

HW 06 - Money in US politics

Contents

1 Getting started	2
1.1 Warm up	2
1.2 Packages	2
1.3 Data	2
2 Exercises	2
2.1 Data collection via web scraping	2
2.2 Data cleaning	4
2.3 Data visualization and interpretation	5



Figure 1: Photo by Sharon McCutcheon on Unsplash

Every election cycle brings its own brand of excitement – and lots of money. Political donations are of particular interest to political scientists and other researchers studying politics and voting patterns. They are also of interest to citizens who want to stay informed of how much money their candidates raise and where that money comes from.

In the United States, “only American citizens (and immigrants with green cards) can contribute to federal politics, but the American divisions of foreign companies can form political action committees (PACs) and collect contributions from their American employees.”¹

In this assignment we will scrape and work with data from foreign connected PACs that donate to US political

¹Source: Open Secrets - Foreign Connected PACs.

campaigns. First, we will get data foreign connected PAC contributions in the 2022 election cycle. Then, you will use a similar approach to get data such contributions from previous years so that we can examine trends over time.

In order to complete this assignment you will need a Chrome browser with the Selector Gadget extension installed.

1 Getting started

Go to the course GitHub organization and locate your homework repo, clone it in RStudio and open the R Markdown document. Knit the document to make sure it compiles without errors.

1.1 Warm up

Before we introduce the data, let's warm up with some simple exercises. Update the YAML of your R Markdown file with your information, knit, commit, and push your changes. Make sure to commit with a meaningful commit message. Then, go to your repo on GitHub and confirm that your changes are visible in your Rmd **and** md files. If anything is missing, commit and push again.

1.2 Packages

We'll use the **tidyverse** package for much of the data wrangling and visualisation, the **robotstxt** package to check if we're allowed to scrape the data, the **rvest** package for data scraping, and the **scales** package for better formatting of labels on visualisations. These packages are already installed for you. You can load them by running the following in your Console:

```
library(tidyverse)
library(robotstxt)
library(rvest)
library(scales)
```

1.3 Data

This assignment does not come with any prepared datasets. Instead you'll be scraping the data!

2 Exercises

2.1 Data collection via web scraping



The data come from OpenSecrets.org, a “website tracking the influence of money on U.S. politics, and how that money affects policy and citizens’ lives”. This website is hosted by The Center for Responsive Politics, which is a nonpartisan, independent nonprofit that “tracks money in U.S. politics and its effect on elections and public policy.”²

Before getting started, let’s check that a bot has permissions to access pages on this domain.

```
library(robotstxt)
paths_allowed("https://www.opensecrets.org")
## [1] TRUE
```

Our goal is to scrape data for contributions in all election years Open Secrets has data for. Since that means repeating a task many times, let’s first write a function that works on the first page. Confirm it works on a few others. Then iterate it over pages for all years.

Complete the following set of steps in the `scrape-pac.R` file in the `scripts` folder of your repository. This file already contains some starter code to help you out.

- Write a function called `scrape_pac()` that scrapes information from the Open Secrets webpage for foreign-connected PAC contributions in a given year. This function should
 - have one input: the URL of the webpage and should return a data frame.
 - rename variables scraped, using `snake_case` naming.
 - clean up the `Country of Origin/Parent Company` variable with `str_squish()`.
 - add a new column to the data frame for `year`. We will want this information when we ultimately have data from all years, so this is a good time to keep track of it. Our function doesn’t take a `year` argument, but the year is embedded in the URL, so we can extract it out of there, and add it as a new column. Use the `str_sub()` function to extract the last 4 characters from the URL. You will probably want to look at the help for this function to figure out how to specify “last 4 characters”.
- Define the URLs for 2022, 2020, and 2000 contributions. Then, test your function using these URLs as inputs. Does the function seem to do what you expected it to do?
- Construct a vector called `urls` that contains the URLs for each webpage that contains information on foreign-connected PAC contributions for a given year.
- Map the `scrape_pac()` function over `urls` in a way that will result in a data frame called `pac_all`.
- Write the data frame to a csv file called `pac-all.csv` in the `data` folder.

If you haven’t yet done so, now is definitely a good time to commit and push your changes to GitHub with an appropriate commit message (e.g. “Data scraping complete”). Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.

Complete the following set of steps in the `hw-06.Rmd` file in your repository.

