Problem set 4

2025-02-09

Total points: 15.

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

- As usual, copy and place the `pset-04-dataviz.qmd` file in a new directory called `p4`.
- Within your `p4` directory, create the following directory:
  - `code`
- Inside the `code` directory, include the following files:
  - `funcs.R`
  - `wrangle.R`

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. (1 point) 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 created, where the regions are read in, and where we combine these.

Test that your wrangling code works. Comment the following code out:

```
# comment this out to run
source("./code/wrangle.R")
```

```
<httr2_response>

GET
https://api.census.gov/data/2021/pep/population?get=POP_2020%2CPOP_2021%2CNAME&for=state%3A%

Status: 200 OK

Content-Type: application/json

Body: In memory (2112 bytes)

-- 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

Attaching package: 'janitor'


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

    chisq.test, fisher.test

Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
`.name_repair` is omitted as of tibble 2.0.0.
i Using compatibility `.name_repair`.

# A tibble: 6 x 4
  state_name year  population state
  <chr>      <chr>      <dbl> <chr>
1 Oklahoma   2020     3962031 OK
2 Oklahoma   2021     3986639 OK
3 Nebraska   2020     1961455 NE
4 Nebraska   2021     1963692 NE
5 Hawaii     2020     1451911 HI
6 Hawaii     2021     1441553 HI
```

```r
head(population)
```

```
# A tibble: 6 x 4
  state_name year  population state
  <chr>      <chr>      <dbl> <chr>
1 Oklahoma   2020     3962031 OK
2 Oklahoma   2021     3986639 OK
3 Nebraska   2020     1961455 NE
4 Nebraska   2021     1963692 NE
5 Hawaii     2020     1451911 HI
6 Hawaii     2021     1441553 HI
```

2. (1 point) 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 `funcs.R`.

3. (1 point) Use the function `get_cdc_data` to 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`.

```r
source("./code/funcs.R")
```

```
Attaching package: 'jsonlite'

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

    flatten
```

```r
# Load necessary packages
library(dplyr)
library(readr)


# Fetch data from CDC API
cases_raw <- get_cdc_data("pwn4-m3yp")
hosp_raw <- get_cdc_data("39z2-9zu6")
deaths_raw <- get_cdc_data("r8kw-7aab")
vax_raw <- get_cdc_data("rh2h-3yt2")

save(cases_raw, file = "./data/cases_raw.rda")
save(hosp_raw, file = "./data/hosp_raw.rda")
save(deaths_raw, file = "./data/deaths_raw.rda")
save(vax_raw, file = "./data/vax_raw.rda")
```

