SPEC REU R Resources: Introduction to Bar Plots with ggplot2

Jasmine Chu, Gaea Morales, Claudia Salas Gimenez and Ben Graham

Summer 2024

In this final walkthrough for Data Visualization II, you'll dive into creating bar plots with the `ggplot2` package in R. Building on what you've learned so far, we'll focus on interpreting categorical data using bar plots, using real survey data from the Lewis Registry.

Initial Setup

Before we start plotting, make sure you're set up to load the necessary libraries and data. The data in this document is real survey data from the Lewis Registry, national database collected in Fall 2020 that gathers information on police officers separated or terminated due to serious misconduct.

```
# Set working directory
#setwd("YourFolderPath")

# Load required libraries
library(ggplot2)
library(dplyr)
library(scales)

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

Exploring Bar Plots

Bar plots are ideal for comparing categorical data. In this walkthrough, we'll create a bar plot to examine support and opposition for a national registry of police officers terminated due to misconduct. The column `usc_omni_102020` in our data contains binary responses ("Yes" or "No") to whether respondents support the creation of a national registry of police officers who have been terminated/separated due to serious misconduct.

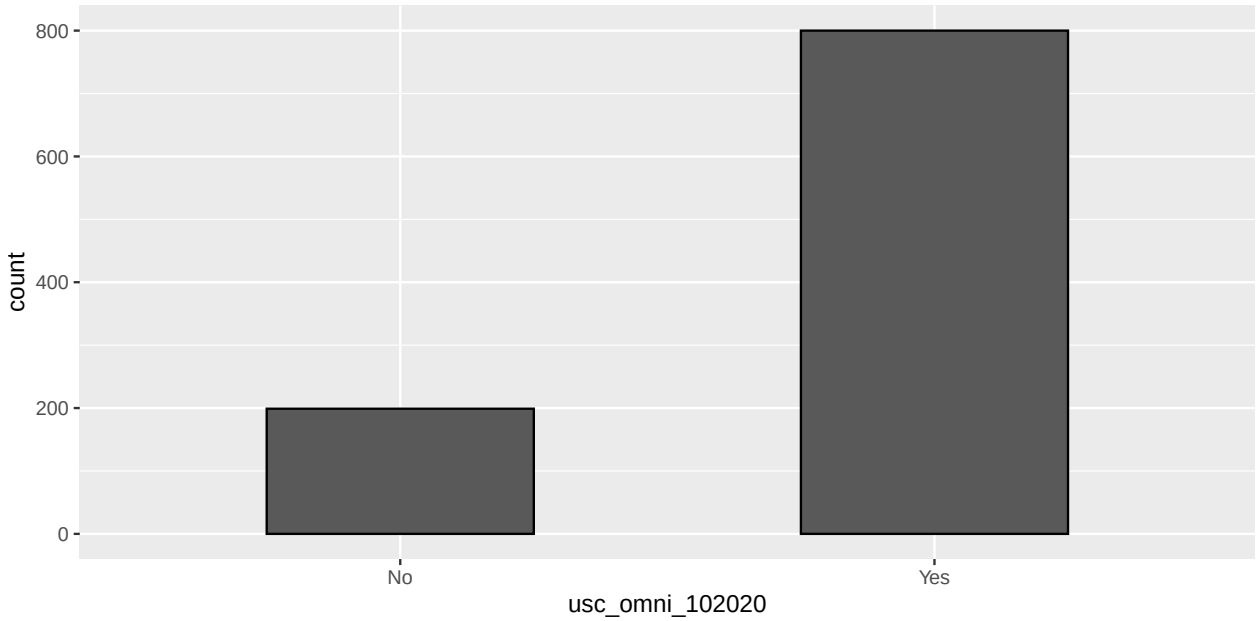
Note: Before plotting, we'll clean our data with `dplyr`'s `filter()` to remove any "skipped" responses, focusing solely on the "Yes" and "No" answers.

```
# Filter out non-responses
lewis <- lewis %>%
  filter(usc_omni_102020 != "skipped") # Remove 'skipped' entries
```

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

```
# Creating the bar plot
stat_count <- ggplot(data = lewis, aes(x = usc_omni_102020)) +
  geom_bar(stat = "count", width = 0.5, colour = "Black")
```

