Topic 6. Conditional Relationships and Visualization, Part 1

Prof. Josh Clinton, Vanderbilt University

2022-06-14

Today's Agenda

- Conditional data: when a variable varies with respect to some other variable.
- How does the value of the outcome of interest vary *depending* on the value of another variable of interest?
- Typically: outcome of interest (dependent variable), Y-axis.
- Other variables possibly related to the outcome (independent variables): X-axis

Our tools depend on the **type of variables** we are trying to graph.

Returning to the "gender" gap

Conditional variation involves examining how the values of two or more variables are related to one another. Earlier we made these comparisons by creating different tibbles and then comparing across tibbles, but we can also make comparisons without creating multiple tibbles.

So load in the Michigan 2020 Exit Poll Data.

```
library(tidyverse)
library(scales)
mi_ep <- readRDS("MI2020_ExitPoll_small.Rds")

MI_final_small <- mi_ep %>%
  filter(preschoice=="Donald Trump, the Republican" | preschoice=="Joe Biden, the Democrat") %>%
  mutate(BidenVoter=ifelse(preschoice=="Joe Biden, the Democrat",1,0),
         TrumpVoter=ifelse(BidenVoter==1,0,1),
         AGE10=ifelse(AGE10==99,NA,AGE10))
```

We learned that if we `count` using multiple variables that R will count within values. Can we use this to analyze how this varies by groups? Let's see!

```
MI_final_small %>%
  filter(AGE10==1) %>%
  count(preschoice,SEX) %>%
  mutate(PctSupport = n/sum(n),
         PctSupport = round(PctSupport, digits=2))
```

```
## # A tibble: 4 x 4
##   preschoice                     SEX     n PctSupport
##   <chr>                        <dbl> <int>      <dbl>
## 1 Donald Trump, the Republican     1     7       0.22
```

```
## 2 Donald Trump, the Republican     2     1       0.03
## 3 Joe Biden, the Democrat          1     9       0.28
## 4 Joe Biden, the Democrat          2    15       0.47
```

Here we have broken everything out by both `preschoice` and `SEX` but the `PctSupport` is not quite what we want because it is the fraction of responses (out of 1) that are in each row rather than the proportion of support for each candidate **by** sex.

To correct this and to perform the functions within a value we need to use the `group_by` function.

We can use the `group_by` command to organize our data a bit better. What `group_by` does is to run all subsequent code separately according to the defined group.

So instead of running a count or summarize separately for both Males and Females as we did above, we can `group_by` the variable `SEX.chr` (or `FEMALE` or `SEX` – it makes no difference as they are all equivalent) and then preform the subsequent commands. So here we are going to filter to select those who are 24 and below and then we are going to count the number of Biden and Trump supporters within each value of `SEX.chr`

```
MI_final_small %>%
  filter(AGE10==1) %>%
  group_by(SEX) %>%
  count(preschoice)
```

```
## # A tibble: 4 x 3
## # Groups:   SEX [2]
##     SEX preschoice                          n
##   <dbl> <chr>                           <int>
## 1     1 Donald Trump, the Republican     7
## 2     1 Joe Biden, the Democrat          9
## 3     2 Donald Trump, the Republican     1
## 4     2 Joe Biden, the Democrat         15
```

Note that any functions of the data are also now organized by that grouping, so if we were to manually compute the proportions using the mutation approach discussed above we would get:

```
MI_final_small %>%
  filter(AGE10==1) %>%
  group_by(SEX) %>%
  count(preschoice) %>%
  mutate(PctSupport = n/sum(n),
         PctSupport = round(PctSupport, digits=2))
```

```
## # A tibble: 4 x 4
## # Groups:   SEX [2]
##     SEX preschoice                          n PctSupport
##   <dbl> <chr>                           <int>      <dbl>
## 1     1 Donald Trump, the Republican     7       0.44
## 2     1 Joe Biden, the Democrat          9       0.56
## 3     2 Donald Trump, the Republican     1       0.06
## 4     2 Joe Biden, the Democrat         15       0.94
```

So you can see that `PctSupport` sums to 2.0 because it sums to 1.0 within each value of the grouping variable `SEX`.

If we wanted the fraction of voters who are in each unique category - so that the percentage of all the categoreis sum to 1.0 – we would want to `ungroup` before doing the mutation that calculates the percentage. So here we are doing the functions after the `group_by()` separately for each value of the grouping variables (here `SEX`) and then we are going to then undo that and return to the entire dataset.

```
MI_final_small %>%
  filter(AGE10==1) %>%
  group_by(SEX) %>%
  count(preschoice) %>%
  ungroup() %>%
  mutate(PctSupport = n/sum(n),
         PctSupport = round(PctSupport, digits=2))
```

```
## # A tibble: 4 x 4
##     SEX preschoice                          n PctSupport
##   <dbl> <chr>                           <int>      <dbl>
## 1     1 Donald Trump, the Republican     7       0.22
## 2     1 Joe Biden, the Democrat          9       0.28
## 3     2 Donald Trump, the Republican     1       0.03
## 4     2 Joe Biden, the Democrat         15       0.47
```

If we are just interested in the proportion and we do not care about the number of respondents in each value (although here it seems relevant!) we could also `group_by` and then `summarize` as follows:

```
MI_final_small %>%
  filter(AGE10==1) %>%
  group_by(SEX) %>%
  summarize(PctBiden = mean(BidenVoter),
            PctTrump = mean(TrumpVoter)) %>%
  mutate(PctBiden = round(PctBiden, digits =2),
         PctTrump = round(PctTrump, digits =2))
```

```
## # A tibble: 2 x 3
##     SEX PctBiden PctTrump
##   <dbl>    <dbl>    <dbl>
## 1     1     0.56     0.44
## 2     2     0.94     0.06
```

Because we have already filtered to focus only on Biden and Trump voters, we don't actually need both since `PctBiden = 1 - PctTrump` and `PctTrump = 1 - PctBiden`.

Note that we can have multiple groups. So if we want to group by age and by sex we can do the following...

```
MI_final_small %>%
  group_by(SEX, AGE10) %>%
  summarize(PctBiden = mean(BidenVoter)) %>%
  mutate(PctBiden = round(PctBiden, digits =2))
```

```
## `summarise()` has grouped output by 'SEX'. You can override using the `.groups` argument.
```

```
## # A tibble: 22 x 3
## # Groups:   SEX [2]
##      SEX AGE10 PctBiden
##    <dbl> <dbl>    <dbl>
##  1     1     1     0.56
##  2     1     2     0.58
##  3     1     3     0.58
##  4     1     4     0.76
##  5     1     5     0.58
##  6     1     6     0.42
##  7     1     7     0.46
##  8     1     8     0.56
```

```
##  9     1     9     0.61
## 10     1    10     0.57
## # ... with 12 more rows
```

We can also save it for later analysis and then filter or select the results. For example:

```
SexAge <- MI_final_small %>%
  group_by(SEX, AGE10) %>%
  summarize(PctBiden = mean(BidenVoter)) %>%
  mutate(PctBiden = round(PctBiden, digits =2)) %>%
  drop_na()
```

```
## `summarise()` has grouped output by 'SEX'. You can override using the `.groups` argument.
```

So if we want to look at the Biden support by age among females (i.e., `SEX==2`) we can look at:

```
SexAge %>%
  filter(SEX == 2)
```

```
## # A tibble: 10 x 3
## # Groups:   SEX [1]
##      SEX AGE10 PctBiden
##    <dbl> <dbl>    <dbl>
##  1     2     1     0.94
##  2     2     2     0.93
##  3     2     3     0.69
##  4     2     4     0.71
##  5     2     5     0.52
##  6     2     6     0.6
##  7     2     7     0.63
##  8     2     8     0.74
##  9     2     9     0.69
## 10     2    10     0.61
```

And for Men...

Dicrete Variable By Discrete Variable (Barplot)

If we are working with discrete/categorical/ordinal/data — i.e., variables that take on a finite (and small) number of unique values then we are interested in how to compare across bar graphs.

Before we used `geom_bar` to plot the number of observations associated with each value of a variable. But we often want to know how the number of observations may vary according to a second variable. For example, we care not only about why voters reported that they supported Biden or Trump in 2020 but we are also interested in knowing whether Biden and Trump voters were voting for similar or different reasons. Did voters differ in terms of why they were voting for a candidate in addition to who they were voting for? If so, this may suggest something about what each set of voters were looking for in a candidate.

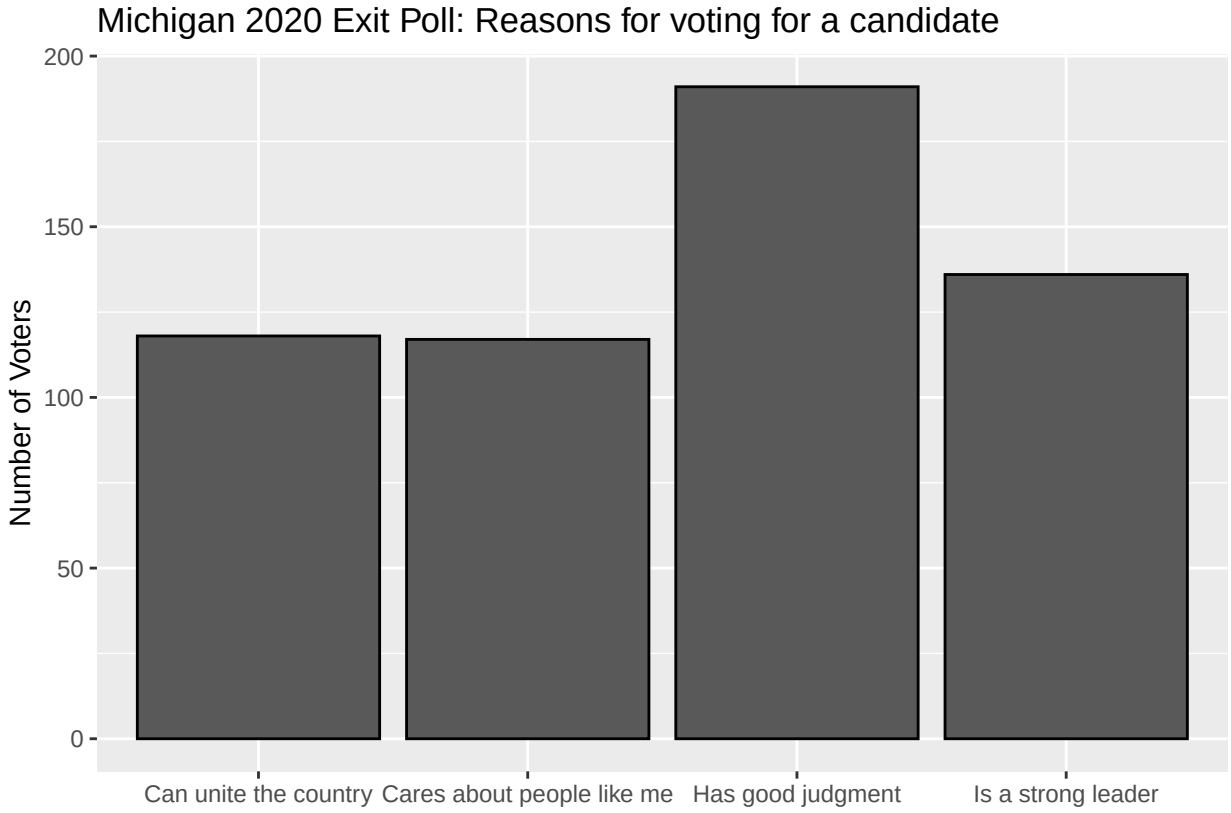
Let's first plot the barplot and then plot the barplot by presidential vote choice for the Michigan Exit Poll we were just analyzing.

We are interested in the distribution of responses to the variable `Quality` and we only care about voters who voted for either Biden or Trump (`preschoice`) so let's select those variables and `filter` using `preschoice` to select those respondents. We have an additional complication that the question was only asked of half of the respondents and some that were asked refused to answer. To remove these respondents we want to `drop_na` (note that this will drop every observation with a missing value – this is acceptable because we have used `select` to focus on the variables we are analyzing, but if we did not use `select` it would have dropped

an observation with missing data in **any** variable. We could get around this using `drop_na(Quality)` if we wanted). A final complication is that some respondents did not answer the question they were asked so we have to use `filter` to remove respondents with missing observations.

Now we include labels – note how we are suppressing the x-label because the value labels are self-explanatory in this instance and add the `geom_bar` as before.

