Problem set 7

2024-11-04

For this problem set we want you to predict the election. You will enter you predictions to this form. You you will report a prediction of the number of electoral votes for Harris and an interval. You will do the same for the popular vote. We will give prizes for those that report the shortest interval but with the true result inside the interval.

1. Read in the data provided here:

```r
url <- "https://projects.fivethirtyeight.com/polls/data/president_polls.csv"
```

Examine the data frame paying particular attention to the `poll_id` `question_id`, `population`, and `candidate`. Note that some polls have more than one question based on different population types.

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(rvest)
```

```
Attaching package: 'rvest'

The following object is masked from 'package:readr':

    guess_encoding
```

```r
raw_dat <- read_csv(url)
```

```
Rows: 18095 Columns: 52
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (25): pollster, sponsors, display_name, pollster_rating_name, methodolog...
dbl (16): poll_id, pollster_id, pollster_rating_id, numeric_grade, pollscore...
num  (1): sponsor_ids
lgl (10): endorsed_candidate_id, endorsed_candidate_name, endorsed_candidate...

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

```r
raw_dat|>select('poll_id', 'question_id','candidate_name', 'population')
```

```
# A tibble: 18,095 x 4
   poll_id question_id candidate_name population
     <dbl>       <dbl> <chr>          <chr>
 1   89372      216453 Kamala Harris  lv
 2   89372      216453 Donald Trump   lv
 3   89372      216453 Jill Stein     lv
 4   89372      216453 Cornel West    lv
 5   89372      216453 Chase Oliver   lv
 6   89372      216454 Kamala Harris  lv
 7   89372      216454 Donald Trump   lv
 8   89373      216464 Kamala Harris  lv
 9   89373      216464 Donald Trump   lv
10   89373      216464 Jill Stein     lv
# i 18,085 more rows
```

2. Polls are based on either likely voters (lv), registered voters (rv), all voters (a), or voters (v). Polls based on 'voters' are exit polls. We want to remove these because exit polls are too old or might be biased due to differences in the likelihood of early voter by party. We prefer likely voter (lv) polls because they are more predictive. Registered voter polls

are more predictive than all voter (a) polls. Remove the exit poll (v) polls and then redefine `population` to be a factor ordered from best to worse predictive power: (lv, rv, a). You should also remove hypothetical polls and make the date columns into date objects. Name the resulting data frame `dat`.

```
dat <- raw_dat |> filter(!hypothetical & population != 'v')|>
  mutate(population = factor(population, levels = c('lv','rv','a')))|>
  mutate(start_date = mdy(start_date),
         end_date = mdy(end_date))
dat
```

```
# A tibble: 5,611 x 52
   poll_id pollster_id pollster   sponsor_ids sponsors display_name
     <dbl>       <dbl> <chr>            <dbl> <chr>    <chr>
 1   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 2   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 3   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 4   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 5   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 6   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 7   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 8   89373        1528 AtlasIntel          NA <NA>     AtlasIntel
 9   89373        1528 AtlasIntel          NA <NA>     AtlasIntel
10   89373        1528 AtlasIntel          NA <NA>     AtlasIntel
# i 5,601 more rows
# i 46 more variables: pollster_rating_id <dbl>, pollster_rating_name <chr>,
#   numeric_grade <dbl>, pollscore <dbl>, methodology <chr>,
#   transparency_score <dbl>, state <chr>, start_date <date>, end_date <date>,
#   sponsor_candidate_id <dbl>, sponsor_candidate <chr>,
#   sponsor_candidate_party <chr>, endorsed_candidate_id <lgl>,
#   endorsed_candidate_name <lgl>, endorsed_candidate_party <lgl>, ...
```

3. Some polls asked more than one questions. So if you filter to one poll ID in our dataset, you might see more than one question ID associated with the same poll. The most common reason for this is that they asked a head-to-head question (Harris versus Trump) and, in the same poll, a question about all candidates. We want to prioritize the head-to-head questions.

Add a column that tells us, for each question, how many candidates where mentioned in that question.

