# Lab 4: UN votes

Dhyey Mavani

September 7, 2021

## About this lab

In the first lab, you had the option to create a visualization that captured how the voting record of different countries changed over time on a variety of issues. We will revisit the UN voting record data again in this lab, with a focus on learning how to change features such as shapes, colors, and line types with the **ggplot2** package.

## Data

The **unvotes** package provides three datasets that capture the voting history of countries in the United Nations General Assembly: `un_roll_calls`, `un_roll_call_issues`, and `un_votes`. Each of these datasets contains a variable called `rcid`, the roll call id, which can be used as a unique identifier to join the three datasets together.

The `un_votes` dataset provides information on the voting history of the United Nations General Assembly. It contains one row for each country-vote pair.

The `un_roll_calls` dataset contains information on each roll call vote of the United Nations General Assembly.

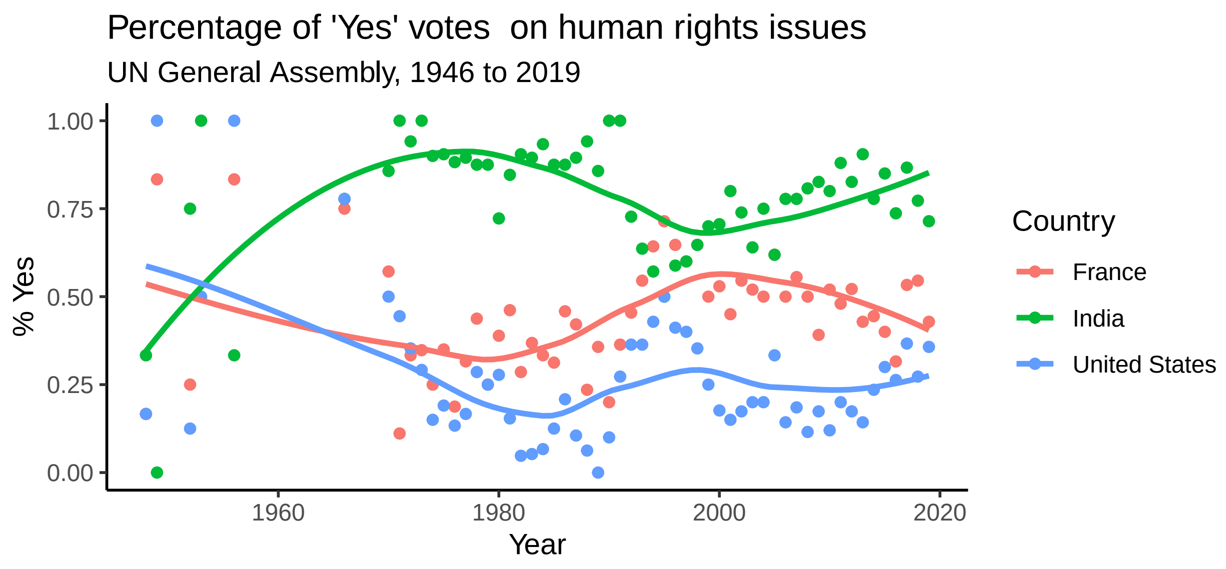
The `un_roll_call_issues` dataset contains (topic) classifications of roll call votes of the United Nations General Assembly. Many votes had no topic, and some have more than one.

Problem 1 **Data prep** The code below prepares our data in several ways. First, it combines our three datasets into one. Then, it limits the dataset to focus only on one of the six issues ("Human Rights") and three of the countries. Finally, some wrangling is done so that we are only using records where there are more than 5 votes on an issue. *Update the code below to select three countries of interest to you.* The country names should be spelled and capitalized exactly the same way as they appear in the dataset. A full list of the countries is provided in the UN country list at the end of this lab.

```r
# ready data for plotting
unvotes_hr <- un_votes %>%
  inner_join(un_roll_calls, by = "rcid") %>%
  inner_join(un_roll_call_issues, by = "rcid") %>%
  filter(issue == "Human rights",
         country %in% c("United States",
                        "India",
                        "France")) %>%
  group_by(country, year = year(date), issue) %>%
  summarize(votes = n(),
            percent_yes = mean(vote == "yes")) %>%
  filter(votes > 5)
```

Problem 2 **Shapes and sizes** The default symbol for `geom_point()` is, well, a point. But you can change the symbol shape and size using the `shape =` and `size =` options, respectively, within the `geom_point()` function. Use the code below as a starting point to modify the plot in the questions that follow.

