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

VICTOR RIOS FARIA

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?

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)

#| label: problem-1  
  
payments <- dbGetQuery(sky\_westminster, "SELECT \* FROM payments") # creates the payments table  
  
members <- dbGetQuery(sky\_westminster, "SELECT \* FROM members") # creates the members table  
  
payments %>%  
 group\_by(member\_id) %>% # groups the data by the member\_id representing each MP  
 summarise(money\_received = sum(value)) %>% # calculates the total value that each MP received in the period  
 left\_join(members,by = c("member\_id" = "id")) %>% # joins the summarised payment data with the members' information data  
 arrange(desc(money\_received)) %>% # orders the MPs by descending order of total money received  
 slice(1) # gets the MP who received the most amount of money, namely Mrs Theresa May, who received a total of 2,809,765 in donations in the period

## # A tibble: 1 × 8  
## member\_id money\_received name gender constituency party\_id short\_name status  
## <chr> <dbl> <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 m8 2809765. There… F Maidenhead p4 Mrs May active

## 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?

#| label: problem-2  
  
entities\_5perc <- payments %>% # creates a new object to receive the list of entities that account for more than 5% of total payments  
 mutate(year = as.numeric(substr(date, nchar(date) - 3, nchar(date)))) %>% # creates a new variable containing the information about the year of the donation  
 filter(year %in% c(2020, 2021, 2022)) %>% # filters only the donations that happened between 2020 and 2022  
 group\_by(entity) %>% # groups the payment data by entity  
 summarise(total\_donations = sum(value)) %>% # calculates the total payments per entity  
 ungroup %>% # ungroup the data so the percentages can be calculated after  
 mutate(donations\_perc = total\_donations/sum(total\_donations)) %>% # calculates the percentage of total payments each entity accounts for  
 filter(donations\_perc>0.05) %>% # filters the data to retain only entities that accounted for more than 5% of the total payments  
 print() # prints the results so we can observe that Whiters LLP donated a total of 1,812,732, accounting for 5.25% of total payments and being the only entity that contributed with more than 5%

## # A tibble: 1 × 3  
## entity total\_donations donations\_perc  
## <chr> <dbl> <dbl>  
## 1 Withers LLP 1812732. 0.0534

payments %>%  
 filter(entity %in% entities\_5perc$entity) %>% # filters the payment data to keep only observations whose entity is among the ones that accounted for at least 5% of total payments  
 group\_by(member\_id) %>% # groups the data by MP member's id  
 summarise(total\_donations = sum(value)) %>% # calculates the total value received by each of the MPs that those entities donated for  
 left\_join(members,by = c("member\_id" = "id")) # joins the payment data with the members' information data to get the details on the MPs that received money from those entities. We can observe that Sir Geoffrey Cox is the only person Whiters LLP gave money to

## # A tibble: 1 × 8  
## member\_id total\_donations name gender constituency party\_id short\_name status  
## <chr> <dbl> <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 m1508 1812732. Sir … M Torridge an… p4 Sir Geoff… active

## 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?

#| label: problem-3  
  
payments %>%  
 summarize(qt\_entity=n\_distinct(entity)) # counts how many distinct entities paid money to MPS in the assigned period, getting a total of 2,213 distinct entities

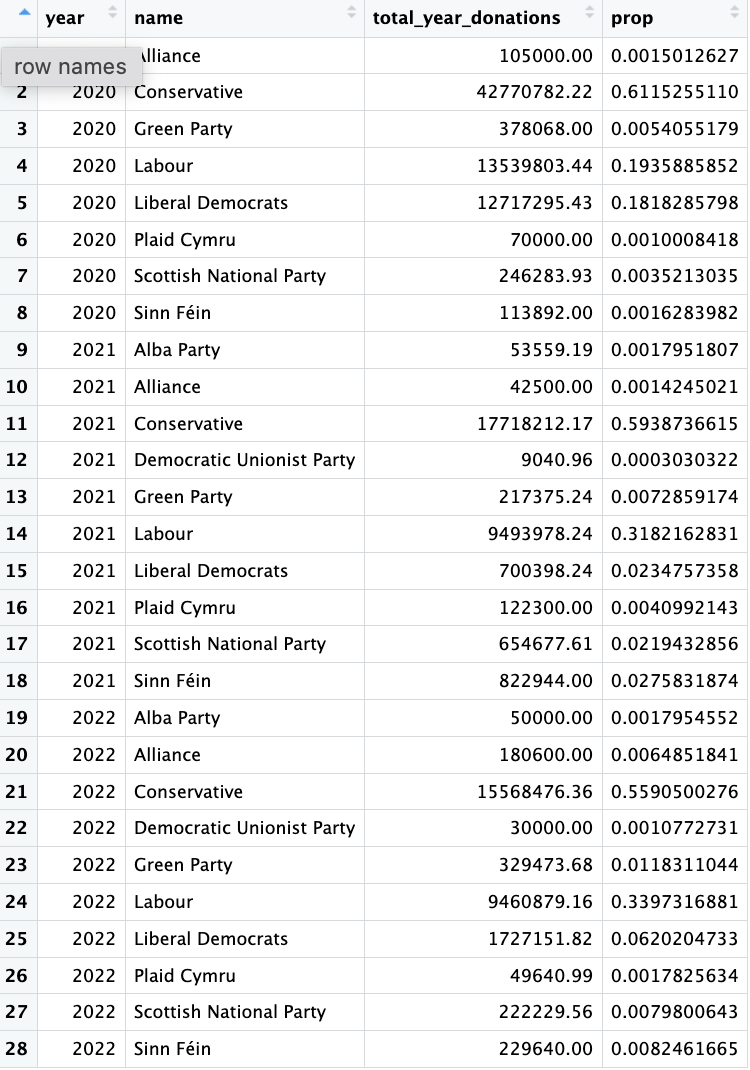
## qt\_entity  
## 1 2213

payments %>%  
 left\_join(members, by = c("member\_id" = "id")) %>% # joins the payment data with the members' information data  
 group\_by(entity) %>% # groups the data by entity and member who received the payment  
 summarize(qt\_parties = n\_distinct(party\_id)) %>% # counts how many different parties whose members received payments from each entity  
 mutate(single\_party = if\_else(qt\_parties > 1, 0, 1)) %>% # creates a new dummy variable that takes the value of 1 if the entity payed to member(s) of a single party and 0 otherwise  
 summarize(qt\_single\_party = sum(single\_party), perc\_single\_party = mean(single\_party)) # calculates the total of entities that payed to member(s) of a single party and the what percentage they represent in the dataset. In the case, we have a total of 2,036 of such entities, which represent 92% of the entities that performed payments in the period

