Forecasting the 2024 U.S. Presidential Election: A Poll-of-Polls Approach for Predicting the Outcome[*]

Applying Aggregated Poll Data and Statistical Model to Reveal a Narrow Path to Victory in a Highly Competitive Race

Chendong Fei Xinze Wu Claire Ma

November 3, 2024

This paper develops a model to forecast the outcome of the 2024 U.S. presidential election by analyzing aggregated polling data, or "poll-of-polls," sourced from FiveThirtyEight. Using a generalized linear model, we assess national trends alongside key battleground state polls to predict each candidate's likelihood of victory. The findings indicate a closely contested race, with specific demographic and regional factors creating narrow pathways to winning the presidency. This analysis highlights the value of aggregated polling data in understanding electoral dynamics and demonstrates the importance of statistical modeling in making informed predictions about major political events.

1 Introduction

The outcome of the U.S. presidential election has far-reaching implications, shaping both domestic policies and international relations. As the 2024 election approaches, voters and analysts turn to polls to understand the state of the race between Vice President Kamala Harris, the Democratic candidate, and former President Donald Trump, the Republican candidate. However, individual polls are often limited by their methodologies, timing, and sample demographics, leading to variations in predictions. To overcome these limitations, aggregating multiple polls—a technique known as "poll-of-polls"—provides a more stable and reliable indicator of public opinion. This paper applies a poll-of-polls approach, informed by methodologies from individual polls(**blumenthal2014?**; **pasek2015?**), to predict the outcome of the 2024 U.S. presidential election, focusing on data aggregated by FiveThirtyEight(**fivethirtyeight2024?**).

[*]Code and data are available at: https://github.com/ke3w/Prediction_US_presidential_election.git

The primary objective of this analysis is to forecast which candidate is likely to win the 2024 election based on aggregated national and battleground state polling data. By constructing a generalized linear model, we aim to distill insights from the extensive polling data available, examining trends and key demographic indicators.

The primary estimand in this analysis is the probability of each candidate winning the 2024 U.S. presidential election based on aggregated polling data. This probability is derived from a weighted average of poll results across national and battleground states, with adjustments for factors such as recent polling trends, sample sizes, and state-specific electoral significance.

Our analysis reveals a highly competitive race, with key battleground states playing a pivotal role in determining the overall outcome. The model identifies specific regions and demographics that are likely to influence the election results, highlighting the polarized nature of the electorate.As of November 1, 2024, FiveThirtyEight's national polling average indicates a slight edge for Harris, who has 48.1% support compared to Trump's 46.7%. Despite this narrow national lead, the race in critical battleground states remains highly competitive. For instance, Pennsylvania is evenly split, with Harris holding marginal leads in states like Wisconsin and Michigan, while Trump shows slight advantages in Nevada, Georgia, and Arizona. These tight margins highlight the crucial role battleground states play in determining the election's outcome.

These findings underscore the importance of aggregated poll data in capturing the broader political landscape, offering insights that single polls may miss. By understanding the dynamics at play, this study contributes to a broader understanding of electoral processes and the predictive power of statistical models in forecasting complex political events.

This paper is organized as follows: Section 2 discusses the details of the dataset. **?@sec-Methodology** describes the methodology, including generalized linear model. **?@sec-Results** presents the results, highlighting trends in polling, and **?@sec-Discussion** considers the implications of these results for future research on polling and public opinion.

2 Data

2.1 Overview

We use the statistical programming language R (R Core Team 2023) to analyze polling data from FiveThirtyEight's U.S. Presidential election polls (**fivethirtyeight2024?**). The dataset contains information such as pollster, polling date, methodology, sample size, state, and candidate support percentages. It allows us to track voter sentiment across different regions and polling methods, providing a comprehensive view of the election landscape. Additionally, irrelevant or incomplete entries were removed to ensure clean, high-quality data, and we retained only key variables to streamline the analysis. This careful selection and cleaning process ensure that the dataset offers a precise and representative snapshot of the election landscape.

2.2 Measurement

The dataset measures public opinion on the 2024 U.S. presidential election by aggregating polling data to estimate voter support for each candidate at both national and state levels. These polling data entries are then aggregated, which applies a weighted adjustment to reflect the reliability, sample size, and recency of each poll. This weighting process addresses the natural variation in polling methodologies (e.g., online survey, phone), sample diversity, and timing, which influence the reliability of each poll as a measure of the broader population's preferences.For instance, if this poll was conducted a week before the election, it might be weighted more heavily than a poll from three months prior, as it better represents current voter sentiment. By weighting higher-quality and more recent polls more heavily, it creates a comprehensive measure that accounts for both regional and national voter sentiment, smoothing out biases from individual polls.