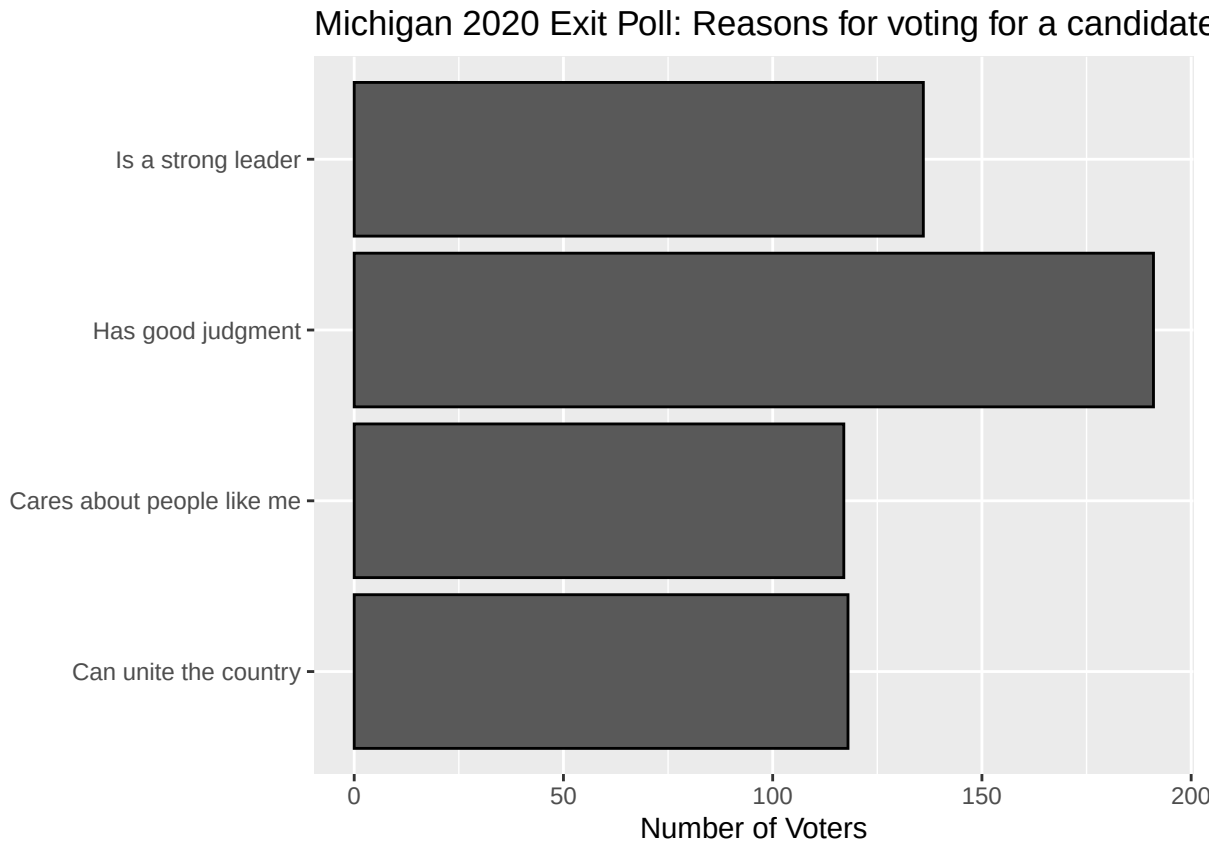
```
mi_ep %>%
    select(Quality,preschoice) %>%
    filter(preschoice == "Joe Biden, the Democrat" | preschoice == "Donald Trump, the Republican") %>%
    drop_na() %>%
    filter(Quality != "[DON'T READ] Don't know/refused") %>%
    ggplot(aes(x= Quality)) +
    labs(y = "Number of Voters",
         x = "",
         title = "Michigan 2020 Exit Poll: Reasons for voting for a candidate") +
    geom_bar(color="black")
```



Note that if we add `coord_flip` that we will flip the axes of the graph. (We could also have done this by changing `aes(y= Quality)`, but then we would also have to change the associated labels.)

```
mi_ep %>%
    select(Quality,preschoice) %>%
    filter(preschoice == "Joe Biden, the Democrat" | preschoice == "Donald Trump, the Republican") %>%
    drop_na() %>%
    filter(Quality != "[DON'T READ] Don't know/refused") %>%
    ggplot(aes(x= Quality)) +
    labs(y = "Number of Voters",
         x = "",
         title = "Michigan 2020 Exit Poll: Reasons for voting for a candidate") +
```

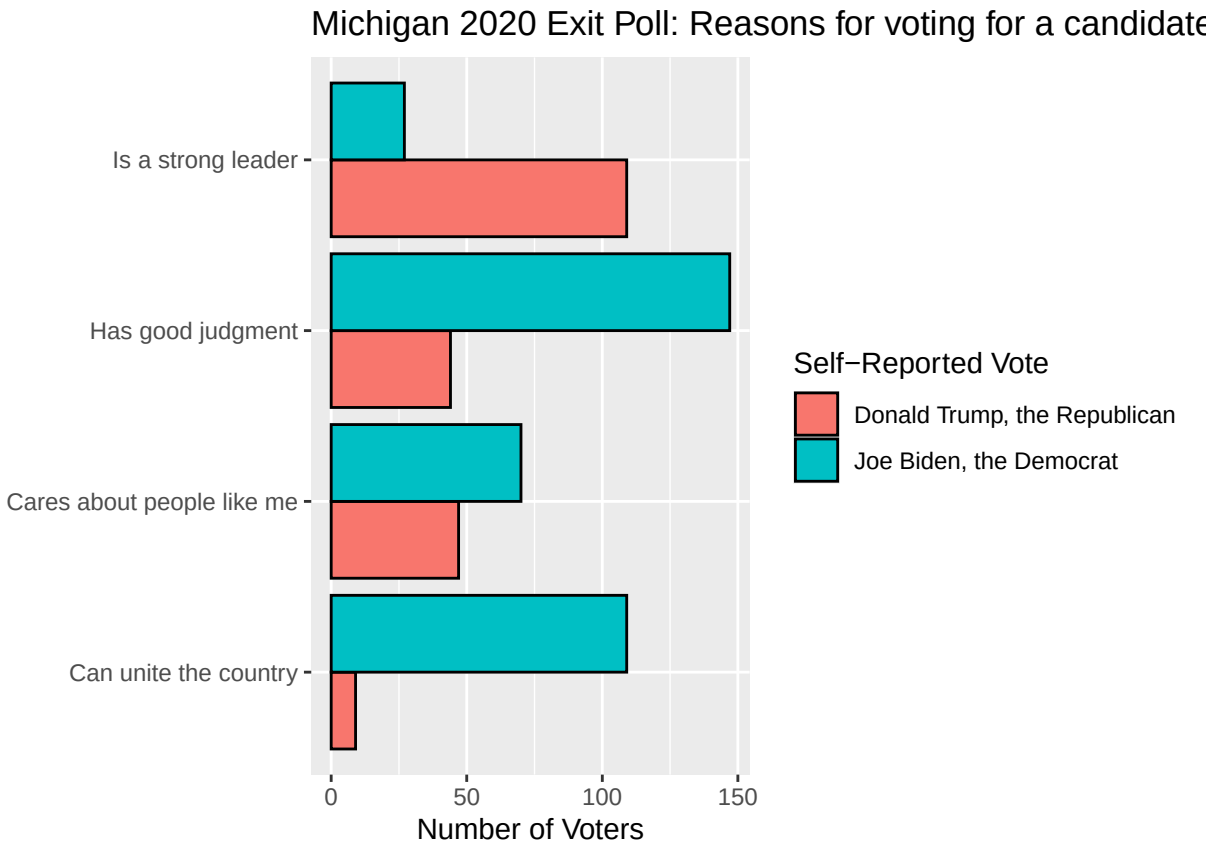
```
    geom_bar(color="black") +
  coord_flip()
```



So enough review, lets add another dimension to the data. To show how the self-reported reasons for voting for a presidential candidate varied by vote choice we are going to use the `fill` of the graph to create different color bars depending on the value of the character or factor variable that is used to `fill`.

So we are going to include as a `ggplot` aesthetic a character or factor variable as a `fill` (here `fill=preschoice`) and then we are going to also include `fill` in the `labs` function to make sure that we label the meaning of the values being plotted. The other change we have made is in `geom_bar` where we used `position=dodge` to make sure that the bars are plotted next to one-another rather than on top of one another.

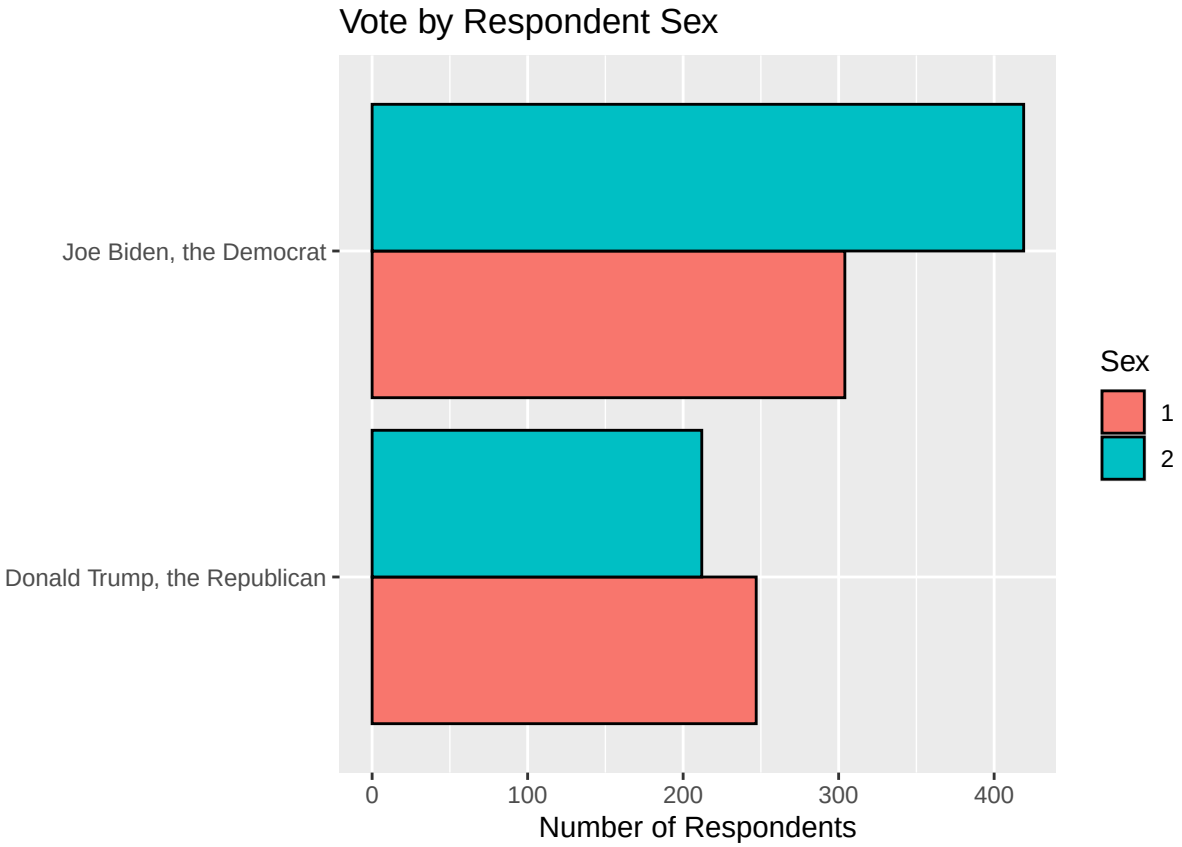
```
mi_ep %>%
    select(Quality,preschoice) %>%
    filter(preschoice == "Joe Biden, the Democrat" | preschoice == "Donald Trump, the Republican") %>%
    drop_na() %>%
    filter(Quality != "[DON'T READ] Don't know/refused") %>%
    ggplot(aes(x= Quality, fill = preschoice)) +
    labs(y = "Number of Voters",
         x = "",
         title = "Michigan 2020 Exit Poll: Reasons for voting for a candidate",
         fill = "Self-Reported Vote") +
    geom_bar(color="black", position="dodge") +
    coord_flip()
```



For fun, see what happens when you do not use `postion=dodge`. Also see what happens if you do not flip the coordinates using `coord_flip`.

It is important to note that the `fill` variable has to be a character or a factor. If we want to graph self-reported vote by sex, for example, we need to redefine the variable for the purposes of `ggplot` as follows. Note that because we are not mutating it and we are only defining it to be a factor within the '`ggplot` object, this redefinition will not stick. Note also the problem caused by uninformative values in `SEX` – can you change it.

```
mi_ep %>%
    filter(preschoice == "Joe Biden, the Democrat" | preschoice == "Donald Trump, the Republican") %>%
    ggplot(aes(x= preschoice, fill = factor(SEX))) +
    labs(y = "Number of Respondents",
         x = "",
         title = "Vote by Respondent Sex",
         fill = "Sex") +
    geom_bar(color="black", position="dodge") +
    coord_flip()
```



Quick Exercise The barplot we just produced does not satisfy our principles of visualization because the fill being used is uninterpretable to those unfamiliar with the dataset. Redo the code to use a `fill` variable that produces an informative label. Hint: don't overthink.

```
# INSERT CODE HERE
```

Continuous Variable By Discrete Variable (Histogram, Boxplot/Violinplot)

```
Pres2020.PV <- readRDS(file="Pres2020.PV.Rds")

Pres2020.PV <- Pres2020.PV %>%
                mutate(Trump = Trump/100,
                       Biden = Biden/100,
                       margin = Biden - Trump)
```

Suppose we were concerned with whether some polls might give different answers because of variation in who the poll is able to reach using that method. People who take polls via landline phones (do you even know what that is?) might differ from those who take surveys online. Or people contacted using randomly generated phone numbers (RDD) may differ from those contacted from a voter registration list that has had telephone numbers merged onto it.

