Problem set 5 sample solutions

2024-10-11

Introduction

In this problem set, we aim to use data visualization to explore the following questions:

1. Based on SARS-Cov-2 cases, COVID-19 deaths and hospitalizations what periods defined the worst two waves of 2020-2021?
2. Did states with higher vaccination rates experience lower COVID-19 death rates?
3. Were there regional differences in vaccination rates?

We are not providing definitive answers to these questions but rather generating visualizations that may offer insights.

Objective

We will create a single data frame that contains relevant observations for each jurisdiction, for each Morbidity and Mortality Weekly Report (MMWR) period in 2020 and 2021. The key outcomes of interest are:

- SARS-CoV-2 cases
- COVID-19 hospitalizations
- COVID-19 deaths
- Individuals receiving their first COVID-19 vaccine dose
- Individuals receiving a booster dose

Task Breakdown

Your task is divided into three parts:

1. **Download the data**: Retrieve population data from the US Census API and COVID-19 statistics from the CDC API.

2. **Wrangle the data**: Clean and join the datasets to create a final table containing all the necessary information.
3. **Create visualizations**: Generate graphs to explore potential insights into the questions posed above.

Instructions

- **Create a Git repository** that includes the following directories:
  - `data`
  - `docs`
  - `figs`
- Inside the `code` directory, include the following files:
  - `funcs.R`
  - `wrangle.R`
  - `analysis.qmd`
- The `figs` directory should contain three PNG files, with each file corresponding to one of the figures you are asked to create.

Detailed instructions follow for each of the tasks.

Download data

For this part we want the following:

- Save all your code in a file called `wrangle.R` that produces the final data frame.
- When executed, this code should save the final data frame in an RDA file in the `data` directory.

1. Copy the relevant code from the previous homework to create the `population` data frame. Put this code in the the `wrangle.R` file in the `code` directory. Comment the code so we know where the population is create, where the regions are read in, and where we combine these.

Solution: The following code should be in the `wrangle.R`:

```
library(httr2)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(jsonlite)
```

```
Attaching package: 'jsonlite'

The following object is masked from 'package:purrr':

    flatten
```

```
library(janitor)
```

```
Attaching package: 'janitor'

The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```
library(purrr)
source("census-key.R")

## Read-in population
api <- "https://api.census.gov/data/2021/pep/population"
request <- request(api) |>
  req_url_query(get = I("POP_2020,POP_2021,NAME"),
                `for` = I("state:*"),
                key = census_key)
response <- request |> req_perform()
```

```r
population <- resp_body_json(response, simplifyVector = TRUE) |>
 janitor::row_to_names(1) |>
  as_tibble() |>
  select(-state) |>
  rename(state_name = NAME) |>
  pivot_longer(-state_name, names_to = "year", values_to = "population") |>
  mutate(year = str_remove(year, "POP_")) |>
  mutate(across(-state_name, as.numeric)) |>
  mutate(state = state.abb[match(state_name, state.name)]) |>
  mutate(state = case_when(
    state_name == "District of Columbia" ~ "DC",
    state_name == "Puerto Rico" ~ "PR",
    .default = state))

## Read-in regions
url <-
 ↪  "https://github.com/datasciencelabs/2024/raw/refs/heads/main/data/regions.json"
regions <- fromJSON(url, simplifyDataFrame = FALSE) |>
  map_df(function(x){
  data.frame(state_name = x$states, region = x$region, region_name =
   ↪  x$region_name)
}) |>
  mutate(region = factor(as.numeric(region))) |>
  mutate(region_name = ifelse(nchar(region_name) > 50, "NY,NJ,PR,USVI",
   ↪  region_name))

## Join tables
population <- left_join(population, regions, by = "state_name")
```

2. In the previous problem set we wrote the following script to download cases data:

```r
api <- "https://data.cdc.gov/resource/pwn4-m3yp.json"
cases_raw <- request(api) |>
  req_url_query("$limit" = 10000000) |>
  req_perform() |>
  resp_body_json(simplifyVector = TRUE)
```

We are now going to download three other datasets from CDC that provide hospitalization, provisional COVID deaths, and vaccine data. A different endpoint is provided for each one, but the requests are the same otherwise. To avoid rewriting the same code more than once,

write a function called `get_cdc_data` that receives and endpoint and returns a data frame. Save this code in a file called `functions.R`.

