Final Project: Predicting the 2024 presidential election

Bowen Gu

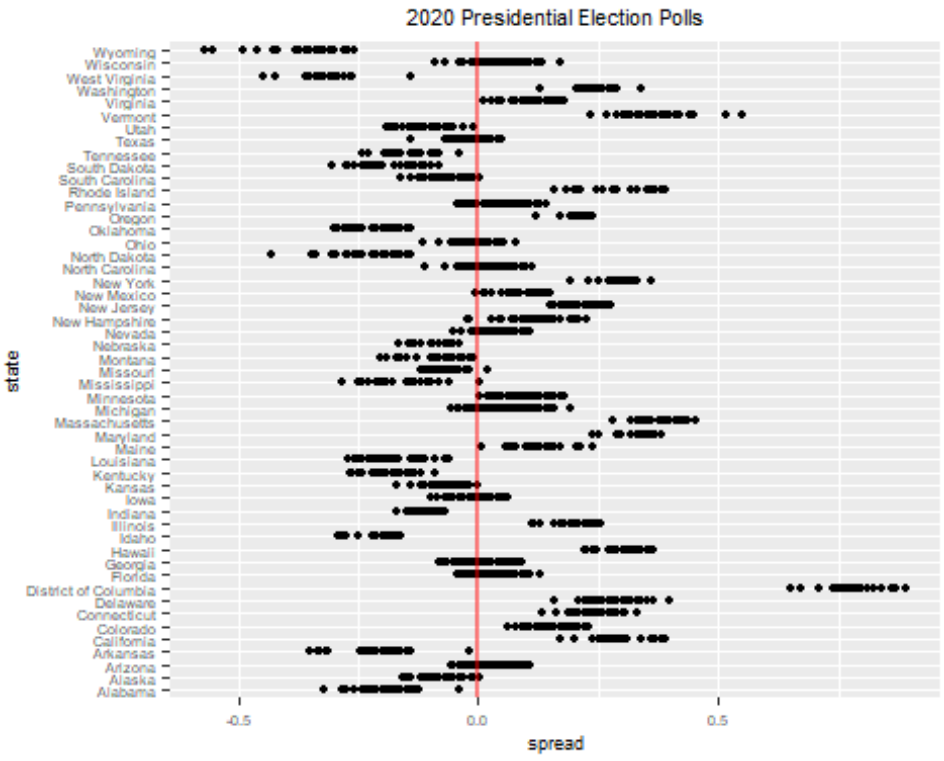
2022-11-13

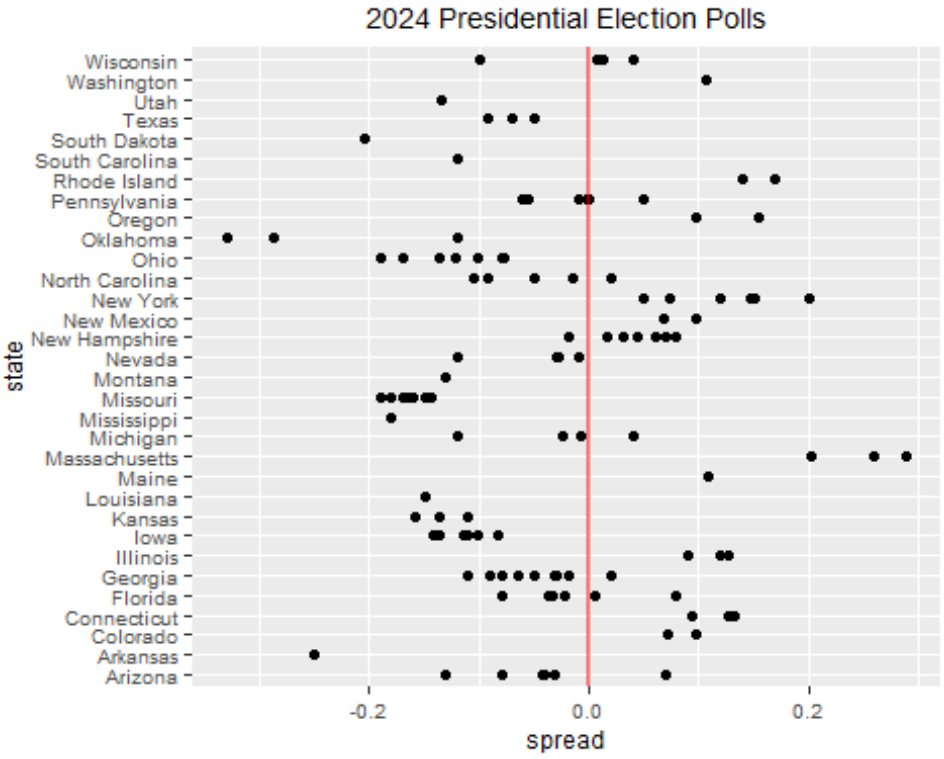
The project website is at https://gubowen2.github.io/BST-260-Final-Project/

Introduction

The US presidential election is coming up on November 5, 2024. Based on the current situation, it is possible that we see a “Biden VS Trump 2.0” in the next presidential election. If that is the case, how many electoral votes will Joe Biden win in this election?

To answer this question, I used two data sets from FiveThirtyEight named “president_polls” and “president_polls_historical”. The first data set contains polls about the incoming 2024 presidential election and the second data set contains polls about the previous 2020 presidential election. The major information recorded in the data sets are the states where the poll is for, and the supporting rate of both Biden and Trump, which can be visualized in the following two plots, where the “spread” in the x-axis is defined as the supporting rate of Biden minus the supporting rate of Trump, and each dot in the plot represents the result of one poll.





This project went through the process of predicting how many electoral votes will Joe Biden win in the 2024 presidential election in a detailed way using poll data from FiveThirtyEight. The methodology I used is the following. First, I used web scraping to get poll data from FiveThirtyEight. I also get the table of electoral votes

for each state as well as the table of the affiliation of each state from two other websites since these two pieces of information are essential when it comes to estimating the number of electoral votes that Joe Biden will win in the 2024 presidential election. Second, since the original data contains unnecessary and redundant information, I used data wrangling to remove unnecessary information in the poll data so that the data will have a clearer representation of the results. After that, since Bayesian statistical analysis is a useful and important approach used in prediction and it will be affected by different priors, I used Bayesian statistical analysis with three different priors together with the bias term to estimate the posterior of election of each state. Finally, since the posterior is a distribution and we need a method to simulate the election results of each state to calculate the number of electoral votes that Biden will win in the presidential election, I performed the Monte Carlo simulation to get the final estimation based on the posterior of each state.

Results

The data used was from FiveThirtyEight. The original data has the following major problems that makes data wrangling necessary.

1. Some polls in the data only aim at getting the general supporting rate of Biden and Trump and do not target a specific state, which is irrelevant of our question when the electoral votes of each state need to be estimated.

2. Although the majority of polls featuring the supporting rate of Biden and Trump, some polls show candidates besides Biden and Trump. Moreover, these candidates have very low supporting rate and should be unable to be elected as the next president. These candidates are thus unnecessary data that needs to be cleaned.

3. Some pollsters' poll grade is too low to be counted as credible data.

4. Some columns in the data are not in an ideal data type (e.g. the "start_date" and "end_date" column), which makes filtering according to time difficult.

5. The polls shows the supporting rate of Biden and Trump in two separate rows instead of one row. Also, not all polls show the supporting rate of both Biden and Trump. These makes the calculation of the supporting rate difference difficult.

6. Some states have too few polls to conduct a meaningful Bayesian statistical analysis.

Apart from data wrangling, web scraping is also necessary for this project due to the following reasons.

1. Information about the electoral votes for each state is needed and the information is on a table of a website.

2. Information about the party affiliation of each state is needed and the information is on a table of a website.

As an illustration of the data wrangling process, below are the 2024 presidential election poll data before and after the data wrangling process.

The 2024 presidential election before the data wrangling process

```
## # A tibble: 1,151 × 42
##    poll_id pollster_id pollster  spons…¹ spons…² displ…³ polls…⁴ pol
ls…⁵ fte_g…⁶
##      <dbl>       <dbl> <chr>       <dbl> <chr>   <chr>     <dbl> <ch
r>   <chr>
##  1   81688        1562 Redfield…      NA <NA>    Redfie…     562 Red
fie… B/C
##  2   81688        1562 Redfield…      NA <NA>    Redfie…     562 Red
fie… B/C
##  3   81688        1562 Redfield…      NA <NA>    Redfie…     562 Red
fie… B/C
##  4   81688        1562 Redfield…      NA <NA>    Redfie…     562 Red
fie… B/C
##  5   81688        1562 Redfield…      NA <NA>    Redfie…     562 Red
fie… B/C
##  6   81688        1562 Redfield…      NA <NA>    Redfie…     562 Red
fie… B/C
##  7   81688        1562 Redfield…      NA <NA>    Redfie…     562 Red
fie… B/C
##  8   81688        1562 Redfield…      NA <NA>    Redfie…     562 Red
fie… B/C
##  9   81688        1562 Redfield…      NA <NA>    Redfie…     562 Red
fie… B/C
## 10   81688        1562 Redfield…      NA <NA>    Redfie…     562 Red
fie… B/C
## # … with 1,141 more rows, 33 more variables: methodology <chr>, stat
e <chr>,
## #   start_date <chr>, end_date <chr>, sponsor_candidate_id <dbl>,
## #   sponsor_candidate <chr>, sponsor_candidate_party <chr>, question
_id <dbl>,
## #   sample_size <dbl>, population <chr>, subpopulation <lgl>,
## #   population_full <chr>, tracking <lgl>, created_at <chr>, notes <
chr>,
## #   url <chr>, source <dbl>, internal <lgl>, partisan <chr>, race_id
 <dbl>,
## #   cycle <dbl>, office_type <chr>, seat_number <dbl>, seat_name <lg
l>, …
```

The 2024 presidential election after the data wrangling process

```
## # A tibble: 115 × 40
##    poll_id pollster_id pollster  spons…¹ spons…² displ…³ polls…⁴ pol
ls…⁵ fte_g…⁶
##      <dbl>       <dbl> <chr>       <dbl> <chr>   <chr>     <dbl> <ch
r>   <chr>
##  1   81636         460 SurveyUSA     761 WNYT-TV Survey…     325 Sur
vey… A
##  2   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  3   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  4   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  5   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  6   81627        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  7   81627        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  8   81628        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  9   81628        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
## 10   81544        1102 Emerson …      NA <NA>    Emerso…      88 Eme
rso… A-
## # … with 105 more rows, 31 more variables: methodology <chr>, state
<chr>,
## #   start_date <date>, end_date <date>, sponsor_candidate_id <dbl>,
## #   sponsor_candidate <chr>, sponsor_candidate_party <chr>, question
_id <dbl>,
## #   sample_size <dbl>, population <chr>, subpopulation <lgl>,
## #   population_full <chr>, tracking <lgl>, created_at <chr>, notes <
chr>,
## #   url <chr>, source <dbl>, internal <lgl>, partisan <chr>, race_id
 <dbl>,
## #   cycle <dbl>, office_type <chr>, seat_number <dbl>, seat_name <lg
l>, …
```

The data wrangling process mainly did the following.

1. It clears all polls that do not target a specific state.
2. It clears all candidates that are not Biden or Trump.
3. It clears all pollsters with fte grade below C.
4. It uses the “lubridate” package to convert the date columns of the original data into the ideal data type.

5. It merges the rows of Biden and Trump of each polls into one row and creates a new column that shows the spread of the supporting rate between Biden and Trump.

6. It clears all rows that does not have both the supporting rate of Biden and Trump.

Besides, below were the electoral votes for each state and the party affiliation of each state table that I got after applying web scraping

The electoral votes for each state table

```
## # A tibble: 51 × 2
##    state          electoral_votes
##    <chr>                    <dbl>
##  1 California                  54
##  2 Texas                       40
##  3 Florida                     30
##  4 New York                    28
##  5 Illinois                    19
##  6 Pennsylvania                19
##  7 Ohio                        17
##  8 Georgia                     16
##  9 North Carolina              16
## 10 Michigan                    15
## # … with 41 more rows
```

In this table, each row shows a state and the electoral votes it has in the 2024 presidential election.

The party affiliation for each state table

```
## # A tibble: 51 × 5
##    state                Rep   None  Dem   affiliation
##    <chr>                <chr> <chr> <chr> <chr>
##  1 Alabama              52%   13%   35%   R
##  2 Alaska               39%   29%   32%   R
##  3 Arizona              40%   21%   39%   R
##  4 Arkansas             46%   16%   38%   R
##  5 California           30%   21%   49%   B
##  6 Colorado             41%   17%   42%   B
##  7 Connecticut          32%   18%   50%   B
##  8 Delaware             29%   17%   55%   B
##  9 District of Columbia 11%   15%   73%   B
## 10 Florida              37%   19%   44%   B
## # … with 41 more rows
```

In this table, each row shows a state and the party affiliation that the adults in the states consider themselves as, where the data under column “Rep” indicates the percentage that the adults in the states who consider themselves as Republicans, the data under column “None” indicates the percentage that the adults in the states who

has no party affiliation, and the data under column "Dem" indicates the percentage that the adults in the states who consider themselves as Democrats. Here, I consider the state as a "Blue state" ("B" in the "affiliation" column) if it has more adults that think they are Democrat/lean Dem. and a "Red state" ("R" in the "affiliation" column) if it has more adults that think they are Republican/lean Rep. Otherwise, I label the state as a "White state" ("W" in the "affiliation" column) since it has no affiliation.

For the Bayesian statistical analysis part, I used the following priors.

1. Normal distribution with $\mu = 0$ and $\sigma = 0.04$ for all states.
2. Normal distribution with μ equals the mean spread of the final month before the 2020 presidential election for each state and σ equals the standard deviation of spread from 2018 to 2020 for each state.
3. Normal distribution with μ equals the mean of the current mean spread and the mean spread of the final month before the 2020 presidential election for each state and σ equals the standard deviation of spread from 2018 to 2020 for each state.

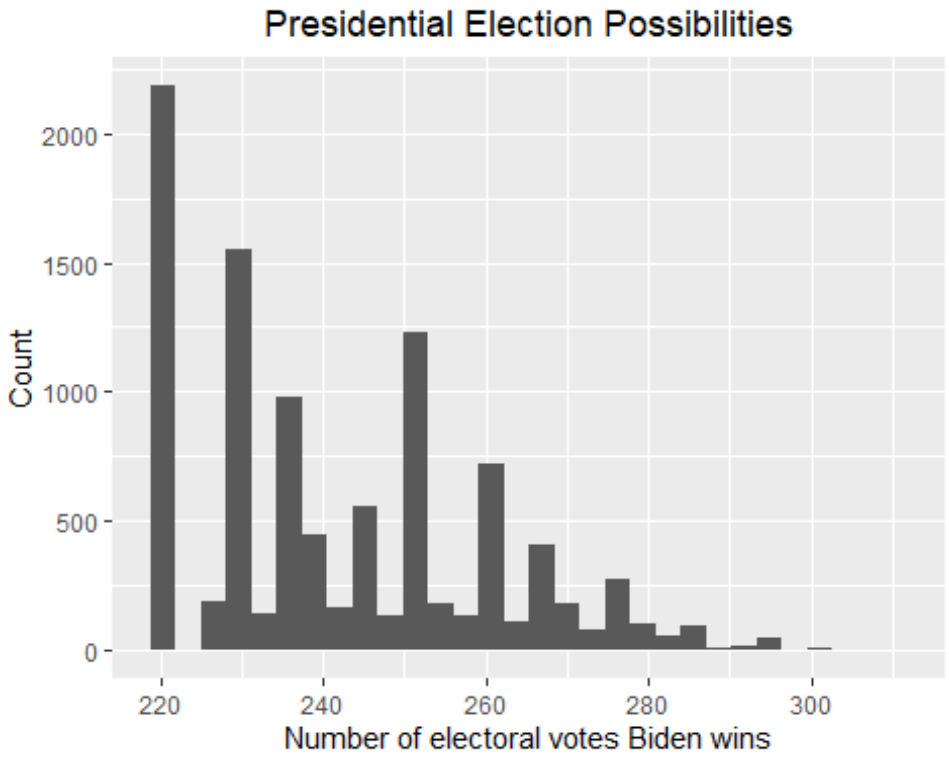
I chose the normal distribution since the election result is usually normally distributed. As for the parameters in the normal distribution, the first prior is a basic one based on no knowledge about the historical election. The second prior utilized the historical election information. I chose this prior mean since we want to predict the results that are close to the election day. I chose this standard deviation since the earliest poll data from FiveThirtyEight is since 2018 and I want the standard deviation to be relatively large to account for uncertainties since I am predicting the 2024 presidential election using the data in 2022 and a lot can happen during the next two years. The third prior utilized both the historical and the current presidential election information. I chose this new prior mean since for the last month of the 2020 election, people have seen Trump's performance for the four years but they have not seen Biden's performance. As a result, the polls will go against Trump and favor Biden. However, in the 2024 presidential election, people will see how Biden performed in the four years and the polls are more likely to go against Biden. So I assume that in 2024, people have seen both Biden and Trump's performance during their presidency and their favor towards both candidates is mitigated, namely they do not favor strongly for either Biden or Trump. Based on this assumption, we can have the prior mean to be the mean of the current mean spread and the mean spread of the final month before the 2020 presidential election. The standard deviation for the new prior does not change since I still want the standard deviation to be relatively large to account for uncertainties.