Polls were done using lots of different methods in 2020.

```
Pres2020.PV %>%
  count(Mode)
```

```
## # A tibble: 9 x 2
##   Mode                 n
##   <chr>            <int>
## 1 IVR                  1
## 2 IVR/Online          47
## 3 Live phone - RBS    13
## 4 Live phone - RDD    51
## 5 Online             366
## 6 Online/Text          1
## 7 Phone - unknown      1
## 8 Phone/Online        19
## 9 <NA>                29
```

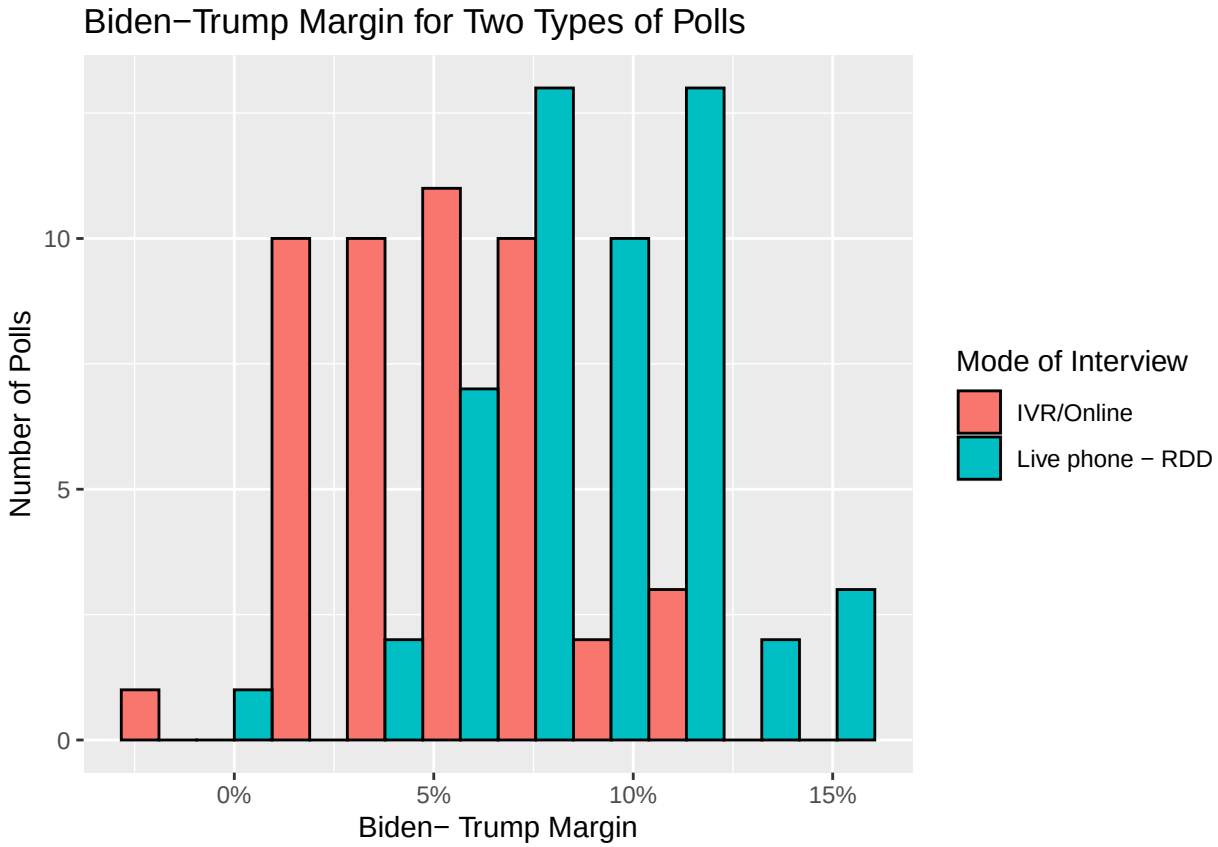
This raises the question of – how do we visualization variation in a variable by another variable? More specifically, how can we visualize how the `margin` we get using one type of survey compares to the `margin` from another type of poll? (We cannot use a scatterplot because the data is from different observations (here polls).)

We could do this using earlier methods by `select`ing polls with a specific interview method ("mode") and then plotting the `margin` (or `Trump` or `Biden`), but that will produce a bunch of separate plots that may be hard to directly compare. (In addition to having more things to look at we would want to make sure that the scale of the x-axis and y-axis are similar.)

We can plot another "layer" of data in 'ggplot using the `fill` paramter. Previously we used it to make the graphs look nice by choosing a particular color. But if we set `fill` to be a variable in our `tibble` then `ggplot` will plot the data seperately for each unique value in the named variable.

So if we want to plot the histogram of `margin` for two types of polls we can use the `fill` argument in `ggplot` to tell R to produce different fills depending on the value of that variable.

```
Pres2020.PV %>%
  filter(Mode == "IVR/Online" | Mode == "Live phone - RDD") %>%
    ggplot(aes(x= margin, fill = Mode)) +
  labs(y = "Number of Polls",
         x = "Biden- Trump Margin",
         title = "Biden-Trump Margin for Two Types of Polls",
        fill = "Mode of Interview") +
    geom_histogram(bins=10, color="black", position="dodge") +
    scale_x_continuous(breaks=seq(-.1,.2,by=.05),
                       labels= scales::percent_format(accuracy = 1))
```



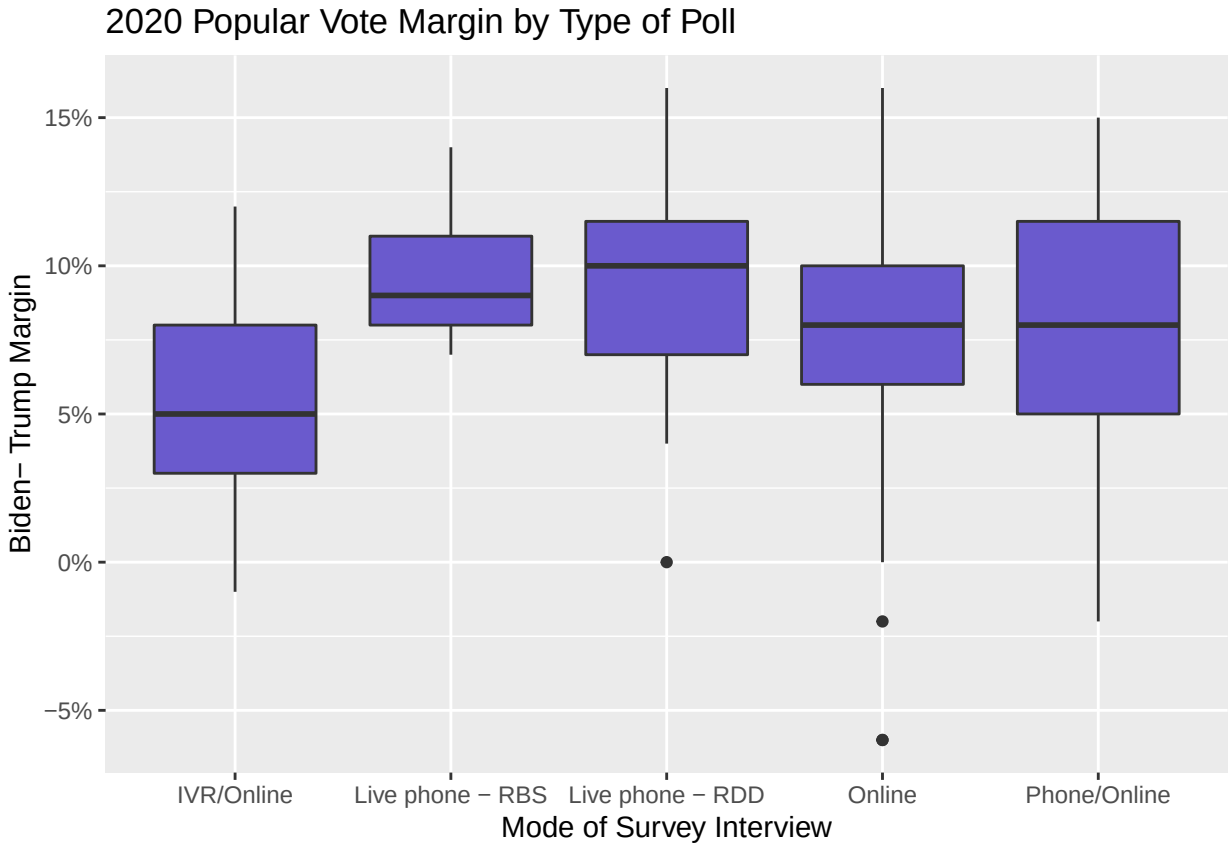
Quick Exercise Try running the code without the `filter`. What do you observe? How useful is this? Why or why not?

```
# INSERT CODE
```

While informative, it can be hard to compare the distribution of more than two categories using such methods. To compare the variation across more types of surveys we need to use a different visualization that summarizes the variation in the variable of interest a bit more. One common visualization is the `boxplot` which reports the mean, 25th percentile (i.e., the value of the data if we sort the data from lowest to highest and take the value of the observation that is 25% of the way through), the 75th percentile, the range of values, and notable outliers.

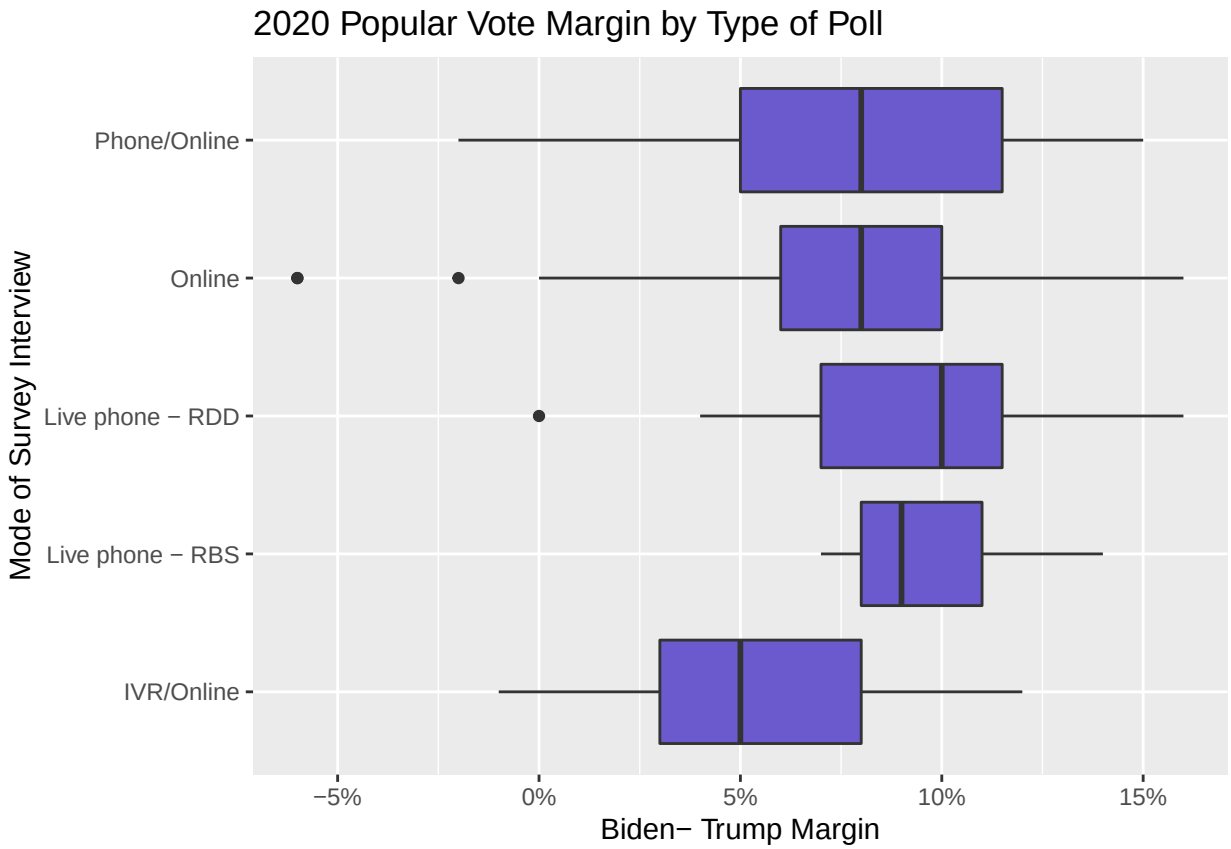
Let's see what the `boxplot` of survey mode looks like after we first drop surveys that were conducted using modes that were hardly used (or missing).

```
Pres2020.PV %>%
  filter(Mode != "IVR" & Mode != "Online/Text" & Mode != "Phone - unknown" & Mode != "NA") %>%
  ggplot(aes(x = Mode, y = margin)) +
    labs(x = "Mode of Survey Interview",
         y = "Biden- Trump Margin",
         title = "2020 Popular Vote Margin by Type of Poll") +
    geom_boxplot(fill = "slateblue") +
    scale_y_continuous(breaks=seq(-.1,.2,by=.05),
                       labels= scales::percent_format(accuracy = 1))
```



