Session 3: Graphics in R

Jennifer Lin

2022-04-12

This week, we will talk about ways to make a good graph using R. Making a god graph is essential to communicating your research clearly. For this workshop, our goals are threefold.

1. Learn the basics of making graphs in R with `ggplot2`
2. Reinforce fundamental R skills with additional practice on creating vectors and loading datasets
3. Apply the fundamental plotting skills to spatial stat
4. Develop preferences for a publication quality 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])?

[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 just some characteristics that I find to be universal among good graphs.

- *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*

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 layer +
  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. If you are interested, come talk to me about these or consult Google.

The Data Layer

```
Pew <- read.csv("Pew_March2021.csv")
```

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 am going to demo a plot where I look at the correlation between feelings towards Donald Trump and Joe Biden. 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 = Pew (see variable read-in code)
- X_VAR = `FT_Biden`
- Y_VAR = `FT_Trump`
- FILL_VAR = DOES NOT APPLY – This is a scatterplot
- COLOR_VAR = `pid3` – coloring the dots by party

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

```
ggplot(Pew,
       aes(x = FT_Biden, y = FT_Trump, color = pid3))+
```

Exercise: Find Variables and Build Data Layer

1. Load the Pew 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/`.

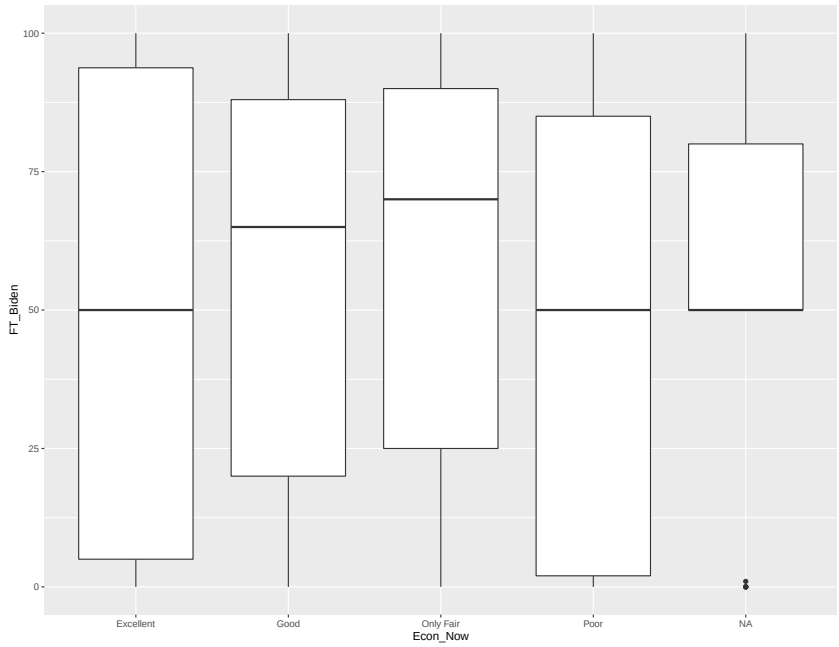
Operator	Description
`geom_line()`	line graph
`geom_point()`	scatterplot
`geom_bar()`	bar plot
`geom_histogram()`	histogram
`geom_boxplot()`	boxplot
`geom_violin()`	violin plot
`geom_sf()`	maps with shapefiles

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.

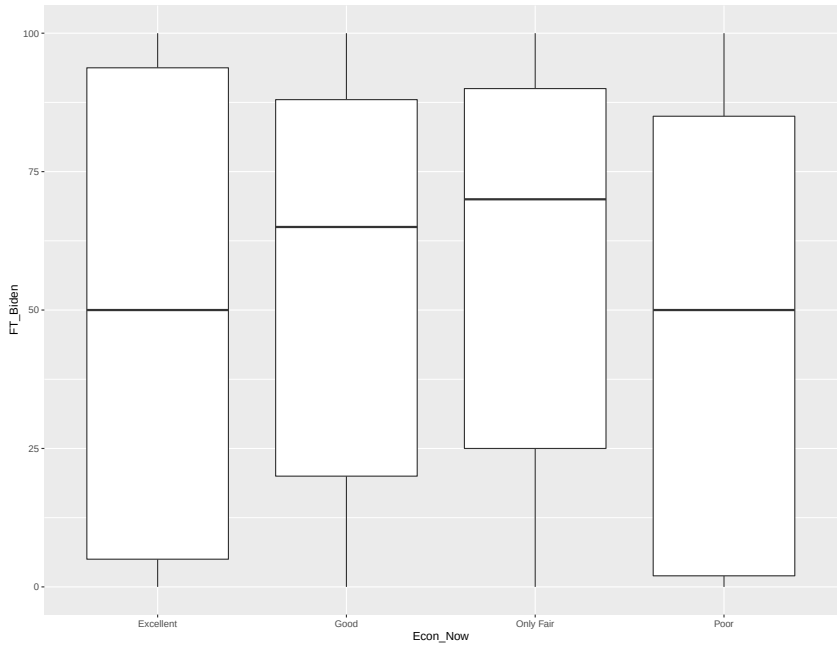
```
ggplot(Pew, aes(x = Econ_Now, y = FT_Biden))+
  geom_boxplot()
```



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 `%>%`.

