EVEN MORE conditional relationships

Homework

Prof. Bisbee

Due Date: 2024-02-13

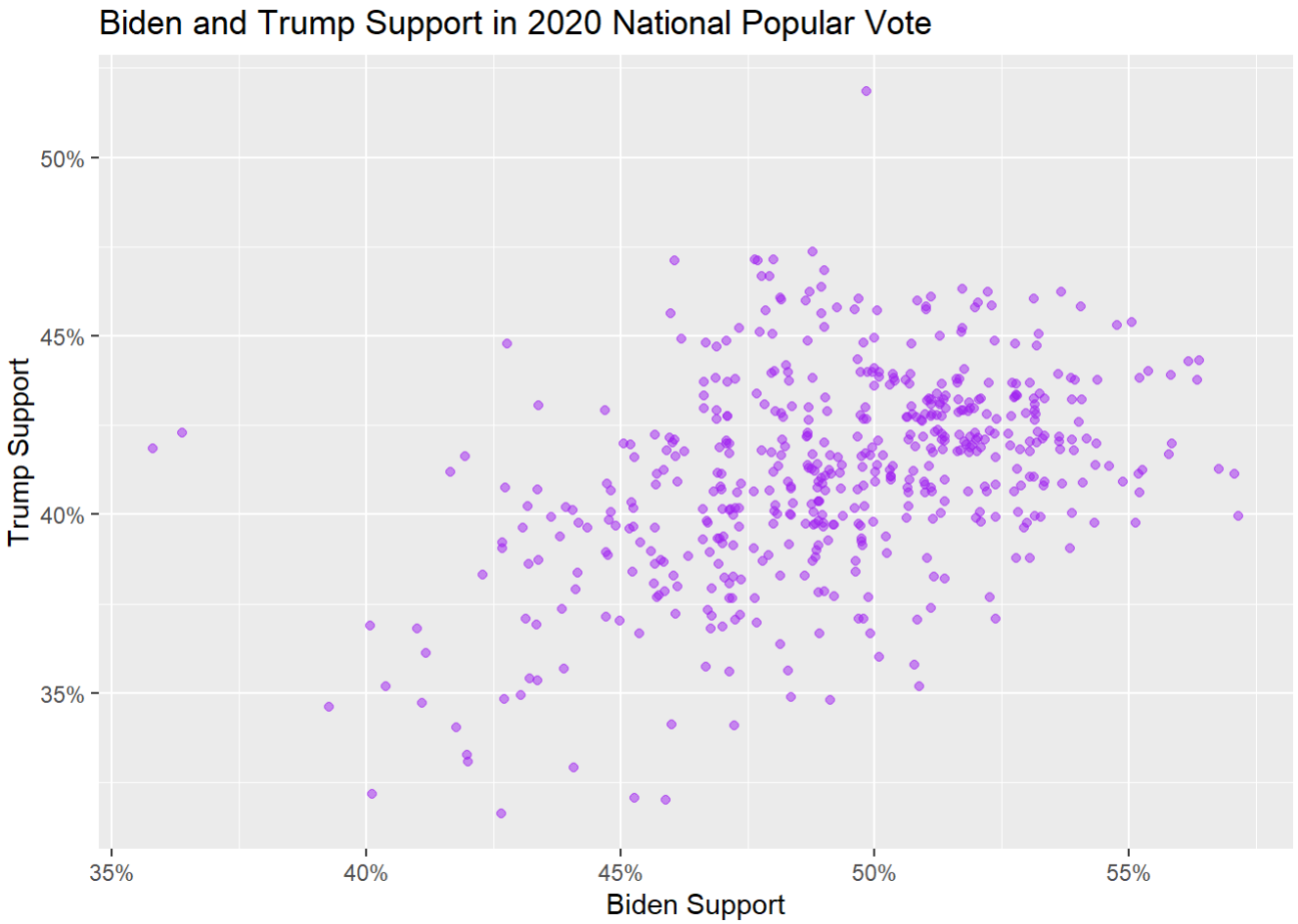
Recap

Recall that we have tested Trump's theory that the MSM was biased against him. We found that polls that underpredicted Trump **also** underpredicted Biden. This is not what we would expect if the polls favored one candidate over another.

Loading the data

```
require(tidyverse)
require(scales)
Pres2020.PV <- read_rds(file="https://github.com/jbisbee1/DS1000_S2024/raw/main/data/Pres2020_PV.Rds")
Pres2020.PV <- Pres2020.PV %>%
                mutate(Trump = Trump/100,
                       Biden = Biden/100,
                       margin = Biden - Trump)
```

```
Pres2020.PV %>%
  ggplot(aes(x = Biden, y = Trump)) +
  labs(title="Biden and Trump Support in 2020 National Popular Vote",
       y = "Trump Support",
       x = "Biden Support") +
  geom_jitter(color="purple",alpha = .5) +
    scale_y_continuous(breaks=seq(0,1,by=.05),
                       labels= scales::percent_format(accuracy = 1)) +
  scale_x_continuous(breaks=seq(0,1,by=.05),
                       labels= scales::percent_format(accuracy = 1))
```



What is an alternative explanation for these patterns? Why would polls underpredict *both* Trump and Biden?

Perhaps they were fielded earlier in the year, when more people were interested in third party candidates, or hadn't made up their mind.

Visualizing More Dimensions – And Introducing Dates!

How did the support for Biden and Trump vary across the course of the 2020 Election?

- What should we measure?
- How do we summarize, visualize, and communicate?

Give you some tools to do some *amazing* things!

Telling Time

- Time is often a critical *descriptive* variable. (Not causal!)
- Also useful for *prediction* ?
- We want to evaluate the properties of presidential polling as Election Day 2020 approached.
- Necessary for prediction – we want most recent data to account for last-minute shift.

- Necessary for identifying when changes occurred (and why?)

Dates in R

- Dates are a special format in R (character with quasi-numeric properties)

```
election.day <- as.Date("11/3/2020", "%m/%d/%Y")
election.day16 <- as.Date("11/8/2016", "%m/%d/%Y")
```

- Difference in “dates” versus difference in integers?

```
election.day - election.day16
```

```
## Time difference of 1456 days
```

```
as.numeric(election.day - election.day16)
```

```
## [1] 1456
```

Initial Questions

- How many polls were publicly done and reported in the media about the national popular vote?
- When did the polling occur? Did most of the polls occur close to Election Day?

So, for every day, how many polls were reported by the media?

Note that we could also look to see if the poll results depend on how the poll is being done (i.e., the `Mode` used to contact respondents) or even depending on who funded (`Funded`) or conducted (`Conducted`) the polls. We could also see if larger polls (i.e., polls with more respondents `SampleSize`) or polls that took longer to conduct (`DaysInField`) were more or less accurate.

Let’s Wrangle…

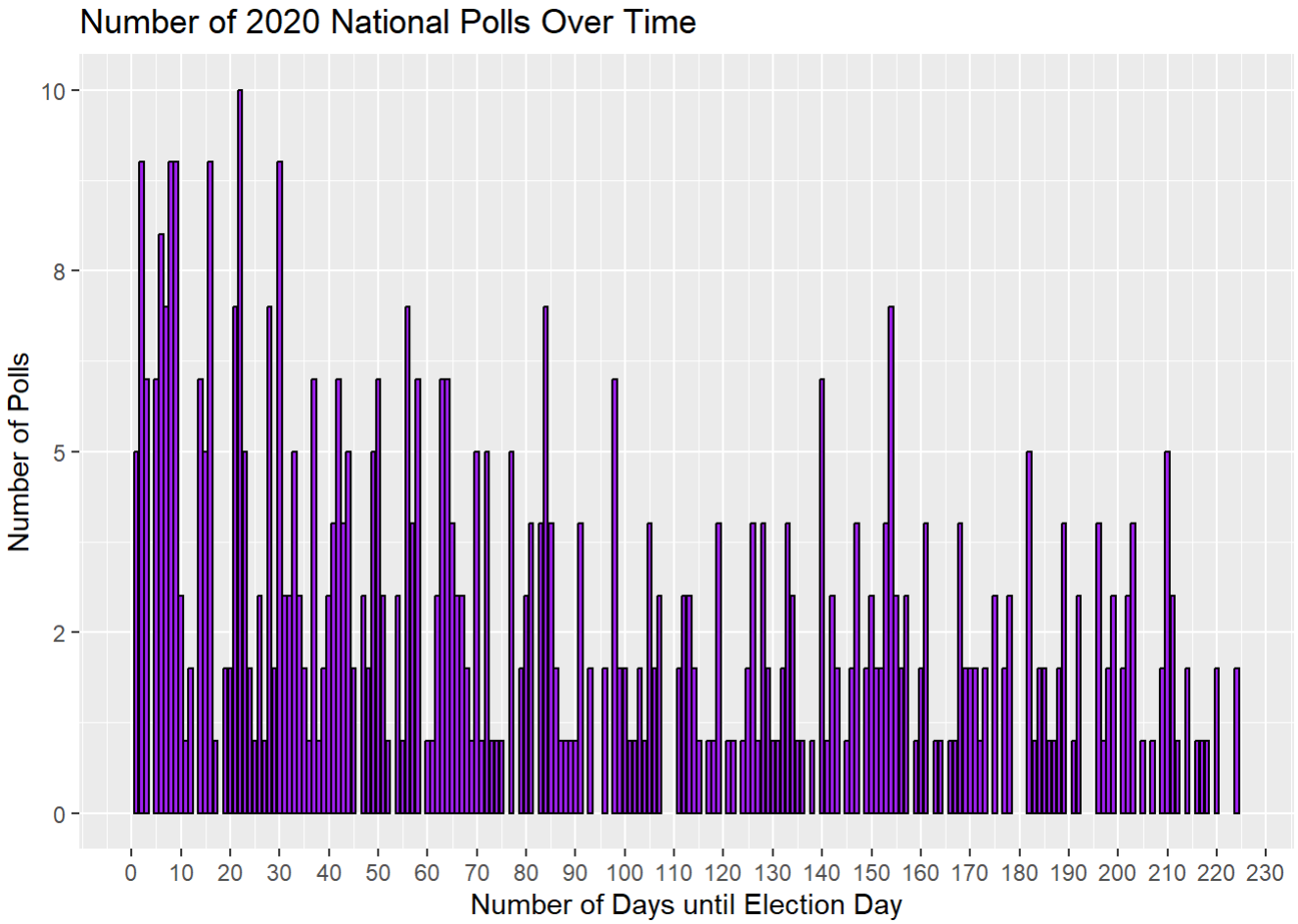
```
Pres2020.PV <- Pres2020.PV %>%
                mutate(EndDate = as.Date(EndDate, "%m/%d/%Y"),
                       StartDate = as.Date(StartDate, "%m/%d/%Y"),
                       DaysToED = as.numeric(election.day - EndDate),
                       Trump = Trump/100,
                       Biden = Biden/100,
                       margin = Biden - Trump)
```

What are we plotting?

