Advanced Data Visualization

Mason Buczek

Components

Part 1 Understand the context

Figure 1:

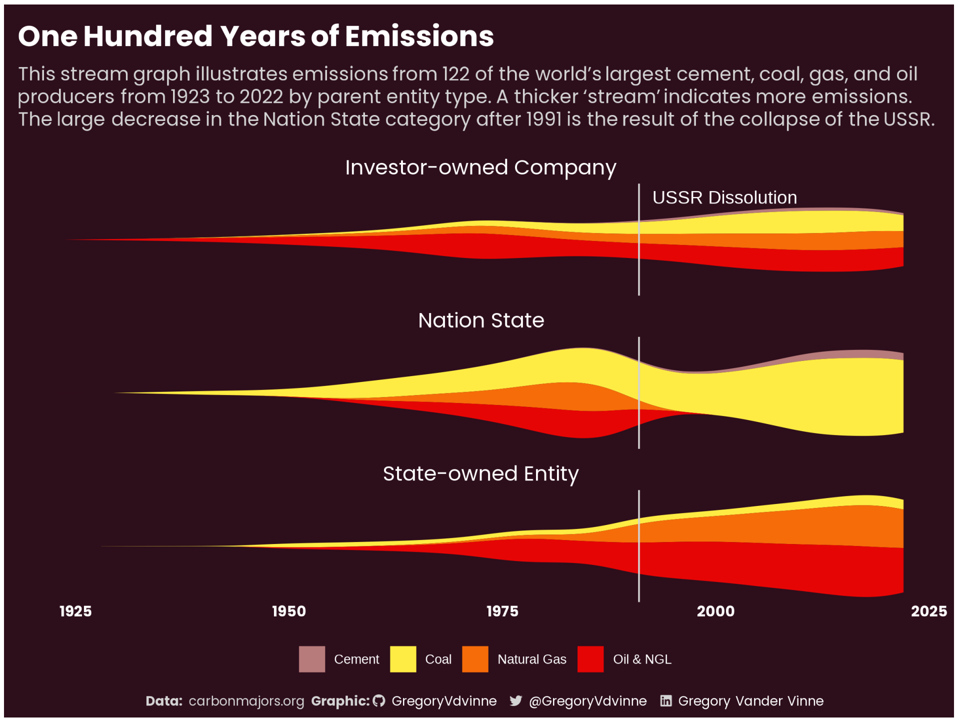


Figure 1: **Figure 1**: One Hundred Years of Emissions

Author's Github: https://github.com/GregoryVdvinne/Tidy_Tuesday/blob/main/2024-05-21/Tidy_Tuesday%202024-05-21.R

What I Like About This Figure: The visualization in this code section impresses me with its clarity and storytelling ability. The use of a stream graph effectively communicates the trends in emissions from major producers over a century. The color scheme is visually appealing and helps differentiate between different commodity types. The addition of text annotations provides valuable context, such as highlighting the impact of the USSR dissolution on emissions. Overall, the visualization effectively conveys complex data but more importantly, presents in an engaging and informative manner.

Packages Used and Their Functions

- **'tidyverse':** A collection of R packages for data manipulation 'dplyr', visualization 'ggplot2', and other tasks.
- **'here'**: Provides a reliable way to construct file paths relative to the project root.
- **'showtext'**: Enables custom fonts in plots.
- **'ggtext'**: Allows for rich text formatting in 'ggplot2' plots.
- **'colorspace'**: For advanced color manipulation.
- **'janitor'**: Contains functions for cleaning data.
- **'camcorder'**: Records the plot-making process into a GIF.
- **'tidytuesdayR'**: Downloads datasets for the Tidy Tuesday project.
- **'paletteer'**: Provides access to various color palettes.
- **'ggstream'**: Specifically for creating stream graphs.
- **'glue'**: For string interpolation and manipulation.

Data Cleaning/Summarizing Steps: The data was loaded using "tidytuesdayR::tt_load()" and he cleaned the commodities containing "Coal" so they were consolidated under a single label "Coal". Emissions data were also grouped by 'year', 'commodity', and 'parent_type', summing up 'total_emissions_MtCO2e' to get total emissions. Also, he filtered the data to include only years from 1923 onwards.

Structure of the Final Data Frame: 'emissions_agg'

- **'year'**: The year of the emissions data.
- **'commodity'**: The type of commodity (e.g., Coal, Oil).
- **'parent_type'**: The type of parent entity (e.g., Investor-owned Company, Nation State).
- **'emissions'**: The summed emissions for the given year, commodity, and parent type.