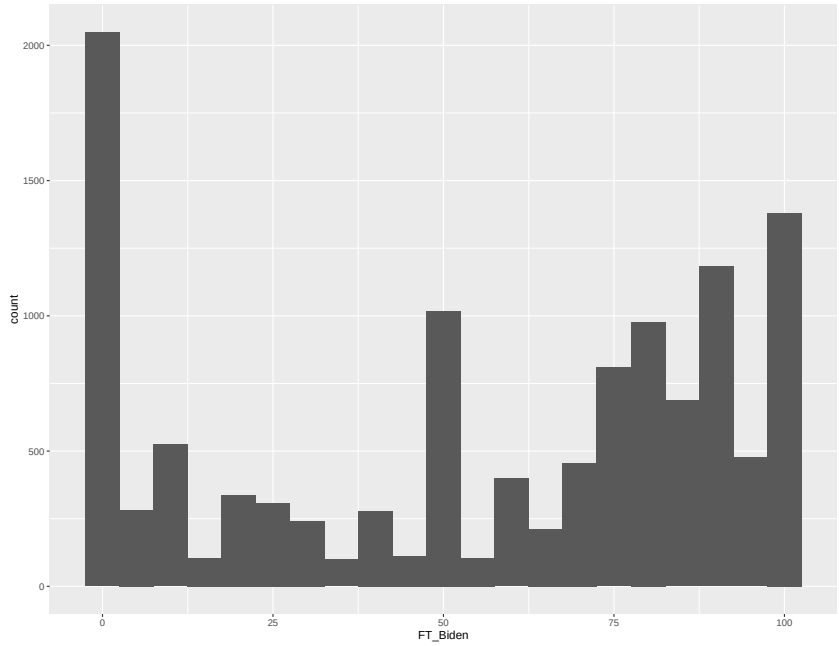
```
Pew %>%
  filter(!is.na(Econ_Now)) %>%
  ggplot(aes(x = Econ_Now, y = FT_Biden))+
    geom_boxplot()
```



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(Pew, aes(x = FT_Biden))+
  geom_histogram(binwidth = 5)
```



Bar Graph

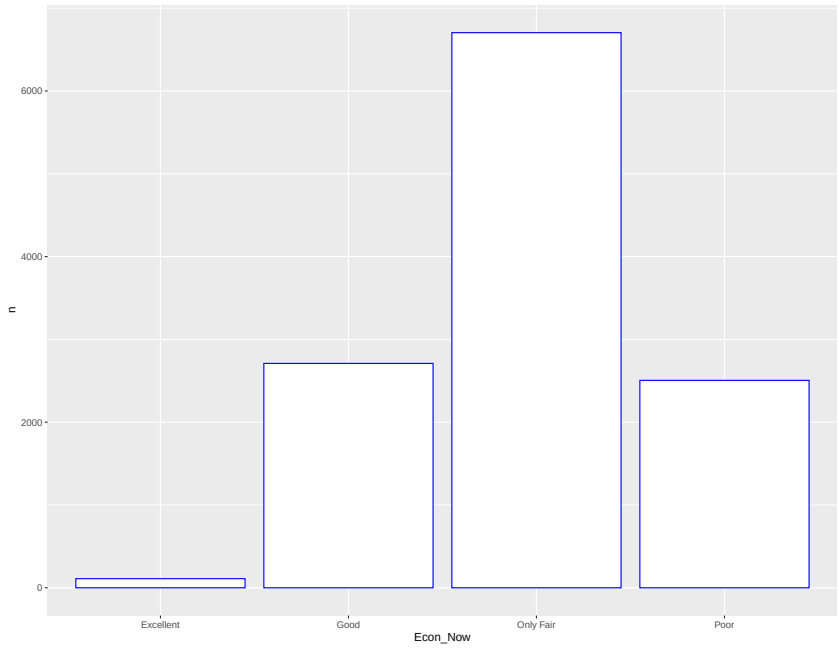
To make a basic bar graph, we first need to make a summary table. Like the boxplot example, we can do this all in one chunk, but I want to separate it out to show you another example of how to make this plot.

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 perceptions of the economy.

```
Econ_Now <- Pew %>%
  group_by(Econ_Now) %>%
  summarise(
    n = n(),
    .groups = 'keep'
  ) %>%
  filter(!is.na(Econ_Now))
```

Using the `Econ_Now` 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].

