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

Ignacio Landerreche

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?

payments<-dplyr::tbl(sky\_westminster,"payments")  
#glimpse(payments)

members\_appgs<-dplyr::tbl(sky\_westminster,"member\_appgs")  
#glimpse(members\_appgs)

members<-dplyr::tbl(sky\_westminster,"members")  
#glimpse(members)

parties<-dplyr::tbl(sky\_westminster,"parties")  
#glimpse(parties)

party\_donations<-dplyr::tbl(sky\_westminster,"party\_donations")  
#glimpse(party\_donations)

## 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?

payments %>%   
 group\_by(entity,date) %>%   
 summarise(total\_value=sum(value)) %>%   
 collect() %>% #Collecting to then being able to mutate the strings  
 mutate(date2=strtoi(str\_sub(date,-4))) %>% #taking the last 4 digits to have the year and converting it to number format  
 filter(date2 %in% c(2020,2021,2022)) %>%   
 ungroup() %>% #ungroupping to then be able to group by entity  
 group\_by(entity) %>%   
 summarise(total\_value=sum(total\_value)) %>%   
 mutate(total\_percentage=total\_value/sum(total\_value)\*100) %>% #Creating the percentage values  
 arrange(desc(total\_percentage))

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

## 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.

## # A tibble: 2,146 × 3  
## entity total\_value total\_percentage  
## <chr> <dbl> <dbl>  
## 1 Withers LLP 1812732. 5.34   
## 2 Fiona Bruce and Co LLP 711749. 2.10   
## 3 Charles Stanley 674821. 1.99   
## 4 Unite 633313. 1.86   
## 5 Cambridge Speaker Series 408200 1.20   
## 6 GMB Union 388738. 1.14   
## 7 MPM Connect 345217 1.02   
## 8 Centerview Partners LLP 277724. 0.818  
## 9 Council of Insurance Agents & Brokers 276130 0.813  
## 10 Hindustan Times 261652. 0.770  
## # ℹ 2,136 more rows

#Creating a dataset in my laptop for members only with desirable variables  
members\_sublist<-members %>%   
 select(id,name,party\_id) %>%   
 collect()  
  
#Creating the final database only with "Withers LLP" and left joining it with the previously made data base  
payments %>%   
 filter(entity=="Withers LLP") %>%   
 collect() %>%   
 left\_join(members\_sublist,by=c("member\_id"="id")) %>%   
 group\_by(name)%>%   
 summarise(total\_value=sum(value))

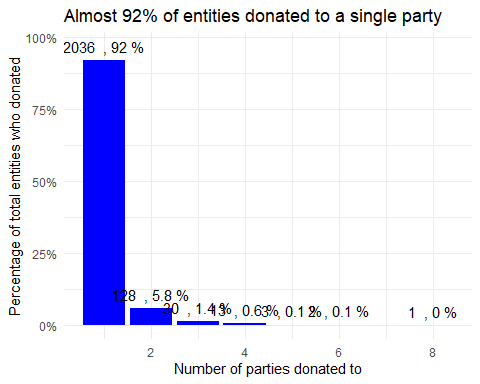
## # A tibble: 1 × 2  
## name total\_value  
## <chr> <dbl>  
## 1 Sir Geoffrey Cox 1812732.

**ANSWER**: Entity called Withers LLP accounts for around 5.3% of total payments, and they give the money to Sir Geoffrey Cox

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

* How many distinct entities who paid money to MPS are there? *ANSWER*: 2,213 entities
* How many (as a number and %) donated to MPs belonging to a single party only? *ANSWER* : 2036, which represent 92% approximately

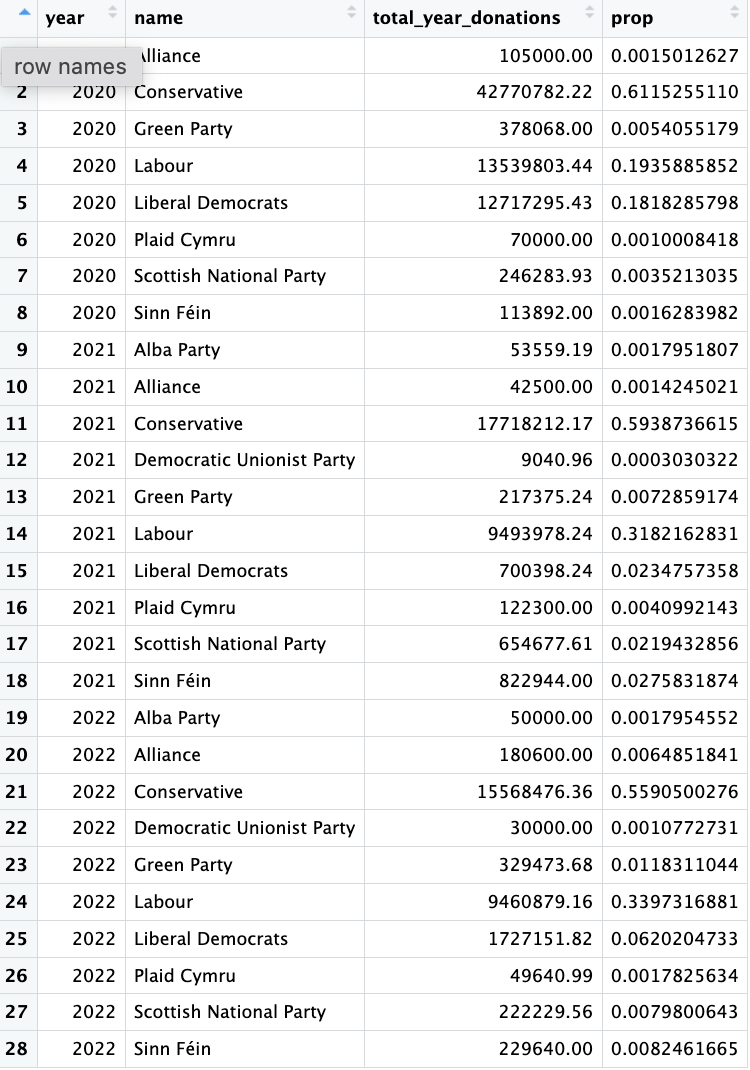
#Counting how many distinct entities are there  
total\_entities<-payments %>%   
 summarise(n\_distinct(entity)) %>%   
 collect()  
  