For each prior mentioned above, the corresponding posterior was obtained by either including a bias term or not. The bias added follows the normal distribution with $\mu = 0$ and $\sigma = 0.03$ for all states. I chose the normal distribution since the bias is usually normally distributed. The mean of the bias is set to be 0 since I do not think the bias will go towards either party. The standard deviation of the bias is set to be 0.03 according to the textbook of this course. The purpose of adding the bias

term is to account for the potential uncertainty of the election, making the results more similar to a normal distribution.

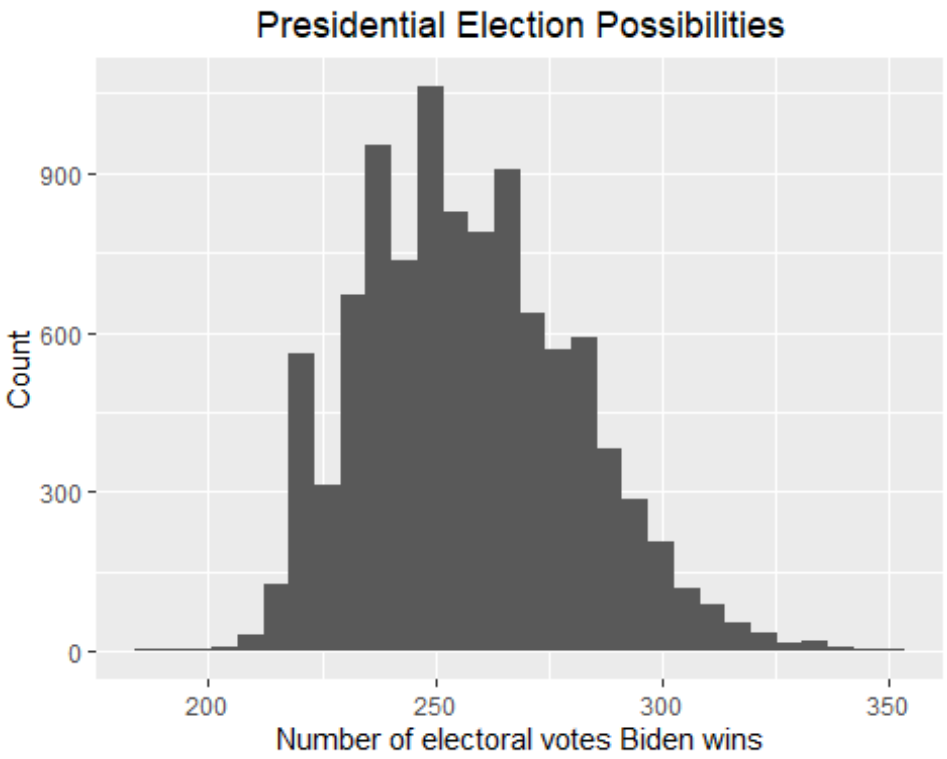
After the Bayesian statistical analysis, I used the Monte Carlo simulation to simulate the election results of the states based on each posterior mean and standard deviation corresponds to the prior I used. Since not all states are included in the Bayesian statistical analysis due to the lack of poll data, I used the state affiliation data for the states that are not included in the Bayesian statistical analysis and assumed that the blue states will still be in favor of Biden and the red states will still be in favor of Trump. Then I calculated the number of electoral votes that Biden will win by combining the simulated election results and the number of electoral votes in each state and plotted a histogram to show the results. The histogram showing the number of electoral votes that Biden will win for the 2024 presidential election is the following, accompanied by Biden's winning rate and the 80% credible interval under these situations.

Prior 1, no bias



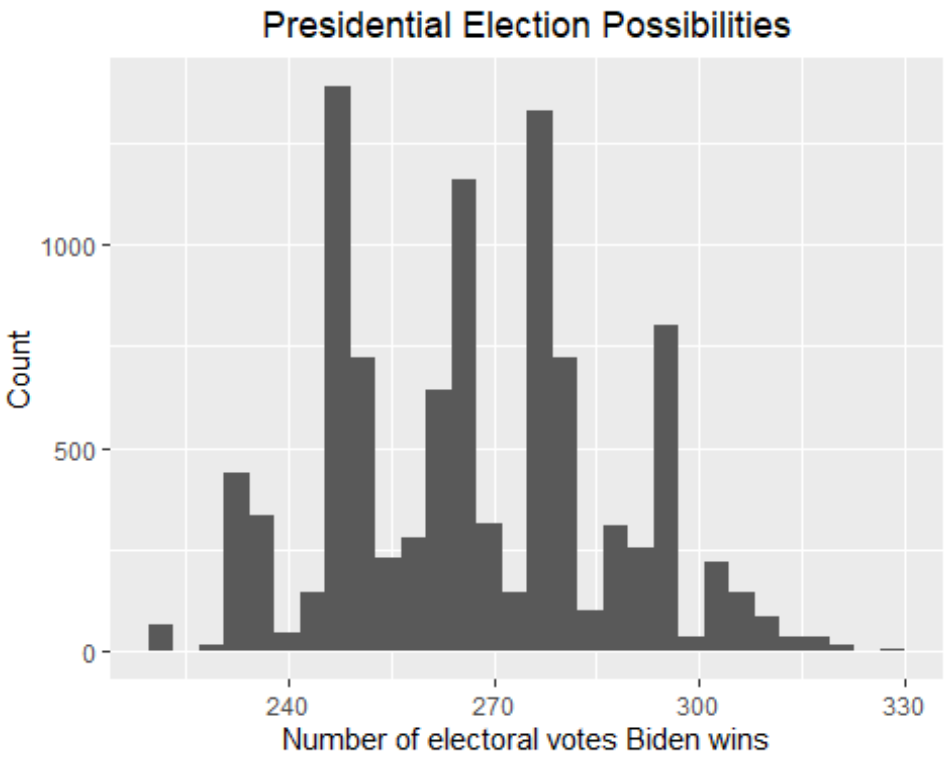

```
## [1] "Biden has about  8.66 % probability to win the 2024 presidentia
l election. The corresponding 80% credible interval is [ 221 ,  261 ]"
```

Prior 1, with bias



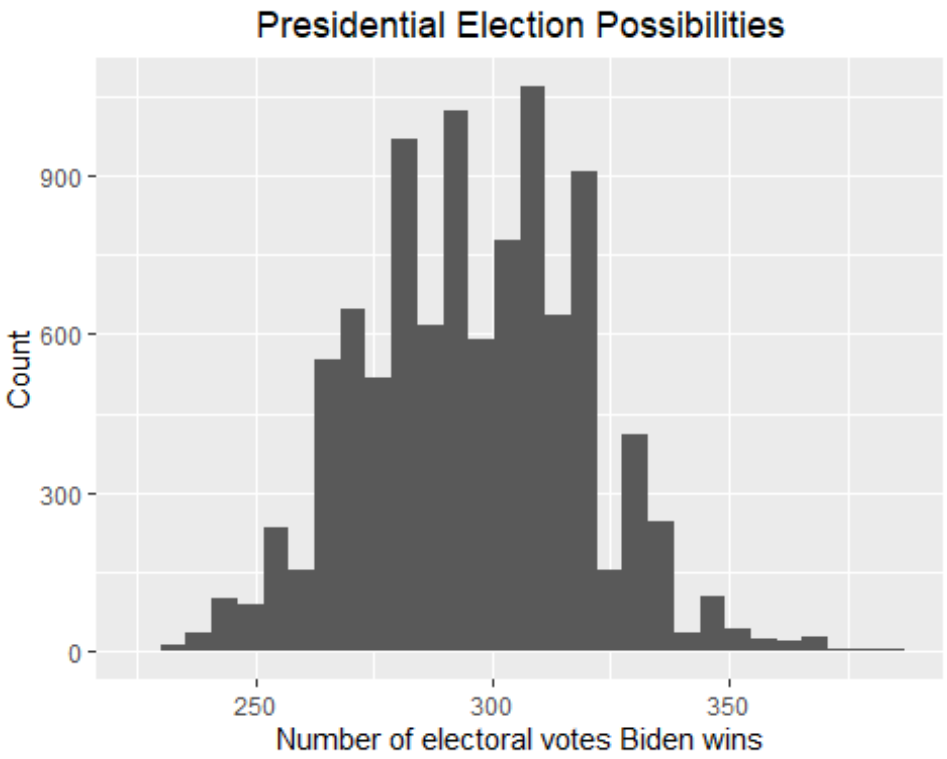

```
## [1] "Biden has about  29.76 % probability to win the 2024 presidenti
al election. The corresponding 80% credible interval is [ 221 ,  279 ]"
```

Prior 2, no bias



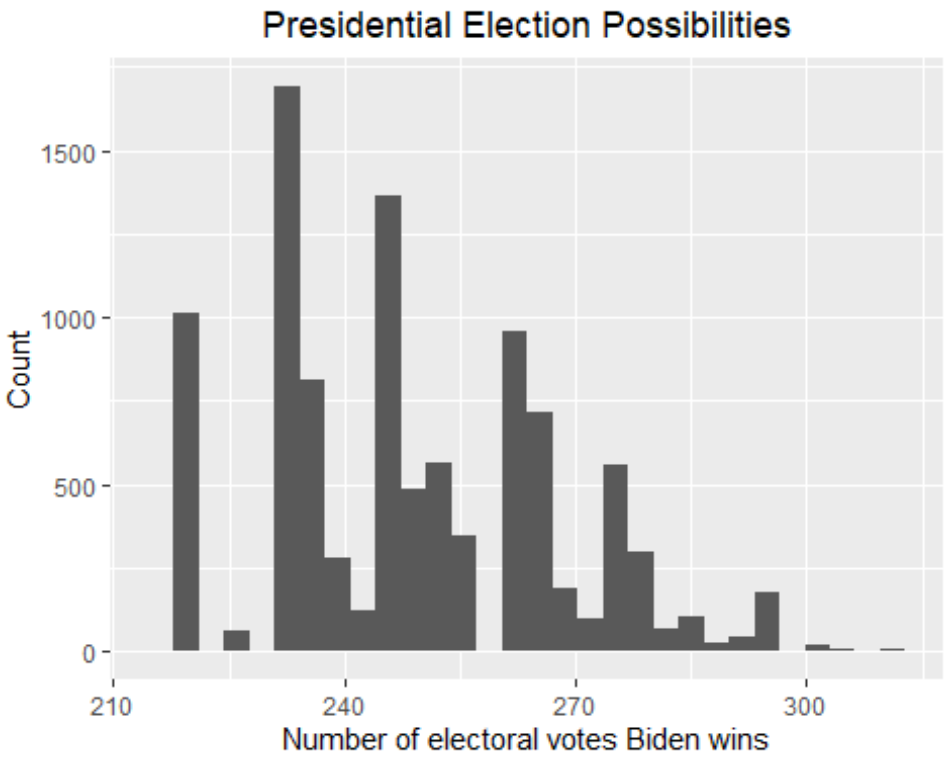

```
## [1] "Biden has about  44.93 % probability to win the 2024 presidenti
al election. The corresponding 80% credible interval is [ 246 ,  295 ]"
```

Prior 2, with bias



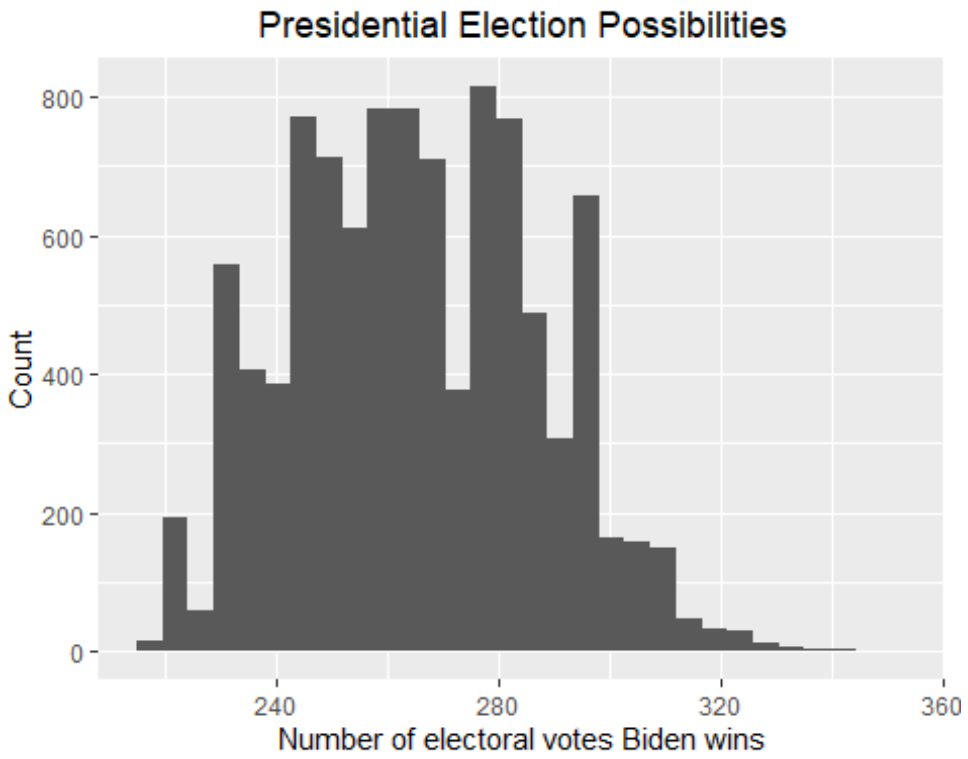

```
## [1] "Biden has about  87.49 % probability to win the 2024 presidenti
al election. The corresponding 80% credible interval is [ 261 ,  317 ]"
```

Prior 3, no bias




```
## [1] "Biden has about  15.24 % probability to win the 2024 presidenti
al election. The corresponding 80% credible interval is [ 221 ,  265 ]"
```

Prior 3, with bias




```
## [1] "Biden has about  41.71 % probability to win the 2024 presidenti
al election. The corresponding 80% credible interval is [ 231 ,  287 ]"
```

Conclusion

In this project, I predicted the number of electoral votes that Joe Biden will win in the 2024 presidential election if he and Donald Trump are the candidates of the Democrats and Republicans, respectively. I used data wrangling strategies to process the poll data from FiveThirtyEight and web scraping strategies to extract tables from websites to get information about the electoral votes for each state and the party affiliation of each state. Then I conducted the Bayesian statistical analysis. I used three different priors and got six corresponding posteriors by controlling if a bias term is added. Finally, I performed the Monte Carlo simulation to get the final estimation based on the posterior of each state.

According to the results above, we can see that the election results get more uncertain if the bias term is added. It also increases Biden's winning rate for all three priors. As for the comparison between different priors, we can see that if we assume no knowledge about the previous elections and the election trends and only use the data for the current election cycle, Biden is likely to lose the next presidential election and Trump will be the next president. Interestingly, if we rely heavily on the last presidential election to set our prior, we notice that the results are overturned. It is Biden who are likely to win the next presidential election. This is anticipated since for the last month of the election, people have seen Trump's

performance for the four years but they have not seen Biden's performance. As a result, the polls will go against Trump and favor Biden. However, in the 2024 presidential election, people will see how Biden performed in the four years and the polls are more likely to go against Biden. This leads to the third prior. Since the third prior assumes that in 2024, people have seen both Biden and Trump's performance during their presidency and their favor towards both candidates is mitigated, namely they do not favor strongly for either Biden or Trump, we get a much neutralized result. I believe that this result is more reasonable since it is sometimes the case that a president lose the midterm election but win in the presidential election two years later. Since I think the polls in 2020 favor Biden and the polls in 2022 favor Trump, by mitigating the polls results, we should approach the results better.

Therefore, I would like to conclude that according to my analysis, despite the current low supporting rate of Biden compared to Trump, the result of the 2024 presidential election is still a close call, though the probability of Biden wins the election is still low than that of Trump.

The future work of this project can be the following.

1. Update the 2024 presidential election polls from FiveThirtyEight as time get closer to the election day to get a better estimation of the results.

2. Gather more historical presidential election poll data to get a better idea about the election trends and bias distribution, which leads to a more reasonable and robust prior and bias term in the Bayesian statistical analysis.

3. Combine the election prediction with analysis about the situation of the country and its connection to the world at that time (e.g. the economy situation, the political relationship with other countries), which leads to more sophisticated and realistic prior and bias term in the Bayesian statistical analysis.