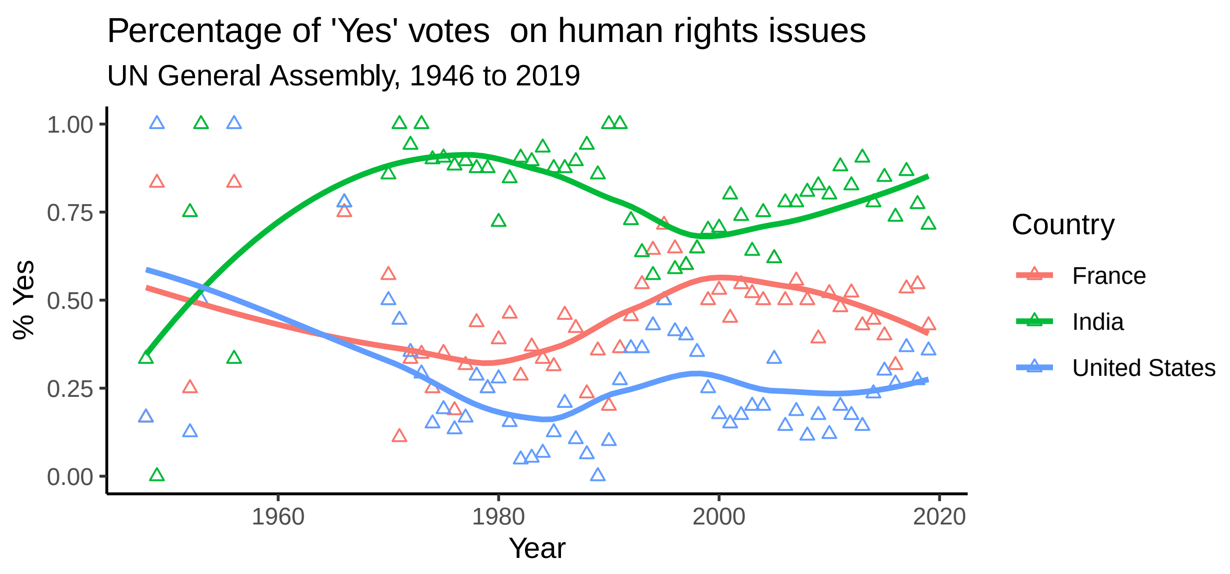
```
# plot the data
ggplot(data = unvotes_hr, mapping = aes(x = year, y = percent_yes,
                                        color = country)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Percentage of 'Yes' votes  on human rights issues",
       subtitle = "UN General Assembly, 1946 to 2019",
       y = "% Yes",
       x = "Year",
       color = "Country")
```



2.1 Try adding `shape = 2` to the `geom_point()` function. What happens?

It makes the datapoint indicators triangles.

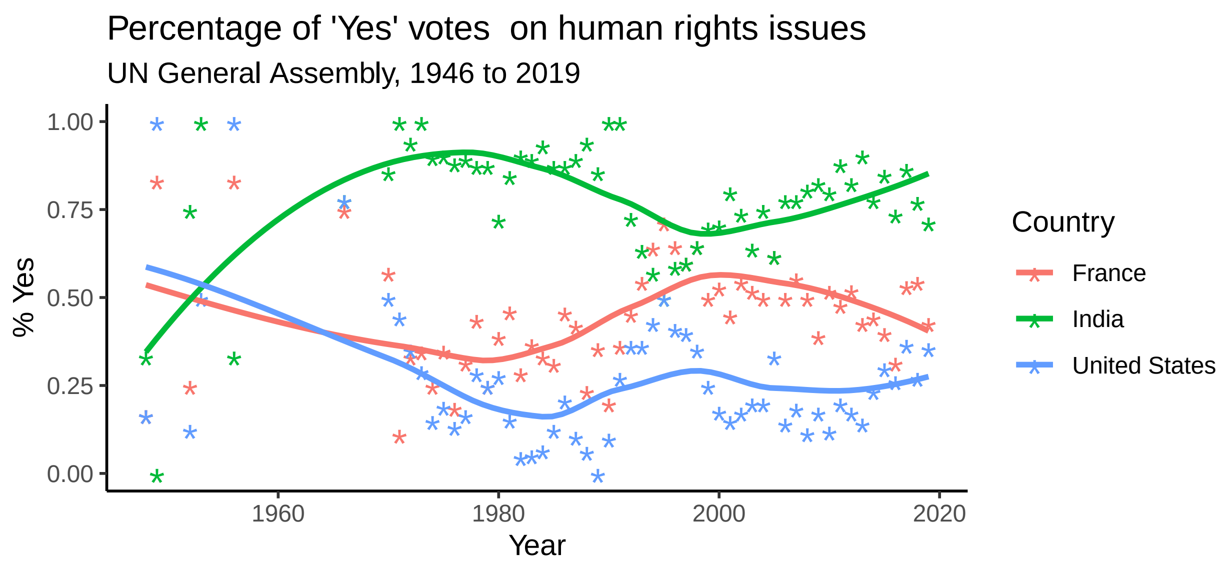
```
ggplot(data = unvotes_hr, mapping = aes(x = year, y = percent_yes,
                                        color = country)) +
  geom_point(shape = 2) +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Percentage of 'Yes' votes  on human rights issues",
       subtitle = "UN General Assembly, 1946 to 2019",
       y = "% Yes",
       x = "Year",
       color = "Country")
```