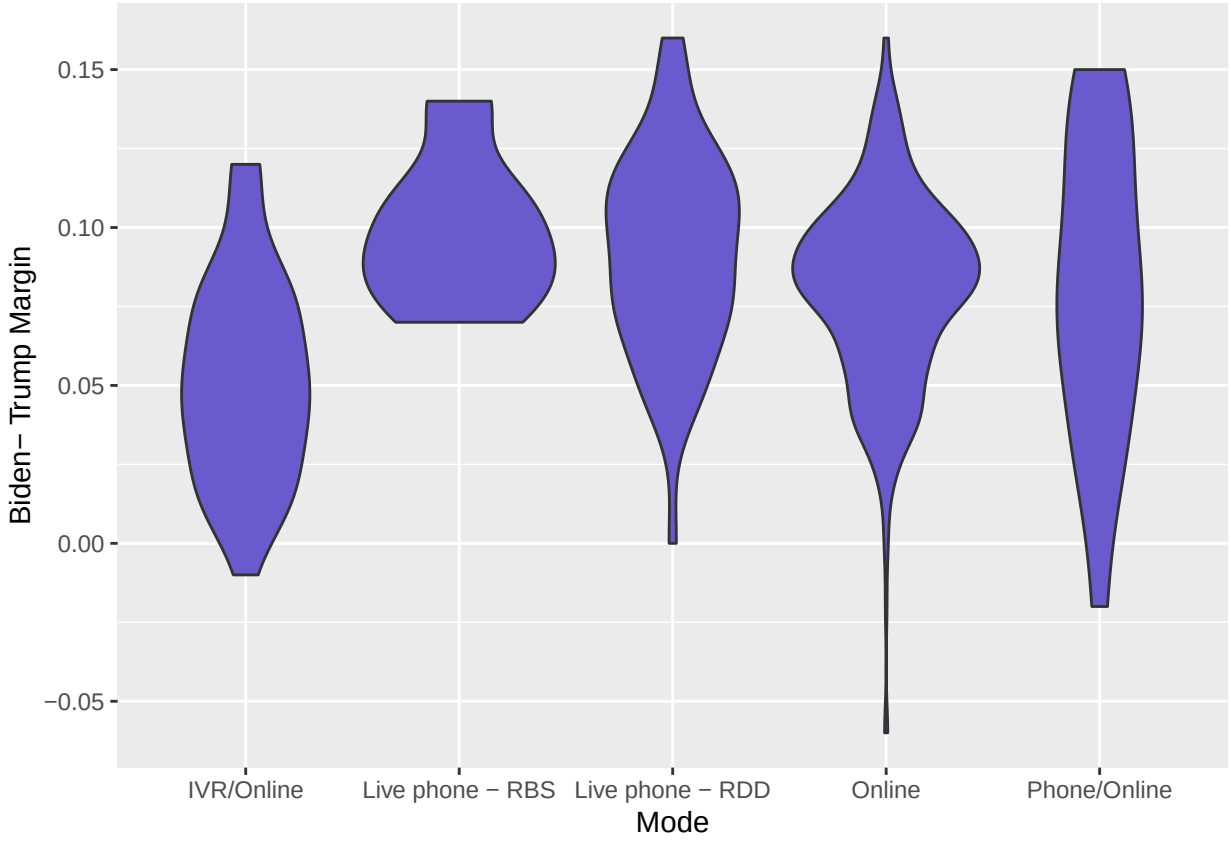
We can also flip the graph if we think it makes more sense to display it in a different orientation using `coord_flip`. (We could, of course, also redefine the x and y variables in the 'ggplot object, but it is useful to have a command to do this to help you determine which orientiation is most useful).

```
Pres2020.PV %>%
  filter(Mode != "IVR" & Mode != "Online/Text" & Mode != "Phone - unknown" & Mode != "NA") %>%
  ggplot(aes(x = Mode, y = margin)) +
    labs(x = "Mode of Survey Interview",
         y = "Biden- Trump Margin",
         title = "2020 Popular Vote Margin by Type of Poll") +
    geom_boxplot(fill = "slateblue") +
    scale_y_continuous(breaks=seq(-.1,.2,by=.05),
                       labels= scales::percent_format(accuracy = 1)) +
    coord_flip()
```



A downside of the boxplot is that it can be hard to tell how the data varies within each box. Is it equally spread out? How much data are contained in the lines (which are simply 1.5 times the height of the box)? To get a better handle on this we can use a "violin" plot that dispenses with a standard box and instead tries to plot the distribution of data within each category.

```
Pres2020.PV %>%
  filter(Mode != "IVR" & Mode != "Online/Text" & Mode != "Phone - unknown" & Mode != "NA") %>%
  ggplot(aes(x=Mode, y=margin)) +
    xlab("Mode") +
    ylab("Biden- Trump Margin") +
    geom_violin(fill="slateblue")
```



It is also hard to know **how much** data is being plotted. If some modes have 1000 polls and others have only 5 that seems relevant.

Quick Exercise We have looked at the difference in `margin`. How about differences in the percent who report supporting `Biden` and `Trump`? What do you observe. Does this suggest that the different ways of contacting respondents may matter in terms of who responds? Is there something else that may explain the differences (i.e., what are we assuming when making this comparison)?

```
# INSERT CODE HERE
```

Quick Exercise Some claims have been made that polls that used multiple ways of contacting respondents were better than polls that used just one. Can you evaluate whether there were differences in so-called "mixed-mode" surveys compared to single-mode surveys? (This requires you to define a new variable based on `Mode` indicating whether survey is mixed-mode or not.)

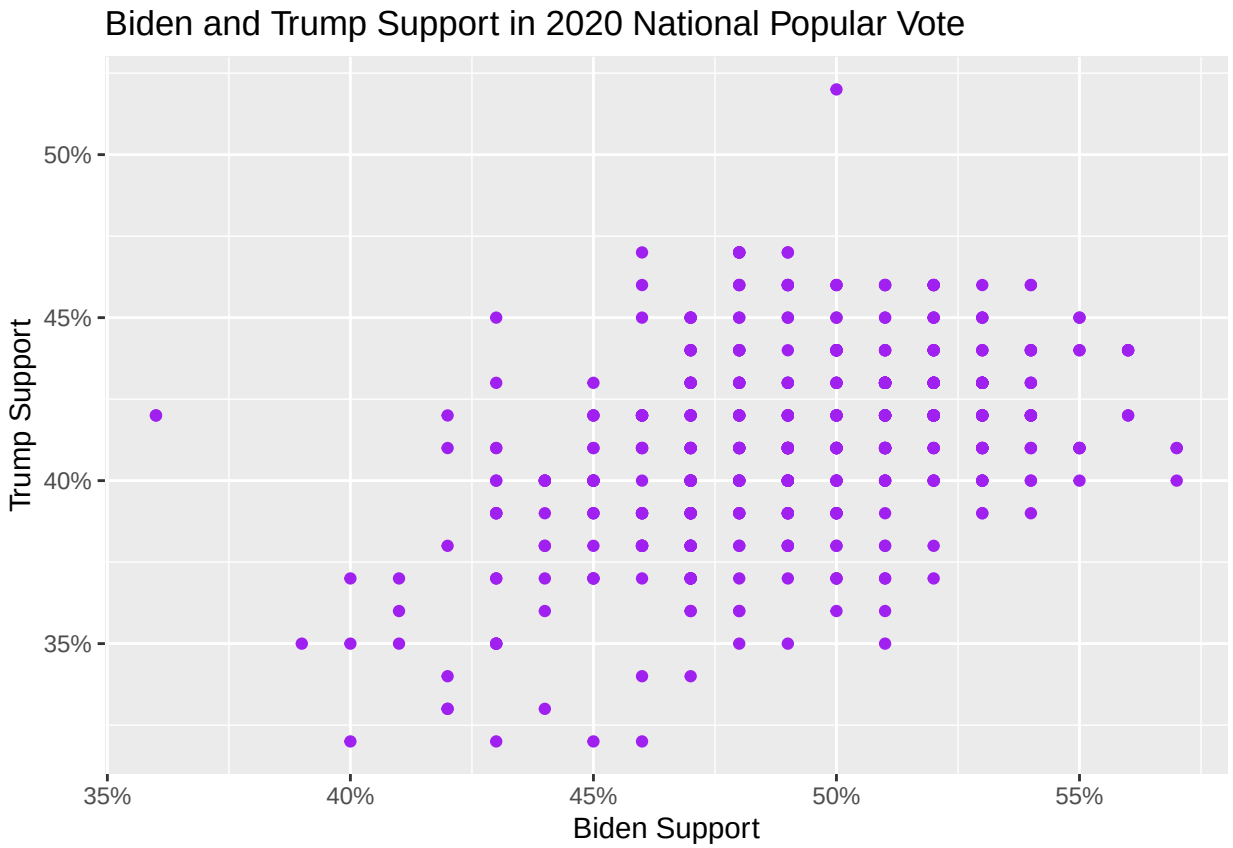
```
# INSERT CODE HERE
```

Continuous Variable By Continuous Variable (Scatterplot)

When we have two continuous variables we use a scatterplot to visualize the relationship. A scatterplot is simply a graph of every point in (x,y) where x is the value associated with the x-variable and y is the value associated with the y-variable. For example, we may want to see how support for Trump and Biden within a poll varies. So each observation is a poll of the national popular vote and we are going to plot the percentage of respondents in each poll supporting Biden against the percentage who support Trump.

To include two variables we are going to change our aesthetic to define both an x variable and a y variable – here `aes(x = Biden, y = Trump)` and we are going to label and scale the axes appropriately.

```
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_point(color="purple") +
    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))
```

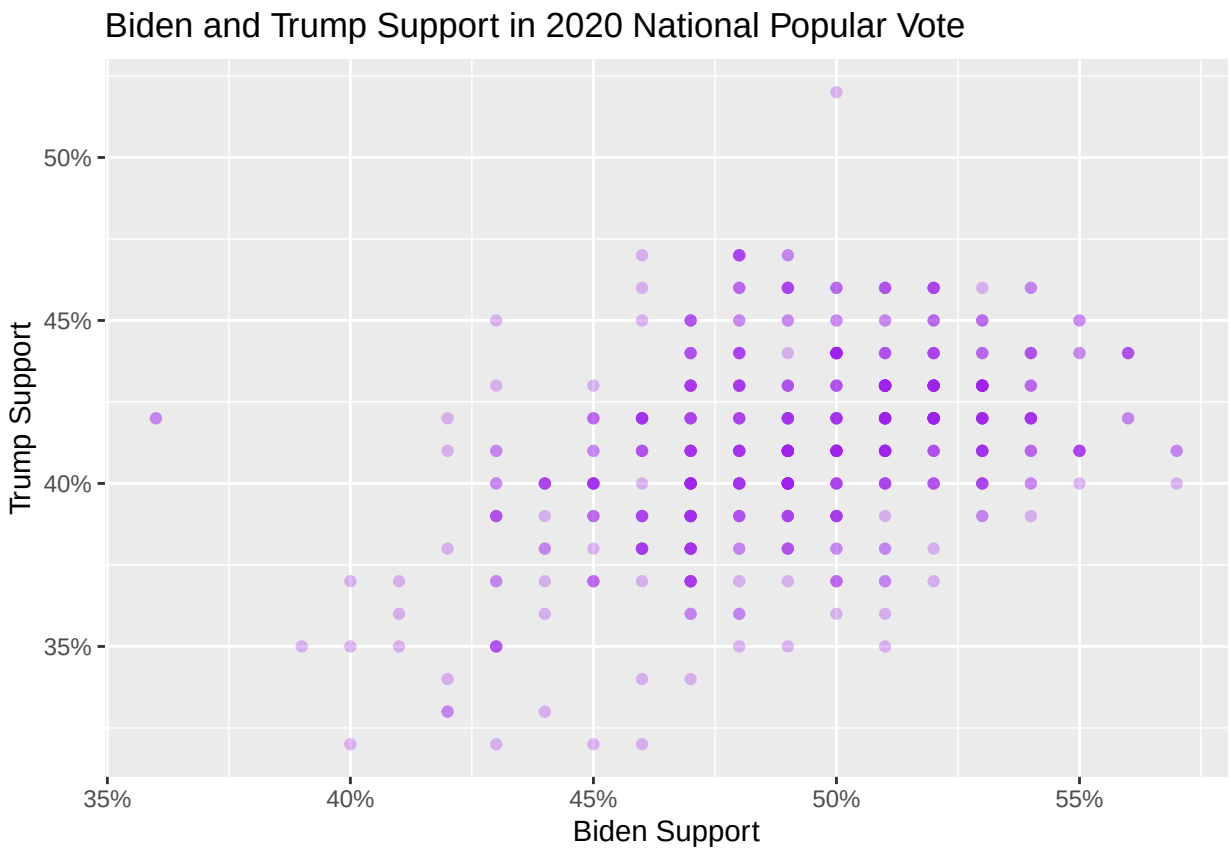


The results are intriguing! First the data seems like it falls along a grid. This is because of how poll results are reported in terms of percentage points and it highlights that even continuous variables may be reported in discrete values. This is consequential because it is hard to know how many polls are associated with each point on the graph. How many polls are at the point (Biden 50%, Trump 45%)? This matters for trying to determine what the relationship might be. Second, it is clear that there are some questions that need to be asked – why doesn't `Biden + Trump = 100\%`?