  
  
df\_1<-payments %>%   
 collect() %>%   
 left\_join(members\_sublist,by=c("member\_id"="id")) %>% #Joining the payments table with the members list previously made  
 group\_by(entity) %>%   
 count(party\_id)%>%   
 select(-n) %>% #deleting the counting column as we want to then group and count by entity  
 count(entity) %>%   
 arrange(desc(n))  
   
#Creating a graph  
  
df\_1 %>%   
 group\_by(n) %>%   
 summarise(total=n()) %>%   
 mutate(freq=total/2213)%>% #We know total entities is 2213 from question part 1.   
 ggplot(aes(x=n,y=freq))+geom\_col(fill="blue")+ #Creating the column chart  
 theme\_minimal()+   
 scale\_y\_continuous(labels = scales::percent)+ #Modifying the scale of y to percentage  
 labs(  
 title="Almost 92% of entities donated to a single party",  
 x="Number of parties donated to",  
 y="Percentage of total entities who donated"  
   
 )+ #Adding labs  
 geom\_text(aes(label = paste(total," ,",round(freq\*100,1),"%"),y=freq+0.05),colour = "black") #finalizing by adding the legend



## 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)



#Creating a subset so I can left\_join it  
subset\_parties<-parties %>%   
 select(id,name) %>%   
 collect()  
  
#Creating dataset for table  
data\_for\_table<-party\_donations %>%   
 collect() %>%   
 mutate(date2=ymd(date)) %>% #Using lubridate to create a year column  
 mutate(year=year(date2))%>%   
 left\_join(subset\_parties,by=c("party\_id"="id")) %>% #Left joining for the parties  
 group\_by(year,name) %>%   
 summarise(total\_year\_donations=sum(value)) %>%   
 mutate(prop=total\_year\_donations/sum(total\_year\_donations)) %>%   
 mutate(name=fct\_rev(fct\_reorder(name,total\_year\_donations))) #Reordering to then print the table

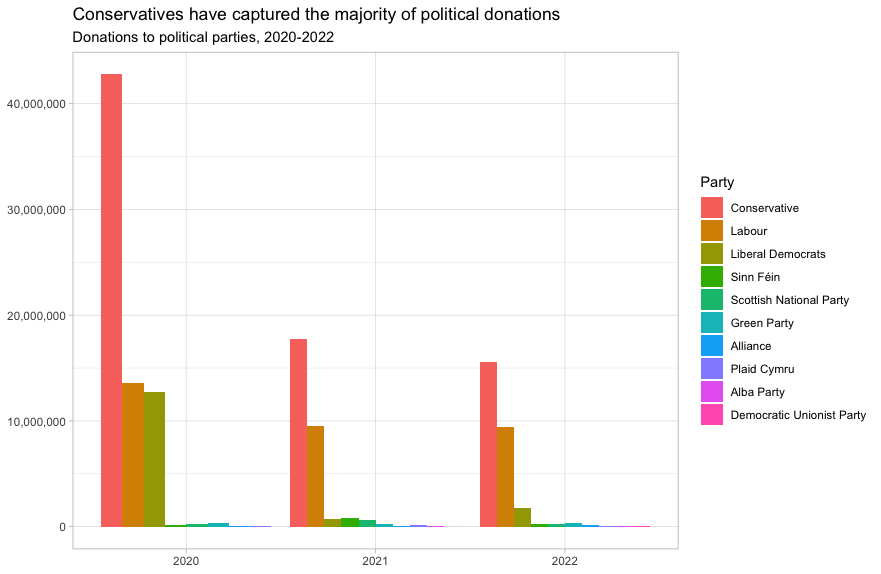
## `summarise()` has grouped output by 'year'. You can override using the  
## `.groups` argument.

