Homework 3: Databases, web scraping, and a basic Shiny app

Isabel Gonzalez

2023-05-31

Table of Contents

# Money in UK politics

[The Westminster Accounts](https://news.sky.com/story/the-westminster-accounts-12786091), a recent collaboration between Sky News and Tortoise Media, examines the flow of money through UK politics. It does so by combining data from three key sources:

1. [Register of Members’ Financial Interests](https://www.parliament.uk/mps-lords-and-offices/standards-and-financial-interests/parliamentary-commissioner-for-standards/registers-of-interests/register-of-members-financial-interests/),
2. [Electoral Commission records of donations to parties](http://search.electoralcommission.org.uk/English/Search/Donations), and
3. [Register of All-Party Parliamentary Groups](https://www.parliament.uk/mps-lords-and-offices/standards-and-financial-interests/parliamentary-commissioner-for-standards/registers-of-interests/register-of-all-party-party-parliamentary-groups/).

You can [search and explore the results](https://news.sky.com/story/westminster-accounts-search-for-your-mp-or-enter-your-full-postcode-12771627) through the collaboration’s interactive database. Simon Willison [has extracted a database](https://til.simonwillison.net/shot-scraper/scraping-flourish) and this is what we will be working with. If you want to read more about [the project’s methodology](https://www.tortoisemedia.com/2023/01/08/the-westminster-accounts-methodology/).

## Open a connection to the database

The database made available by Simon Willison is an SQLite database

sky\_westminster <- DBI::dbConnect(  
 drv = RSQLite::SQLite(),  
 dbname = here::here("data", "sky-westminster-files.db")  
)

How many tables does the database have? 7 tables

DBI::dbListTables(sky\_westminster)

## [1] "appg\_donations" "appgs" "member\_appgs" "members"   
## [5] "parties" "party\_donations" "payments"

## Which MP has received the most amount of money?

You need to work with the payments and members tables and for now we just want the total among all years. To insert a new, blank chunk of code where you can write your beautiful code (and comments!), please use the following shortcut: Ctrl + Alt + I (Windows) or cmd + option + I (mac)

## Reference the tables "payments" and "members" from R  
  
  
payments\_db <- dplyr::tbl(sky\_westminster, "payments")  
payments\_db

## # Source: table<payments> [?? x 13]  
## # Database: sqlite 3.41.2 [/Users/isabelglezb/Desktop/dsb2023-isabel/data/sky-westminster-files.db]  
## category category\_name charity date date\_visited description  
## <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 4. Visits outside the UK Gifts and ot… "" Regi… "Dates of v… Internatio…  
## 2 2. (b) Any other suppor… Cash donatio… "" Regi… "" Use of a h…  
## 3 4. Visits outside the UK Gifts and ot… "" Regi… "Dates of v… Flights £1…  
## 4 2. (a) Support linked t… Cash donatio… "" Regi… "" 2000   
## 5 3. Gifts, benefits and … Gifts and ot… "" Regi… "" Two box ti…  
## 6 2. (b) Any other suppor… Cash donatio… "" Regi… "" 1800   
## 7 2. (b) Any other suppor… Cash donatio… "" Regi… "" 10000   
## 8 4. Visits outside the UK Gifts and ot… "" Regi… "Dates of v… Flights an…  
## 9 5. Gifts and benefits f… Gifts and ot… "" Regi… "" Guest at Q…  
## 10 4. Visits outside the UK Gifts and ot… "" Regi… "Dates of v… Flights, a…  
## # ℹ more rows  
## # ℹ 7 more variables: destination\_of\_visit <chr>, entity <chr>, hours <chr>,  
## # id <chr>, member\_id <chr>, purpose\_of\_visit <chr>, value <dbl>

members\_db <- dplyr::tbl(sky\_westminster, "members") %>%   
 rename(member\_id = id)  
members\_db

## # Source: SQL [?? x 7]  
## # Database: sqlite 3.41.2 [/Users/isabelglezb/Desktop/dsb2023-isabel/data/sky-westminster-files.db]  
## member\_id name gender constituency party\_id short\_name status  
## <chr> <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 m8 Theresa May F Maidenhead p4 Mrs May active  
## 2 m1508 Sir Geoffrey Cox M Torridge and We… p4 Sir Geoff… active  
## 3 m1423 Boris Johnson M Uxbridge and So… p4 Mr Johnson active  
## 4 m4514 Keir Starmer M Holborn and St … p15 Mr Starmer active  
## 5 m1211 Andrew Mitchell M Sutton Coldfield p4 Mr Mitche… active  
## 6 m3958 Fiona Bruce F Congleton p4 Ms Bruce active  
## 7 m14 John Redwood M Wokingham p4 Mr Redwood active  
## 8 m4483 Rishi Sunak M Richmond (Yorks) p4 Mr Sunak active  
## 9 m4097 Liz Truss F South West Norf… p4 Ms Truss active  
## 10 m188 Ed Davey M Kingston and Su… p17 Mr Davey active  
## # ℹ more rows

payments\_members\_db <- payments\_db %>%   
 left\_join(members\_db, "member\_id")  
payments\_members\_db

## # Source: SQL [?? x 19]  
## # Database: sqlite 3.41.2 [/Users/isabelglezb/Desktop/dsb2023-isabel/data/sky-westminster-files.db]  
## category category\_name charity date date\_visited description  
## <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 4. Visits outside the UK Gifts and ot… "" Regi… "Dates of v… Internatio…  
## 2 2. (b) Any other suppor… Cash donatio… "" Regi… "" Use of a h…  
## 3 4. Visits outside the UK Gifts and ot… "" Regi… "Dates of v… Flights £1…  
## 4 2. (a) Support linked t… Cash donatio… "" Regi… "" 2000   
## 5 3. Gifts, benefits and … Gifts and ot… "" Regi… "" Two box ti…  
## 6 2. (b) Any other suppor… Cash donatio… "" Regi… "" 1800   
## 7 2. (b) Any other suppor… Cash donatio… "" Regi… "" 10000   
## 8 4. Visits outside the UK Gifts and ot… "" Regi… "Dates of v… Flights an…  
## 9 5. Gifts and benefits f… Gifts and ot… "" Regi… "" Guest at Q…  
## 10 4. Visits outside the UK Gifts and ot… "" Regi… "Dates of v… Flights, a…  
## # ℹ more rows  
## # ℹ 13 more variables: destination\_of\_visit <chr>, entity <chr>, hours <chr>,  
## # id <chr>, member\_id <chr>, purpose\_of\_visit <chr>, value <dbl>, name <chr>,  
## # gender <chr>, constituency <chr>, party\_id <chr>, short\_name <chr>,  
## # status <chr>

most\_paid\_MP <- payments\_members\_db %>%   
 group\_by(name) %>%   
 summarise(total\_value = sum(value)) %>%   
 arrange(desc(total\_value)) %>%   
 collect()

## Warning: Missing values are always removed in SQL aggregation functions.  
## Use `na.rm = TRUE` to silence this warning  
## This warning is displayed once every 8 hours.

most\_paid\_MP

## # A tibble: 595 × 2  
## name total\_value  
## <chr> <dbl>  
## 1 Theresa May 2809765.  
## 2 Sir Geoffrey Cox 2191387.  
## 3 Boris Johnson 1282402   
## 4 Keir Starmer 799936.  
## 5 Andrew Mitchell 769373.  
## 6 Fiona Bruce 712321.  
## 7 John Redwood 692438.  
## 8 Rishi Sunak 546043   
## 9 Liz Truss 538678.  
## 10 Ed Davey 441681.  
## # ℹ 585 more rows

## Any entity that accounts for more than 5% of all donations?

Is there any entity whose donations account for more than 5% of the total payments given to MPs over the 2020-2022 interval? Who are they and who did they give money to?

donations\_entity\_top5per <- payments\_members\_db %>%   
 group\_by(entity) %>%   
 summarise(total\_value = sum(value)) %>%   
 mutate(percent\_total\_don = total\_value / sum(total\_value)\*100) %>%   
 filter(percent\_total\_don > 5)   
donations\_entity\_top5per

## # Source: SQL [1 x 3]  
## # Database: sqlite 3.41.2 [/Users/isabelglezb/Desktop/dsb2023-isabel/data/sky-westminster-files.db]  
## entity total\_value percent\_total\_don  
## <chr> <dbl> <dbl>  
## 1 Withers LLP 1812732. 5.25

beneficiaries <- payments\_members\_db %>%   
 semi\_join(donations\_entity\_top5per, "entity") %>%   
 group\_by(name) %>%   
 summarise(entity, total\_value = sum(value)) %>%   
 collect()  
beneficiaries

## # A tibble: 1 × 3  
## name entity total\_value  
## <chr> <chr> <dbl>  
## 1 Sir Geoffrey Cox Withers LLP 1812732.

## Do entity donors give to a single party or not?

* How many distinct entities who paid money to MPS are there?
* How many (as a number and %) donated to MPs belonging to a single party only?