References

1. Rafael A. Irizarry, Introduction to Data Science - Data Analysis and Prediction Algorithms with R http://rafalab.dfci.harvard.edu/dsbook/

2. BST 260, Homework 2: Predicting the midterm elections

3. FiveThirtyEight: 2024 presidential election polls data https://projects.fivethirtyeight.com/polls/data/president_polls.csv

4. FiveThirtyEight: 2020 presidential election polls data https://projects.fivethirtyeight.com/polls/data/president_polls_historical.csv

5. 1keydata: List of State Electoral Votes For 2024
   https://state.1keydata.com/state-electoral-votes.php

6. Pew Research Center: Party affiliation by state
   https://www.pewresearch.org/religion/religious-landscape-study/compare/party-affiliation/by/state/

Appendix

Load the FiveThirtyEight president polls from https://projects.fivethirtyeight.com/polls/data/president_polls.csv into an object called `president_polls`. Remove polls that are not part of the 2024 president elections. Keep only the polls that are "general" elections and target a specific state. Also, only keep polls that have Biden or Trump as the candidate.

```
president_polls <- read_csv("https://projects.fivethirtyeight.com/polls/data/president_polls.csv")
president_polls <- president_polls |> filter(cycle == 2024 & stage == "general" & !is.na(state) & candidate_name %in% c("Joe Biden", "Donald Trump"))
president_polls
```

```
## # A tibble: 304 × 42
##    poll_id pollster_id pollster  spons…¹ spons…² displ…³ polls…⁴ polls…⁵ fte_g…⁶
##      <dbl>       <dbl> <chr>       <dbl> <chr>   <chr>     <dbl> <chr>   <chr>
##  1   81693        1672 Victory …      NA <NA>    Victor…     673 Victor… <NA>
##  2   81693        1672 Victory …      NA <NA>    Victor…     673 Victor… <NA>
##  3   81693        1672 Victory …      NA <NA>    Victor…     673 Victor… <NA>
##  4   81636         460 SurveyUSA     761 WNYT-TV Survey…     325 Survey… A
##  5   81636         460 SurveyUSA     761 WNYT-TV Survey…     325 Survey… A
##  6   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Targoz… B/C
##  7   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Targoz… B/C
##  8   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Targoz… B/C
##  9   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Targoz… B/C
## 10   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Targoz… B/C
## # … with 294 more rows, 33 more variables: methodology <chr>, state <chr>,
```

```
## #   start_date <chr>, end_date <chr>, sponsor_candidate_id <dbl>,
## #   sponsor_candidate <chr>, sponsor_candidate_party <chr>, question
_id <dbl>,
## #   sample_size <dbl>, population <chr>, subpopulation <lgl>,
## #   population_full <chr>, tracking <lgl>, created_at <chr>, notes <
chr>,
## #   url <chr>, source <dbl>, internal <lgl>, partisan <chr>, race_id
 <dbl>,
## #   cycle <dbl>, office_type <chr>, seat_number <dbl>, seat_name <lg
l>, …
```

Look at a table of the `fte_grade` given to each of the polls. Filter the `president_polls` data set to keep only polls with a grade in the top 10 grades, up to C.

```
president_polls <- president_polls |> filter(fte_grade <= "C" | fte_gra
de == "C+")
president_polls
```

```
## # A tibble: 256 × 42
##    poll_id pollster_id pollster  spons…¹ spons…² displ…³ polls…⁴ pol
ls…⁵ fte_g…⁶
##      <dbl>       <dbl> <chr>       <dbl> <chr>   <chr>     <dbl> <ch
r>   <chr>
##  1   81636         460 SurveyUSA     761 WNYT-TV Survey…     325 Sur
vey… A
##  2   81636         460 SurveyUSA     761 WNYT-TV Survey…     325 Sur
vey… A
##  3   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  4   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  5   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  6   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  7   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  8   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  9   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
## 10   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
## # … with 246 more rows, 33 more variables: methodology <chr>, state
<chr>,
## #   start_date <chr>, end_date <chr>, sponsor_candidate_id <dbl>,
## #   sponsor_candidate <chr>, sponsor_candidate_party <chr>, question
_id <dbl>,
## #   sample_size <dbl>, population <chr>, subpopulation <lgl>,
```

```
## #   population_full <chr>, tracking <lgl>, created_at <chr>, notes <
chr>,
## #   url <chr>, source <dbl>, internal <lgl>, partisan <chr>, race_id
 <dbl>,
## #   cycle <dbl>, office_type <chr>, seat_number <dbl>, seat_name <lg
l>, …
```

Make a table showing how many states have available poll data and what race_id is it.

```
temp <- president_polls |> distinct(across(c(race_id, state, candidate_
id)))
temp <- temp |> group_by(race_id, state) |> summarize(candidates = leng
th(candidate_id))
temp
```

```
## # A tibble: 32 × 3
## # Groups:   race_id [32]
##    race_id state       candidates
##      <dbl> <chr>            <int>
##  1    8755 Arkansas             2
##  2    8759 Arizona              2
##  3    8765 Colorado             2
##  4    8768 Connecticut          2
##  5    8778 Florida              2
##  6    8781 Georgia              2
##  7    8788 Iowa                 2
##  8    8794 Illinois             2
##  9    8800 Kansas               2
## 10    8806 Louisiana            2
## # … with 22 more rows
```

To make the following processing easier, change the name of the candidates to "Biden" and "Trump" respectively.

```
president_polls["candidate_name"][president_polls["candidate_name"] ==
"Joe Biden"] = "Biden"
president_polls["candidate_name"][president_polls["candidate_name"] ==
"Donald Trump"] = "Trump"
```

Identify the columns we need to remove for `pivot_wider` to work and remove them. After removing these columns, use `pivot_wider` to create columns called `Biden` and `Trump` for the Democrat and Republican poll percentage, respectively.

```
president_polls <- president_polls |> select(c(-answer, -candidate_id,
-party)) |> pivot_wider(names_from = "candidate_name", values_from = "p
ct")
president_polls
```

```
## # A tibble: 141 × 39
##    poll_id pollster_id pollster  spons…¹ spons…² displ…³ polls…⁴ pol
```

```
ls…⁵ fte_g…⁶
##      <dbl>       <dbl> <chr>       <dbl> <chr>   <chr>     <dbl> <ch
r>   <chr>
##  1   81636         460 SurveyUSA     761 WNYT-TV Survey…     325 Sur
vey… A
##  2   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  3   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  4   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  5   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  6   81627        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  7   81627        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  8   81628        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  9   81628        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
## 10   81544        1102 Emerson …      NA <NA>    Emerso…      88 Eme
rso… A-
## # … with 131 more rows, 30 more variables: methodology <chr>, state
<chr>,
## #   start_date <chr>, end_date <chr>, sponsor_candidate_id <dbl>,
## #   sponsor_candidate <chr>, sponsor_candidate_party <chr>, question
_id <dbl>,
## #   sample_size <dbl>, population <chr>, subpopulation <lgl>,
## #   population_full <chr>, tracking <lgl>, created_at <chr>, notes <
chr>,
## #   url <chr>, source <dbl>, internal <lgl>, partisan <chr>, race_id
 <dbl>,
## #   cycle <dbl>, office_type <chr>, seat_number <dbl>, seat_name <lg
l>, …
```

Notice that there are some rows that either miss record for the supporting rate for Biden or the supporting rate for Trump. Let's show these rows.

```
temp <- president_polls |> filter(is.na(Biden) | is.na(Trump)) |> selec
t(pollster_id, question_id, Biden, Trump)
temp
```

```
## # A tibble: 26 × 4
##    pollster_id question_id Biden Trump
##          <dbl>       <dbl> <dbl> <dbl>
##  1         383      163416  NA    42
##  2        1102      163285  37.6  NA
##  3         458      161811  43.6  NA
##  4         458      161813  NA    45.6
```

```
##  5        1102      161438  38.4  NA
##  6        1302      161201  43    NA
##  7        1302      161203  42    NA
##  8        1302      161204  47    NA
##  9        1302      161205  39    NA
## 10        1302      161206  33    NA
## # … with 16 more rows
```

According to the above table, we can see that it is the different question_id that prevent the rows from merging. Since there are not too many rows affected by this and there are only the supporting rate for Biden for most rows, I decide to remove all rows with missing supporting rate for either candidate.

```
president_polls <- president_polls |> filter(!is.na(Biden) & !is.na(Tru
mp))
```

Now define a new column `spread` as `(Biden-Trump)/100`.

```
president_polls <- president_polls |> mutate(spread = (Biden - Trump) /
 100)
president_polls
```

```
## # A tibble: 115 × 40
##    poll_id pollster_id pollster  spons…¹ spons…² displ…³ polls…⁴ pol
ls…⁵ fte_g…⁶
##      <dbl>       <dbl> <chr>       <dbl> <chr>   <chr>     <dbl> <ch
r>   <chr>
##  1   81636         460 SurveyUSA     761 WNYT-TV Survey…     325 Sur
vey… A
##  2   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  3   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  4   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  5   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  6   81627        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  7   81627        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  8   81628        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  9   81628        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
## 10   81544        1102 Emerson …      NA <NA>    Emerso…      88 Eme
rso… A-
## # … with 105 more rows, 31 more variables: methodology <chr>, state
<chr>,
## #   start_date <chr>, end_date <chr>, sponsor_candidate_id <dbl>,
## #   sponsor_candidate <chr>, sponsor_candidate_party <chr>, question
```

```
_id <dbl>,
## #   sample_size <dbl>, population <chr>, subpopulation <lgl>,
## #   population_full <chr>, tracking <lgl>, created_at <chr>, notes <
chr>,
## #   url <chr>, source <dbl>, internal <lgl>, partisan <chr>, race_id
 <dbl>,
## #   cycle <dbl>, office_type <chr>, seat_number <dbl>, seat_name <lg
l>, …
```

Use the `mdy` function in the `lubridate` package to mutate the start and end date columns in place to be Dates instead of characters.

```
president_polls["start_date"] <- mdy(president_polls$start_date)
president_polls["end_date"] <- mdy(president_polls$end_date)
president_polls
```

```
## # A tibble: 115 × 40
##    poll_id pollster_id pollster  spons…¹ spons…² displ…³ polls…⁴ pol
ls…⁵ fte_g…⁶
##      <dbl>       <dbl> <chr>       <dbl> <chr>   <chr>     <dbl> <ch
r>   <chr>
##  1   81636         460 SurveyUSA     761 WNYT-TV Survey…     325 Sur
vey… A
##  2   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  3   81625        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  4   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  5   81626        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  6   81627        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  7   81627        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  8   81628        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
##  9   81628        1477 Targoz M…    1477 PollSm… Targoz…     454 Tar
goz… B/C
## 10   81544        1102 Emerson …      NA <NA>    Emerso…      88 Eme
rso… A-
## # … with 105 more rows, 31 more variables: methodology <chr>, state
<chr>,
## #   start_date <date>, end_date <date>, sponsor_candidate_id <dbl>,
## #   sponsor_candidate <chr>, sponsor_candidate_party <chr>, question
_id <dbl>,
## #   sample_size <dbl>, population <chr>, subpopulation <lgl>,
## #   population_full <chr>, tracking <lgl>, created_at <chr>, notes <
chr>,
## #   url <chr>, source <dbl>, internal <lgl>, partisan <chr>, race_id
```

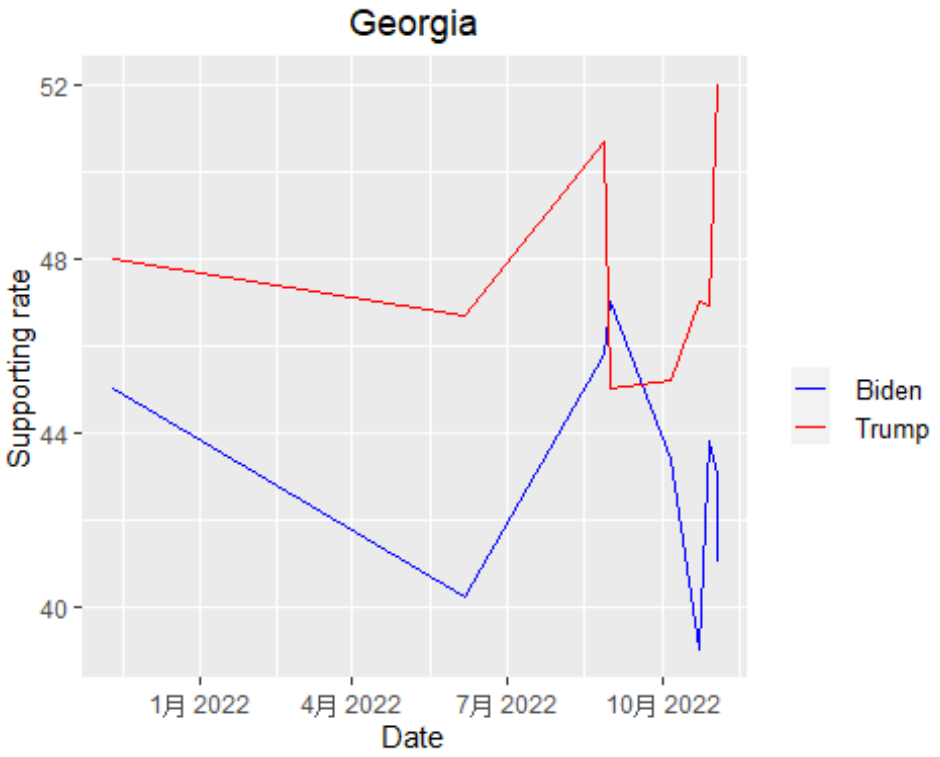
```
 <dbl>,
## #   cycle <dbl>, office_type <chr>, seat_number <dbl>, seat_name <lg
l>, …
```

Now let's see how many polls we have for each state.

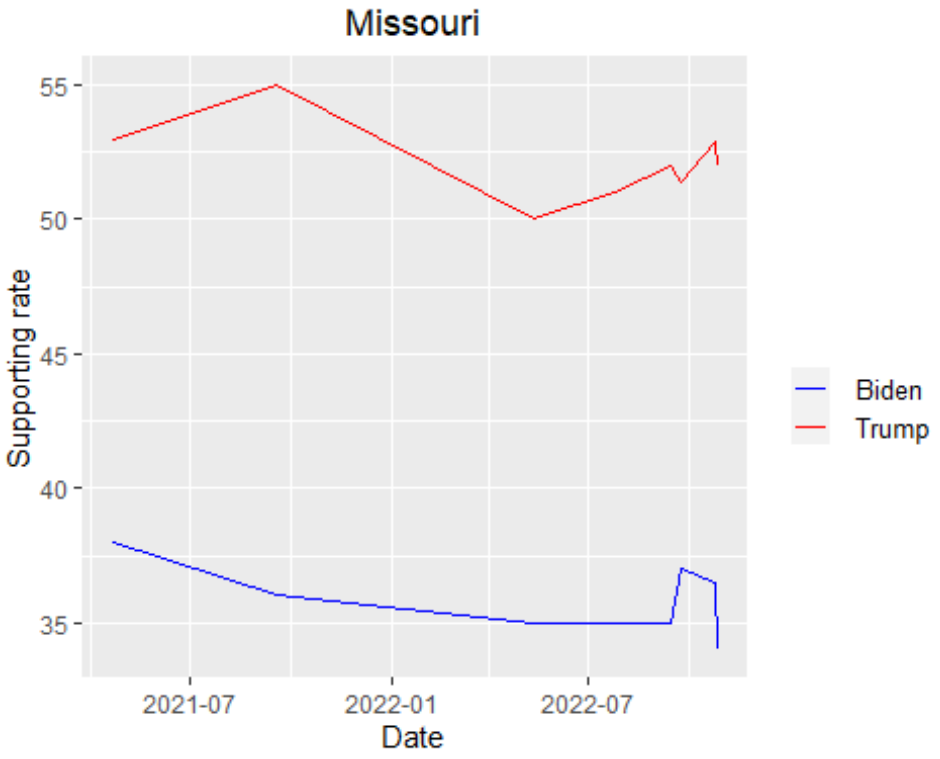
```
president_polls |> group_by(state) |> summarize(count = length(state))
|> arrange(-count)

## # A tibble: 32 × 2
##    state          count
##    <chr>          <int>
##  1 Georgia            9
##  2 Missouri           8
##  3 Arizona            7
##  4 New Hampshire      7
##  5 Ohio               7
##  6 Pennsylvania       7
##  7 Florida            6
##  8 Iowa               6
##  9 New York           6
## 10 North Carolina     5
## # … with 22 more rows
```

For states with available polls > 6, let's see how Biden and Trump's supporting rates change with respect to time.