To try to display how many observations are located at each point we have two tools at our disposal. First, we can alter the "alpha transparency" by setting `alpha-.5` in the `geom_point` call. By setting a low level of transparency, this means that the point will become less transparent as more points occur at the same coordinate. Thus, a faint point indicates that only a single poll (observation) is located at a coordinate whereas a solid point indicates that there are many polls. When we apply this to the scatterplot you can immediately see that most of the polls are located in the neighborhood of Biden 50%, Trump 42%.

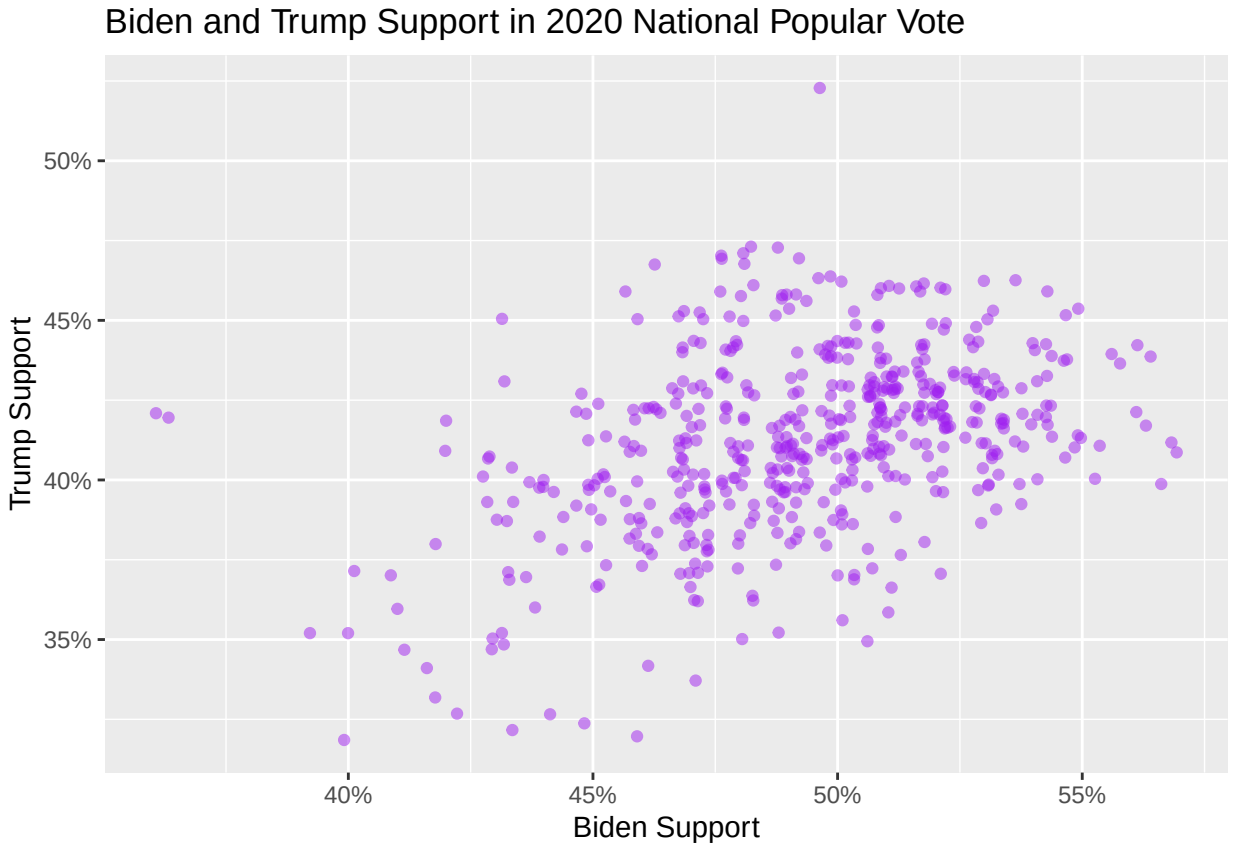
```
Pres2020.PV %>%
  ggplot(aes(x = Biden, y = Trump)) +
  labs(title="Biden and Trump Support in 2020 National Popular Vote",
```

```
      y = "Trump Support",
      x = "Biden Support") +
  geom_point(color="purple",alpha = .3) +
    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))
```



However, the grid-like nature of the plot is still somewhat hard to interpret as it can be hard to discern variations in color gradient. Another tool is to add a tiny bit of randomness to the x and y values associated with each plot. Instead of values being constrained to vary by a full percentage point, for example, the jitter allows it to vary by less. To do so we replace `geom_point` with `geom_jitter`.

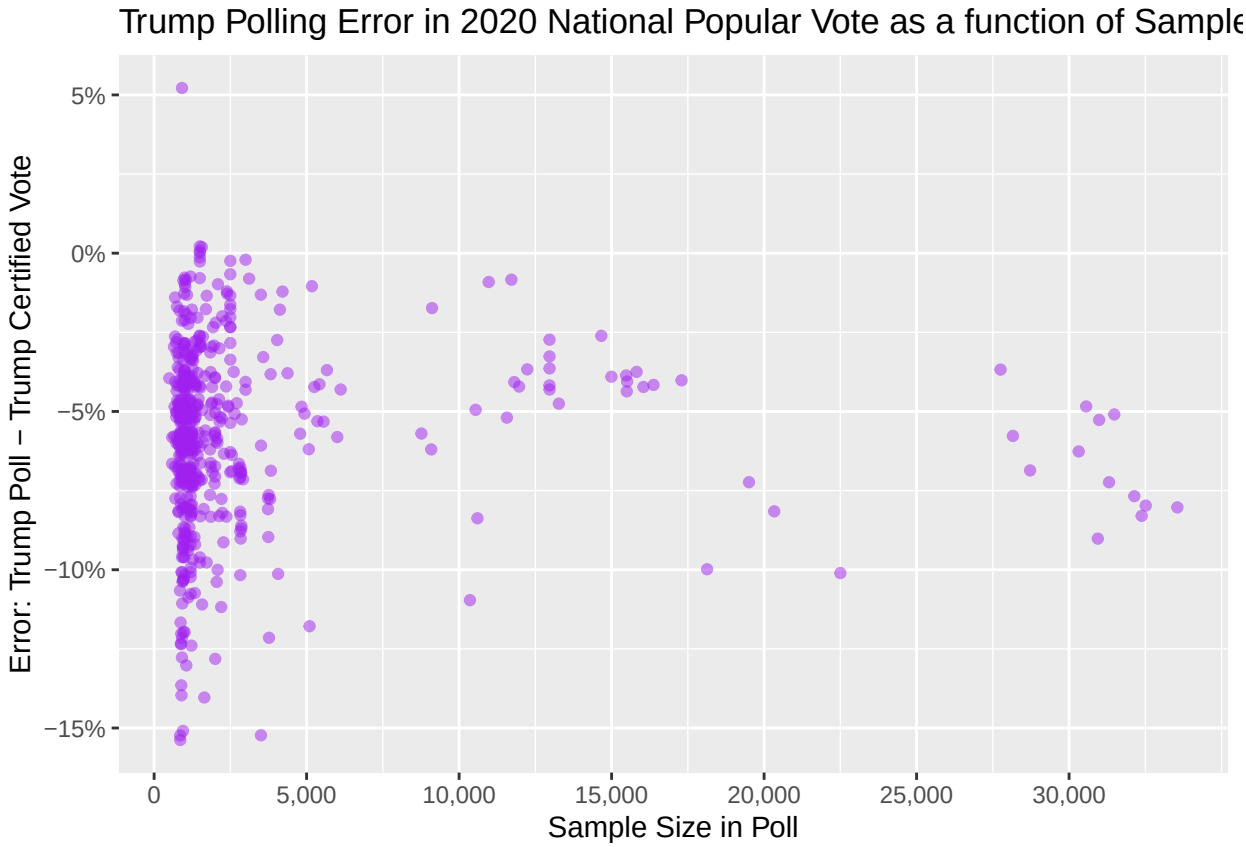
```
Pres2020.PV %>%
  ggplot(aes(x = Biden, y = Trump)) +
  labs(title="Biden and Trump Support in 2020 National Popular Vote",
       y = "Trump Support",
       x = "Biden Support") +
  geom_jitter(color="purple",alpha = .5) +
    scale_y_continuous(breaks=seq(0,1,by=.05),
                       labels=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



Note how much the visualization changes. Whereas before the eye was focused on – and arguably distracted by – the grid-like orientation imposed by the measurement, once we jitter the points we are immediately made aware of the relationship between the two variables. While we are indeed slightly changing our data by adding random noise, the payoff is that the visualization arguably better highlights the nature of the relationship. Insofar the goal of visualization is communication, this trade-off seems worthwhile in this instance. But here again is where data science is sometimes art as much as science. The decision of which visualization to use depends on what you think most effectively communicates the nature of the relationship to the reader.

We can also look at the accuracy of a poll as a function of the sample size. This is also a relationship between two continuous variables – hence a scatterplot! Are polls with more respondents more accurate? There is one poll with nearly 80,000 respondents that we will filter out to me able to show a reasonable scale. Note that we are going to use `labels = scales::comma` when plotting the x-axis to report numbers with commas for readability.

```
Pres2020.PV %>%
  filter(SampleSize < 50000) %>%
  mutate(TrumpError = Trump - RepCertVote/100,
         BidenError = Biden - DemCertVote/100) %>%
  ggplot(aes(x = SampleSize, y = TrumpError)) +
  labs(title="Trump Polling Error in 2020 National Popular Vote as a function of Sample Size",
       y = "Error: Trump Poll - Trump Certified Vote",
       x = "Sample Size in Poll") +
  geom_jitter(color="purple",alpha = .5) +
  scale_y_continuous(breaks=seq(-.2,1,by=.05),
                     labels= scales::percent_format(accuracy = 1)) +
  scale_x_continuous(breaks=seq(0,30000,by=5000),
                     labels= scales::comma)
```



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)

```
load(file="data/Pres2020.PV.Rdata")
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...