```
ggplot(Econ_Now, aes(x = Econ_Now, y = n))+
  geom_bar(stat = "identity", position = position_dodge(),
           color = "blue", fill = "white")
```

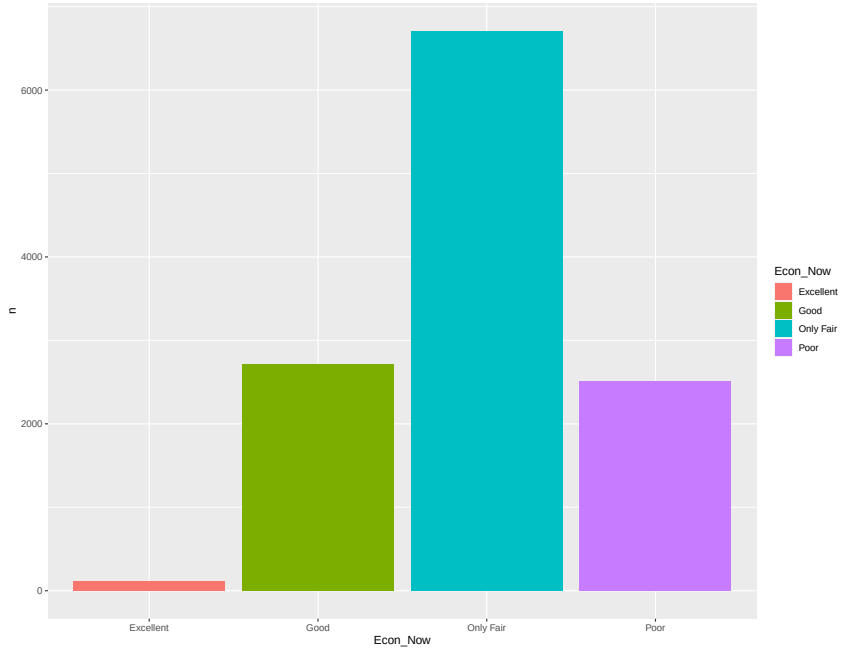


But suppose that I want to fill the bars based on the response. Instead of noting the fill in the graph layer, I need to move that up to

[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.)

the data layer.

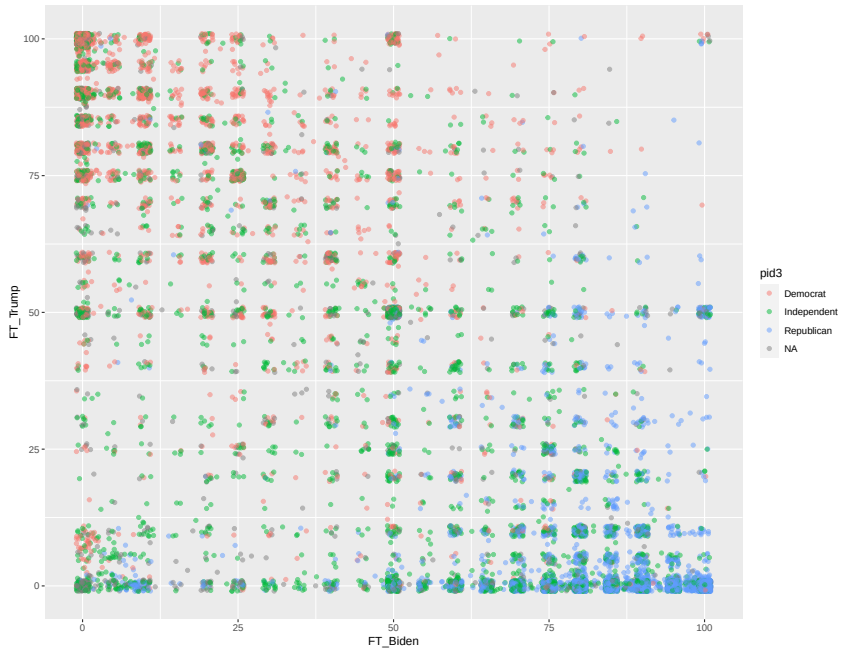
```
ggplot(Econ_Now, aes(x = Econ_Now, y = n, fill = Econ_Now))+
  geom_bar(stat = "identity", position = position_dodge())
```



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 Donald Trump and Joe Biden and group it by party.

```
ggplot(Pew,
       aes(x = FT_Biden, y = FT_Trump, color = pid3))+
  geom_point(position = position_jitter(1, 1), alpha = 0.5)
```



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

The Label Layer

The label layer allows us to add and change exiting labels for titles, scales and legends.

To change the x and y axis labels, we use `xlab()` and `ylab()` respectively. Within the parentheses, we add the new labels that we want in quotes (“ “) as such:

```
xlab("X axis title")+
ylab("Y axis title")
```

To add a title, subtitle, caption, or legend label to our graph, we

can use the options provided by `labs()`. It generally takes the following form:

```
labs(
  color    = "",
  fill     = "",
  title    = "",
  subtitle = "",
  caption  = ""
)
```

- The `color`/`fill` option labels the legend and it is dependent on whether you used COLOR or FILL in the data layer
- `title` adds a main title to the top of the plot
- `subtitle` adds a subtitle to the top of the plot
- `caption` adds a caption to the bottom of the plot