2.3 Outcome variables

In our analysis, the primary outcome variable is labeled `win`, which is a binary indicator representing the likelihood of a candidate "winning" in each poll based on their support percentage. Specifically, `win` is defined as follows: if a candidate's support percentage (`pct`) in a given poll exceeds 50%, then `win` is assigned a value of 1, indicating a projected win for that candidate in that poll. If the support percentage is 50% or below, `win` is assigned a value of 0, indicating that the candidate is not the likely winner in that poll. This binary outcome variable is particularly useful for logistic regression analysis, as it allows us to model the probability of a candidate achieving majority support in each poll. Using win provides a clear and interpretable framework to assess factors influencing a candidate's chances of gaining majority support, which aligns well with election forecasting goals. Additionally, this threshold reflects the electoral concept of a "win," as it represents the point at which a candidate has more than half of the vote share, an essential consideration in political analysis.

2.4 Predictor variables

The predictor variables in this analysis were chosen based on their potential influence on polling outcomes and candidate support. Each predictor reflects characteristics of the poll, the pollster, or the candidate's support environment. These variables aim to capture the factors that could impact the likelihood of a candidate reaching majority support ($win = 1$). Key predictor variables include:

- **sample_size**: Represents the number of respondents in each poll, with larger sample sizes generally leading to more reliable results.
- **pollster Rating**: Indicates the quality and historical accuracy of the polling organization, helping to account for differences in poll reliability.

- **state**: Captures the geographical region of the poll, reflecting regional differences in voter support that are crucial in U.S. elections.

These three variables provide a balanced view of poll quality, reliability, and regional influence, enhancing the model's ability to predict election outcomes accurately. This combination ensures that the model is interpretable and captures essential factors influencing voter sentiment. We apply Bayesian Information Criterion (BIC) method to select significant predictors, and a summary statistics for this method shown in (**Appendix?**)

3 Model

4 Description of the Model

The model used for this analysis is a generalized linear model (GLM) with a logistic regression link function, designed to predict the probability of a candidate winning an election based on polling data. The model takes into consideration several key predictors that may influence the likelihood of a candidate winning, including poll score, sample size, candidate party, and poll percentage. Specifically, the formula used in the model is:

$$\text{logit}(P(Y=1)) = \beta_0 + \beta_1 \cdot \text{pollscore} + \beta_2 \cdot \text{sample\_size} + \beta_3 \cdot \text{party\_binary} + \beta_4 \cdot \text{pct}$$