Solution:

The following code should be in the `funcs.R` file in the `code` directory:

```
get_cdc_data <- function(api){
  request(api) |>
    req_url_query("$limit" = 10000000) |>
    req_perform() |>
    resp_body_json(simplifyVector = TRUE)
}
```

The file `source("funcs.R")` should be added to the `wrangle.R` file.

3. Use the `get_cdc` Download the cases, hospitalization, deaths, and vaccination data and save the data frames. We recommend saving them into objects called: `cases_raw`, `hosp_raw`, `deaths_raw`, and `vax_raw`.

- cases - `https://data.cdc.gov/resource/pwn4-m3yp.json`
- hospitalizations - `https://data.cdc.gov/resource/39z2-9zu6.json`
- deaths - `https://data.cdc.gov/resource/r8kw-7aab.json`
- vaccinations `https://data.cdc.gov/resource/rh2h-3yt2.json`

We recommend saving them into objects called: `cases_raw`, `hosp_raw`, `deaths_raw`, and `vax_raw`. Add the code to the `wranling.R` file. Add comments to describe we read in data here.

```
# source("./code/funcs.R")
source("./pset-05-code/funcs.R")
cases_raw <- get_cdc_data("https://data.cdc.gov/resource/pwn4-m3yp.json")
hosp_raw <- get_cdc_data("https://data.cdc.gov/resource/39z2-9zu6.json")
deaths_raw <- get_cdc_data("https://data.cdc.gov/resource/r8kw-7aab.json")
vax_raw <- get_cdc_data("https://data.cdc.gov/resource/rh2h-3yt2.json")
```

Wrangling Challenge

In this section, you will wrangle the files downloaded in the previous step into a single data frame containing all the necessary information. We recommend using the following column names: `date`, `state`, `cases`, `hosp`, `deaths`, `vax`, `booster`, and `population`.

Key Considerations

- **Align reporting periods**: Ensure that the time periods for which each outcome is reported are consistent. Specifically, calculate the totals for each Morbidity and Mortality Weekly Report (MMWR) period.
- **Harmonize variable names**: To facilitate the joining of datasets, rename variables so that they match across all datasets.

4. One challenge is data frames use different column names to represent the same variable. Examine each data frame and report back 1) the name of the column with state abbreviations, 2) if the it's yearly, monthly, or weekly, daily data, 3) all the column names that provide date information.

Outcome	Jurisdiction variable name	Rate	time variable names
cases	`state`	weekly	`start_date`,`end_date`
hospitalizations	`jurisdiction`	daily	`collection_date`
deaths	`state`	weekly	`start_date`, `end_date`, `year`, `mmwr_week`, `week_ending_date`
vaccines	`location`	daily	`date`, `mmwr_week`

5. Wrangle the cases data frame to keep state MMWR year, MMWR week, and the total number of cases for that week in that state. Keep only states for which we have population estimates. Hint: Use `as_date`, `ymd_hms`, `epiweek` and `epiyear` functions in the **lubridate** package. Comment appropriately.

```
library(lubridate)
cases <- cases_raw |> mutate(cases = parse_number(new_cases),
                             date = as_date(ymd_hms(end_date))) |>
  filter(state %in% population$state) |>
  mutate(mmwr_week = epiweek(date), mmwr_year = epiyear(date)) |>
  select(state, mmwr_year, mmwr_week, cases) |>
  arrange(state, mmwr_year, mmwr_week)
```

6. Now repeat the same exercise for hospitalizations. Note that you will have to collapse the data into weekly data and keep the same columns as in the cases dataset, except keep total weekly hospitalizations instead of cases. Remove weeks with less than 7 days reporting. Add this code to `wrangle.R` and comment appropriately.

```
hosp <- hosp_raw |>
  filter(jurisdiction %in% population$state) |>
  rename(hosp = new_covid_19_hospital, state = jurisdiction) |>
  mutate(hosp = parse_number(hosp),
         date =  as_date(ymd_hms(collection_date)),
         mmwr_week = epiweek(date), mmwr_year = epiyear(date)) |>
  select(state, mmwr_year, mmwr_week, hosp) |>
  group_by(state, mmwr_year, mmwr_week) |>
  summarize(hosp = sum(hosp), n = n(), .groups = "drop") |>
  filter(n == 7) |>
  select(-n) |>
  arrange(mmwr_year, mmwr_week)
```

