Case Study

Zahlen Zbinden

2023-11-16

Load Packages

First the author loads all the required packages, and sets global options for the chunks, the global options do not affect the current chunk. Global options display code in output, suppress any messages/warnings, as well as options for the plots, including the width and height of the plots, and in what order to evaluate chunks that contain plots and variables.

Data

Next the author loads and cleans the data in a single pipeline, starting from the first line the following steps are performed.

1. The author loads the data from a local file, I will change this to load the tidy_tuesday data from his github to alleviate the need to download and store the csv file, this will allow the code to be run from any workspace.
2. Drop all rows that contain an NA in the company column
3. Group the data by company
4. Create a column called tot_spend that contains the total amount (sum) of the spending of each company
5. Filter the data to only include companies that have spend more than 65000

```
url = "https://raw.githubusercontent.com/hardin47/TidyTuesday/gh-pages/2022-06-07/contribu
pride <- read_csv(url) |>
            drop_na(Company) |>
            group_by(Company) |>
            mutate(tot_spend = sum(Amount)) |>
            filter(tot_spend >= 65000) |>
            ungroup()
```

Company	Pride and Sponsor Match?	Pride?	HRC Business Pledge	Donor Name	Politician	State	Amount	Date	Citation	Donor Type	Comments	ARCHIVE - Company Manually Determined (May Not Match Pride Sponsors List)	tot_spend
Southern Company (Georgia Power, Alabama Power, Mississippi Power)	Both	TRUE	FALSE	ALABAMA POWER CO. EMPLOYEES STATE PAC	Danny Garrett	AL	1000	2013-08-15	NA	PAC	NA	NA	375500.0
Southern Company (Georgia Power, Alabama Power, Mississippi Power)	Both	TRUE	FALSE	AL POWER CO. EMPLOYEES STATE PAC	Dickie Drake	AL	1000	2013-08-26	NA	PAC	NA	NA	375500.0
AT&T	Both	TRUE	TRUE	AT&T ALABAMA	David Standridge	AL	300	2013-08-31	NA	PAC	NA	NA	307137.5

We will take a step away from the pipeline to get an idea of how the data is currently formated after being ungrouped. We can see that after ungrouping the data, company has been deaggregated, however for each occurance of the company there is still a tot_spend column. (Note: I couldn't find a good way to control the scale of the table, except to make it fit the width, so you will have to zoom in to see the actual details of the table format)

```
pride |>
    head(3) |>
    knitr::kable() |>
    kable_styling(latex_options = "scale_down")
```

Now to continue with the data pipeline (again starting with one as the first line of code in the block)

1. Assign pride data to pride variable
2. Modify the Company column to trim down "Southern Company…" to just "Southern Company Power", do not change any other company names.
3. Modify the Company column as a factor, where each unique company is a factor level, ordered by the total spending, with the company that spent the most at the top of the list, and decreasing expenditure as you go down, and tell the factor function that the levels should be ordered, such that the factor of the company that spend the most is greater than the factor of the company that spend the least. i.e. the factors are ordinal.

```
pride <- pride |>
            mutate(
                Company = ifelse(
                    Company == "Southern Company (Georgia Power, Alabama Power, Mississipp
                    "Southern Company Power",
                    Company
                )
            ) |>
            mutate(
                Company = factor(
                    Company,
                    levels = unique(Company[order(desc(tot_spend))]),
                    ordered = TRUE
                )
            )
```

The next step is to make the levels a 12 characters in length so when they get printed they are

wrapped after 12 characters, as opposed to just printing in one long line, this behavior doesn't change the original column, only changes the behavior when the level is being print such as with print() or during a plot

```
levels(pride$Company) <- stringr::str_wrap(levels(pride$Company), width = 12)
```

Plot Data

Again we will describe the code step by step, breaking it out as necessary.

1. The glue function from the glue package, is used to format and interpolate strings, allowing the user to embed variables or R code into strings. If you are familiar with python, this is essentially an f-string. The library is not loaded in the begining instead the author has opted to only add the glue function to the namespace as oppsed to everything from the glue library and put the function into the variable gg.
2. Create the data set that will be used for the plot, group by Company and then State, and then summarise by count company per state, and create a new tot_spend column which is the total spent by each company per state.

```
gg <- glue::glue

pride_waffle <- pride |>
                    group_by(Company, State) |>
                    summarise(n = n(), tot_spend = sum(Amount))
```

Let's take a look at the pride_waffle dataset format, we can see that there is a column that has been aggregated by the table view of gt() however this column in the dataset would repeat the company name for each group of the state, and then count up the occurance of that company in that state, and sum the spending from that company in that state

```
pride_waffle |>
    head(3) |>
    gt()
```

State	n	tot_spend
Toyota		
AL	3	5000
FL	2	1500
TX	5	595000

The author creates a second data set for their waffle plot by grouping only by company, and creating the tot_spend column as which represents the total spent by each company across all states, and the total number of politicians that were backed in that state.

```
pride_waffle2 <- pride_waffle |>
                    group_by(Company) |>
                    summarise(tot_spend = sum(tot_spend), tot_pol = sum(n))
```

The plot

ggplot2 is used to create the plot, with the add on library of waffle. As this code black is all self contained, I will add comments to the code as the review.

