Week 3.1: Introduction to ggplot

Jennifer Lin

2021-11-06

Goals for Module 3

Welcome to Module 3 for the R workshop. In this module, our focus is the `ggplot` package. Here are the goals for the meetings

Week 1 – TODAY!

- Overview of `ggplot`
- GOAL: Make a very basic graph

Week 2 – NEXT WEEK!

- Colors, Fills, Labels and Axis limits
- GOAL: Add to the Plot from Week 1 with color and axis labels

Week 3 – WEEK AFTER NEXT WEEK!

- Themes and Fonts and Positioning
- GOAL: Finishing touches on the "publication ready" graph

Ultimately, the goal is to learn the process to building a publication ready graph through the components of the `ggplot` package. So, what are the components of a good graph (in my opinion[1])?

- *One x and one y axis*
- *Clearly labeled, with titles, axis and legends containing helpful information on graph contents*
- *Accessible color schemes*
- *Minimal background, maximal foreground*
- *Readable fonts and font sizes*
- *Use graph sections to tell a meaningful story*

[1] You should/will develop your own tastes for graphics as you progress in your research and in data visualization skills over time. Here are some of the tastes I have developed, but please do not feel obliged to adopt them in your own work.

An Overview of `ggplot`

When you want to bake a cake, it is absurd to expect that you will get a beautiful product in the first step. You need to bake the cake, assemble it and decorate it. It is a step-by-step, layer-by-layer process. This is the same thing with making graphics. You do it layer by layer and `ggplot` helps us make our graph of a cake by layers.

As such, the layers of our `ggplot` can be conceptualized as follows

```
ggplot(data layer)+
  graph layer +
  label layer +
  scale loayer +
  theme layer +
  others
```

where each layer (roughly) shows what should be there. Graphs do NOT need to follow a standard order so long as you get your desired outcome.

This week, we will cover the data and graph layer. Next week, the label and scales and finally, the theme and others[2].

In `ggplot`, components in the layers are joined by the plus sign (`+`). Each line must include a `+` if you want to connect it to the line below it or R will not recognize the components of the graph.

[2] Note: `ggplot` is a very wide universe and there are numerous sister packages that come off of the `ggplot` model that I won't have time to cover. In the last week, I will show you some tips on how to ask your own questions and search for advanced help online to solve your own R problems.

The Data Layer

The data that we will be interacting with for the next three weeks come from the American national Elections Studies 2020 edition. I have cleaned the data and included them in the GitHub folder for this unit. The data include items on demographics (race, gender, age) along with an assortment of feeling thermometer variables. Each feeling thermometer variable is scaled from 0 - 100.

For the data layer, most code follows the following format

```
ggplot(DATAFRAME,
       aes(x = X_VAR, y = Y_VAR, fill = FILL_VAR, color = COLOR_VAR))
```

Where

- DATAFRAME = the dataframe that you want to use
- X_VAR = the x-variable
- Y_VAR = the y-variable
- FILL_VAR = the fill variable – use to FILL graphs that have a space to fill
- COLOR_VAR = the color variable – use to COLOR lines or points in graphs[3].

[3] NOTE – you should only include FILL and COLOR arguments IF you want to fill by a variable. If you want a standard color throughout… stay tuned!

Here is an example of how the data layer arguments should apply. For my graph, I look at the correlation between feelings towards Rural Americans and the BLM movement. Specifically, I want to see if there is a partisan divide between Democrats and Republicans in this interaction. Therefore, the variables that I will be inputting are

- DATAFRAME = ANES (see variable read-in code)

- X_VAR = `FT_rural`
- Y_VAR = `FT_BLM`
- FILL_VAR = DOES NOT APPLY – This is a scatterplot
- COLOR_VAR = `PARTY` – coloring the dots by party

When the components are included, my code should look like this:

```
ggplot(ANES, aes(x = FT_rural, y = FT_BLM, color = PARTY))+
```

Exercise: Find Variables and Build Data Layer

1. Load the ANES dataset (Code in script)
2. Use `names()` to find the variables and `table()` to explore the variables.
3. Identify one x and one y variable (along with an optional fill or color variable) with which you want to make a graph
4. Without code, determine what kind of graph you want to make
5. Build your data layer accordingly

The Graph layer

`ggplot` has the capacity to build many types of graphs[4]. Each component in the graph layer starts with `geom_*` where the `*` denotes the kind of graph you want to construct. It is often very intuitive to the name of the graph that you want to build. Here is a table of some of the more common graphs. This is by no way an exhaustive listing.