7. Repeat what you did in the previous two exercises for provisional COVID-19 deaths. Add this code to `wrangle.R` and comment appropriately.

```
deaths <- deaths_raw |>
  filter(state %in% population$state_name) |>
  mutate(end_date = as_date(end_date),
         mmwr_year = epiyear(end_date)) |>
  rename(deaths_prov = covid_19_deaths,
         flu = influenza_deaths) |>
  mutate(mmwr_week = parse_number(mmwr_week),
        deaths = parse_number(deaths_prov)) |>
  select(state, mmwr_week, mmwr_year, deaths)
```

8. Repeat this now for vaccination data. Keep the variables `series_complete` and `booster` along with state and date. Add this code to `wrangle.R` and comment appropriately.

```
vax <- vax_raw |> filter(date_type == "Admin" & location %in%
 ↪  population$state) |>
  rename(state = location, series_complete = series_complete_cumulative,
         booster = booster_cumulative) |>
  mutate(date = as_date(ymd_hms(date)),
         mmwr_week = as.numeric(mmwr_week), mmwr_year = epiyear(date),
         series_complete = parse_number(series_complete),
         booster = parse_number(booster)) |>
  select(state, date, mmwr_week, mmwr_year, series_complete, booster) |>
  group_by(state, mmwr_week, mmwr_year) |>
  summarize(series_complete = max(series_complete),
            booster = max(booster),
```

```
            .groups = "drop") |>
  arrange(state, mmwr_year, mmwr_week)
```

9. Now we are ready to join the tables. We will only consider 2020 and 2021 as we don't have population sizes for 2020. However, because we want to guarantee that all dates are included we will create a data frame with all possible weeks. Add this code to your `wrangle.R` file. We can use this:

```
## Make dates data frame
all_dates <- data.frame(date = seq(make_date(2020, 1, 25),
                                   make_date(2021, 12, 31),
                                   by = "week")) |>
  mutate(date = ceiling_date(date, unit = "week", week_start = 7) - days(1))
   ↪  |>
  mutate(mmwr_year = epiyear(date), mmwr_week = epiweek(date))

dates_and_pop <- cross_join(all_dates, data.frame(state =
 ↪  unique(population$state))) |> left_join(population, by = c("state",
 ↪  "mmwr_year" = "year"))
```

Now join all the table to create your final table. Make sure it is ordered by date within each state. Call it `dat` and save an RDS file to the `data` directory. Add this code to `wrangle.R` and comment appropriately.

```
dat <- dates_and_pop |>
  left_join(cases,  by = c("state", "mmwr_week", "mmwr_year")) |>
  left_join(hosp, by = c("state", "mmwr_week", "mmwr_year")) |>
  left_join(deaths,  by = c("state_name" = "state", "mmwr_week",
   ↪  "mmwr_year")) |>
  left_join(vax, by = c("state", "mmwr_week", "mmwr_year")) |>
  arrange(state, date)
```