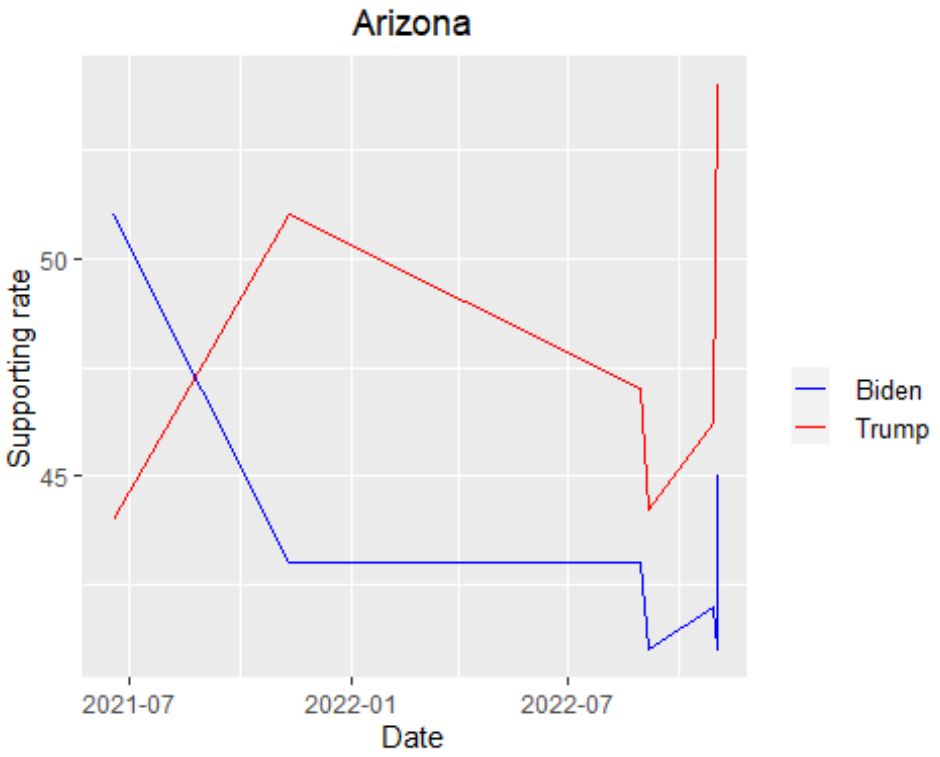
```
get_plot <- function(target_state){
  president_polls |> filter(state == target_state) |> ggplot(aes(x = st
art_date)) + geom_line(aes(y = Biden, colour = "Biden")) + geom_line(ae
s(y = Trump, colour = "Trump")) + xlab("Date") + ylab("Supporting rate")
 + ggtitle(target_state) + scale_colour_manual("",
                      breaks = c("Biden", "Trump"),
                      values = c("blue", "red")) + theme(legend.key.siz
e = unit(0.5, 'cm'), #change legend key size
        legend.key.height = unit(0.5, 'cm'), #change legend key height
        legend.key.width = unit(0.5, 'cm'), #change legend key width
        legend.title = element_text(size=20), #change legend title font
 size
        legend.text = element_text(size=10), #change legend text font s
ize
        plot.title = element_text(hjust = 0.5))
}

get_plot("Georgia")
```



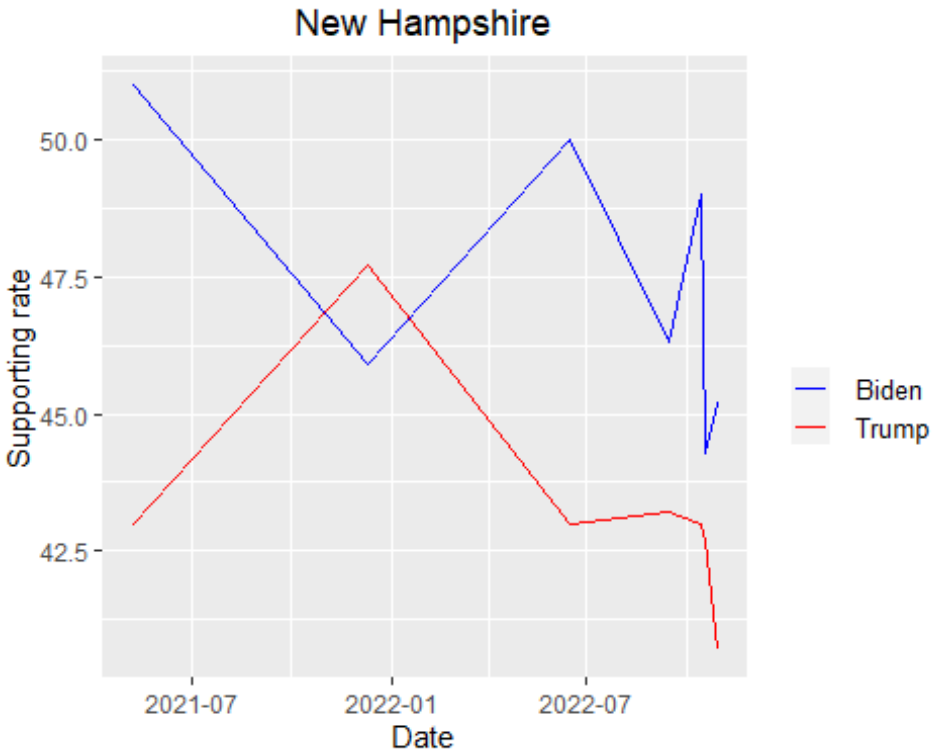

```
get_plot("Missouri")
```



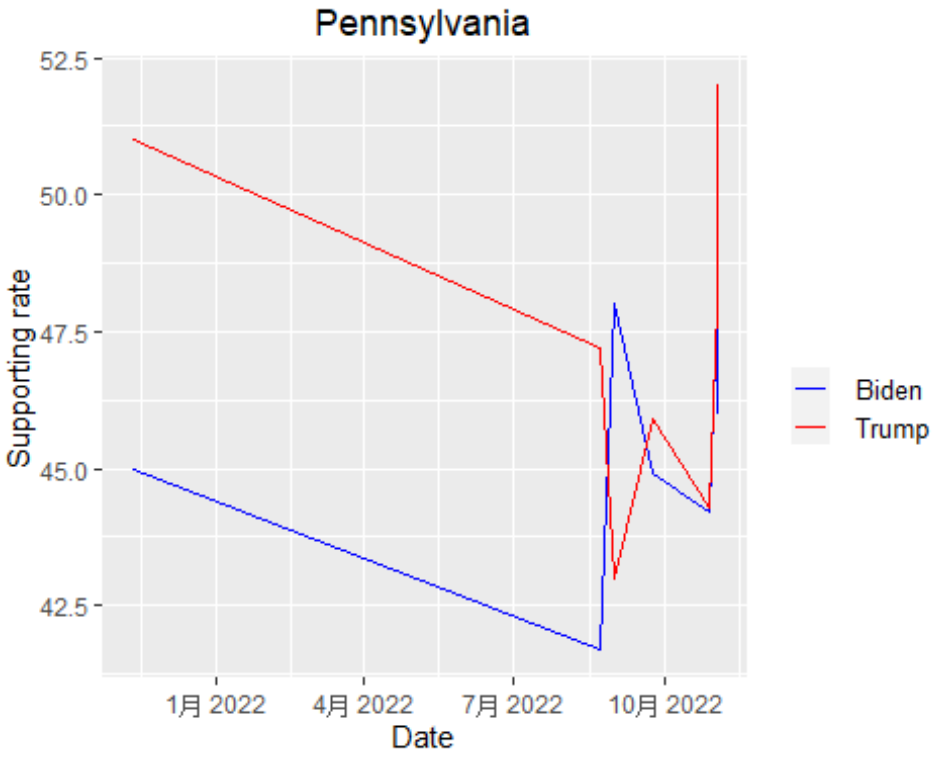

```
get_plot("Arizona")
```



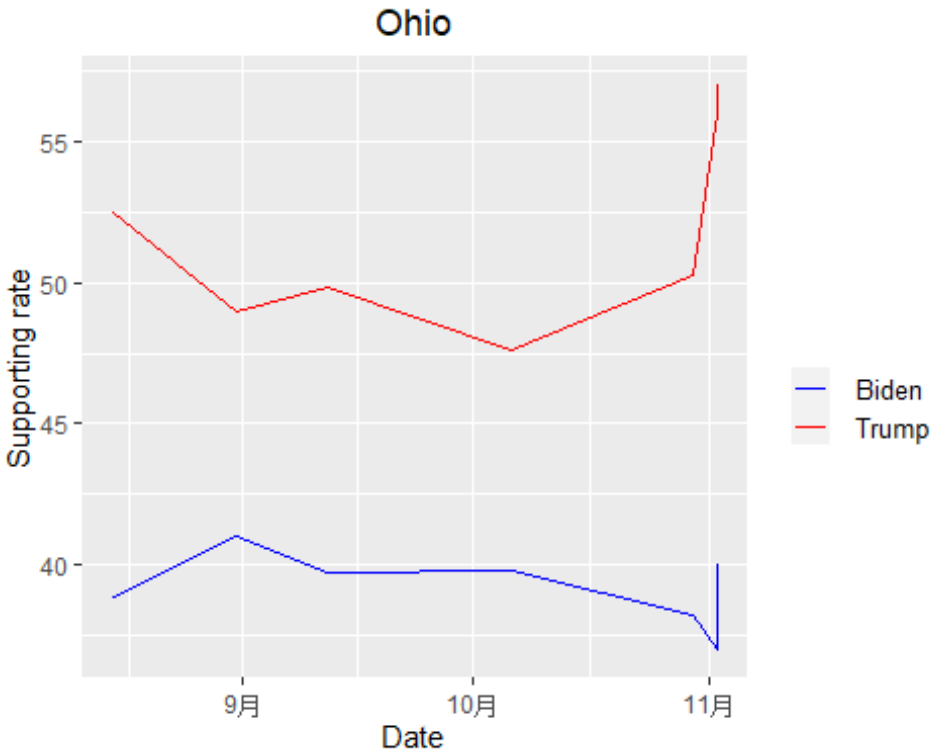

```
get_plot("New Hampshire")
```




```
get_plot("Pennsylvania")
```




```
get_plot("Ohio")
```



According to the plots above, we can see that Trump is leading Biden in all states except New Hampshire when it come to the midterm election

Now calculate the mean and the standard deviation of the spread for each state. Since there are still two years before the 2024 presidential election and I do not have a lot of polls for some states, I decide to keep all the states that I have at least two polls available.

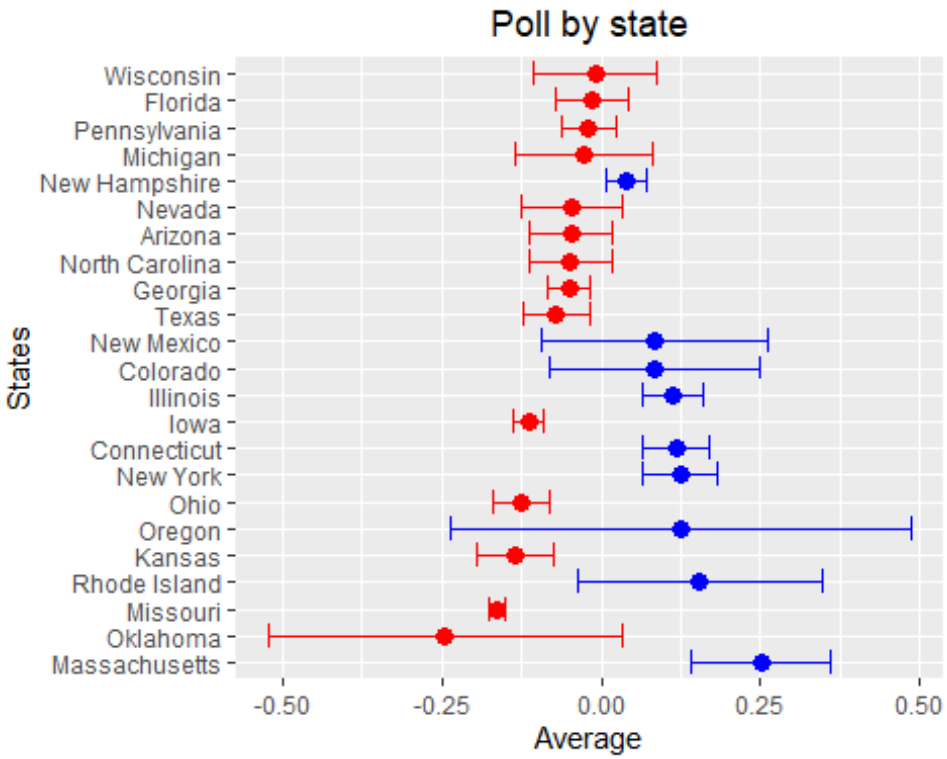
```
results <- president_polls |> group_by(race_id, state) |> summarize(ave
rage = mean(spread), standard_deviation = sd(spread), count = length(un
ique(poll_id))) |> select(race_id, state, average, standard_deviation,
count) |> filter(count > 1)
results
```

```
## # A tibble: 23 × 5
## # Groups:   race_id [23]
##    race_id state         average standard_deviation count
##      <dbl> <chr>           <dbl>              <dbl> <int>
##  1    8759 Arizona       -0.0477             0.0620     6
##  2    8765 Colorado       0.085              0.0184     2
##  3    8768 Connecticut    0.118              0.0210     3
##  4    8778 Florida       -0.0145             0.0540     6
##  5    8781 Georgia       -0.0503             0.0403     8
##  6    8788 Iowa          -0.114              0.0219     6
##  7    8794 Illinois       0.112              0.0195     3
##  8    8800 Kansas        -0.135              0.0240     3
##  9    8811 Massachusetts  0.251              0.0442     3
## 10    8820 Michigan      -0.0275             0.0672     4
## # … with 13 more rows
```

Plot the results with confidence intervals assuming the average is t-distributed. Put state on the y-axis and the average along the x-axis, using `geom_errorbar` to specify the confidence interval width about the average. Order the states by the absolute value of the average difference so the closer races are on top. Color them blue if Biden is winning and red if Trump is winning (don't show a legend).

```
# Calculate 95% confidence interval
standard_error <- results$standard_deviation / sqrt(results$count)
alpha <- 0.05
degrees_of_freedom <- results$count - 1
t_score = qt(p=alpha/2, df=degrees_of_freedom,lower.tail=F)
margin_error <- t_score * standard_error
lower_bound <- results$average - margin_error
upper_bound <- results$average + margin_error
results |> ggplot(aes(x = average, y = reorder(state, -abs(average))))
+ geom_point(aes(col = ifelse(average > 0, "blue", ifelse(average < 0,
"red", "black"))), size = 3) + geom_errorbar(aes(xmin = lower_bound, xm
ax = upper_bound, colour = ifelse(average > 0, "blue", ifelse(average <
 0, "red", "black")))) + scale_color_manual(values = c("blue" = "blue",
 "red"="red", "black"="black")) + xlab("Average") + ylab("States") + gg
```

```
title("Poll by state") + theme(legend.position="none") + theme(plot.tit
le = element_text(hjust = 0.5))
```



Now comes the important part. We need to predict how many electoral votes that Biden will win in the 2024 presidential election, so we need a table showing how many electoral votes that each state has. According to the demographic census in 2020, the electoral votes for each state have been updated and will firstly take effect in the 2024 presidential election. Below is the how I extract the table from the website.