- Media Question: how does the number of polls change over time?

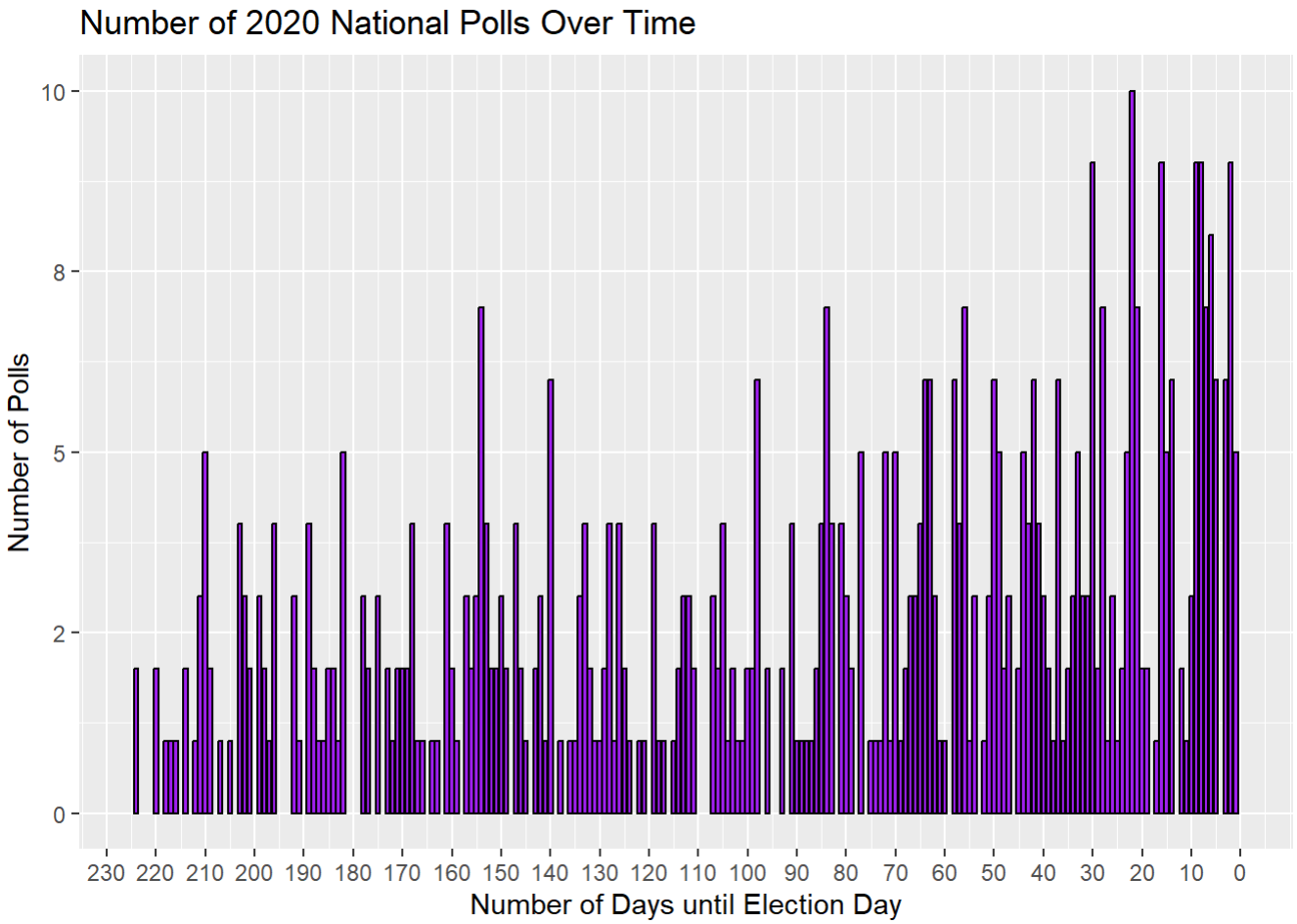
- Data Scientist Question: What do we need to plot? `margin` or `DaysToED` ?
- What will each produce?
- Are they *discrete/categorical* (barplot) or *continuous* (histogram)?

```
ggplot(data = Pres2020.PV, aes(x = DaysToED)) +
  labs(title = "Number of 2020 National Polls Over Time",
       x = "Number of Days until Election Day",
       y = "Number of Polls") +
  geom_bar(fill="purple", color= "black") +
  scale_x_continuous(breaks=seq(0,230,by=10)) +
  scale_y_continuous(labels = label_number(accuracy = 1))
```



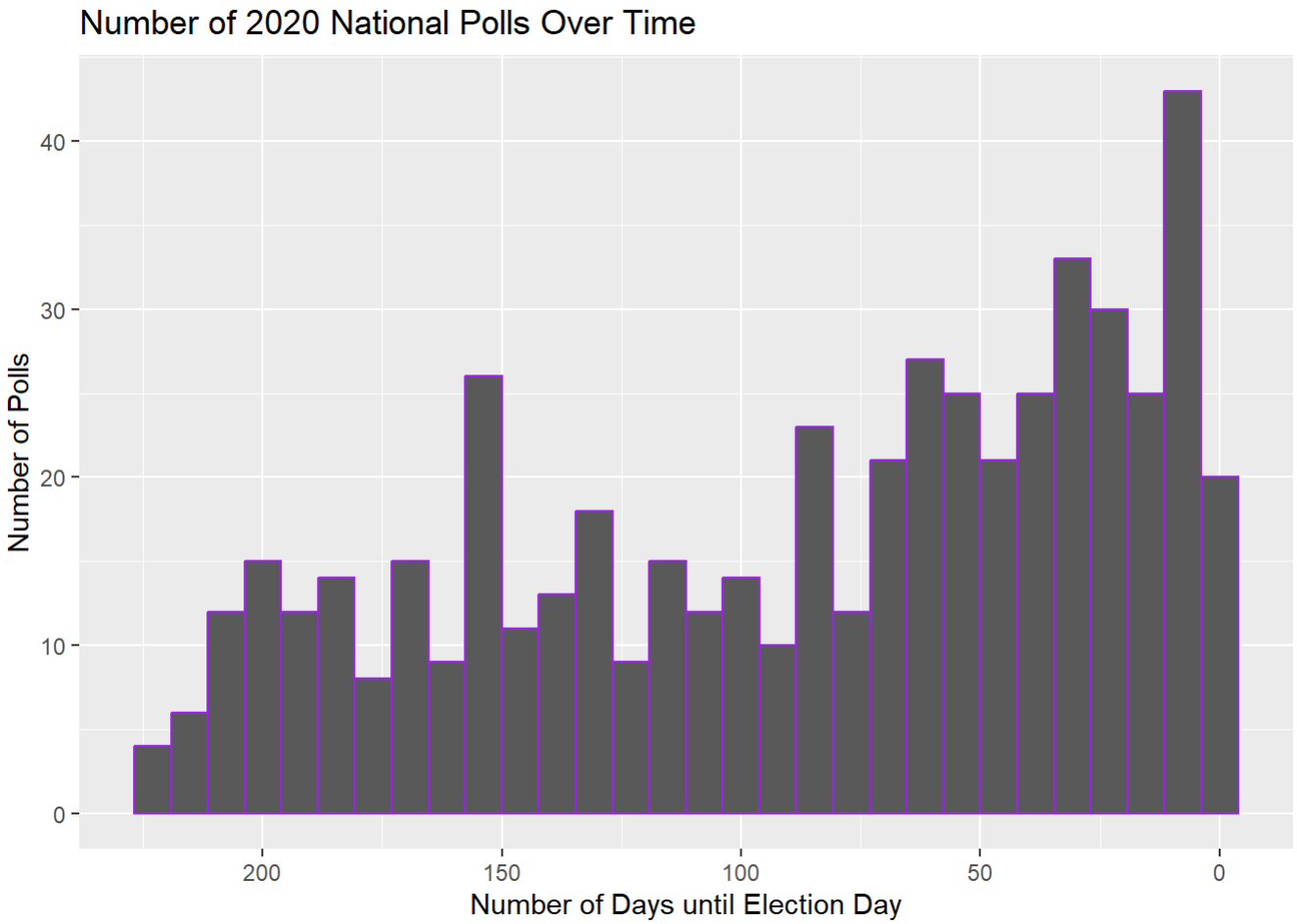
So this is a bit weird because it arranges the axis from smallest to largest even though that is in reverse chronological order. Since November comes after January it may make sense to flip the scale so that the graph plots polls that are closer to Election Day as the reader moves along the scale to the left.

```
ggplot(data = Pres2020.PV, aes(x = DaysToED)) +
  labs(title = "Number of 2020 National Polls Over Time") +
  labs(x = "Number of Days until Election Day") +
  labs(y = "Number of Polls") +
  geom_bar(fill="purple", color= "black") +
  scale_x_reverse(breaks=seq(0,230,by=10)) +
  scale_y_continuous(labels = label_number(accuracy = 1))
```



But is the bargraph the right plot to use? Should we plot every single day given that we know some days might contain fewer polls (e.g., weekends)? What if we to use a histogram instead to plot the number of polls that occur in a set interval of time (to be defined by the number of bins chosen)?

```
ggplot(data = Pres2020.PV, aes(x = DaysToED)) +
  labs(title = "Number of 2020 National Polls Over Time") +
  labs(x = "Number of Days until Election Day") +
  labs(y = "Number of Polls") +
  geom_histogram(color="PURPLE",bins = 30) +
  scale_x_reverse()
```



Which do you prefer? Why or why not? Data Science is sometimes as much art as it is science - especially when it comes to data visualization!

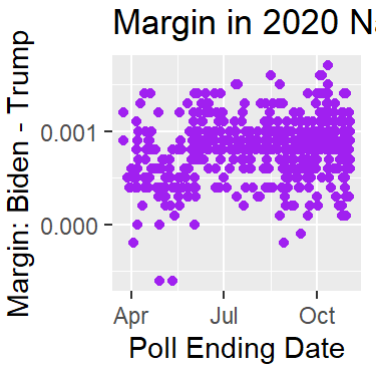
Bivariate/Multivariate relationships

- Most of what we do is a relationship between (at least) 2 variables.
- Here we are interested in how the margin varies as Election Day approaches: `margin` by `DaysToED`.
- Want to plot X (variable that "explains") vs. Y (variable being "explained"):

A very frequently used plot is the scatterplot that shows the relationship between two variables. To do so we are going to define an aesthetic when calling `ggplot` that defines both an x-variable (here `EndDate`) and a y-variable (here `margin`).

