Conditional Relationships

Homework

Prof. Bisbee

Due Date: 2024-02-06

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 <- read_rds("https://github.com/jbisbee1/DS1000_S2024/raw/main/data/MI2020_ExitPol
l_small.rds")
MI_final_small <- mi_ep %>%
  filter(preschoice=="Donald Trump, the Republican" | preschoice=="Joe Biden, the Democr
at") %>%
  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 × 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 × 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 × 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 categories 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 × 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 × 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))
```

```
## # A tibble: 22 × 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
## # ℹ 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()
```

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 × 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.

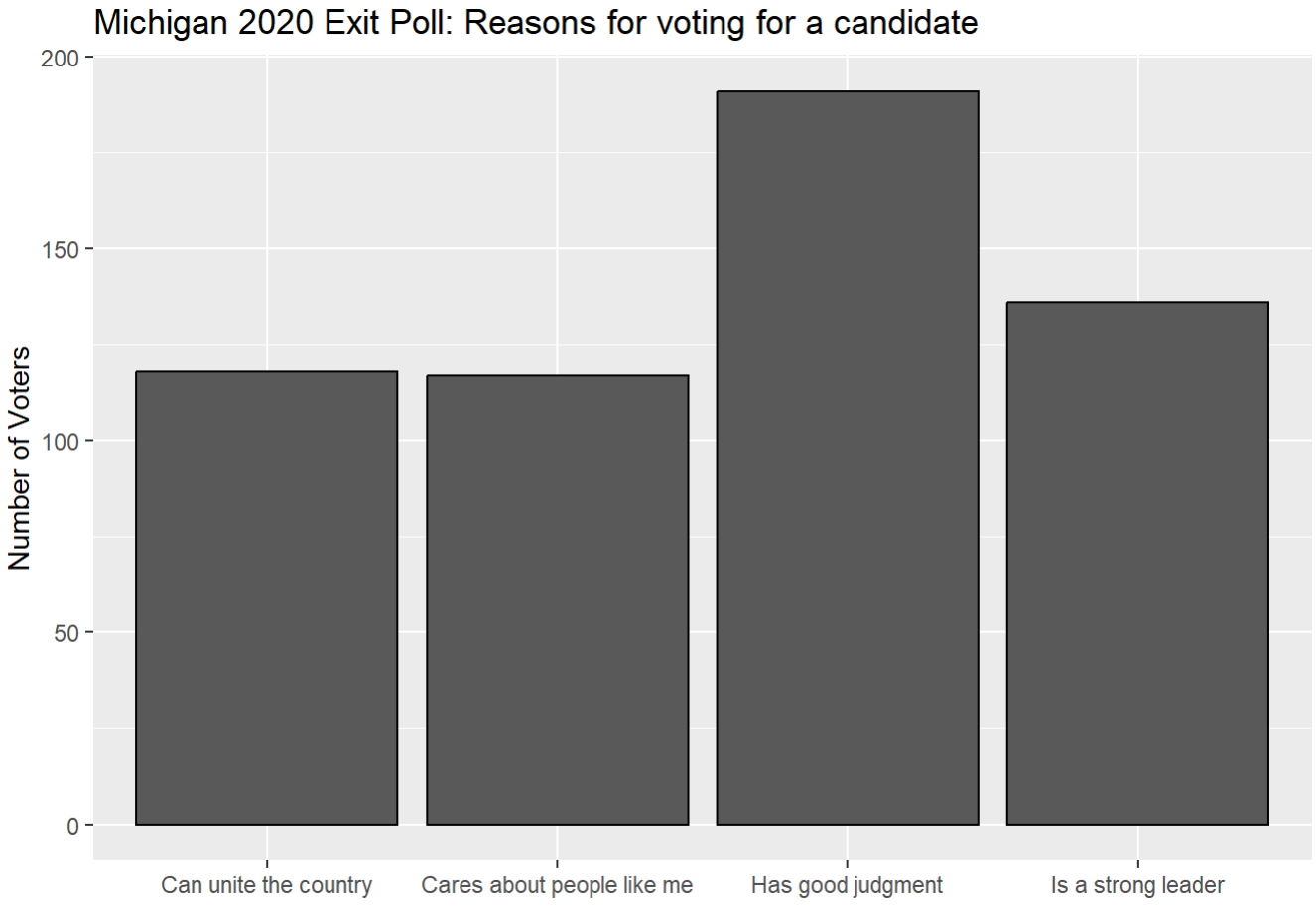
Before we used `geom_bar` to plot the number of observations associated with each value of a variable. But we often want to know how the number of observations may vary according to a second variable. For example, we care not only about why voters reported that they supported Biden or Trump in 2020 but 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.

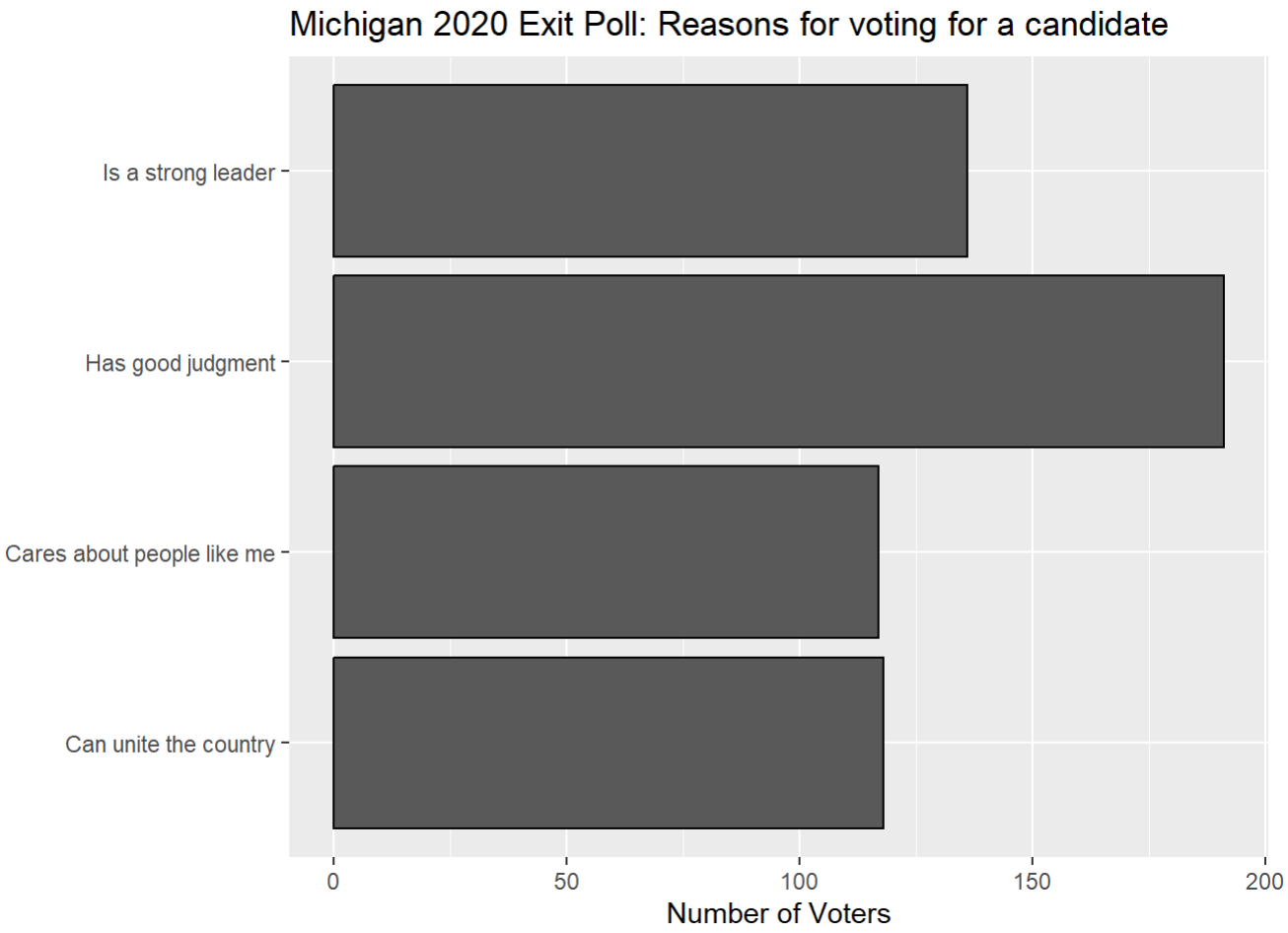
Now we include labels – note how we are suppressing the x-label because the value labels are self-explanatory in this instance and add the `geom_bar` as before.