The table is located in the following website: https://state.1keydata.com/state-electoral-votes.php

```
url <- "https://state.1keydata.com/state-electoral-votes.php"
dat <- read_html(url)
```

Use the `html_nodes` function and the `table` node to extract the table we need.

```
nodes <- dat |> html_nodes("table")
nodes

## {xml_nodeset (11)}
##  [1] <table border="0" class="container" cellpadding="0" cellspacing
="10" bgc ...
##  [2] <table border="0" cellpadding="0" bgcolor="FFFFFF"><tr valign="
top"><td> ...
##  [3] <table border="1" width="80%" class="content1">\n<tr bgcolor="0
```

```
033FF">\n ...
##  [4] <table border="1" class="content2">\n<tr bgcolor="0033FF">\n<td>
<center> ...
##  [5] <table border="1" width="80%" class="content1">\n<tr bgcolor="0
033FF">\n ...
##  [6] <table border="1" width="80%" class="content1">\n<tr bgcolor="0
033FF">\n ...
##  [7] <table border="0" cellpadding="0" cellspacing="0" width="100%">
<tr>\n<td ...
##  [8] <table border="0" cellpadding="0" cellspacing="0" width="100%">
<tr>\n<td ...
##  [9] <table border="1" width="80%" class="content1">\n<tr bgcolor="0
033FF">\n ...
## [10] <table border="0" cellpadding="0" cellspacing="0" width="100%">
<tr>\n<td ...
## [11] <table border="1" width="80%" class="content1">\n<tr bgcolor="0
033FF">\n ...
```

Our table of interest is the fourth table, so we extract it below and do some modifications.

```
votes <- html_table(nodes[[4]], header = TRUE) |> select(2, 3)
colnames(votes)[1] <- 'state'
colnames(votes)[2] <- 'electoral_votes'
votes
```

```
## # A tibble: 50 × 2
##    state          electoral_votes
##    <chr>                    <int>
##  1 California                  54
##  2 Texas                       40
##  3 Florida                     30
##  4 New York                    28
##  5 Illinois                    19
##  6 Pennsylvania                19
##  7 Ohio                        17
##  8 Georgia                     16
##  9 North Carolina              16
## 10 Michigan                    15
## # … with 40 more rows
```

The table above does not include District of Columbia, which has 3 electoral votes, so I add it into the table.

```
add <- data.frame("District of Columbia", 3)
names(add)=c("state","electoral_votes")
votes <- votes |> rbind(add) |> arrange(-electoral_votes, state)
votes
```

```
## # A tibble: 51 × 2
##    state          electoral_votes
```

```
##    <chr>                    <dbl>
##  1 California                  54
##  2 Texas                       40
##  3 Florida                     30
##  4 New York                    28
##  5 Illinois                    19
##  6 Pennsylvania                19
##  7 Ohio                        17
##  8 Georgia                     16
##  9 North Carolina              16
## 10 Michigan                    15
## # … with 41 more rows
```

Let's see how many electoral votes will be needed to win the 2024 presidential election.

```
total <- sum(votes$electoral_votes)
print(paste(total %/% 2 + 1, " electoral votes are needed to win the 20
24 presidential election"))
```

```
## [1] "270  electoral votes are needed to win the 2024 presidential el
ection"
```

Join this table and the "results" table above.

```
results <- left_join(results, votes, by = "state")
results
```

```
## # A tibble: 23 × 6
## # Groups:   race_id [23]
##    race_id state         average standard_deviation count electoral_
votes
##      <dbl> <chr>           <dbl>              <dbl> <int>
<dbl>
##  1    8759 Arizona       -0.0477             0.0620     6
   11
##  2    8765 Colorado       0.085              0.0184     2
   10
##  3    8768 Connecticut    0.118              0.0210     3
    7
##  4    8778 Florida       -0.0145             0.0540     6
   30
##  5    8781 Georgia       -0.0503             0.0403     8
   16
##  6    8788 Iowa          -0.114              0.0219     6
    6
##  7    8794 Illinois       0.112              0.0195     3
   19
##  8    8800 Kansas        -0.135              0.0240     3
    6
##  9    8811 Massachusetts  0.251              0.0442     3
```

```
   11
## 10    8820 Michigan      -0.0275             0.0672     4
   15
## # … with 13 more rows
```

Now implement a Bayesian approach. Firstly, let's use a basic prior ($\mu = 0, \sigma = 0.04$) assuming no knowledge about the historical elections and the election trends.

```
mu <- 0.00
tau <- 0.04
results_no_bias <- results |> mutate(prior_average = mu,
                             prior_sd = tau,
                             sigma = standard_deviation/sqrt(count),
                             B = sigma^2 / (sigma^2 + tau^2),
                             posterior_mean = B * mu + (1 - B) * averag
e,
                             posterior_sd = sqrt(1/ (1/sigma^2 + 1/tau^
2)),
                             Biden_win_prob = 1-pnorm(0, posterior_mean,
 posterior_sd)) |> select(-sigma, -B)
results_no_bias
```

```
## # A tibble: 23 × 11
## # Groups:   race_id [23]
##    race_id state   average stand…¹ count elect…² prior…³ prior…⁴ pos
te…⁵ poste…⁶
##      <dbl> <chr>     <dbl>   <dbl> <int>   <dbl>   <dbl>   <dbl>   <
dbl>   <dbl>
##  1    8759 Arizona -0.0477  0.0620     6      11       0    0.04 -0.
0341 0.0214
##  2    8765 Colora…  0.085   0.0184     2      10       0    0.04  0.
0769 0.0124
##  3    8768 Connec…  0.118   0.0210     3       7       0    0.04  0.
108  0.0116
##  4    8778 Florida -0.0145  0.0540     6      30       0    0.04 -0.
0111 0.0193
##  5    8781 Georgia -0.0503  0.0403     8      16       0    0.04 -0.
0447 0.0134
##  6    8788 Iowa    -0.114   0.0219     6       6       0    0.04 -0.
109  0.00872
##  7    8794 Illino…  0.112   0.0195     3      19       0    0.04  0.
104  0.0108
##  8    8800 Kansas  -0.135   0.0240     3       6       0    0.04 -0.
120  0.0131
##  9    8811 Massac…  0.251   0.0442     3      11       0    0.04  0.
178  0.0215
## 10    8820 Michig… -0.0275  0.0672     4      15       0    0.04 -0.
0161 0.0257
## # … with 13 more rows, 1 more variable: Biden_win_prob <dbl>, and ab
breviated
```

```
## #   variable names ¹standard_deviation, ²electoral_votes, ³prior_ave
rage,
## #   ⁴prior_sd, ⁵posterior_mean, ⁶posterior_sd
```

Show a table of states with the probability (in percentages) of the Biden winning ordered by that probability. If a probability is larger than 99%, show >99%, and if smaller than 1% then show <1%.

```
temp <- results_no_bias |> select(state, Biden_win_prob) |> mutate(Bide
n_win = paste(as.character(round(100 * Biden_win_prob, 2)), "%")) |> ar
range(-Biden_win_prob)
temp["Biden_win"][temp["Biden_win_prob"] > 0.99] <- "> 99 %"
temp["Biden_win"][temp["Biden_win_prob"] < 0.01] <- "< 1 %"
temp |> select(state, Biden_win)
```

```
## # A tibble: 23 × 3
## # Groups:   race_id [23]
##    race_id state         Biden_win
##      <dbl> <chr>         <chr>
##  1    8768 Connecticut   > 99 %
##  2    8794 Illinois      > 99 %
##  3    8811 Massachusetts > 99 %
##  4    8877 Rhode Island  > 99 %
##  5    8765 Colorado      > 99 %
##  6    8854 New Mexico    > 99 %
##  7    8860 New York      > 99 %
##  8    8869 Oregon        > 99 %
##  9    8848 New Hampshire > 99 %
## 10    8905 Wisconsin     40.14 %
## # … with 13 more rows
```

Note that there are several states that do not appear in the “results_no_bias” table above. This is because the election in these states are usually certain. As a result, I use the following strategy. If the state is not in the “results_no_bias” table, I consider the election results in these states are certain and count the electoral votes towards their party affiliation. To get the party affiliation, I extracted the following table from https://www.pewresearch.org/religion/religious-landscape-study/compare/party-affiliation/by/state/ using the similar web scraping strategy mentioned above.

```
url <- "https://www.pewresearch.org/religion/religious-landscape-study/
compare/party-affiliation/by/state/"
dat <- read_html(url)
nodes <- dat |> html_nodes("table")
nodes
```

```
## {xml_nodeset (1)}
## [1] <table class="ui\tcelled table" data-iframe-height>\n<thead><tr>
\n<th>Sta ...
```

```
affiliation <- html_table(nodes[[1]], header = TRUE)
colnames(affiliation)[1] <- 'state'
```

```
colnames(affiliation)[2] <- 'Rep'
colnames(affiliation)[3] <- 'None'
colnames(affiliation)[4] <- 'Dem'
affiliation
```

```
## # A tibble: 51 × 5
##    state                Rep   None  Dem   `Sample\tsize`
##    <chr>                <chr> <chr> <chr> <chr>
##  1 Alabama              52%   13%   35%   511
##  2 Alaska               39%   29%   32%   310
##  3 Arizona              40%   21%   39%   653
##  4 Arkansas             46%   16%   38%   311
##  5 California           30%   21%   49%   3,697
##  6 Colorado             41%   17%   42%   504
##  7 Connecticut          32%   18%   50%   377
##  8 Delaware             29%   17%   55%   301
##  9 District of Columbia 11%   15%   73%   303
## 10 Florida              37%   19%   44%   2,020
## # … with 41 more rows
```

The table above shows the percentage of the party affiliation that the adults in the states consider themselves as. Here, I consider the state as a "Blue state" if it has more adults that think they are Democrats. and a "Red state" if it has more adults that think they are Republicans. Otherwise, I label the state as a "White state" since it has no affiliation. As a result, I get the following table.

```
affiliation <- affiliation |> mutate(affiliation = ifelse(Dem > Rep, "B
", ifelse(Rep > Dem, "R", "W")))  |> select(-"Sample size")
affiliation
```

```
## # A tibble: 51 × 5
##    state                Rep   None  Dem   affiliation
##    <chr>                <chr> <chr> <chr> <chr>
##  1 Alabama              52%   13%   35%   R
##  2 Alaska               39%   29%   32%   R
##  3 Arizona              40%   21%   39%   R
##  4 Arkansas             46%   16%   38%   R
##  5 California           30%   21%   49%   B
##  6 Colorado             41%   17%   42%   B
##  7 Connecticut          32%   18%   50%   B
##  8 Delaware             29%   17%   55%   B
##  9 District of Columbia 11%   15%   73%   B
## 10 Florida              37%   19%   44%   B
## # … with 41 more rows
```

Now let's only keep the states that are not in the "results_no_bias" table since we will use a Monte Carlo simulation to determine the election results in those states.

```
affiliation <- affiliation |> filter(!state %in% results_no_bias$state)
affiliation
```

```
## # A tibble: 28 × 5
##    state                Rep   None  Dem   affiliation
##    <chr>                <chr> <chr> <chr> <chr>
##  1 Alabama              52%   13%   35%   R
##  2 Alaska               39%   29%   32%   R
##  3 Arkansas             46%   16%   38%   R
##  4 California           30%   21%   49%   B
##  5 Delaware             29%   17%   55%   B
##  6 District of Columbia 11%   15%   73%   B
##  7 Hawaii               28%   20%   51%   B
##  8 Idaho                49%   19%   32%   R
##  9 Indiana              42%   20%   37%   R
## 10 Kentucky             44%   13%   43%   R
## # … with 18 more rows
```

Let's assume that Biden manage to win in all blue states that remain in the "affiliation" table and lose in all red states that remain in the "affiliation" table. Before using the Monte Carlo simulation, let's see how many electoral votes that Biden has already won.

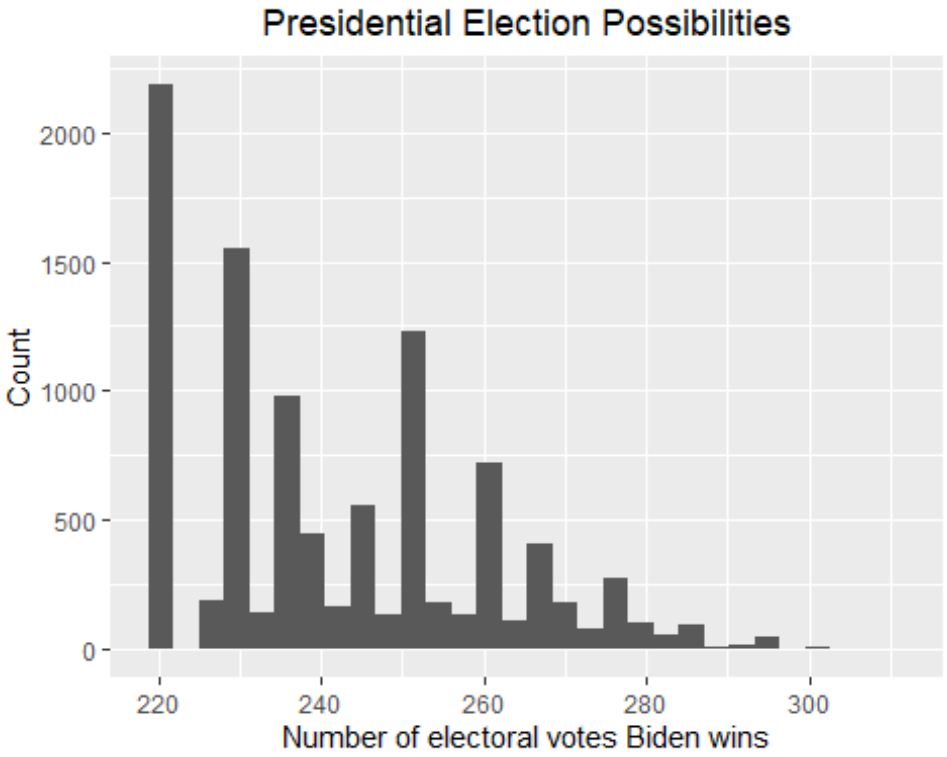
```
affiliation <- left_join(affiliation, votes, by = "state")
total_votes <- affiliation |> filter(affiliation == "B") |> pull(electo
ral_votes)
Biden_win_electoral_votes <- sum(total_votes)
print(paste("Based on the assumption, Biden has already won ", Biden_wi
n_electoral_votes, " electoral votes"))
```

```
## [1] "Based on the assumption, Biden has already won  125  electoral
votes"
```

Now Create a Monte Carlo simulation of each state's election using number of simulations `B=10000`, where each simulation outputs the total number of electoral votes that Biden will win in the 2024 presidential election. Assume each state's election is normally distributed with the posterior mean and standard error in the `results_no_bias` data frame. Show a histogram of the results for the number of electoral votes.

```
# We now start the Monte Carlo simulation based on the above result
set.seed(1)
B <- 10000
simulation <- function(){
  temp <- c()
  for (i in 1:length(results_no_bias$state)){
    temp <- c(sample(c(1, 0), size = 1, replace = TRUE, prob = c(pnorm
(0, mean = results_no_bias$posterior_mean[i], sd = results_no_bias$post
erior_sd[i], lower.tail = FALSE), pnorm(0, mean = results_no_bias$poste
rior_mean[i], sd = results_no_bias$posterior_sd[i], lower.tail = TRUE)))
 * results_no_bias$electoral_votes[i], temp)
  }
  sum(temp) + Biden_win_electoral_votes
```

```
}
result_mc <- data.frame(electoral_votes_count = replicate(B, simulation
()))
result_mc |> ggplot(aes(x = electoral_votes_count)) + geom_histogram()
+ xlab("Number of electoral votes Biden wins") + ylab("Count") + ggtitl
e("Presidential Election Possibilities") + theme(plot.title = element_t
ext(hjust = 0.5))
```



Show the probability that Biden wins the 2024 presidential election, assuming win means 270 or more electoral votes.

```
print(paste("According to the above plot, the probability that Biden wi
ns the 2024 presidential election is ", round(mean(result_mc >= 270) *
100, 2), "%"))
```

```
## [1] "According to the above plot, the probability that Biden wins th
e 2024 presidential election is  8.66 %"
```

Give the 80% credible interval of the 2024 presidential election.

```
# I use the credible_interval function in the ArchaeoPhases package to
get the 80% credible interval
result <- credible_interval(data = result_mc |> pull(electoral_votes_co
unt), level = 0.8, round_to = 0)

print(paste("According to the redible_interval function in the ArchaeoP
```

```
hases package, the 80% credible interval should be [", result$ci[1], ",
 ", result$ci[2], "]"))
```

```
## [1] "According to the redible_interval function in the ArchaeoPhases
 package, the 80% credible interval should be [ 221 ,  261 ]"
```

Now including a bias with average `bias_avg` and standard error of `bias_sd` to add the uncertainty to the election.

```
# According to section 16.8.4 in the book, I set bias_sd to be 0.03. I
also set bias_avg to be 0.00 since I do not think the bias will go towa
rds either party
mu <- 0.00
tau <- 0.04
bias_avg <- 0.00
bias_sd <- 0.03
results_bias <- results |> mutate(prior_average = mu,
                                  prior_sd = tau,
                                  sigma = sqrt(standard_deviation^2/count  +
 bias_sd^2),
                                  B = sigma^2 / (sigma^2 + tau^2),
                                  posterior_mean = B * mu + (1 - B) * averag
e,
                                  posterior_sd = sqrt(1/ (1/sigma^2 + 1/tau^
2)),
                                  Biden_win_prob = 1-pnorm(0, posterior_mean,
 posterior_sd)) |> select(-sigma, -B)
results_bias
```

```
## # A tibble: 23 × 11
## # Groups:   race_id [23]
##    race_id state  average stand…¹ count elect…² prior…³ prior…⁴ post
er…⁵ poste…⁶
##      <dbl> <chr>    <dbl>   <dbl> <int>   <dbl>   <dbl>   <dbl>    <
dbl>   <dbl>
##  1    8759 Arizo… -0.0477  0.0620     6      11       0    0.04 -0.0
243   0.0280
##  2    8765 Color…  0.085   0.0184     2      10       0    0.04  0.0
510   0.0253
##  3    8768 Conne…  0.118   0.0210     3       7       0    0.04  0.0
713   0.0252
##  4    8778 Flori… -0.0145  0.0540     6      30       0    0.04 -0.0
0777  0.0273
##  5    8781 Georg… -0.0503  0.0403     8      16       0    0.04 -0.0
298   0.0256
##  6    8788 Iowa   -0.114   0.0219     6       6       0    0.04 -0.0
708   0.0247
##  7    8794 Illin…  0.112   0.0195     3      19       0    0.04  0.0
682   0.0250
##  8    8800 Kansas -0.135   0.0240     3       6       0    0.04 -0.0
800   0.0255
```

```
##  9    8811 Massa…  0.251   0.0442     3      11       0    0.04  0.1
27    0.0281
## 10    8820 Michi… -0.0275  0.0672     4      15       0    0.04 -0.0
121   0.0299
## # … with 13 more rows, 1 more variable: Biden_win_prob <dbl>, and ab
breviated
## #   variable names ¹standard_deviation, ²electoral_votes, ³prior_ave
rage,
## #   ⁴prior_sd, ⁵posterior_mean, ⁶posterior_sd
```

Print out the table of states with Biden's probabilities of winning. If a probability is larger than 99%, show >99%, and if smaller than 1% then show <1%.