data\_for\_table

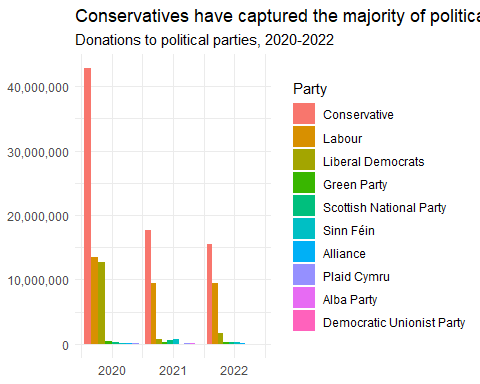
## # A tibble: 28 × 4  
## # Groups: year [3]  
## year name total\_year\_donations prop  
## <dbl> <fct> <dbl> <dbl>  
## 1 2020 Alliance 105000 0.00150  
## 2 2020 Conservative 42770782. 0.612   
## 3 2020 Green Party 378068 0.00541  
## 4 2020 Labour 13539803. 0.194   
## 5 2020 Liberal Democrats 12717295. 0.182   
## 6 2020 Plaid Cymru 70000 0.00100  
## 7 2020 Scottish National Party 246284. 0.00352  
## 8 2020 Sinn Féin 113892 0.00163  
## 9 2021 Alba Party 53559. 0.00180  
## 10 2021 Alliance 42500 0.00142  
## # ℹ 18 more rows

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

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



ggplot(data\_for\_table,aes(x=year,y=total\_year\_donations,fill=name))+  
 geom\_col(position="dodge")+ #Using dodge position so it is not stacked  
 theme\_minimal()+  
 labs(  
 title="Conservatives have captured the majority of political donations",  
 subtitle="Donations to political parties, 2020-2022",  
 x=NULL,  
 y=NULL,  
 fill="Party"  
)+  
 scale\_y\_continuous(labels=comma) #Formatting the y axis



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

dbDisconnect(sky\_westminster)

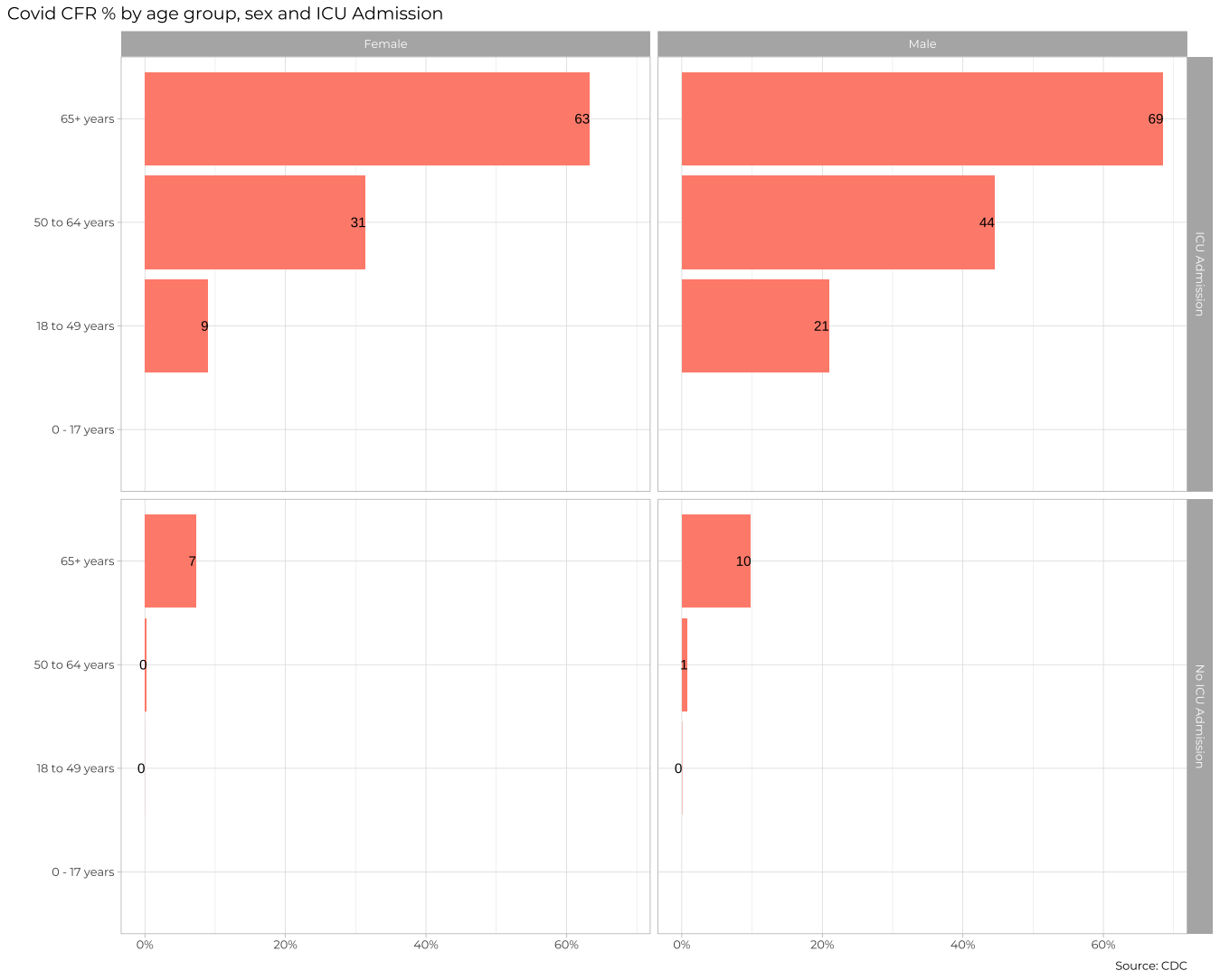
# Anonymised Covid patient data from the CDC

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.