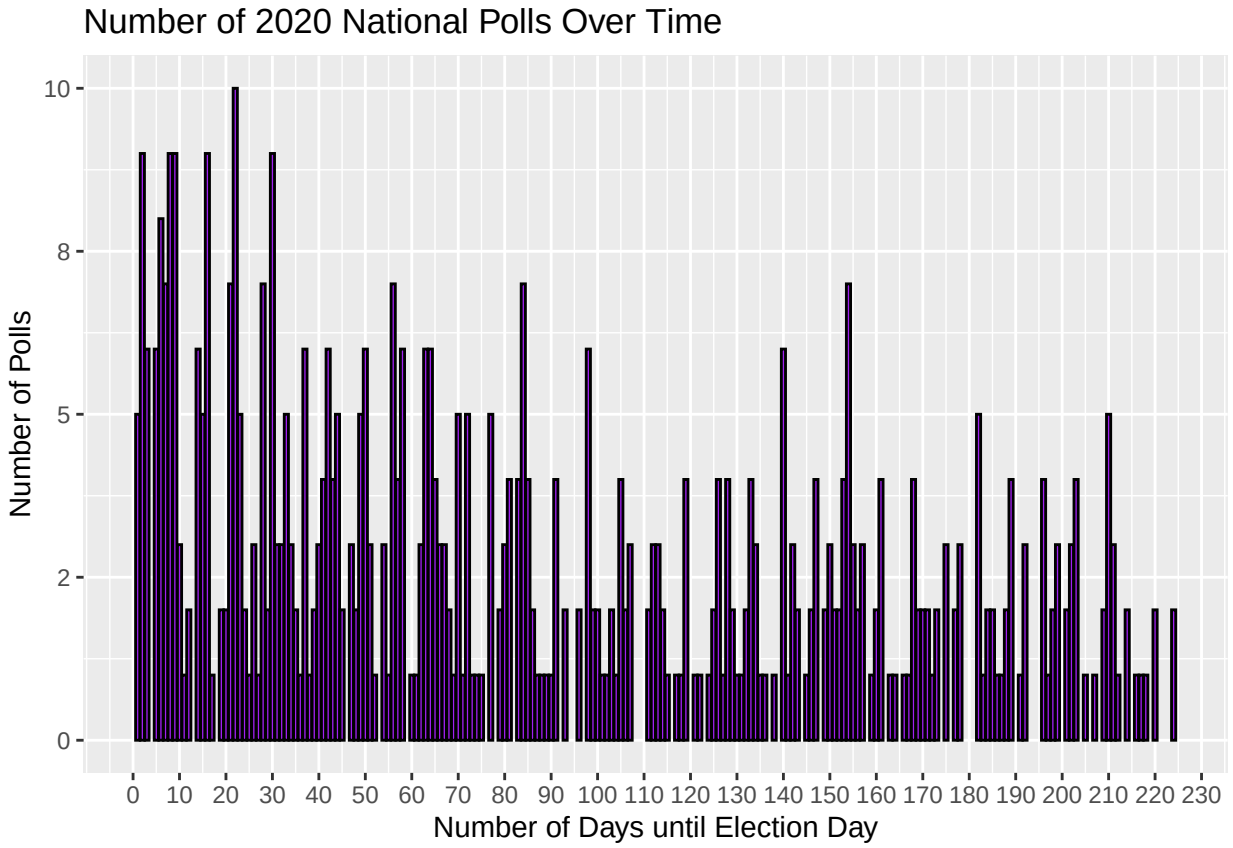
```
Pres2020.PV <- readRDS(file="Pres2020.PV.Rds")

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)?

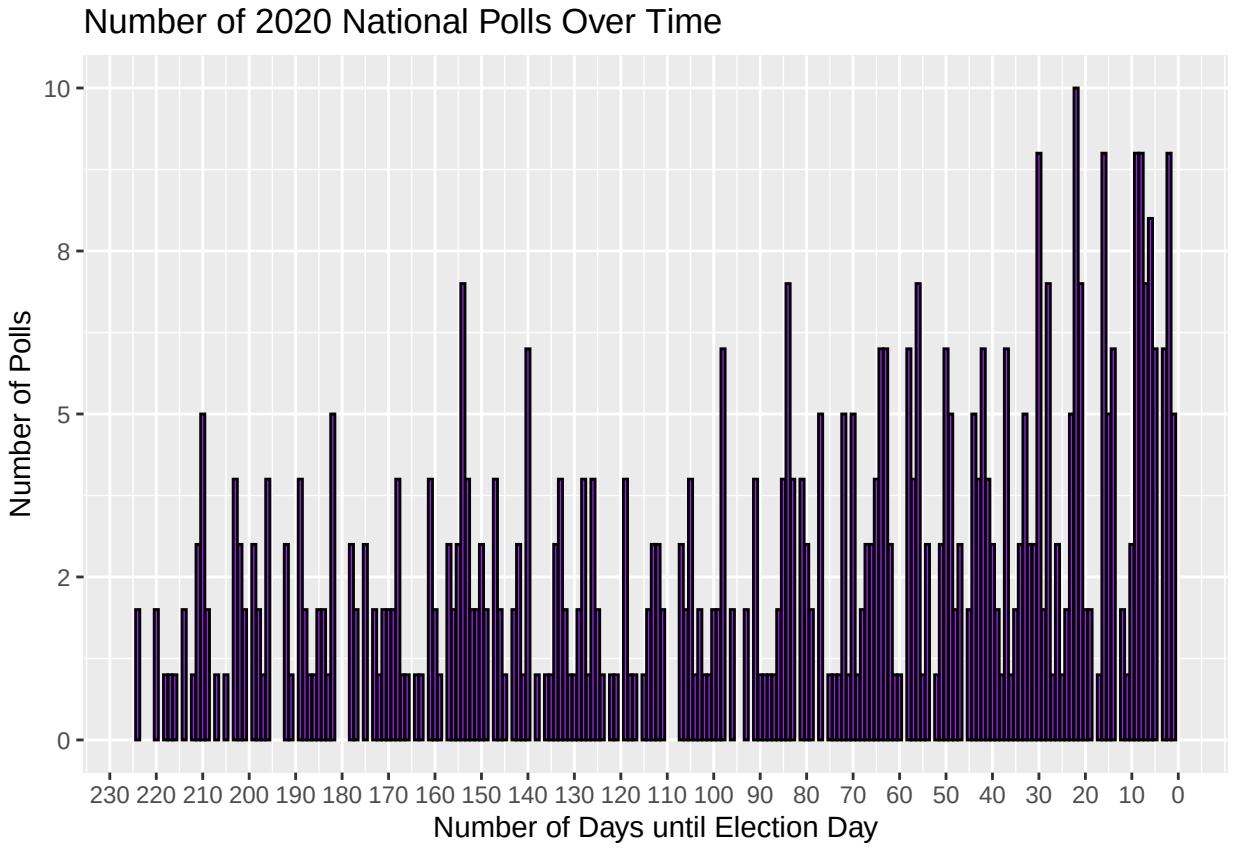
```
ggplot(data = Pres2020.PV, aes(x = DaysToED)) +
  labs(title = "Number of 2020 National Polls Over Time",
       x = "Number of Days until Election Day",
       y = "Number of Polls") +
```

```
  geom_bar(fill="purple", color= "black") +
  scale_x_continuous(breaks=seq(0,230,by=10)) +
  scale_y_continuous(labels = label_number(accuracy = 1))
```



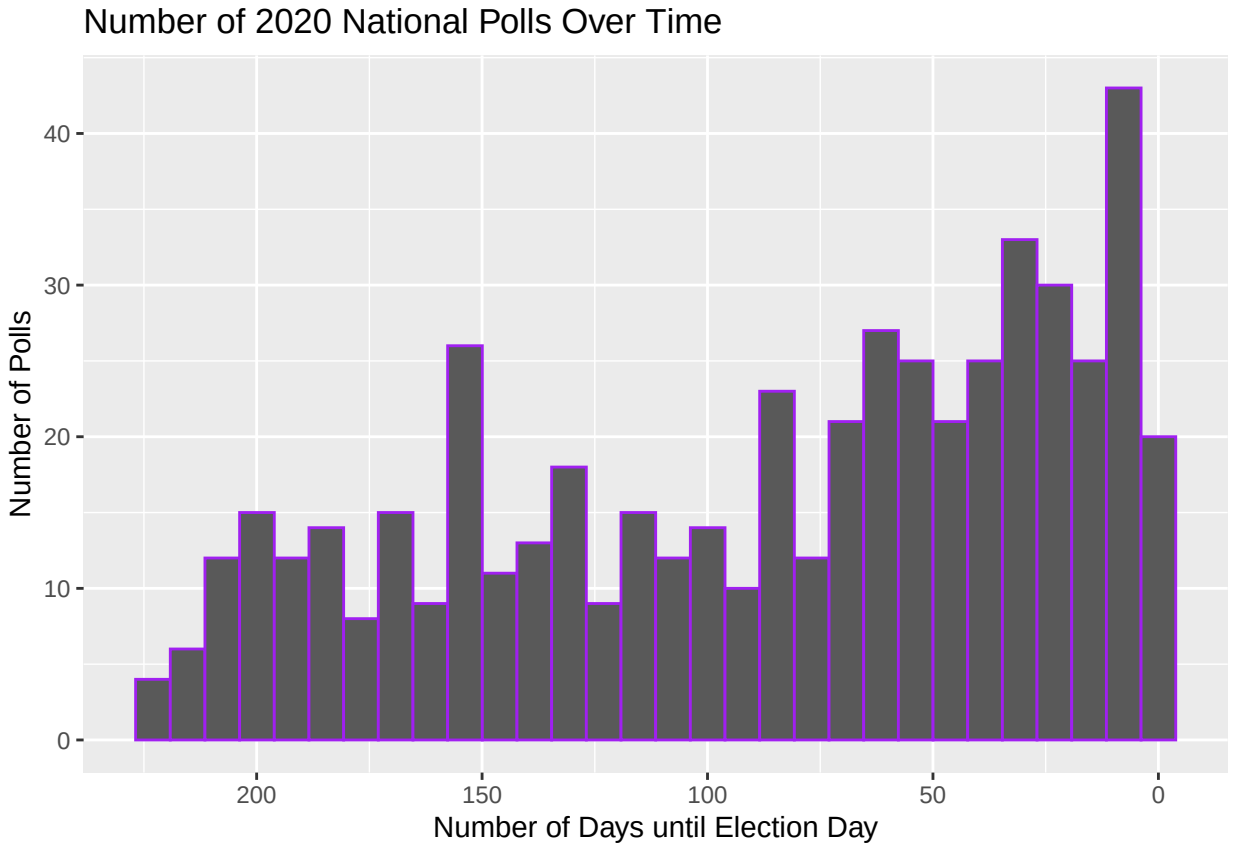
So this is a bit weird because it arranges the axis from smallest to largest even though that is in reverse chronological order. Since November comes after January it may make sense to flip the scale so that the graph plots polls that are closer to Election Day as the reader moves along the scale to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

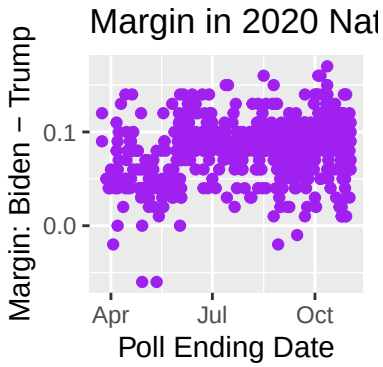
- Most of what we do is a relationship between (at least) 2 variables.
- Here we are interested in how the margin varies as Election Day approaches: `margin` by `DaysToED`.
- Want to plot X (variable that "explains") vs. Y (variable being "explained"):

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.

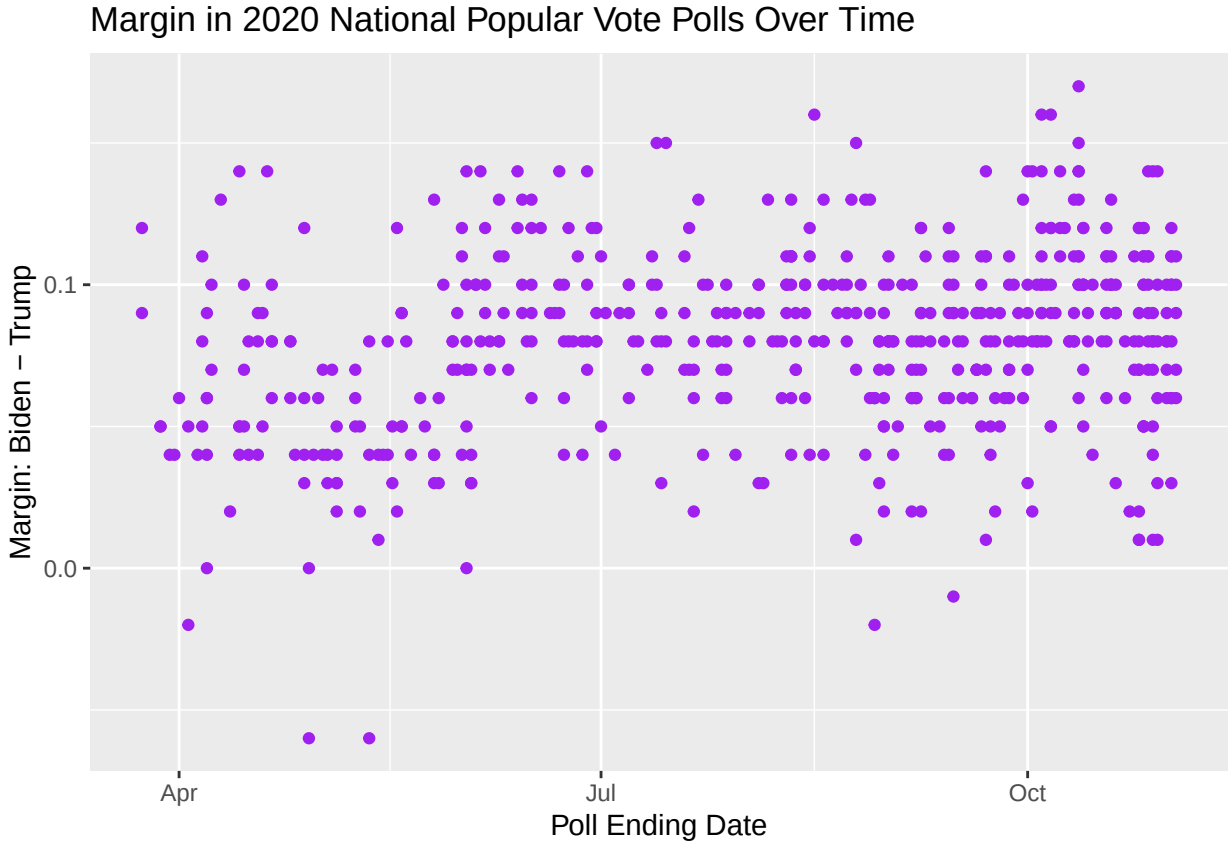
```
margin_over_time_plot <- Pres2020.PV %>%
  ggplot(aes(x = EndDate, y = margin)) +
  labs(title="Margin in 2020 National Popular Vote Polls Over Time",
       y = "Margin: Biden - Trump",
```

```
       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?