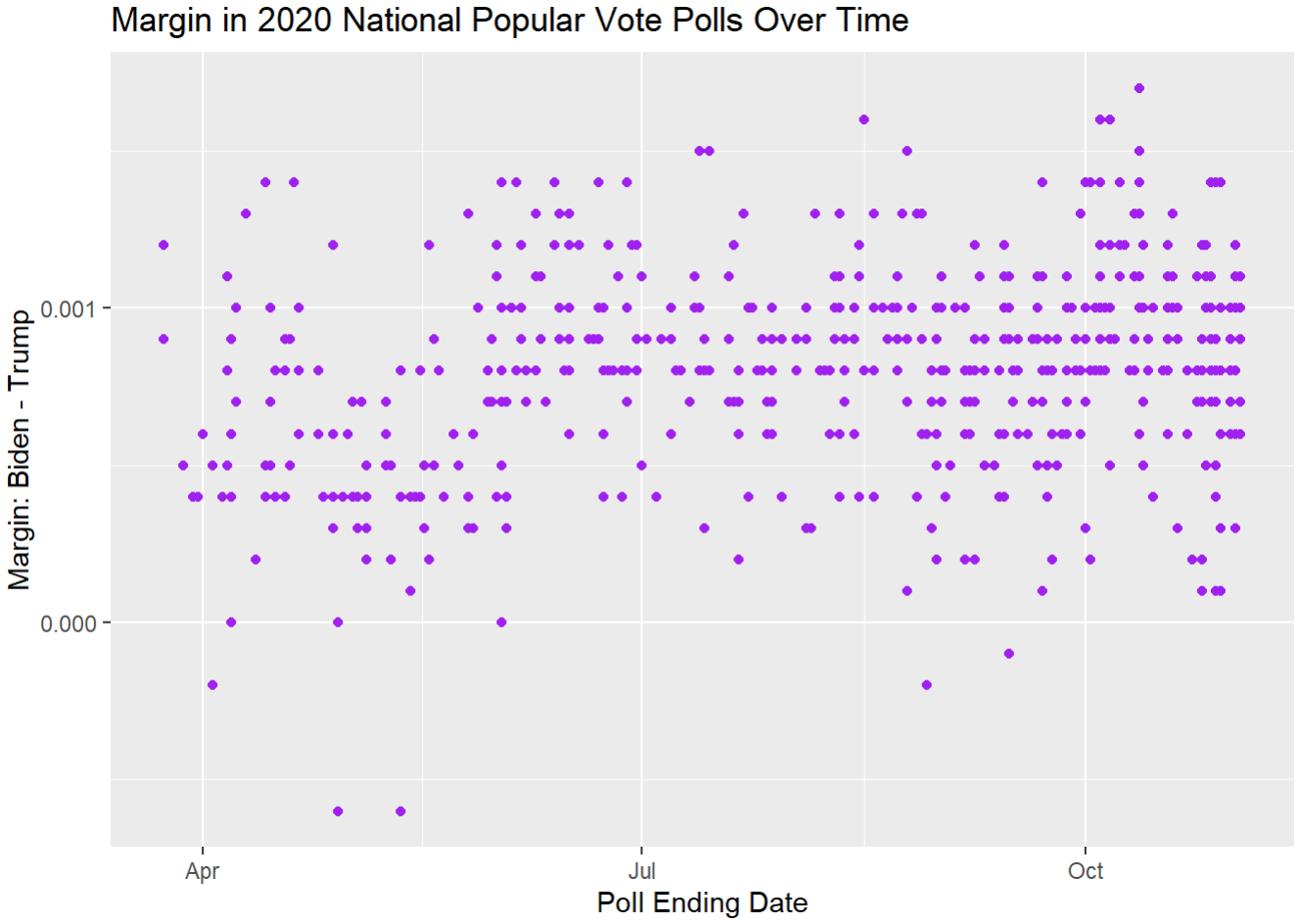
First, a brief aside, we can change the size of the figure being printed in our Rmarkdown by including some commands when defining the R chunk. Much like we could suppress messages (e.g., `message=FALSE` or tell R not to actually evaluate the chunk `eval=FALSE` or to run the code but not print the code `echo=FALSE`) we can add some arguments to this line. For example, the code below defines the figure to be 2 inches tall, 2 inches wide and to be aligned in the center (as opposed to left-justified). As you can see, depending on the choices you make you can destroy the readability of the graphics as `ggplot` will attempt to rescale the figure accordingly. The dimensions are in inches and they refer to the plotting area – an area that includes labels and margins so it is *not* the area where the data itself appears.

```
margin_over_time_plot <- Pres2020.PV %>%
  ggplot(aes(x = EndDate, y = margin)) +
  labs(title="Margin in 2020 National Popular Vote Polls Over Time",
       y = "Margin: Biden - Trump",
       x = "Poll Ending Date") +
  geom_point(color="purple")
margin_over_time_plot
```



To ``fix" this we can call the ggplot object without defining the graphical parameters.

```
margin_over_time_plot
```



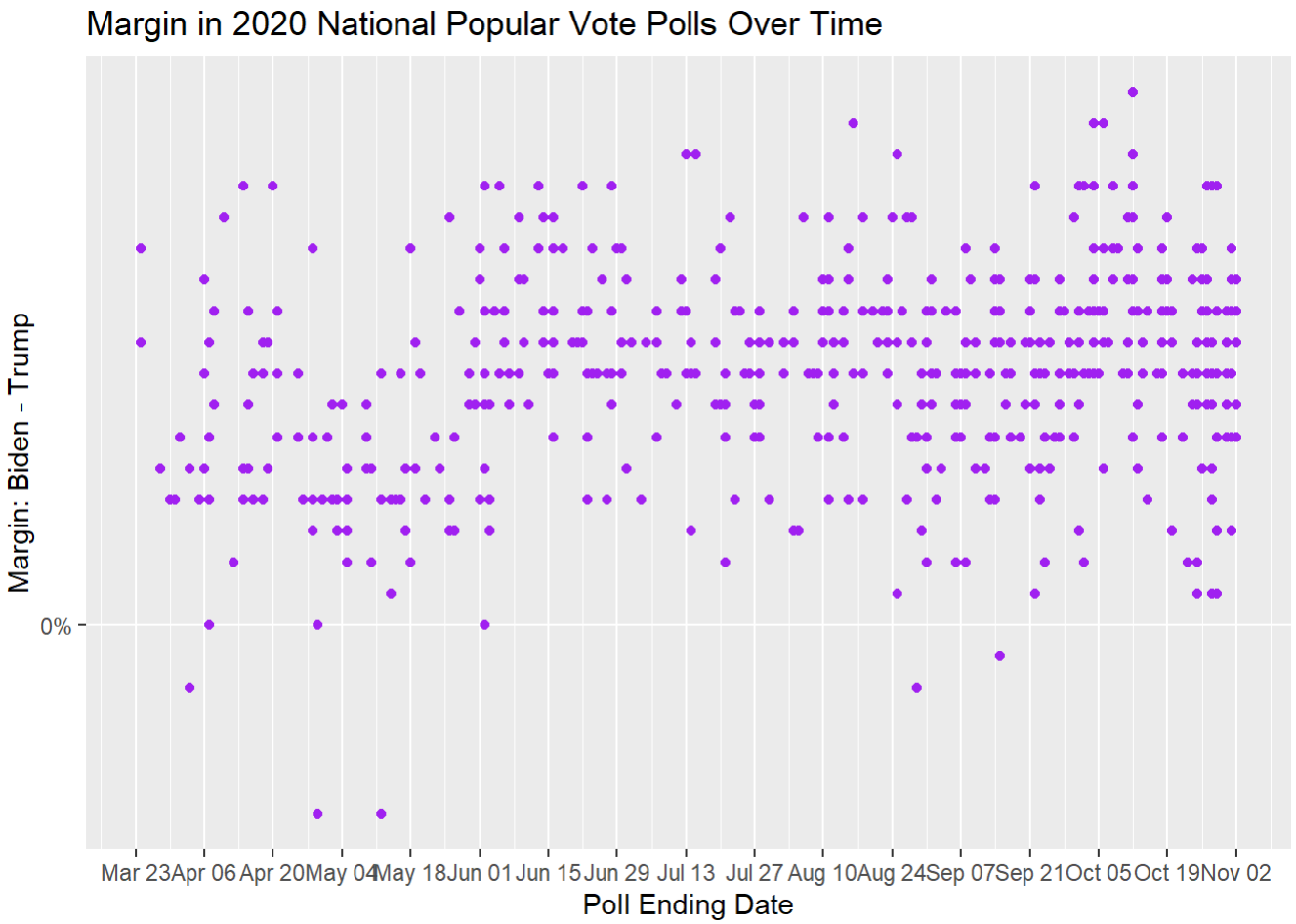
Ok, back to fixing the graph. What do you think?

1. Axes looks weird - lots of interpolation required by the consumer.
2. Data looks “chunky”? How many data points are at each point?

To fix the axis scale we can use `scale_y_continuous` to chose better labels for the y-axis and we can use `scale_x_date` to determine how to plot the dates we are plotting. Here we are going to plot at two-week intervals (`date_breaks = "2 week"`) using labels that include the month and date (`date_labels - "%b %d"`).

Note here that we are going to adjust the plot by adding new features to the ggplot object `margin_over_time_plot` . We could also have created the graph without creating the object.