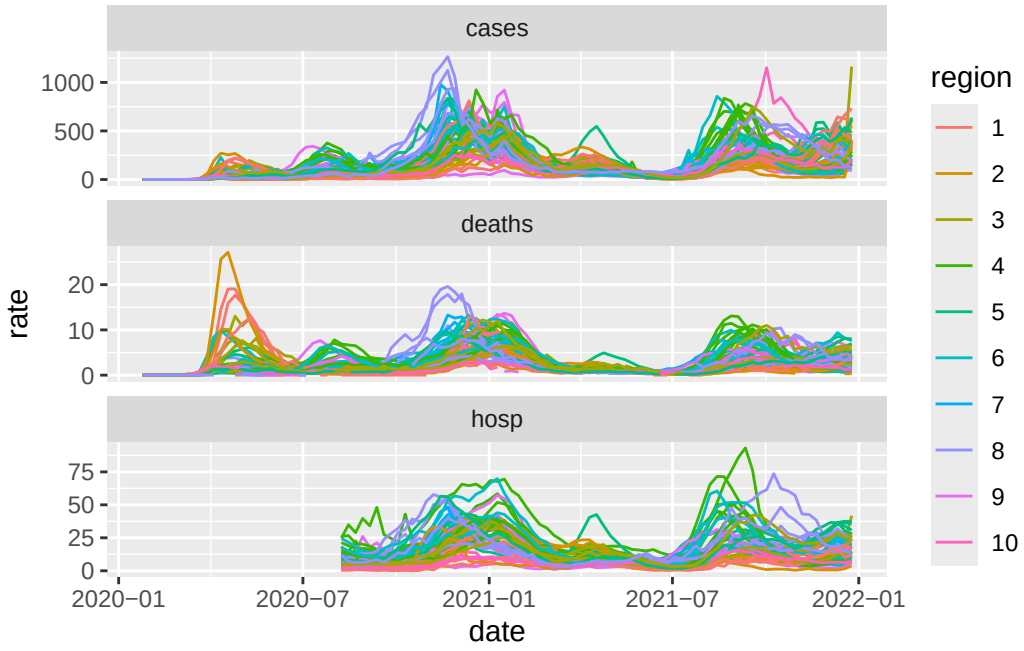
Data visualization generate some plots

We are now ready to create some figures. In the `analysis.qmd` file create a section for each figure. You should load the `dat` object stored in the RDS file in the `dat` directory.

You can call these sections Figure 1, Figure 2, and so on. Inlcude a short description of what the figure is before the code chunk. The rendered file should show both the code and figure.

10. Plot a trend plot for cases, hospitalizations and deaths. Plot rates per 100,000 people. Place the plots on top of each other. Hint: Use `pivot_longer` and `facet_wrap`.

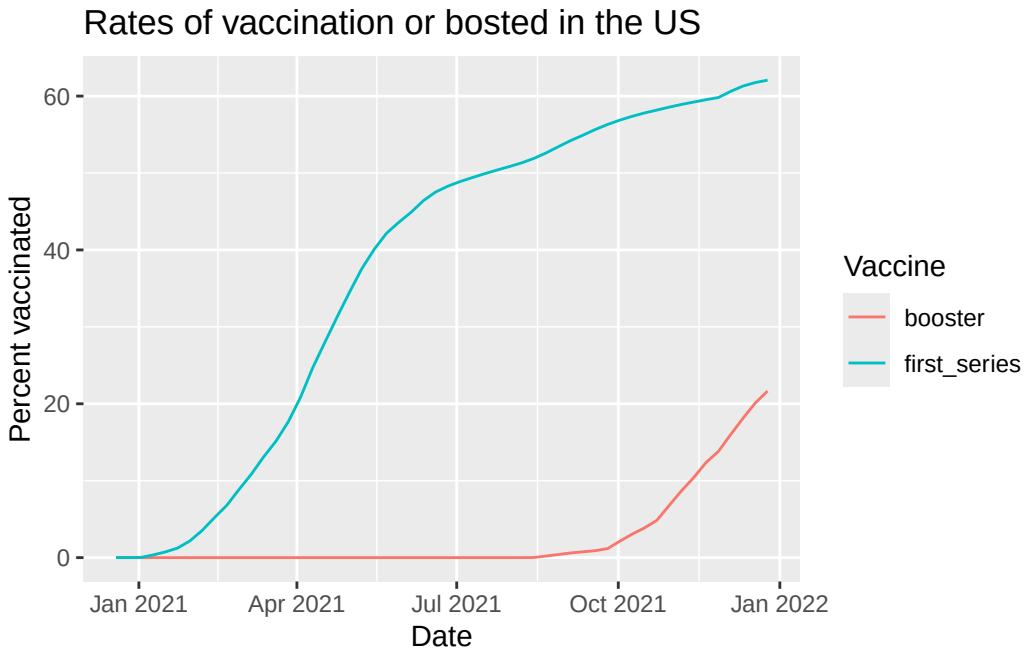
```
p <- dat |> mutate(cases =  cases/population*100000,
                   hosp = hosp/population*100000,
                  deaths = deaths/population*100000) |>
  select(date, cases, hosp, deaths, state, region) |>
  pivot_longer(c(cases, deaths, hosp), values_to = "rate", names_to =
   ↪  "outcome") |>
  ggplot(aes(date, rate, color = region, group = state)) +
  geom_line() +
  facet_wrap(~outcome, nrow = 3, scales = "free_y")
print(p)
```



11. To determine when vaccination started and when most of the population was vaccinated, compute the percent of the US population (including DC and Puerto Rico) were vaccinated by date. Do the same for the booster. Then plot both percentages.