Here is a view of the data that is used in the geom text call

```
tibble(
    idx = c(1:12),
    Company = pride_waffle2$Company[idx],
    y = pride_waffle2$tot_pol[idx] %/% 10,
    actual_spend = pride_waffle2$tot_spend[idx],
    lab = gg("{scales::comma(actual_spend, 1, prefix = '$')} spent")
) |>
    gt()
```

idx	Company	y	actual_spend	lab
1	Toyota	1	601500.0	$601,500 spent
2	Southern Company Power	12	375500.0	$375,500 spent
3	AT&T	13	307137.5	$307,138 spent
4	Charter Communications	5	168077.1	$168,077 spent
5	Enterprise Products Partners	0	160000.0	$160,000 spent
6	Comcast	7	121350.0	$121,350 spent
7	Berkshire Hathaway	0	100000.0	$100,000 spent
8	Valero Energy	1	93500.0	$93,500 spent
9	Amazon	3	87900.0	$87,900 spent
10	FedEx	4	79700.0	$79,700 spent
11	State Farm	6	79550.0	$79,550 spent
12	American Electric Power	0	65000.0	$65,000 spent

```r
pride_waffle |>
    ggplot() + # Pipe the pride_waffle data into ggplot
        geom_waffle( # Call the waffle geometry
            aes(fill = State, values = n), # Map the aesthetics, fill by state,
            # each segment in the waffle represents a politician
            color = "white", # color the borders of the segments white
            size = .25, # Set the size of the borders of the segments
            n_rows = 10, # This sets the number of rows that will contain the segments
            flip = TRUE  # This flips the axis so that the rows are now the columns
        ) +
        scale_x_discrete(limits = "") + # Removes the scale on the x axis such that no
        # ticks are shown, and plot is full width
        scale_y_continuous(
            labels = function(x) x * 10, # Takes the original y labels and
            # multiplies them each by 10
            expand = c(0, 0), # Sets the padding on the plot such that the limits
            # will fill the axis and there is no padding
            limits = c(0, 16) # Sets the y-limits, essentially zooming into the plot
        ) +
        geom_text( # Call text geometry
            data = tibble( # Create tibble for the text data
                idx = c(1: 12), # Column called idx that goes from 1 -> 12
                Company = pride_waffle2$Company[idx], # Gets company from pride waffle,
                # starts with company that spent the most
                y = pride_waffle2$tot_pol[idx] %/% 10, # Takes tot_pol corresponding to
                # company divides by 10 and removes remainder
                actual_spend = pride_waffle2$tot_spend[idx], # actual spend = total spend
                # for each company
                lab = gg("{scales::comma(actual_spend, 1, prefix = '$')} spent")
                # label is the actual spend, formatted with a comma
            ),
            aes(1, y, label = lab), # x=1, y=y, label = lab, position each on the x axis
            # at 1, the y axis at y with label=lab
            vjust = 0, # no vertical adjust
            hjust = 0, # no horizontal adjust
            nudge_y = 2, # nudge the y direction position 2 (up, away from bottom)
            size = 3.5, # size of the text
            lineheight = .875 # sets the line height to 87.5% of default
        ) +
        facet_wrap(~ Company, nrow = 2, strip.position = "bottom") + # facets for each
        # company, 2 rows, label on bottom
```

```
    ggthemes::scale_fill_tableau(name = NULL) + # set the color pallete of the
    # fill to tableau
    coord_equal() + # equal units on both x and y axis, 1 in X is 1 in Y
    labs( # set labels title, subtitle, x axis, y, and caption
        title = "Support for anti-LGBTQ+ politicians",
        subtitle = "number of politicians + $ spent",
        x = "Company",
        y = "",
        caption = "Tidy Tuesday Plot: #hardin47 | Data: Data for Progress"
    ) +
    theme_minimal() + # add the minimal theme to the plot,
    # removes some shading/fluff
    guides(fill = guide_legend(reverse = TRUE)) # Reverse the order of the legend
```