```
ggplot(Pew,
       aes(x = FT_Biden, y = FT_Trump, color = pid3))+
  geom_point(position = position_jitter(1, 1), alpha = 0.5)+
  labs(
    title = "Feelings Towards Donald Trump and Joe Biden",
    subtitle = "March 2021",
    caption = "Data: Pew Research Center
    American Trends Panel Wave 84
    Author: Jennifer Lin",
    color = "Political Party"
  )+
  xlab("Feelings towards Joe Biden")+
  ylab("Feelings towards Donald Trump")
```

A few things to note:

1. Since it is a scatterplot, and I used color in the data layer, my label option for my legend is `color`
2. When you press `ENTER` to break a line in the quotes, it inserts a line break in the outcome.

Exercise: Adding labels to your graph

1. Revisit your code from last class. Copy the results that you have from the data and graph layers onto the code for this week.
2. Using `xlab()`, `ylab()` and `labs()` to add a x axis label, y axis label, title, subtitle, legend label and optional caption.

The Scale Layer

An additional thing that you might want or need to do is adjust the scales. By default, `ggplot` adjusts the scales based on the best fit of the graph from the data presented. However, sometimes, you want to override these settings.

To start, if you want to adjust the limits on continues x or y axes, you can use `xlim()` or `ylim()` with your desired start and end encased in the parentheses.

However, if you want to adjust the scales and legends for continuous and categorical variables, `ggplot`also contains a variety of `scales` functions to help you do just that. These come in the format of `scale_[SOMETHING]_[SOMEHOW]()` and their arguments vary depending on the thing you are scaling and how you are doing it.

Most scales take the following arguments

- `name` = Name the thing you are scaling
- `breaks` = Locate where you want to break it
- `values` = Assign each break point a value (for colors or fills)
- `limits` = Set upper and lower bounds (if applicable)

Here are some examples:

- `scale_x_continuous()`
  - `[SOMETHING]` = x-axis
  - `[SOMEHOW]` = continuously
- `scale_fill_manual()`
  - `[SOMETHING]` = shape fill
  - `[SOMEHOW]` = manually
- `scale_colour_brewer(palette = "[PALETTE NAME]")`
  - `[SOMETHING]` = color
  - `[SOMEHOW]` = Using the R Color Brewer palette

Here are some examples for how to apply these scales:

SCALE COLOR MANUAL

If you are using a `fill`, this situation is the same thing, just change `color` to `fill`

So I want to introduce my own colors to the mix since the R default colors are completely trash. I can do that with `scale_color_manual()`

```
scale_color_manual(
  name = "Party",
  breaks = c("Democrat",  "Republican", "Independent"),
  values = c("Democrat" = "#3182bd", "Republican" = "#de2d26", "Independent" = "#636363")
)
```

Here, I am using the `name`, `breaks` and `values` arguments

- `name` changes my legend label name from the `labs()` argument earlier (or from defaults)
- `breaks` set the categories for the legend. NOTICE that your legend will appear in this order
- `values` allows you to set colors manually. You can list the color ("red", "blue" etc) or use HEX codes (but keep them in quotes!)

SCALE X CONTINUOUS

Now suppose I do not like the fact that the axis labels of my graph are too far apart, and I want to manipulate the continuous scale so that it shows more breaks.

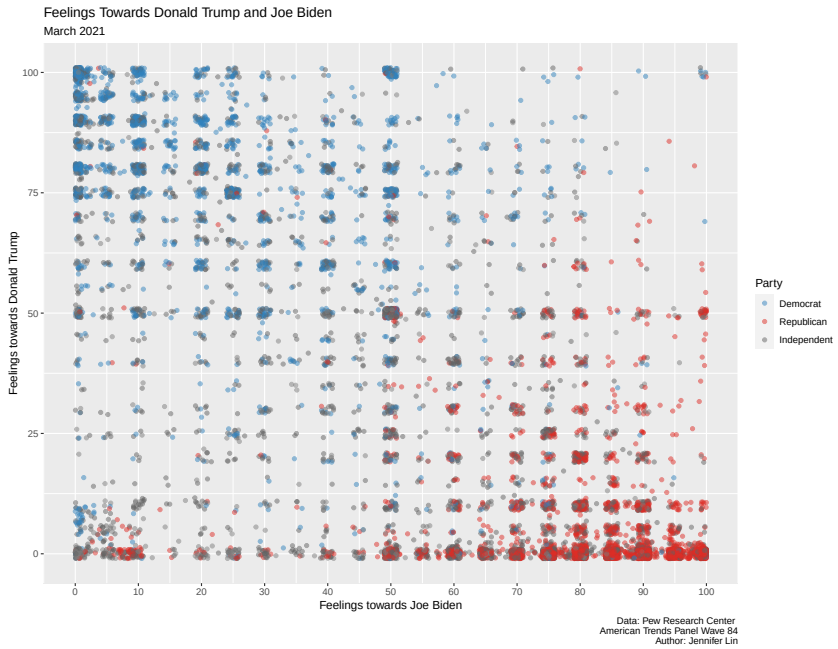
```
  scale_x_continuous(
    breaks = seq(0, 100, 10),
    limits = c(0, 100)
  )+
```

Here, I use the `breaks` and `limits` options

- `breaks` tell R the numbers to include on the axis and how frequently I want to scale to add a new tick for a number. Since I am lazy, I am not writing all 10 numbers. Rather, I use `seq(BEGINNING, END, BY)` to count.
- `limits` tell R to cut the scale off at a certain point. In this case, I am telling it to go for the full scale (0-100)

Here is how everything looks on the graph:

```
ggplot(Pew,
       aes(x = FT_Biden, y = FT_Trump, color = pid3))+
  geom_point(position = position_jitter(1, 1), alpha = 0.5)+
  labs(
    title = "Feelings Towards Donald Trump and Joe Biden",
    subtitle = "March 2021",
    caption = "Data: Pew Research Center
    American Trends Panel Wave 84
    Author: Jennifer Lin",
    color = "Political Party"
  )+
  xlab("Feelings towards Joe Biden")+
  ylab("Feelings towards Donald Trump")+
  scale_color_manual(
    name = "Party",
    breaks = c("Democrat",  "Republican", "Independent"),
    values = c("Democrat" = "#3182bd", "Republican" = "#de2d26", "Independent" = "#636363")
  )+
  scale_x_continuous(
    breaks = seq(0, 100, 10),
    limits = c(0, 100)
  )
```



Exercise: Changing your scales

1. Using the graph that you created in the last exercise, change the scales on your continuous variable
2. Change the color of your fills or lines, either by variable, or revisit your graph layer and change it globally
3. CHALLENGE: Go online and find a HEX code generator and add a color that is not an R default color (i.e. your color should include a `#`).

The Theme Layer

The theme layer can be conceptualized in two broad categories: Global and specific options.

Starting with the global themes, there are many options to change the overall look and feel of your graph. Some of these settings are already loaded in the `ggplot2` package but there are still others that you can get using the `ggthemes` package.

The options built in `ggplot2` include some of the following:

- `theme_bw()`
- `theme_classic()`
- `theme_light()`
- `theme_linedraw()`

Options in `gthemes` expand the ones in `ggplot2` and can include things like:

- `theme_tufte()`
- `theme_gdocs()`
- `theme_calc()`

To use these, simply add them to your `ggplot` code chunks and follow this with a `+` to add other theme options.

Speaking of other theme options, we can use `theme()` to specify things like font colors, sizes, angles and orientations. Options here are based on the *specific* part of the graph that you are looking to customize. A brief synopsis of how to use `theme()` is as follows:

```
theme(
  plot.title      = element_text(hjust  = 0.5),
  plot.subtitle   = element_text(face = "bold")
  axis.title      = element_text(color ="black"),
  axis.title.y    = element_text(size   = 12),
  axis.text.x     = element_text(angle = 45),
  legend.position = 'bottom'
  legend.title    = element_text(family="serif"),
)
```

Where:

- `hjust`: Left (0), Center (0.5), Right (1) justified
- `color`: Color – can use names or HEX codes
- `size`: Font size
- `face`: Takes "plain", "italic", "bold", "bold.italic"
- `font`: Font family – assumes sans serif
- `angle`: Angle of object (0 - 360)

To determine what arguments in `theme()` you need, think about the *specific* graph section you want to change.

These follow the pattern:

`PART OF THE GRAPH + (.) + WHAT ABOUT THIS PART + (.) + ANYTHING ELSE?`

Looking at the `PART OF THE GRAPH`, this aspect generally contains three categories

- `plot.*`: Addresses the entire plot
- `axis.*`: Addresses the axis
- `legend.*`: Addresses the legends

In the `WHAT ABOUT THIS PART`, these components can include things like:

- `title`
- `subtitle`

- `text`
- `caption`
- `background`
- `position`

All that describe the feature that you are seeking to change

Usually, you can append `.x` or `.y` to the argument to specify axis changes. This is usually the only component in the `ANYTHING ELSE?` component.

Putting the theme layer together, it can look something like this

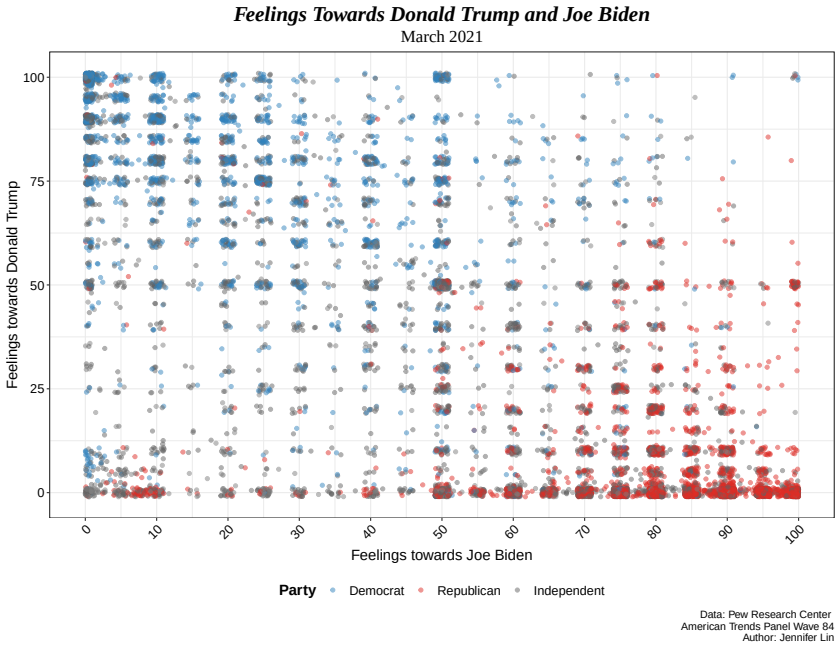
```
theme_bw()+
theme(
  plot.title         = element_text(
    hjust = 0.5, size = 20, colour="black",
    face = "bold.italic", family="serif"),
  plot.subtitle      = element_text(
    hjust = 0.5, size = 16,
    colour="black", family="serif"),
  legend.title       = element_text(
    hjust = 0.5, size = 14,
    colour="black", face = "bold"),
  plot.caption       = element_text(size = 10, colour="black"),
  axis.title         = element_text(size = 14, colour="black"),
  axis.text.x        = element_text(
    size = 12, colour="black",
    angle = 45, hjust = 1),
  axis.text.y        = element_text(size = 12, colour="black"),
  legend.position    = 'bottom',
  legend.direction   = "horizontal",
  legend.text        = element_text(size = 12, colour="black")
)
```

Now, our full graph looks something like so[6]:

[6] This is *not* a publication quality graph as fonts should be in the same font family if submitting to a journal

