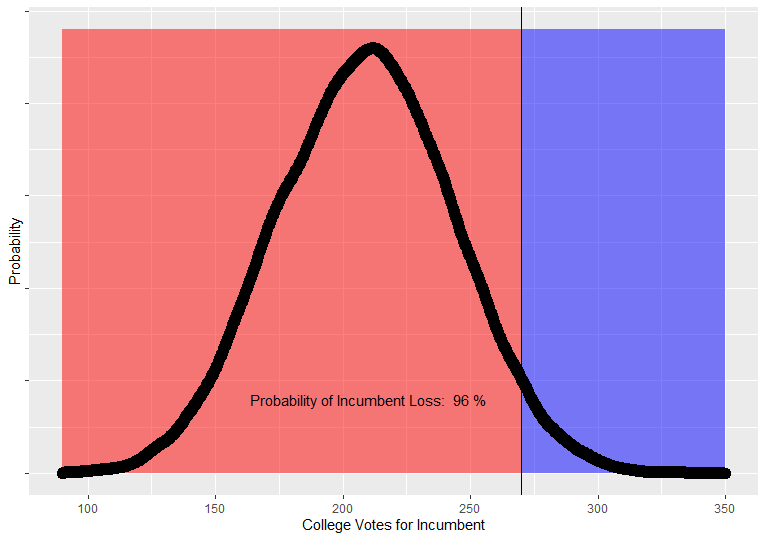
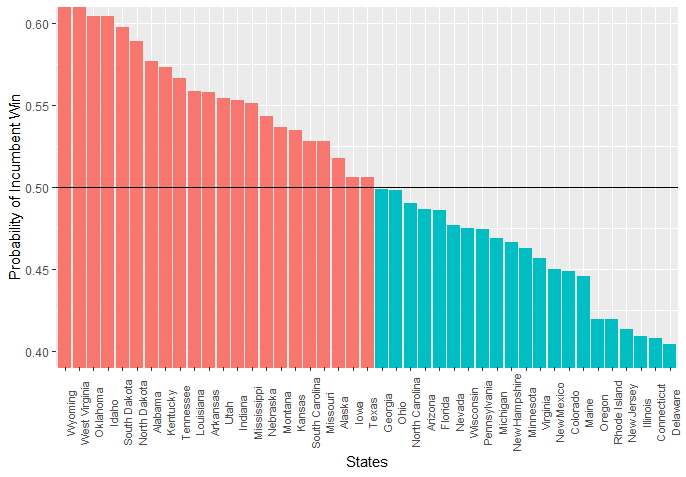
Making your own Presidential Election Forecast: Begginners’ Tutorial in R

Jack Collins

27/12/2020

## Introduction

Come Presidential Election season in the USA, we most often hear that candidate A is X number of point ahead of their opponent. This is typically just the number of percentage points different between the two parties in national polls. Of course, if you think about it, you’ll realize this means very little for who will win the Presidency, because this doesn’t account for the Electoral College - the means by which the States select the President.

In this article, we’ll demonstrate a simple means of taking polling data, and converting it into a metric which does take the Electoral system into account:

We’re going to calculate the probability of the incumbent’s electoral victory, based on State polls.

It is timely for us to use the polling data on the 2020 Presidential election as a test.

Here’s our raw data source: <https://projects.fivethirtyeight.com/2020-election-forecast/>

Note that we shouldn’t expect our result to be an accurate forecast, because most validated forecasts involve some very sophisticated weighting and methodological choices we are not using. We’re just trying to show what we could expect if the polls by State were representative, which is a bit more indicative of the probable result than national polling averages.

Now we step through our R pipeline…

library(dplyr)  
library(ggplot2)  
  
## Set the date used to filter out polls taken before this day  
AFTER\_DATE = "01/09/20"  
  
## Get raw data set.   
rawData = read.csv(url("https://projects.fivethirtyeight.com/polls-page/president\_polls.csv"), na.strings=c("", "NA"))  
  
## Filter out responses for independent candidates and polls which were not state specific.   
  
  
subSet = filter(rawData, !is.na(state) & answer %in% c("Biden","Trump")) %>%  
 ## Add derived columns which is the number of individuals who responded for each row  
 mutate(votes = sample\_size \* (pct/100)) %>%  
 ## Filter out the polls taken after the end date  
 filter(end\_date > AFTER\_DATE) %>%  
 ## Order the table so that each pair of rows for results from the same poll appear paired together.   
 group\_by(question\_id)   
  
## There are a number of polls that only have one response in the data, this inflates Republican numbers.   
## We filter to only include polls with have responses for both candidates.   
  
questionIDFilter = subSet %>% summarize(count = n()) %>% filter(count == 2)  
  
subSet = filter(subSet, question\_id %in% questionIDFilter$question\_id )  
  
Data = summarize(group\_by(subSet, state, answer), total\_votes = sum(votes))  
  
head(Data,10)

## # A tibble: 10 x 3  
## # Groups: state [5]  
## state answer total\_votes  
## <chr> <chr> <dbl>  
## 1 Alabama Biden 47287.  
## 2 Alabama Trump 69891.  
## 3 Alaska Biden 20368.  
## 4 Alaska Trump 24097.  
## 5 Arizona Biden 182727.  
## 6 Arizona Trump 163472.  
## 7 Arkansas Biden 28936.  
## 8 Arkansas Trump 42601.  
## 9 California Biden 617337.  
## 10 California Trump 348505.