## # A tibble: 1 × 2  
## qt\_single\_party perc\_single\_party  
## <dbl> <dbl>  
## 1 2036 0.920

## 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.



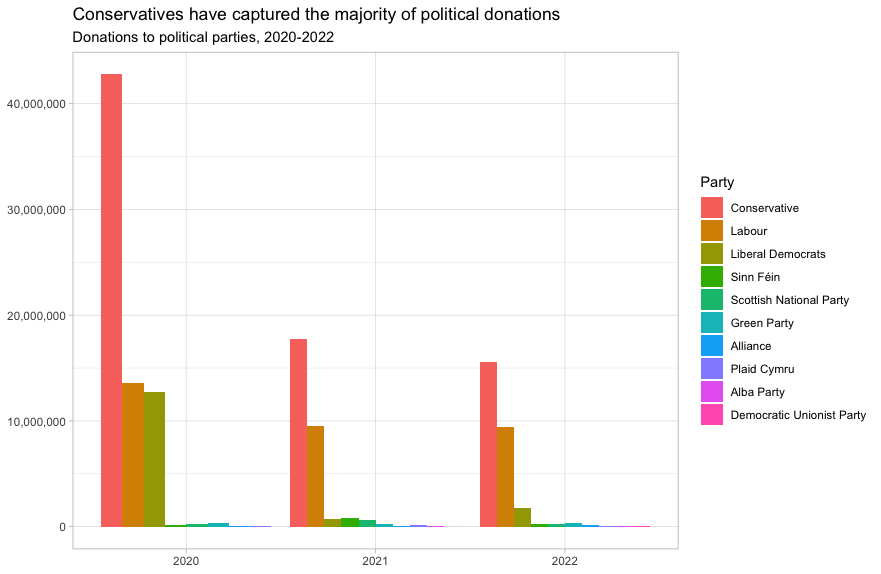
#| label: problem-4  
  
parties <- dbGetQuery(sky\_westminster, "SELECT \* FROM parties") # creates the parties table  
  
party\_year\_payments <- payments %>%  
 mutate(year = as.numeric(substr(date, nchar(date) - 3, nchar(date)))) %>% # creates a new variable containing the information about the year of the donation  
 filter(year %in% c(2020, 2021, 2022)) %>% # filters only the donations that happened between 2020 and 2022  
 select(year, member\_id, value) %>% # drops all irrelevant features for the problem  
 left\_join(members, by = c("member\_id" = "id")) %>% # joins the payment data with the members' information data  
 rename(member\_name = name) %>% # renames the variable that contains the member name information  
 left\_join(parties, by = c("party\_id" = "id")) %>% # joins the previous data with the parties' information data  
 group\_by(year, name) %>% # groups the data by year and party name  
 summarize(total\_year\_donations = sum(value)) %>% # calculates the total payment to each party each year  
 mutate (prop = total\_year\_donations / sum(total\_year\_donations)) # creates a new variable with the % of the total payments of the year each party received

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

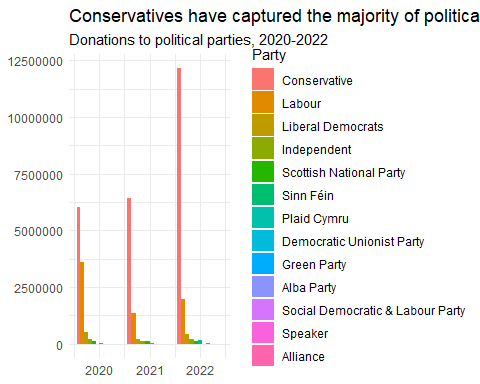
kable(party\_year\_payments) # prints the new table

| year | name | total\_year\_donations | prop |
| --- | --- | --- | --- |
| 2020 | Alba Party | 1320.00 | 0.0001248 |
| 2020 | Conservative | 6035343.83 | 0.5706716 |
| 2020 | Democratic Unionist Party | 5715.38 | 0.0005404 |
| 2020 | Green Party | 9500.00 | 0.0008983 |
| 2020 | Independent | 230103.46 | 0.0217574 |
| 2020 | Labour | 3615844.04 | 0.3418959 |
| 2020 | Liberal Democrats | 537694.00 | 0.0508416 |
| 2020 | Plaid Cymru | 23072.00 | 0.0021816 |
| 2020 | Scottish National Party | 108598.71 | 0.0102685 |
| 2020 | Sinn Féin | 1911.00 | 0.0001807 |
| 2020 | Social Democratic & Labour Party | 4900.00 | 0.0004633 |
| 2020 | Speaker | 1859.35 | 0.0001758 |
| 2021 | Alliance | 798.00 | 0.0000963 |
| 2021 | Conservative | 6422873.38 | 0.7747131 |
| 2021 | Democratic Unionist Party | 4970.20 | 0.0005995 |
| 2021 | Independent | 115603.04 | 0.0139438 |
| 2021 | Labour | 1373769.76 | 0.1657011 |
| 2021 | Liberal Democrats | 216295.64 | 0.0260891 |
| 2021 | Plaid Cymru | 26565.50 | 0.0032043 |
| 2021 | Scottish National Party | 128411.72 | 0.0154887 |
| 2021 | Social Democratic & Labour Party | 320.00 | 0.0000386 |
| 2021 | Speaker | 1040.00 | 0.0001254 |
| 2022 | Alba Party | 7579.35 | 0.0005021 |
| 2022 | Conservative | 12147904.41 | 0.8048007 |
| 2022 | Democratic Unionist Party | 15679.97 | 0.0010388 |
| 2022 | Green Party | 4346.47 | 0.0002880 |
| 2022 | Independent | 217817.05 | 0.0144304 |
| 2022 | Labour | 1980878.42 | 0.1312335 |
| 2022 | Liberal Democrats | 446046.06 | 0.0295506 |
| 2022 | Plaid Cymru | 1326.21 | 0.0000879 |
| 2022 | Scottish National Party | 106833.56 | 0.0070777 |
| 2022 | Sinn Féin | 161259.35 | 0.0106835 |
| 2022 | Social Democratic & Labour Party | 4630.00 | 0.0003067 |

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



#| label: problem-5  
  
ggplot(party\_year\_payments, aes(x=year, y=total\_year\_donations, fill = reorder(name, -total\_year\_donations))) + # creates a plot of the total donations per party each year and calls the reorder function to sort the data by the total year donations value in descending order  
geom\_bar(stat = "identity", position = "dodge") + # configures the display of the bars in the chart  
labs(fill = "Party", # changes the colour legend's title  
 title = "Conservatives have captured the majority of political donations",# sets the chart title...  
 subtitle = "Donations to political parties, 2020-2022") + # and its subtitle  
theme\_minimal() +  
xlab(NULL) + # removes the title from the x-axis  
ylab(NULL) # removes the title from the y-axis



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?

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.