```r
# Display the plot
stat_count
```



Great! We've created a basic bar plot showing respondents' views on the creation of a national registry, with approximately 800 people in support and 200 opposed. To make this plot publication-ready and more insightful, let's refine it with a few enhancements.

Upgrading the Plot

Grouping by Political Affiliation

Now, let's refine the plot by grouping responses based on political affiliation. The `pid3` column in our dataset shows each police officer's political affiliation as "Democrat," "Republican," "Independent," or "Other." To group responses by political affiliation, we'll set `fill = pid3` in the aesthetic argument.

We also want to see "Yes" and "No" responses side-by-side for each affiliation. Setting `position = 'dodge'` in `geom_bar()` will arrange the bars next to each other instead of stacking them, making it easier to read.

Next, we'll customize bar colors with `scale_fill_manual(values = c())` to show "Democrat" in blue, "Republican" in red, and "Independent" in gray.

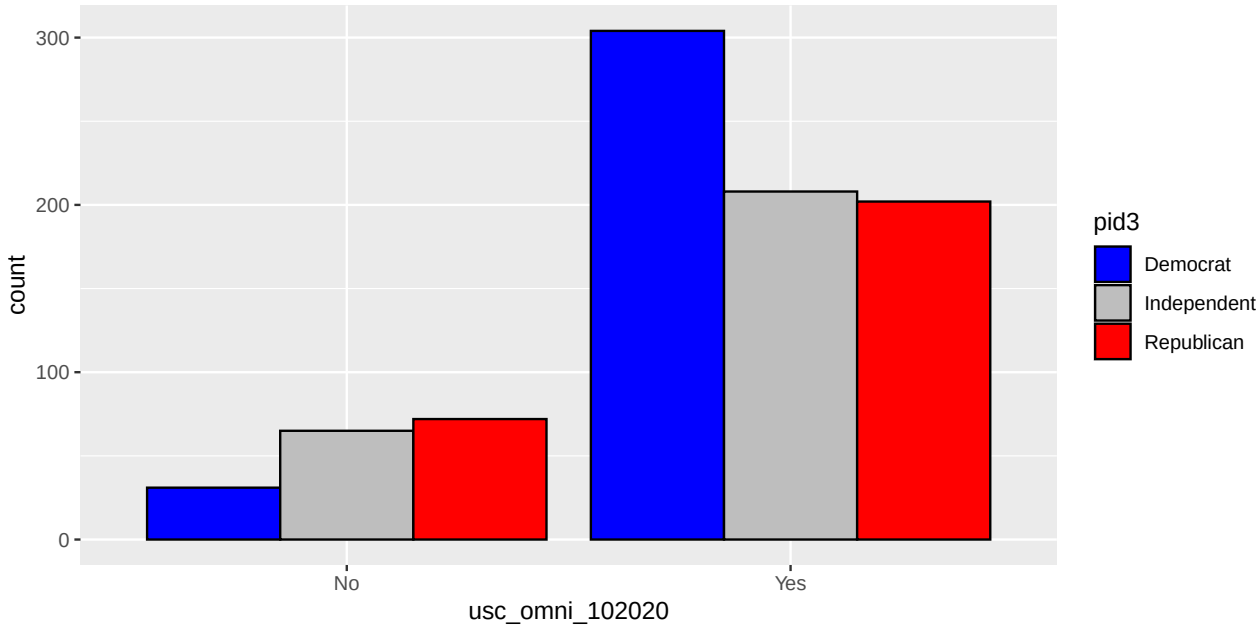
Before creating our second bar plot, let's filter out the "Not Sure" and "Other" entries from the `pid3` column.

```r
# Further filter the data
lewis <- lewis %>%
  filter(pid3 != "Other" & pid3 !="Not sure")
```

Now let's produce our second bar plot.

```r
# Creating the bar plot with grouping
stat_count2 <- ggplot(data = lewis,aes(x = usc_omni_102020, fill = pid3)) +
  geom_bar(stat = "count", colour = "Black", position = 'dodge') +
  scale_fill_manual(values = c("Democrat" = "blue",
                               "Republican" = "red",
                               "Independent" = "gray"))
```

