Lab3_VisualizationUNVotes

BilalTariq

2024-02-05

```
library(tidyverse)
library(kableExtra)
library(unvotes)
library(lubridate)
library(DT)
library(plotly)

# set black & white default plot theme
theme_set(theme_classic())

# improve digit and NA display
options(scipen = 1, knitr.kable.NA = '')
```

Lab Purpose

In the first lab activity, you examined a visualization that captured how the voting record of three randomly selected 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 in visualizations such as shapes, colors, and line types with the **ggplot2** package.

You'll work on the lab in the company of classmates to help each other and share/see results with different options selected.

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.

```
# head shows the first 6 observations by default
# what does including the 4 do?
head(un_votes, 4)
```

```
# A tibble: 4 x 4
   rcid country       country_code vote
  <dbl> <chr>         <chr>        <fct>
1     3 United States US           yes
2     3 Canada        CA           no
3     3 Cuba          CU           yes
4     3 Haiti         HT           yes
```

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

```
head(un_roll_calls, 4)
```

```
# A tibble: 4 x 9
   rcid session importantvote date       unres   amend  para short         descr
  <int>   <dbl>         <int> <date>     <chr>   <int> <int> <chr>         <chr>
1     3       1             0 1946-01-01 R/1/66      1     0 AMENDMENTS, ~ "TO ~
2     4       1             0 1946-01-02 R/1/79      0     0 SECURITY COU~ "TO ~
3     5       1             0 1946-01-04 R/1/98      0     0 VOTING PROCE~ "TO ~
4     6       1             0 1946-01-04 R/1/107     0     0 DECLARATION ~ "TO ~
```

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.

```
head(un_roll_call_issues, 4)
```

```
# A tibble: 4 x 3
   rcid short_name issue
  <int> <chr>      <fct>
1    77 me         Palestinian conflict
```

```
2  9001 me         Palestinian conflict
3  9002 me         Palestinian conflict
4  9003 me         Palestinian conflict
```

1 - Data prep

The code below prepares our data in several ways (you saw parts of this code previously).

First, it combines our three datasets into one (we'll learn these commands in our next unit on wrangling). 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. You will be learning the details of the wrangling commands in the coming weeks. For now, just trust this does as specified.

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. The interactive data table is created by the *DT* package.

```
# 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("Cameroon",
                        "Iceland",
                        "Fiji")) %>%
  group_by(country, year = year(date), issue) %>%
  summarize(votes = n(),
            percent_yes = mean(vote == "yes")) %>%
  filter(votes > 5)
```

```
Warning in inner_join(., un_roll_call_issues, by = "rcid"): Detected an unexpected many-to-ma
i Row 382 of `x` matches multiple rows in `y`.
i Row 3009 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

```
`summarise()` has grouped output by 'country', 'year'. You can override using
the `.groups` argument.
```

This new data set is called *unvotes_hr* where the hr is for Human Rights, the issue we are focusing on.

Coding Details

> What might be a reasonable name for a new data set if you wanted to focus on the issue: Economic development?

Solution:

Saving the data set is different than what was done in the previous lab. There, since we only made one visual, we didn't really need to save it. Saving it is useful if we expect to re-use it several times.

> What piece of the code does the assignment/saving of the new data set?

Solution:

> The %>% set of symbols occurs often here. Do you know what this symbol is? If so, what is it?

Solution:

(This will come up more with wrangling, but since it's here now, we can discuss it.)

There is a newer "pipe" |> that you can use as well. For details/differences, you can check out the following article: [R pipes] (https://www.tidyverse.org/blog/2023/04/base-vs-magrittr-pipe/)

> The issue == "Human Rights" line pulls out only that issue. What do you think changing the == to != would do?

Solution:

(If you change this to check, be sure to put it back to == before continuing and re-run the chunk.)

> Code options - What would setting eval: false in the code chunk do? Would this be wise if we plan to use the data set for visuals in the rest of the document that we want to show in the knitted pdf?

Solution:

2 - Shapes and sizes

From the reading you saw that ggplot2 objects start with choosing the data set and setting the aesthetics (which can range from x and y variables to color and beyond). After that, you need to add a *geom* and a *stat* to make a layer in the plot. Geoms describe the geometric objects being plotted and stats are the statistical transformation to be used.

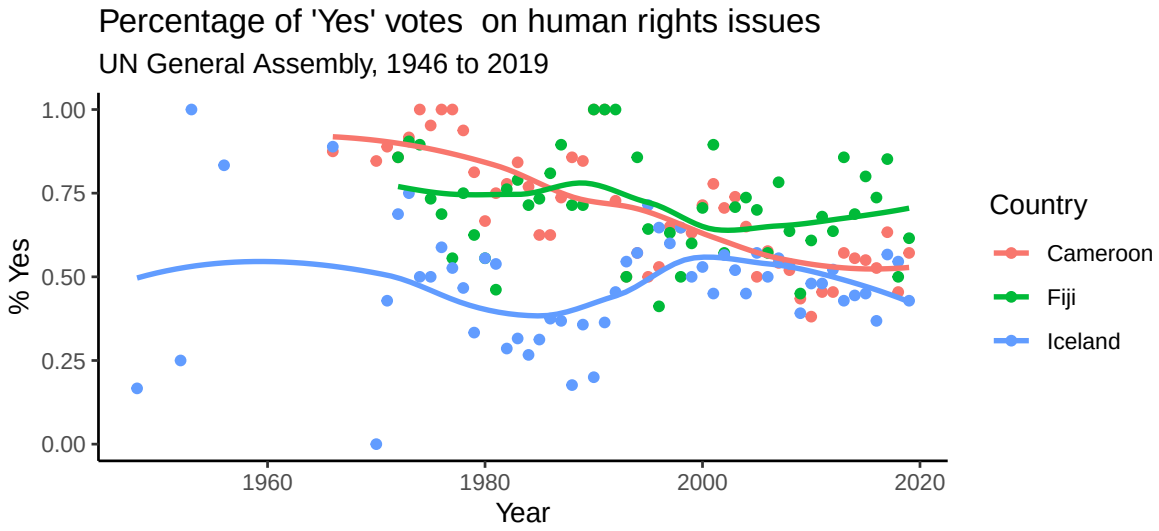
stat? Professor, we didn't see *stat* in the reading or in your examples! Every *geom* has a default *stat* (and vice versa), so you only need to specify one of them. For most of what we'll be doing, the *stat* is the identity function. That is, we plot the data as is. Thus, you really need to only worry about the *geom*s for now.

Many different *geom*s were explored in the reading. For scatterplots, we tend to use `geom_point` as a starting point.

The default symbol for `geom_point()` is 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. Copy/paste the code so you can compare without having to edit.