Take a look at all the dataframes you just read in.

```r
### Uncomment this to run this
print("Cases Dataset:")
```

```
[1] "Cases Dataset:"
```

```r
print(head(cases_raw))
```

```
             date_updated state              start_date                end_date
1 2023-02-23T00:00:00.000    AZ 2023-02-16T00:00:00.000 2023-02-22T00:00:00.000
2 2022-12-22T00:00:00.000    LA 2022-12-15T00:00:00.000 2022-12-21T00:00:00.000
3 2023-02-23T00:00:00.000    GA 2023-02-16T00:00:00.000 2023-02-22T00:00:00.000
4 2023-03-30T00:00:00.000    LA 2023-03-23T00:00:00.000 2023-03-29T00:00:00.000
5 2023-02-02T00:00:00.000    LA 2023-01-26T00:00:00.000 2023-02-01T00:00:00.000
6 2023-03-23T00:00:00.000    LA 2023-03-16T00:00:00.000 2023-03-22T00:00:00.000
  tot_cases new_cases tot_deaths new_deaths new_historic_cases
```

```
1 2434631.0    3716.0    33042.0       39.0              23150
2 1507707.0    4041.0    18345.0       21.0              21397
3 3061141.0    5298.0    42324.0       88.0               6800
4 1588259.0    2203.0    18858.0       23.0               5347
5 1548508.0    5725.0    18572.0       47.0               4507
6 1580709.0    1961.0    18835.0       35.0               2239
  new_historic_deaths
1                   0
2                   0
3                   0
4                   0
5                   0
6                   0
```

```
print("Hospitalizations Dataset:")
```

```
[1] "Hospitalizations Dataset:"
```

```
print(head(hosp_raw))
```

```
          collection_date jurisdiction new_covid_19_hospital
1 2023-09-23T00:00:00.000     Region 5                   418
2 2023-09-24T00:00:00.000     Region 5                   319
3 2023-09-25T00:00:00.000     Region 5                   365
4 2023-09-26T00:00:00.000     Region 5                   407
5 2023-09-27T00:00:00.000     Region 5                   414
6 2023-09-28T00:00:00.000     Region 5                   377
  new_covid_19_hospital_1 cumulative_covid_19_hospital
1      389.14285714285717                       1032508
2       389.2857142857143                       1032827
3                   395.0                       1033192
4      395.14285714285717                       1033599
5       395.7142857142857                       1034013
6      389.42857142857144                       1034390
  cumulative_covid_19_hospital_1 new_covid_19_hospital_2
1              1965.1074606644204      5.1844176731317155
2              1965.7145932773901      5.1863209101629675
3               1966.409274793797       5.262450391413067
4              1967.1838922655168       5.264353628444319
5              1967.9718323964555       5.271966576569329
6              1968.6893527572374        5.18822414719422
```

```
  new_covid_19_hospital_3 new_covid_19_hospital_4 total_hospitalized_covid
1      5.1844176731317155      5.1844176731317155                      2068
2      5.1863209101629675      5.1863209101629675                      2083
3       5.262450391413067       5.262450391413067                      2174
4       5.264353628444319       5.264353628444319                      2152
5       5.271966576569329       5.271966576569329                      2162
6        5.18822414719422        5.18822414719422                      2093
  total_hospitalized_covid_1 covid_19_inpatient_bed covid_19_inpatient_bed_1
1          1.870273923891379      1.870273923891379      0.09361361930724477
2          1.8851070852003566     1.8851070852003566     0.09186845392105569
3          1.9075027355011414     1.9075027355011414     0.10331496654982586
4          1.9255064482138189     1.9255064482138189     0.10896499704873186
5          1.9394246936011394     1.9394246936011394     0.11411168394863225
6          1.9398872335133022     1.9398872335133022     0.10257298885314814
  covid_19_icu_bed_occupancy covid_19_icu_bed_occupancy_1
1          1.870273923891379          0.04724889876464955
2          1.8851070852003566         0.05957545544748699
3          1.9075027355011414         0.04755327929547159
4          1.9255064482138189         0.05397827305300184
5          1.9394246936011394          0.07890068573990994
6          1.9398872335133022          0.08368752552323033
```

```
print("Deaths Dataset:")
```

```
[1] "Deaths Dataset:"
```

```
print(head(deaths_raw))
```

```
              data_as_of              start_date                end_date
1 2025-02-06T00:00:00.000 2019-12-29T00:00:00.000 2020-01-04T00:00:00.000
2 2025-02-06T00:00:00.000 2020-01-05T00:00:00.000 2020-01-11T00:00:00.000
3 2025-02-06T00:00:00.000 2020-01-12T00:00:00.000 2020-01-18T00:00:00.000
4 2025-02-06T00:00:00.000 2020-01-19T00:00:00.000 2020-01-25T00:00:00.000
5 2025-02-06T00:00:00.000 2020-01-26T00:00:00.000 2020-02-01T00:00:00.000
6 2025-02-06T00:00:00.000 2020-02-02T00:00:00.000 2020-02-08T00:00:00.000
    group      year mmwr_week        week_ending_date         state
1 By Week 2019/2020         1 2020-01-04T00:00:00.000 United States
2 By Week      2020         2 2020-01-11T00:00:00.000 United States
3 By Week      2020         3 2020-01-18T00:00:00.000 United States
4 By Week      2020         4 2020-01-25T00:00:00.000 United States
5 By Week      2020         5 2020-02-01T00:00:00.000 United States
```

```
6 By Week      2020         6 2020-02-08T00:00:00.000 United States
  covid_19_deaths total_deaths percent_of_expected_deaths pneumonia_deaths
1               0        60170                      98.00             4111
2               1        60734                      97.00             4153
3               2        59362                      98.00             4066
4               3        59162                      99.00             3915
5               0        58834                      99.00             3818
6               4        59482                        100             3823
  pneumonia_and_covid_19_deaths influenza_deaths
1                             0              434
2                             1              475
3                             2              468
4                             0              500
5                             0              481
6                             1              520
  pneumonia_influenza_or_covid_19_deaths footnote month
1                                   4545     <NA>  <NA>
2                                   4628     <NA>  <NA>
3                                   4534     <NA>  <NA>
4                                   4418     <NA>  <NA>
5                                   4299     <NA>  <NA>
6                                   4346     <NA>  <NA>
```

```
print("Vaccines Dataset:")
```

```
[1] "Vaccines Dataset:"
```

```
print(head(vax_raw))
```

```
                   date date_type mmwr_week location administered_daily
1 2023-05-10T00:00:00.000    Report        19       CO              15097
2 2023-05-10T00:00:00.000    Report        19       AZ              16505
3 2023-05-10T00:00:00.000    Report        19       MN              16020
4 2023-05-10T00:00:00.000    Report        19       ID               3526
5 2023-05-10T00:00:00.000    Report        19       DC                 31
6 2023-05-10T00:00:00.000    Report        19       AK               1582
  administered_cumulative admin_dose_1_daily admin_dose_1_cumulative
1                13033446               1527                 4837792
2                14647405               2955                 5704677
3                12829141               1282                 4461994
4                 2894361                323                 1146055
```

```
5                2137377               264                  836680
6                1328221               130                  535718
  administered_dose1_pop_pct series_complete_daily series_complete_cumulative
1                       84.0                 1218                     4248431
2                       78.4                 1101                     4821350
3                       79.1                  932                     4082263
4                       64.1                  267                     1012257
5                       95.0                  212                      644085
6                       73.2                   86                      477592
  series_complete_pop_pct booster_daily booster_cumulative
1                    73.8          1569            2460212
2                    66.2          1401            2418342
3                    72.4          1698            2595884
4                    56.6           333             494214
5                    91.3           196             330888
6                    65.3           158             237614
  additional_doses_vax_pct second_booster_50plus_daily
1                     57.9                       1062
2                     50.2                       1312
3                     63.6                       1196
4                     48.8                        281
5                     51.4                        106
6                     49.8                        164
  second_booster_50plus_cumulative second_booster_50plus_vax_pct
1                           794838                          62.9
2                           794699                          54.3
3                           983284                          67.8
4                           173862                          54.8
5                            80880                          55.9
6                            69502                          54.7
  bivalent_booster_daily bivalent_booster_cumulative bivalent_booster_pop_pct
1                   9725                     1272115                     22.1
2                  11388                     1148060                     15.8
3                   5497                     1510743                     26.8
4                   2032                      248989                     13.9
5                    509                      226857                     32.1
6                    640                      103624                     14.2
  administered_7_day_rolling admin_dose_1_day_rolling
1                       <NA>                    <NA>
2                       <NA>                    <NA>
3                       <NA>                    <NA>
4                       <NA>                    <NA>
5                       <NA>                    <NA>
```

```
6                       <NA>                       <NA>
  series_complete_day_rolling booster_7_day_rolling_average
1                        <NA>                          <NA>
2                        <NA>                          <NA>
3                        <NA>                          <NA>
4                        <NA>                          <NA>
5                        <NA>                          <NA>
6                        <NA>                          <NA>
  second_booster_50plus_7_day_rolling_average
1                                        <NA>
2                                        <NA>
3                                        <NA>
4                                        <NA>
5                                        <NA>
6                                        <NA>
  bivalent_booster_7_day_rolling_average administered_daily_change
1                                   <NA>                      <NA>
2                                   <NA>                      <NA>
3                                   <NA>                      <NA>
4                                   <NA>                      <NA>
5                                   <NA>                      <NA>
6                                   <NA>                      <NA>
  administered_daily_change_1
1                        <NA>
2                        <NA>
3                        <NA>
4                        <NA>
5                        <NA>
6                        <NA>
```