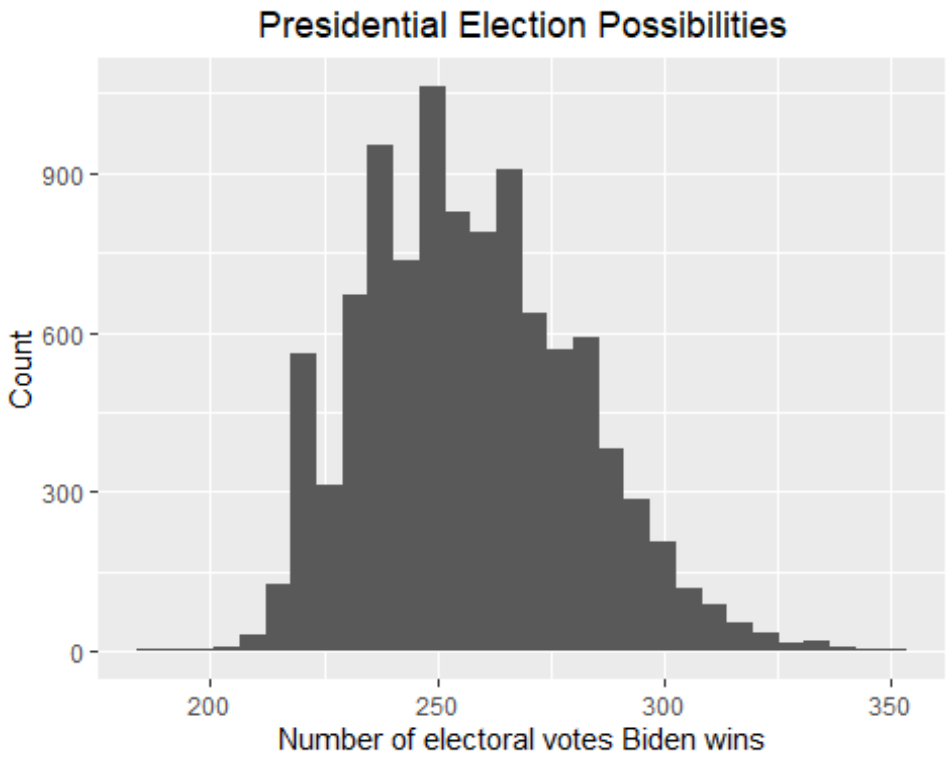
```
# Print out the table of states with Biden's probabilities of winning
temp <- results_bias |> select(state, Biden_win_prob) |> mutate(Biden_w
in = paste(as.character(round(100 * Biden_win_prob, 2)), "%"))
temp["Biden_win"][temp["Biden_win_prob"] > 0.99] <- "> 99 %"
temp["Biden_win"][temp["Biden_win_prob"] < 0.01] <- "< 1 %"
temp |> select(state, Biden_win)
```

```
## # A tibble: 23 × 3
## # Groups:   race_id [23]
##    race_id state         Biden_win
##      <dbl> <chr>         <chr>
##  1    8759 Arizona       19.28 %
##  2    8765 Colorado      97.79 %
##  3    8768 Connecticut   > 99 %
##  4    8778 Florida       38.78 %
##  5    8781 Georgia       12.18 %
##  6    8788 Iowa          < 1 %
##  7    8794 Illinois      > 99 %
##  8    8800 Kansas        < 1 %
##  9    8811 Massachusetts > 99 %
## 10    8820 Michigan      34.26 %
## # … with 13 more rows
```

Now run a simulation to get a final answer for the total of electoral votes that Biden will win in the 2024 presidential election.

```
# Use the updated results table to perform the Monte Carlo simulation
set.seed(1)
B <- 10000
simulation <- function(){
  temp <- c()
  for (i in 1:length(results_bias$state)){
    temp <- c(sample(c(1, 0), size = 1, replace = TRUE, prob = c(pnorm
(0, mean = results_bias$posterior_mean[i], sd = results_bias$posterior_
sd[i], lower.tail = FALSE), pnorm(0, mean = results_bias$posterior_mean
[i], sd = results_bias$posterior_sd[i], lower.tail = TRUE))) * results_
bias$electoral_votes[i], temp)
  }
```

```
  sum(temp) + Biden_win_electoral_votes
}
result_mc <- data.frame(electoral_votes_count = replicate(B, simulation
()))
result_mc |> ggplot(aes(x = electoral_votes_count)) + geom_histogram()
+ xlab("Number of electoral votes Biden wins") + ylab("Count") + ggtitl
e("Presidential Election Possibilities") + theme(plot.title = element_t
ext(hjust = 0.5))
```



Let's see now how likely Biden will win against Trump with the new posterior.

```
print(paste("According to the above plot, the probability that the Demo
crats control the Senate is ", round(mean(result_mc >= 270) * 100, 2),
"%"))
```

```
## [1] "According to the above plot, the probability that the Democrats
 control the Senate is  29.76 %"
```

Give the 80% credible interval of the 2024 presidential election.

```
# I use the credible_interval function in the ArchaeoPhases package to
get the 80% credible interval
result <- credible_interval(data = result_mc |> pull(electoral_votes_co
unt), level = 0.8, round_to = 0)

print(paste("According to the redible_interval function in the ArchaeoP
hases package, the 80% credible interval should be [", result$ci[1], ",
 ", result$ci[2], "]"))
```

```
## [1] "According to the redible_interval function in the ArchaeoPhases
 package, the 80% credible interval should be [ 221 ,  279 ]"
```

According to the above results, by adding a bias term, the election results become more uncertain and Biden's winning rate increases. However, Biden will still suffer great loss regardless of the bias term is added or not. However, the above results are based on the prior that has no knowledge about the previous presidential elections and we are subjected to limited poll data.

The previous analysis may give a more plausible prediction if we can set a better prior for each state. To achieve this, we use the poll results from the 2020 presidential election. This is because this election is also between Biden and Trump. First, let's get the data from FiveThirtyEight again and conduct necessary data wrangling. The data is available at https://projects.fivethirtyeight.com/polls/data/president_polls_historical.csv

```
president_polls_2020 <- read_csv("https://projects.fivethirtyeight.com/
polls/data/president_polls_historical.csv")
president_polls_2020 <- president_polls_2020 |> filter(cycle == 2020 &
stage == "general" & !is.na(state) & candidate_name %in% c("Joe Biden",
 "Donald Trump") & fte_grade <= "C" | fte_grade == "C+")
president_polls_2020["start_date"] <- mdy(president_polls_2020$start_da
te)
president_polls_2020["end_date"] <- mdy(president_polls_2020$end_date)
president_polls_2020["election_date"] <- mdy(president_polls_2020$elect
ion_date)
president_polls_2020["candidate_name"][president_polls_2020["candidate_
name"] == "Joe Biden"] = "Biden"
president_polls_2020["candidate_name"][president_polls_2020["candidate_
name"] == "Donald Trump"] = "Trump"
president_polls_2020
```

```
## # A tibble: 9,597 × 42
##    poll_id pollster_id pollster  spons…¹ spons…² displ…³ polls…⁴ pol
ls…⁵ fte_g…⁶
##      <dbl>       <dbl> <chr>       <dbl> <chr>   <chr>     <dbl> <ch
r>   <chr>
##  1   72621         383 PPP            NA <NA>    Public…     263 Pub
lic… A-
##  2   72621         383 PPP            NA <NA>    Public…     263 Pub
lic… A-
##  3   72647         461 Susqueha…      NA <NA>    Susque…     326 Sus
que… B+
##  4   72647         461 Susqueha…      NA <NA>    Susque…     326 Sus
que… B+
##  5   72722         235 InsiderA…     364 FOX35 … Inside…     243 Opi
nio… B
##  6   72722         235 InsiderA…     364 FOX35 … Inside…     243 Opi
nio… B
##  7   72806        1250 Trafalga…      NA <NA>    Trafal…     338 Tra
```

```
fal… A-
##  8   72806        1250 Trafalga…      NA <NA>    Trafal…     338 Tra
fal… A-
##  9   72861        1250 Trafalga…      NA <NA>    Trafal…     338 Tra
fal… A-
## 10   72861        1250 Trafalga…      NA <NA>    Trafal…     338 Tra
fal… A-
## # … with 9,587 more rows, 33 more variables: methodology <chr>, stat
e <chr>,
## #   start_date <date>, end_date <date>, sponsor_candidate_id <dbl>,
## #   sponsor_candidate <chr>, sponsor_candidate_party <chr>, question
_id <dbl>,
## #   sample_size <dbl>, population <chr>, subpopulation <lgl>,
## #   population_full <chr>, tracking <lgl>, created_at <chr>, notes <
chr>,
## #   url <chr>, source <dbl>, internal <lgl>, partisan <chr>, race_id
 <dbl>,
## #   cycle <dbl>, office_type <chr>, seat_number <dbl>, seat_name <lg
l>, …
```

Use "pivot_wider" to merge the rows and create the "spread" column that indicates the supporting rate spread between Biden and Trump.

```
president_polls_2020 <- president_polls_2020 |> select(c(-answer, -cand
idate_id, -party)) |> pivot_wider(names_from = "candidate_name", values
_from = "pct") |> mutate(spread = (Biden - Trump) / 100) |> filter(!is.
na(spread))
president_polls_2020
```

```
## # A tibble: 4,407 × 44
##    poll_id pollster_id pollster  spons…¹ spons…² displ…³ polls…⁴ pol
ls…⁵ fte_g…⁶
##      <dbl>       <dbl> <chr>       <dbl> <chr>   <chr>     <dbl> <ch
r>   <chr>
##  1   72621         383 PPP            NA <NA>    Public…     263 Pub
lic… A-
##  2   72647         461 Susqueha…      NA <NA>    Susque…     326 Sus
que… B+
##  3   72722         235 InsiderA…     364 FOX35 … Inside…     243 Opi
nio… B
##  4   72806        1250 Trafalga…      NA <NA>    Trafal…     338 Tra
fal… A-
##  5   72861        1250 Trafalga…      NA <NA>    Trafal…     338 Tra
fal… A-
##  6   72862        1250 Trafalga…      NA <NA>    Trafal…     338 Tra
fal… A-
##  7   72864        1240 Øptimus        NA <NA>    Øptimus     245 Opt
imus B/C
##  8   72754        1613 Universi…      NA <NA>    Univer…     609 Uni
ver… B/C
```

```
##  9   72757        1365 Change R…      NA <NA>    Change…      48 Cha
nge… B-
## 10   72762        1365 Change R…      NA <NA>    Change…      48 Cha
nge… B-
## # … with 4,397 more rows, 35 more variables: methodology <chr>, stat
e <chr>,
## #   start_date <date>, end_date <date>, sponsor_candidate_id <dbl>,
## #   sponsor_candidate <chr>, sponsor_candidate_party <chr>, question
_id <dbl>,
## #   sample_size <dbl>, population <chr>, subpopulation <lgl>,
## #   population_full <chr>, tracking <lgl>, created_at <chr>, notes <
chr>,
## #   url <chr>, source <dbl>, internal <lgl>, partisan <chr>, race_id
 <dbl>,
## #   cycle <dbl>, office_type <chr>, seat_number <dbl>, seat_name <lg
l>, …
```

After consideration, I think a good prior should be set as a normal distribution since the election result is usually normally distributed. The mean of prior should be the average of the spread within the final month of the election since we want to predict the results that are close to the election day. The standard deviation of the prior should be variance between 2018 to 2020. This is because the earliest poll data from FiveThirtyEight is since 2018 and I want the standard deviation to be relatively large to account for uncertainties since I am predicting the 2024 presidential election using the data in 2022 and a lot can happen during the next two years. Let's find the mean and the standard deviation of the prior following the definitions above.

```
temp_1 <- president_polls_2020 |> filter(start_date > election_date - m
onths(1)) |> group_by(state) |> summarize(prior_average = mean(spread))
 |> filter(!is.na(prior_average))
temp_1
```

```
## # A tibble: 54 × 2
##    state                prior_average
##    <chr>                        <dbl>
##  1 Alabama                    -0.195
##  2 Alaska                     -0.0846
##  3 Arizona                     0.0349
##  4 Arkansas                   -0.210
##  5 California                  0.278
##  6 Colorado                    0.134
##  7 Connecticut                 0.239
##  8 Delaware                    0.229
##  9 District of Columbia        0.863
## 10 Florida                     0.0207
## # … with 44 more rows
```

```
temp_2 <- president_polls_2020 |> group_by(state) |> summarize(prior_sd
 = sd(spread)) |> filter(!is.na(prior_sd))
temp_2
```

```
## # A tibble: 54 × 2
##    state                prior_sd
##    <chr>                   <dbl>
##  1 Alabama                0.0464
##  2 Alaska                 0.0300
##  3 Arizona                0.0368
##  4 Arkansas               0.0548
##  5 California             0.0380
##  6 Colorado               0.0391
##  7 Connecticut            0.0383
##  8 Delaware               0.0455
##  9 District of Columbia   0.0494
## 10 Florida                0.0317
## # … with 44 more rows
```

Note that there are more than 50 rows in the tables above. This is because there are polls for some regions of one state, but this does not matter since we only want to have the prior mean and standard deviation that are in "results" table above. Now we add what we have found into the "results" table by joining it with the "temp_1" and "temp_2" table.

```
updated_results <- left_join(results, temp_1, by = "state")
updated_results <- left_join(updated_results, temp_2, by = "state")
updated_results
```

```
## # A tibble: 23 × 8
## # Groups:   race_id [23]
##    race_id state         average standard_devia…¹ count elect…² prio
r_…³ prior…⁴
##      <dbl> <chr>           <dbl>            <dbl> <int>   <dbl>    <
dbl>   <dbl>
##  1    8759 Arizona       -0.0477           0.0620     6      11  3.4
9e-2  0.0368
##  2    8765 Colorado       0.085            0.0184     2      10  1.3
4e-1  0.0391
##  3    8768 Connecticut    0.118            0.0210     3       7  2.3
9e-1  0.0383
##  4    8778 Florida       -0.0145           0.0540     6      30  2.0
7e-2  0.0317
##  5    8781 Georgia       -0.0503           0.0403     8      16  2.1
0e-2  0.0362
##  6    8788 Iowa          -0.114            0.0219     6       6 -1.0
0e-4  0.0303
##  7    8794 Illinois       0.112            0.0195     3      19  1.6
8e-1  0.0310
##  8    8800 Kansas        -0.135            0.0240     3       6 -1.0
```

```
5e-1  0.0347
##  9    8811 Massachusetts  0.251            0.0442     3      11  3.8
4e-1  0.0312
## 10    8820 Michigan      -0.0275           0.0672     4      15  7.5
3e-2  0.0385
## # … with 13 more rows, and abbreviated variable names ¹standard_devi
ation,
## #   ²electoral_votes, ³prior_average, ⁴prior_sd
```

Perform a Bayesian approach using the updated prior average and standard deviation. First, let's assume no bias term.

```
updated_results_no_bias <- updated_results |> mutate(sigma = standard_d
eviation/sqrt(count),
                              B = sigma^2 / (sigma^2 + prior_sd^2),
                              posterior_mean = B * prior_average + (1 -
B) * average,
                              posterior_sd = sqrt(1/ (1/sigma^2 + 1/prio
r_sd^2)),
                              Biden_win_prob = 1-pnorm(0, posterior_mean,
 posterior_sd)) |> select(-sigma, -B)
updated_results_no_bias
```

```
## # A tibble: 23 × 11
## # Groups:   race_id [23]
##    race_id state average stand…¹ count elect…² prior_…³ prior…⁴ post
er…⁵ poste…⁶
##      <dbl> <chr>   <dbl>   <dbl> <int>   <dbl>    <dbl>   <dbl>    <
dbl>   <dbl>
##  1    8759 Ariz… -0.0477  0.0620     6      11  3.49e-2  0.0368 -0.0
212  0.0209
##  2    8765 Colo…  0.085   0.0184     2      10  1.34e-1  0.0391  0.0
899  0.0123
##  3    8768 Conn…  0.118   0.0210     3       7  2.39e-1  0.0383  0.1
29   0.0116
##  4    8778 Flor… -0.0145  0.0540     6      30  2.07e-2  0.0317 -0.0
0302 0.0181
##  5    8781 Geor… -0.0503  0.0403     8      16  2.10e-2  0.0362 -0.0
407  0.0133
##  6    8788 Iowa  -0.114   0.0219     6       6 -1.00e-4  0.0303 -0.1
05   0.00857
##  7    8794 Illi…  0.112   0.0195     3      19  1.68e-1  0.0310  0.1
19   0.0106
##  8    8800 Kans… -0.135   0.0240     3       6 -1.05e-1  0.0347 -0.1
31   0.0129
##  9    8811 Mass…  0.251   0.0442     3      11  3.84e-1  0.0312  0.3
04   0.0198
## 10    8820 Mich… -0.0275  0.0672     4      15  7.53e-2  0.0385  0.0
170  0.0253
## # … with 13 more rows, 1 more variable: Biden_win_prob <dbl>, and ab
```

```
breviated
## #   variable names ¹standard_deviation, ²electoral_votes, ³prior_ave
rage,
## #   ⁴prior_sd, ⁵posterior_mean, ⁶posterior_sd
```

Print out the table of states with Biden's probabilities of winning. If a probability is larger than 99%, show >99%, and if smaller than 1% then show <1%.