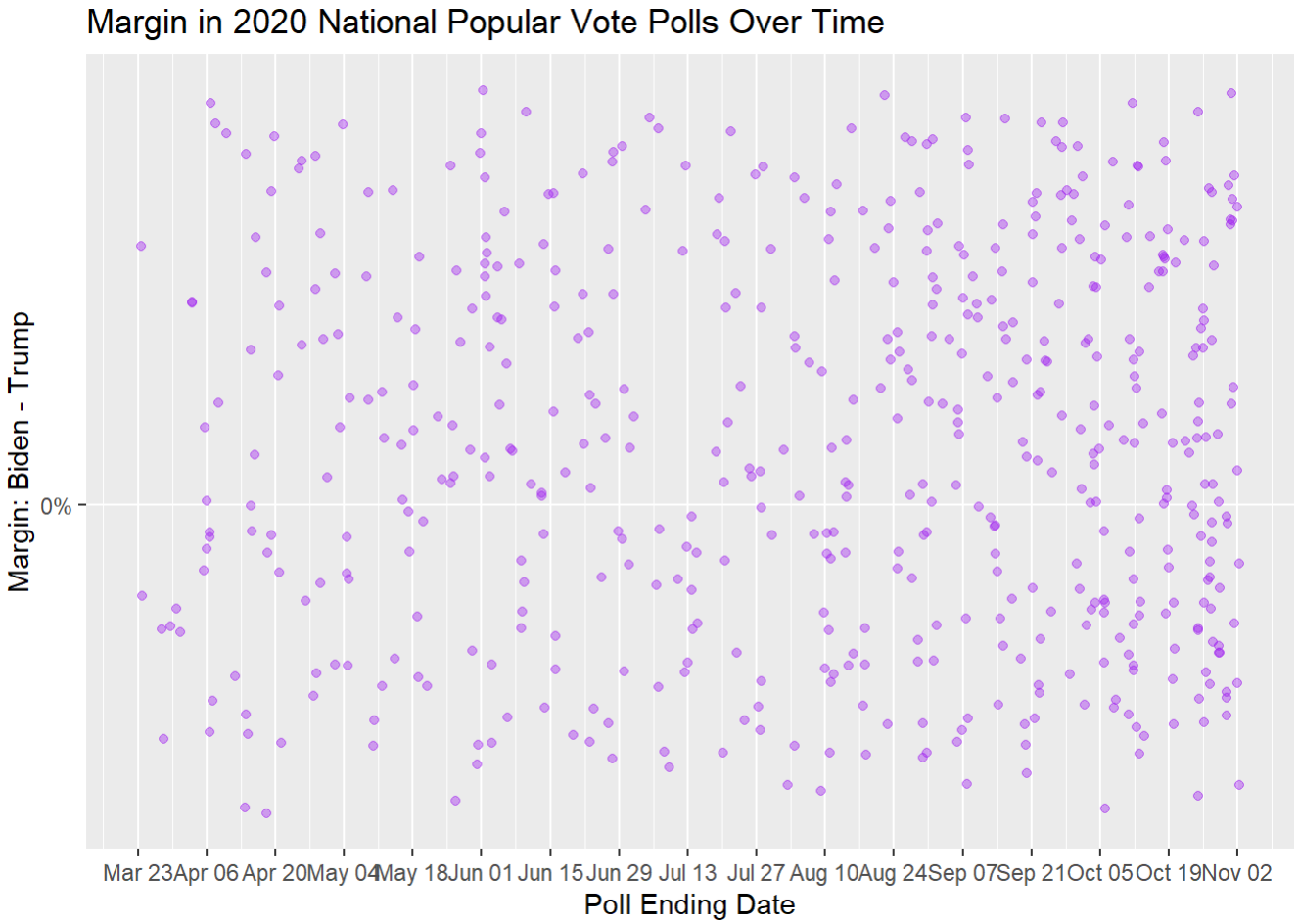
```
margin_over_time_plot <- margin_over_time_plot  +
    scale_y_continuous(breaks=seq(-.1,.2,by=.05),
                       labels= scales::percent_format(accuracy = 1)) +
    scale_x_date(date_breaks = "2 week", date_labels = "%b %d")
margin_over_time_plot
```



Now one thing that is hard to know is how many polls are at a particular point. If some points contain a single poll and others contain 1000 polls that matters a lot for how we interpret the relationship.

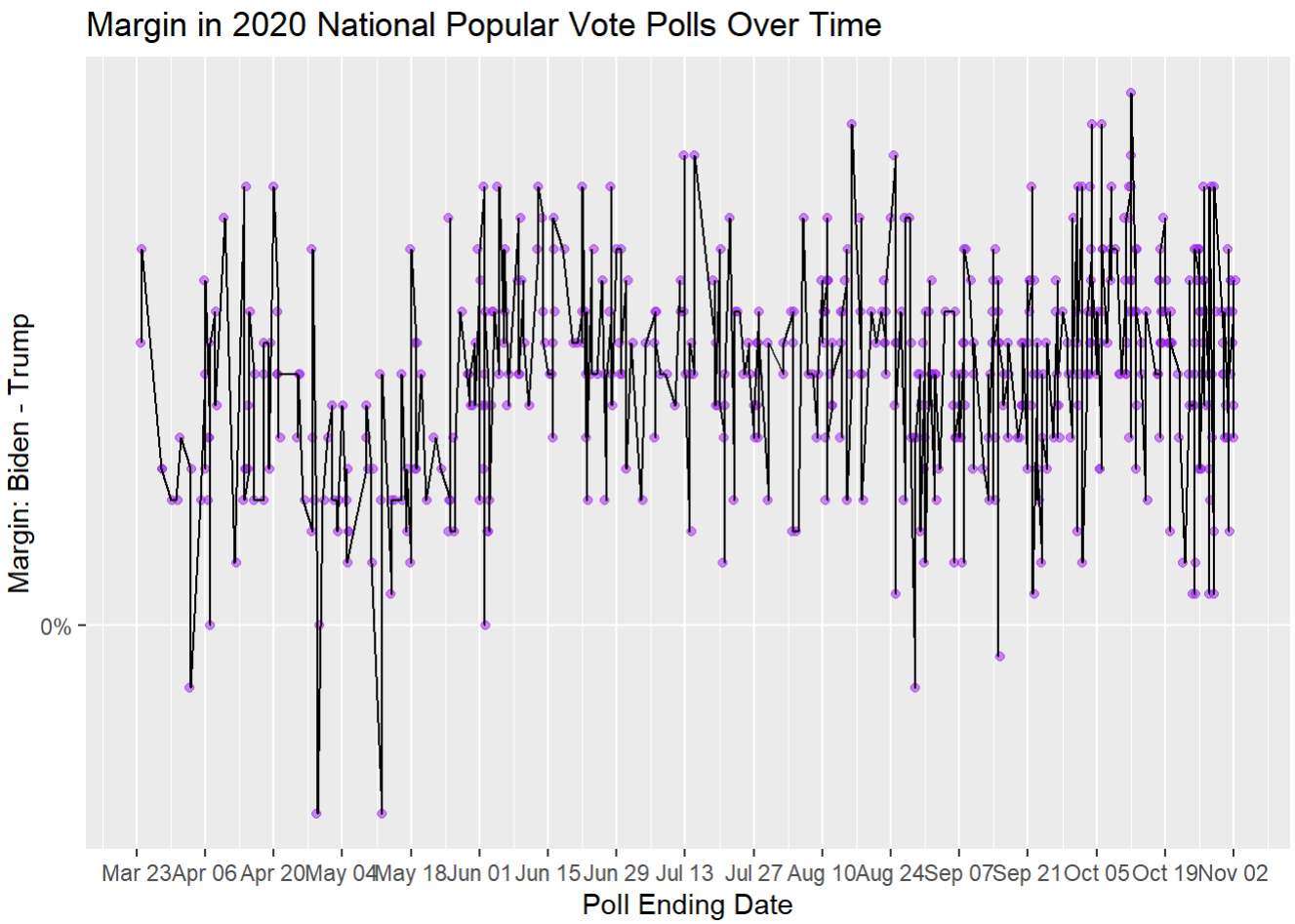
To help convey this information we can again use `geom_jitter` and alpha transparency instead of `geom_point` . Here we are adding +/- .005 to the y-value to change the value of the poll, but not the date. (Note that we could also use `position=jitter` when calling `geom_point`). This adds just enough error in the x (width) and y (height) values associated with each point so as to help distinguish how many observations might share a value. We are also going to use the alpha transparency to denote when lots of points occur on a similar (jittered) point. Note that when doing this code we are going to redo the plot from start to finish.

```
Pres2020.PV %>%
  ggplot(aes(x = EndDate, y = margin)) +
  labs(title="Margin in 2020 National Popular Vote Polls Over Time",
       y = "Margin: Biden - Trump",
       x = "Poll Ending Date") +
    geom_jitter(color = "PURPLE",height=.005, alpha = .4) +
    scale_y_continuous(breaks=seq(-.1,.2,by=.05),
                       labels= scales::percent_format(accuracy = 1)) +
    scale_x_date(date_breaks = "2 week", date_labels = "%b %d")
```



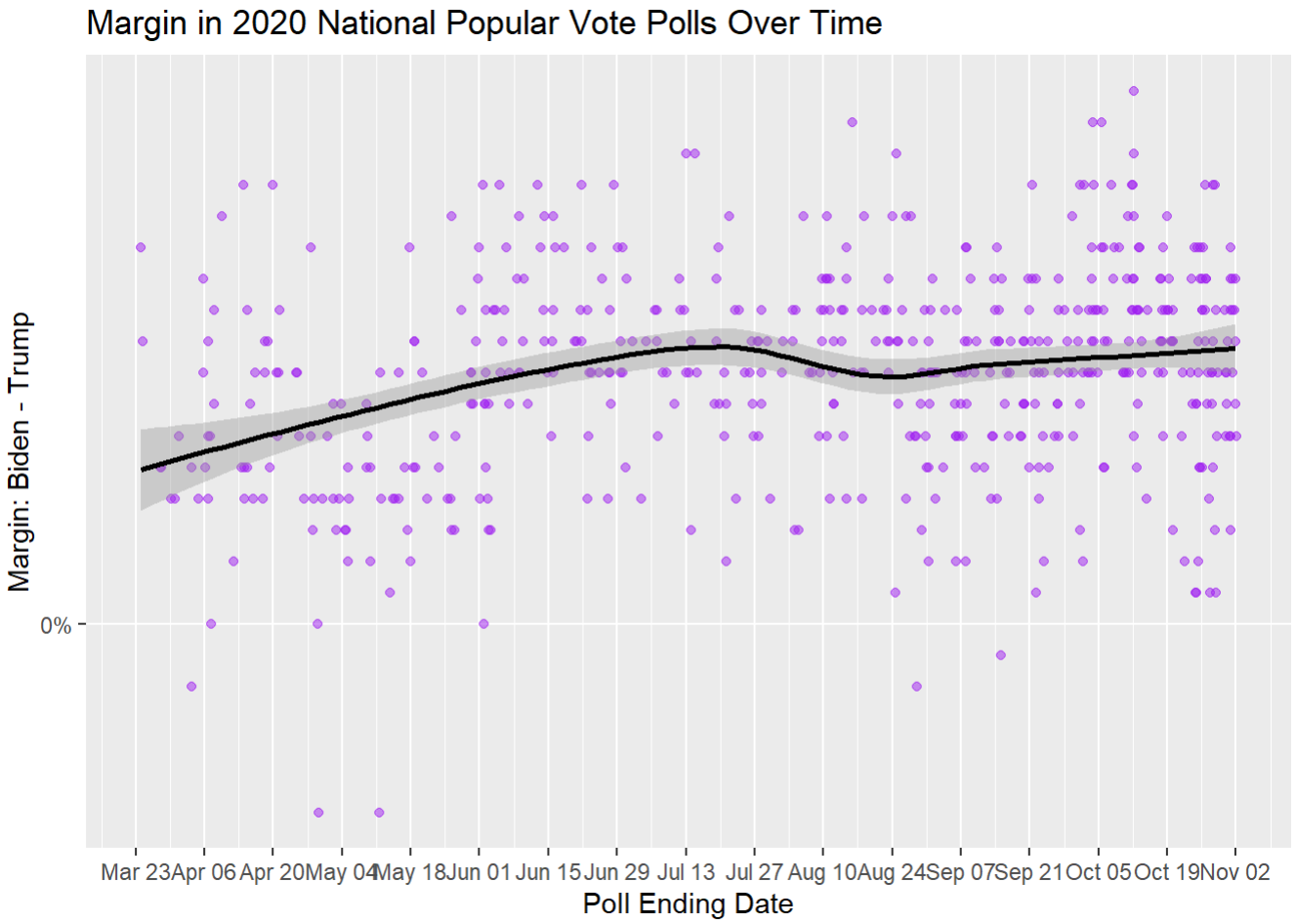
In addition to plotting plotting points using `geom_point` we can also add lines to the plot using `geom_line`. The line will connect the values in sequence so it does not always make sense to include. For example, if we add the `geom_line` to the plot it is hard to make the case that the results are meaningful.

```
Pres2020.PV %>%
  ggplot(aes(x = EndDate, y = margin)) +
  labs(title="Margin in 2020 National Popular Vote Polls Over Time",
       y = "Margin: Biden - Trump",
       x = "Poll Ending Date") +
    geom_jitter(color="purple", alpha = .5) +
    scale_y_continuous(breaks=seq(-.1,.2,by=.05),
                       labels= scales::percent_format(accuracy = 1)) +
    scale_x_date(date_breaks = "2 week", date_labels = "%b %d") +
  geom_line()
```