```
# 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")
```

```
`geom_smooth()` using formula = 'y ~ x'
```



Suppose you want to have the original plot but with different plots for each country side-by-side (you can have separate colors or not). What is the option to do this? Implement it.

Solution:

Undo the option you added in the last plot (i.e. return to the original). Try adding `shape = 2` to the `geom_point()` function. What happens?

Solution:

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

Solution:

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 use different shapes for your selected countries. Can you also figure out how to clean up the legend? (*Hint*: add something to the `labs()` function.)

Solution:

3 - Line types

Often, we want to examine variable relationships and follow them with estimated best fitting curves or regression lines. To do that, we use `geom_smooth()`. It can do both smoothed lines (LOESS fits) or regression lines. It does smoothed lines by default (not regression).

The default line type for `geom_smooth()` is a solid line. You can change the line type and thickness using the `lty =` (or write out `linetype`) and `size =` options, respectively, within the `geom_smooth()` function. Copy and paste the code from your last figure as a starting point below.

> Remove the different shapes for the countries. Then, 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? (We removed the shapes so you can see the linetypes in the legend better.)

Solution:

The `geom_smooth` function also allows you to fit regression lines. You have to set the method to "lm".

> Add regression lines instead of smoothed lines to the plot. Keep different line types. Does regression seem appropriate?

Solution:

> Return to smoothed lines without error bars, removing the regression lines. Here, we explore where the color aesthetic is set. Set the color aesthetic in the geom_point call, not in the overall aesthetic. Is the overall plot still as useful?

4 - Colors

There are 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.

> Copy your smoothed lines by country code from the last exercise (first plot from that section). Add a layer `scale_color_brewer()` before the `labs()` layer (don't forget to add a plus sign). What happens?

Solution:

> 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?)