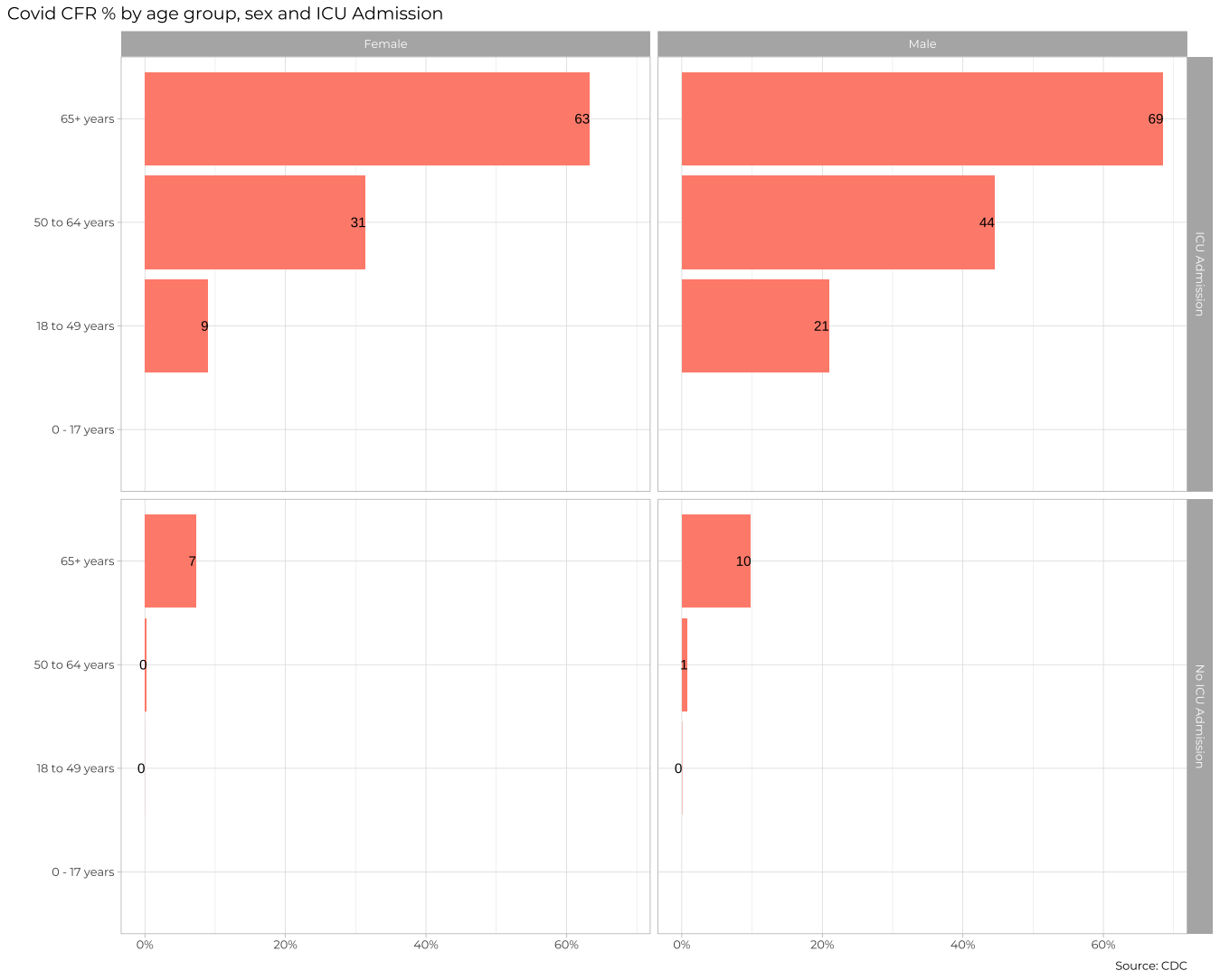
## Obtain the data

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

## 0.01 sec elapsed

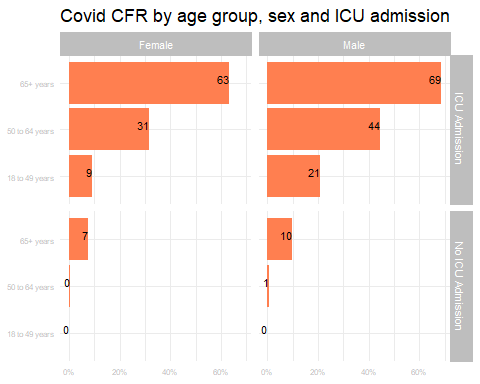
## FileSystemDataset with 1 Parquet file  
## 97,799,772 rows x 19 columns  
## $ case\_month <string> "2021-09", "2022-09", "2022-01", "2020…  
## $ res\_state <string> "TX", "TX", "TX", "CA", "IL", "CA", "N…  
## $ state\_fips\_code <int32> 48, 48, 48, 6, 17, 6, 36, 36, 36, 53, …  
## $ res\_county <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?