## 0.06 sec elapsed

Can you query the database and replicate the following plot?

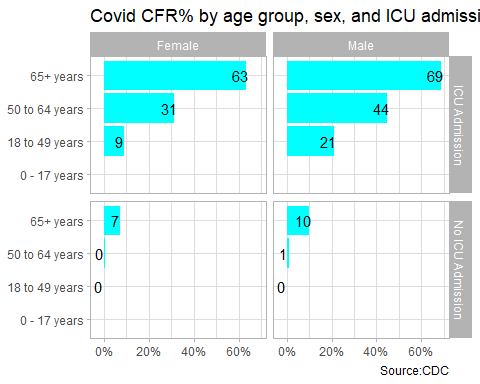
knitr::include\_graphics(here::here("images", "covid-CFR-ICU.png"), error = FALSE)



graph\_1<-cdc\_data %>%   
 filter(icu\_yn %in% c("Yes","No")) %>% #Cleansing data on different variables, to drop NA and Missings  
 filter(!(age\_group %in% c("Missing","NA"))) %>%  
 filter(sex %in% c("Male","Female")) %>%   
 filter(death\_yn %in% c("Yes","No")) %>%   
 group\_by(death\_yn,icu\_yn,sex,age\_group) %>% #Groupping by desirable variables   
 count(death\_yn) %>%   
 collect()  
  
graph\_1\_pivot<-graph\_1 %>%   
 pivot\_wider(names\_from="death\_yn",values\_from="n") %>% #Pivoting wider to easily manipulate data and generate the CFR variable  
 mutate(CFR=Yes/(Yes+No)) %>%   
 mutate(icu\_yn=factor(icu\_yn,levels=c("Yes","No"),labels=c("ICU Admission", "No ICU Admission"))) #Creating categories out of this variable  
  
#Creating the graph using the pivotting table  
ggplot(graph\_1\_pivot,aes(x=CFR,y=age\_group))+geom\_col(fill="cyan")+  
 geom\_text(aes(label = round(CFR\*100),x=CFR-0.025),colour = "black")+ #Adding the labels to each point in the column  
 facet\_grid(rows=vars(icu\_yn), cols=vars(sex),scales="free\_y")+#Faceting via ICU and Sex the same chart previously created  
 theme\_light()+  
 labs(  
 title="Covid CFR% by age group, sex, and ICU admission",  
 x=NULL,  
 y=NULL,  
 caption="Source:CDC")+  
 scale\_x\_continuous(labels=scales::percent) #Using x scale to be in percentage

## Warning: Removed 4 rows containing missing values (`position\_stack()`).

## Warning: Removed 4 rows containing missing values (`geom\_text()`).

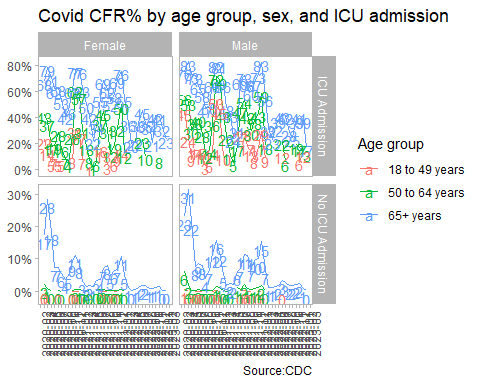


The previous plot is an aggregate plot for all three years of data. What if we wanted to plot Case Fatality Ratio (CFR) over time? Write code that collects the relevant data from the database and plots the following

