SPEC Lab REU R Resources: Data visualization with `ggplot2`: An Introduction to Barplots

Jasmine Chu, Gaea Morales, and Benjamin Graham

Summer 2022

ggplot2 continued

This is part 4 of the Data Visualization 2 walk-through-work, introduction to bar plots.

In this part of the module, we will work with survey data from the Lewis Registry (see thelewisregistry.org), using `Lewis_support_survey.csv`. The Lewis Registry is a national database that collects information on police officers separated or terminated due to serious misconduct. That way, these officers don't end up getting rehired by another department. In this module, we are working with a data set from a survey conducted in Fall 2020 to assess public support for creation of a registry of this kind.

Intro to bar plots

Bar plots are primarily used to show comparisons across individuals or groups on a single dimension, such as comparing average salary across workers with different levels of education, or comparing average GDP per capita across different regions of the world. Bar plots can be easily produced using the `ggplot2` package, but the data management needed to create a bar plot can sometimes be a bit tricky, especially if we are comparing across groups (comparing across individuals is easier).

Let's first load in the libraries necessary for this exercise as well as the data from the Lewis Registry.

```
rm(list=ls())
library(ggplot2)
library(dplyr)
library(scales)

lewis <- read.csv("Lewis_support_survey.csv") #load dataset
```

Stat Count

Let's say that we want to produce a bar plot that displays the number of respondents support/oppose the creation of a national registry of police officers fired for misconduct. In the data, column `usc_omni_102020` is a binary variable, with responses "yes" or "no" when asked if they would support the creation of a national registry of police officers who have been terminated/separated due to serious misconduct. How can we use `ggplot2` to create a simple bar plot?

As we briefly learned in Data Visualization 1 Module, bar graphs can easily be made by including three main components to ggplot: (1) data, (2) aesthetic mapping, and (3) geometry.

Here, we know that we want to use the `lewis` dataframe as our data. Second, we know that for our aesthetic mapping, our x variable will be the various responses of Yes's and No's from the the column, `usc_omni_102020`. On the y axis, we want to see the total count of those responses. Lastly, we know that we want to make a bar plot, so the geometry of our graph will be `geom_bar()`.

In this case, we would use `stat = "count"` as the argument within `geom_bar()`. If we use `stat = "count"`, then we are simply counting up the number of observations that take on each value of the X variable. Each discrete value of that variable will get its own bar, and the height of each bar will represent the number of observations that take that particular value of the x variable – so the number of people who answer a question in a particular way.

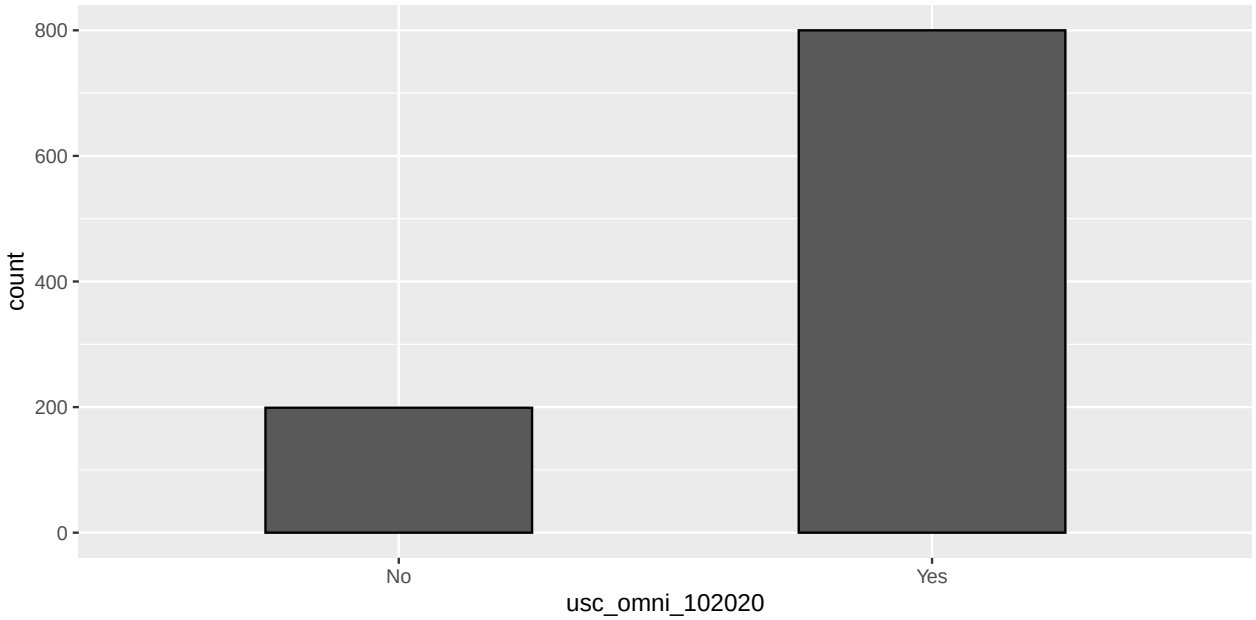
Before we begin making our bar plot, let's use the `filter()` command from the `dplyr` package, since we want to exclude the "Skipped" responses and are only interested in the Yes's and No's.

```
lewis <- lewis %>%
  filter(usc_omni_102020!= "skipped") # removing non-responses
```