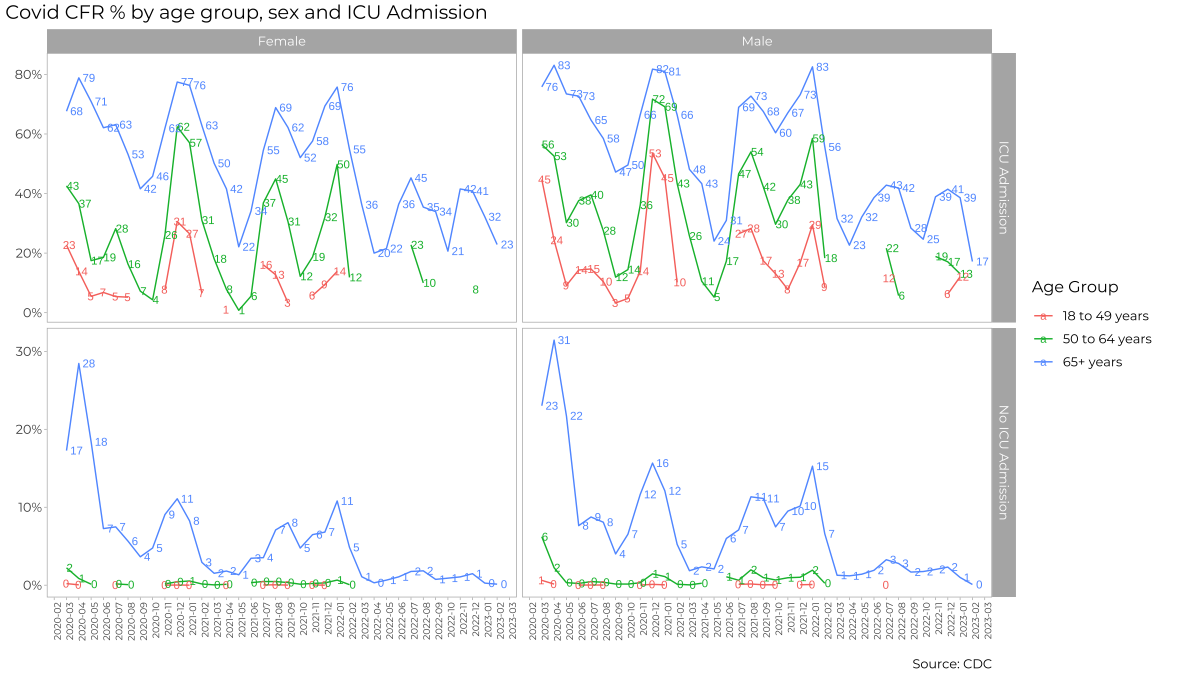


#| label: problem-6  
  
fatality\_rate = cdc\_data %>% # creates a new table to store the cdc data  
 group\_by(icu\_yn,age\_group,sex) %>% # groups the data by ICU admission status, age group and sex  
 count(death\_yn,sort=TRUE) %>% # counts how many cases there was both with and without deaths  
 collect() # collects the data from the query and saves it to the table  
  
fatality\_rate %>%   
 drop\_na(death\_yn, sex) %>% # removes all entries with no information about either the death or the sex  
 group\_by(icu\_yn,age\_group,sex) %>% # groups the data by ICU admission status, age group and sex  
 filter(!death\_yn %in% c("Unknown","Missing")) %>% # filters the data to remove all observations with unknown or missing values for the death status  
 filter(sex!="Unknown") %>% # filters the data to remove all observations with unknown gender  
 mutate(fatality\_rate = n/sum(n)\*100) %>% # calculates the fatality rate for each group  
 filter(!icu\_yn %in% c("Unknown","Missing")) %>% # filters the data to remove all observations with unknown or missing values for the ICU admission status  
 mutate(icu\_yn = if\_else(icu\_yn == "Yes", "ICU Admission", "No ICU Admission")) %>% # changes the values of the ICU admission status variable to match the labels wanted in the plot  
 filter(death\_yn=="Yes") %>% # filters only the groups relative to death cases, so that each group will be represented only once in the plot  
  
 ggplot(aes(x=fatality\_rate,y=age\_group)) + # creates a plot of fatality rate vs. age group  
 geom\_bar(position = "dodge",stat = "identity", fill = "coral") + # sets the plot type as a bar chart  
 facet\_grid(icu\_yn ~ sex) + # creates a grid with the graphics, with ICU status in the horizontal and gender in the vertical  
 labs(title = "Covid CFR by age group, sex and ICU admission") + # adds the chart's title  
 geom\_text(aes(label = round(fatality\_rate,0)), size = 3, vjust = 0, hjust = 1) + # adds the labels to the bars  
 scale\_x\_continuous(labels = function(x) paste0(x, "%")) + # adds the x-axis' labels  
 theme\_minimal() +  
 theme(axis.text = element\_text(size = 6, color = "gray"),  
 strip.text = element\_text(size = 8, color = "white"), # adds a title to each column and row  
 strip.background = element\_rect(fill = "gray", color = "gray", size = 1)) + # adds a background to the columns and rows' titles  
 xlab(NULL) + # removes the title from the x-axis  
 ylab(NULL) # removes the title from the y-axis

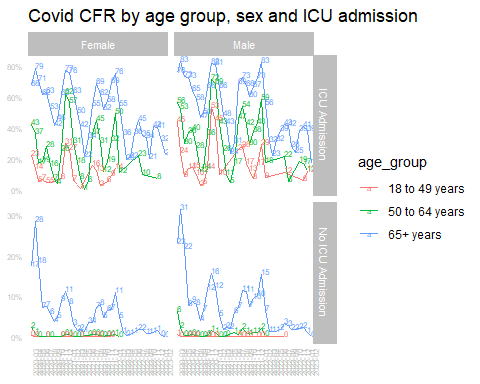
## Warning: The `size` argument of `element\_rect()` is deprecated as of ggplot2 3.4.0.  
## ℹ Please use the `linewidth` argument instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



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



#| label: problem-7  
  
fatality\_year = cdc\_data %>% # creates a new table to store the cdc data  
 group\_by(case\_month,icu\_yn,age\_group,sex) %>% # groups the data by date, admission status, age group and sex  
 count(death\_yn,sort=TRUE) %>% # counts how many cases there was both with and without deaths  
 collect() # collects the data from the query and saves it to the table  
  
#Filtering the data  
fatality\_year %>%  
 drop\_na(death\_yn, sex) %>% # removes all entries with no information about either the death or the sex  
 group\_by(case\_month,icu\_yn,age\_group,sex) %>% # groups the data by date, ICU admission status, age group and sex  
 filter(!death\_yn %in% c("Unknown","Missing")) %>% # filters the data to remove all observations with unknown or missing values for the death status  
 filter(sex!="Unknown") %>% # filters the data to remove all observations with unknown gender  
 mutate(fatality\_rate = n/sum(n)\*100) %>% # calculates the fatality rate for each group  
 filter(!icu\_yn %in% c("Unknown","Missing")) %>% # filters the data to remove all observations with unknown or missing values for the ICU admission status  
 mutate(icu\_yn = if\_else(icu\_yn == "Yes", "ICU Admission", "No ICU Admission")) %>% # changes the values of the ICU admission status variable to match the labels wanted in the plot  
 filter(death\_yn=="Yes") %>% # filters only the groups relative to death cases, so that each group will be represented only once in the plot  
 filter(!case\_month %in% c("2020-02","2020-01")) %>% # removes cases from the first 2 months of the dataset to match the example  
 arrange(case\_month) %>% # sorts the data by chronological order, so that the plot represents the evolution of the fatality rate  
   
 ggplot(aes(x=case\_month, y=fatality\_rate, color=age\_group, group=age\_group)) + # creates a plot of fatality rate vs. date for each age group  
 geom\_line() + # defines the plot type as a line chart  
 facet\_grid(icu\_yn ~ sex, scales = "free\_y") + # creates a grid with the graphics, with ICU status in the horizontal and gender in the vertical, with free scales for the y-axis (fatality rate)  
 labs(title = "Covid CFR by age group, sex and ICU admission") + # adds the chart's title  
 geom\_text(aes(label = round(fatality\_rate,0)), size = 2, vjust = 0, hjust = 0) + # adds the labels to the datapoints in the chart  
 scale\_y\_continuous(labels = function(x) paste0(x, "%")) + # adds the y-axis' labels and adds the % symbol to it  
 theme\_minimal() +  
 theme(axis.text = element\_text(size = 6, color = "gray"),  
 strip.text = element\_text(size = 8, color = "white"), # adds a title to each column and row  
 strip.background = element\_rect(fill = "gray", color = "gray", size = 1), # adds a background to the columns and rows' titles  
 axis.text.x = element\_text(angle = 90, hjust = 1), # turns the x-axis labels 90 degrees for better visualization  
 panel.grid = element\_blank())+ # removes the grid lines from the plots  
 xlab(NULL) + # removes the title from the x-axis  
 ylab(NULL) # removes the title from the y-axis