While `geom_line` may help accentuate variation over time, it is really not designed to summarize a relationship in the data as it is simply connecting sequential points (arranged according to the x-axis). In contrast, if we were to add `geom_smooth` then the plot would add the average value of nearby points to help summarize the trend. (Note that we have opted to include the associated uncertainty in the moving average off using the `se=T` parameter in `geom_smooth`.)

```
# Using message=FALSE to suppress note about geom_smooth
Pres2020.PV %>%
  ggplot(aes(x = EndDate, y = margin)) +
  labs(title="Margin in 2020 National Popular Vote Polls Over Time",
       y = "Margin: Biden - Trump",
       x = "Poll Ending Date") +
    geom_jitter(color="purple", alpha = .5) +
    scale_y_continuous(breaks=seq(-.1,.2,by=.05),
                       labels= scales::percent_format(accuracy = 1)) +
    scale_x_date(date_breaks = "2 week", date_labels = "%b %d") +
    geom_smooth(color = "BLACK", se=T)
```



Plotting Multiple Variables Over Time (Time-Series)

- Can we plot support for Biden and support for Trump separately over time (on the same plot)?

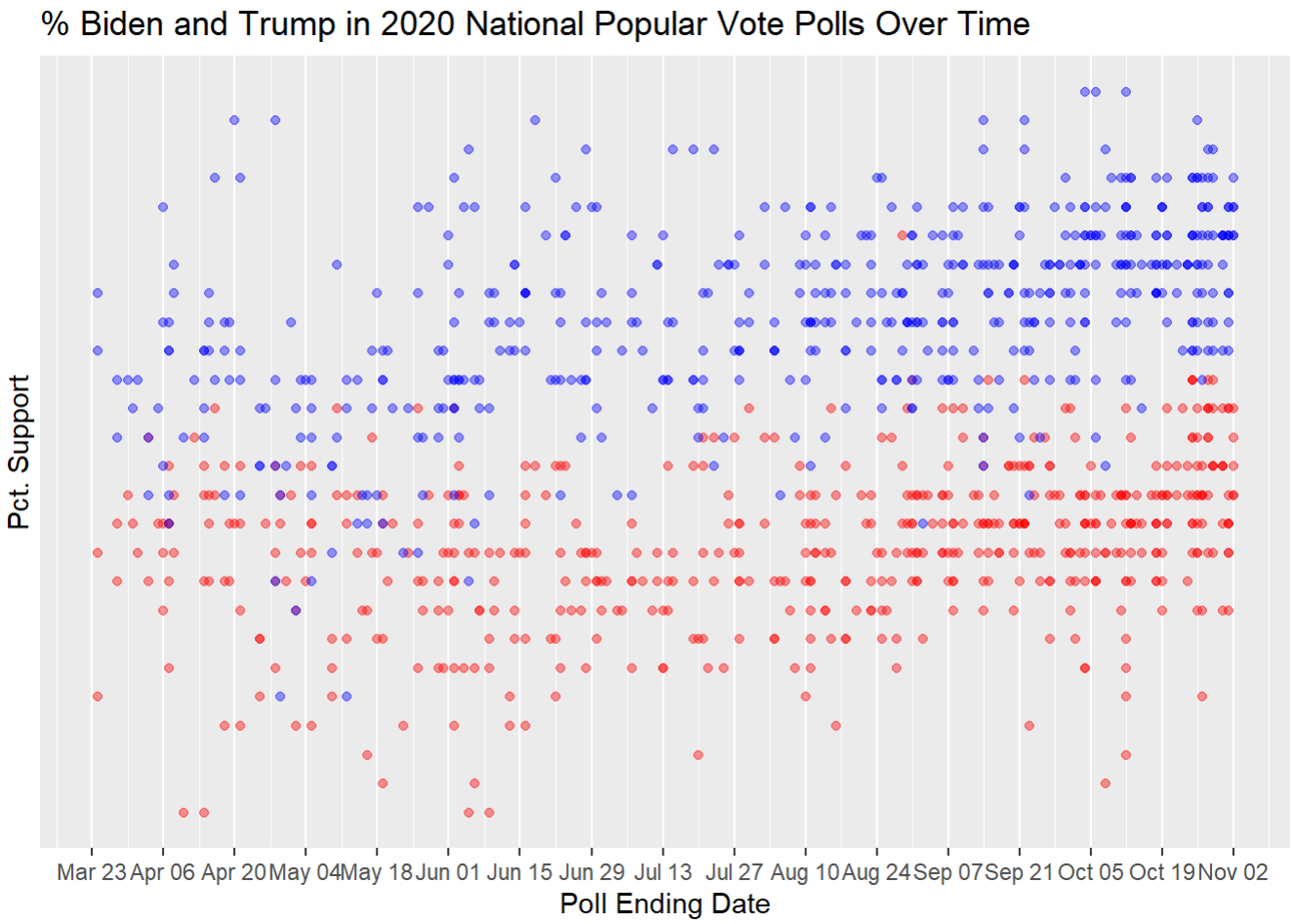
Let's define a `ggplot` object to add to later. Note that this code chuck will not print anything because we need to call the object to see what it produced.

```
BidenTrumpplot <- Pres2020.PV %>%
  ggplot()  +
  geom_point(aes(x = EndDate, y = Trump),
             color = "red", alpha=.4)  +
  geom_point(aes(x = EndDate, y = Biden),
             color = "blue", alpha=.4) +
  labs(title="% Biden and Trump in 2020 National Popular Vote Polls Over Time",
       y = "Pct. Support",
       x = "Poll Ending Date") +
  scale_x_date(date_breaks = "2 week", date_labels = "%b %d") +
  scale_y_continuous(breaks=seq(.3,.7,by=.05),
                     labels= scales::percent_format(accuracy = 1))
```

- Note the use of `aes` in `geom_point()` !