Great! We are now ready to produce our first bar plot.

```
stat_count <- ggplot(data = lewis, aes(x = usc_omni_102020)) +
  geom_bar(stat = "count", width = 0.5, colour = "Black") # stat = "count"
        # specific that we want a bar plot counting the number of respondents for each
        # unique response

stat_count
```



Perfect! We just created a simple bar plot that displays respondents' views regarding the creation of a national registry. As we can see in our bar graph above, there are about 800 people in support while there are 200 folks who oppose the creation of a national database. Of course, the bar plot could be a bit better in giving more meaningful information about the respondents. Let's make some minor changes to our plot.

Differentiating by other variables

Let's say we want to group those responses by the political affiliation of the respondent. How can we modify our bar plot to incorporate these changes? Well, we know that column `pid3` in our `lewis` dataset

is a categorical variable that identifies the police officer's political affiliation, ranging from "Democrat" and "Republican" to "Independent" and "Other." Since we want to group the police officers' responses by political affiliation, we can do so by including `fill=pid3` within the aesthetic argument.

We also want to see two clusters of side-by-side bars of "Yes" and "No" responses. We can do so by including `position='dodge'` within `geom_bar()`. If we we were to not include `position='dodge'` in our code, our bar plot would simply default to stacking the bar charts vertically, which is harder to read. For now, let's say we are not interested in stacking our bars, but rather, interested in clustering them side-by-side. `position='dodge'` will do the trick.

In addition, let's customize the colors for our bars, so that the bar for Democrats is blue and the bar for Republicans is red. That makes the graph a bit more intuitive for readers. You can assign colors by including an argument for `scale_fill_manual(values=c())`.