## 2.2 What happens if you add `shape = "*", size = 5` to `geom_point()` intead?

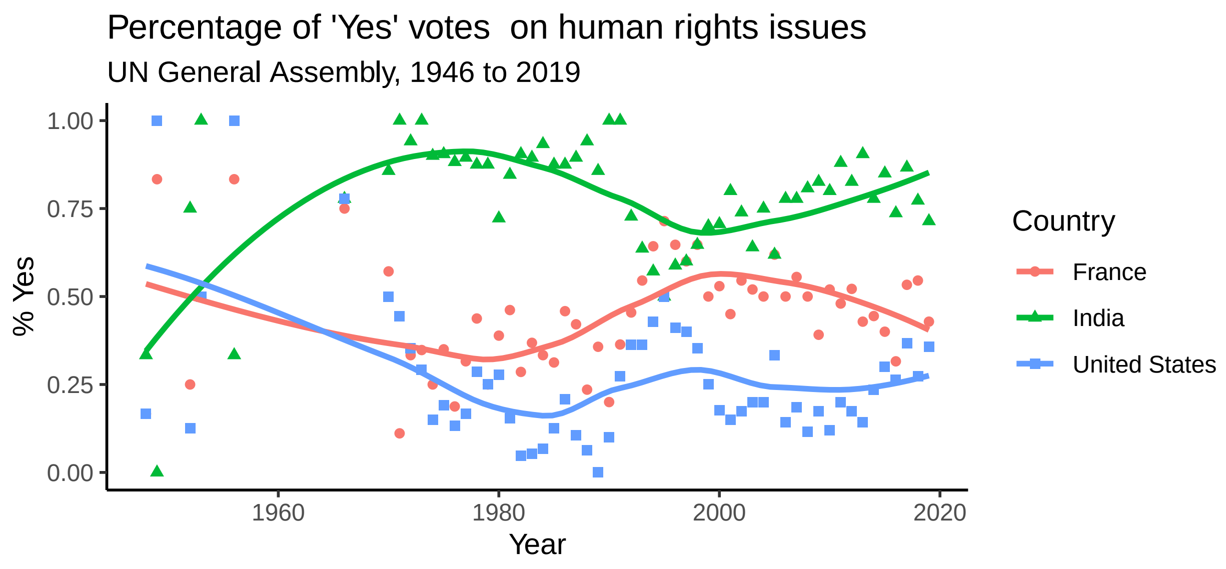
This makes the data indicators a "*" and changes the size.

```r
ggplot(data = unvotes_hr, mapping = aes(x = year, y = percent_yes,
                                        color = country)) +
  geom_point(shape = "*", size = 5) +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Percentage of 'Yes' votes  on human rights issues",
       subtitle = "UN General Assembly, 1946 to 2019",
       y = "% Yes",
       x = "Year",
       color = "Country")
```