Now call the plot

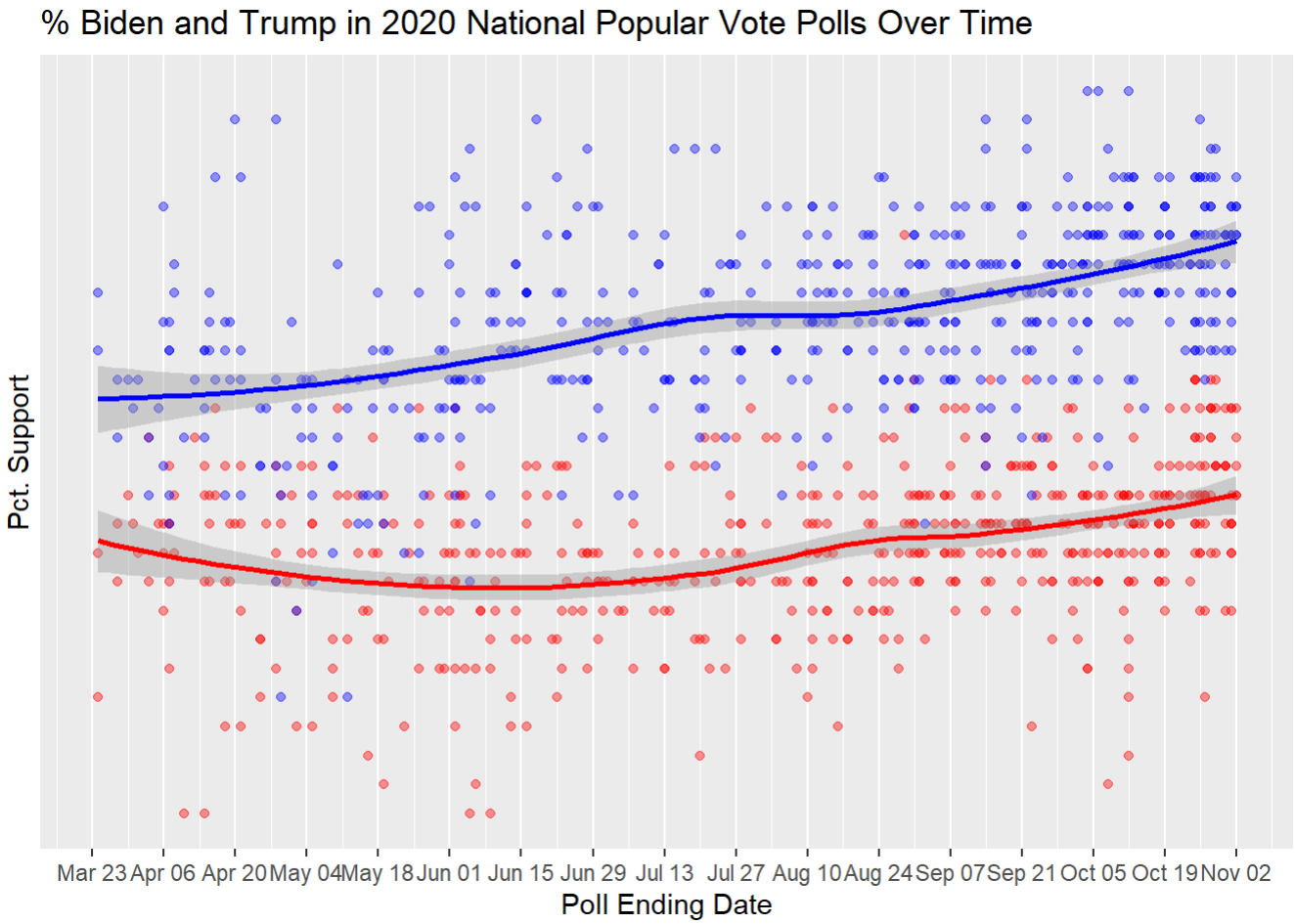
```
BidenTrumpplot
```



Now we are going to add smoothed lines to the ggplot object. The nice thing about having saved the ggplot object above is that to add the smoothed lines we can simply add it to the existing plot. `geom_smooth` summarizes the average using a subset of the data (the default is 75%) by computing the average value for the closest 75% of the data. Put differently, after sorting the polls by their date it then summarizes the predicted value for a poll taken at that date using the closest 75% of polls according to the date. (It is not taking a mean of those values, but it is doing something similar.) After doing so for the first date, it then does the same for the second date, but now the "closest" data includes polls that happened both before an after. The smoother "moves along" the x-axis and generates a predicted value for each date. Because it is using so much of the data, the values of the line will change very slowly because the only change between two adjacent dates is by dropping and adding new information.

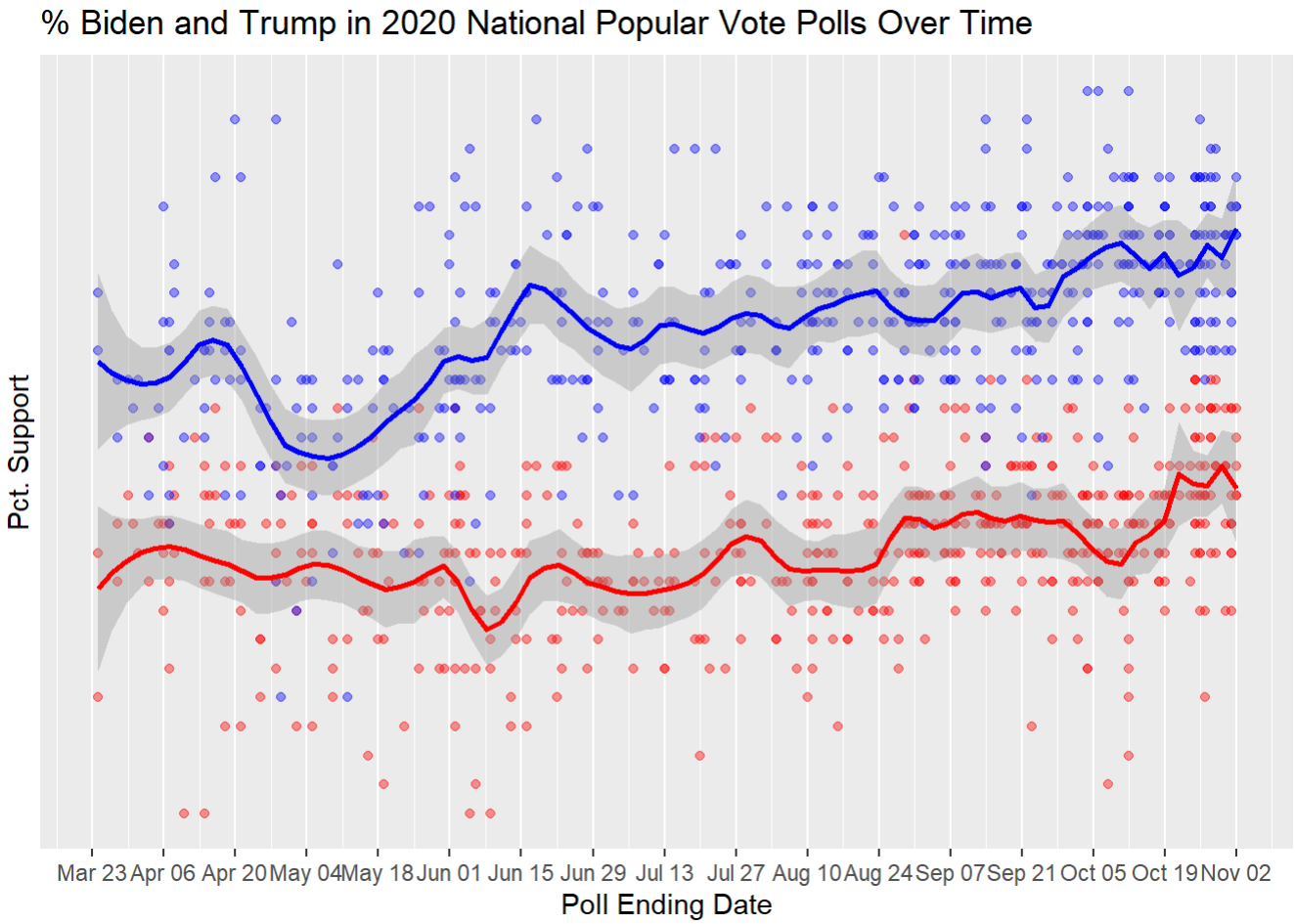
When calling the `geom_smooth` we used `se=T` to tell `ggplot` to produce an estimate of how much the prediction may vary. (This is the 95% confidence interval for the prediction being graphed.)

```
# Using message=FALSE to suppress note about geom_smooth
BidenTrumpplot +
  geom_smooth(aes(x = EndDate, y = Trump),
              color = "red",se=T) +
  geom_smooth(aes(x = EndDate, y = Biden),
              color = "blue",se=T)
```



If we change it to using only the closest 10% of the data by changing `span=.1` we get a slightly different relationship. The smoothed lines are less smooth because the impact of adding and removing points is much greater when we are using less data to compute the smoothed value – a single observation can change the overall results much more than when we used so much more data. In addition, the error-bars around the smoother are larger because we are using less data to calculate each point.

```
# Using message=FALSE to suppress note about geom_smooth
BidenTrumpplot +
  geom_smooth(aes(x = EndDate, y = Trump),
              color = "red",se=T,span=.1) +
  geom_smooth(aes(x = EndDate, y = Biden),
              color = "blue",se=T,span=.1)
```



Try some other values to see how things change. Note that you are not changing the data, you are only changing how you are visualizing the relationship over time by deciding how much data to use. In part, the decision of how much data to use is a question of how much you you want to allow public opinion to vary over the course of the campaign – how much of the variation is "real" versus how much is "noise"?

Quick Exercise Choose another span and see how it changes how you interpret how much variation there is in the data over time.

```
# INSERT CODE
```

ADVANCED! We can also use `fill` to introduce another dimension into the visualization. Consider for example, plotting support for Trump over time for mixed-mode polls versus telephone polls versus online-only polls. How would you go about doing this? You want to be careful not to make a mess however. Just because you can doesn't mean you should!

State Polls and the Electoral College

New functions and libraries:

- `plotly` library
- Working with dates
- Simple looping

First load the data of all state-level polls for the 2020 presidential election.

```
Pres2020.StatePolls <- read_rds(file="https://github.com/jbisbee1/DS1000_S2024/raw/main/
data/Pres2020_StatePolls.Rds")
glimpse(Pres2020.StatePolls)
```

```
## Rows: 1,545
## Columns: 19
## $ StartDate      <date> 2020-03-21, 2020-03-24, 2020-03-24, 2020-03-28, 2020-0…
## $ EndDate        <date> 2020-03-30, 2020-04-03, 2020-03-29, 2020-03-29, 2020-0…
## $ DaysinField    <dbl> 10, 11, 6, 2, 3, 5, 2, 2, 7, 3, 3, 3, 2, 2, 3, 4, 10, 1…
## $ MoE            <dbl> 2.8, 3.0, 4.2, NA, 4.0, 1.7, 3.0, 3.1, 4.1, 4.4, NA, NA…
## $ Mode           <chr> "Phone/Online", "Phone/Online", "Live phone - RDD", "Li…
## $ SampleSize     <dbl> 1331, 1000, 813, 962, 602, 3244, 1035, 1019, 583, 500, …
## $ Biden          <dbl> 41, 47, 48, 67, 46, 46, 46, 48, 52, 42, 48, 50, 52, 38,…
## $ Trump          <dbl> 46, 34, 45, 29, 46, 40, 48, 45, 39, 49, 47, 41, 43, 49,…
## $ Winner         <chr> "Rep", "Dem", "Dem", "Dem", "Dem", "Rep", "Dem", "Dem",…
## $ poll.predicted <dbl> 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ Funded         <chr> "UtahPolicy.com & KUTV 2News", "Sacred Heart University…
## $ Conducted      <chr> "Y2 Analytics", "GreatBlue Research", "LHK Partners Inc…
## $ margin         <dbl> -5, 13, 3, 38, 0, 6, -2, 3, 13, -7, 1, 9, 9, -11, 6, -1…
## $ DaysToED       <drtn> 218 days, 214 days, 219 days, 219 days, 216 days, 213 …
## $ StateName      <chr> "Utah", "Connecticut", "Wisconsin", "California", "Mich…
## $ EV             <int> 6, 7, 10, 55, 16, 29, 16, 16, 12, 15, 10, 16, 11, 6, 16…
## $ State          <chr> "UT", "CT", "WI", "CA", "MI", "FL", "GA", "MI", "WA", "…
## $ BidenCertVote  <dbl> 38, 59, 49, 64, 51, 48, 50, 51, 58, 49, 49, 51, 49, 41,…
## $ TrumpCertVote  <dbl> 58, 39, 49, 34, 48, 51, 49, 48, 39, 50, 49, 48, 49, 58,…
```

Variables of potential interest include:

- *Biden* : Percentage of respondents supporting Biden, the Democrat, in poll (0-100)
- *Trump* : Percentage of respondents supporting Trump, the Republican, in poll (0-100)
- *BidenCertVote* : Percentage of vote Biden, the Democrat, actually received in the election (0-100)
- *TrumpCertVote* : Percentage of vote Trump, the Democrat, actually received in the election (0-100)
- *Winner* : whether Biden won (“Dem”) or Trump won (“Rep”)
- *poll.predicted* : Indicator for whether the poll correctly predicted who won (1) or not (0)
- *State* & *StateName* : the state where the poll was conducted
- *EV* : the number of Electoral College Votes the state is worth for the winning candidate

Overall Task:

- How do we use this data to calculate the probability that Biden will win the presidency of the United States by winning the Electoral College?