1. In your R Markdown file, load `pac-all.csv` and report its number of observations and variables using inline code.

```
# Load the scraped data
pac_all <- read_csv("data/pac-all.csv")

## Rows: 2404 Columns: 6
## -- Column specification -----
## Delimiter: ","
## chr (5): name, country_parent, total, dems, repubs
## dbl (1): year
##
```

²Source: Open Secrets - About.

```
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The dataset contains 2404 observations and 6 variables.

2.2 Data cleaning

In this section we clean the `pac_all` data frame to prepare it for analysis and visualization. We have two goals in data cleaning:

- Separate the `country_parent` into two such that country and parent company appear in different columns for country-level analysis.
- Convert contribution amounts in `total`, `dems`, and `repubs` from character strings to numeric values.

The following exercises walk you through how to make these fixes to the data.

2. Use the `separate()` function to separate `country_parent` into `country` and `parent` columns. Note that country and parent company names are separated by \ (which will need to be specified in your function) and also note that there are some entries where the \ sign appears twice and in these cases we want to only split the value at the first occurrence of \. This can be accomplished by setting the `extra` argument in to "merge" so that the cell is split into only 2 segments, e.g. we want "Denmark/Novo Nordisk A/S" to be split into "Denmark" and "Novo Nordisk A/S". (See help for `separate()` for more on this.) End your code chunk by printing out the top 10 rows of your data frame (if you just type the data frame name it should automatically do this for you).

```
# Separate country_parent into country and parent columns
pac_all <- pac_all %>%
  separate(country_parent, into = c("country", "parent"), sep = "/", extra = "merge")

# Print top 10 rows
pac_all
```

```
## # A tibble: 2,404 x 7
##   name          country parent total dems repubs year
##   <chr>        <chr>   <chr>  <dbl> <dbl>  <dbl> 
## 1 7-Eleven      Japan    Ito-Y~ $8,5~ $1,5~ $7,000 2000
## 2 ABB Group     Switze~ Asea ~ $46,~ $17,~ $28,5~ 2000
## 3 Accenture     UK       Accen~ $75,~ $23,~ $52,9~ 2000
## 4 ACE INA       UK       ACE G~ $38,~ $12,~ $26,0~ 2000
## 5 Acuson Corp (Siemens AG) Germany Sieme~ $2,0~ $2,0~ $0 2000
## 6 Adtranz (DaimlerChrysler) Germany Daiml~ $10,~ $10,~ $500 2000
## 7 AE Staley Manufacturing (Tate & Lyle) UK       Tate ~ $24,~ $10,~ $14,0~ 2000
## 8 AEGON USA (AEGON NV)   Nether~ Aegon~ $58,~ $10,~ $47,7~ 2000
## 9 AIM Management Group   UK       AMVES~ $25,~ $10,~ $15,0~ 2000
## 10 Air Liquide America France  L'Air~ $0     $0     $0 2000
## # i 2,394 more rows
```

3. Remove the character strings including \$ and , signs in the `total`, `dems`, and `repubs` columns and convert these columns to numeric. End your code chunk by printing out the top 10 rows of your data frame (if you just type the data frame name it should automatically do this for you). A couple hints to help you out:

- The \$ character is a special character so it will need to be escaped.
- Some contribution amounts are in the millions (e.g. Anheuser-Busch contributed a total of \$1,510,897 in 2008). In this case we need to remove all occurrences of ,, which we can do by using `str_remove_all()` instead of `str_remove()`.

```

# Clean currency columns and convert to numeric
pac_all <- pac_all %>%
  mutate(
    total = str_remove_all(total, "\\$|,") %>% as.numeric(),
    dems = str_remove_all(dems, "\\$|,") %>% as.numeric(),
    repubs = str_remove_all(repubs, "\\$|,") %>% as.numeric()
  )

# Print top 10 rows
pac_all

## # A tibble: 2,404 x 7
##   name      country parent total  dems repubs year
##   <chr>     <chr>  <dbl> <dbl> <dbl> <dbl>
## 1 7-Eleven Japan   Ito-Y~  8500  1500  7000  2000
## 2 ABB Group  Switze~ Asea ~ 46000 17000 28500 2000
## 3 Accenture UK     Accen~ 75984 23000 52984 2000
## 4 ACE INA   UK     ACE G~ 38500 12500 26000 2000
## 5 Acuson Corp (Siemens AG) Germany Sieme~ 2000 2000 0 2000
## 6 Adtranz (DaimlerChrysler) Germany Daiml~ 10500 10000 500 2000
## 7 AE Staley Manufacturing (Tate & Lyle) UK     Tate ~ 24000 10000 14000 2000
## 8 AEGON USA (AEGON NV) Nether~ Aegon~ 58250 10500 47750 2000
## 9 AIM Management Group UK     AMVES~ 25000 10000 15000 2000
## 10 Air Liquide America France  L'Air~ 0 0 0 2000
## # i 2,394 more rows

```

Now is a good time to knit your document, commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.

2.3 Data visualization and interpretation

4. Create a line plot of total contributions from all foreign-connected PACs in the Canada and Mexico over the years. Once you have made the plot, write a brief interpretation of what the graph reveals. Few hints to help you out:

- Filter for only Canada and Mexico.
- Calculate sum of total contributions from PACs for each year for each country by using a sequence of `group_by()` then `summarise()`.
- Make a plot of total contributions (y-axis) by year (x-axis) where two lines identified by different colours represent each of Canada and Mexico.