#number of distinct entities  
distinct\_entities <- payments\_db %>%   
 distinct(entity) %>%   
 count() %>%   
 collect()  
  
distinct\_entities

## # A tibble: 1 × 1  
## n  
## <int>  
## 1 2213

#How many (as a number and %) donated to MPs belonging to a single party only?  
  
parties\_db <- dplyr::tbl(sky\_westminster, "parties")  
  
payments\_members\_parties\_db <- payments\_members\_db %>%   
 left\_join(parties\_db, by = c("party\_id" = "id"))   
  
party\_donations <- payments\_members\_parties\_db %>%   
 group\_by(entity, name.y) %>%   
 summarise(  
 total\_donations = sum(value,  
 na.rm = TRUE)) %>%   
 mutate(prop = total\_donations / sum(total\_donations)) %>%   
 mutate(single\_party = ifelse(prop == 1, "TRUE", "FALSE")) %>%   
 collect()

## `summarise()` has grouped output by "entity". You can override using the  
## `.groups` argument.

single\_party\_count <- party\_donations %>%  
 filter(single\_party == "TRUE") %>%   
 count() %>%   
 collect()  
single\_party\_count

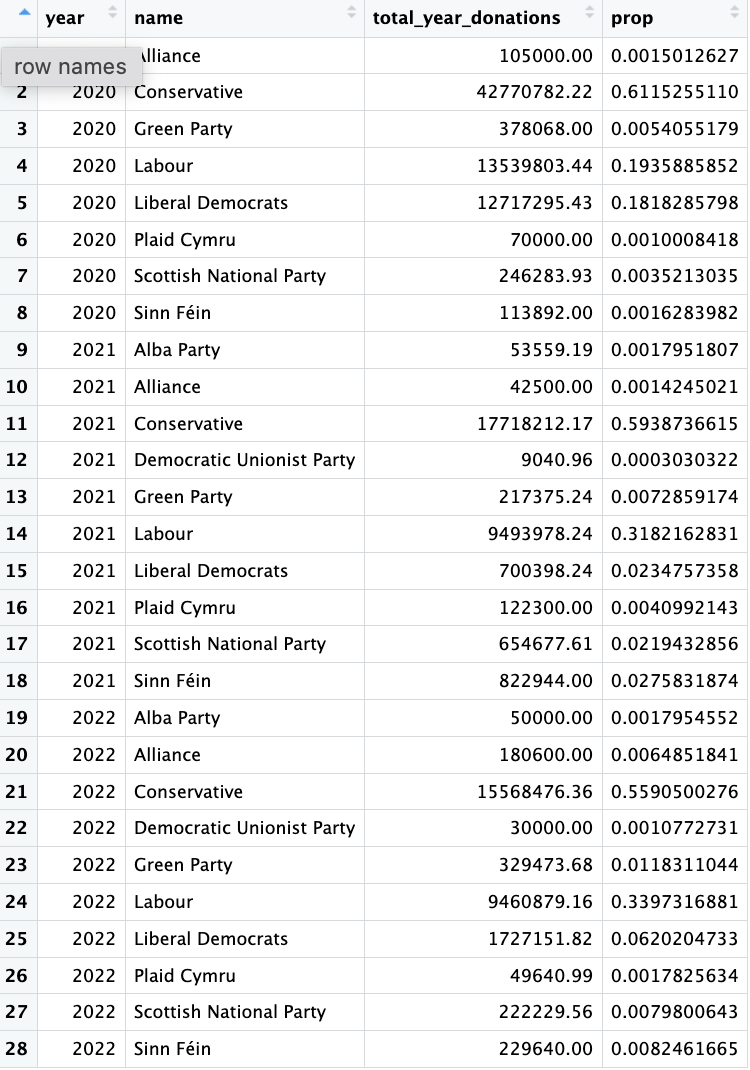
## # A tibble: 2,036 × 2  
## # Groups: entity [2,036]  
## entity n  
## <chr> <int>  
## 1 12 Property FE 1  
## 2 1912 Club 1  
## 3 39th Street Strategies LLC 1  
## 4 3V International 1  
## 5 5 Oceans Partnership 1  
## 6 5x15 1  
## 7 79 Borough Road (trading as 'The Ministry') 1  
## 8 89UP 1  
## 9 8hwe 1  
## 10 97 Dining Club 1  
## # ℹ 2,026 more rows

## I couldn't calculate the proportion

## Which party has raised the greatest amount of money in each of the years 2020-2022?

I would like you to write code that generates the following table.

knitr::include\_graphics(here::here("images", "total\_donations\_table.png"), error = FALSE)



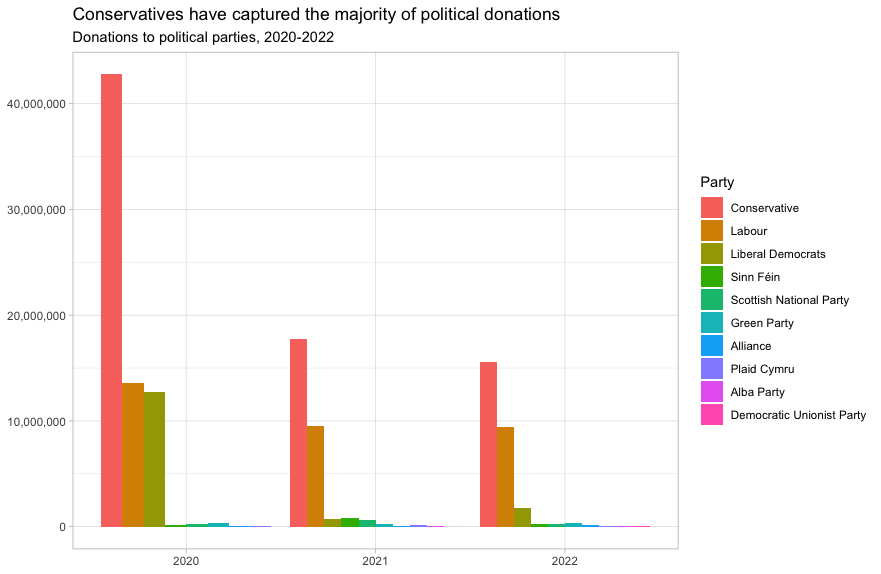
# From the table payments\_members\_parties\_db obtain -> Year, Party, Donations  
  
donations\_by\_party <- payments\_members\_parties\_db %>%   
 mutate(year = str\_sub(date, -4, -1)) %>% #str\_sub -> extracts the date   
 filter(year %in% c("2020", "2021", "2022")) %>%   
 group\_by(year, name.y) %>%  
 summarise(donations = sum(value))  
  
donations\_by\_year <- donations\_by\_party %>%   
 group\_by(year) %>%   
 summarise(year\_total\_donations = sum(donations))  
  
donations\_by\_party\_year <-  
 left\_join(x = donations\_by\_party, y = donations\_by\_year, by = "year") %>%   
 mutate(prop = donations / year\_total\_donations) %>%   
 select(year, name = name.y, total\_year\_donations = donations, prop) %>%   
 collect()

## `summarise()` has grouped output by "year". You can override using the  
## `.groups` argument.  
## `summarise()` has grouped output by "year". You can override using the  
## `.groups` argument.

donations\_by\_party\_year

## # A tibble: 33 × 4  
## # Groups: year [3]  
## year name total\_year\_donations prop  
## <chr> <chr> <dbl> <dbl>  
## 1 2020 Alba Party 1320 0.000125  
## 2 2020 Conservative 6035344. 0.571   
## 3 2020 Democratic Unionist Party 5715. 0.000540  
## 4 2020 Green Party 9500 0.000898  
## 5 2020 Independent 230103. 0.0218   
## 6 2020 Labour 3615844. 0.342   
## 7 2020 Liberal Democrats 537694 0.0508   
## 8 2020 Plaid Cymru 23072 0.00218   
## 9 2020 Scottish National Party 108599. 0.0103   
## 10 2020 Sinn Féin 1911 0.000181  
## # ℹ 23 more rows

… and then, based on this data, plot the following graph.