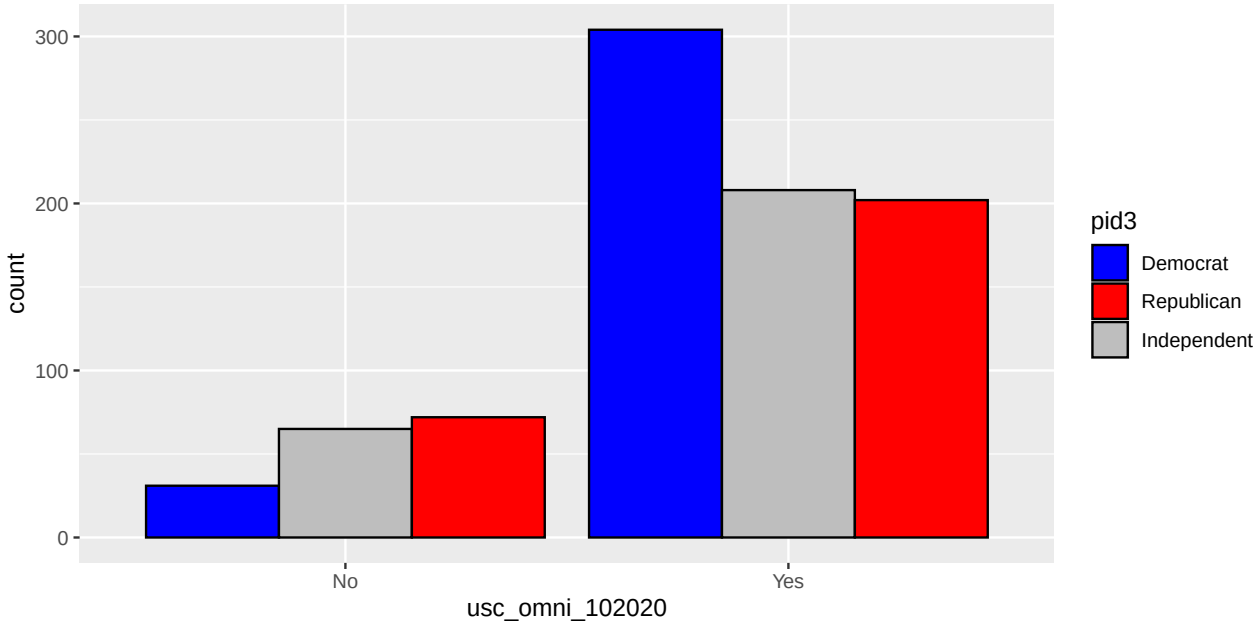
Before we begin making our second bar plot, let's use the `filter()` command from the `dplyr` package, since we are mostly interested in the responses from Democrats, Republicans, and independents, and therefore want to exclude those that are "Not Sure" and "Other."

```
lewis <- lewis %>%
  filter(pid3!= "Other" & pid3!="Not sure")
```

Great! We're now ready to produce our second bar plot.

```
stat_count2 <- ggplot(data=lewis,aes(x=usc_omni_102020, fill=pid3)) + #group fill by
    # political id
  geom_bar(stat = "count", colour = "Black", position='dodge') +
  scale_fill_manual(values = c("Democrat" = "blue",
                               "Republican" = "red",
                               "Independent" = "gray")) # manually choose colors

stat_count2
```



Amazing! We now have a bar plot that displays the distribution of support for the creation of a national registry, clustered by political affiliation. Notice how the bar plot has a legend that shows which colors correspond to what political affiliation. Looking at the bar graph above, it looks like about 300 Democrats

and 200 Republicans were in favor of the creation of a national database, while about 30 Democrats and 70 Republicans opposed the creation of a national registry.

Stat Count vs. Stat Identity

In our bar plot earlier, we used `stat = "count"` within the `geom_bar()` function. However, there is another condition you could have used instead: `stat = "identity"`. What is the difference between these two?

`stat = "count"` is used when you want to count up the number of observations that take on each value of the X variable. Note that when we are using `stat = "count"`, we do not have to include a y-variable, because the y-variable is the count of the x-variable. Furthermore, the default setting for geom_bar() is actually `stat = "count"`. In other words, if you left geom_bar() blank, R will use `stat = "count"` as the default argument.

On the other hand, `stat = "identity"` is used within the `geom_bar()` function when you want to customize your bar plot by specifying what dependent (Y) variable you want for your plot. When using `stat = "identity"`, R automatically overrides the default count value and instead uses the y-variable provided in the argument. As a result, `stat = "identity"` requires the coder to include a Y variable for the bar plot.

