2024 Election Model Methodology

Alex Bass

2024-01-04

Introduction

Last US Presidential Election, I created a model that was a t-test of the 5 most recent quality polls by state. This time, I wanted to apply a more rigorous approach using a heirarchal bayesian linear regression.

Model Choice Justification

T-Test

While in the end, my last model performed decently well, a simple t-test can't control for variables such as survey audience, survey question type, or survey distribution type. So, if you are only performing t-tests, you have to make one of two concessions:

1. use only a fraction of the data (e.g. I will only use likely-voter polls)
2. use all data and try and implement corrections (e.g. all voter polls over estimated Trump by 4 percentage points, so subtract 4 before t-test). However, this sacrifices some statistical rigor since it may be an oversimplification of the effects. Things likely have changed in the last 4 years and the true effect of total population choice and voter choice will vary from poll to poll.

I tried to do the latter last time, but then the question becomes how granular of changes are you going to correct for? For example, If I want to correct for audience, question type, pollster effects, survey mode, etc. the list starts to add up fast. Can I correct for all of these? Also, what if there are omitted variables interacting with these variables and the dependent variable? For instance, maybe registered voters look different in Florida than they do in Maine etc. These questions present challenges for this methodology.

Hierarchal Model

This election, I am opting for a hierarchal regression model for a few reasons:

1. Unlike a T-Test, we can naturally control for important variables in our predictions.
2. We can use the natural structure of the data to our advantage. In the electoral college, elections are won by winning states. Each state is unqiue in makeup and attitudes toward candidates and parties. Allowing our model intercept to vary by states helps us account for their differences while utilizing a larger n-size and use information from control variables across states.
3. Predictions can be easily generated for this type of model.

Model Structure

Given the nature of the dependent variable, a percentage between 0-100, using a beta regression is ideal. In a previous version of this project, I used a typical linear regression with a gaussian distribution. However, using this method, it is possible to get both negative predictions and predictions over 100 which do not make sense in our problem context. For this reason, I decided to change to using beta regression where the predictions will be bounded between (0,1).

The model is structured in the following manner:

$$TrumpVotePercent \sim \mathcal{B}(a, b)$$

We assume $TrumpVotePercent$ is beta distributed with parameters a and b.

$$\eta = \beta_{state} + \beta_{controls} \cdot X_{controls}$$

$$\mu = g(\eta)$$

$$\hat{a} = \mu \cdot \phi$$

$$\hat{b} = \phi - a$$

Lets define some of these terms:

- μ represents our model mean
- ϕ represents our model precision
- β_{state} represents the intercept term for each state
- $\beta_{controls}$ and $X_{controls}$ represent the control coefficients and data sampled in our model.
- $g()$ represents the logit function which is our link function

$$\beta_{var} \sim \mathcal{N}(\mu_{var}, \sigma^2_{var})$$

In the model, I assume each independent variable in our model is normally distributed with a unique μ_{var} and σ_{var}.

Priors

I initially estimated the model with uninformative priors for independent variables with $\mu_{var} = 0$ and $\sigma_{var} = 1$ across all independent variables and each β_{state}. After an initial run, for future runs, I used the parameter estimates for μ_{var} and σ_{var} the day previous as priors in the model for the current day for each independent variable (see callout note below). I did this for two reasons:

1. The simulation percentage won tracked over time can be a bit unstable and jumpy. Using the previous days estimates as priors for the current day, hopefully, would help stabilize the and smooth the trendline over time.
2. The model will converge more easily and quickly with previously working priors. This will help me not exceed any limits for running servers with github actions and therefore have to pay.

Additionally, I set the prior for ϕ as normally distributed with mean as 100 and standard deviation as 1. I set the prior for this term as positive because this term inluences both a and b which must be positive.

> **Note**
>
> After using the exact priors from the previous model for a few weeks, I noticed that the priors were a bit too strong, so each day I use the previous mean and the previous standard deviation multiplied by 15, so the model changes more flexibly over time. I experimented with several different values as a multiplier, but settled on 15 tailored to this specific problem.

Data Processing and Feature Engineering

Filtering to Quality Polls

Not all polls/pollsters are created equal and I want to make sure quality polls are informing my model. Fortunately, FiveThirtyEight goes through the work or rating polls based on a number of factors and assigning a letter grade (A+ thru D). In my code, I order the grades and filter out C+ and below.

Dependent Variable

To simplify my model, I use a few cleaning steps. First of all, I am not interested in third party candidates as they do not usually win elections. So in my model, I just am predicting whether Trump will win each state (proportion $> 50\%$) and assuming if Trump does not win, Biden does. This of course does not match reality as it is possible for another candidate to win, but not possible in my model.

Surely, you may be thinking of cases where there may be a right wing (or left wing) independent candidate on the ballot who may take some of the vote, having perhaps a significant impact on the state winner. This is an important thing to consider. So, to incororate this into my model, I use third - party ballot questions and simply recale them to just Biden and Trump. After rescaling, I include an indicator variable for these observations. An example is illustrated below in Python:

```
#example code chunk
trump_perc = 40
biden_perc = 46
third_perc = 4

trump_perc_rescaled = trump_perc/(trump_perc + biden_perc)

print(f"Originally, Trump had {trump_perc}%, but after" \
    " rescaling (excluding third party candidates)\n"\
    f" he now has {round(trump_perc_rescaled,3)*100} %"\
    " with Biden having the rest.")
```

```
Originally, Trump had 40%, but after rescaling (excluding third party candidates)
 he now has 46.5 % with Biden having the rest.
```

With this method, third party questions are accounted for in the model as an indicator variable, but also if a third party candidate is taking votes from one sie or the other, this is reflectedin the rescaled number. In the example above, the third party candidate may be taking votes from Trump.

Control Variables

Creating and including the right control variables can have a large impact on analyses for better or for worse. Here is a table of control variables I include:

Var Description	Type	Justification
third party question	Indicator	Including a third party question fundamentally changes the obs. and should be accounted for
Republican pollster	Indicator	House effects are documented in research and affect our dependent variables
Nsize in survey	Integer	In theory, we would want to give more weight to polls with higher sample sizes and thus lower margins of error
Survey Mode	One Hot Encoded	Different modes reach different populations and these differences should be accounted for
Survey Population	One Hot Encoded	Political surveys are typically asked of one in three groups: likely voter, registered voter, all population. Each group is different
Date	Integer	Time is an important component in our model
Survey Quality	Indicator	I include this variable as a way to account for survey quality.

Note

I treat Survey Mode and Date slightly different than the others.

- For time, I include a sequential count of month since the first election polls for 2024 began.
- For survey mode, there are a lot of different categories and several categories are quite sparse. So, I collapsed the sparse categories into "other" and kept the big three: probability panel, online panel, and live phone.

Obtaining Predictions

Using pymc's `sample_posterior_predictive`, I use curated out of sample data for prediction. My settings are below:

Variable	Data	Justification
Date	Set to final month	I am interested in the latest month in the model when making predictions
Mode	Set to Probability Sample	Ad hoc analysis showed this was most accurate
Pollster	Set to Republican	Given the liberal bias of polls, I set this to republican for a correction
Sample Size	Set to 2000	2000 is not an uncommon sample size and is a large one
Question Type	Set to MultiCandidate	These questions will account for third party votes
Survey Grade	Set to >= 2 Stars	538 recently changed their pollster quality ranking. Instead of a letter grade from A-D its a number of stars from 0.5 to 3. I predict using greater than or equal to 2 stars representing more quality polls and hopefully better predictions.
Survey Population	Set to Likely Voters	Using population most likely to turn out

Given these settings, I predict each state and post the results daily.