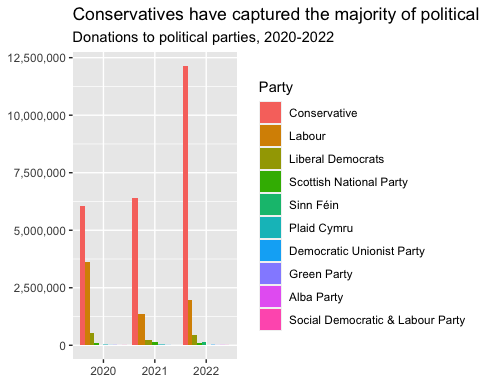


This uses the default ggplot colour pallete, as I dont want you to worry about using the [official colours for each party](https://en.wikipedia.org/wiki/Wikipedia:Index_of_United_Kingdom_political_parties_meta_attributes). However, I would like you to ensure the parties are sorted according to total donations and not alphabetically. You may even want to remove some of the smaller parties that hardly register on the graph. Would facetting help you?

# using the table created donations\_by\_party\_year, create a new table with only the top 10 parties that received donatons and removing "independent"   
  
top10\_party <- donations\_by\_party\_year %>%   
 filter(name != "Independent") %>%   
 group\_by(name) %>%   
 summarise(donations = sum(total\_year\_donations)) %>%  
 slice\_max(n = 10, donations)  
  
top10\_party\_year <-  
semi\_join(x = donations\_by\_party\_year ,y = top10\_party ,by = "name")   
  
top10\_party\_year

## # A tibble: 27 × 4  
## # Groups: year [3]  
## year name total\_year\_donations prop  
## <chr> <chr> <dbl> <dbl>  
## 1 2020 Alba Party 1320 0.000125  
## 2 2020 Conservative 6035344. 0.571   
## 3 2020 Democratic Unionist Party 5715. 0.000540  
## 4 2020 Green Party 9500 0.000898  
## 5 2020 Labour 3615844. 0.342   
## 6 2020 Liberal Democrats 537694 0.0508   
## 7 2020 Plaid Cymru 23072 0.00218   
## 8 2020 Scottish National Party 108599. 0.0103   
## 9 2020 Sinn Féin 1911 0.000181  
## 10 2020 Social Democratic & Labour Party 4900 0.000463  
## # ℹ 17 more rows

#plot a bar graph with year in the x-axis and donations in the y-axis, with the color property being the party   
  
top10\_party\_year %>%   
 ggplot(aes(x = year, y = total\_year\_donations, fill = reorder(name, -total\_year\_donations))) +   
 geom\_col(position = "dodge") + #make bars not stacked but next to each other  
 labs(x = NULL, y = NULL, title = "Conservatives have captured the majority of political donations", subtitle = "Donations to political parties, 2020-2022", fill = "Party") +  
 scale\_y\_continuous(labels = comma)



Finally, when you are done working with the databse, make sure you close the connection, or disconnect from the database.

dbDisconnect(sky\_westminster)

# Anonymised Covid patient data from the CDC

We will be using a dataset with [anonymous Covid-19 patient data that the CDC publishes every month](https://data.cdc.gov/Case-Surveillance/COVID-19-Case-Surveillance-Public-Use-Data-with-Ge/n8mc-b4w4). The file we will use was released on April 11, 2023, and has data on 98 million of patients, with 19 features. This file cannot be loaded in memory, but luckily we have the data in parquet format and we will use the {arrow} package.

## Obtain the data

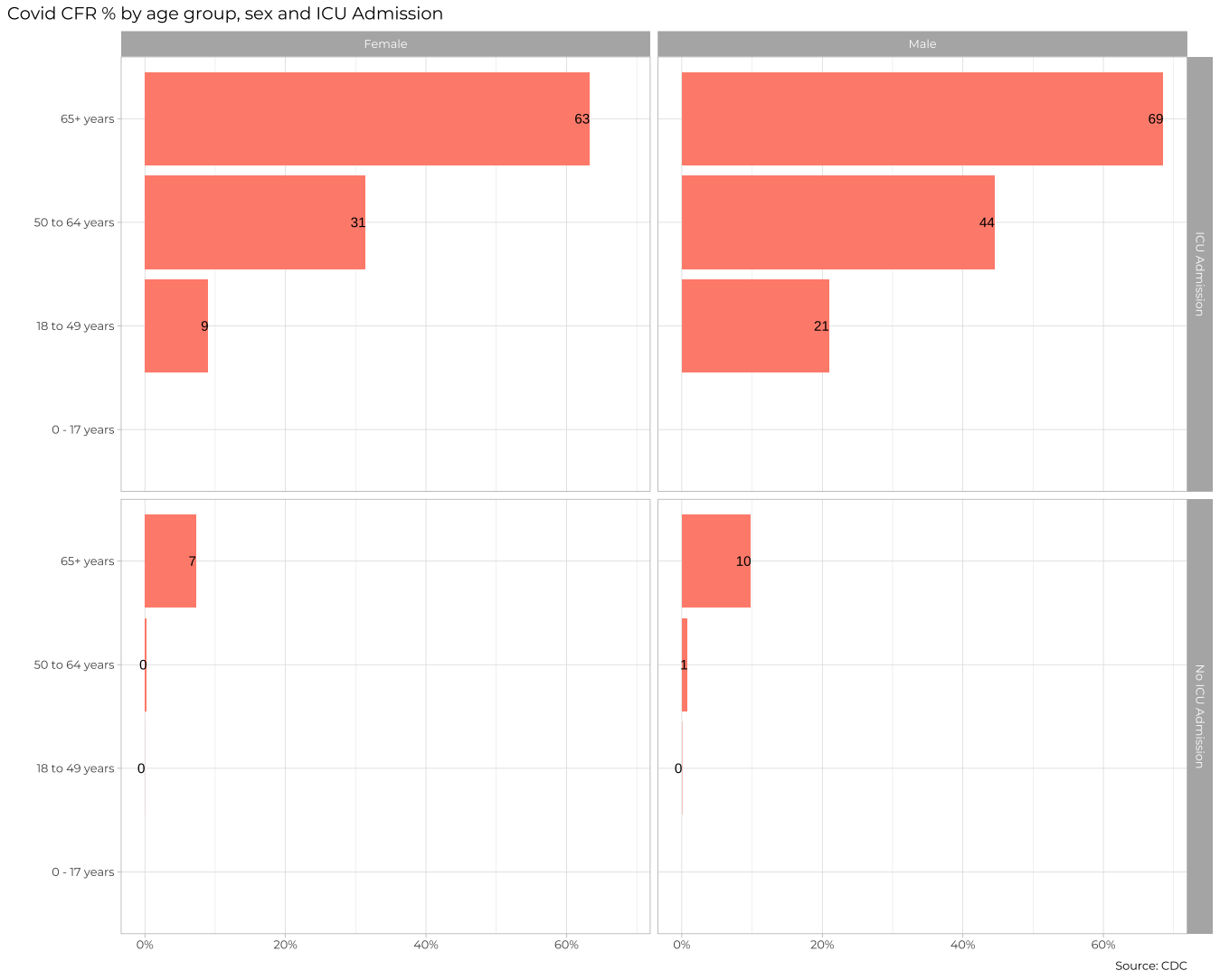
The dataset cdc-covid-geography in in parquet format that {arrow}can handle. It is > 600Mb and too large to be hosted on Canvas or Github, so please download it from dropbox <https://www.dropbox.com/sh/q1yk8mmnbbrzavl/AAAxzRtIhag9Nc_hODafGV2ka?dl=0> and save it in your dsb repo, under the data folder