Solution:

Check out the scale brewer reference manual 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.

Solution:

> 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.

Solution:

> 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.

Solution:

To identify hex codes for more colors, check out: color-hex.com.

> Create one last figure with three colors of your choice assigned manually.

Solution:

5 - Other Geoms

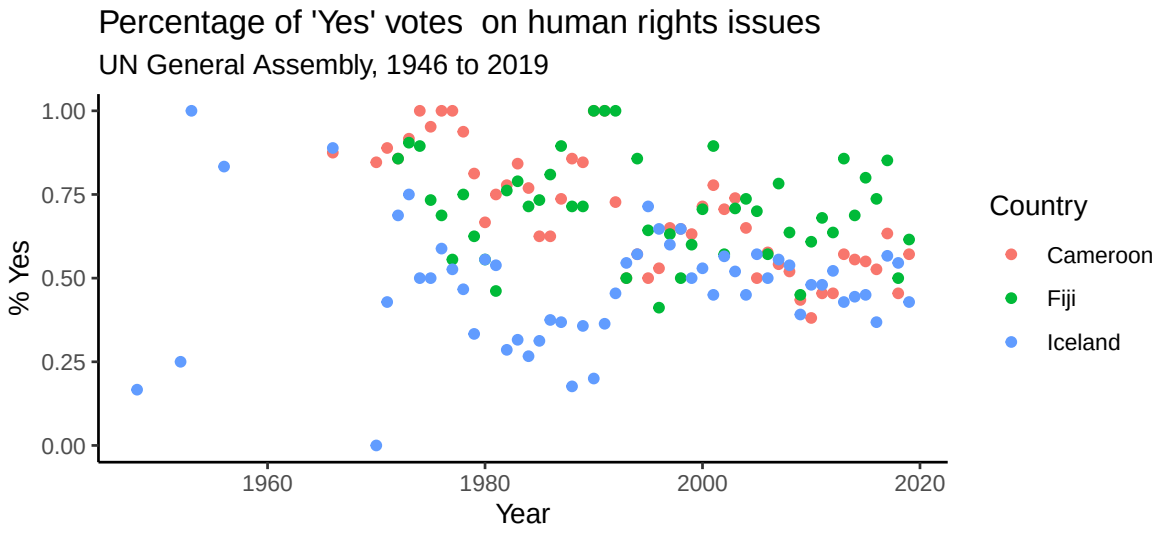
While geom_point and geom_smooth will do a lot for us, you'll see other geoms that can be useful as well, depending on what you aim to do.

Let's go back to our (almost) original plot to explore a few. This plot has the smoothed lines removed.

> Change the geom_point() to geom_line(). What happens?

Solution:

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



(geom_path gives the same results as geom_line here due to observations already being in chronological order in the data set.)

For these next geoms, our data isn't set up in such a way that they really make sense, but I want you to see if you can figure out what each does, and how you might use them for other data sets in the future.

> Copy over the code from above and use geom_area instead of geom_line. What happens?

Solution:

> Now try geom_step. What happens?

Solution:

Many geoms will show up if you search. Some are designed for univariate plots, etc. Feel free to experiment around to see what exists. For example, you might explore a violin plot over a boxplot.

There are many other things you can do with plots - annotate to add text, add interactivity, etc.

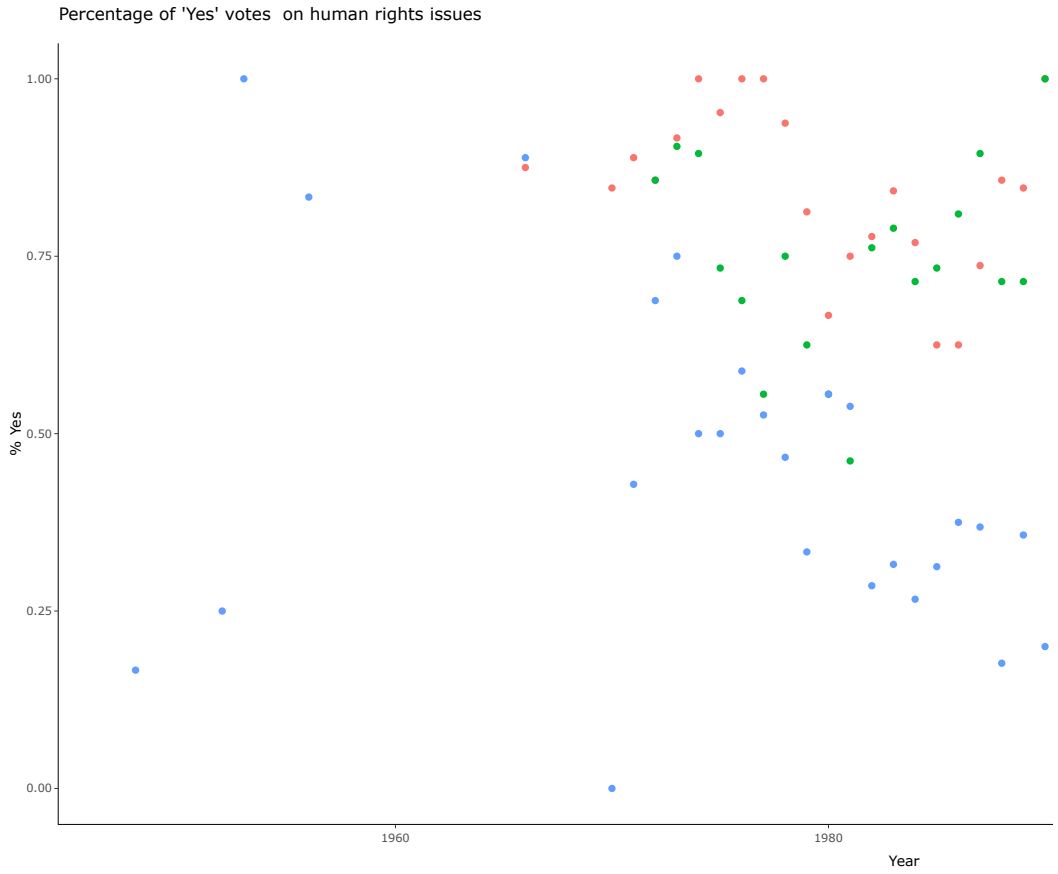
Here's an example of interactivity.

> What happens when you hover over this plot?

Solution:

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

# run for interactivity
plotly::ggplotly(g)
```


Percentage of 'Yes' votes on human rights issues
% Yes
1.00
0.75
0.50
0.25
0.00
1960
1980
Year

Is the interactivity going to remain when you compile to pdf? Explain.

Solution:

If you've finished and there is still time in the class period, be sure to think about what plots/geoms are appropriate for different types of variables, and ask questions you have about the material on visualizations.

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()
```