1-26.

[7] Bradley Efron, *The Jackknife, the Bootstrap, and Other Resampling Plans*, SIAM Monograph 38, Society for Industrial and Applied Mathematics, Philadelphia, 1982.

[8] Bradley Efron, ‘ ‘Computer-intensive methods in statistics”, *Scientific American*, May 1983, pages 116-130.

[9] Bradley Efron and Robert J. Tibshirani, *An Introduction to the Bootstrap*, Chapman and Hall/CRC, New York, 1993.

[10] Federal Election Commission, *Official 2016 Presidential General Election Results*, January 30, 2017. Available at https://transition.fec.gov/pubrec/fe2016/2016presgeresults.pdf, accessed on January 23, 2018.

[11] FiveThirtyEight.com, ‘2016 Election Forecast: Who will win the presidency?’, November 8, 2016. Available at https://projects.fivethirtyeight.com/2016-election-forecast/, accessed on January 23, 2018.

[12] FiveThirtyEight.com, ‘2016 Election Forecast: Who will win Florida?’, November 8, 2016; available at https://projects.fivethirtyeight.com/2016-election-forecast/florida/, accessed on June 11, 2017.

[13] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, Donald B. Rubin, *Bayesian Data Analysis*, 3rd edition, Chapman and Hall/CRC, Boca Raton, FL, 2013.

[14] Garrett Grolemund, *Hands-On Programming with R*, O’Reilly Media, Sebastopol, CA, 2014.

[15] Natalie Jackson, ‘Why HuffPost’s presidential forecast didn’t see a Donald Trump win coming’, *The Huffington Post*, November 10, 2016. Available at http://www.huffingtonpost.com/entry/pollster-forecast forecast-donald-trump-wrong_us_5823e1e5e4b0e80b02ceca15, last accessed on January 23, 2018.

[16] Owen Jones, Robert Maillardet and Andrew Robinson, *Introduction to Scientific Programming and Simulation Using R*, CRC Press, 2nd edition, Boca Raton, FL, 2014.

[17] Robert Kabacoff, *R in Action: Data Analysis and Graphics with R*, Manning Publications, Shelter Island, NY, 2015.

[18] Boyan Kostadinov, "Simulation insights using R", *PRIMUS: Problems, Resources, and Issues in Mathematics Undergraduate Studies*, Volume **23**, issue 3 (2013), pages 208-223; doi:10.1080/10511970.2012.718729

[19] John K. Kruschke, *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan*, 2nd edition, Academic Press, London, 2015.

[20] R Core Team (2017). R: A language and environment for statistical computing. *R Foundation for Statistical Computing*, Vienna, Austria. https://www.R-project.org/, accessed on June 11, 2017.

[21] Rock the Vote: https://www.rockthevote.com/, accessed on June 11, 2017.

[22] Jeffrey Rosenthal, *Struck by Lightning: The Curious World of Probabilities*, Harper Collins, Toronto, 2005.

[23] Sheldon Ross, *Simulation*, 5th edition, Academic Press, San Diego, CA, 2013.

[24] Nate Silver, *The Signal And The Noise*, The Penguin Press, New York, 2012.

[25] Nate Silver, 'The Real Story of 2016', *FiveThirtyEight.com*, January 19, 2017. Available at http://fivethirtyeight.com/features/the-real-story-of-2016/, last accessed on January 23, 2018.

[26] Julian Simon, *Resampling: The New Statistics*, 2nd edition 1997; available at http://www.resample.com/intro-text-online/, last accessed on January 23, 2018.

[27] Julian Simon, *Basic Research Methods in Social Science*, Random House, New York, 1969.

[28] Julian Simon and Alan Holmes, "A really new way to teach probability and statistics", *The Mathematics Teacher*, Volume **LXII** (April 1969), pages 283-288.

[29] Julian Simon, David T. Atkinson, and Carolyn Shevokas, "Probability and statistics: Experimental results of a radically different teaching method", *The American Mathematical Monthly*, Volume **83** (November 1976), pages 733-739.

[30] Paul Teetor, *R Cookbook*, O'Reilly Media, Sebastopol, CA, 2011.

[31] Henk Tijms, *Understanding Probability*, 3rd ed., Cambridge University Press, New York, 2012.

[32] Wikipedia contributors, United States presidential election 2016, *Wikipedia, The Free Encyclopedia*; available at https://en.wikipedia.org/wiki/United_States_presidential_election,_2016, last accessed on January 23, 2018.