Add a new column `n` to `dat` that provides the number of candidates mentioned for each question. For example the relevant column of your final table will looks something like this:

poll_id	question_id	candidate	n
1	1	Harris	2
1	1	Trump	2
1	2	Harris	3
1	2	Trump	3
1	2	Stein	3

```
dat <- dat |>
  group_by(question_id) |>
  mutate(n=n()) |>
  ungroup()
dat
```

```
# A tibble: 5,611 x 53
   poll_id pollster_id pollster   sponsor_ids sponsors display_name
     <dbl>       <dbl> <chr>            <dbl> <chr>    <chr>
 1   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 2   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 3   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 4   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 5   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 6   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 7   89372        1528 AtlasIntel          NA <NA>     AtlasIntel
 8   89373        1528 AtlasIntel          NA <NA>     AtlasIntel
 9   89373        1528 AtlasIntel          NA <NA>     AtlasIntel
10   89373        1528 AtlasIntel          NA <NA>     AtlasIntel
# i 5,601 more rows
# i 47 more variables: pollster_rating_id <dbl>, pollster_rating_name <chr>,
#   numeric_grade <dbl>, pollscore <dbl>, methodology <chr>,
#   transparency_score <dbl>, state <chr>, start_date <date>, end_date <date>,
#   sponsor_candidate_id <dbl>, sponsor_candidate <chr>,
#   sponsor_candidate_party <chr>, endorsed_candidate_id <lgl>,
#   endorsed_candidate_name <lgl>, endorsed_candidate_party <lgl>, ...
```

4. We are going to focus on the Harris versus Trump comparison. Redefine `dat` to only include the rows providing information for Harris and Trump. Then pivot the dataset so that the percentages for Harris and Trump are in their own columns. Note that for pivot to work you will have to remove some columns. To avoid this keep only the columns you are pivoting and along with `poll_id`, `question_id`, `state`, `pollster`, `start_date`, `end_date`, `numeric_grade`, `sample_size`. Once you accomplish the pivot, add a column called `spread` with the difference between Harris and Trump.

Note that the values stored in `spread` are estimates of the popular vote difference that we will use to predict for the competition:

spread = % of the popular vote for Harris - % of the popular vote for Trump

However, for the calculations in the rest of problem set to be consistent with the sampling model we have been discussing in class, save `spread` as a proportion, not a percentage. But remember to turn it back to a percentage when submitting your entry to the competition.

```
dat <- dat |>
  filter(candidate_name %in% c('Kamala Harris','Donald Trump'))|>
  select(poll_id, question_id, state, pollster, start_date, end_date, numeric_grade,
         sample_size, pct, candidate_name, n, population)|>
  mutate(candidate_name=str_remove(candidate_name,'\\w+\\s'))|>
  pivot_wider(names_from = candidate_name, values_from = pct)|>
  mutate(spread=(Harris-Trump)/100)
dat
```

```
# A tibble: 2,189 x 13
   poll_id question_id state    pollster   start_date end_date   numeric_grade
     <dbl>       <dbl> <chr>    <chr>      <date>     <date>             <dbl>
 1   89372      216453 <NA>     AtlasIntel 2024-11-03 2024-11-04           2.7
 2   89372      216454 <NA>     AtlasIntel 2024-11-03 2024-11-04           2.7
 3   89373      216464 Arizona  AtlasIntel 2024-11-03 2024-11-04           2.7
 4   89373      216465 Arizona  AtlasIntel 2024-11-03 2024-11-04           2.7
 5   89374      216466 Georgia  AtlasIntel 2024-11-03 2024-11-04           2.7
 6   89374      216467 Georgia  AtlasIntel 2024-11-03 2024-11-04           2.7
 7   89375      216468 Michigan AtlasIntel 2024-11-03 2024-11-04           2.7
 8   89375      216469 Michigan AtlasIntel 2024-11-03 2024-11-04           2.7
 9   89378      216474 Nevada   AtlasIntel 2024-11-03 2024-11-04           2.7
10   89378      216475 Nevada   AtlasIntel 2024-11-03 2024-11-04           2.7
# i 2,179 more rows
# i 6 more variables: sample_size <dbl>, n <int>, population <fct>,
#   Harris <dbl>, Trump <dbl>, spread <dbl>
```