[4] An excellent overview is `https://socviz.co/`.

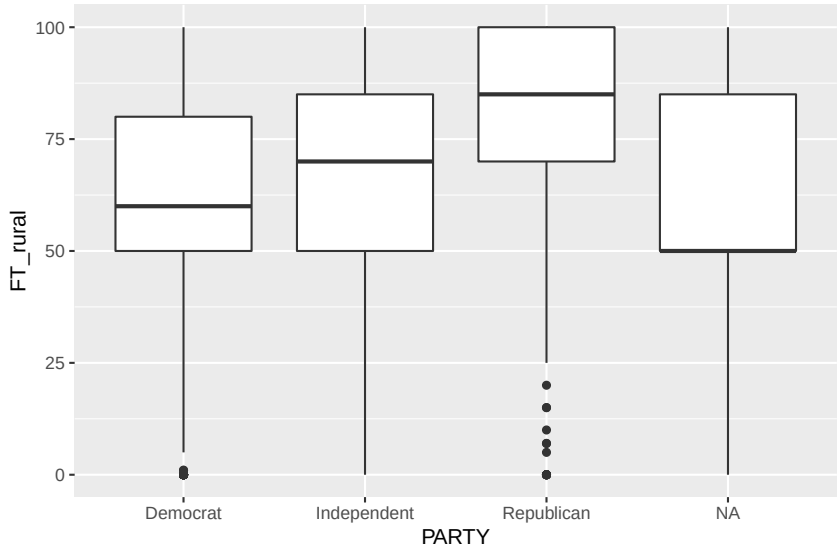
Operator	Description
`geom_line()`	line graph
`geom_point()`	scatterplot
`geom_bar()`	bar plot
`geom_histogram()`	histogram
`geom_boxplot()`	boxplot
`geom_violin()`	violin plot
`geom_sf()`	maps with shapefiles

For today, I will demonstrate the bar graph, histogram, box plot and scatterplot. You will notice that the components to building each one are quite similar and that you can mix and match your graphs.

Boxplots

To construct a simple boxplot, simply append `geom_boxplot()` to the end of your data layer.

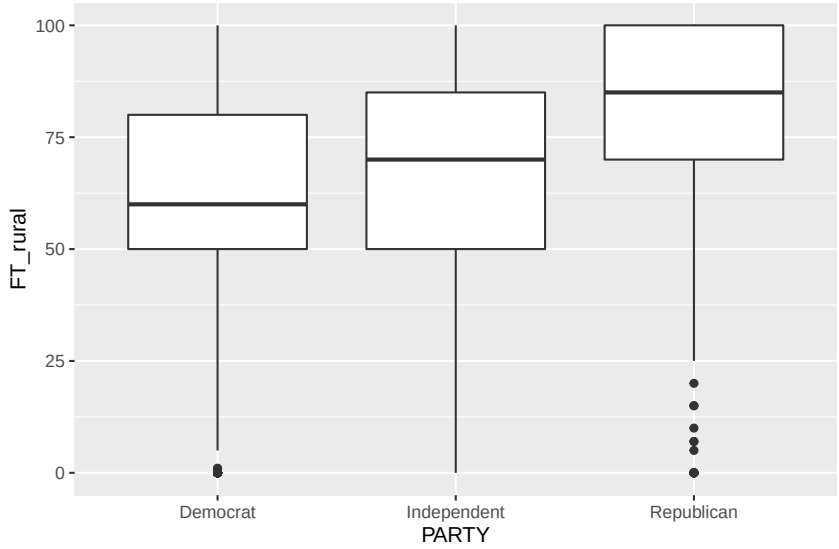
```
ggplot(ANES, aes(x = PARTY, y = FT_rural))+
  geom_boxplot()
```



Besides the background and default fill colors, this graph does not look too great. We will work on changing labels and colors next week, but for now, there is boxplot for people who did not indicate a party in their survey. We can remove this using some of the `dplyr` skills that you learned from previous R workshops, specifically, the `filter()` function.

Notice how you can append `dplyr` before `ggplot` and connect them with the `%>%`. This is because the functions are part of the same `tidyverse` universe that makes all the packages within it more or less compatible. So here is our boxplot, slightly more jazzed up. You will notice that the DATAFRAME that was within the `ggplot()` function is now in the outside and connected to the sequence with a `%>%`.