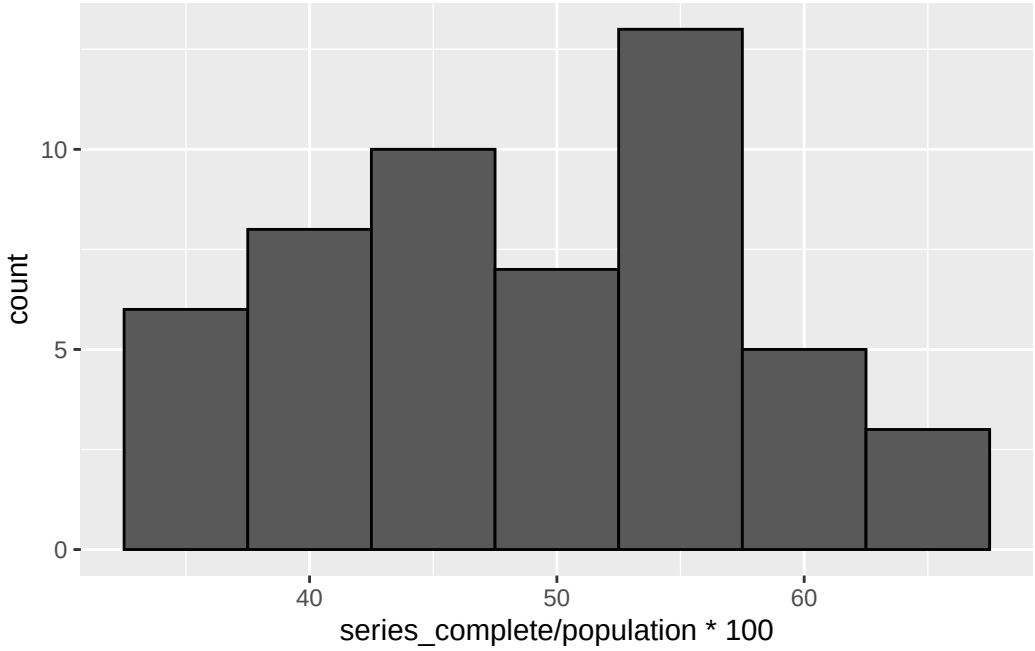
```
p <- dat |>
  filter(!is.na(series_complete)) |>
  group_by(date) |>
  summarize(first_series = sum(series_complete, na.rm =
   ↪  TRUE)/sum(population)*100,
```

```
            booster = sum(booster, na.rm = TRUE)/sum(population)*100) |>
  ungroup() |>
  pivot_longer(-date) |>
  ggplot(aes(date, value, color = name)) +
  geom_line() +
  labs(x = "Date", y = "Percent vaccinated", color = "Vaccine",
       title = "Rates of vaccination or bosted in the US")
print(p)
```