```
ggplot(Pew,
       aes(x = FT_Biden, y = FT_Trump, color = pid3))+
  geom_point(position = position_jitter(1, 1), alpha = 0.5)+
  labs(
    title = "Feelings Towards Donald Trump and Joe Biden",
    subtitle = "March 2021",
    caption = "Data: Pew Research Center
    American Trends Panel Wave 84
    Author: Jennifer Lin",
    color = "Political Party"
  )+
```

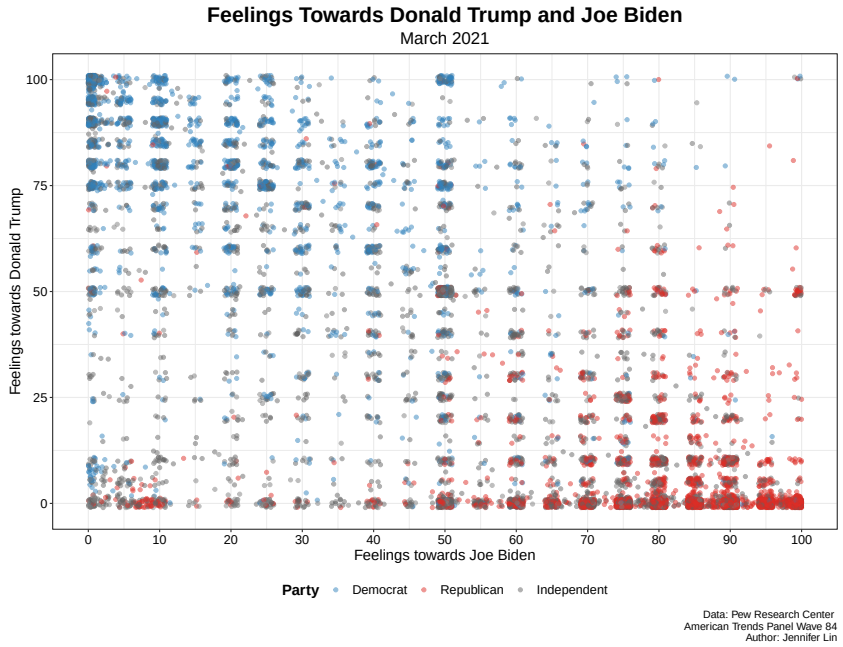
```
xlab("Feelings towards Joe Biden")+
ylab("Feelings towards Donald Trump")+
scale_color_manual(
  name = "Party",
  breaks = c("Democrat",  "Republican", "Independent"),
  values = c("Democrat" = "#3182bd", "Republican" = "#de2d26", "Independent" = "#636363")
)+
scale_x_continuous(
  breaks = seq(0, 100, 10),
  limits = c(0, 100)
)+
theme_bw()+
theme(
  plot.title         = element_text(
    hjust = 0.5, size = 20, colour="black",
    face = "bold.italic", family="serif"),
  plot.subtitle      = element_text(
    hjust = 0.5, size = 16,
    colour="black", family="serif"),
  legend.title       = element_text(
    hjust = 0.5, size = 14,
    colour="black", face = "bold"),
  plot.caption       = element_text(size = 10, colour="black"),
  axis.title         = element_text(size = 14, colour="black"),
  axis.text.x        = element_text(
    size = 12, colour="black",
    angle = 45, hjust = 1),
  axis.text.y        = element_text(size = 12, colour="black"),
  legend.position    = 'bottom',
  legend.direction   = "horizontal",
  legend.text        = element_text(size = 12, colour="black")
)
```



This is not a publication ready graph mainly because of the font differences between the title and the rest of the graph. For teaching purposes, I used different sizes and styles to show you what happens. However, if I were submitting it as part of a paper, here is what it would look like:

```
ggplot(Pew,
       aes(x = FT_Biden, y = FT_Trump, color = pid3))+
  geom_point(position = position_jitter(1, 1), alpha = 0.5)+
  labs(
    title = "Feelings Towards Donald Trump and Joe Biden",
    subtitle = "March 2021",
    caption = "Data: Pew Research Center
    American Trends Panel Wave 84
    Author: Jennifer Lin",
    color = "Political Party"
  )+
  xlab("Feelings towards Joe Biden")+
  ylab("Feelings towards Donald Trump")+
  scale_color_manual(
    name = "Party",
    breaks = c("Democrat",  "Republican", "Independent"),
    values = c("Democrat" = "#3182bd", "Republican" = "#de2d26", "Independent" = "#636363")
  )+
  scale_x_continuous(
    breaks = seq(0, 100, 10),
    limits = c(0, 100)
```