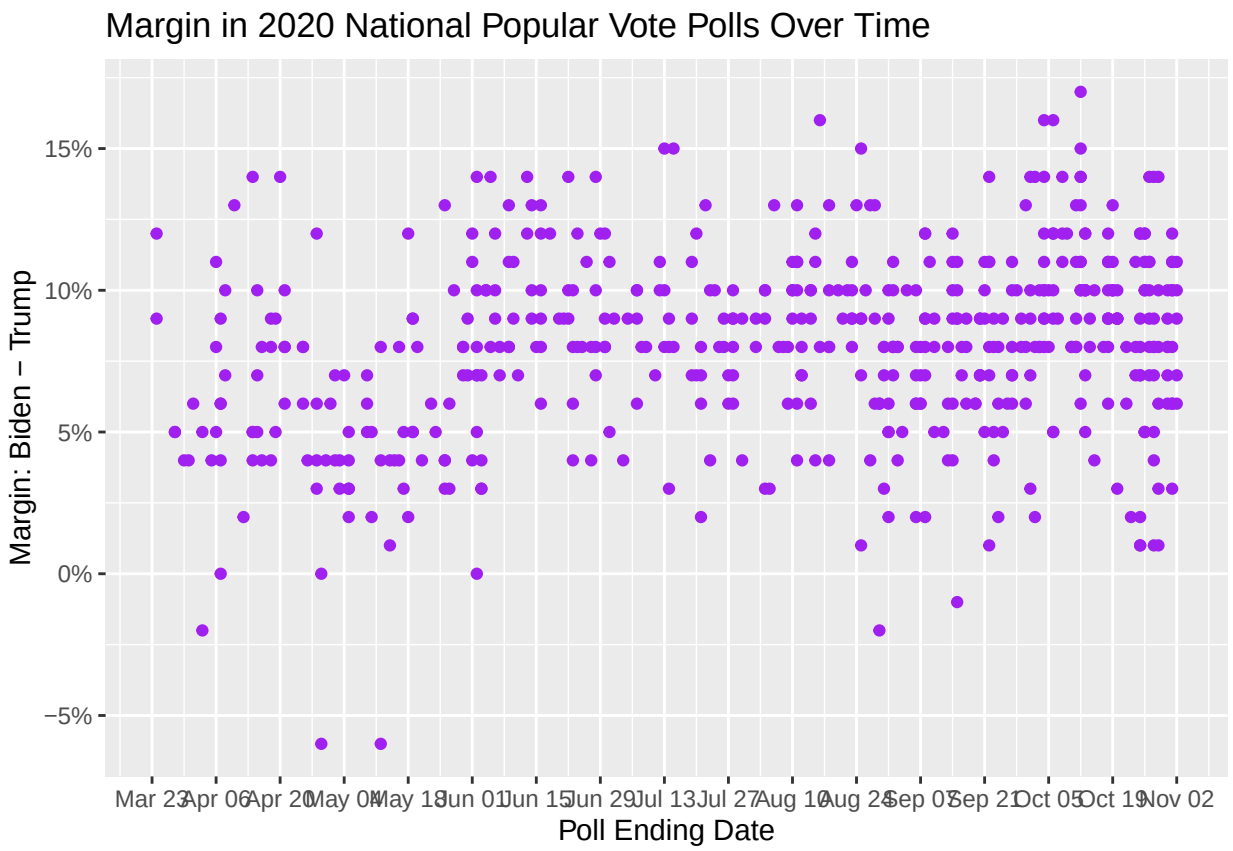
1. Axes looks weird - lots of interpolation required by the consumer.
2. Data looks "chunky"? How many data points are at each point?

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
```

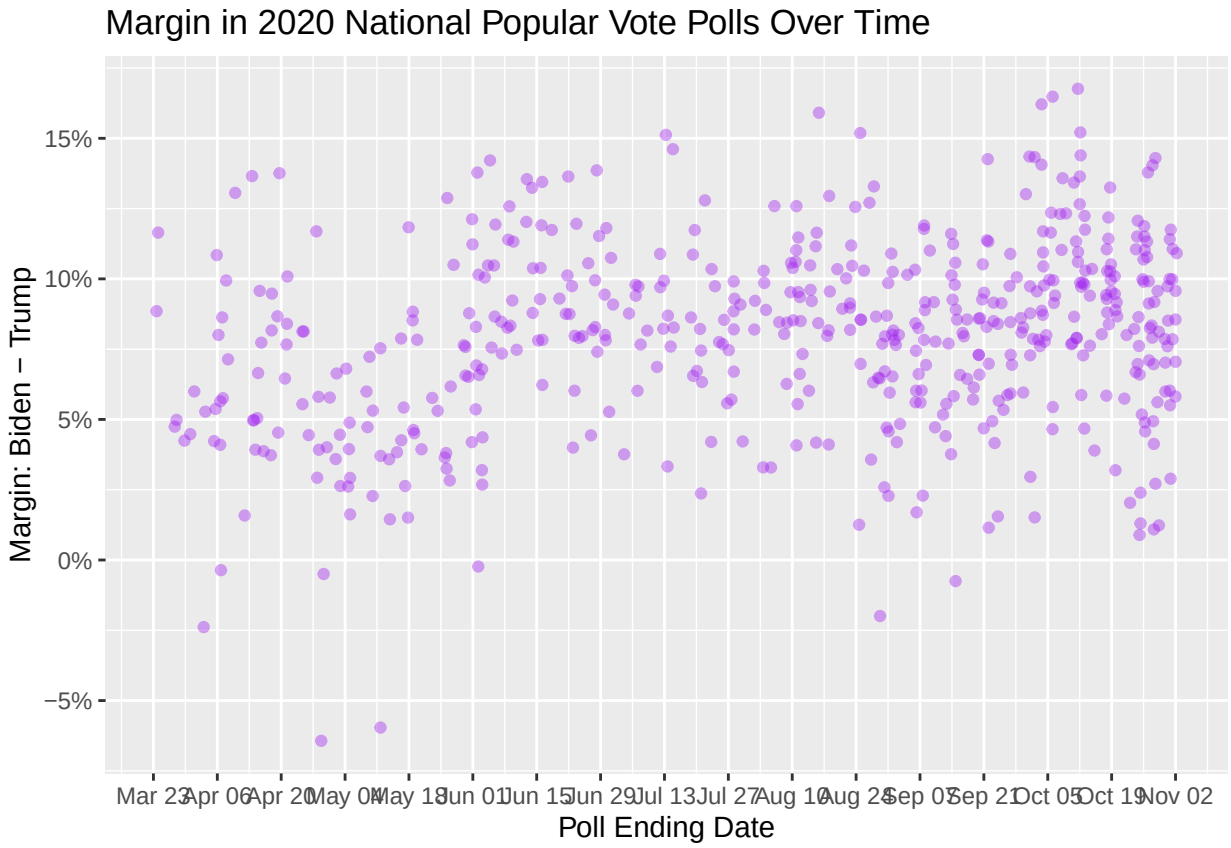


Now one thing that is hard to know is how many polls are at a particular point. If some points contain a single poll and others contain 1000 polls that matters a lot for how we interpret the relationship.

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.

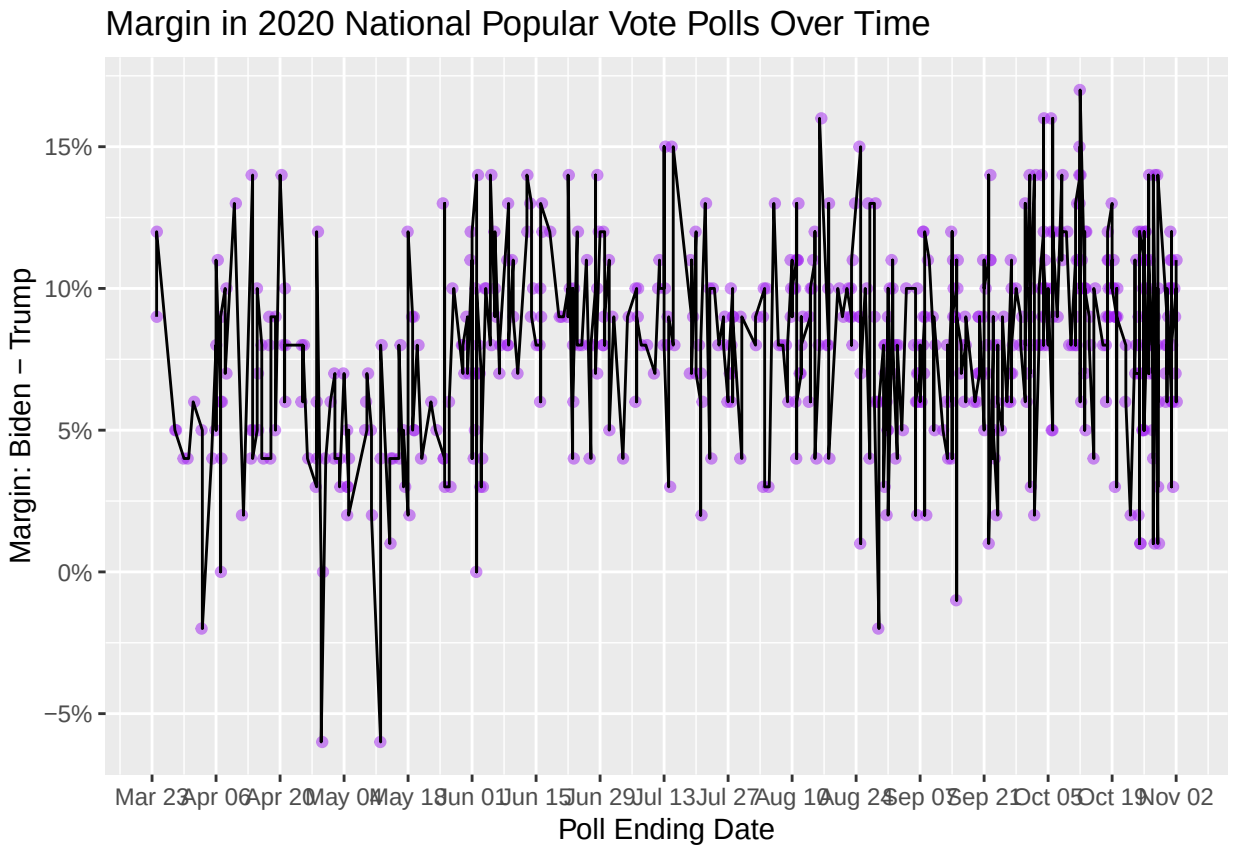
```
Pres2020.PV %>%
  ggplot(aes(x = EndDate, y = margin)) +
  labs(title="Margin in 2020 National Popular Vote Polls Over Time",
       y = "Margin: Biden - Trump",
       x = "Poll Ending Date") +
    geom_jitter(color = "PURPLE",height=.005, alpha = .4) +
```