```

# Filter for Canada and Mexico, group by year and country, then plot
pac_all %>%
  filter(country %in% c("Canada", "Mexico")) %>%
  group_by(year, country) %>%
  summarise(total_contributions = sum(total, na.rm = TRUE), .groups = "drop") %>%
  ggplot(aes(x = year, y = total_contributions, color = country)) +
  geom_line(size = 1) +
  scale_y_continuous(labels = dollar_format()) +
  labs(
    x = "Year",
    y = "Total Contributions",
    color = "Country",
    title = "Total Contributions from Canada and Mexico PACs Over Time"

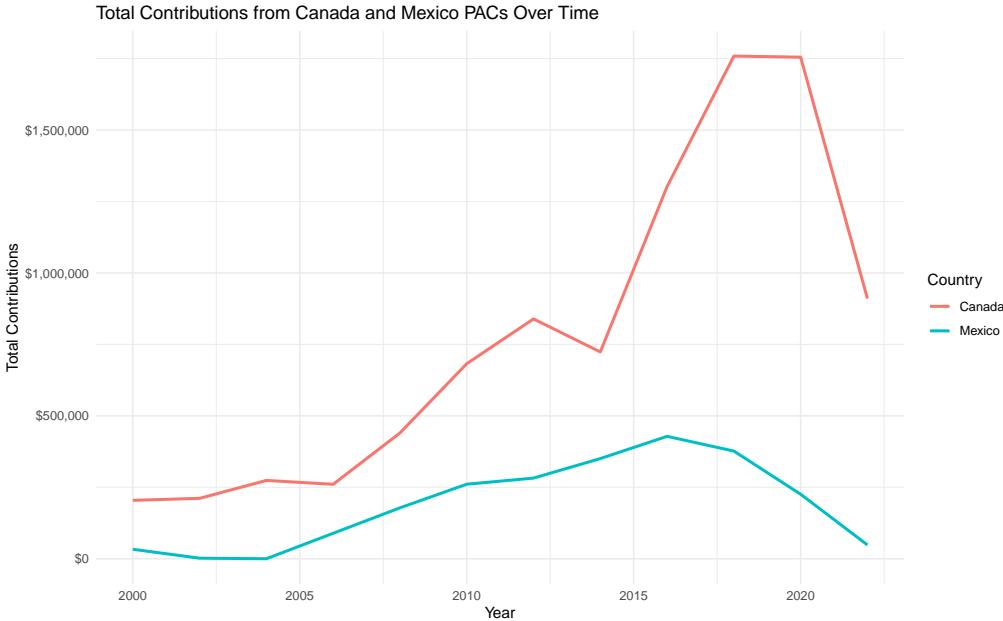
```

```

) +
theme_minimal()