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)

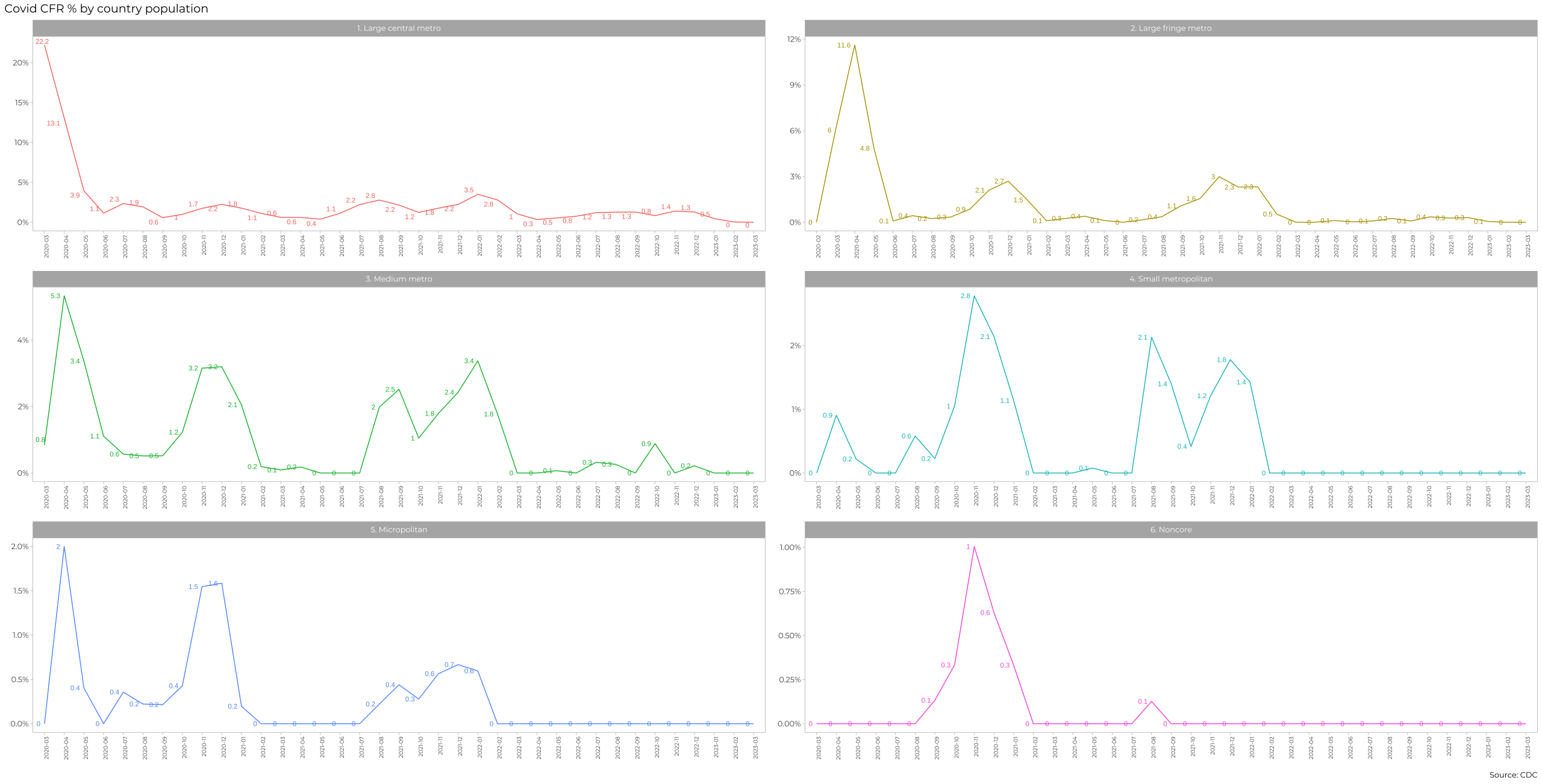
urban\_rural <- read\_xlsx(here::here("data", "NCHSURCodes2013.xlsx")) %>%   
 janitor::clean\_names()

Each county belongs in six 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

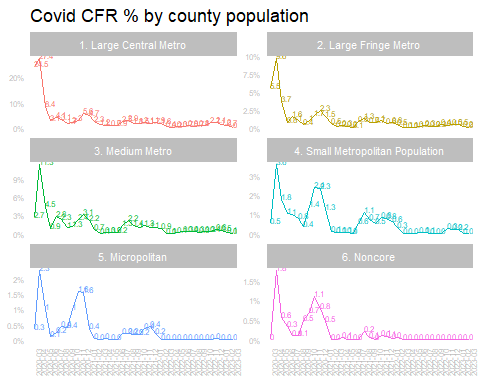
1. Large central metro - 1 million or more population and contains the entire population of the largest principal city
2. large fringe metro - 1 million or more poulation, but does not qualify as 1
3. Medium metro - 250K - 1 million population
4. Small metropolitan population < 250K
5. Micropolitan
6.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