## 0.013 sec elapsed

## FileSystemDataset with 1 Parquet file  
## 97,799,772 rows x 19 columns  
## $ case\_month <string> "2021-09", "2022-09", "2022-01", "2020…  
## $ res\_state <string> "TX", "TX", "TX", "CA", "IL", "CA", "N…  
## $ state\_fips\_code <int32> 48, 48, 48, 6, 17, 6, 36, 36, 36, 53, …  
## $ res\_county <string> "TARRANT", NA, "HARRIS", "SAN BERNARDI…  
## $ county\_fips\_code <int32> 48439, NA, 48201, 6071, 17031, 6085, 3…  
## $ age\_group <string> "18 to 49 years", "18 to 49 years", "1…  
## $ sex <string> "Male", "Male", "Female", "Female", "F…  
## $ race <string> "White", "White", "Unknown", "Asian", …  
## $ ethnicity <string> "Non-Hispanic/Latino", "Non-Hispanic/L…  
## $ case\_positive\_specimen\_interval <int32> NA, NA, NA, NA, 0, NA, 0, 0, 0, 0, 0, …  
## $ case\_onset\_interval <int32> NA, NA, -1, NA, 0, NA, NA, NA, NA, 0, …  
## $ process <string> "Missing", "Missing", "Missing", "Miss…  
## $ exposure\_yn <string> "Missing", "Missing", "Missing", "Miss…  
## $ current\_status <string> "Laboratory-confirmed case", "Probable…  
## $ symptom\_status <string> "Missing", "Missing", "Symptomatic", "…  
## $ hosp\_yn <string> "Missing", "Missing", "No", "No", "No"…  
## $ icu\_yn <string> "Missing", "Missing", "Missing", "Miss…  
## $ death\_yn <string> "Missing", "Missing", "Missing", "Miss…  
## $ underlying\_conditions\_yn <string> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…

Can you query the database and replicate the following plot?

knitr::include\_graphics(here::here("images", "covid-CFR-ICU.png"), error = FALSE)



cdc\_data\_db <-cdc\_data %>%   
 group\_by(age\_group, sex, icu\_yn, death\_yn) %>%   
 summarise(number\_cases = n()) %>%   
 collect()  
  
cdc\_data\_db

## # A tibble: 427 × 5  
## # Groups: age\_group, sex, icu\_yn [103]  
## age\_group sex icu\_yn death\_yn number\_cases  
## <chr> <chr> <chr> <chr> <int>  
## 1 18 to 49 years Male Missing Missing 11257544  
## 2 18 to 49 years Female Missing Missing 13706049  
## 3 0 - 17 years Female Missing Missing 4055476  
## 4 50 to 64 years Female Missing Missing 4520812  
## 5 0 - 17 years Male Missing Missing 4139694  
## 6 50 to 64 years Male Missing Missing 3876407  
## 7 50 to 64 years Female Missing No 2378237  
## 8 0 - 17 years Male Missing No 2015531  
## 9 65+ years Female Missing No 1693218  
## 10 65+ years Female Missing Missing 2892222  
## # ℹ 417 more rows

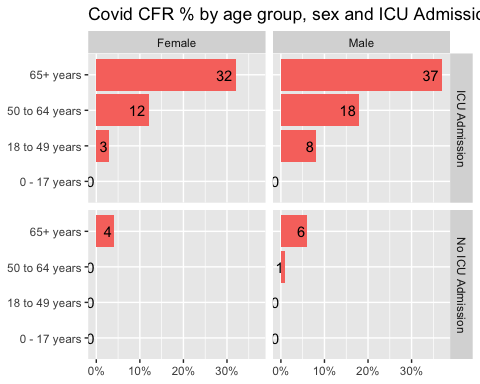
cdc\_data\_clean <- cdc\_data\_db %>%  
 filter(!age\_group %in% c("Missing", NA), sex %in% c("Female", "Male"), icu\_yn %in% c("No", "Yes")) %>%   
 group\_by(age\_group, sex, icu\_yn) %>%   
 summarise(number\_deaths = sum(number\_cases[death\_yn == "Yes"], na.rm = TRUE), number\_cases = sum(number\_cases), percent\_cfr = round(number\_deaths / number\_cases \* 100,0)) %>%  
 mutate(icu\_yn = case\_when(icu\_yn == "Yes" ~ "ICU Admission", icu\_yn == "No" ~ "No ICU Admission"))

## `summarise()` has grouped output by 'age\_group', 'sex'. You can override using  
## the `.groups` argument.

cdc\_data\_clean

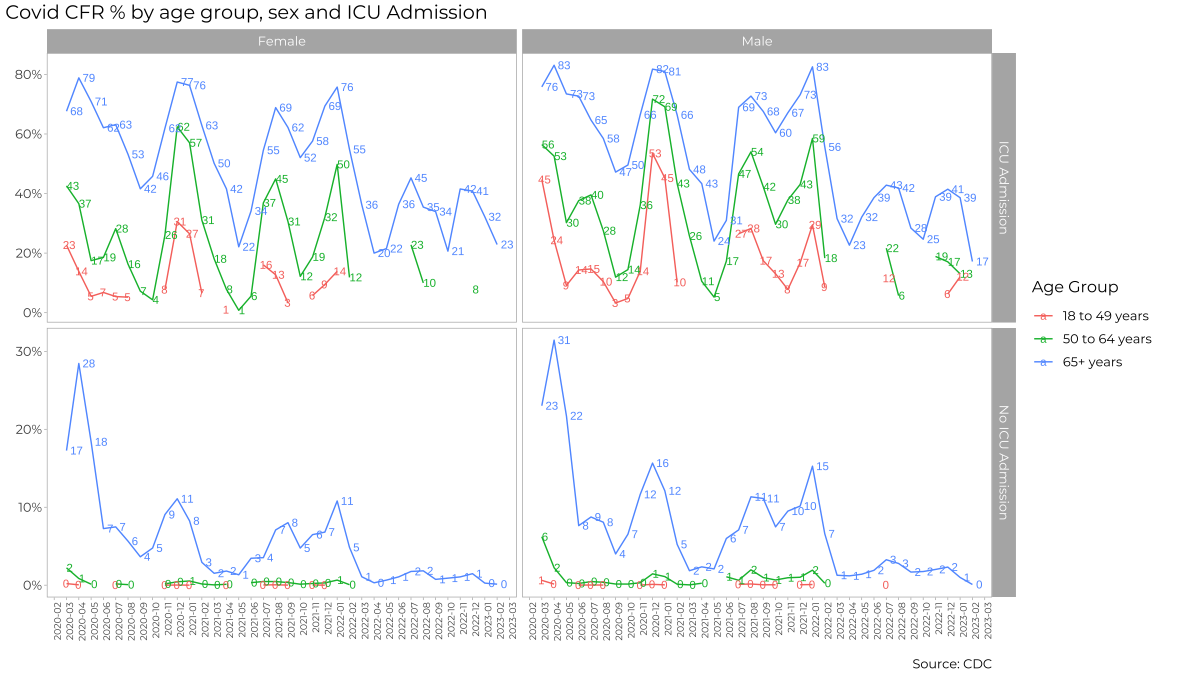
## # A tibble: 16 × 6  
## # Groups: age\_group, sex [8]  
## age\_group sex icu\_yn number\_deaths number\_cases percent\_cfr  
## <chr> <chr> <chr> <int> <int> <dbl>  
## 1 0 - 17 years Female No ICU Admission 0 225584 0  
## 2 0 - 17 years Female ICU Admission 0 1878 0  
## 3 0 - 17 years Male No ICU Admission 0 229700 0  
## 4 0 - 17 years Male ICU Admission 0 2230 0  
## 5 18 to 49 years Female No ICU Admission 58 726042 0  
## 6 18 to 49 years Female ICU Admission 515 14897 3  
## 7 18 to 49 years Male No ICU Admission 190 573257 0  
## 8 18 to 49 years Male ICU Admission 1722 22072 8  
## 9 50 to 64 years Female No ICU Admission 495 270583 0  
## 10 50 to 64 years Female ICU Admission 2607 21998 12  
## 11 50 to 64 years Male No ICU Admission 1270 238528 1  
## 12 50 to 64 years Male ICU Admission 6585 35941 18  
## 13 65+ years Female No ICU Admission 11866 266668 4  
## 14 65+ years Female ICU Admission 13800 42466 32  
## 15 65+ years Male No ICU Admission 13223 228072 6  
## 16 65+ years Male ICU Admission 21502 57990 37