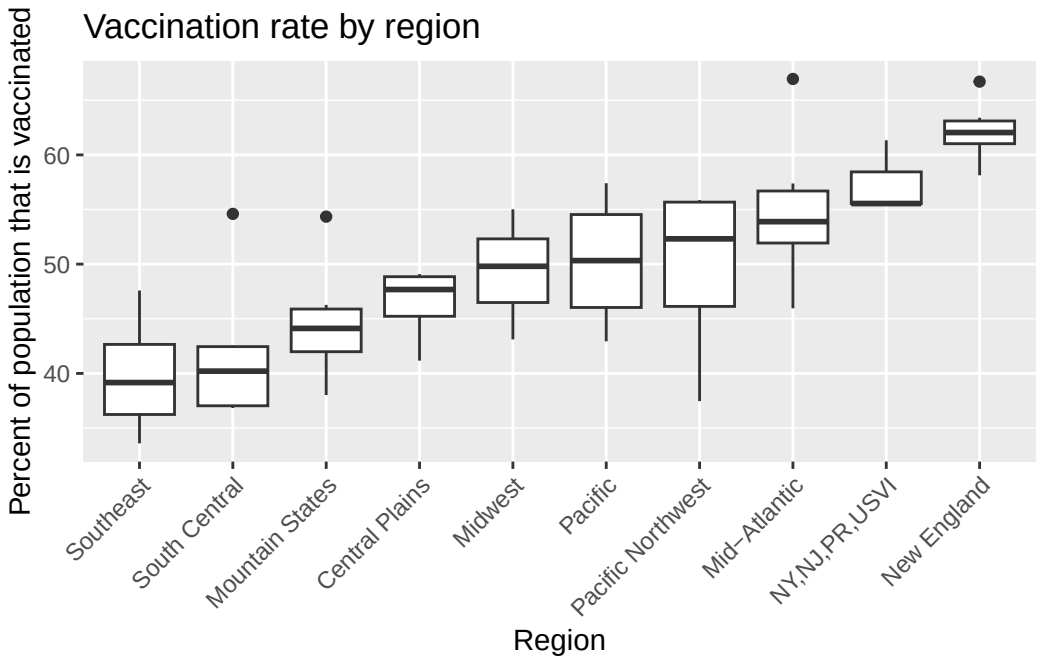
12. Describe the distribution of vaccination rates on July 1, 2021.

```
dat |>
  filter(mmwr_year == 2021 & mmwr_week == epiweek(make_date(2021, 7, 1))) |>
  ggplot(aes(series_complete/population*100)) +
  geom_histogram(color = I("Black"), binwidth = 5)
```



13. Is there a difference across region? Discuss what the plot shows?

```
dat |>
  filter(mmwr_year == 2021 & mmwr_week == epiweek(make_date(2021, 7, 1))) |>
  mutate(vax_rate = series_complete/population*100) |>
  mutate(region_name = reorder(region_name, vax_rate, median)) |>
  ggplot(aes(region_name, vax_rate)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Vaccination rate by region", x = "Region", y = "Percent of
   ↪  population that is vaccinated")
```



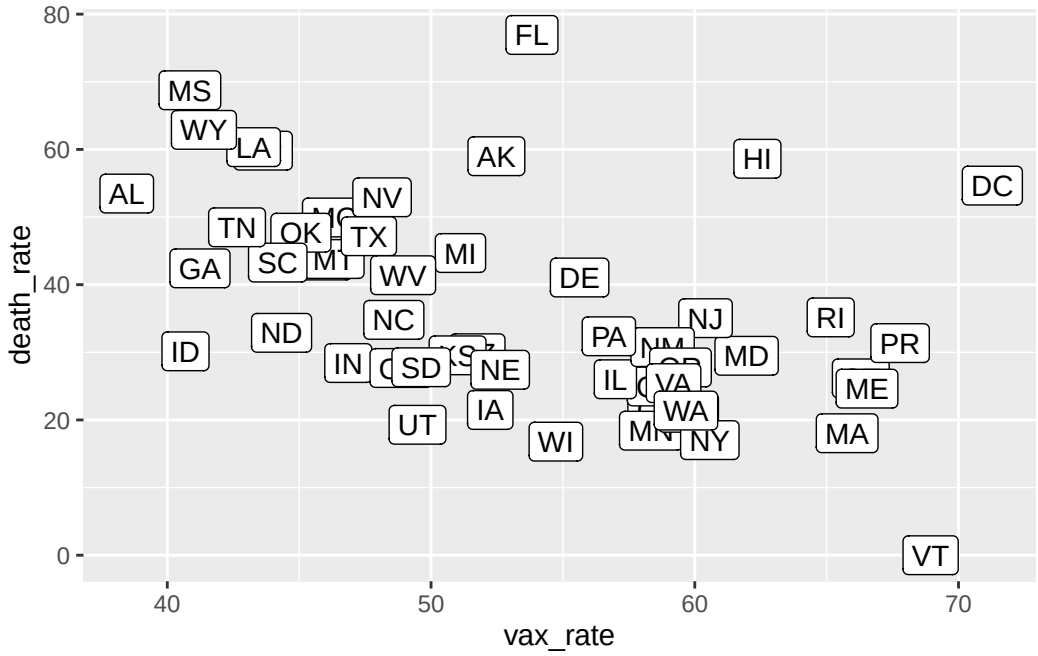
14. Using the two previous figures, identify two time periods that meet the following criteria:

- A significant COVID-19 wave occurred across the United States.
- A sufficient number of people had been vaccinated.