```r
# Display the plot
stat_count2
```



This enhanced bar plot now clearly displays support levels segmented by political affiliation, providing deeper insight into the data. Notice that about 300 Democrats and 200 Republicans support the registry, while approximately 30 Democrats and 70 Republicans oppose it.

`stat = "count"` vs. `stat = "identity"`

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

- `stat = "count"` counts the number of observations for each value of the x-variable, so no y-variable is needed. It's the default for `geom_bar()`.
- `stat = "identity"` allows you to specify the y-variable directly, using the provided values for the plot. This customization requires including a y-variable in the argument.

Now, let's create a bar plot showing the percentage support for a national registry of police officers fired for misconduct, grouped by respondents' political affiliation. We'll use `usc_omni_102020` for support responses and `pid3` for political affiliation. First, we'll calculate the support percentage for each group with `dplyr`, then use `stat = "identity"` in `geom_bar()` to make the bar plot.

Let's first make a few changes to our data by calculating the percentage support by political affiliation.

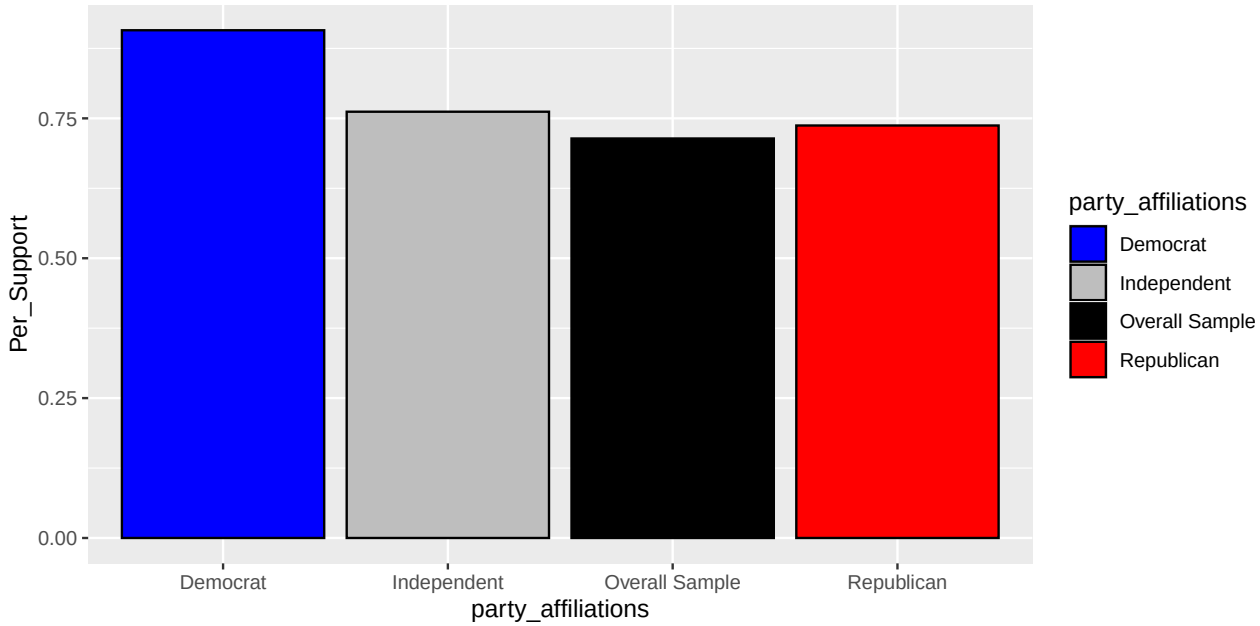
- Using `dplyr`, for each political affiliation we can take the number people who approved the creation of the national registry and divide that by the total number of people with that political affiliation.
- Merge the two dataframes. This will combine the overall percentage of support with the percentage of support for each political party. Since the dataframes have identical columns, we can use the `rbind()` function to bind them by row.

```r
# Calculate the overall support percentage for creating a registry
lewis_party_support_total <- lewis %>%
  group_by(usc_omni_102020) %>%
  summarise(Per_Support = n() / 1000) %>%  # Calculate the proportion of responses
  filter(usc_omni_102020 == "Yes") %>%     # Filter to keep only "Yes" responses
```

```
  rename(party_affiliations = usc_omni_102020)  # Rename column for clarity
lewis_party_support_total[1, 1] <- "Overall Sample"  # Label this as "Overall Sample"

# Calculate the percentage support by political affiliation
lewis_party_support_party <- lewis %>%
  group_by(pid3) %>%
  summarise(Per_Support = sum(usc_omni_102020 == "Yes") / n()) %>%
  # Calculate support percentage for each group
  rename(party_affiliations = pid3)  # Rename column for clarity