cdc\_data\_clean %>%   
 ggplot(aes(x= age\_group, y=percent\_cfr/100, fill= "orange")) +   
 geom\_col(show.legend =FALSE) +  
 coord\_flip() + #turn axes   
 labs(x = NULL, y = NULL, title = "Covid CFR % by age group, sex and ICU Admission") +  
 geom\_text(aes(label = percent\_cfr, hjust = 1.2)) +  
 facet\_grid(rows = vars(icu\_yn),cols = vars(sex))+   
 scale\_y\_continuous(labels = scales::percent\_format(accuracy = 1))



The previous plot is an aggregate plot for all three years of data. What if we wanted to plot Case Fatality Ratio (CFR) over time? Write code that collects the relevant data from the database and plots the following

knitr::include\_graphics(here::here("images", "cfr-icu-overtime.png"), error = FALSE)



cdc\_data\_date\_db <- cdc\_data %>%  
 group\_by(case\_month, age\_group, sex, icu\_yn, death\_yn) %>%   
 summarise(number\_cases = n()) %>%   
 collect()  
  
cdc\_data\_date\_db

## # A tibble: 11,287 × 6  
## # Groups: case\_month, age\_group, sex, icu\_yn [3,023]  
## case\_month age\_group sex icu\_yn death\_yn number\_cases  
## <chr> <chr> <chr> <chr> <chr> <int>  
## 1 2021-09 18 to 49 years Male Missing Missing 393216  
## 2 2022-09 18 to 49 years Male Missing Missing 131026  
## 3 2022-01 18 to 49 years Female Missing Missing 2702579  
## 4 2020-12 0 - 17 years Female Missing Missing 193999  
## 5 2022-10 50 to 64 years Female Missing Missing 65216  
## 6 2022-07 0 - 17 years Male Missing Missing 126341  
## 7 2021-12 50 to 64 years Male Missing Missing 261150  
## 8 2020-04 50 to 64 years Female Missing No 34813  
## 9 2022-08 0 - 17 years Male Missing Missing 130759  
## 10 2021-12 18 to 49 years Female Missing Missing 1204423  
## # ℹ 11,277 more rows

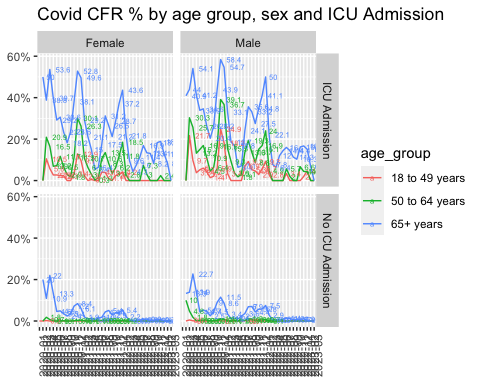
cdc\_data\_date\_clean <- cdc\_data\_date\_db %>%  
 filter(!age\_group %in% c("Missing", NA, "0 - 17 years"), sex %in% c("Female", "Male"), icu\_yn %in% c("No", "Yes")) %>%   
 group\_by(case\_month, age\_group, sex, icu\_yn) %>%   
summarise(number\_deaths = sum(number\_cases[death\_yn == "Yes"], na.rm = TRUE), number\_cases = sum(number\_cases), percent\_cfr = round(number\_deaths / number\_cases \* 100, 1)) %>%  
 mutate(icu\_yn = case\_when(icu\_yn == "Yes" ~ "ICU Admission", icu\_yn == "No" ~ "No ICU Admission"))

## `summarise()` has grouped output by 'case\_month', 'age\_group', 'sex'. You can  
## override using the `.groups` argument.

cdc\_data\_date\_clean

## # A tibble: 457 × 7  
## # Groups: case\_month, age\_group, sex [229]  
## case\_month age\_group sex icu\_yn number\_deaths number\_cases percent\_cfr  
## <chr> <chr> <chr> <chr> <int> <int> <dbl>  
## 1 2020-01 18 to 49 years Fema… No IC… 0 16 0  
## 2 2020-02 18 to 49 years Fema… No IC… 0 9 0  
## 3 2020-02 18 to 49 years Fema… ICU A… 0 7 0  
## 4 2020-02 18 to 49 years Male No IC… 0 12 0  
## 5 2020-02 18 to 49 years Male ICU A… 0 12 0  
## 6 2020-02 50 to 64 years Fema… No IC… 0 10 0  
## 7 2020-02 50 to 64 years Fema… ICU A… 0 12 0  
## 8 2020-02 50 to 64 years Male No IC… 1 10 10  
## 9 2020-02 50 to 64 years Male ICU A… 0 24 0  
## 10 2020-02 65+ years Fema… No IC… 5 25 20  
## # ℹ 447 more rows

cdc\_data\_date\_clean %>%   
 ggplot( aes(y = percent\_cfr / 100, x = case\_month, color = age\_group, group = age\_group)) +  
 geom\_line() +  
 facet\_grid(rows = vars(icu\_yn), cols = vars(sex)) +  
 scale\_y\_continuous(labels = scales::percent\_format(accuracy = 1)) +  
 labs(x = NULL, y = NULL, title = "Covid CFR % by age group, sex and ICU Admission") +  
 geom\_text(data = . %>% filter(percent\_cfr != 0), aes(label = percent\_cfr, hjust = -0.4), size = 2) +  
 theme(axis.text.x = element\_text(angle = 90))



For each patient, the dataframe also lists the patient’s states and county [FIPS code](https://en.wikipedia.org/wiki/Federal_Information_Processing_Standard_state_code). The CDC also has information on the [NCHS Urban-Rural classification scheme for counties](https://www.cdc.gov/nchs/data_access/urban_rural.htm)

urban\_rural <- read\_xlsx(here::here("data", "NCHSURCodes2013.xlsx")) %>%   
 janitor::clean\_names()   
  
glimpse(urban\_rural)