Task 1: How should we translate a poll result into a predicted probability?

Suppose that I give you 10 polls from a state.

Load in the data and create the some mutations to create new variables.

```
Pres2020.StatePolls <- Pres2020.StatePolls %>%
   mutate(BidenNorm = Biden/(Biden+Trump),
          TrumpNorm = 1-BidenNorm,
          Biden = Biden/100,
          Trump=Trump/100)
```

How can you use them to create a probability? Discuss! (I can think of 3 ways.)

- Measure 1: Fraction of polls with Biden in the lead
- Measure 2: Biden Pct = Probability Biden wins
- Measure 3: Normalized Biden Pct = Probability Biden wins (i.e., all voters either vote for Biden or Trump). Sometimes called “two-party” vote.

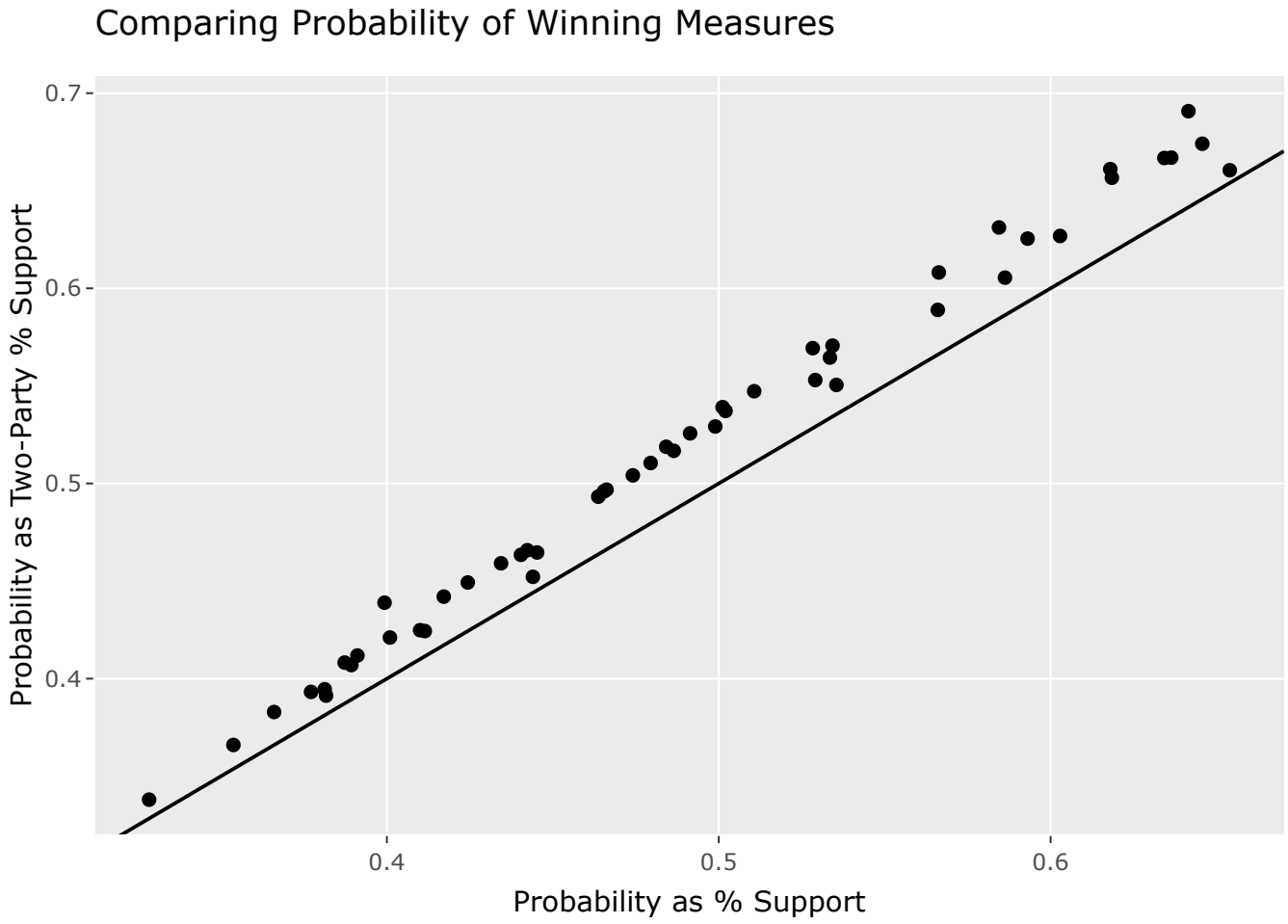
How does this vary across states? The joys of `group_by()`. Note that `group_by()` defines what happens for all subsequent code in that code chunk. So here we are going to calculate the mean separately for each state.

```
stateprobs <- Pres2020.StatePolls %>%
    group_by(StateName) %>%
      summarize(BidenProbWin1 = mean(Biden > Trump),
                BidenProbWin2 = mean(Biden),
                BidenProbWin3 = mean(BidenNorm))

stateprobs
```

```
## # A tibble: 50 × 4
##    StateName   BidenProbWin1 BidenProbWin2 BidenProbWin3
##    <chr>               <dbl>         <dbl>         <dbl>
##  1 Alabama             0             0.389         0.407
##  2 Alaska              0             0.442         0.466
##  3 Arizona             0.840         0.484         0.519
##  4 Arkansas            0             0.381         0.395
##  5 California          1             0.618         0.661
##  6 Colorado            1             0.534         0.571
##  7 Connecticut         1             0.584         0.631
##  8 Delaware            1             0.603         0.627
##  9 Florida             0.798         0.486         0.517
## 10 Georgia             0.548         0.474         0.504
## # ℹ 40 more rows
```

Clearly they differ, so let’s visualize to try to understand what is going on. Install the library `plotly`

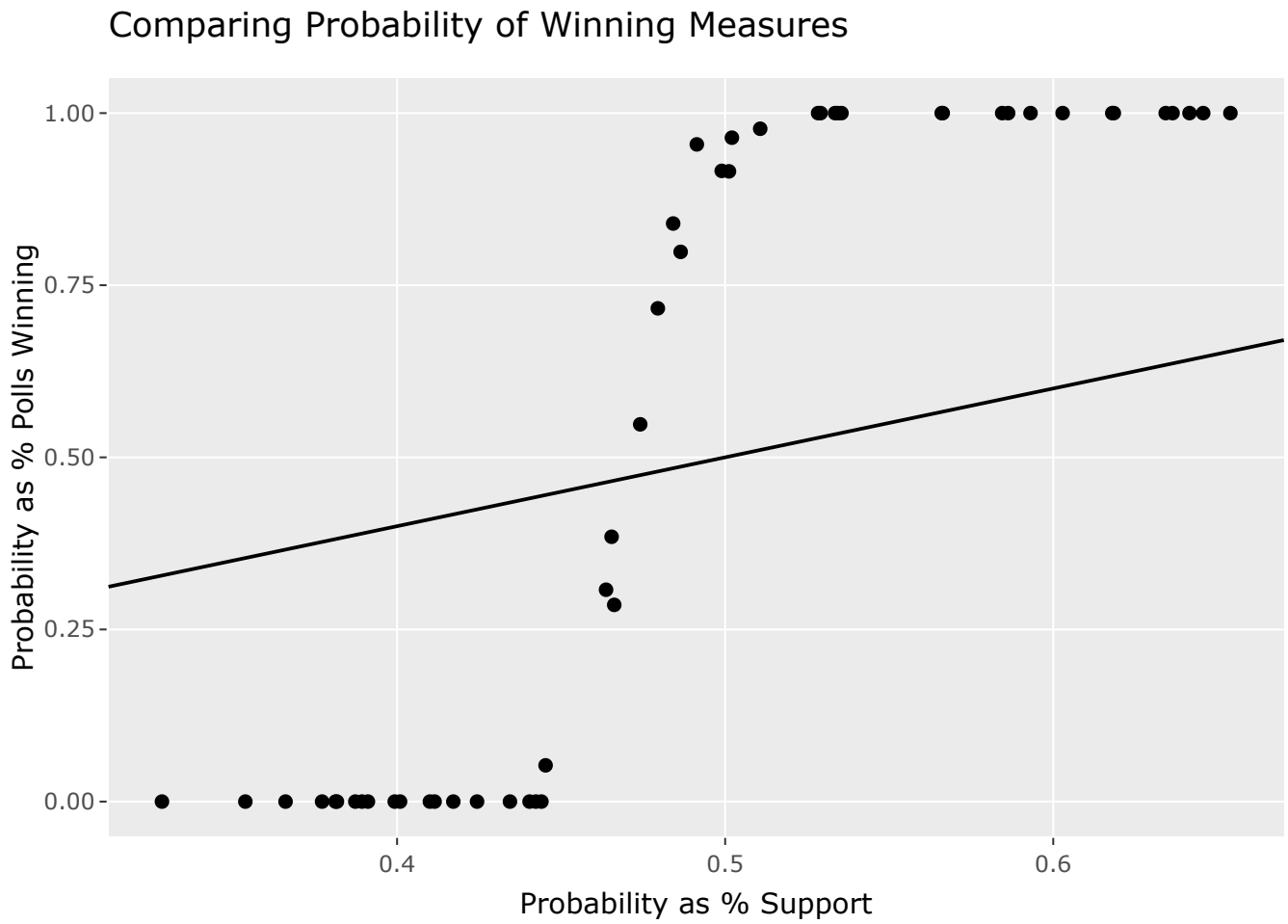
```
library(plotly)
gg <- stateprobs %>%
  ggplot(aes(x=BidenProbWin2, y=BidenProbWin3,text=paste(StateName))) +
  geom_point() +
  geom_abline(intercept=0,slope=1) +
  labs(x= "Probability as % Support",
       y = "Probability as Two-Party % Support",
       title = "Comparing Probability of Winning Measures")

ggplotly(gg,tooltip = "text")
```



So removing the undecided and making the probabilities for Biden and Trump sum to 100% is consequential.