Next, follow these steps:

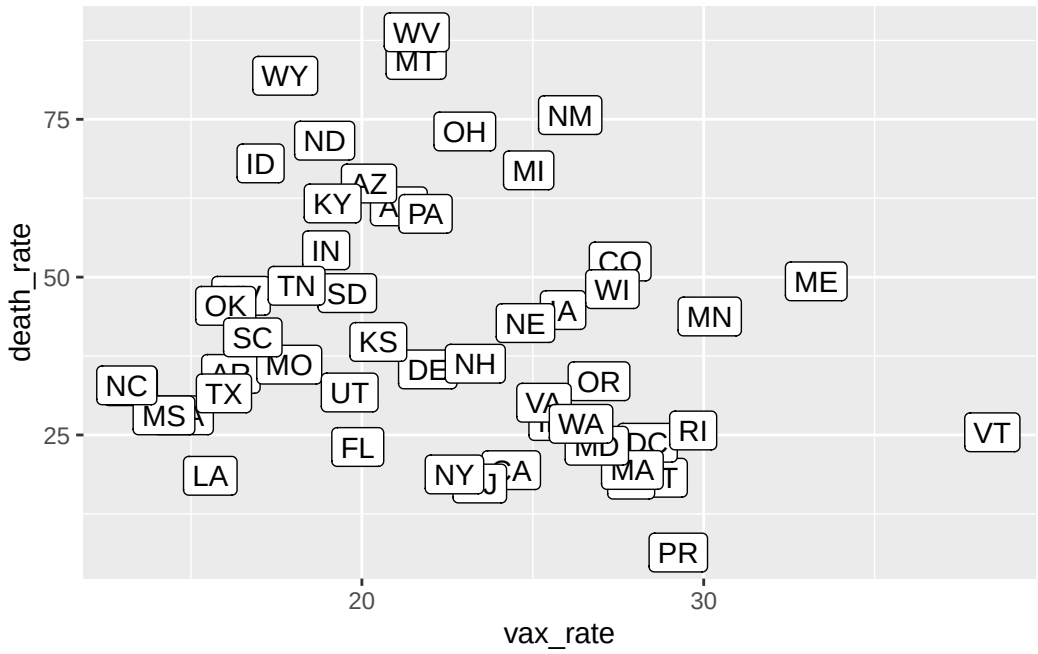
- For each state, calculate the **COVID-19 deaths per day per 100,000 people** during the selected time period.
- Determine the **vaccination rate (primary series)** in each state as of the last day of the period.
- Create a scatter plot to visualize the relationship between these two variables:
  - The **x-axis** should represent the vaccination rate.
  - The **y-axis** should represent the deaths per day per 100,000 people.

```
dat |>
  filter(date >= make_date(2021, 3, 1) & date <= make_date(2021, 9, 1)) |>
  group_by(state) |>
  summarize(death_rate = mean(deaths, na.rm = TRUE)*n() /
   ↪  population[1]*100000,
            vax_rate = max(series_complete)/population[1]*100) |>
  ggplot(aes(vax_rate, death_rate, label = state)) + geom_label()
```



14. Repeat the exercise for the booster.

```
dat |>
  filter(date >= make_date(2021, 10, 1) & date <= make_date(2021, 12, 31)) |>
  group_by(state) |>
  summarize(death_rate = mean(deaths, na.rm = TRUE)*n() /
   ↪  population[1]*100000,
            vax_rate = max(booster)/population[1]*100) |>
  ggplot(aes(vax_rate, death_rate, label = state)) + geom_label()
```


death_rate
75
50
25
20
30
vax_rate
WV
MT
WY
NM
OH
ND
ID
MI
AZ
KY
PA
IN
CO
ME
TN
SD
WI
OK
MN
NE
SC
KS
MO
DE
NH
NC
OR
TX
UT
VA
MS
WA
RI
VT
FL
LA
NY
MA
PR