```
)+
theme_bw()+
theme(
  plot.title          = element_text(
    hjust = 0.5, size = 20, colour="black",
    face = "bold"),
  plot.subtitle       = element_text(
    hjust = 0.5, size = 16,
    colour="black"),
  legend.title        = element_text(
    hjust = 0.5, size = 14,
    colour="black", face = "bold"),
  plot.caption        = element_text(size = 10, colour="black"),
  axis.title          = element_text(size = 14, colour="black"),
  axis.text.x         = element_text(
    size = 12, colour="black"),
  axis.text.y         = element_text(size = 12, colour="black"),
  legend.position     = 'bottom',
  legend.direction    = "horizontal",
  legend.text         = element_text(size = 12, colour="black")
)
```



Exercise: Adding Themes to your Plots

1. Take your code from last week and add it into this week's script
2. Add a global theme

3. Add specific theme options to adjust your titles, axis labels and legend

Other Components

There are some other components to `ggplot2` and these include:

- `facet_grid()` and `facet_wrap()` can group your data by a designated grouping variable
- `coord_flip()` changes what is on your x-axis to y-axis and vice versa

But the possibilities of other components in `ggplot2` are not limited to these components, or the things you learned in the past 3 weeks. There is a lot that a module, or a course even, cannot cover for each of your specific research project's graphics needs.

Applications

While the data that we looked at are based in "real data", it is ultimately a cleaned out version of a larger dataset. In this application section, we will look at how to load data from an original source. Also, the application example looks at a spatial model in `ggplot`, which might be useful for an assignment.

The data for this application section come from the DW Nominate scale. This dataset rates members of Congress by their ideology where -1 is the most liberal and 1 is the most conservative. This provides us a spatial model such that we can plot all the members of Congress for any given Congressional session on a number line. The examples thus far are not necessarily spatial and they cover two or more variables. This example departs from this, looking at one variable on a spatial graph, coloring by another variable (party identification).

To start, let's lad the data from its original source. Notice that I did not give you a data file for this part[7]. Here, I build the URL and load the data using `read.csv()`. I also do some light cleaning of the data. Don't worry about understanding this for now.

[7] Notice that `paste0()` combines two strings without a space in the middle. I am using `paste0()` because the URL is a bit long for a script (See script writing notes from Session 1).

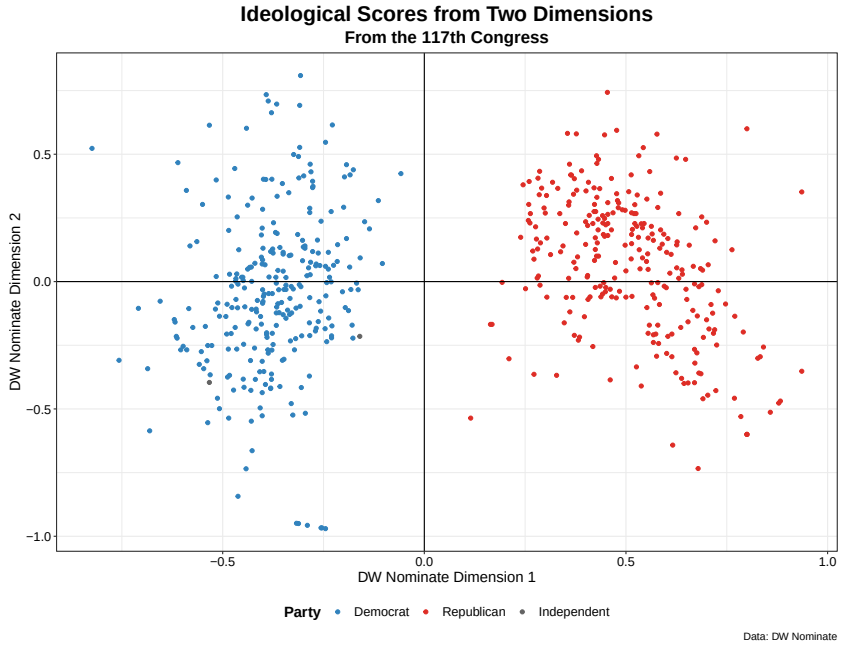
```
# Build the URL
url <- "https://voteview.com/static/data/"
extension <- "out/members/HS117_members.csv"
data_url <- paste0(url, extension)

# Load the Data
N117 <- read.csv(data_url) %>%
  filter(chamber != "President") %>%
```

```
  mutate(
    # Recode party
    party = case_when(
      party_code == 100 ~ "Democrat",
      party_code == 200 ~ "Republican",
      TRUE ~ "Independent"
    )
  )
```

DW Nominate places members of Congress on two dimensions, with the first dimension being the more preferred metric in research. Before considering how to plot data on one dimension, let's quickly revisit what we covered from the previous section and plot both dimensions on a scatter plot. Here, I use `geom_hline()` and `geom_vline()` to draw explicit axes to get the four quadrants that makes the Cartesian coordinate plane.