Setting up our data

Let's say that now we are interested in producing a figure that displays the percentage support for the creation of a national registry of police officers fired for misconduct, grouped by political affiliation of the respondent. As we discussed earlier, column `usc_omni_102020` is a binary variable that contains the various responses when folks are asked if they would support the creation of a national registry of police officers who have been terminated/separated due to serious misconduct, and column `pid3` is a categorical variable that identifies the respondent's political affiliation. What we can do is produce a bar graph by using these two columns from our data by using `stat = "identity"` as the argument for `geom_bar()`.

Let's first make a few changes to our data by calculating the percentage support by political affiliation. Using `dplyr`, for each political affiliation we can take the number people who approved the creation of the national registry and divide that by the total number of people with that political affiliation.

```
  #First let's figure out what share of all respondents support creation of a registry
lewis_party_support_total <- lewis %>%
  group_by(usc_omni_102020) %>%
  summarise(Per_Support = n()/1000) %>%  # creates a new variable that is the number of
   # "yes" votes over the total number of observations (1000)
  filter(usc_omni_102020 == "Yes") %>%   # we only want to the percent supporting
  rename(party_affiliations = usc_omni_102020)
lewis_party_support_total[1,1] <- "Overall Sample" # replaces "Yes" with Overall Sample

#now we do this by party ID
lewis_party_support_party <- lewis %>%
  group_by(pid3) %>% # grouping by partisanship
  summarise(Per_Support = sum(usc_omni_102020 %in% "Yes")/n()) %>%  # creates a new variable
    # that is the number of "yes" votes over the total number of individuals for each party
    # identification
  rename(party_affiliations = pid3) #renaming column name
```

The last step before we make our bar plot is to merge the two dataframes into one so that we have one dataframe that contains the percentage of support in the population overall, and the percentage of support

for each different party ID. Because these two mini-datasets have exactly the same columns, we can stick these two mini-datasets together using the `rbind()` function. This stacks our two little mini datasets on top of each other (i.e. binds them by row).

```
# using rbind to merge the proportions for total and party samples
lewis_party_support <- rbind(lewis_party_support_total, lewis_party_support_party)
```

Stat Identity

Now, we finally have our neatly, organized data to set up our bar graph. As we've already recapped in our earlier exercise, producing a bar graph has three main requirements in ggplot: (1) data, (2) aesthetic mapping, and (3) geometry.

We know that we want to use the `lewis_party_support` dataframe as our data, which we created just now. Second, we know that for our aesthetic mapping, our x variable will be the party affiliation, with the dependent variable (y) being the percentage of support. We also know that we want our bars to be colored according to party affiliations, so we include `fill=party_affiliations` within the aesthetic argument. Lastly, we know that we want to make a bar plot, so the geometry of our graph will be `geom_bar()`.