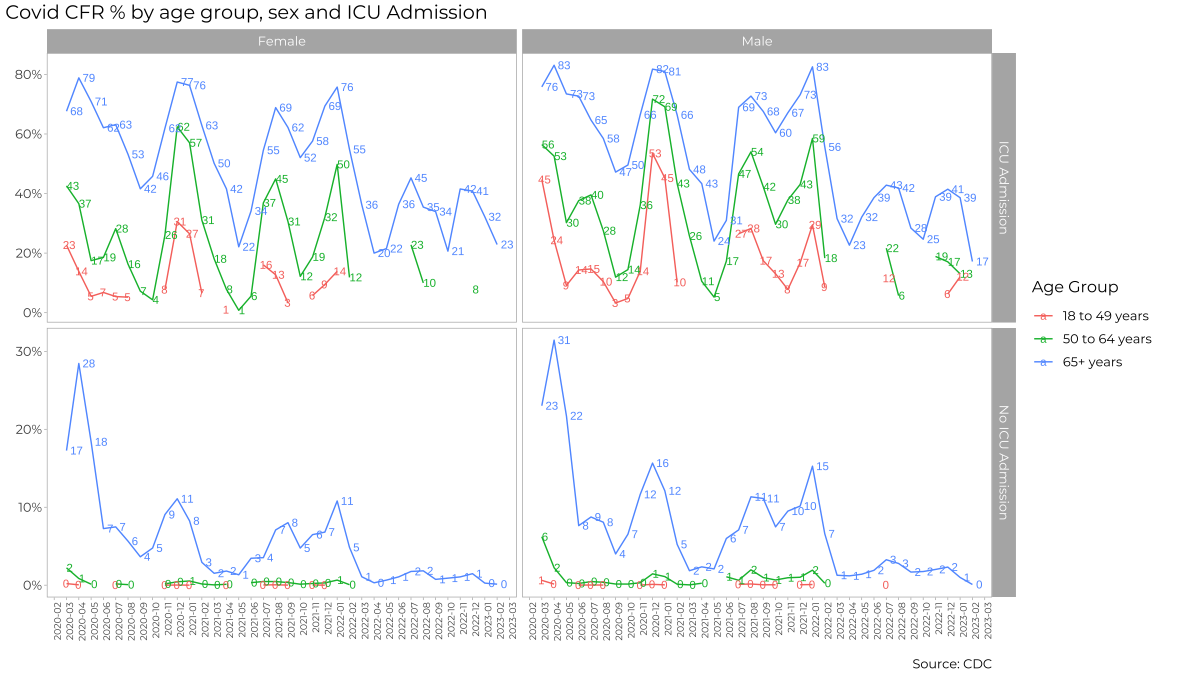
graph\_2<-cdc\_data %>%   
 filter(icu\_yn %in% c("Yes","No")) %>% #Cleansing data same as the previous graph  
 filter(!(age\_group %in% c("Missing","NA"))) %>%   
 filter(sex %in% c("Male","Female")) %>%   
 filter(death\_yn %in% c("Yes","No")) %>%   
 group\_by(death\_yn,icu\_yn,sex,age\_group,case\_month) %>% #Groupping differently to get to the final graph  
 count(death\_yn) %>%   
 collect()  
  
graph\_2\_pivot<-graph\_2 %>%   
 pivot\_wider(names\_from="death\_yn",values\_from="n") %>%   
 mutate(CFR=Yes/(Yes+No)) %>%   
 mutate(icu\_yn=factor(icu\_yn,levels=c("Yes","No"),labels=c("ICU Admission", "No ICU Admission"))) %>% #Factoring this variable  
 filter(age\_group!="0 - 17 years") #Taking out this age as there is no relevant information  
  
  
#Creating the second graph  
ggplot(graph\_2\_pivot,aes(x=case\_month,y=CFR,group=age\_group))+  
 geom\_line(aes(color=age\_group))+  
 geom\_text(aes(label = round(CFR\*100),y=CFR-0.025,color=age\_group))+#Adding the label in colors  
 facet\_grid(rows=vars(icu\_yn), cols=vars(sex),scales="free\_y")+ #Gridding through these variables  
 theme\_light(base\_size=11)+  
 theme(axis.text.x = element\_text(angle = 90),panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank())+ #Rotating the x axis dates vertically  
 labs(  
 title="Covid CFR% by age group, sex, and ICU admission",  
 colour = "Age group",  
 x=NULL,  
 y=NULL,  
 caption="Source:CDC")+  
 scale\_y\_continuous(labels=scales::percent) #Putting y scale in percentage

## Warning: Removed 24 rows containing missing values (`geom\_line()`).

## Warning: Removed 141 rows containing missing values (`geom\_text()`).



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



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()