where: - Y = 1 represents the outcome that a candidate wins the poll. - pollscore refers to the quality score assigned to the poll, reflecting its reliability. - sample_size represents the number of respondents in the poll. - party_binary is a binary indicator of the candidate's party (1 for Democratic, 0 for others). - pct represents the percentage of support the candidate received in the poll.

```r
# Load necessary libraries
library(rstanarm)
```

```
Loading required package: Rcpp
```

```
This is rstanarm version 2.32.1
```

```
- See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!
```

```
- Default priors may change, so it's safest to specify priors, even if equivalent to the defa
- For execution on a local, multicore CPU with excess RAM we recommend calling

  options(mc.cores = parallel::detectCores())
```

```
library(tidyverse)

# Load the cleaned data
polls_data_cleaned <- read_csv("../data/02-analysis_data/cleaned_president_polls.csv")
```

```
Rows: 7990 Columns: 9

-- Column specification --------------------------------------------------------
Delimiter: ","
chr (4): state, party, candidate_name, end_date
dbl (5): pollscore, sample_size, pct, win, party_binary

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Fit the logistic regression model
win_model <- stan_glm(
  formula = win ~ pollscore + sample_size + party_binary + pct,
  data = polls_data_cleaned,
  family = binomial(link = "logit"),
  prior = normal(location = 0, scale = 2.5, autoscale = TRUE),
  prior_intercept = normal(location = 0, scale = 2.5, autoscale = TRUE),
  seed = 853
)
```

```

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000351 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.51 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 7.553 seconds (Warm-up)
Chain 1:                7.125 seconds (Sampling)
Chain 1:                14.678 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.00026 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.6 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 7.184 seconds (Warm-up)
Chain 2:                9.498 seconds (Sampling)
Chain 2:                16.682 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000287 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.87 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 7.636 seconds (Warm-up)
Chain 3:                7.364 seconds (Sampling)
Chain 3:                15 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000257 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.57 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 6.756 seconds (Warm-up)
Chain 4:                7.346 seconds (Sampling)
Chain 4:                14.102 seconds (Total)
Chain 4:
```

```r
# Print summary of the model
summary(win_model)
```

```
Model Info:
 function:     stan_glm
 family:       binomial [logit]
 formula:      win ~ pollscore + sample_size + party_binary + pct
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 7990
 predictors:   5

Estimates:
               mean   sd    10%   50%   90%
(Intercept)  -16.5    0.5 -17.2 -16.5 -15.8
pollscore      0.4    0.1   0.4   0.4   0.5
sample_size    0.0    0.0   0.0   0.0   0.0
party_binary  -0.4    0.1  -0.5  -0.4  -0.4
pct            0.3    0.0   0.3   0.3   0.4

Fit Diagnostics:
           mean   sd   10%   50%   90%
mean_PPD 0.2    0.0  0.2   0.2   0.2

The mean_ppd is the sample average posterior predictive distribution of the outcome variable

MCMC diagnostics
              mcse Rhat n_eff
(Intercept)   0.0  1.0  1941
pollscore     0.0  1.0  3067
sample_size   0.0  1.0  3073
party_binary  0.0  1.0  3069
pct           0.0  1.0  1916
mean_PPD      0.0  1.0  4105
log-posterior 0.0  1.0  1572
```

```
For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

4.1 Model Performance and Interpretation

The model's output includes estimates for each predictor, providing insights into their relative importance in predicting a candidate's probability of winning. The summary of the model reveals the following key points:

Poll Score (pollscore): The coefficient for pollscore indicates the extent to which the quality of a poll influences the predicted probability of a candidate winning. Higher-quality polls are expected to provide more reliable estimates, and therefore have a stronger effect on the predicted outcome.

Sample Size (sample_size): The coefficient for sample_size shows the influence of the number of respondents on the model. Polls with larger sample sizes are generally more reliable, contributing positively to the accuracy of predictions.

Party (party_binary): The party_binary variable highlights any potential advantage or disadvantage linked to party affiliation, with 1 representing Democratic candidates. This variable helps capture any systemic biases or influences based on party identification.

Poll Percentage (pct): The pct variable is a key predictor as it directly measures the level of support a candidate receives. A higher percentage is expected to lead to a higher probability of winning.

```r
# Calculate predicted win probabilities
polls_data_cleaned <- polls_data_cleaned %>%
  mutate(predicted_win_prob = predict(win_model, type = "response"))

# Display a few rows of data with predicted probabilities
polls_data_cleaned %>%
  dplyr::select(state, candidate_name, pct, predicted_win_prob) %>%
  head()
```

```
# A tibble: 6 x 4
  state         candidate_name      pct predicted_win_prob
  <chr>         <chr>             <dbl>              <dbl>
1 Massachusetts Kamala Harris        61       0.978
2 Massachusetts Donald Trump         31       0.00201
3 Massachusetts Robert F. Kennedy     2       0.0000000841
4 Massachusetts Jill Stein            1       0.0000000594
5 Massachusetts Cornel West           0       0.0000000420
6 Massachusetts Chase Oliver          0       0.0000000420
```

```r
# Save the model and predictions for later use
saveRDS(win_model, "../models/election_win_model.rds")
write_csv(polls_data_cleaned, "../data/04-model_results/polls_with_predictions.csv")
```

The model described above allows us to predict the likelihood of each candidate winning based on the given polling data, using a logistic regression approach. The model was implemented using rstanarm to provide a Bayesian estimation, which helps incorporate uncertainty into our predictions.

5 Discussion

5.1 First discussion point

5.2 Second discussion point

5.3 Third discussion point

5.4 Weaknesses and next steps

Appendix

A Additional data details

B Model details

B.1 Posterior predictive check

In **?@fig-ppcheckandposteriorvsprior-1** we implement a posterior predictive check. This shows...

In **?@fig-ppcheckandposteriorvsprior-2** we compare the posterior with the prior. This shows...

B.2 Diagnostics

?@fig-stanareyouokay-1 is a trace plot. It shows... This suggests...

?@fig-stanareyouokay-2 is a Rhat plot. It shows... This suggests...

References

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.