```
mi_ep %>%
    select(Quality,preschoice) %>%
    filter(preschoice == "Joe Biden, the Democrat" | preschoice == "Donald Trump, the Re
publican") %>%
    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.)

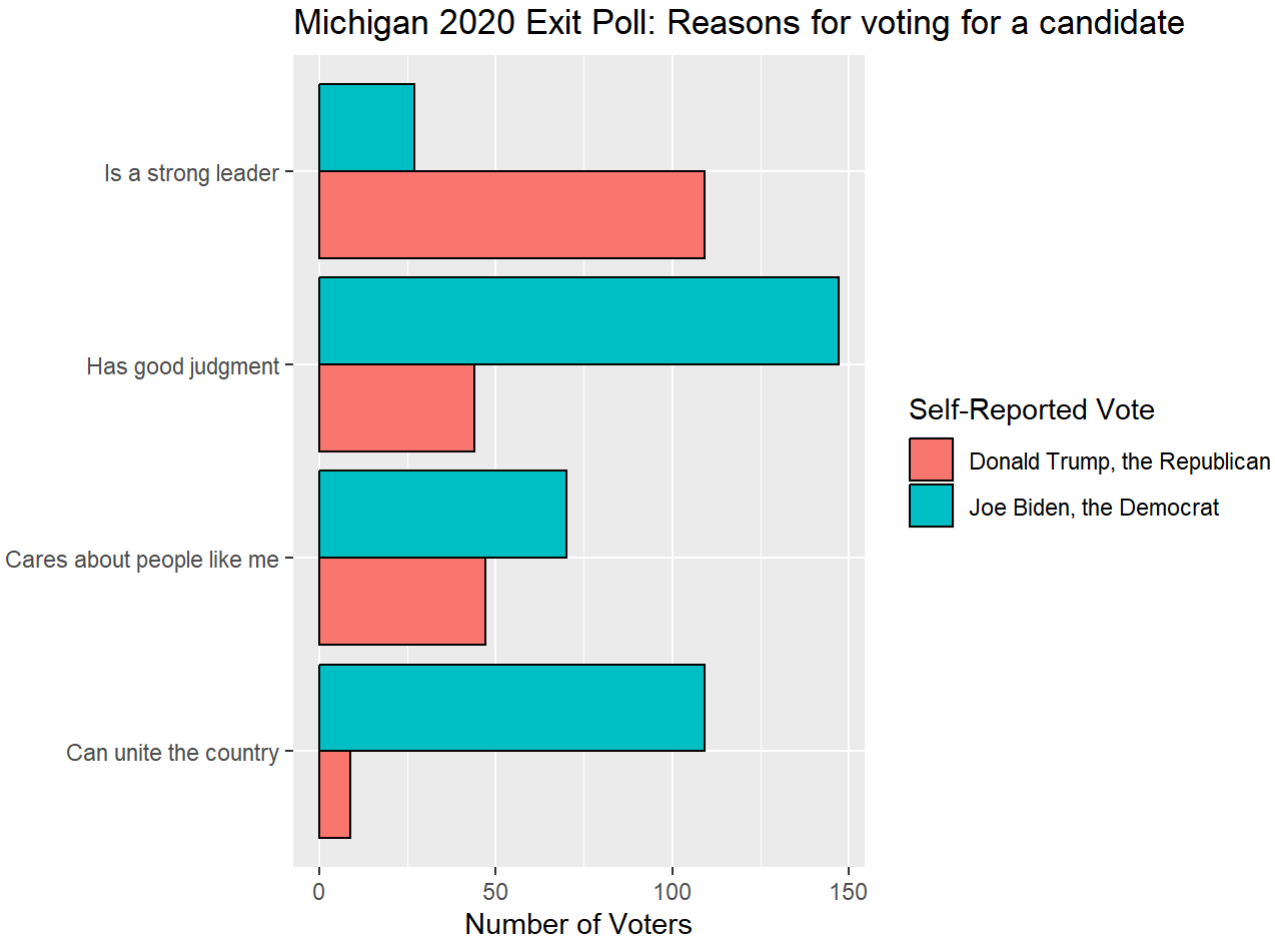
```
mi_ep %>%
    select(Quality,preschoice) %>%
    filter(preschoice == "Joe Biden, the Democrat" | preschoice == "Donald Trump, the Re
publican") %>%
    drop_na() %>%
    filter(Quality != "[DON'T READ] Don’t know/refused") %>%
    ggplot(aes(x= Quality)) +