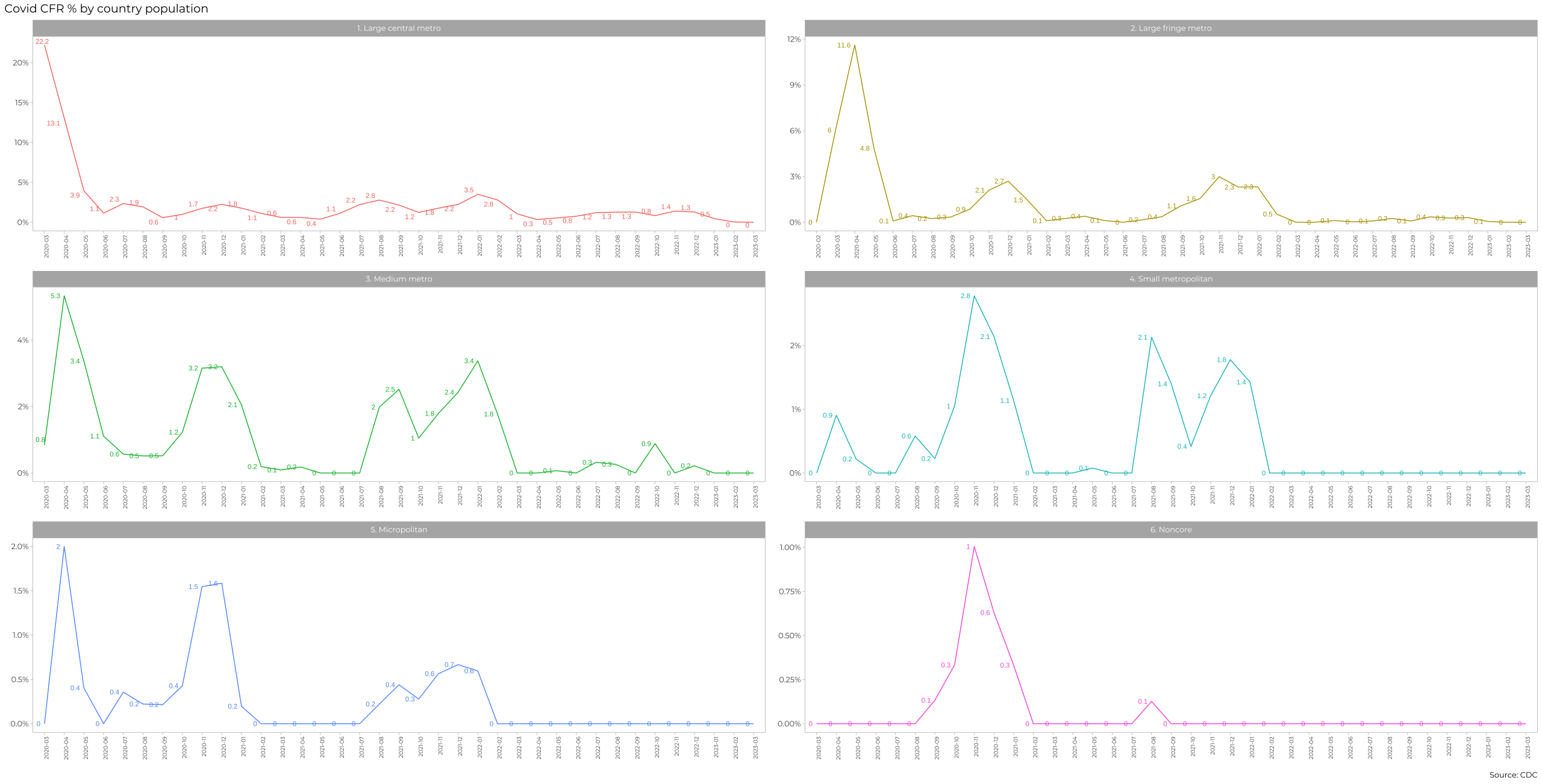
Each county belongs in seix diffent categoreis, with categories 1-4 being urban areas and categories 5-6 being rural, according to the following criteria captured in x2013\_code

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?

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



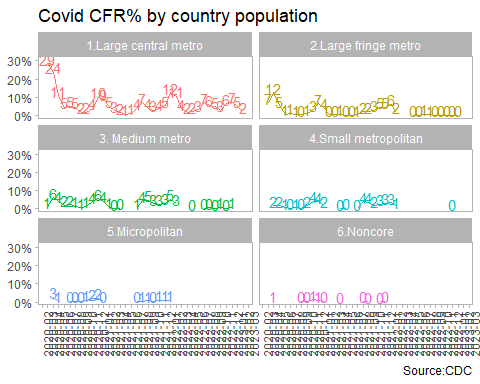
#Keeping just the columns I want to use  
new\_table\_urban<-urban\_rural %>%   
 select(fips\_code,x2013\_code)  
  
  
graph\_3<-cdc\_data %>%   
 filter(icu\_yn %in% c("Yes","No")) %>% #Cleansing data as the previous graphs  
 filter(!(age\_group %in% c("Missing","NA"))) %>%  
 filter(sex %in% c("Male","Female")) %>%   
 filter(death\_yn %in% c("Yes","No")) %>%   
 group\_by(death\_yn,county\_fips\_code,case\_month) %>% #Different groupping  
 count(death\_yn) %>%   
 collect() %>% #Collecting before left joining  
 left\_join(new\_table\_urban,by=c("county\_fips\_code"="fips\_code")) %>%   
 drop\_na(county\_fips\_code) %>% # Dropping NA  
 group\_by(x2013\_code,death\_yn,case\_month) %>%   
 summarise(n=sum(n))