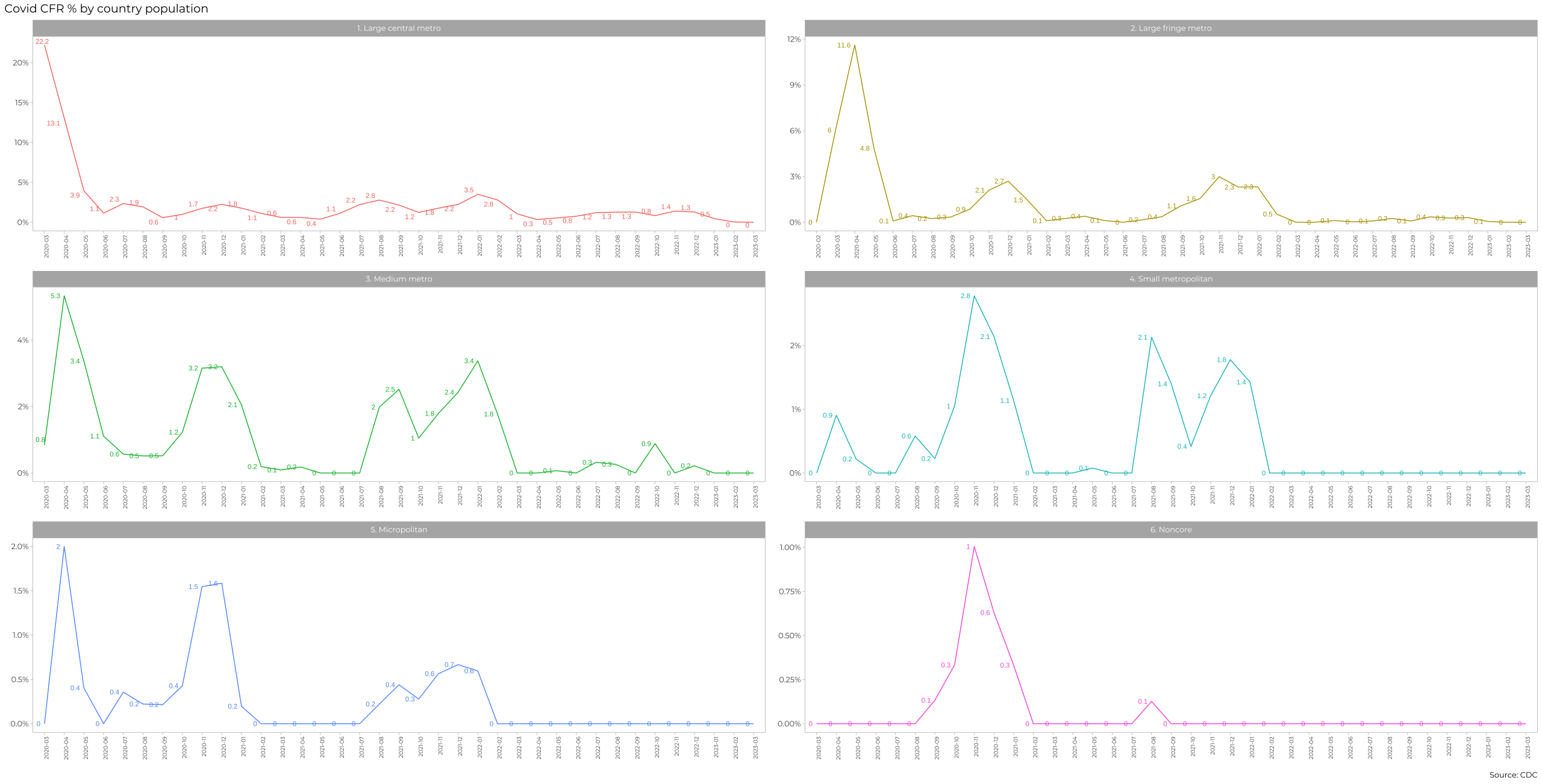
## Rows: 3,149  
## Columns: 9  
## $ fips\_code <dbl> 1001, 1003, 1005, 1007, 1009, 1011, 1013, 1015, 1017,…  
## $ state\_abr <chr> "AL", "AL", "AL", "AL", "AL", "AL", "AL", "AL", "AL",…  
## $ county\_name <chr> "Autauga County", "Baldwin County", "Barbour County",…  
## $ cbsa\_title <chr> "Montgomery, AL", "Daphne-Fairhope-Foley, AL", NA, "B…  
## $ cbsa\_2012\_pop <chr> "377149", "190790", ".", "1136650", "1136650", ".", "…  
## $ county\_2012\_pop <chr> "55514", "190790", "27201", "22597", "57826", "10474"…  
## $ x2013\_code <dbl> 3, 4, 6, 2, 2, 6, 6, 4, 5, 6, 2, 6, 6, 6, 6, 5, 4, 6,…  
## $ x2006\_code <dbl> 3, 5, 5, 2, 2, 6, 6, 4, 5, 6, 2, 6, 6, 6, 6, 5, 4, 6,…  
## $ x1990\_based\_code <chr> "3", "3", "5", "6", "3", "6", "6", "4", "6", "6", "6"…

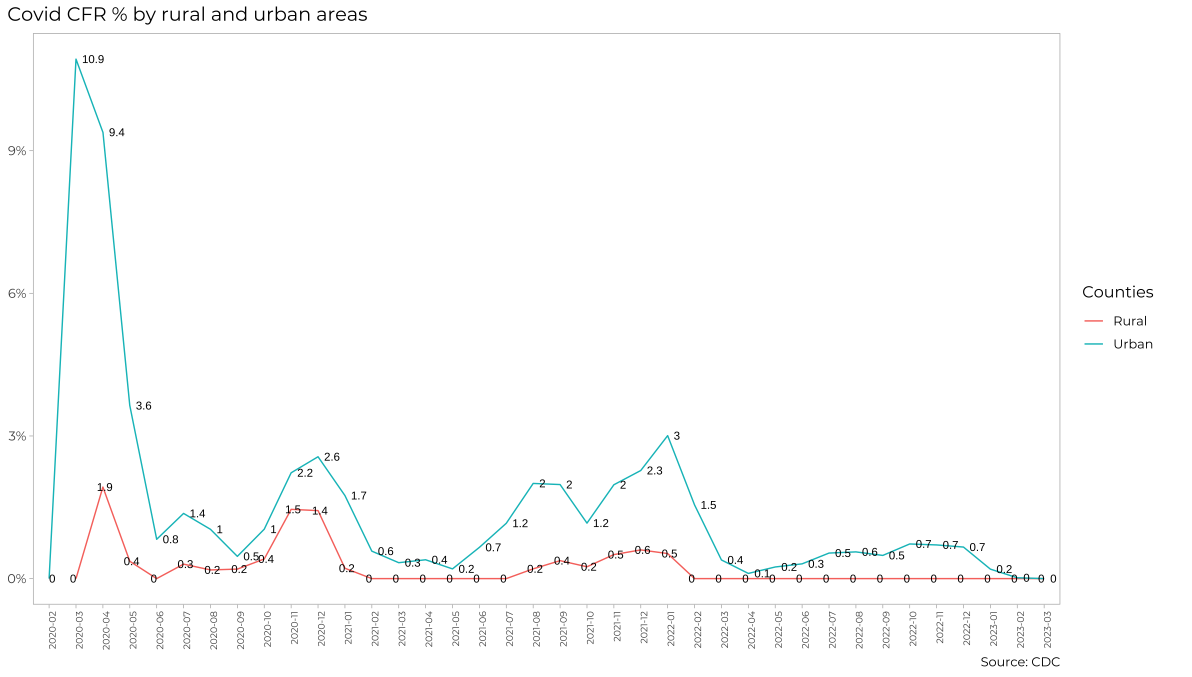
Each county belongs in sex different categories, with categories 1-4 being urban areas and categories 5-6 being rural, according to the following criteria captured in x2013\_code

Category name

1. Large central metro - 1 million or more population and contains the entire population of the largest principal city
2. large fringe metro - 1 million or more poulation, but does not qualify as 1
3. Medium metro - 250K - 1 million population
4. Small metropolitan population < 250K
5. Micropolitan
6. Noncore

Can you query the database, extract the relevant information, and reproduce the following two graphs that look at the Case Fatality ratio (CFR) in different counties, according to their population?