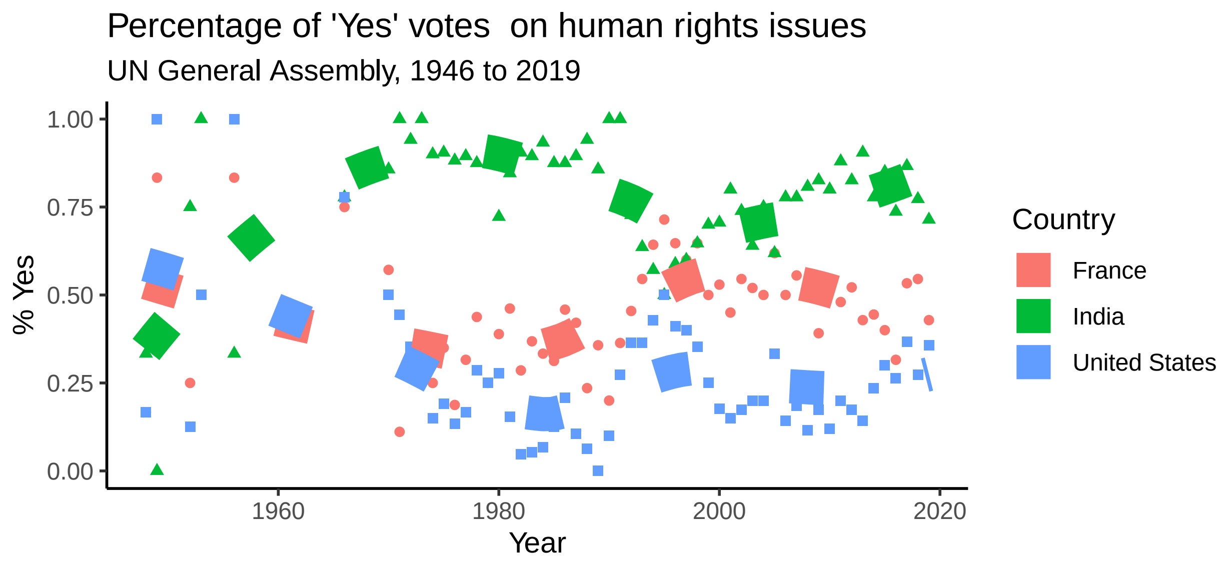
2.3 You can also specify different shapes for the different countries. Since country is a variable in our dataset, we can use the `aes()` function within `geom_point()` to specify different shapes for different countries. See if you can figure out the correct syntax to do this. Can you also figure out how to clean up the legend? (*Hint*: add something to the `labs()` function.)

```
ggplot(data = unvotes_hr, mapping = aes(x = year, y = percent_yes,
                                        color = country)) +
  geom_point(aes(shape = country)) +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Percentage of 'Yes' votes  on human rights issues",
       subtitle = "UN General Assembly, 1946 to 2019",
       y = "% Yes",
       x = "Year",
       color = "Country", shape = "Country")
```



Problem 3 **Line types** The default line type for `geom_smooth()` is a solid line. You can change the line type and thickness using the `lty =` and `size =` options, respectively, within the `geom_smooth()` function. Copy and paste the code from your last figure in Part 2.3. Update the figure using the `aes()` function within `geom_smooth()` so that each country has a different line type. Can you also figure out how to clean up the legend?

```
ggplot(data = unvotes_hr, mapping = aes(x = year, y = percent_yes,
                                        color = country)) +
  geom_point(aes(shape = country)) +
  geom_smooth(method = "loess", se = FALSE, lty = 3, size = 6, aes(shape = country)) +
  labs(title = "Percentage of 'Yes' votes  on human rights issues",
       subtitle = "UN General Assembly, 1946 to 2019",
       y = "% Yes",
       x = "Year",
       color = "Country", shape = "Country")
```



Problem 4 **Colors** There are many, many, many different ways to change the colors of points, lines, etc. in `ggplot()`. Today we'll explore just a few of them! We'll change colors both manually and using *color brewer*, which provides color schemes designed by a professional to help people choose good color schemes for their graphs.

4.1 **Color Brewer**: Copy and paste the code from your last figure created in Part 3. Add a layer `scale_color_brewer()` before the `labs()` layer (don't forget to add a plus sign). What happens?

4.2 Within the `scale_color_brewer()` function, add the options `type = "div"` and `palette = 1`. Is this a good color scheme for this figure? Why or why not? (Can you think of a figure for which this would be a good color scheme?)

4.3 Check out the [scale brewer reference manual]https://ggplot2.tidyverse.org/reference/scale_brewer.html for more information about sequential, diverging, and qualitative color schemes available from Color Brewer. The color brewer site is also helpful to visualize the different palettes (also shown in Figure 2.11 of MDSR). Does a sequential, diverging, or qualitative color scheme make sense for this figure? Update the figure with one of the palettes from the appropriate scheme.

4.4 **Manual**: Replace the `scale_color_brewer()` line with `scale_color_manual(values = c("green", "purple", "blue"))`. Notice there are three colors I'm specifying, one for each of the three countries.

4.5 Don't like those colors? You can be more exact by specifying hex color codes. Try replacing "green", "purple" and "blue" with "#05a05a", "#844185", "#024a81", respectively.

4.6 To identify hex codes for more colors, check out: color-hex.com. Create one last figure with three colors of your choice assigned manually.

## Additional information

### References

1. David Robinson (2017). unvotes: United Nations General Assembly Voting Data. R package version 0.2.0. https://CRAN.R-project.org/package=unvotes.
2. Erik Voeten "Data and Analyses of Voting in the UN General Assembly" Routledge Handbook of International Organization, edited by Bob Reinalda (published May 27, 2013).
3. Much of the analysis has been modeled on the examples presented in the unvotes package vignette.

### UN country list

Below is a list of countries in the dataset:

```
un_votes %>%
  select(country) %>%
  arrange(country) %>%
  distinct() %>%
  datatable()
```