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. (1 point) 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 rate is 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	new_cases	`date_updated`, `start_date`, `end_date`
hospitalizations	jurisdiction	new_covid_19_hospita	collection_date
deaths	state	covid_19_deaths	`week_ending_date`, `start_date`, `end_date`
vaccines	location	administered_daily	`date`, `mmwr_week`

```
extract_data_info <- function(data, outcome_name) {
  jurisdiction_col <- colnames(data)[grepl("jurisdiction|state|location", colnames(data), ig
  time_vars <- colnames(data)[grepl("date|year|week|month|day", colnames(data), ignore.case =
  rate_type <- "Weekly"

  return(data.frame(
    Outcome = outcome_name,
    Jurisdiction_Variable = jurisdiction_col,
    Rate = rate_type,
    Time_Variable_Names = paste(time_vars, collapse = ", ")
  ))
}

cases_info <- extract_data_info(cases_raw, "Cases")
hosp_info <- extract_data_info(hosp_raw, "Hospitalizations")
deaths_info <- extract_data_info(deaths_raw, "Deaths")
vax_info <- extract_data_info(vax_raw, "Vaccines")

data_summary <- bind_rows(cases_info, hosp_info, deaths_info, vax_info)

print(data_summary)
```

```
            Outcome Jurisdiction_Variable   Rate
```

```
1            Cases                  state Weekly
2 Hospitalizations           jurisdiction Weekly
3           Deaths                  state Weekly
4         Vaccines               location Weekly

1
2
3
4 date, date_type, mmwr_week, administered_7_day_rolling, admin_dose_1_day_rolling, series_c
```