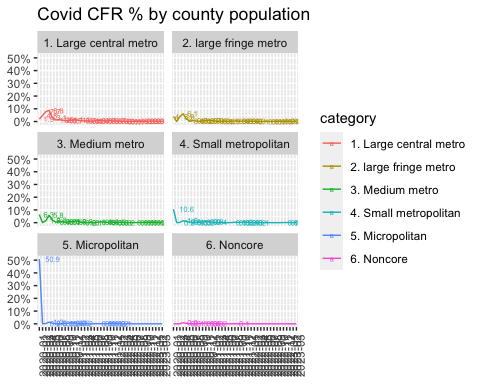
# COVID CFR % by county population graph   
  
cdc\_data\_codes\_db <- cdc\_data %>%  
 group\_by(case\_month, county\_fips\_code, death\_yn) %>%   
 summarise(number\_cases = n()) %>%   
 collect()  
  
 cdc\_data\_codes <- cdc\_data\_codes\_db %>%   
 inner\_join(y = select(urban\_rural, fips\_code, x2013\_code), by = c("county\_fips\_code" = "fips\_code"))  
  
cdc\_data\_codes\_clean <- cdc\_data\_codes %>%  
 mutate(category = case\_when(x2013\_code == 1 ~ "1. Large central metro", x2013\_code == 2 ~ "2. large fringe metro", x2013\_code == 3 ~ "3. Medium metro", x2013\_code == 4 ~ "4. Small metropolitan", x2013\_code == 5 ~ "5. Micropolitan", x2013\_code == 6 ~ "6. Noncore")) %>%   
 group\_by(case\_month, category) %>%  
 summarise(number\_deaths = sum(number\_cases[death\_yn == "Yes"], na.rm = TRUE), number\_cases = sum(number\_cases), percent\_cfr = round(number\_deaths / number\_cases \* 100, 1))

## `summarise()` has grouped output by 'case\_month'. You can override using the  
## `.groups` argument.

cdc\_data\_codes\_clean

## # A tibble: 233 × 5  
## # Groups: case\_month [39]  
## case\_month category number\_deaths number\_cases percent\_cfr  
## <chr> <chr> <int> <int> <dbl>  
## 1 2020-01 1. Large central metro 14 755 1.9  
## 2 2020-01 2. large fringe metro 23 570 4   
## 3 2020-01 3. Medium metro 24 378 6.3  
## 4 2020-01 4. Small metropolitan 12 113 10.6  
## 5 2020-01 5. Micropolitan 28 55 50.9  
## 6 2020-01 6. Noncore 0 11 0   
## 7 2020-02 1. Large central metro 87 1795 4.8  
## 8 2020-02 2. large fringe metro 0 678 0   
## 9 2020-02 3. Medium metro 0 92 0   
## 10 2020-02 4. Small metropolitan 0 13 0   
## # ℹ 223 more rows

cdc\_data\_codes\_clean %>%   
 ggplot(aes(x= case\_month, y = percent\_cfr/ 100, color = category, group = category)) +  
 geom\_line() +  
 facet\_wrap( ~ category, nrow = 3) +  
 scale\_y\_continuous(labels = scales::percent\_format(accuracy = 1)) +  
 labs(x = NULL, y = NULL, title = "Covid CFR % by county population") +  
 geom\_text(data = . %>% filter(percent\_cfr != 0), aes(label = percent\_cfr, hjust = -0.4), size = 2) +  
 theme(axis.text.x = element\_text(angle = 90))



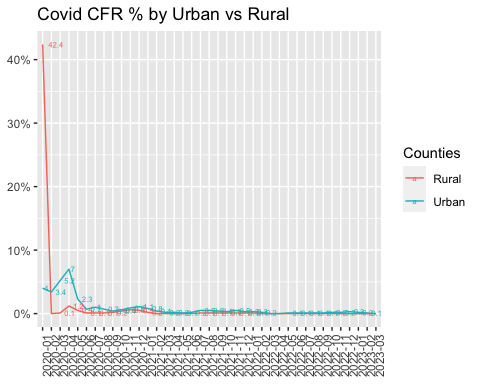
# COVID CFR % by rural and urban areas  
   
cdc\_data\_urban\_rural <- cdc\_data\_codes %>%   
 mutate(category = case\_when(x2013\_code %in% c(1, 2, 3, 4) ~ "Urban", x2013\_code %in% c(5, 6) ~ "Rural")) %>%  
 group\_by(case\_month, category) %>%  
 summarise(number\_deaths = sum(number\_cases[death\_yn == "Yes"], na.rm = TRUE), number\_cases = sum(number\_cases), percent\_cfr = round(number\_deaths / number\_cases \* 100, 1))

## `summarise()` has grouped output by 'case\_month'. You can override using the  
## `.groups` argument.

cdc\_data\_urban\_rural

## # A tibble: 78 × 5  
## # Groups: case\_month [39]  
## case\_month category number\_deaths number\_cases percent\_cfr  
## <chr> <chr> <int> <int> <dbl>  
## 1 2020-01 Rural 28 66 42.4  
## 2 2020-01 Urban 73 1816 4   
## 3 2020-02 Rural 0 12 0   
## 4 2020-02 Urban 87 2578 3.4  
## 5 2020-03 Rural 14 10446 0.1  
## 6 2020-03 Urban 18238 350191 5.2  
## 7 2020-04 Rural 518 42708 1.2  
## 8 2020-04 Urban 51312 735626 7   
## 9 2020-05 Rural 244 52865 0.5  
## 10 2020-05 Urban 14374 615953 2.3  
## # ℹ 68 more rows

cdc\_data\_urban\_rural %>%   
ggplot(aes(x= case\_month, y = percent\_cfr/ 100, color = category, group = category)) +  
 geom\_line() +  
 scale\_y\_continuous(labels = scales::percent\_format(accuracy = 1)) +  
 labs(x = NULL, y = NULL, title = "Covid CFR % by Urban vs Rural", color = "Counties") +  
 geom\_text(data = . %>% filter(percent\_cfr != 0), aes(label = percent\_cfr, hjust = -0.4), size = 2) +  
 theme(axis.text.x = element\_text(angle = 90))



# Money in US politics

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.”*](https://www.opensecrets.org/political-action-committees-pacs/foreign-connected-pacs)

We will scrape and work with data foreign connected PACs that donate to US political campaigns. The data for foreign connected PAC contributions in the 2022 election cycle can be found at <https://www.opensecrets.org/political-action-committees-pacs/foreign-connected-pacs/2022>. Then, we will use a similar approach to get data such contributions from previous years so that we can examine trends over time.

All data come from [OpenSecrets.org](https://www.opensecrets.org), a *“website tracking the influence of money on U.S. politics, and how that money affects policy and citizens’ lives”*.

library(robotstxt)  
paths\_allowed("https://www.opensecrets.org")

## [1] TRUE

base\_url <- "https://www.opensecrets.org/political-action-committees-pacs/foreign-connected-pacs/2022"  
  
contributions\_tables <- base\_url %>%  
 read\_html()

* First, make sure you can scrape the data for 2022. Use janitor::clean\_names() to rename variables scraped using snake\_case naming.

library(janitor)

##   
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':  
##   
## chisq.test, fisher.test

scrape\_contributions\_table <- function(html){  
   
 tables <-  
 html %>%   
 html\_nodes(css = "table") %>%   
 html\_table()  
   
 table <-  
 tables[[1]] %>%   
 janitor::clean\_names()  
   
 table  
}  
  
contributions <-  
 scrape\_contributions\_table(contributions\_tables)  
  
contributions

## # A tibble: 215 × 5  
## pac\_name\_affiliate country\_of\_origin\_pa…¹ total dems repubs  
## <chr> <chr> <chr> <chr> <chr>   
## 1 Accenture (Accenture) Ireland/Accenture plc $3,0… $0 $3,000  
## 2 Acreage Holdings Canada/Acreage Holdin… $0 $0 $0   
## 3 Air Liquide America France/L'Air Liquide … $17,… $14,… $2,500  
## 4 Airbus Group Netherlands/Airbus Gr… $193… $82,… $111,…  
## 5 Alexion Pharmaceuticals (AstraZene… UK/AstraZeneca PLC $186… $104… $82,2…  
## 6 Alkermes Inc Ireland/Alkermes Plc $84,… $34,… $50,0…  
## 7 Allianz of America (Allianz) Germany/Allianz AG Ho… $31,… $20,… $11,0…  
## 8 AMG Vanadium Netherlands/AMG Advan… $2,5… $0 $2,525  
## 9 Anheuser-Busch (Anheuser-Busch InB… Belgium/Anheuser-Busc… $457… $218… $239,…  
## 10 AON Corp (AON plc) UK/AON PLC $98,… $52,… $46,5…  
## # ℹ 205 more rows  
## # ℹ abbreviated name: ¹​country\_of\_origin\_parent\_company

* Clean the data:
  + Write a function that converts contribution amounts in total, dems, and repubs from character strings to numeric values.
  + Separate the country\_of\_origin\_parent\_company into two such that country and parent company appear in different columns for country-level analysis.