```
ggplot(N117,
       aes(x = nominate_dim1, y = nominate_dim2,
           color = party))+
  geom_point()+
  geom_hline(yintercept=0)+
  geom_vline(xintercept=0)+
  scale_color_manual(
    name = "Party",
    breaks = c(
      "Democrat",  "Republican", "Independent"),
    values = c(
      "Democrat" = "#3182bd",
      "Republican" = "#de2d26",
      "Independent" = "#636363")
  )+
  xlab("DW Nominate Dimension 1")+
  ylab("DW Nominate Dimension 2")+
  labs(
    title = "Ideological Scores from Two Dimensions",
    subtitle = "From the 117th Congress",
    caption = "Data: DW Nominate"
  )+
  theme_bw()+
  theme(
    plot.title         = element_text(
      hjust = 0.5, size = 20, colour="black", face = "bold"),
    plot.subtitle      = element_text(
      hjust = 0.5, size = 16, colour="black", face = "bold"),
```

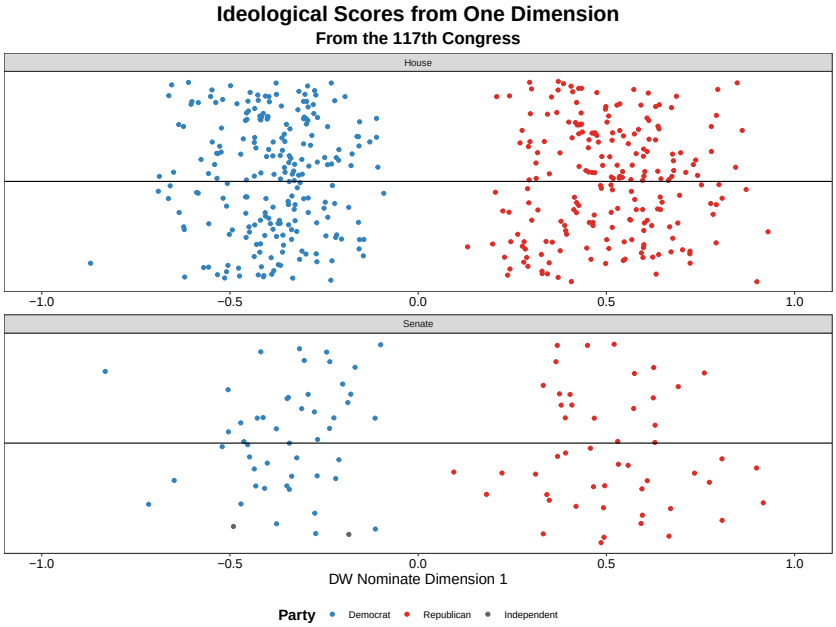
```
  legend.title       = element_text(
    hjust = 0.5, size = 14, colour="black", face = "bold"),
  plot.caption       = element_text(
    size = 10, colour="black"),
  axis.title         = element_text(
    size = 14, colour="black"),
  axis.text.x        = element_text(
    size = 12, colour="black", angle = 0, hjust = 0.5),
  axis.text.y        = element_text(
    size = 12, colour="black"),
  legend.position    = 'bottom',
  legend.direction   = "horizontal",
  legend.text        = element_text(size = 12, colour="black")
)
```



Now, we turn to the singular dimension. The code is rather similar compared to the previous graph. We are not supplying a y variable in the data layer and therefore, do not require settings for y-axis limits in the code.

```
ggplot(N117, aes(x = nominate_dim1, y = 0, color = party))+
  geom_point(position = position_jitter(.1, .5))+
  geom_hline(yintercept=0)+
  xlim(-1, 1)+
  facet_wrap(~chamber, scales = "free_x", ncol = 1)+
  scale_color_manual(
    name = "Party",
    breaks = c(
```

```
    "Democrat",  "Republican", "Independent"),
  values = c(
    "Democrat" = "#3182bd",
    "Republican" = "#de2d26",
    "Independent" = "#636363")
)+
xlab("DW Nominate Dimension 1")+
labs(
  title = "Ideological Scores from One Dimension",
  subtitle = "From the 117th Congress",
  caption = "Data: DW Nominate"
)+
theme_bw()+
theme(
  plot.title          = element_text(
    hjust = 0.5, size = 20, colour="black", face = "bold"),
  plot.subtitle       = element_text(
    hjust = 0.5, size = 16, colour="black", face = "bold"),
  legend.title        = element_text(
    hjust = 0.5, size = 14, colour="black", face = "bold"),
  plot.caption        = element_text(
    size = 10, colour="black"),
  axis.title          = element_text(
    size = 14, colour="black"),
  axis.text.x         = element_text(
    size = 12, colour="black", angle = 0, hjust = 0.5),
  axis.title.y = element_blank(),
  axis.ticks.y = element_blank(),
  axis.text.y = element_blank(),
  plot.background = element_blank(),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  legend.position = "bottom"
)
```


Ideological Scores from One Dimension
From the 117th Congress
House
Senate
−1.0
−0.5
0.0
0.5
1.0
DW Nominate Dimension 1
Party
Democrat
Republican
Independent
Data: DW Nominate