# Merge the datasets using 'rbind()'
lewis_party_support <- rbind(lewis_party_support_total, lewis_party_support_party)
```

With our organized data, we can now set up the bar graph using the `lewis_party_support` dataframe. For the aesthetic mapping, the x-axis will represent party affiliation, and the y-axis will show the percentage of support. We'll color the bars by party affiliation using `fill = party_affiliations`. Since we are customizing the Y variable, we'll use `stat = "identity"` instead of `stat = "count"`. Finally, we'll set the bar colors to blue for `"Democrat"`, red for `"Republican"`, and gray for `"Independent"`.

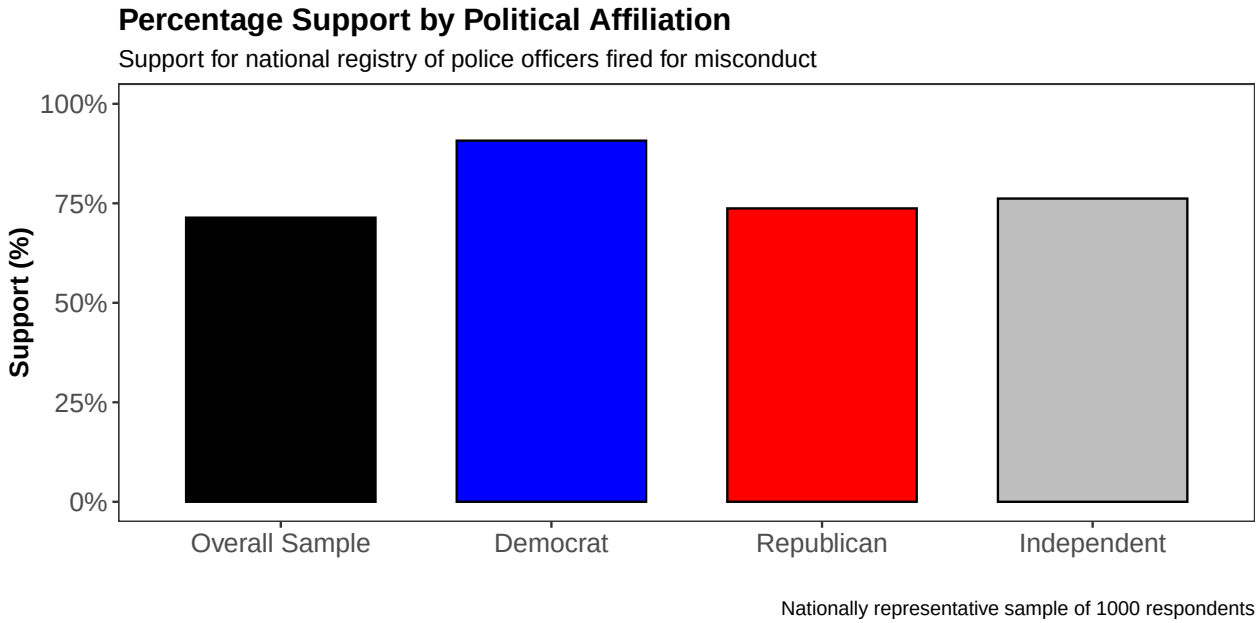
```
# Create the bar plot
stat_identity <- ggplot(data = lewis_party_support,
                        aes(x = party_affiliations,
                            y = Per_Support,
                            fill = party_affiliations)) +
  geom_bar(stat = "identity", colour = "Black") +
  scale_fill_manual(values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

# Display the plot
stat_identity
```



This bar graph is functional but has several issues. It lacks a title and subtitle, the legend needs clearer labels,

and the y-axis should display percentages. Additionally, the political affiliations are double-labeled on the x-axis and in the legend, making the legend redundant. The font size is also too small to be easily readable. The great thing about `ggplot2` is that it allows us to improve our graphs with minor code adjustments. Let's make these enhancements in the next version of the bar plot:

```
# Create the 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") +
  # Add plot title
  geom_bar(stat = "identity", width = 0.7, show.legend = FALSE, colour = "Black") +
  # Hide legend
  theme_bw() +
  # Apply black and white background
  scale_x_discrete(limits = c("Overall Sample", "Democrat",
                              "Republican", "Independent")) +
  # Set x-axis order
  scale_fill_manual(values = c("Overall Sample" = "black",
                               "Democrat" = "blue",
                               "Republican" = "red",
                               "Independent" = "gray")) +
  scale_y_continuous(labels = scales::percent, limits = c(0, 1)) +
  # Set y-axis to percentage format
  theme(panel.grid.major = element_blank()) +
  # Remove major grid lines
  theme(panel.grid.minor = element_blank()) +
  # Remove minor grid lines
  labs(title = "Percentage Support by Political Affiliation",
       # Main title
       subtitle = "Support for national registry of police officers fired for misconduct",
       # Subtitle
       caption = "Nationally representative sample of 1000 respondents",
       # Caption
       x = " ", y = "Support (%)") + # Axis labels
  theme(axis.text = element_text(size = 12),
        # Set axis text size
        axis.title = element_text(size = 12, face = "bold")) +
  # Set axis title size and bold
  theme(plot.title = element_text(size = 14, face = "bold"))
# Set plot title size and bold

# Display the plot
print(stat_identity2)
```



This final plot shows how simple tweaks like adjusting titles, labels, and colors can significantly improve clarity. Now you're ready to apply similar techniques to other visualizations in your work.

With this final walkthrough, you've built essential skills in `ggplot2` for creating bar plots, scatter plots, and other key visualizations. Through these assignments, you've practiced transforming raw data into clear, compelling graphics, ready for publication. Now, as you move into group work and homework assignments, use this opportunity to deepen your understanding, refine your visualizations, and convey data-driven insights effectively.