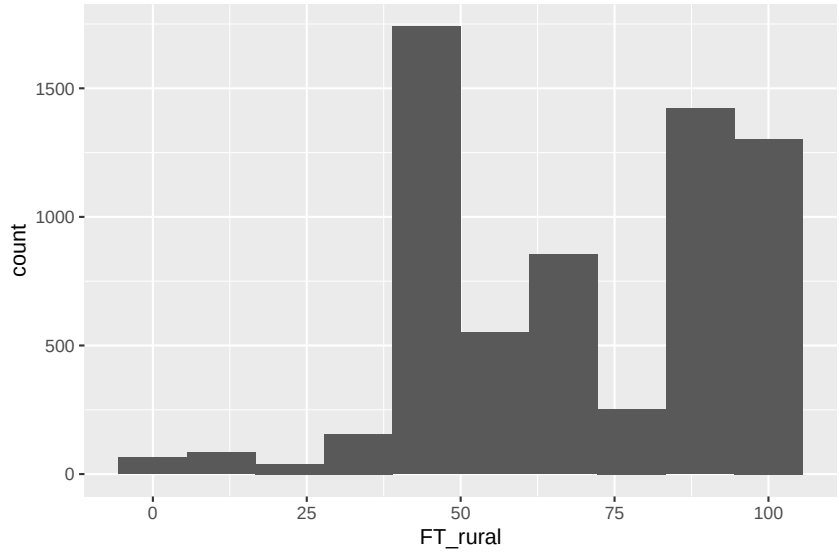
```
ANES %>%
  filter(!is.na(PARTY)) %>%
  ggplot(aes(x = PARTY, y = FT_rural))+
  geom_boxplot()
```



Histograms

In the case of Histograms, we only have one variable since the y-axis are counts. As such, this significantly simplifies our data layer. For the graph, you can make a histogram with `geom_histogram()` and indicate the number of bins that you want using `bins =`.

```
ggplot(ANES, aes(x = FT_rural))+
  geom_histogram(bins = 10)
```



Bar Graph

To make a basic bar graph, we first need to make a summary table. Like the boxplot example, we can do this all in one chunk, but I want

to separate it out to show you another example of how to make this plot.

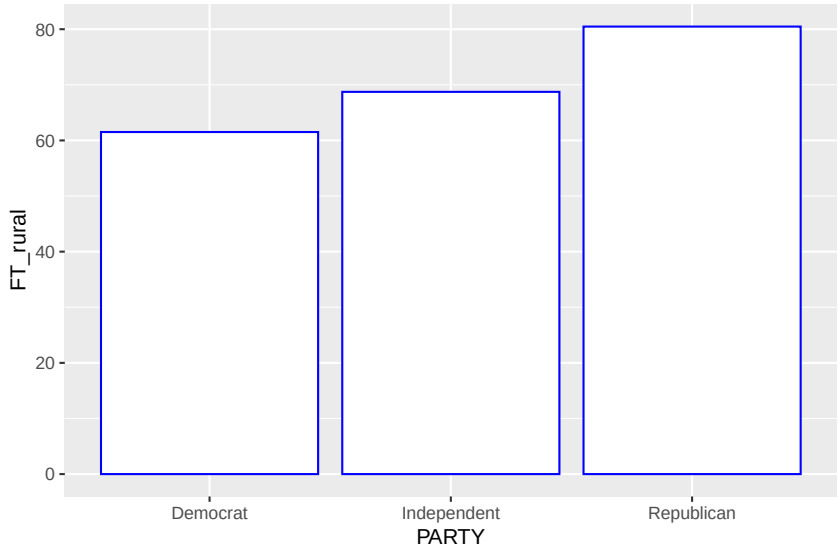
So let us start with the summary table, using the `dplyr` skills that you learned over the past few weeks. Here, I am planning on plotting average feelings towards rural Americans by political party.

```
by_Party <- ANES %>%
  filter(!is.na(PARTY)) %>%
  group_by(PARTY) %>%
  summarise(FT_rural = mean(FT_rural, na.rm = TRUE))
```

Using the `by_Party` data frame, we can make a bar graph as follows. In this case, notice that I am making the graph using a static color by adding fill and color options to the graph layer. I am filling the bars white and coloring the lines blue, regardless of party[5].