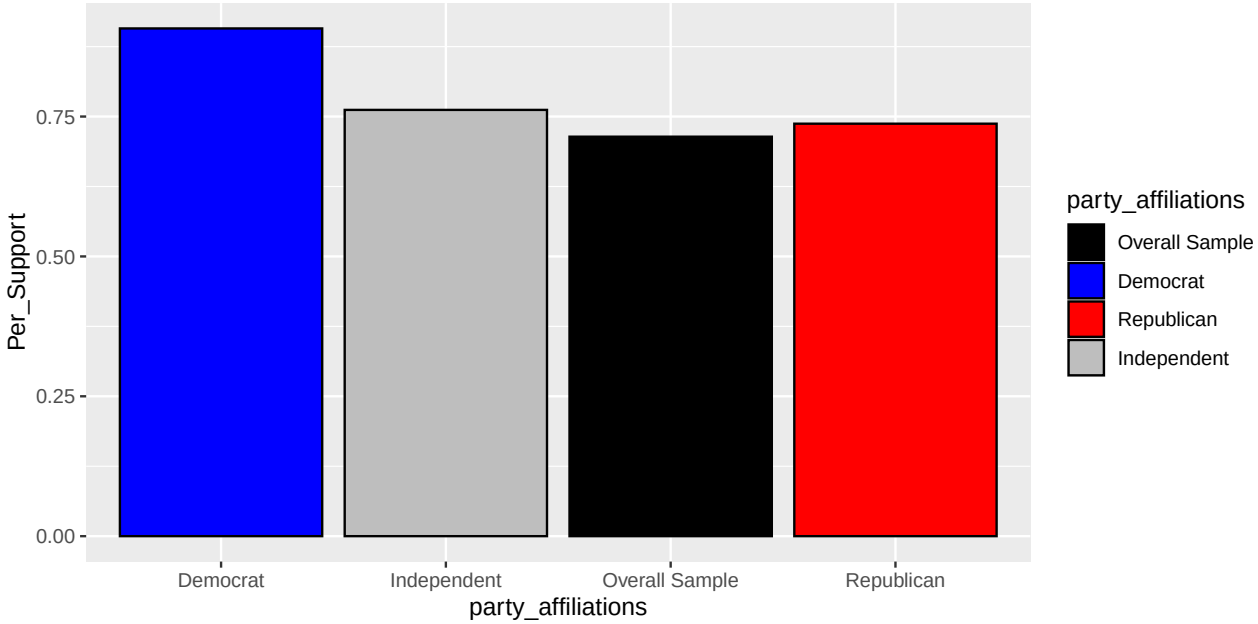
Instead of using `stat = "count"`, let's use `stat = "identity"`, since we no longer want to count up the number of observations for the X variable. Instead, we want to customize our bar plot by setting the Y variable as the proportion of support for the national registry (i.e. `Per_support` in `aes()`).

Once again, let's customize the colors of our bar chart, so that Democrats are blue, Republicans are red, and Independents are gray.

Let's integrate those three components into our bar graph:

```
stat_identity <- ggplot(data=lewis_party_support, aes(x=party_affiliations,y=Per_Support,
                                                      fill=party_affiliations)) +
  geom_bar(stat = "identity", colour = "Black") +
  scale_fill_manual(values = c("Overall Sample" = "black",
                               "Democrat" = "blue",
                               "Republican" = "red",
                               "Independent" = "gray")) #manually choose colors

stat_identity
```



Hmmm. It seems like this bar graph does the job, but there seem to be a few flaws.

For one, the bar plot seems to be missing a few important things, such as a title and/or subtitle. The bar plot could also benefit from a better-labeled legend and correct labeling of units in the Y axis (percentages should be in % as opposed to decimals). And it looks like political affiliations are double-labeled: once on the bottom of the x-axis, and another on the legend located on the right of the bar plot. We can easily get rid of the legend, since having it is redundant and doesn't add much meaning to our bar plot.

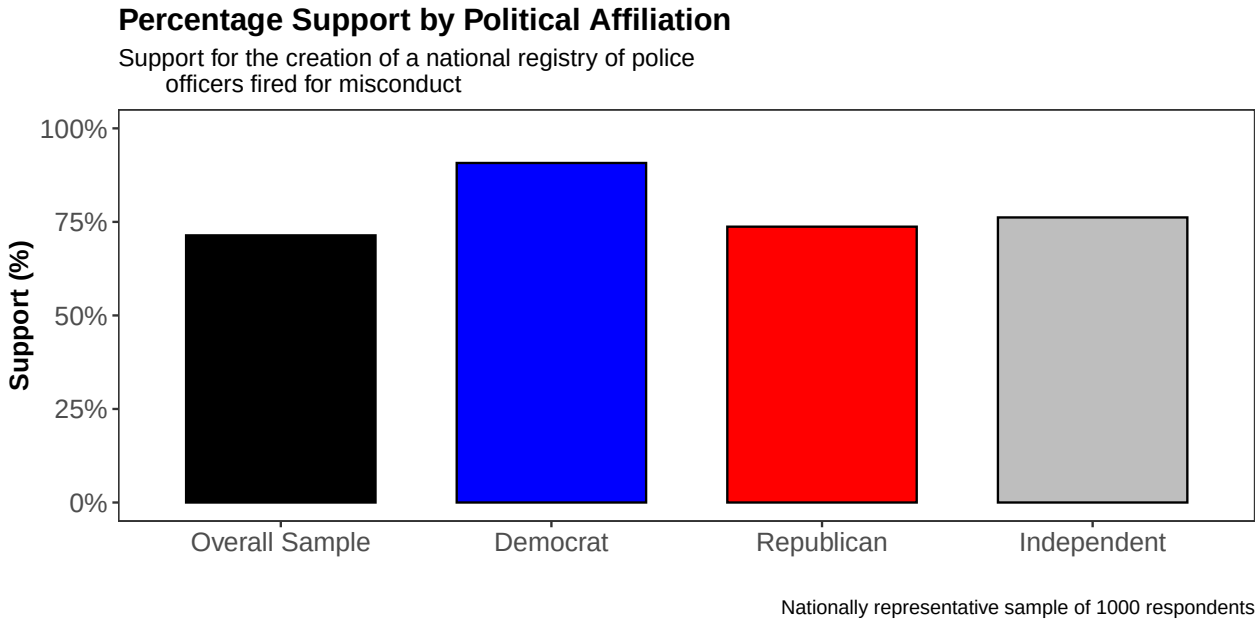
Lastly, the font in all of our labels are barely legible! Our plot would really benefit from a larger, more readable font.