5. (1 point) Wrangle the cases data frame to keep state, MMWR year, MMWR week, and the total number of cases for that week in that state. Hint: Use `as_date`, `ymd_hms`, `epiweek` and `epiyear` functions in the **lubridate** package. Comment appropriately. Display the result.

```r
# Load required package
library(dplyr)
library(httr2)
library(jsonlite)

# Define function to fetch data from CDC API
get_cdc_data <- function(endpoint) {
  api_url <- paste0("https://data.cdc.gov/resource/", endpoint, ".json?$limit=100000")
  response <- request(api_url) %>%
    req_perform() %>%
    resp_body_json(simplifyVector = TRUE) %>%
    as_tibble()
  return(response)
}

# Fetch cases data
cases_raw <- get_cdc_data("pwn4-m3yp")

library(lubridate)
library(stringr)


cases_raw <- cases_raw %>%
  mutate(

    date_updated = as_date(str_sub(date_updated, 1, 10)),
    new_cases = as.numeric(new_cases)
```

```
  )
summary(cases_raw$new_cases)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
     0.0    665.8   3102.0  10082.6   9935.0 790954.0
```

```
summary(cases_raw$date_updated)
```

```
        Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
"2020-01-23" "2020-11-19" "2021-09-16" "2021-09-16" "2022-07-14" "2023-05-11"
```

```
cases_raw %>%
  filter(state == "AK") %>%
  select(state, date_updated, new_cases) %>%
  arrange(date_updated) %>%
  head(20)
```

```
# A tibble: 20 x 3
   state date_updated new_cases
   <chr> <date>           <dbl>
 1 AK    2020-01-23           0
 2 AK    2020-01-30           0
 3 AK    2020-02-06           0
 4 AK    2020-02-13           0
 5 AK    2020-02-20           0
 6 AK    2020-02-27           0
 7 AK    2020-03-05           0
 8 AK    2020-03-12           0
 9 AK    2020-03-19          11
10 AK    2020-03-26          52
11 AK    2020-04-02          86
12 AK    2020-04-09          86
13 AK    2020-04-16          65
14 AK    2020-04-23          37
15 AK    2020-04-30          18
16 AK    2020-05-07          19
17 AK    2020-05-14          15
18 AK    2020-05-21          15
19 AK    2020-05-28          23
20 AK    2020-06-04          91
```

```r
cases_raw <- cases_raw %>%
  mutate(new_cases = as.numeric(new_cases))

cases_weekly <- cases_raw %>%
  mutate(
    date_updated = as_date(date_updated),
    mmwr_year = epiyear(date_updated),
    mmwr_week = epiweek(date_updated),
    new_cases = as.numeric(new_cases)
  ) %>%
  group_by(state, mmwr_year, mmwr_week) %>%
  summarise(total_cases = sum(new_cases, na.rm = TRUE), .groups = "drop")

head(cases_weekly)
```

```
# A tibble: 6 x 4
  state mmwr_year mmwr_week total_cases
  <chr>     <dbl>     <dbl>       <dbl>
1 AK         2020         4           0
2 AK         2020         5           0
3 AK         2020         6           0
4 AK         2020         7           0
5 AK         2020         8           0
6 AK         2020         9           0
```

```r
save(cases_weekly, file = "./code/cases_weekly.rda")
```