## `summarise()` has grouped output by 'x2013\_code', 'death\_yn'. You can override  
## using the `.groups` argument.

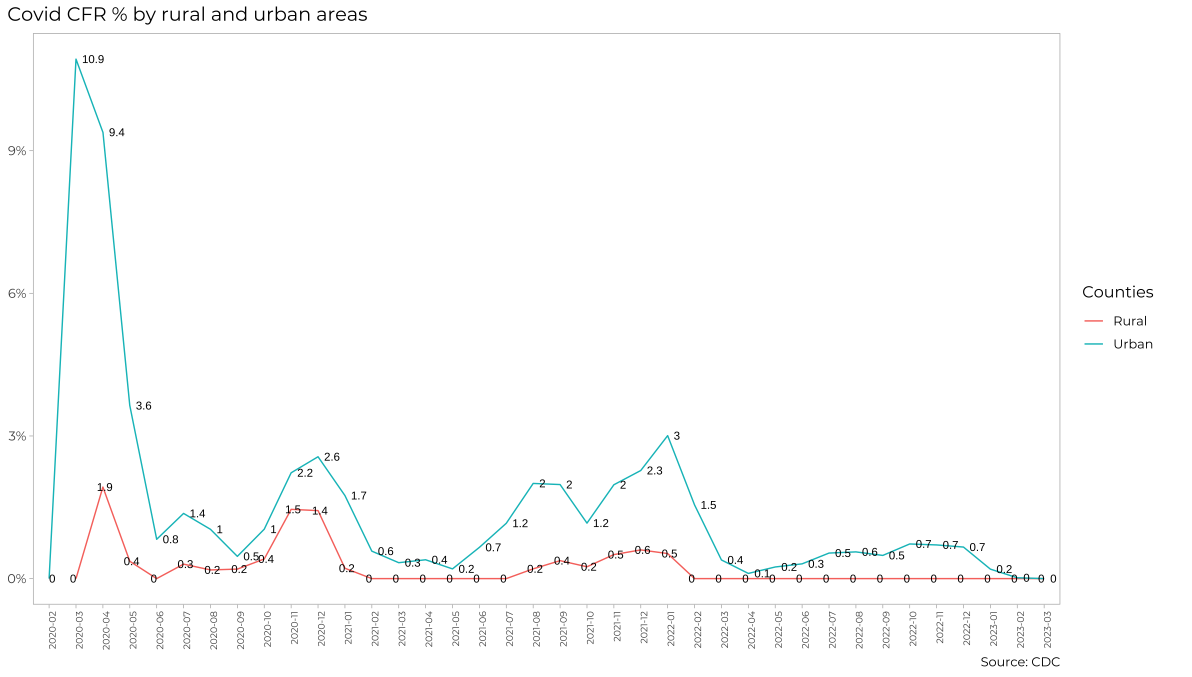
graph\_3\_pivot<-graph\_3 %>%   
 pivot\_wider(names\_from="death\_yn",values\_from="n") %>% #Creating a wider table that is easier to manipulate for graphing  
 mutate(CFR=Yes/(Yes+No)) %>%   
 mutate(x2013\_code=factor(x2013\_code,levels=c(1,2,3,4,5,6),labels=c("1.Large central metro","2.Large fringe metro","3. Medium metro","4.Small metropolitan","5.Micropolitan","6.Noncore"))) #Creating factors and ordering  
   
#Creating third graph  
ggplot(graph\_3\_pivot,aes(x=case\_month,y=CFR,group=x2013\_code))+  
 geom\_line(aes(color=x2013\_code))+  
 geom\_text(aes(label = round(CFR\*100),y=CFR+0.025,color=x2013\_code))+#Adding the label in colors  
 facet\_wrap(~x2013\_code,ncol=2)+ #Gridding through these variables with 2 columns  
 theme\_light(base\_size=11)+  
 theme(axis.text.x = element\_text(angle = 90),panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank(),legend.position="none")+ #Rotating the x axis dates vertically, deleting gridlines and hiding the legend  
 labs(  
 title="Covid CFR% by country population",  
 x=NULL,  
 y=NULL,  
 caption="Source:CDC")+  
 scale\_y\_continuous(labels=scales::percent) #Putting y scale in percentage

## Warning: Removed 43 rows containing missing values (`geom\_line()`).

## Warning: Removed 78 rows containing missing values (`geom\_text()`).



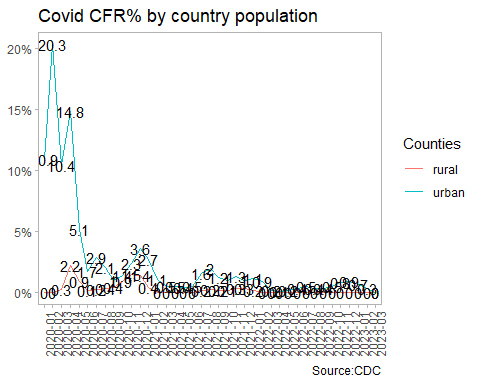
knitr::include\_graphics(here::here("images", "cfr-rural-urban.png"), error = FALSE)



#Selecting only variables from database and classifying them into urban (1-4) and rural  
new\_table\_urban2<-urban\_rural %>%   
 select(fips\_code,x2013\_code) %>%   
 mutate(code=case\_when(  
 x2013\_code %in% c(1,2,3,4)~ "urban",  
 .default="rural"  
 ))   
  
  
#Creating database and lefjoining it with the previously created one  
graph\_4<-cdc\_data %>%   
 filter(death\_yn %in% c("Yes","No")) %>%   
 group\_by(death\_yn,county\_fips\_code,case\_month) %>%   
 count(death\_yn) %>%   
 collect() %>%   
 left\_join(new\_table\_urban2,by=c("county\_fips\_code"="fips\_code")) %>%   
 drop\_na(county\_fips\_code) %>%   
 group\_by(code,death\_yn,case\_month) %>%   
 summarise(n=sum(n))

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

graph\_4\_pivot<-graph\_4 %>% #Creating it wider table  
 pivot\_wider(names\_from="death\_yn",values\_from="n") %>%   
 mutate(CFR=case\_when((Yes+No)!=0~Yes/(Yes+No),  
 .default=0))   
  
  
ggplot(graph\_4\_pivot,aes(x=case\_month,y=CFR,group=code))+  
 geom\_line(aes(color=code))+  
 geom\_text(aes(label = round(CFR\*100,1),y=CFR))+  
 theme\_light(base\_size=11)+  
 theme(axis.text.x = element\_text(angle = 90),panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank())+ #Rotating the x axis dates vertically, deleting gridlines  
 labs(  
 title="Covid CFR% by country population",  
 x=NULL,  
 y=NULL,  
 color="Counties",  
 caption="Source:CDC")+  
 scale\_y\_continuous(labels=scales::percent) #Putting y scale in percentage



# 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.

#Reading the table from the base\_url  
tables <- contributions\_tables %>%  
 html\_nodes(css="table") %>%   
 html\_table()   
   
#Creating table from above and naming it "contributions"  
contributions<-tables[[1]] %>% janitor::clean\_names()

* 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.