    labs(y = "Number of Voters",
         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.

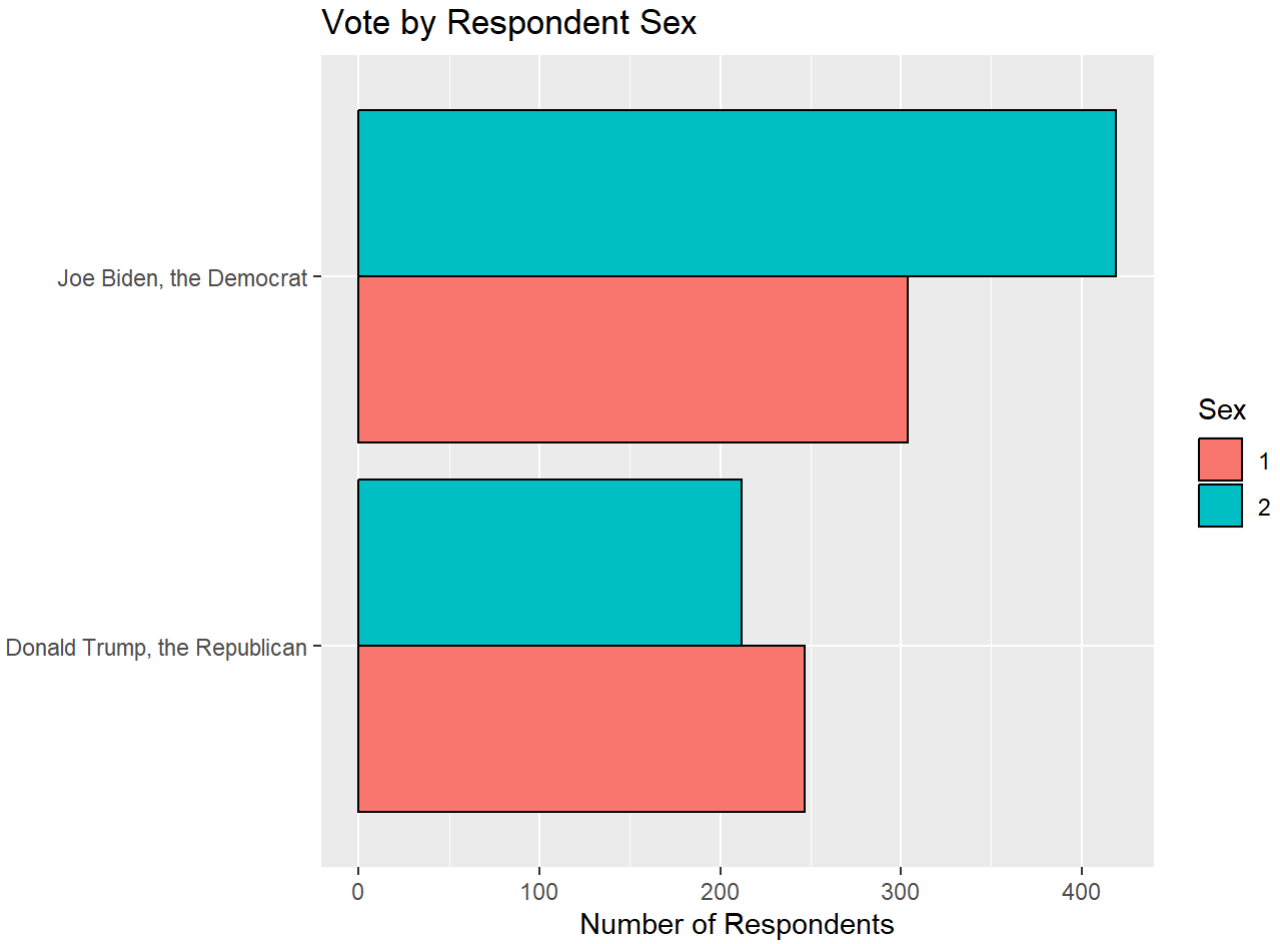
```
mi_ep %>%
    select(Quality,preschoice) %>%
    filter(preschoice == "Joe Biden, the Democrat" | preschoice == "Donald Trump, the Re
publican") %>%
    drop_na() %>%
    filter(Quality != "[DON'T READ] Don’t know/refused") %>%
    ggplot(aes(x= Quality, fill = preschoice)) +
    labs(y = "Number 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
```