5. Note that some polls have multiple questions. We want to keep only one question per poll. We will keep likely voter (lv) polls when available, and prefer register voter (rv) over all voter polls (a). If more than one question was asked in one poll, take the most targeted question (smallest `n`). Save the resulting table`dat`. Note that now each after you do this each row will represents exactly one poll/question, so can remove `n`, `poll_id` and `question_id`.

```
dat <- dat |>
  arrange(population, n)|>
  group_by(poll_id)|>
  slice(1)|>
  ungroup()|>
  select(-n,-poll_id,-question_id)
dat
```

```
# A tibble: 1,516 x 10
   state   pollster   start_date end_date   numeric_grade sample_size population
   <chr>   <chr>      <date>     <date>             <dbl>       <dbl> <fct>
 1 <NA>    McLaughlin 2021-05-12 2021-05-18           0.5        1000 lv
 2 <NA>    Echelon I~ 2021-06-18 2021-06-22           2.7        1001 rv
 3 <NA>    McLaughlin 2021-06-16 2021-06-20           0.5        1000 lv
 4 <NA>    McLaughlin 2021-07-29 2021-08-03           0.5        1000 lv
 5 <NA>    McLaughlin 2021-09-09 2021-09-14           0.5        1000 lv
 6 <NA>    Rasmussen  2021-09-21 2021-09-22           2.1        1000 lv
 7 Florida Victory I~ 2021-09-16 2021-09-18           1.3         450 lv
 8 <NA>    McLaughlin 2021-10-14 2021-10-18           0.5        1000 lv
 9 <NA>    Redfield ~ 2021-11-15 2021-11-15           1.8        1500 rv
10 <NA>    McLaughlin 2021-11-11 2021-11-16           0.5        1000 lv
# i 1,506 more rows
# i 3 more variables: Harris <dbl>, Trump <dbl>, spread <dbl>
```

6. Separate `dat` into two data frames: one with popular vote polls and one with state level polls. Call them `popular_vote` and `polls` respectively.

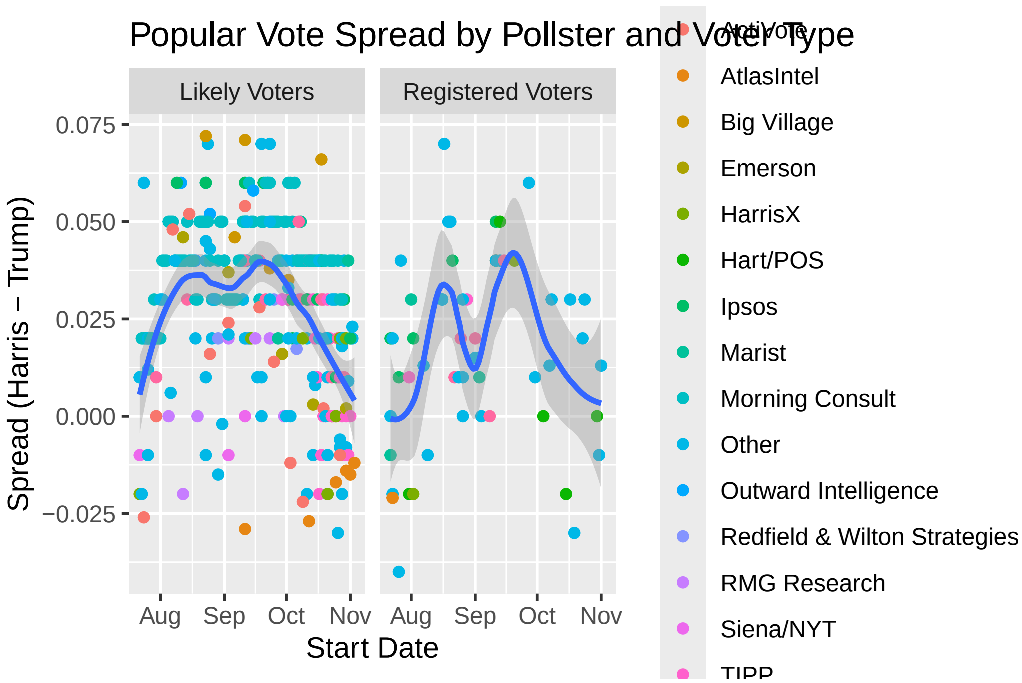
```
popular_vote <- filter(dat, is.na(state))
polls <- filter(dat,!is.na(state))
```

