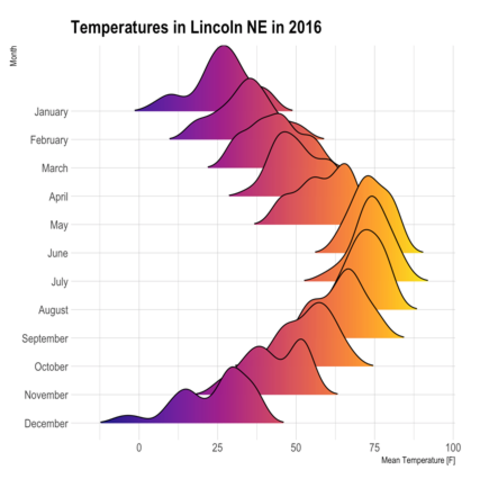
Visualization Challenge

2024-08-20

Now it’s time to try a little visualization challenge!



Look back to the figures we have made. Out of the visualizations we coded, which one do you think displays the data best? The most interesting? The sky is the limit here! You do not have to use a visualization we made, either. If you want to try something else, check out this resource: The [R Graph Gallery](https://www.r-graph-gallery.com/). It has code to help you make all sorts of cool visualizations in R.

Ultimately, choose one type of graph that you want to make. Choose the data you would like to display, by considering a question you’d like to answer from the data.

How do I choose my data? Take a look at the plots we made together in class. Think about what was the most interesting to you. With the Harry Potter data, we looked at aggressions vs. mentions, aggressions per book, and total mentions by top characters. With the friction data, we plotted friction force vs. normal force, position vs. time, and a few other plots. You could also experiment plotting different combinations, or even use a different data set if you’re feeling bold!

***Note:*** The point of this assignment is not to answer some important question or do some complex data wrangling. The point of this assignment is to explore options for plotting beautiful and interesting data visualizations in R.

# Instructions

1. Create a Quarto document in which you will write code to create a beautiful data visualization.
   * Option 1: Make a copy of the day\_3 Quarto notebook, where we loaded the Harry Potter data and did some data wrangling and plotting.
   * Option 2: Create a new Quarto notebook from scratch, and write code chunks to load the Harry Potter data (or other data if you prefer). If you do this, make sure to have embed-resources: true in your Quarto header:

* format:   
   html:  
   embed-resources: true

1. You may keep all the existing code chunks in place, and skip to the end of the document where you will create your data visualization. Optionally, you may delete any text and code chunk in the existing document that is not relevant to your data visualization (always a good idea to keep your analysis tightly focused!).
2. Come up with a question you can answer using a data visualization and the data you have on hand. At the end of the document, write your question, and format it in bold, or perhaps as a heading (using # signs).
3. Based on this question, write a description of the data visualization you plan to make. Consider:
   * What is the independent variable (x axis)? What is the dependent variable (y axis)?
   * Are there any other interesting variables you can include as color or other aesthetics?
   * What type of plot can you use (or possibly multiple types you can try) to help the viewer understand the question?
4. If your question requires some data wrangling (e.g., mutating new columns or group\_by/summarizing existing columns), write the steps you will need to do that.
5. Finally, create a code chunk to create your plot (and wrangle data if necessary).
6. Once you have decided on your graph and plotted it with code, let’s make it publication ready! Check out the advice below to format your visualization.
7. Write a few thoughts about your final plot - what you particularly like, what you wish you could change (even if you’re not sure how to change it), etc.
8. Save your final plot as a .png file (or .jpg or .gif, your choice).
9. Render your Quarto document to a HTML file.

You will submit your saved figure and the rendered HTML file to Canvas.

## Your final figure should include:

* Title
* Human readable x and y-axis labels
* Different colors than the default in ggplot
* Legend with human-readable title

When you are done, save as a .png, .jpg, or .gif (see code below)

# Tips to format your data visualization

## Title, subtitle, and axes labels

**NOTE** All of the code below in code chunks is commented out (on purpose, with a # in front). Remember that in order for R to recognize code, you must remove the # within the code chunk. Do this when you’re ready to write your own code!

# Add a title, a subtitle, and x and y-axis labels   
  
# ggplot(data = <your data here>, aes(<your variables here>)) +  
# geom\_something() +  
# labs(title = "your title here",  
# subtitle = "your subtitle here",  
# x = "what is this variable",  
# y = "what is that variable")  
  
# Center the title and subtitle   
  
# theme(plot.title = element\_text(hjust = 0.5),   
# plot.subtitle = element\_text(hjust = 0.5))  
  
# alternately, hjust = 0 is left justified, and hjust = 1 is right justified

## General themes

Explore different themes that change the general appearance of your visualization.

* theme\_light()
* theme\_gray()
* theme\_bw()
* theme\_minimal()
* theme\_dark()
* theme\_classic()
* theme\_void()

…And many more! Check out [this resource](https://ggplot2.tidyverse.org/reference/ggtheme.html) for more details.

And if you really want to get into the weeds, you can hand-modify pretty much any part of the plot. See [this reference](https://ggplot2.tidyverse.org/reference/theme.html) for more details.

## Explore different color palettes

[Check out this page](https://www.r-graph-gallery.com/ggplot2-color.html) to see how the colors change with different color palettes (scroll to “Changing the Color Scale with ggplot2”)

Some color palettes are included with ggplot, like:

* hue
* manual
* grey

Palettes from [R Color Brewer](https://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3) are also included:

* BuPu
* RdYlBu
* Paired
* PuOr
* Spectral
* Pastel1

Some you will need to install a new package to explore, like ‘viridis’ and ‘paletteer’ (install.packages("viridis"))

* Magma
* Inferno
* Plasma
* viridis
* Cividis
* Nord
* Awtools
* Dutchmasters
* ggsci
* ggthemes

## Legend:

Check out [this great resource](http://www.sthda.com/english/wiki/ggplot2-legend-easy-steps-to-change-the-position-and-the-appearance-of-a-graph-legend-in-r-software) for lots of details how to change your legend.

# How to save as a .png file

Use ggsave() to save the most recent plot as a file. Save your final figure as ‘lastname\_final\_fig.png’ (example: ohara\_final\_fig.png). .png is “portable network graphics”, a great file format for images on the web. Other common ones are .jpg and .gif. The different file types have advantages and disadvantages, but it shouldn’t make much difference here.

You can specify the width and height for a certain look, or leave those alone and let it auto generate.

# ggsave("ohara\_final\_fig.png", width = 7.29, height = 4.5, units = "in")  
  
# ggsave("ohara\_final\_fig.png")