Geoms Used:

- **'geom_stream()'**: Created the main stream graph showing emissions over time by commodity.
- **'geom_text()'**: Adds text annotations for significant events, such as "USSR Dissolution".
- **'geom_vline()'**: Adds a vertical line at the year 1991 to mark the dissolution of the USSR. I like this one.

Figure 2:

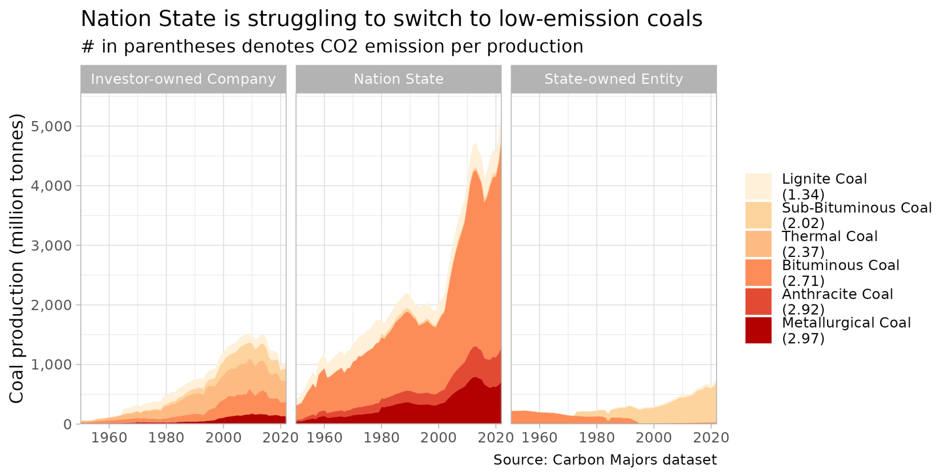


Figure 2: **Figure 2**: Nation is struggling to switch to low-emission coals

Author's Github: https://github.com/mitsuoxv/tidytuesday/blob/main/2024_05_21_carbon_majors_emission_data.qmd

What I Like About This Figure: I like the simplicity and effectiveness in highlighting a specific trend that this visualization has. The use of an area plot to depict coal production over time for different entity types is visually appealing enough to tell the differences between different coal type and it draws attention to the variations in coal production. The title and subtitle provide clear insights into the trend being depicted, while the caption offers additional context. I think this visualization really effectively communicates a single, focused message about coal production dynamics.

Packages Used and Their Functions:

- **'tidyverse'**: A set of R packages for data manipulation and visualization, including 'dplyr' for data manipulation and 'ggplot2' for plotting.
- **'tidytuesdayR'**: Provides easy access to the TidyTuesday datasets.

- **‘glue’**: Used for string interpolation to format text.
- **‘scales’**: Provides functions for formatting and scaling data in plots.

Data Cleaning/Summarizing Steps: The ‘tidytuesdayR’ package was used to load the ‘TidyTuesday’ dataset for the week. They used functions like “glimpse’ and ‘count’ are used to inspect and understand the structure of the dataset. The dataset is grouped by ‘year’, ‘commodity’, and ‘production_unit’ to summarize production values and total emissions. This was then used to calculate the total emissions per unit of production ‘(co2_per_unit)’. This author also filtered to focus on coal-related emissions from 1950 onward. In the grouping by ‘parent_type’, ‘year’, and ‘commodity’; the summarized data includes the number of records, total production value, total emissions, and emissions per unit.

Structure of the Final Data Frame:

- **‘parent_type’**: The type of parent entity (e.g., Investor-owned Company, Nation State).
- **‘year’**: The year of the emissions data.
- **‘commodity’**: The type of commodity (looking at only Coal).
- **‘n’**: The count of records.
- **‘production_value’**: The summed production value for the year, commodity, and parent type.
- **‘total_emissions_MtCO2e’**: The total emissions for the year, commodity, and parent type.
- **‘co2_per_unit’**: The calculated emissions per unit of production.