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")  
  
contributions\_all<-data.frame() #generating an empty dataframe that will be populated with the below function  
  
#Creating the desired function  
scrape\_pac<-function(url){  
 contributions\_tables <- url %>%  
 read\_html() %>%   
 html\_nodes(css="table") %>% #Reading the table from the url input  
 html\_table()  
   
 contributions<-contributions\_tables[[1]] %>% janitor::clean\_names() #creating a contributions table  
   
 contributions <- contributions %>% #Re-using the above function  
 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)  
 )  
return(contributions)  
}  
  
#Iterating in the 3 elements of the urls vector to add them all together in the previously generated empty dataframe  
for (i in 1:3){  
 contributions\_all<-rbind(contributions\_all,scrape\_pac(urls[i]))  
}  
  
#Writing CSV  
write.csv(contributions\_all,here::here("data", "contributions-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 %>% #Reading html  
 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?

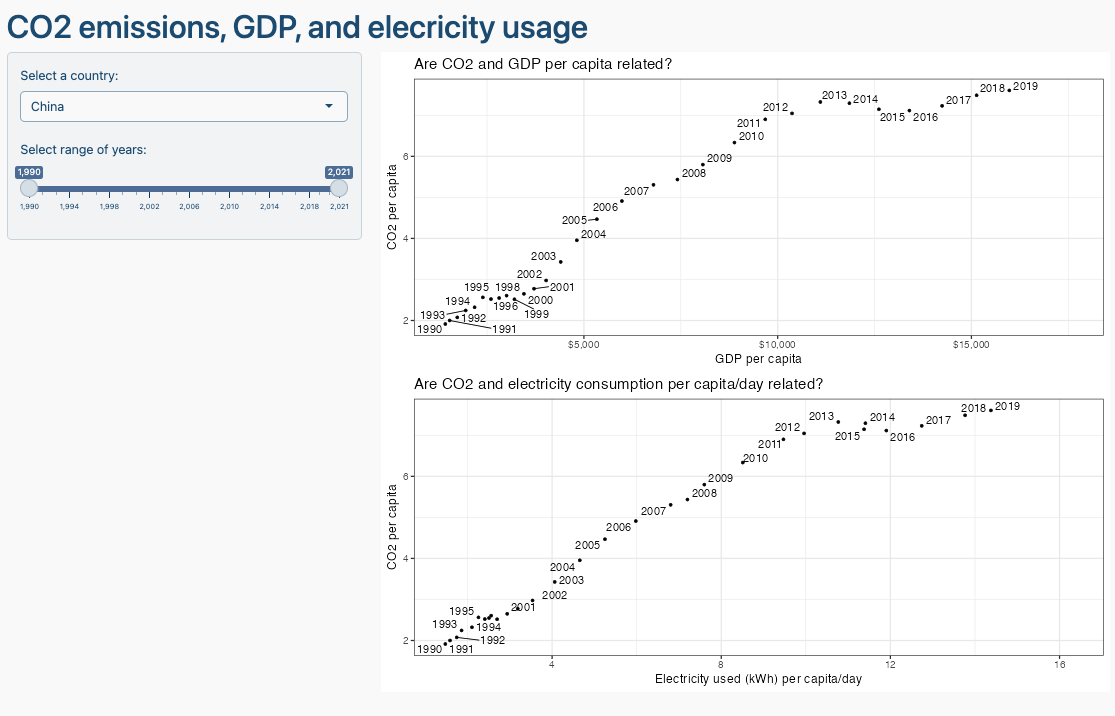
base\_url <- "https://www.consultancy.uk/jobs/page/1"  
url <- str\_c(base\_url, page)

* 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.

all\_consulting\_jobs<-data.frame() #Creating empty dataframe  
  
#Creating a function that will extract table from each url- same logic as the function above, the input will be numeric and will be added to the base\_url to move to all the 8 pages  
scrape\_jobs<-function(page){  
 base\_url <- "https://www.consultancy.uk/jobs/page/"  
 url <- str\_c(base\_url, as.character({{page}})) #adding the page number and converting it to character  
 trabajos\_tabla<- url %>%  
 read\_html() %>%   
 html\_nodes(css="table") %>%   
 html\_table()  
 trabajos<-trabajos\_tabla[[1]]  
 return(trabajos)  
 }  
  
#Using for loop in the 8 base urls and merging them in the empty dataset  
for (i in 1:8){  
 all\_consulting\_jobs<-rbind(all\_consulting\_jobs,scrape\_jobs(i))  
}  
  
#Writing CSV  
  
write.csv(all\_consulting\_jobs,here::here("data", "all\_consulting\_jobs.csv"))

# Create a shiny app

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: Myself
* Approximately how much time did you spend on this problem set: 5-6 hours
* What, if anything, gave you the most trouble: web scrapping and empty dataframes

**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.