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

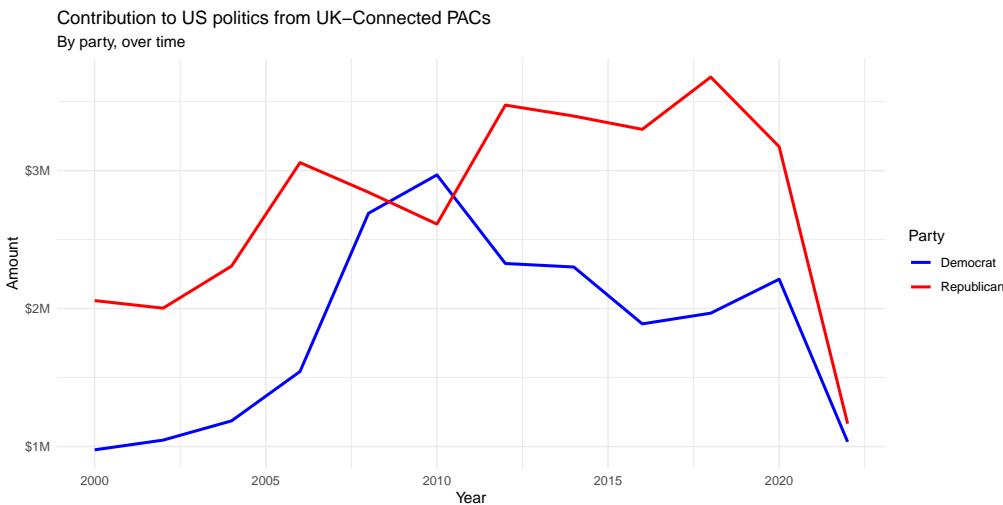
```



The plot shows that contributions from Canadian PACs have generally been higher than those from Mexican PACs over most years. Both countries show variability in their contribution patterns, with some years showing significant spikes in contributions.

****Note:**** The figure you create might look slightly different than this one if the data on the website has changed.

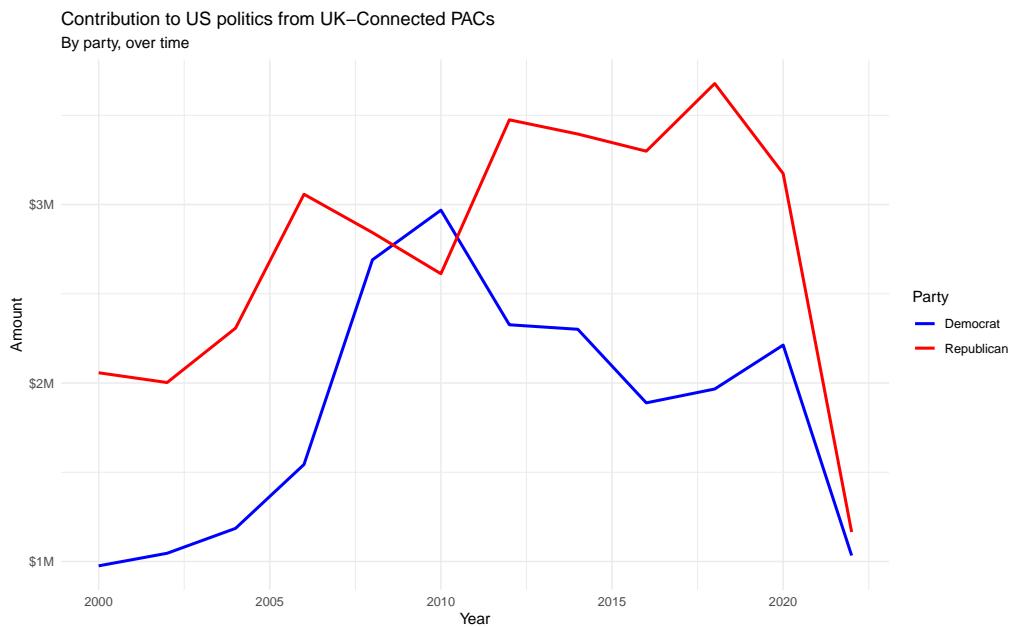
- Recreate the following visualisation. Once you have made the plot, write a brief interpretation of what the graph reveals. Note that these are only UK contributions. You will need to make use of functions from the **scales** package for axis labels as well as from **ggplot2**.



```

# Create UK contributions by party plot
pac_all %>%
  separate(country_parent, into = c("country", "parent"), sep = "/", extra = "merge") %>%
  mutate(
    total = str_remove_all(total, "\\$|,") %>% as.numeric(),
    dems = str_remove_all(dems, "\\$|,") %>% as.numeric(),
    repubs = str_remove_all(repubs, "\\$|,") %>% as.numeric()
  ) %>%
  filter(country == "UK") %>%
  group_by(year) %>%
  summarise(
    Democrat = sum(dems, na.rm = TRUE),
    Republican = sum(repubs, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  pivot_longer(
    cols = c(Democrat, Republican),
    names_to = "party",
    values_to = "amount"
  ) %>%
  ggplot(aes(x = year, y = amount, color = party)) +
  geom_line(size = 1) +
  scale_color_manual(values = c("Democrat" = "blue", "Republican" = "red")) +
  scale_y_continuous(labels = dollar_format(scale = 0.000001, suffix = "M")) +
  labs(
    x = "Year",
    y = "Amount",
    color = "Party",
    title = "Contribution to US politics from UK-Connected PACs",
    subtitle = "By party, over time"
  ) +
  theme_minimal()

```



The plot reveals that UK-connected PACs have generally contributed more to Republican candidates than

Democratic candidates over most election cycles. There are notable fluctuations in both parties' contributions over time, with some years showing particularly high contributions to one party or the other.

Knit, commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards and review the md document on GitHub to make sure you're happy with the final state of your work.