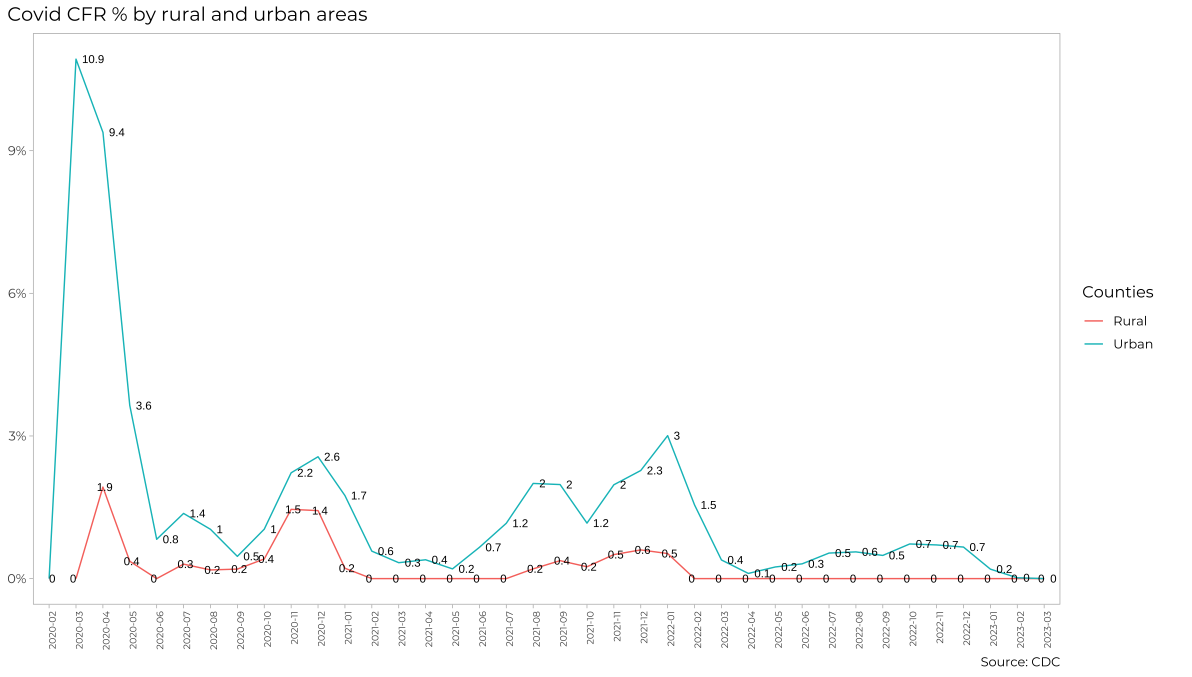
Can you query the database, extract the relevant information, and reproduce the following two graphs that look at the Case Fatality ratio (CFR) in different counties, according to their population?



#| label: problem-8  
  
#Query  
fatality\_county = cdc\_data %>% # creates a new table to store the cdc data  
 group\_by(case\_month,county\_fips\_code) %>% # groups the data by date and county code  
 count(death\_yn,sort=TRUE) %>% # counts how many cases there was both with and without deaths  
 collect() # collects the data from the query and saves it to the table  
  
#Joining the tables  
fatality\_urban\_rural = left\_join(fatality\_county,urban\_rural,by=c("county\_fips\_code"="fips\_code")) # joins the previous table with detailed information about the FIPS code and county demographics  
  
#Filtering the data  
fatality\_urban\_rural %>%  
 drop\_na(county\_fips\_code, death\_yn) %>% # removes all entries with no information about either the death or the sex  
 rename("n\_deaths"="n") %>% # renames the variable containing the count of number of deaths per grouping  
 ungroup() %>% # ungroup the data  
 select(case\_month,death\_yn,n\_deaths,x2013\_code) %>% # selects only the relevant features to keep  
 filter(!death\_yn %in% c("Unknown","Missing")) %>% # filters the data to remove all observations with unknown or missing values for the death status   
 filter(!case\_month %in% c("2020-02","2020-01")) %>% # removes cases from the first 2 months of the dataset to match the example  
 group\_by(case\_month,death\_yn,x2013\_code) %>% # groups the data by the date, death status and county category  
 summarize(cat\_deaths = sum(n\_deaths)) %>% # summarizes the data to get the total deaths per category  
 pivot\_wider(names\_from = death\_yn, values\_from = cat\_deaths) %>% # pivots the data into a wider format, creating a column for letal cases and another for non letal cases  
 mutate(Yes = ifelse(is.na(Yes), 0, Yes), No = ifelse(is.na(No), 0, No)) %>% # replaces null values with 0 in both columns just created  
 rename("death\_yes" = "Yes", "death\_no" = "No") %>% # rename those columns so that their meaning is more explicit  
 mutate(fatality\_rate = death\_yes/(death\_yes+death\_no)\*100) %>% # calculates the fatality rate for each group   
 mutate(x2013\_code = case\_when( # replaces the values in the x2013\_code variable so that the code is overwritten with its matching description  
 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 Population",  
 x2013\_code == 5 ~ "5. Micropolitan",  
 TRUE ~ "6. Noncore")  
 )%>%  
 arrange(case\_month) %>% # sorts the data by chronological order, so that the plot represents the evolution of the fatality rate   
   
 ggplot(aes(x=case\_month, y=fatality\_rate, color=x2013\_code, group=x2013\_code)) + # creates a plot of fatality rate vs. date for each age county category  
 geom\_line() + # defines the plot type as a line chart  
 facet\_wrap(~x2013\_code, scales = "free\_y", ncol = 2, nrow = 3, ) + # creates a facet wrap based on the county category with a specific grid size of 2 columns and 3 rows  
 labs(title = "Covid CFR % by county population") + # adds the chart's title   
 geom\_text(aes(label = round(fatality\_rate,1)), size = 2, vjust = 0, hjust = 0) + # adds the labels to the datapoints in the chart  
 scale\_y\_continuous(labels = function(x) paste0(x, "%")) + # adds the y-axis' labels and adds the % symbol to it  
 theme\_minimal() +  
 theme(axis.text = element\_text(size = 6, color = "gray"),  
 strip.text = element\_text(size = 8, color = "white"), # adds a title to each column and row  
 strip.background = element\_rect(fill = "gray", color = "gray", size = 5), # adds a background to the columns and rows' titles  
 axis.text.x = element\_text(angle = 90, hjust = 1), # turns the x-axis labels 90 degrees for better visualization  
 panel.grid = element\_blank(), # removes the grid lines from the plots  
 legend.position = "none" # removes the legend from the plot  
 )+  
 xlab(NULL) + # removes the title from the x-axis  
 ylab(NULL) # removes the title from the y-axis

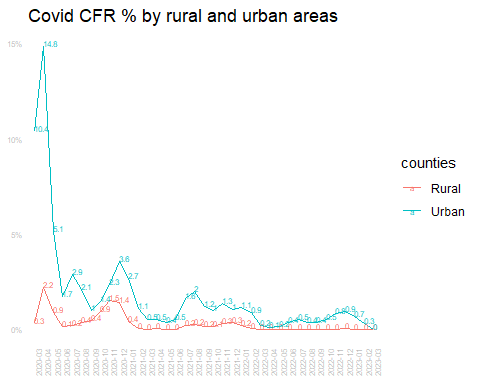
## `summarise()` has grouped output by 'case\_month', 'death\_yn'. You can override  
## using the `.groups` argument.