## We need to convert our data into a table where each row is a state and our   
## columns show the polling vote share for each   
## candidate. Then, we derive columns for statistical data such as the   
## standard deviation and the probability of the incumbent  
## making 50% or more votes from the population, given what we know of that   
## population from our sample.   
  
bidenVotes = filter(Data, answer == "Biden")  
trumpVotes = filter(Data, answer == "Trump")  
  
pollingData = merge(x = bidenVotes, y = trumpVotes, by = "state", all = TRUE) %>%   
 select(state, total\_votes.x, total\_votes.y) %>%   
 rename(biden\_votes = total\_votes.x, trump\_votes = total\_votes.y) %>%  
 mutate(sample\_size = biden\_votes + trump\_votes) %>%  
 mutate(incumbent\_pct = trump\_votes / sample\_size)   
  
## In order to calculate standard deviation, and therefore the probability of   
## an incumbent win in each state,  
## We have to pass to the 'sd' function, an array of 0s and 1s which represent   
## every individuals response, with 0 being a vote for the challenger and 1   
## being a vote for the incumbent.  
  
pollingData$std = 0.0  
for (row in 1:nrow(pollingData))  
{  
 pollingData[row, "std"] = sd(c(rep(0, times = pollingData$biden\_votes[row]) , rep(1, times = pollingData$trump\_votes[row])))  
}  
pollingData$incumbent\_win\_probability = pnorm(0.5, mean = pollingData$incumbent\_pct, sd = pollingData$std, lower.tail = FALSE)  
pollingData$incumbent\_loss\_probability = 1 - pollingData$incumbent\_win\_probability  
  
  
head(select(pollingData,state, incumbent\_pct,std,incumbent\_win\_probability),10)

## state incumbent\_pct std incumbent\_win\_probability  
## 1 Alabama 0.59645272 0.4906110 0.57792860  
## 2 Alaska 0.54193372 0.4982441 0.53353659  
## 3 Arizona 0.47219054 0.4992267 0.47778838  
## 4 Arkansas 0.59551263 0.4907960 0.57714993  
## 5 California 0.36082989 0.4802414 0.38598770  
## 6 Colorado 0.40922698 0.4916922 0.42676610  
## 7 Connecticut 0.36218052 0.4806318 0.38715322  
## 8 Delaware 0.35851775 0.4795693 0.38398953  
## 9 District of Columbia 0.09893486 0.2985700 0.08958959  
## 10 Florida 0.49282114 0.4999488 0.49427171

## Importing data on electoral college numbers. I made a simple CSV file with  
## The name of each state and their number of electoral college votes.   
  
electoral\_colleges = read.csv("Electoral\_College\_Votes\_Per\_State.csv", na.strings=c("", "NA")) %>%  
 rename(state = ï..State)   
election\_forecast\_data = merge(x = electoral\_colleges, y = pollingData %>% select(state, incumbent\_win\_probability) , by = "state", all = TRUE) %>%  
 filter(!is.na(Number\_Of\_Votes))  
  
head(election\_forecast\_data,10)

## state Number\_Of\_Votes incumbent\_win\_probability  
## 1 Alabama 9 0.57792860  
## 2 Alaska 3 0.53353659  
## 3 Arizona 11 0.47778838  
## 4 Arkansas 6 0.57714993  
## 5 California 55 0.38598770  
## 6 Colorado 9 0.42676610  
## 7 Connecticut 7 0.38715322  
## 8 Delaware 3 0.38398953  
## 9 District of Columbia 3 0.08958959  
## 10 Florida 29 0.49427171

## Election simulations. We use the probability of incumbent win for each state   
## to make a series of weighted coin tosses and give  
## the incumbent the corresponding number of seats. Over our many simulations,   
## we will get a distribution of the possible outcomes.  
  
NUMBER\_OF\_SIMULATIONS = 20000  
SAMPLE\_SIZE = 100  
SimulationIncumbentSeatsWon = c()  
for(i in 1:NUMBER\_OF\_SIMULATIONS)  
 {  
 SimulatedIncumbentColleges = 0;  
 SimulationResults = c()  
 SimulationVoteShare = c()  
   
 for (row in 1:nrow(election\_forecast\_data))  
 {  
 state\_simulation = sample(c(0, 1), size = SAMPLE\_SIZE, replace = TRUE, prob = c(election\_forecast\_data$incumbent\_win\_probability[row], 1 - election\_forecast\_data$incumbent\_win\_probability[row]))  
 SimulationVoteShare = append(SimulationVoteShare, mean(state\_simulation))  
 if(mean(state\_simulation) <= 0.5){  
 SimulationResults = append(SimulationResults, election\_forecast\_data$Number\_Of\_Votes[row])  
 }  
 else{SimulationResults = append(SimulationResults, 0)}  
 }  
 SimulatedIncumbentColleges = sum(SimulationResults)  
 SimulationIncumbentSeatsWon = append(SimulationIncumbentSeatsWon, SimulatedIncumbentColleges)  
}  
  
## Visualize the results in 'SimulationResults' as you see fit

## Probability of Incumbent’s Win