```
    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")
```



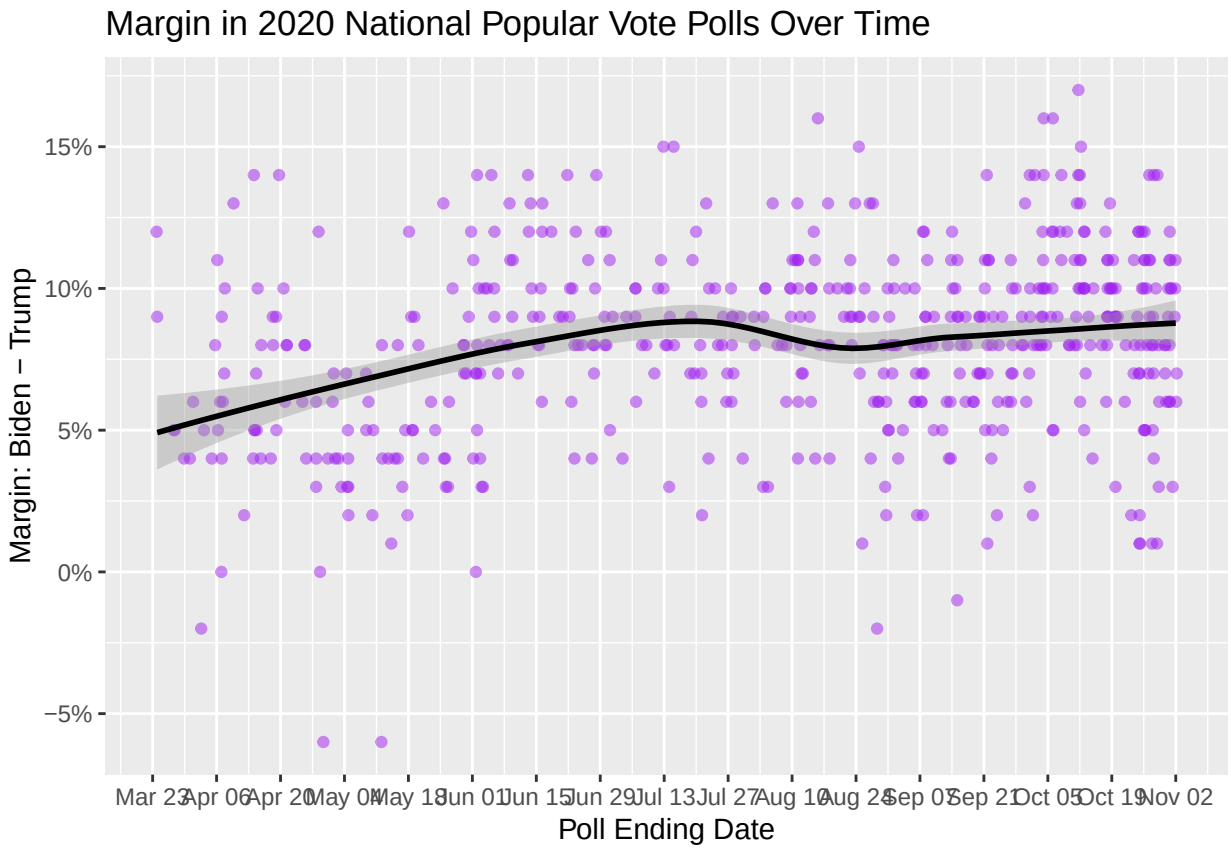
In addition to plotting plotting points using `geom_point` we can also add lines to the plot using `geom_line`. The line will connect the values in sequence so it does not always make sense to include. For example, if we add the `geom_line` to the plot it is hard to make the case that the results are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

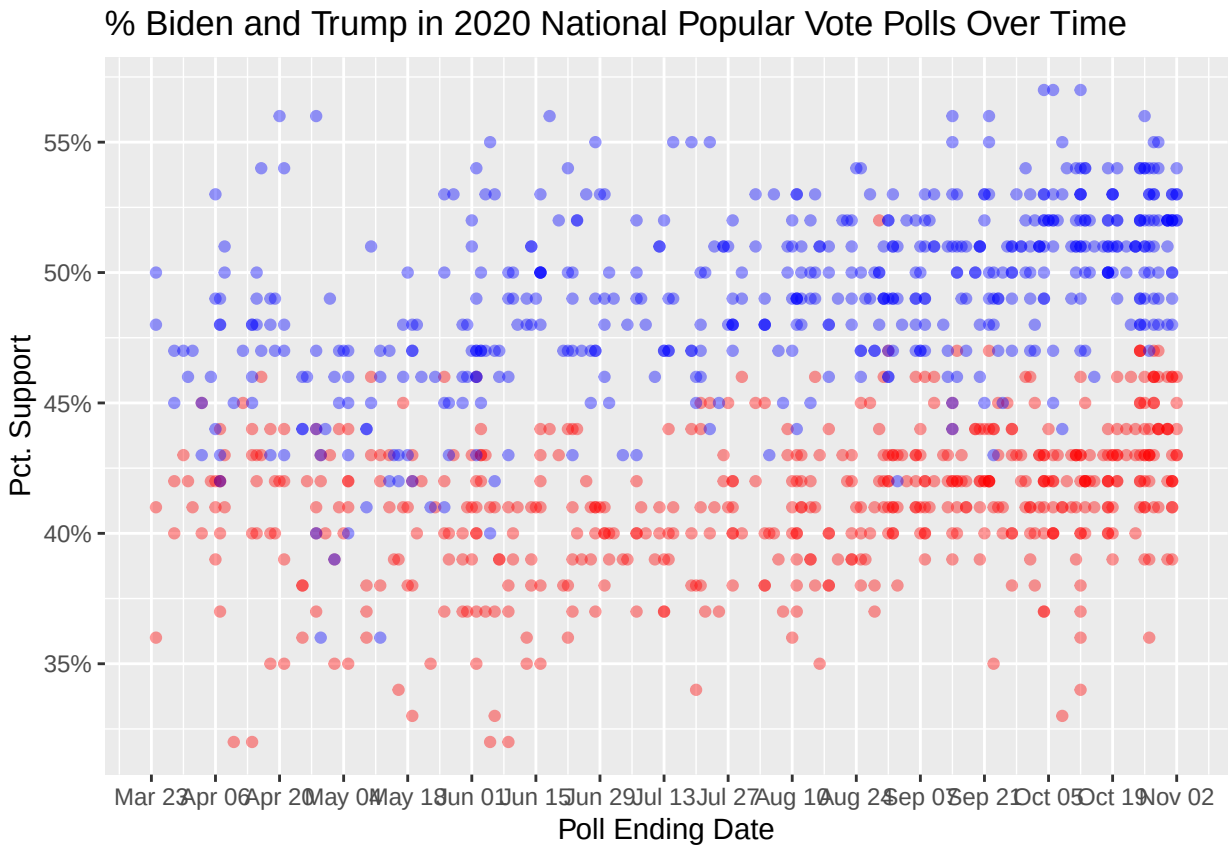
Let's define a `ggplot` object to add to later. Note that this code chuck will not print anything because we need to call the object to see what it produced.

```
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

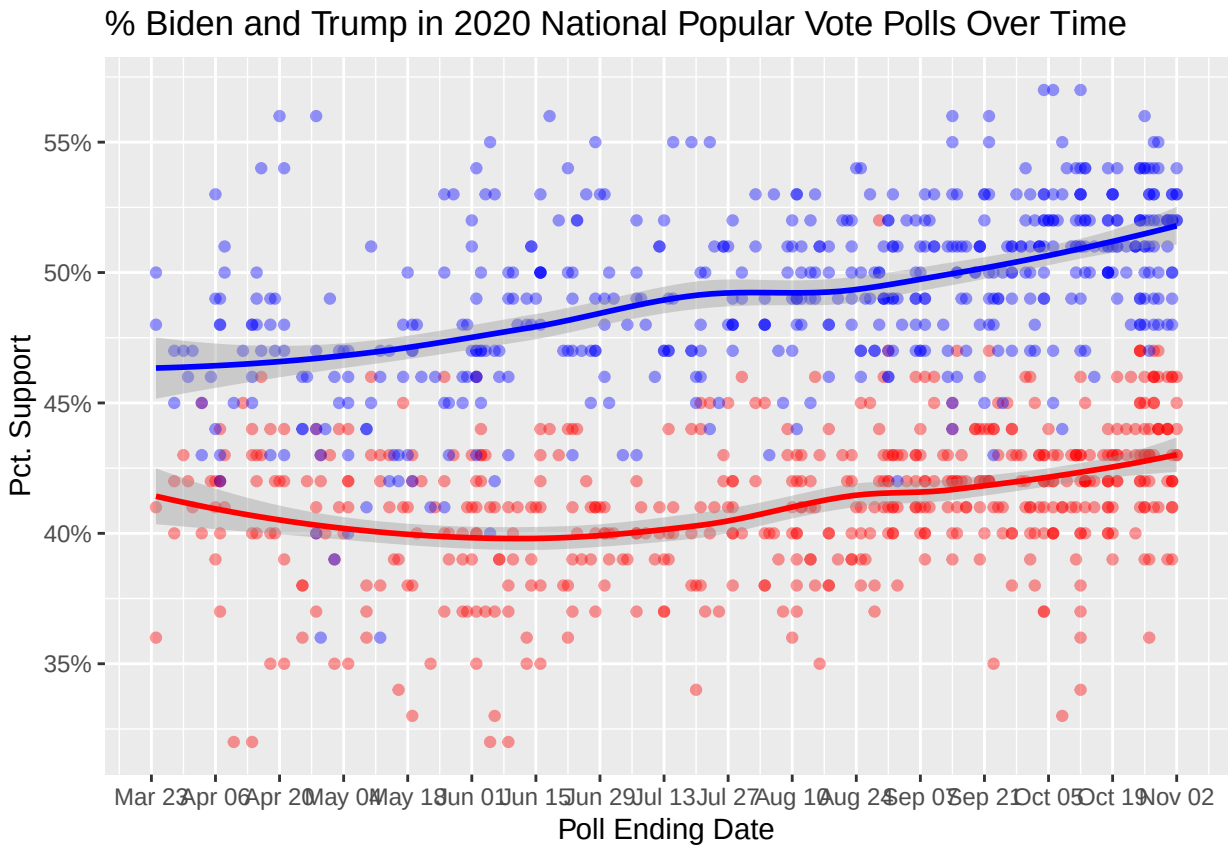
```
BidenTrumpplot
```



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.

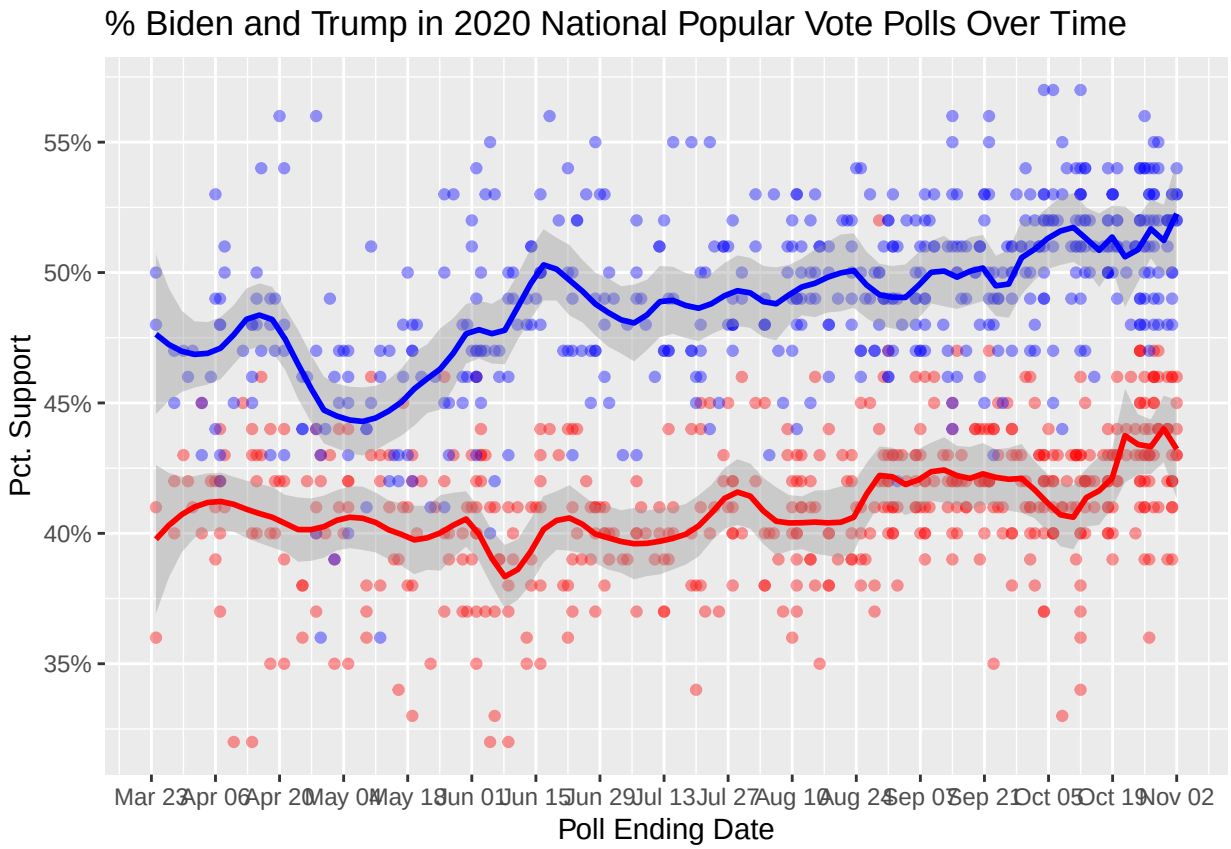
When calling the `geom_smooth` we used `se=T` to tell `ggplot` to produce an estimate of how much the prediction may vary. (This is the 95% confidence interval for the prediction being graphed.)

```
# 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!