# write a function to parse\_currency  
parse\_currency <- function(x){  
 x %>%  
   
 # remove dollar signs  
 str\_remove("\\$") %>%  
   
 # remove all occurrences of commas  
 str\_remove\_all(",") %>%  
   
 # convert to numeric  
 as.numeric()  
}  
  
# clean country/parent co and contributions   
contributions <- contributions %>%  
 separate(country\_of\_origin\_parent\_company,   
 into = c("country", "parent"),   
 sep = "/",   
 extra = "merge") %>%  
 mutate(  
 total = parse\_currency(total),  
 dems = parse\_currency(dems),  
 repubs = parse\_currency(repubs)  
 )

* 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.
  + 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 contributions\_all.
* Write the data frame to a csv file called contributions-all.csv in the data folder.

scrape\_pac <- function(url){  
   
 contributions\_tables <- url %>%  
 read\_html()  
   
 contributions <- scrape\_contributions\_table(contributions\_tables)  
  
 contributions <- contributions %>%  
 separate(country\_of\_origin\_parent\_company, into = c("country", "parent"), sep = "/", extra = "merge") %>%  
 mutate(total = parse\_currency(total), dems = parse\_currency(dems), repubs = parse\_currency(repubs), year = str\_sub(url, -4, -1))  
   
 contributions  
}  
  
urls = c("https://www.opensecrets.org/political-action-committees-pacs/foreign-connected-pacs/2022","https://www.opensecrets.org/political-action-committees-pacs/foreign-connected-pacs/2021","https://www.opensecrets.org/political-action-committees-pacs/foreign-connected-pacs/2020","https://www.opensecrets.org/political-action-committees-pacs/foreign-connected-pacs/2019")  
  
contributions\_all <- map\_df(urls, scrape\_pac)  
  
write.csv(contributions\_all, here::here("data", "contributions\_all.csv"), row.names=FALSE)

# Scraping consulting jobs

The website [https://www.consultancy.uk/jobs/](https://www.consultancy.uk/jobs) lists job openings for consulting jobs.

library(robotstxt)  
paths\_allowed("https://www.consultancy.uk") #is it ok to scrape?  
  
base\_url <- "https://www.consultancy.uk/jobs/page/1"  
  
listings\_html <- base\_url %>%  
 read\_html()

Identify the CSS selectors in order to extract the relevant information from this page, namely

1. job
2. firm
3. functional area
4. type

Can you get all pages of ads, and not just the first one, https://www.consultancy.uk/jobs/page/1 into a dataframe?

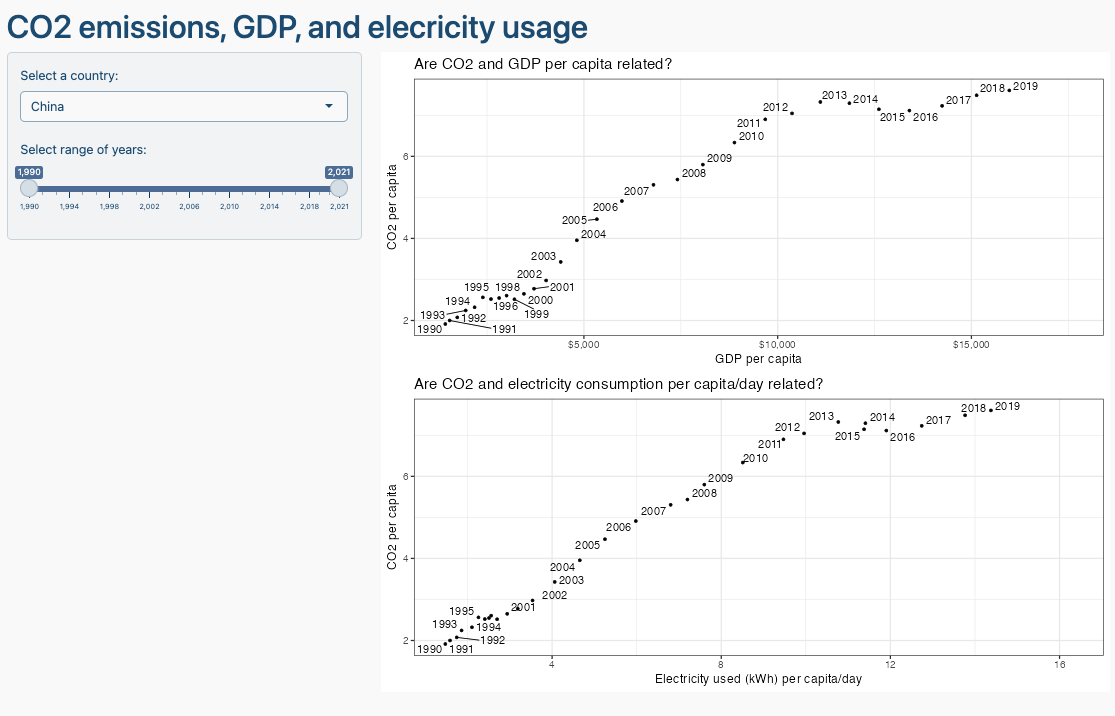
* Write a function called scrape\_jobs() that scrapes information from the webpage for consulting positions. This function should
  + have one input: the URL of the webpage and should return a data frame with four columns (variables): job, firm, functional area, and type
  + Test your function works with other pages too, e.g., <https://www.consultancy.uk/jobs/page/2>. Does the function seem to do what you expected it to do?
  + Given that you have to scrape ...jobs/page/1, ...jobs/page/2, etc., define your URL so you can join multiple stings into one string, using str\_c(). For instnace, if page is 5, what do you expect the following code to produce?

base\_url <- "https://www.consultancy.uk/jobs/page/1"  
url <- str\_c(base\_url, page)

* Construct a vector called pages that contains the numbers for each page available
* Map the scrape\_jobs() function over pages in a way that will result in a data frame called all\_consulting\_jobs.
* Write the data frame to a csv file called all\_consulting\_jobs.csv in the data folder.

# Create a shiny app - OPTIONAL

We have already worked with the data on electricity production and usage, GDP/capita and CO2/capita since 1990. You have to create a simple Shiny app, where a user chooses a country from a drop down list and a time interval between 1990 and 2020 and shiny outputs the following



You can use chatGPT to get the basic layout of Shiny app, but you need to adjust the code it gives you. Ask chatGPT to create the Shiny app using the gapminder data and make up similar requests for the inputs/outpus you are thinking of deploying.

# Deliverables

There is a lot of explanatory text, comments, etc. You do not need these, so delete them and produce a stand-alone document that you could share with someone. Knit the edited and completed R Markdown (Rmd) file as a Word or HTML document (use the “Knit” button at the top of the script editor window) and upload it to Canvas. You must be commiting and pushing your changes to your own Github repo as you go along.

# Details

* Who did you collaborate with: Sean, Angela, Deven, Nicho, Marc
* Approximately how much time did you spend on this problem set: 8 hours
* What, if anything, gave you the most trouble:

**Please seek out help when you need it,** and remember the [15-minute rule](https://dsb2023.netlify.app/syllabus/#the-15-minute-rule). You know enough R (and have enough examples of code from class and your readings) to be able to do this. If you get stuck, ask for help from others, post a question on Slack– and remember that I am here to help too!

As a true test to yourself, do you understand the code you submitted and are you able to explain it to someone else?

# Rubric

13/13: Problem set is 100% completed. Every question was attempted and answered, and most answers are correct. Code is well-documented (both self-documented and with additional comments as necessary). Used tidyverse, instead of base R. Graphs and tables are properly labelled. Analysis is clear and easy to follow, either because graphs are labeled clearly or you’ve written additional text to describe how you interpret the output. Multiple Github commits. Work is exceptional. I will not assign these often.

8/13: Problem set is 60–80% complete and most answers are correct. This is the expected level of performance. Solid effort. Hits all the elements. No clear mistakes. Easy to follow (both the code and the output). A few Github commits.

5/13: Problem set is less than 60% complete and/or most answers are incorrect. This indicates that you need to improve next time. I will hopefully not assign these often. Displays minimal effort. Doesn’t complete all components. Code is poorly written and not documented. Uses the same type of plot for each graph, or doesn’t use plots appropriate for the variables being analyzed. No Github commits.