7. For the popular vote, plot the spread reported by each poll against start date for polls starting after July 21, 2024. Rename all the pollsters with less than 5 polls during this period as `Other`. Use color to denote pollster. Make separate plots for likely voters and registered voters. Do not use *all voter* polls (a). Use `geom_smooth` with method `loess` to show a curve going through the points. You can change how adaptive the curve is to that through the `span` argument.

```
popular_vote |>
  filter(start_date > make_date(2024, 7, 21) & population != "a") |>
  group_by(pollster)|>
  mutate(poll_count = n())|>
  ungroup()|>
```

```
  mutate(pollster=ifelse(poll_count<5,'Other',pollster ))|>
  select(-poll_count)|>
  ggplot(aes(x=start_date,y=spread))+
  geom_point(aes(color=pollster))+
  geom_smooth(method='loess',span=0.5)+
  facet_wrap(~population,labeller = labeller(population = c(lv = "Likely Voters",
                                                            rv = "Registered Voters"))) +
  labs(title = "Popular Vote Spread by Pollster and Voter Type",
       x = "Start Date", y = "Spread (Harris - Trump)")
```

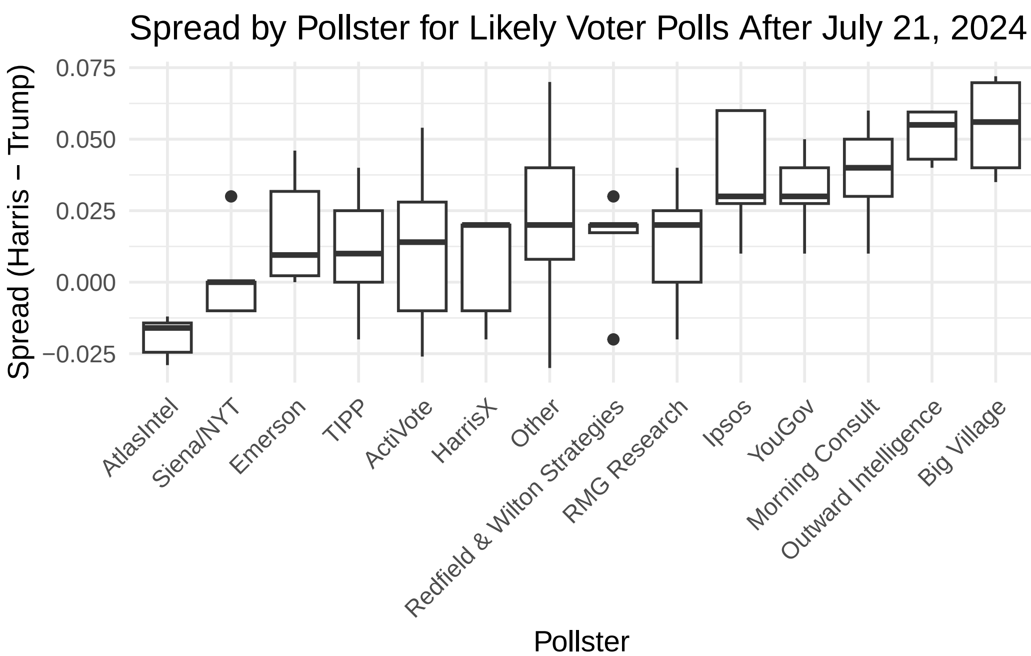
```
`geom_smooth()` using formula = 'y ~ x'
```



8. To show the pollster effect, make boxplots for the the spread for each popular vote poll. Include only likely voter polls starting after July 21, 2024. Rename all the pollsters with less than 5 polls during that time period as `Other`.

```
popular_vote |>
  filter(start_date > make_date(2024, 7, 21) & population == "lv") |>
  group_by(pollster) |>
  mutate(poll_count = n()) |>
  ungroup() |>
```

```
  mutate(pollster = ifelse(poll_count < 5, "Other", pollster)) |>
  mutate(pollster=reorder(pollster,spread, median))|>
  select(-poll_count) |>
  arrange(spread)|>
  ggplot(aes(x = pollster, y = spread)) +
  geom_boxplot() +
  labs(title = "Spread by Pollster for Likely Voter Polls After July 21, 2024",
       x = "Pollster", y = "Spread (Harris - Trump)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



9. Compute a prediction and an interval for the competition and submit here Include the code you used to create your confidence interval for the popular vote here:

```
tmp <- popular_vote|>
  filter(start_date >= make_date(2024, 10, 04) & population == 'lv' &
           !is.na(numeric_grade))|>
  group_by(pollster)|>
  summarize(spread = mean (spread))|>
  ungroup()|>
  summarize(avg =mean(spread), sd = sd(spread),  n= n())
tmp
```

```
# A tibble: 1 x 3
     avg     sd     n
   <dbl>  <dbl> <int>
1 0.0151 0.0216    29
```