#| label: problem-9  
  
fatality\_urban\_rural %>%  
 drop\_na(county\_fips\_code, death\_yn) %>% # removes all entries with no information about either the death or the sex  
 rename("n\_deaths"="n") %>%# renames the variable containing the count of number of deaths per grouping  
 ungroup() %>% # ungroup the data  
 select(case\_month,death\_yn,n\_deaths,x2013\_code) %>% # selects only the relevant features to keep  
 filter(!death\_yn %in% c("Unknown","Missing")) %>% # filters the data to remove all observations with unknown or missing values for the death status   
 filter(!case\_month %in% c("2020-02","2020-01")) %>% # removes cases from the first 2 months of the dataset to match the example  
 mutate(counties = if\_else(x2013\_code == 5 | x2013\_code == 6, "Rural", "Urban")) %>% # creates a new variable to classify the counties into Rural or Urban areas based on their category code  
 group\_by(case\_month,death\_yn,counties) %>% # groups the data by date, death status and county area type  
 summarize(cat\_deaths = sum(n\_deaths)) %>% # summarizes   
 # summarizes the data to get the total deaths per category  
 pivot\_wider(names\_from = death\_yn, values\_from = cat\_deaths) %>% # pivots the data into a wider format, creating a column for letal cases and another for non letal cases  
 mutate(Yes = ifelse(is.na(Yes), 0, Yes), No = ifelse(is.na(No), 0, No)) %>% # replaces null values with 0 in both columns just created  
 rename("death\_yes" = "Yes", "death\_no" = "No") %>% # rename those columns so that their meaning is more explicit  
 mutate(fatality\_rate = death\_yes/(death\_yes+death\_no)\*100) %>% # calculates the fatality rate for each group   
 arrange(case\_month) %>% # sorts the data by chronological order, so that the plot represents the evolution of the fatality rate   
   
 ggplot(aes(x=case\_month, y=fatality\_rate, color=counties, group=counties)) + # creates a plot of fatality rate vs. date for each type of county  
 geom\_line() + # defines the plot type as a line chart  
 labs(title = "Covid CFR % by rural and urban areas") + # adds the chart's title  
 geom\_text(aes(label = round(fatality\_rate,1)), size = 2, vjust = 0, hjust = 0) + # adds the labels to the datapoints in the chart  
 scale\_y\_continuous(labels = function(x) paste0(x, "%")) +  
 theme\_minimal() +  
 theme(axis.text = element\_text(size = 6, color = "gray"),  
 strip.text = element\_text(size = 8, color = "white"), # adds a title to each column and row  
 strip.background = element\_rect(fill = "gray", color = "gray", size = 5), # adds a background to the columns and rows' titles  
 axis.text.x = element\_text(angle = 90, hjust = 1), # turns the x-axis labels 90 degrees for better visualization  
 panel.grid = element\_blank() # removes the grid lines from the plots  
 )+  
 xlab(NULL) + # removes the title from the x-axis  
 ylab(NULL) # removes the title from the y-axis

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



# 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.

#| label: problem-10  
  
opensecrets <- contributions\_tables %>% # creates a new table to store the html information  
 html\_node("table") %>% # gets the information from the html table...  
 html\_table(fill = TRUE) %>% # and fills our new table with its values  
 clean\_names() # clean the variable names using snake\_case style  
  
# View the cleaned table  
opensecrets

## # 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   
opensecrets <- opensecrets %>%  
 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”.

#| label: problem-11  
  
scrape\_pac <- function(URL){ #creates a function called scrape\_pac with a single variable URL  
 URL %>%  
 read\_html() %>% # reads the information in the html from the given URL  
 html\_node("table") %>% # gets the information from the html table...  
 html\_table(fill = TRUE) %>% # and fills our new table with its values  
 clean\_names() %>% # clean the variable names using snake\_case style  
   
 separate(country\_of\_origin\_parent\_company, # separates the information about parent company and country of origin...  
 into = c("country", "parent"), # ... by defining the names of the new columns...  
 sep = "/", # ... and the separator  
 extra = "merge") %>%  
   
 mutate(  
 total = parse\_currency(total), # calls the parse\_currency function to remove the currency character from the total amount  
 dems = parse\_currency(dems), # calls the parse\_currency function to remove the currency character from the dems amount  
 repubs = parse\_currency(repubs), # calls the parse\_currency function to remove the currency character from the repubs amount  
 year = as.numeric(substr(URL, nchar(URL) - 3, nchar(URL))) # creates a new variable containing the information about the year  
 )  
}

* 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?

#| label: problem-12  
  
URL2022 <- "https://www.opensecrets.org/political-action-committees-pacs/foreign-connected-pacs/2022" # defines the URL for 2022  
  
URL2020 <- "https://www.opensecrets.org/political-action-committees-pacs/foreign-connected-pacs/2020" # defines the URL for 2020  
  
URL2000 <- "https://www.opensecrets.org/political-action-committees-pacs/foreign-connected-pacs/2000" # defines the URL for 2000  
  
opensecrets2022 <- scrape\_pac(URL2022) # calls the function to obtain the data for the year of 2022  
  
opensecrets2020 <- scrape\_pac(URL2020) # calls the function to obtain the data for the year of 2020  
  
opensecrets2000 <- scrape\_pac(URL2000) # calls the function to obtain the data for the year of 2000

* Construct a vector called urls that contains the URLs for each webpage that contains information on foreign-connected PAC contributions for a given year.

#| label: problem-13  
  
root\_url <- "https://www.opensecrets.org/political-action-committees-pacs/foreign-connected-pacs/" # defines a root URL not containing the year information  
  
urls <- c(paste0(root\_url, 2022), paste0(root\_url, 2020), paste0(root\_url, 2018), paste0(root\_url, 2016), paste0(root\_url, 2014), paste0(root\_url, 2012), paste0(root\_url, 2010), paste0(root\_url, 2008), paste0(root\_url, 2006), paste0(root\_url, 2004), paste0(root\_url, 2002), paste0(root\_url, 2000)) # adds the year information to the URL for all possible years

* Map the scrape\_pac() function over urls in a way that will result in a data frame called contributions\_all.

#| label: problem-14  
  
contributions\_all <- list() # defines contributions\_all as a list of tables  
  
for (URL in urls) {  
 contributions\_all[[URL]] <- scrape\_pac(URL) # calls the scrape\_pac() function to get the data for all years from the URLs built before  
}

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

#| label: problem-15  
  
write\_csv(bind\_rows(contributions\_all), file = here::here("data","contributors\_all.csv"))

# 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?

#| label: problem-16  
  
base\_url <- "https://www.consultancy.uk/jobs/page/1"  
  
listings\_html <- base\_url %>%  
 read\_html()  
  
  
  
  
jobs = listings\_html %>% # creates a new table to assign the values scraped from the website  
 html\_nodes("#dataTable") %>% # gets the nodes from the datatable object in the html  
 html\_table() # scrapes the data from the html table  
  
#Scrape the number of pages and collects the highest number  
pages = listings\_html %>% # creates a new object to assign the number of pages in the website  
 html\_nodes("#paging") %>% # gets the number of pages from the website  
 html\_text() # scrapes the data from the html text  
  
pages = strsplit(pages, "\\n")[[1]] # formats the text obtained from the scraping to isolate the desired information  
pages = pages[[length(pages)-1]]   
pages = as.numeric(str\_replace(pages," jobs found","")) # gets the total number of jobs found in all pages  
pages = ceiling(pages/45) # divides the number of jobs by the number of jobs per page and round it up to get the total number of pages with job offerings