Geoms Used and Their Appearance in the Final Visualization:

- **‘geom_line()’**: Used in the initial visualization to plot co2_per_unit over time, colored by commodity.
- **‘geom_area’()**: This was used in creating the final visualization of area plots showing the ‘production_value over time for coal’, filled by ‘commodity’. The area plot allows visual comparison of production values across different commodities and parent types.
- **‘facet_wrap()’**: Used to create separate panels for each ‘parent_type’, facilitating comparison between different parent entities.
- **‘scale_y_continuous()’** & ’**scale_x_continuous()’**: These functions format the y-axis with comma separating labels and set custom expansion for both axes.
- **‘scale_fill_brewer()’**: Applies a color palette from the ‘RColorBrewer’ package for filling the areas.

Figure 3:

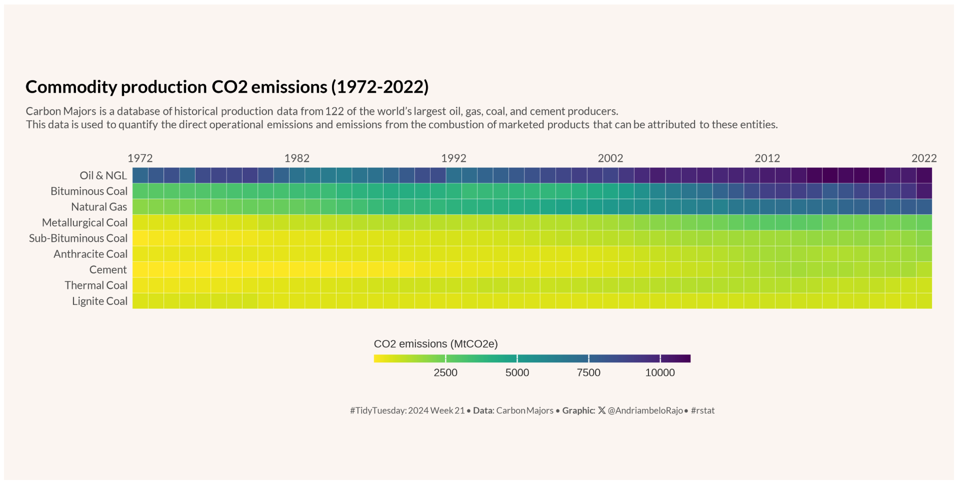


Figure 3: Figure 3: Commodity production CO2 emissions (1972-2022)

Author's Github: https://github.com/R4j0Dm/TidyTuesday/blob/main/w21/week_21_carbon_majors.R

What I Like About This Figure: This last visualization for me stands out to me for its aesthetic appeal and data representation. The use of a heatmap effectively visualizes CO2 emissions by commodity over time. The color scale chosen is perceptually uniform, making it easy to interpret emission intensity. The plot's minimalist design, with clear titles and annotations, enhances readability and comprehension. The inclusion of a caption provides valuable context about the dataset and its source. Overall, I think I liked this one the most followed by figure 1 then figure 2.

Packages Used and Their Functions:

- **'tidyverse'**: Offers tools for data manipulation and visualization.
- **'showtext'**: Facilitates the integration of custom fonts into R graphics.
- **'ggtext'**: Enhances text formatting capabilities within ggplot2.
- **'viridis'**: Provides perceptually uniform color scales for data visualization.

Data Cleaning/Summarizing Steps: Same as the last two I have looked at, this author uses the 'tt_load' function from 'tidytuesdayR' to fetch the 'TidyTuesday' dataset for the 21st week of 2024. While there are data-related operations such as grouping and summarizing to reorder commodities based on emissions, this code is more about data manipulation for visualization purposes rather than data wrangling. Commodities are reordered based on their emissions in the year 2022. This step likely involves calculating total emissions for each commodity in 2022 and ordering them accordingly. They then used the ‘ggplot2’ package to create a heatmap visualization of CO2 emissions by commodity over time and ‘Geoms like geom_tile()’ to represent data points.

Structure of the Final Data Frame:

- **‘year’**: Year of the emissions data.
- **‘commodity’**: Type of commodity, reordered based on total emissions in 2022.
- **‘total_emission’**: Total emissions in MtCO2e for each year and commodity.

Geoms Used and Their Appearance in the Final Visualization:

- **‘geom_tile()’**: Creates a heatmap representing emissions intensity over time for each commodity.
- **‘scale_fill_viridis()’**: Applies a perceptually uniform color scale to the heatmap.
- **‘coord_fixed()’**: Maintains the aspect ratio of the plot.
- **‘scale_y_discrete()’ & ‘scale_x_continuous()’**: Customize the axes for better readability.