```r
cat("95% Prediction Interval:", tmp$avg + c(-1,1)*1.96*tmp$sd / sqrt(tmp$n), "\n")
```

```
95% Prediction Interval: 0.007237021 0.02294744
```

I choose October 1, 2024 as the start date as it gives approximately a month of polling data, which should be recent enough to capture the current status. I will keep only likely voter (lv) polls as Likely voter polls focus on respondents who are more likely to actually cast a ballot on Election Day. I also filter NA quality grade (numeric_grade) to make sure that only polls with a defined quality grade are considered. For each pollster, I calculates the average spread across their polls, which helps to reduce bias if a particular pollster conducted multiple polls. Then I calculated the standard deviation of the spread across pollsters and n, the number of pollsters included in the calculation. With this statistics, I calculate the 95% prediction interval for the average spread. The estimated 95% confidence interval for the average spread, ranging from 0.7% to 2.3%

We now move on to predicting the electoral votes.

10. To obtain the number of electoral votes for each state we will visit this website:

```r
url <- "https://state.1keydata.com/state-electoral-votes.php"
```

We can use the **rvest** package to download and extract the relevant table:

```r
library(rvest)
h <- read_html(url) |>
  html_table()

ev <- h[[4]]
```

Wrangle the data in `ev` to only have two columns `state` and `electoral_votes`. Make sure the electoral vote column is numeric. Add the electoral votes for Maine CD-1 (1), Maine CD-2 (1), Nebraska CD-2 (1), and District of Columbia (3) by hand.

```
ev <- ev |>
  mutate(state = ev$X2, electoral_votes = ev$X3) |>
  select(state, electoral_votes) |>
  slice(-1) |>
  mutate(electoral_votes = as.numeric(electoral_votes),
    electoral_votes = case_when(
      state == "Maine" ~ electoral_votes - 2,
      state == "Nebraska" ~ electoral_votes - 1,
      TRUE ~ electoral_votes
 ))

new_rows <- data.frame(
  state = c("Maine CD-1", "Maine CD-2", "Nebraska CD-2", "District of Columbia"),
  electoral_votes = c(1, 1, 1, 3)
)

ev <- rbind(ev, new_rows)
ev
```

```
# A tibble: 54 x 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
# i 44 more rows
```

11. The presidential race in some states is a forgone conclusion. Because their is practically no uncertainty in who will win, polls are not taken. We will therefore assume that the party that won in 2020 will win again in 2024 if no polls are being collected for a state.

Download the following sheet:

```
library(gsheet)
sheet_url <- "https://docs.google.com/spreadsheets/d/1D-edaVHTnZNhVU840EPUhz3Cgd7m39Urx7HM8Po
raw_res_2020 <- gsheet2tbl(sheet_url)
```

Tidy the `raw_res_2020` dataset so that you have two columns `state` and `party`, with `D` and `R` in the party column to indicate who won in 2020. Add Maine CD-1 (D), Maine CD-2 (R), Nebraska CD-2 (D), and District of Columbia (D) by hand. Save the result to `res_2020`. Hint use the **janitor** `row_to_names` function.

```
library(janitor)
```

```
Attaching package: 'janitor'


The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```
res_2020 <- raw_res_2020[,c(1,4)] |>
  row_to_names(row_number = 1)|>
  filter(!State %in% c("Nationwide", 'Region', 'Midwest','Northeast',
                       'South','West','Washington D.C.', NA) ) |>
  select(state = State, party = P.S.) |>
  mutate(party = str_sub(party, 1, 1))


extra_entries <- data.frame(
  state = c("Maine CD-1", "Maine CD-2", "Nebraska CD-2", "District of Columbia"),
  party = c("D", "R", "D", "D")
)

res_2020 <- bind_rows(res_2020, extra_entries)
res_2020
```

```
# A tibble: 54 x 2
   state       party
   <chr>       <chr>
 1 Alabama     R
 2 Alaska      R
 3 Arizona     R
 4 Arkansas    R
 5 California  D
 6 Colorado    D
 7 Connecticut D
 8 Delaware    D
```

```
 9 Florida     R
10 Georgia     R
# i 44 more rows
```

12. Decide on a period that you will use to compute your prediction. We will use `spread` as the outcome. Make sure the the outcomes is saved as a proportion not percentage. Create a `results` data frame with columns `state`, `avg`, `sd`, `n` and `electoral_votes`, with one row per state.

Some ideas and recommendations:

- If a state has enough polls, consider a short period, such as a week. For states with few polls you might need to increase the interval to increase the number of polls.
- Decide which polls to prioritize based on the `population` and `numeric_grade` columns.
- You might want to weigh them differently, in which you might also consider using `sample_size`.
- If you use fewer than 5 polls to calculate an average, your estimate of the standard deviation (SD) may be unreliable. With only one poll, you wont be able to estimate the SD at all. In these cases, consider using the SD from similar states to avoid unusual or inaccurate estimates.