What about if we compare these measures to the fration of polls with a given winner? After all, it seems implausible that the Biden would ever lose California or Trump would ever lose Tennessee.

```
library(plotly)
gg <- stateprobs %>%
  ggplot(aes(x=BidenProbWin2, y=BidenProbWin1,text=paste(StateName))) +
  geom_point() +
  geom_abline(intercept=0,slope=1) +
  labs(x= "Probability as % Support",
       y = "Probability as % Polls Winning",
       title = "Comparing Probability of Winning Measures")

ggplotly(gg,tooltip = "text")
```



So what do you think? Exactly the same data, but just different impications depending on how you choose to measure the probability of winning a state. Data sciene is as much about argument and reasoning as it is about coding. How we measure a concept is often critical to the conclusions that we get.

Task 2: Start “simple” – calculate the probability that Biden wins PA

But we want to combine these probabilities with the Electoral College votes in each state. Not every state has the same amount of Electoral College votes – it is typically given by the number of Senators (2) plus the number of representatives (at least 1) so we need to account for this if we want to make a projection about who is going to win the Electoral College.

- Create a tibble with just polls from PA.

```
PA.dat <- Pres2020.StatePolls %>%
  filter(State == "PA")
```

- Now compute these three probabilities. What functions do we need?

```
PA.dat %>%
      summarize(BidenProbWin1 = mean(Biden > Trump),
                BidenProbWin2 = mean(Biden),
                BidenProbWin3 = mean(BidenNorm))
```

```
## # A tibble: 1 × 3
##   BidenProbWin1 BidenProbWin2 BidenProbWin3
##           <dbl>         <dbl>         <dbl>
## 1         0.916         0.499         0.529
```

- What do you think about this?

Task 3: Given that probability, how do we change the code to compute the expected number of Electoral College Votes `EV` for Biden?

- Keep the code from above and copy and paste so you can understand how each step changes what we are doing. Note that we have the number of electoral college votes associated with each state `EV` that we want to use to compute the expected number of electoral college votes. But recall that when we `summarize` we change the tibble to be the output of the function. So how do we keep the number of Electoral College votes for a future mutation?

```
PA.dat %>%
      summarize(BidenProbWin1 = mean(Biden > Trump),
                BidenProbWin2 = mean(Biden),
                BidenProbWin3 = mean(BidenNorm),
                EV = mean(EV)) %>%
      mutate(BidenEV1 = BidenProbWin1*EV,
             BidenEV2 = BidenProbWin2*EV,
             BidenEV3 = BidenProbWin3*EV)
```

```
## # A tibble: 1 × 7
##   BidenProbWin1 BidenProbWin2 BidenProbWin3    EV BidenEV1 BidenEV2 BidenEV3
##           <dbl>         <dbl>         <dbl> <dbl>    <dbl>    <dbl>    <dbl>
## 1         0.916         0.499         0.529    20     18.3     9.98     10.6
```

Note that we are calculation the Expected Value of the Electoral College votes using: *Probability that Biden wins state i* X *Electoral College Votes in State i*. This will allocate fractions of Electoral College votes even though the actual election is winner-take all. This is OK because the fractions reflect the probability that an alternative outcome occurs.

Quick Exercise How can we get compute the expected number of Electoral College votes for Trump in each measure? NOTE: There are at least 2 ways to do this because this is a 2 candidate race

```
# INSERT CODE HERE
```

- `EV-BidenEV`, or compute `TrumpProbWin`

Task 4: Now generalize to every state by applying this code to each set of state polls.

- What do we need to do this calculation for every state in our tibble?
- First, compute probability of winning a state. (How?)
- Second, compute expected Electoral College Votes. (How?)

```
Pres2020.StatePolls %>%
  group_by(StateName) %>%
    summarize(BidenProbWin1 = mean(Biden > Trump),
              BidenProbWin3 = mean(BidenNorm),
              EV = mean(EV),
              State = first(State)) %>%
    mutate(State = State,
              BidenECVPredicted1 = EV*BidenProbWin1,
              TrumpECVPredicted1 = EV- BidenECVPredicted1,
              BidenECVPredicted3 = EV*BidenProbWin3,
              TrumpECVPredicted3 = EV- BidenECVPredicted3) %>%
  summarize(BidenECVPredicted1=sum(BidenECVPredicted1),
            BidenECVPredicted3=sum(BidenECVPredicted3),
            TrumpECVPredicted1=sum(TrumpECVPredicted1),
            TrumpECVPredicted3=sum(TrumpECVPredicted3),)
```

```
## # A tibble: 1 × 4
##   BidenECVPredicted1 BidenECVPredicted3 TrumpECVPredicted1 TrumpECVPredicted3
##                <dbl>              <dbl>              <dbl>              <dbl>
## 1               345.               289.               190.               246.
```

Task 5: Now compute total expected vote by adding to that code

- NOTE: Actually 306 - 232
- What do we need to do to the tibble we created in Task 4 to get the overall number of Electoral College Votes?

```
Pres2020.StatePolls %>%
  group_by(StateName) %>%
    summarize(BidenProbWin1 = mean(Biden > Trump),
              BidenProbWin3 = mean(BidenNorm),
              EV = mean(EV)) %>%
    mutate(BidenECVPredicted1 = EV*BidenProbWin1,
              TrumpECVPredicted1 = EV- BidenECVPredicted1,
              BidenECVPredicted3 = EV*BidenProbWin3,
              TrumpECVPredicted3 = EV- BidenECVPredicted3) %>%
    summarize(BidenECV1 = sum(BidenECVPredicted1),
              TrumpECV1 = sum(TrumpECVPredicted1),
              BidenECV3 = sum(BidenECVPredicted3),
              TrumpECV3 = sum(TrumpECVPredicted3))
```

```
## # A tibble: 1 × 4
##   BidenECV1 TrumpECV1 BidenECV3 TrumpECV3
##       <dbl>     <dbl>     <dbl>     <dbl>
## 1      345.      190.      289.      246.
```

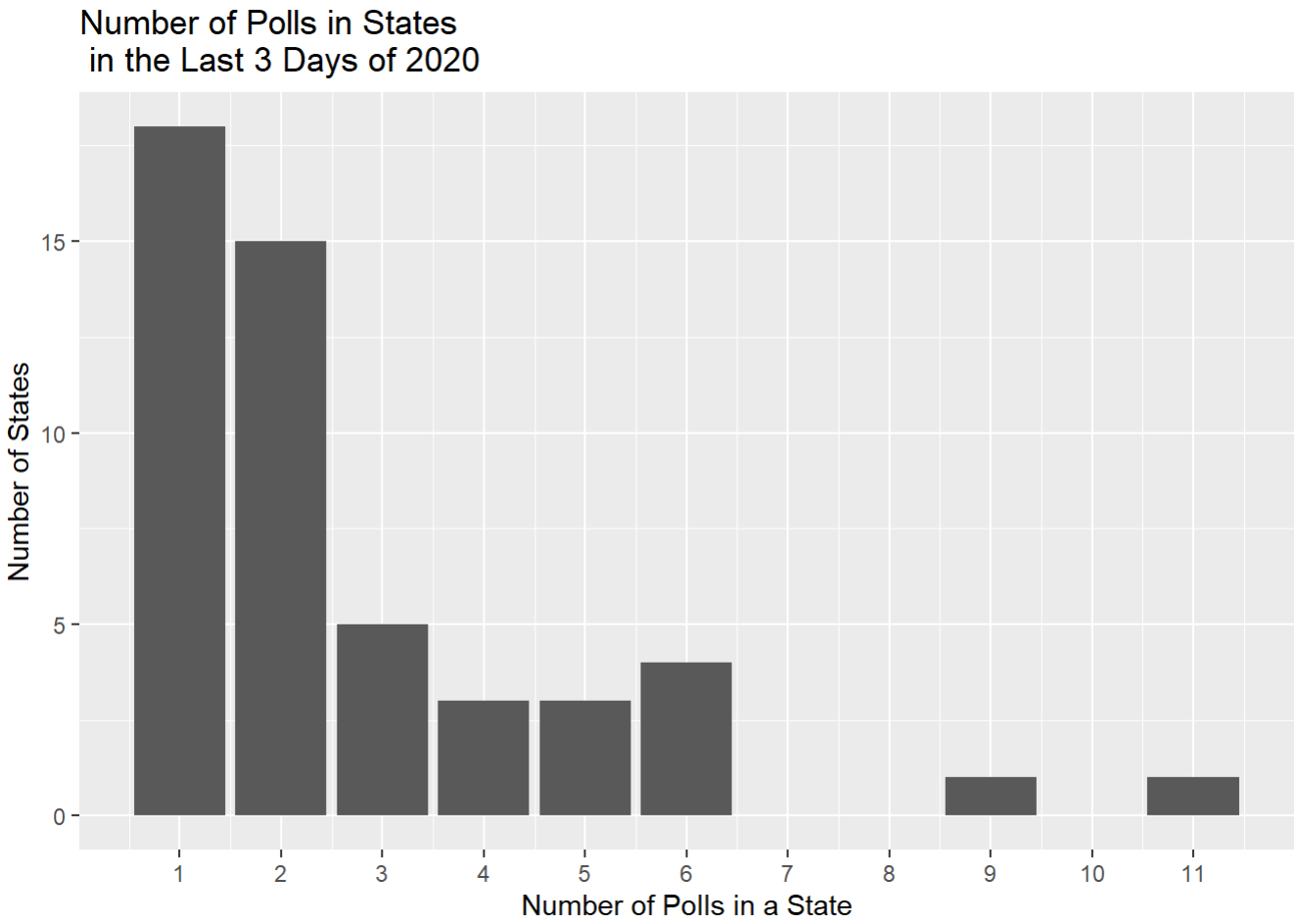
Quick Exercise Could also do this for just polls conducted in the last 7 days. How?

```
# INSERT CODE HERE
```

THINKING: What about states that do not have any polls? What should we do about them? Is there a reason why they might not have a poll? Is that useful information? Questions like this become more relevant when we start to restrict the sample.

Here are the number of polls done in each state in the last 3 days. Note that when we use fewer days our measure based on the percentage of polls won may be more affected?

```
Pres2020.StatePolls %>%
  filter(DaysToED < 3) %>%
  count(State) %>%
  ggplot(aes(x=n)) +
  geom_bar() +
  scale_x_continuous(breaks=seq(0,15,by=1)) +
  labs(x="Number of Polls in a State",
       y="Number of States",
       title="Number of Polls in States \n in the Last 3 Days of 2020")
```


Number of Polls in States
in the Last 3 Days of 2020
Number of States
0
5
10
15
1
2
3
4
5
6
7
8
9
10
11
Number of Polls in a State