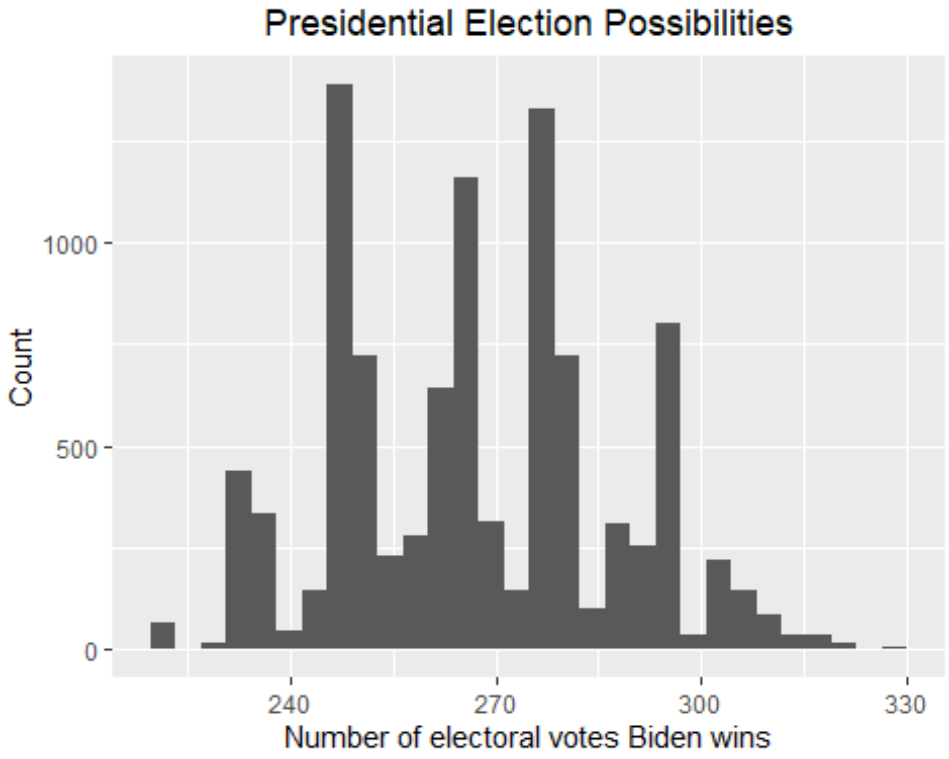
```
temp <- updated_results_no_bias |> select(state, Biden_win_prob) |> mut
ate(Biden_win = paste(as.character(round(100 * Biden_win_prob, 2)), "%
")) |> arrange(-Biden_win_prob)
temp["Biden_win"][temp["Biden_win_prob"] > 0.99] <- "> 99 %"
temp["Biden_win"][temp["Biden_win_prob"] < 0.01] <- "< 1 %"
temp |> select(state, Biden_win)
```

```
## # A tibble: 23 × 3
## # Groups:   race_id [23]
##    race_id state         Biden_win
##      <dbl> <chr>         <chr>
##  1    8768 Connecticut   > 99 %
##  2    8794 Illinois      > 99 %
##  3    8811 Massachusetts > 99 %
##  4    8860 New York      > 99 %
##  5    8869 Oregon        > 99 %
##  6    8877 Rhode Island  > 99 %
##  7    8765 Colorado      > 99 %
##  8    8854 New Mexico    > 99 %
##  9    8848 New Hampshire > 99 %
## 10    8905 Wisconsin     87.12 %
## # … with 13 more rows
```

Create a Monte Carlo simulation using the updated posterior average and standard deviation.

```
# We now start the Monte Carlo simulation based on the above result
set.seed(1)
B <- 10000
simulation <- function(){
  temp <- c()
  for (i in 1:length(updated_results_no_bias$state)){
    temp <- c(sample(c(1, 0), size = 1, replace = TRUE, prob = c(pnorm
(0, mean = updated_results_no_bias$posterior_mean[i], sd = updated_resu
lts_no_bias$posterior_sd[i], lower.tail = FALSE), pnorm(0, mean = updat
ed_results_no_bias$posterior_mean[i], sd = updated_results_no_bias$post
erior_sd[i], lower.tail = TRUE))) * updated_results_no_bias$electoral_v
otes[i], temp)
  }
  sum(temp) + Biden_win_electoral_votes
}
result_mc <- data.frame(electoral_votes_count = replicate(B, simulation
()))
```

```
result_mc |> ggplot(aes(x = electoral_votes_count)) + geom_histogram()
+ xlab("Number of electoral votes Biden wins") + ylab("Count") + ggtitl
e("Presidential Election Possibilities") + theme(plot.title = element_t
ext(hjust = 0.5))
```



Let's see now how likely Biden will win against Trump with the new posterior.

```
print(paste("According to the above plot, the probability that Biden wi
ns the 2024 presidential election is ", round(mean(result_mc >= 270) *
100, 2), "%"))
```

```
## [1] "According to the above plot, the probability that Biden wins th
e 2024 presidential election is  44.93 %"
```

Give the 80% credible interval of the 2024 presidential election.

```
# I use the credible_interval function in the ArchaeoPhases package to
get the 80% credible interval
result <- credible_interval(data = result_mc |> pull(electoral_votes_co
unt), level = 0.8, round_to = 0)

print(paste("According to the redible_interval function in the ArchaeoP
hases package, the 80% credible interval should be [", result$ci[1], ",
 ", result$ci[2], "]"))
```

```
## [1] "According to the redible_interval function in the ArchaeoPhases
 package, the 80% credible interval should be [ 246 ,  295 ]"
```

Now including a bias with average `bias_avg` and standard error of `bias_sd` to add the uncertainty to the election. Print out the table of states with Biden's probabilities of winning.

```
# According to section 16.8.4 in the book, I set bias_sd to be 0.03. I
also set bias_avg to be 0.00 since I do not think the bias will go towa
rds either party
bias_avg <- 0.00
bias_sd <- 0.03
updated_results_bias <- updated_results |> mutate(sigma = sqrt(standard
_deviation^2/count  + bias_sd^2),
                             B = sigma^2 / (sigma^2 + prior_sd^2),
                             posterior_mean = B * prior_average + (1 -
B) * average,
                             posterior_sd = sqrt(1/ (1/sigma^2 + 1/prio
r_sd^2)),
                             Biden_win_prob = 1-pnorm(0, posterior_mean,
 posterior_sd)) |> select(-sigma, -B)


updated_results_bias
```

```
## # A tibble: 23 × 11
## # Groups:   race_id [23]
##    race_id state average stand…¹ count elect…² prior_…³ prior…⁴ post
er…⁵ poste…⁶
##      <dbl> <chr>   <dbl>   <dbl> <int>   <dbl>    <dbl>   <dbl>    <
dbl>   <dbl>
##  1    8759 Ariz… -0.0477  0.0620     6      11  3.49e-2  0.0368 -0.0
0375  0.0268
##  2    8765 Colo…  0.085   0.0184     2      10  1.34e-1  0.0391  0.1
05    0.0251
##  3    8768 Conn…  0.118   0.0210     3       7  2.39e-1  0.0383  0.1
68    0.0247
##  4    8778 Flor… -0.0145  0.0540     6      30  2.07e-2  0.0317  0.0
0591  0.0241
##  5    8781 Geor… -0.0503  0.0403     8      16  2.10e-2  0.0362 -0.0
177   0.0245
##  6    8788 Iowa  -0.114   0.0219     6       6 -1.00e-4  0.0303 -0.0
553   0.0218
##  7    8794 Illi…  0.112   0.0195     3      19  1.68e-1  0.0310  0.1
41    0.0223
##  8    8800 Kans… -0.135   0.0240     3       6 -1.05e-1  0.0347 -0.1
21    0.0239
##  9    8811 Mass…  0.251   0.0442     3      11  3.84e-1  0.0312  0.3
33    0.0245
## 10    8820 Mich… -0.0275  0.0672     4      15  7.53e-2  0.0385  0.0
320   0.0292
## # … with 13 more rows, 1 more variable: Biden_win_prob <dbl>, and ab
breviated
```

```
## #   variable names ¹standard_deviation, ²electoral_votes, ³prior_ave
rage,
## #   ⁴prior_sd, ⁵posterior_mean, ⁶posterior_sd
```

Print out the table of states with Biden's probabilities of winning. If a probability is larger than 99%, show >99%, and if smaller than 1% then show <1%.

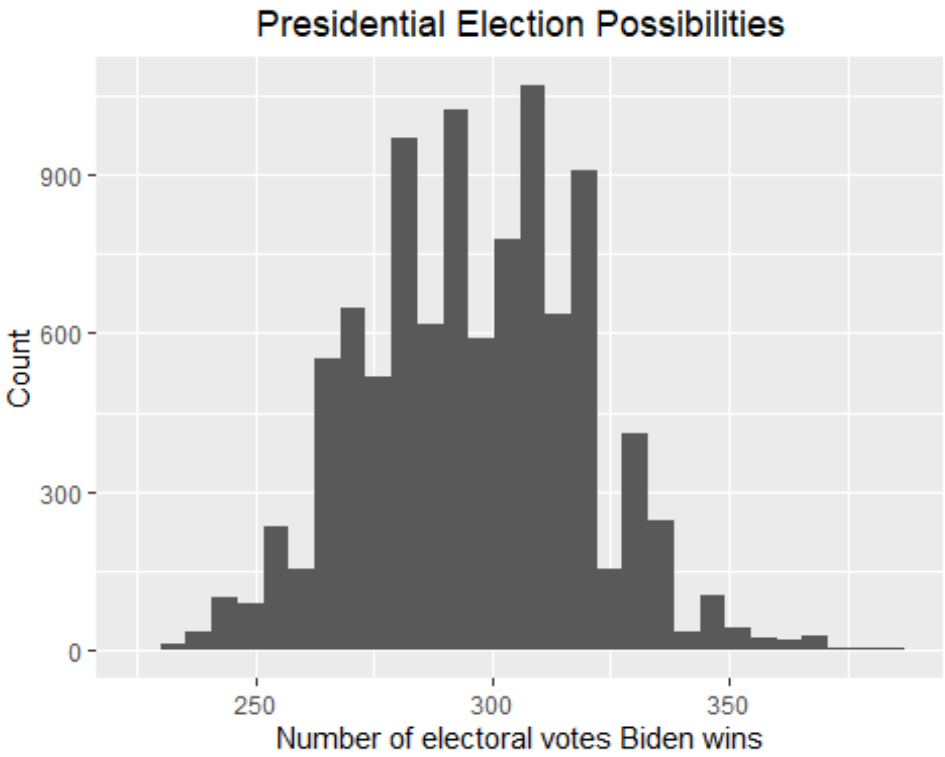
```
# Print out the table of states with Biden's probabilities of winning
temp <- updated_results_bias |> select(state, Biden_win_prob) |> mutate
(Biden_win = paste(as.character(round(100 * Biden_win_prob, 2)), "%"))
temp["Biden_win"][temp["Biden_win_prob"] > 0.99] <- "> 99 %"
temp["Biden_win"][temp["Biden_win_prob"] < 0.01] <- "< 1 %"
temp |> select(state, Biden_win)
```

```
## # A tibble: 23 × 3
## # Groups:   race_id [23]
##    race_id state         Biden_win
##      <dbl> <chr>         <chr>
##  1    8759 Arizona       44.45 %
##  2    8765 Colorado      > 99 %
##  3    8768 Connecticut   > 99 %
##  4    8778 Florida       59.68 %
##  5    8781 Georgia       23.44 %
##  6    8788 Iowa          < 1 %
##  7    8794 Illinois      > 99 %
##  8    8800 Kansas        < 1 %
##  9    8811 Massachusetts > 99 %
## 10    8820 Michigan      86.27 %
## # … with 13 more rows
```

Create a Monte Carlo simulation using the updated posterior average and standard deviation.

```
# We now start the Monte Carlo simulation based on the above result
set.seed(1)
B <- 10000
simulation <- function(){
  temp <- c()
  for (i in 1:length(updated_results_bias$state)){
    temp <- c(sample(c(1, 0), size = 1, replace = TRUE, prob = c(pnorm
(0, mean = updated_results_bias$posterior_mean[i], sd = updated_results
_bias$posterior_sd[i], lower.tail = FALSE), pnorm(0, mean = updated_res
ults_bias$posterior_mean[i], sd = updated_results_bias$posterior_sd[i],
 lower.tail = TRUE))) * updated_results_bias$electoral_votes[i], temp)
  }
  sum(temp) + Biden_win_electoral_votes
}
result_mc <- data.frame(electoral_votes_count = replicate(B, simulation
()))
result_mc |> ggplot(aes(x = electoral_votes_count)) + geom_histogram()
+ xlab("Number of electoral votes Biden wins") + ylab("Count") + ggtitl
```

```
e("Presidential Election Possibilities") + theme(plot.title = element_t
ext(hjust = 0.5))
```



Let's see now how likely Biden will win against Trump with the new posterior.

```
print(paste("According to the above plot, the probability that Biden wi
ns the 2024 presidential election is ", round(mean(result_mc >= 270) *
100, 2), "%"))
```

```
## [1] "According to the above plot, the probability that Biden wins th
e 2024 presidential election is  87.49 %"
```

Give the 80% credible interval of the 2024 presidential election.

```
# I use the credible_interval function in the ArchaeoPhases package to
get the 80% credible interval
result <- credible_interval(data = result_mc |> pull(electoral_votes_co
unt), level = 0.8, round_to = 0)

print(paste("According to the redible_interval function in the ArchaeoP
hases package, the 80% credible interval should be [", result$ci[1], ",
 ", result$ci[2], "]"))
```

```
## [1] "According to the redible_interval function in the ArchaeoPhases
 package, the 80% credible interval should be [ 261 ,  317 ]"
```

According to the results above, we can see that Biden's winning rate increases greatly against Trump in both cases (with or without the bias), and it suggests that it

is Biden who are supposed to win the election, which overturns the results generated using our previous prior. This is anticipated since for the last month of the election, people have seen Trump's performance for the four years but they have not seen Biden's performance. As a result, the polls will go against Trump and favor Biden. However, in the 2024 presidential election, people will see how Biden performed in the four years and the polls are more likely to go against Biden.