```r
short_period_start <- as.Date("2024-10-21")  #two weeks
long_period_start <- as.Date("2024-09-01") #a month

poll_threshold <- 10

results <- polls |>
  filter(start_date >= long_period_start) |>
  group_by(state)|>
  mutate(
    poll_count_short = sum(start_date >= short_period_start),# Count polls in short period
    use_short_period = poll_count_short >= poll_threshold, # Decide if we use short period
  filter_start_date = as.Date(ifelse(use_short_period, short_period_start,
                                     long_period_start))
  )|>
  filter(start_date >= filter_start_date) |>
  ungroup()

results <- results|>
  group_by(state)|>
  summarise(
    avg = sum(spread * sample_size, na.rm = TRUE) / sum(sample_size,
          na.rm = TRUE), # Weighted average based on the `population` and `numeric_grade` co
```

```
    sd = ifelse (n() >= 5, sd(spread, na.rm = TRUE), NA), # Calculate SD if enough polls
    n = n()                                                # Number of polls used
  ) |>
  ungroup()


results <- results %>%
  mutate(
    sd = ifelse(is.na(sd), mean(sd, na.rm = TRUE), sd) # Replace  SD with average SD
  )

results <- results %>%
  left_join(ev, by = "state")

results
```

```
# A tibble: 45 x 5
   state          avg     sd     n electoral_votes
   <chr>        <dbl>  <dbl> <int>           <dbl>
 1 Alaska     -0.0915 0.0285     2               3
 2 Arizona    -0.0240 0.0222    24              11
 3 California  0.265  0.0332     6              54
 4 Colorado    0.128  0.0260     6              10
 5 Delaware    0.17   0.0285     1               3
 6 Florida    -0.0541 0.0268    34              30
 7 Georgia    -0.0139 0.0114    18              16
 8 Illinois    0.172  0.0285     2              19
 9 Indiana    -0.165  0.0285     3              11
10 Iowa       -0.0771 0.0285     4               6
# i 35 more rows
```

13. Note you will not have polls for all states. Assume that lack of polls implies the state is not in play. Use the `res_2020` data frame to compute the electoral votes Harris is practically guaranteed to have.

```
harris_start <- res_2020|>
  filter(party == "D" & !state %in% unique(polls$state)) |>
  left_join(ev, by = "state")|>
  summarise(harris_start = sum(electoral_votes, na.rm = TRUE)) %>%  # Sum electoral votes
  pull(harris_start)
harris_start
```

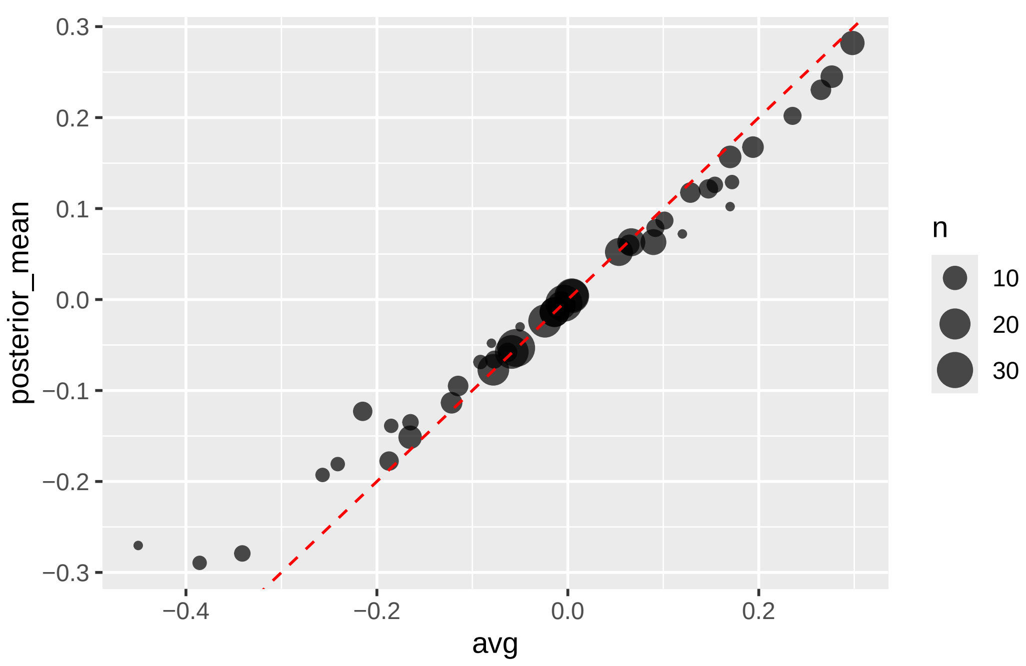
```
[1] 14
```

14. Use a Bayesian approach to compute posterior means and standard deviations for each state in **results**. Plot the posterior mean versus the observed average with the size of the point proportional to the number of polls.