Part 2: Make your own visualization using your data

Part 2C: Personal Sketch

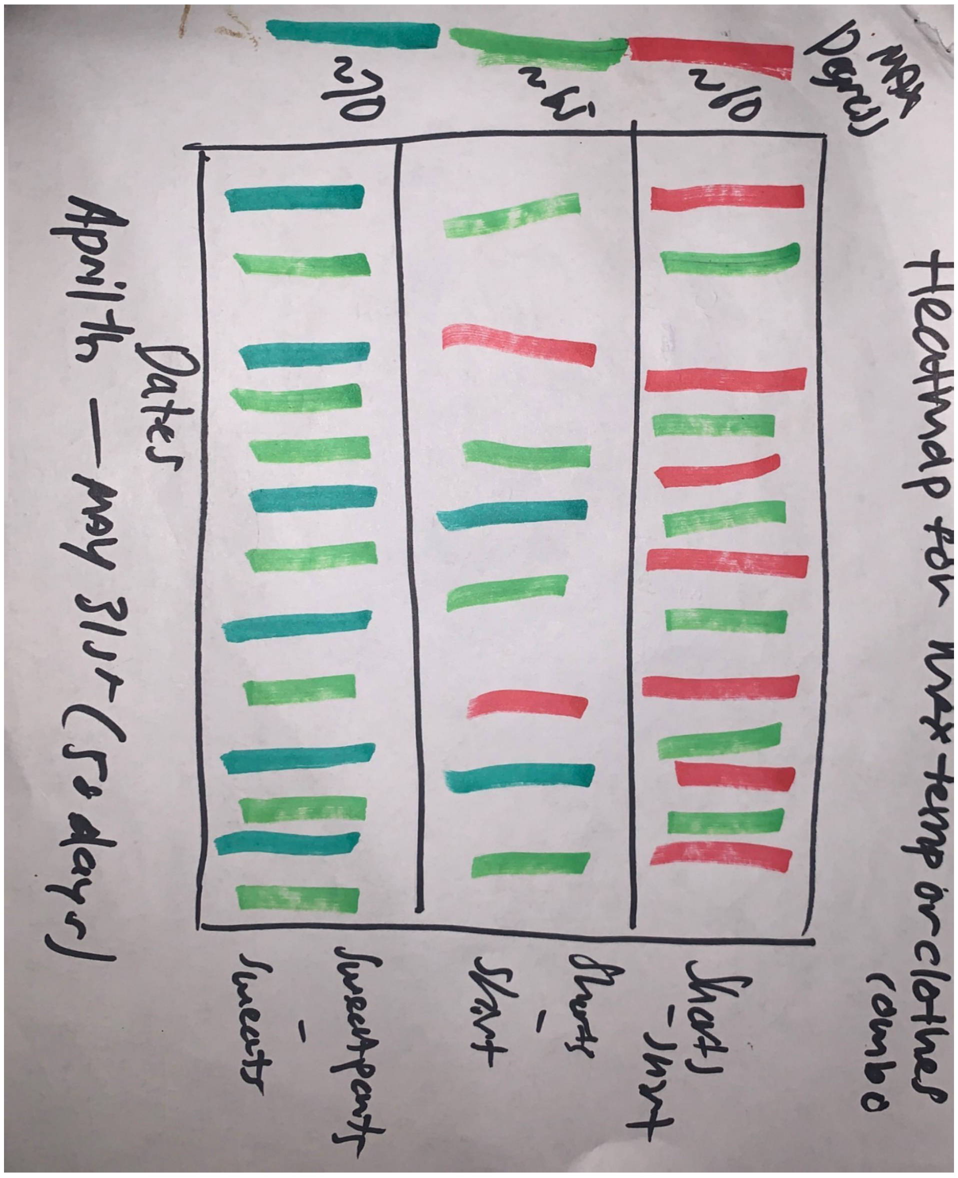


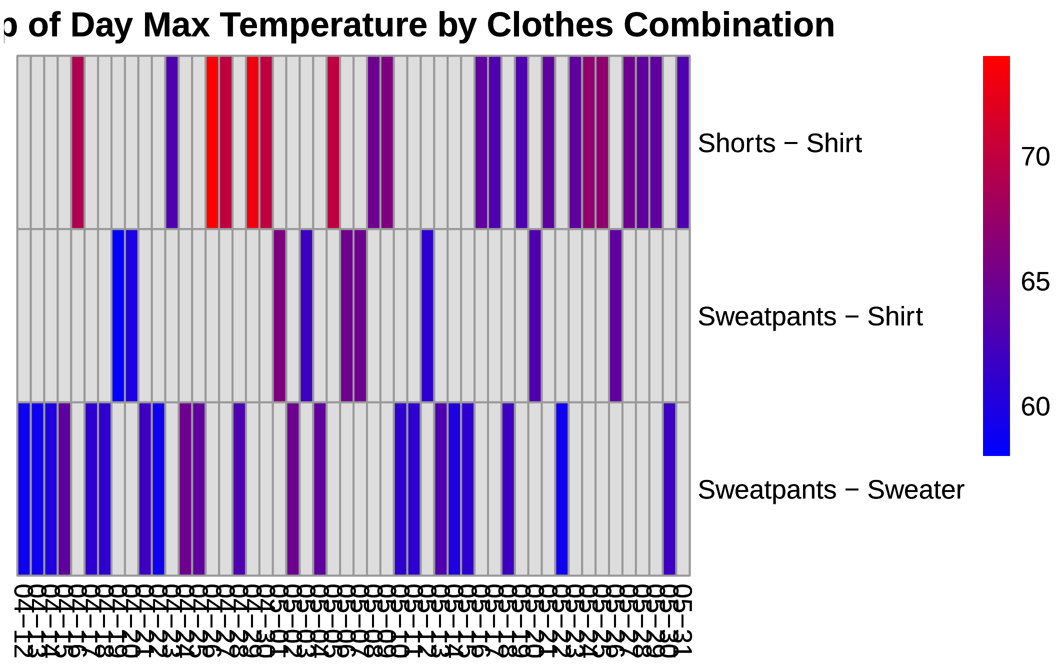
Figure 4: **Figure 4**: Personal Visualization Sketch

Part 2D: Annotated Code and Output

```
# Load libraries
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.3.2
```

```
library(reshape2)
library(readxl)
library(pheatmap)

# Load the data from the Excel file
data <- read_excel("person_data.xlsx")

# Convert Date to Date type and format to remove year
data$Date <- format(as.Date(data$Date, format = "%Y-%m-%d"), "%m-%d")

# Create a column for clothes combination
data$Clothes_Combo <- paste(data$Clothes_Bottom, data$Clothes_Top, sep = " - ")

# Create a matrix for the heatmap
heatmap_data <- dcast(data, Clothes_Combo ~ Date, value.var = "Day_Max_Temp")

# Convert to matrix format for heatmap
heatmap_matrix <- as.matrix(heatmap_data[, -1])
rownames(heatmap_matrix) <- heatmap_data$Clothes_Combo

# Define color palette with blue for lower temperatures and red for higher temperatures
my_palette <- colorRampPalette(c("blue", "red"))(100)

# Plot heatmap with customized color palette
p <- pheatmap(heatmap_matrix,
              cluster_rows = FALSE,
              cluster_cols = FALSE,
              show_rownames = TRUE,
              show_colnames = TRUE,
              main = "Heatmap of Day Max Temperature by Clothes Combination",
              color = my_palette)

# Print the modified plot
print(p)
```



Part 2E: Written Response

In my coding process, I began by loading necessary libraries and importing my personal dataset from an Excel file. The data was structured with dates, clothing combinations, and maximum temperatures. I cleaned the data by converting the date format and creating a combined column for clothing combinations. Initially, I used the 'pheatmap()' function to create a basic heatmap visualization without any additional customization. However, I noticed that the year in the dates was cluttering the visualization, so I reformatted the dates to display only the month and day. This allowed for clearer presentation of the data without losing the temporal aspect. From the previous visualizations in Part 1, my favorite map there was a heat map and I wanted to see if I could implement it im my visualization. I opted for a color palette that transitions from blue to red to symbolize lower and higher temperatures that provides intuitive visual cues for understanding temperature differences. In designing the visualization, I focused on simplicity and clarity. I chose a minimalist approach with clean lines and a clear title to emphasize the main message: the relationship between clothing combinations and daily maximum temperatures. The main takeaway from the visualization is the correlation between clothing choices and daily maximum temperatures over time. By presenting this information in a heatmap format, viewers can easily identify patterns and trends, such as which clothing combinations are preferred in different temperature ranges. You can really see how as temperatures get warmer, there is a general change in preferred clothing combination from Sweatpants-Sweater to Sweatpants-Shirt and finally to Shorts-Shirt.