Let's have another try by assuming that in 2024, people have seen both Biden and Trump's performance during their presidency and their favor towards both candidates is mitigated, namely they do not favor strongly for either Biden or Trump. Based on this assumption, we can have another prior with the prior mean to be the average of the current poll average and the average of the polls in the last month of the 2020 presidential election, with the prior standard deviation the same as the one in our second try.

```
updated_results["prior_average"] <- (updated_results$average + updated_
results$prior_average) / 2
updated_results
```

```
## # A tibble: 23 × 8
## # Groups:   race_id [23]
##    race_id state         average standard_devia…¹ count elect…² prio
r_…³ prior…⁴
##      <dbl> <chr>           <dbl>            <dbl> <int>   <dbl>    <
dbl>   <dbl>
##  1    8759 Arizona       -0.0477           0.0620     6      11 -0.0
0640  0.0368
##  2    8765 Colorado       0.085            0.0184     2      10  0.1
10    0.0391
##  3    8768 Connecticut    0.118            0.0210     3       7  0.1
78    0.0383
##  4    8778 Florida       -0.0145           0.0540     6      30  0.0
0310  0.0317
##  5    8781 Georgia       -0.0503           0.0403     8      16 -0.0
147   0.0362
##  6    8788 Iowa          -0.114            0.0219     6       6 -0.0
571   0.0303
##  7    8794 Illinois       0.112            0.0195     3      19  0.1
40    0.0310
##  8    8800 Kansas        -0.135            0.0240     3       6 -0.1
20    0.0347
##  9    8811 Massachusetts  0.251            0.0442     3      11  0.3
18    0.0312
## 10    8820 Michigan      -0.0275           0.0672     4      15  0.0
239   0.0385
## # … with 13 more rows, and abbreviated variable names ¹standard_devi
ation,
## #   ²electoral_votes, ³prior_average, ⁴prior_sd
```

Perform a Bayesian approach using the updated prior average and standard deviation. First, let's assume no bias term.

```
updated_results_no_bias <- updated_results |> mutate(sigma = standard_d
eviation/sqrt(count),
                              B = sigma^2 / (sigma^2 + prior_sd^2),
                              posterior_mean = B * prior_average + (1 -
B) * average,
                              posterior_sd = sqrt(1/ (1/sigma^2 + 1/prio
r_sd^2)),
                              Biden_win_prob = 1-pnorm(0, posterior_mean,
 posterior_sd)) |> select(-sigma, -B)
updated_results_no_bias
```

```
## # A tibble: 23 × 11
## # Groups:   race_id [23]
##    race_id state average stand…¹ count elect…² prior_…³ prior…⁴ post
er…⁵ poste…⁶
##      <dbl> <chr>   <dbl>   <dbl> <int>   <dbl>    <dbl>   <dbl>    <
dbl>   <dbl>
##  1    8759 Ariz… -0.0477  0.0620     6      11 -0.00640  0.0368 -0.0
345  0.0209
##  2    8765 Colo…  0.085   0.0184     2      10  0.110    0.0391  0.0
875  0.0123
##  3    8768 Conn…  0.118   0.0210     3       7  0.178    0.0383  0.1
24   0.0116
##  4    8778 Flor… -0.0145  0.0540     6      30  0.00310  0.0317 -0.0
0876 0.0181
##  5    8781 Geor… -0.0503  0.0403     8      16 -0.0147   0.0362 -0.0
455  0.0133
##  6    8788 Iowa  -0.114   0.0219     6       6 -0.0571   0.0303 -0.1
10   0.00857
##  7    8794 Illi…  0.112   0.0195     3      19  0.140    0.0310  0.1
15   0.0106
##  8    8800 Kans… -0.135   0.0240     3       6 -0.120    0.0347 -0.1
33   0.0129
##  9    8811 Mass…  0.251   0.0442     3      11  0.318    0.0312  0.2
78   0.0198
## 10    8820 Mich… -0.0275  0.0672     4      15  0.0239   0.0385 -0.0
0525 0.0253
## # … with 13 more rows, 1 more variable: Biden_win_prob <dbl>, and ab
breviated
## #   variable names ¹standard_deviation, ²electoral_votes, ³prior_ave
rage,
## #   ⁴prior_sd, ⁵posterior_mean, ⁶posterior_sd
```

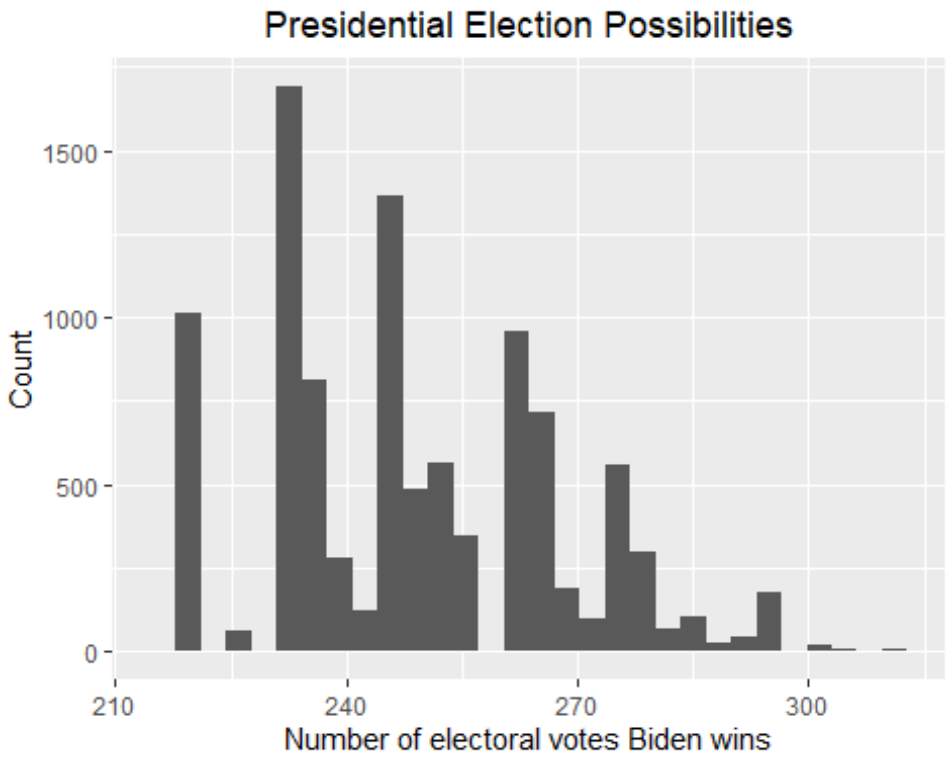
Print out the table of states with Biden's probabilities of winning. If a probability is larger than 99%, show >99%, and if smaller than 1% then show <1%.

```
temp <- updated_results_no_bias |> select(state, Biden_win_prob) |> mut
ate(Biden_win = paste(as.character(round(100 * Biden_win_prob, 2)), "%
")) |> arrange(-Biden_win_prob)
temp["Biden_win"][temp["Biden_win_prob"] > 0.99] <- "> 99 %"
temp["Biden_win"][temp["Biden_win_prob"] < 0.01] <- "< 1 %"
temp |> select(state, Biden_win)
```

```
## # A tibble: 23 × 3
## # Groups:   race_id [23]
##    race_id state         Biden_win
##      <dbl> <chr>         <chr>
##  1    8768 Connecticut   > 99 %
##  2    8794 Illinois      > 99 %
##  3    8811 Massachusetts > 99 %
##  4    8860 New York      > 99 %
##  5    8869 Oregon        > 99 %
##  6    8877 Rhode Island  > 99 %
##  7    8765 Colorado      > 99 %
##  8    8854 New Mexico    > 99 %
##  9    8848 New Hampshire > 99 %
## 10    8905 Wisconsin     64.12 %
## # … with 13 more rows
```

Create a Monte Carlo simulation using the updated posterior average and standard deviation.

```
# We now start the Monte Carlo simulation based on the above result
set.seed(1)
B <- 10000
simulation <- function(){
  temp <- c()
  for (i in 1:length(updated_results_no_bias$state)){
    temp <- c(sample(c(1, 0), size = 1, replace = TRUE, prob = c(pnorm
(0, mean = updated_results_no_bias$posterior_mean[i], sd = updated_resu
lts_no_bias$posterior_sd[i], lower.tail = FALSE), pnorm(0, mean = updat
ed_results_no_bias$posterior_mean[i], sd = updated_results_no_bias$post
erior_sd[i], lower.tail = TRUE))) * updated_results_no_bias$electoral_v
otes[i], temp)
  }
  sum(temp) + Biden_win_electoral_votes
}
result_mc <- data.frame(electoral_votes_count = replicate(B, simulation
()))
result_mc |> ggplot(aes(x = electoral_votes_count)) + geom_histogram()
+ xlab("Number of electoral votes Biden wins") + ylab("Count") + ggtitl
e("Presidential Election Possibilities") + theme(plot.title = element_t
ext(hjust = 0.5))
```



Let's see now how likely Biden will win against Trump with the new posterior.

```
print(paste("According to the above plot, the probability that Biden wi
ns the 2024 presidential election is ", round(mean(result_mc >= 270) *
100, 2), "%"))
```

```
## [1] "According to the above plot, the probability that Biden wins th
e 2024 presidential election is  15.24 %"
```

Give the 80% credible interval of the 2024 presidential election.

```
# I use the credible_interval function in the ArchaeoPhases package to
get the 80% credible interval
result <- credible_interval(data = result_mc |> pull(electoral_votes_co
unt), level = 0.8, round_to = 0)

print(paste("According to the redible_interval function in the ArchaeoP
hases package, the 80% credible interval should be [", result$ci[1], ",
 ", result$ci[2], "]"))
```

```
## [1] "According to the redible_interval function in the ArchaeoPhases
 package, the 80% credible interval should be [ 221 ,  265 ]"
```

Now including a bias with average `bias_avg` and standard error of `bias_sd` to add the uncertainty to the election. Print out the table of states with Biden's probabilities of winning.

```r
# According to section 16.8.4 in the book, I set bias_sd to be 0.03. I
also set bias_avg to be 0.00 since I do not think the bias will go towa
rds either party
bias_avg <- 0.00
bias_sd <- 0.03
updated_results_bias <- updated_results |> mutate(sigma = sqrt(standard
_deviation^2/count  + bias_sd^2),
                            B = sigma^2 / (sigma^2 + prior_sd^2),
                            posterior_mean = B * prior_average + (1 -
B) * average,
                            posterior_sd = sqrt(1/ (1/sigma^2 + 1/prio
r_sd^2)),
                            Biden_win_prob = 1-pnorm(0, posterior_mean,
 posterior_sd)) |> select(-sigma, -B)


updated_results_bias
```

```
## # A tibble: 23 × 11
## # Groups:   race_id [23]
##    race_id state average stand…¹ count elect…² prior_…³ prior…⁴ post
er…⁵ poste…⁶
##      <dbl> <chr>   <dbl>   <dbl> <int>   <dbl>    <dbl>   <dbl>    <
dbl>   <dbl>
##  1    8759 Ariz… -0.0477  0.0620     6      11 -0.00640  0.0368 -0.0
257   0.0268
##  2    8765 Colo…  0.085   0.0184     2      10  0.110    0.0391  0.0
951   0.0251
##  3    8768 Conn…  0.118   0.0210     3       7  0.178    0.0383  0.1
43    0.0247
##  4    8778 Flor… -0.0145  0.0540     6      30  0.00310  0.0317 -0.0
0429  0.0241
##  5    8781 Geor… -0.0503  0.0403     8      16 -0.0147   0.0362 -0.0
340   0.0245
##  6    8788 Iowa  -0.114   0.0219     6       6 -0.0571   0.0303 -0.0
848   0.0218
##  7    8794 Illi…  0.112   0.0195     3      19  0.140    0.0310  0.1
27    0.0223
##  8    8800 Kans… -0.135   0.0240     3       6 -0.120    0.0347 -0.1
28    0.0239
##  9    8811 Mass…  0.251   0.0442     3      11  0.318    0.0312  0.2
92    0.0245
## 10    8820 Mich… -0.0275  0.0672     4      15  0.0239   0.0385  0.0
0223  0.0292
## # … with 13 more rows, 1 more variable: Biden_win_prob <dbl>, and ab
breviated
## #   variable names ¹standard_deviation, ²electoral_votes, ³prior_ave
rage,
## #   ⁴prior_sd, ⁵posterior_mean, ⁶posterior_sd
```

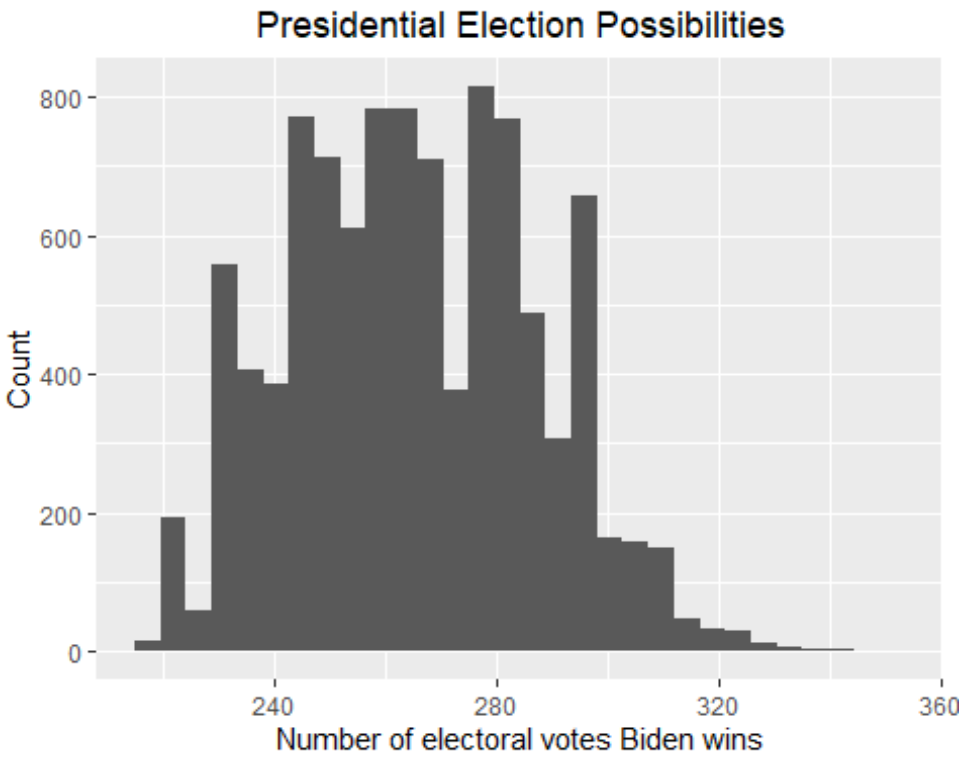
Print out the table of states with Biden's probabilities of winning. If a probability is larger than 99%, show >99%, and if smaller than 1% then show <1%.

```
# Print out the table of states with Biden's probabilities of winning
temp <- updated_results_bias |> select(state, Biden_win_prob) |> mutate
(Biden_win = paste(as.character(round(100 * Biden_win_prob, 2)), "%"))
temp["Biden_win"][temp["Biden_win_prob"] > 0.99] <- "> 99 %"
temp["Biden_win"][temp["Biden_win_prob"] < 0.01] <- "< 1 %"
temp |> select(state, Biden_win)
```

```
## # A tibble: 23 × 3
## # Groups:   race_id [23]
##    race_id state         Biden_win
##      <dbl> <chr>         <chr>
##  1    8759 Arizona       16.89 %
##  2    8765 Colorado      > 99 %
##  3    8768 Connecticut   > 99 %
##  4    8778 Florida       42.94 %
##  5    8781 Georgia       8.21 %
##  6    8788 Iowa          < 1 %
##  7    8794 Illinois      > 99 %
##  8    8800 Kansas        < 1 %
##  9    8811 Massachusetts > 99 %
## 10    8820 Michigan      53.04 %
## # … with 13 more rows
```

Create a Monte Carlo simulation using the updated posterior average and standard deviation.

```
# We now start the Monte Carlo simulation based on the above result
set.seed(1)
B <- 10000
simulation <- function(){
  temp <- c()
  for (i in 1:length(updated_results_bias$state)){
    temp <- c(sample(c(1, 0), size = 1, replace = TRUE, prob = c(pnorm
(0, mean = updated_results_bias$posterior_mean[i], sd = updated_results
_bias$posterior_sd[i], lower.tail = FALSE), pnorm(0, mean = updated_res
ults_bias$posterior_mean[i], sd = updated_results_bias$posterior_sd[i],
 lower.tail = TRUE))) * updated_results_bias$electoral_votes[i], temp)
  }
  sum(temp) + Biden_win_electoral_votes
}
result_mc <- data.frame(electoral_votes_count = replicate(B, simulation
()))
result_mc |> ggplot(aes(x = electoral_votes_count)) + geom_histogram()
+ xlab("Number of electoral votes Biden wins") + ylab("Count") + ggtitl
e("Presidential Election Possibilities") + theme(plot.title = element_t
ext(hjust = 0.5))
```



Let's see now how likely Biden will win against Trump with the new posterior.

```
print(paste("According to the above plot, the probability that Biden wi
ns the 2024 presidential election is ", round(mean(result_mc >= 270) *
100, 2), "%"))
```

```
## [1] "According to the above plot, the probability that Biden wins th
e 2024 presidential election is  41.71 %"
```

Give the 80% credible interval of the 2024 presidential election.

```
# I use the credible_interval function in the ArchaeoPhases package to
get the 80% credible interval
result <- credible_interval(data = result_mc |> pull(electoral_votes_co
unt), level = 0.8, round_to = 0)

print(paste("According to the redible_interval function in the ArchaeoP
hases package, the 80% credible interval should be [", result$ci[1], ",
 ", result$ci[2], "]"))
```

```
## [1] "According to the redible_interval function in the ArchaeoPhases
 package, the 80% credible interval should be [ 231 ,  287 ]"
```

I believe that the result above is more reasonable. It is sometimes the case that a president lose the midterm election but win in the presidential election two years later. Since I think the polls in 2020 favor Biden and the polls in 2022 favor Trump, by mitigating the polls results, we should approach the results better.