```
theta <- 0
tau <- 0.035
sigma <- results$sd/sqrt(results$n)
x_bar <- results$avg
B <- sigma^2 / (sigma^2 + tau^2)

results <- results %>%
  mutate(
    posterior_mean = B*theta + (1 - B)*x_bar,
    posterior_se = sqrt(1/(1/sigma^2 + 1/tau^2))
  )


ggplot(results, aes(x = avg, y = posterior_mean, size = n)) +
  geom_point(alpha = 0.7) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red")
```



```
  labs(
    x = "Observed Average Spread",
    y = "Posterior Mean Spread",
    title = "Posterior Mean vs Observed Average Spread by State",
    size = "Number of Polls"
  ) +
  theme_minimal()
```

```
NULL
```

Before seeing polling data, we don't think any candidate has the advantage, and a difference of up to 7% either way in 5% significance level is possible, so I set theta <- 0 and tau <- 0.035

15. Compute a prediction and an interval for Harris' electoral votes and submit to the competition here. Include the code you used to create your estimate and interval below.

```
total_votes_harris <- harris_start


theta <- 0
tau <- 0.035
bias_sd <- 0.03
harris_EV <- replicate(1000, {
  results |> mutate(sigma = sqrt(sd^2/n  + bias_sd^2),
                    B = sigma^2/(sigma^2 + tau^2),
                    posterior_mean_2 = B*theta  + (1 - B)*avg,
                    posterior_se_2 = sqrt(1/(1/sigma^2 + 1/tau^2)),
                    result = rnorm(length(posterior_mean_2),
                                   posterior_mean_2, posterior_se_2),
                    harris = ifelse(result > 0, electoral_votes, 0)) |> # Harris wins if poste
    summarize(harris = sum(harris,na.rm=TRUE)+ harris_start) |>
    pull(harris)
})


predicted_votes_harris <- mean(harris_EV)

# Calculate a confidence interval
lower_bound <- quantile(harris_EV, 0.025)
upper_bound <- quantile(harris_EV, 0.975)
```

```r
# Display results
cat("Expected Electoral Votes for Harris:", predicted_votes_harris, "\n")
```

```
Expected Electoral Votes for Harris: 267.648
```

```r
cat("95% Prediction Interval:", lower_bound, "to", upper_bound, "\n")
```

```
95% Prediction Interval: 229 to 316
```

First, I got `total_votes_harris`, which initializes the electoral vote count for Harris with votes she is practically guaranteed to have. The priors are the same as Q14, as how I explained in Q14. Here I add general bias 0.03 as it is very likely there is a general bias that affects most pollsters in the same way, making the observed data correlated. I assume the average of polls favors Democrats by 3% based on historical data. Then I simulate Electoral Votes using Bayesian to simulate the number of electoral votes Harris is likely to receive from competitive states with 1000 times. For each run, I got posterior mean and sd. I generate a predicted spread for each state using the posterior distribution and assign electoral votes to Harris if the simulated spread (result) is positive. For each simulation, I calculate Harris's total electoral votes by adding her guaranteed votes from safe states and predicted wins in competitive states. Then taking the avg of simulation I got predicted_votes_harris and its 95% CI. Harris has an predicted electoral vote around 269, and the 95% confidence interval of electoral votes includes 269, showing much uncertainty that Harris will win.