* 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)

#| label: problem-17  
  
scrape\_jobs <- function(base\_url){ # creates a new function to scrape all the pages  
 base\_url <- "https://www.consultancy.uk/jobs/page/1" # defines the base URL for scraping  
   
 listings\_html <- base\_url %>%   
 read\_html() # reads the html from the given URL  
   
# scrapes the number of pages  
 paging <- listings\_html %>%   
 html\_nodes("#paging") %>%  
 html\_text()  
   
 paging <- strsplit(paging, "\\n")[[1]] # formats the text obtained from the scraping to isolate the desired information  
 paging <- paging[[length(paging)-1]]  
 paging <- as.numeric(str\_replace(paging," jobs found","")) # gets the total number of jobs found in all pages  
 paging <- ceiling(paging/45) # divides the number of jobs by the number of jobs per page and round it up to get the total number of pages with job offerings  
   
 pages <- 1:paging # creates a vector of page numbers  
   
 job\_offers <- list()  
 for (n in pages){ # Creates a loop for all the pages  
 url <- paste("https://www.consultancy.uk/jobs/page/",n,sep="") # adds the page number to the base URL  
 listings\_html <- url %>%  
 read\_html() # reads the HTML  
   
#Scrapes the jobs  
 job\_offers[[n]] <- listings\_html %>%  
 html\_nodes("#dataTable") %>%  
 html\_table()  
 }  
 job\_offers <- bind\_rows(job\_offers)  
 return(job\_offers) # returns the listing of jobs offers as the output of the function  
}

* Construct a vector called pages that contains the numbers for each page available

#| label: problem-18  
  
#already answered within the function (line 674)

* Map the scrape\_jobs() function over pages in a way that will result in a data frame called all\_consulting\_jobs.

#| label: problem-19  
  
base\_url <- "https://www.consultancy.uk/jobs/page/1"  
  
all\_consulting\_jobs <- map\_dfr(pages, ~scrape\_jobs(paste0(base\_url, .x))) # performs the mapping in a way that will result in the desired data frama  
  
print(all\_consulting\_jobs) # prints the resulting data frame

## # A tibble: 332 × 4  
## Job Firm `Functional area` Type   
## <chr> <chr> <chr> <chr>  
## 1 "Senior Infrastructure & Cloud Services Adviso… West… "Cloud\n+1\nIT A… Job   
## 2 "Senior 3D/Motion Designer\nYonder Consulting" Yond… "Marketing\n+1\n… Job   
## 3 "Manager - Technology\nFirst Consulting" Firs… "Mobile & Apps\n… Job   
## 4 "HR Manager\nBearingPoint" Bear… "Human Resources" Job   
## 5 "Analyst, satellite and space markets\nAnalysy… Anal… "Strategy\n+4\nD… Job   
## 6 "PH-4804; Test Automation Manager, Python / Az… B2E … "Unknown" Job   
## 7 "Director Client Services - Life Sciences\nGen… Geni… "Strategy\n+2\nM… Job   
## 8 "Internships\nSimon-Kucher" Simo… "Pricing" Inte…  
## 9 "Senior Business Development Manager\nAyming" Aymi… "Sales" Job   
## 10 "PMO Lead\nThreeTwoFour" Thre… "Project Managem… Job   
## # ℹ 322 more rows

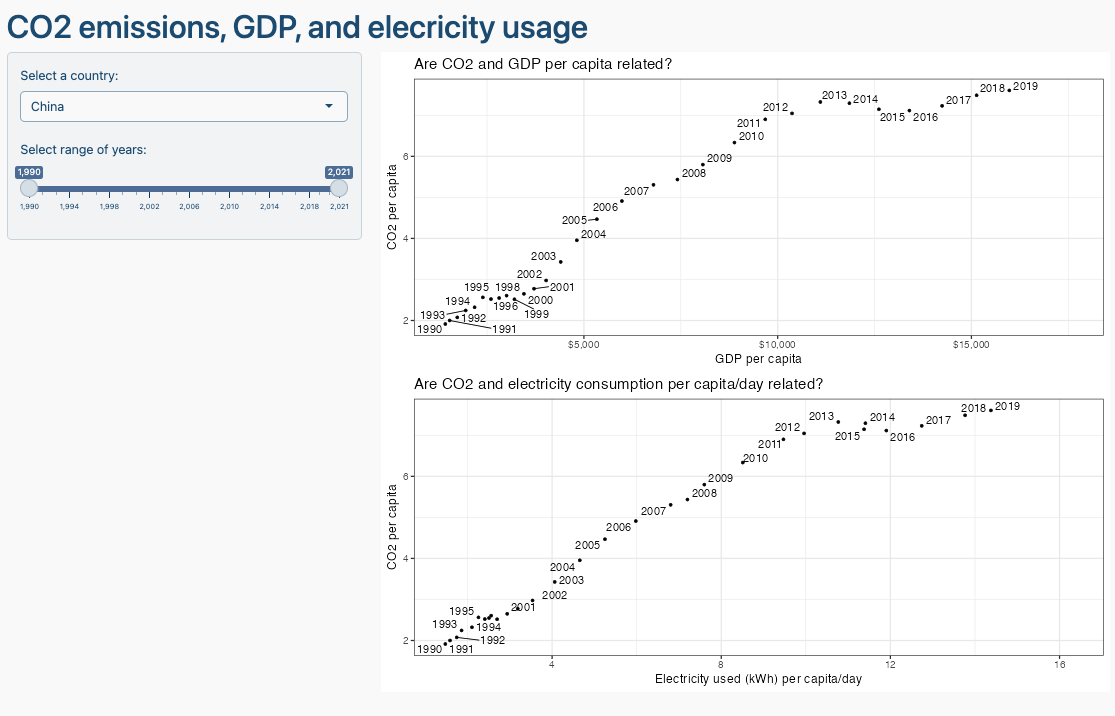
* Write the data frame to a csv file called all\_consulting\_jobs.csv in the data folder.

#| label: problem-20  
  
write.csv(bind\_rows(all\_consulting\_jobs), file = here::here("data","all\_consulting\_jobs.csv")) # creates the desired .csv file  
  
cat("The all\_consulting\_jobs.csv file has been created in the data folder.\n") # prints a message to confirm the file creation

## The all\_consulting\_jobs.csv file has been created 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: GUSTAVO MENDONCA
* Approximately how much time did you spend on this problem set: 10 hours
* What, if anything, gave you the most trouble: Scraping the consulting jobs website

**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.