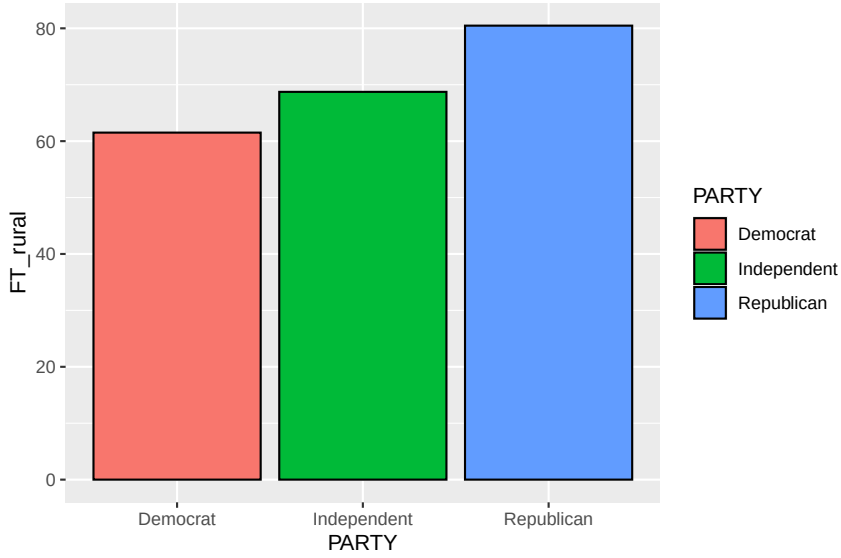
[5] We use `stat = "identity` to tell R, and `ggplot` that we do not want `geom_bar()` to do implicit calculations (which it does by default to count the number of columns if you only supplied an x variable in the data layer rather than an x and y variable. Using `stat = "identity`, means that you will supply your own y variable.)

```
ggplot(by_Party, aes(x = PARTY, y = FT_rural))+
  geom_bar(stat = "identity", fill = "white", color = "blue")
```



But suppose that I want to fill the bars based on the party of the respondent. Instead of noting the fill in the graph layer, I need to move that up to the data layer.

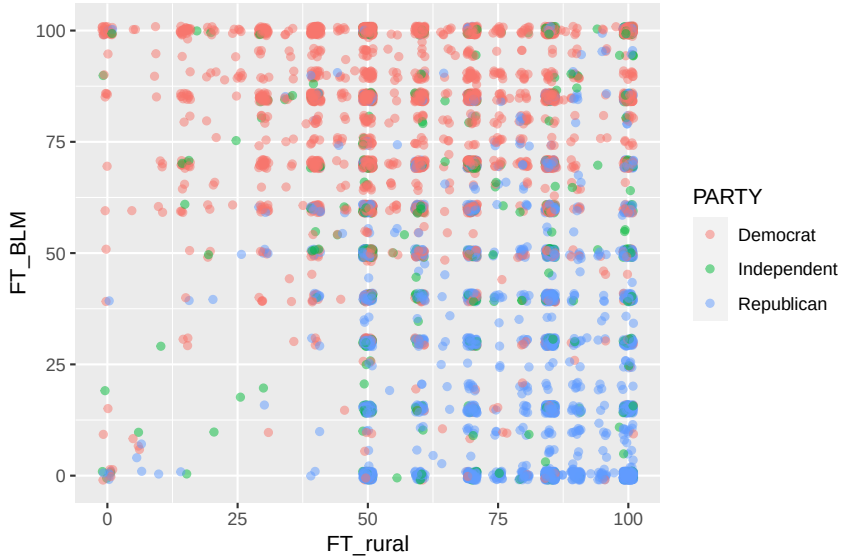
```
ggplot(by_Party, aes(x = PARTY, y = FT_rural, fill = PARTY))+
  geom_bar(stat = "identity", color = "black")
```



Scatterplots

To make a scatterplot, we use `geom_point`. Here, I am making the scatterplot that I mentioned earlier, looking at the relationship between feelings towards Rural Americans and the BLM movement and group it by party.

```
ANES %>%
  filter(!is.na(PARTY)) %>%
  ggplot(aes(x = FT_rural, y = FT_BLM, color = PARTY))+
    geom_point(position = position_jitter(1, 1), alpha = .5)
```



Here are some notes about the arguments that are included in the graph layer. Starting with `position`, `position_jitter()` separates

the points out so you can see how many people are clustered at any particular region. The `(1, 1)` shows the height and width of the area that the jitter applies to a particular intersection. Additionally, the `alpha` alters the transparency of the points. This ranges from 0 to 1, where lower numbers means greater transparency of the points.

Exercise: Adding a basic graph

1. From your data layer code earlier, add a `+` to the end if it is not already there.
2. Using the graphs explored in this section, construct a basic graph and color accordingly

Your Submission to the Lab Assignment for this week

1. Export the graph as a PDF using the "Export" button on the upper left hand corner of the plot window.
2. Upload your PDF AND the code, with your answers to the questions
3. Don't worry about fixing colors or labels this week. Will do that next week.