The great thing about `ggplot2` is that we can always make our graphs look better by making small additions to our code. Let's make those adjustments in this second iteration of the bar plot:

```
# use geom_bar to make a bar plot
stat_identity2 <- ggplot(data=lewis_party_support,aes(x=party_affiliations,y=Per_Support,
                                          fill=party_affiliations)) +
  ggtitle("Percentage Support by Political Affiliation") +
  geom_bar(stat = "identity", width = 0.7, show.legend = FALSE, colour = "Black") +
    # hide legend
  theme_bw() + #black and white background
  scale_x_discrete(limits=c("Overall Sample","Democrat", "Republican", "Independent")) +
  scale_fill_manual(values = c("Overall Sample" = "black",
                               "Democrat" = "blue",
                               "Republican" = "red",
                               "Independent" = "gray")) + #manually choose colors
  scale_y_continuous(labels = percent, limits = c(0,1)) + #set y axis continuous
  theme(panel.grid.major = element_blank()) + #no horizontal grid lines
  theme(panel.grid.minor = element_blank()) + #no vertical grid lines
  labs(title = "Percentage Support by Political Affiliation",
       subtitle = "Support for the creation of a national registry of police
       officers fired for misconduct",
       caption = "Nationally representative sample of 1000 respondents",
       x = " ", y = "Support (%)")+
  theme(axis.text=element_text(size=12), #setting font size
        axis.title=element_text(size=12,face="bold")) +
```

```
  theme(plot.title = element_text(size=14,face="bold"))

stat_identity2
```



Perfect! Our graph already looks so much better, just by fixing those few things.

Notice how we set up the Y axis to be continuous by using `scale_y_continuous(labels = percent, limits = c(0, 1))` while we set up the X axis to be discrete `scale_x_discrete()`. This is an important distinction to make. We want to make sure that `ggplot2` knows that our x axis is discrete, since the different types of party affiliations are, indeed, discrete variables. Respondents are either Democrat, Republican, or Independent – there are no intermediate values. On the other hand, we should also specify that our y axis is continuous, since percentages are a continuous variable. (Note: In setting up the x axis scale, we were also able to re-order the bars from left to right in a more intuitive order).

Another important note to make is that often times, it is common practice for graphs and plots to be black and white or grayscale for research papers and print journals. As a result, we usually create two versions of the same plot – one in color, like the one we created just now, and another graph in grayscale.

Helpful hint: These final modifications to the graph (e.g., adding titles and labels, adjusting legends, clearing up the plot background, etc.) can and should be applied to other forms of plots using ggplot2 (scatter plots, lineplots, maps, in previous parts of the walkthrough) in order to increase legibility and clarity, and allow for more intuitive interpretation of our data.