6. (1 point) 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. Display your result and comment appropriately.

```r
library(dplyr)
library(lubridate)
hosp_raw <- get_cdc_data("39z2-9zu6")


# Process hospitalization data similar to cases data
hosp_weekly <- hosp_raw %>%
  mutate(
    # Convert collection_date to Date format
```

```
    date = as_date(collection_date),

    # Extract MMWR year and MMWR week
    mmwr_year = epiyear(date),
    mmwr_week = epiweek(date)
  ) %>%

  # Group by state, MMWR year, and MMWR week
  group_by(jurisdiction, mmwr_year, mmwr_week) %>%

  # Summarize total hospitalizations per week per state
  summarise(
    total_hospitalizations = sum(as.numeric(new_covid_19_hospital), na.rm = TRUE),
    num_days = n(),  # Count number of records in the week
    .groups = "drop"  # Ensure grouping is removed for further processing
  ) %>%

  # Keep only weeks with full 7-day reporting
  filter(num_days == 7) %>%

  # Remove the auxiliary column
  select(-num_days)

# Check again
summary(hosp_weekly$total_hospitalizations)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0     111     370    1612    1107  150650
```

```
head(hosp_weekly)
```

```
# A tibble: 6 x 4
  jurisdiction mmwr_year mmwr_week total_hospitalizations
  <chr>            <dbl>     <dbl>                  <dbl>
1 AK                2020        32                     28
2 AK                2020        33                     22
3 AK                2020        34                     31
4 AK                2020        35                     31
5 AK                2020        36                     35
6 AK                2020        37                     30
```

```r
save(hosp_weekly, file = "./code/hosp_weekly.rda")
```

7. (1 point) Repeat what you did in the previous two exercises for provisional COVID-19 deaths. Display the result and comment appropriately.

```r
deaths_raw <- get_cdc_data("r8kw-7aab")
deaths <- deaths_raw |>
  # Select required columns
  select(
    state,
    deaths = covid_19_deaths,
    date = week_ending_date
  ) |>

  mutate(
    date = as_date(ymd_hms(date)),
    mmwr_year = epiyear(date),
    mmwr_week = epiweek(date)
  ) |>
  # Group by state and MMWR periods and calculate weekly totals
  group_by(state, mmwr_year, mmwr_week) |>
  summarise(
    deaths = sum(as.numeric(deaths), na.rm = TRUE),  # sum weekly deaths
    .groups = "drop"
  )

deaths <- deaths |>
  mutate(state = state.abb[match(state, state.name)]) |>
  mutate(state = case_when(
    state == "District of Columbia" ~ "DC",
    state == "Puerto Rico" ~ "PC",
    TRUE ~ state
  ))

# Display result
print(head(deaths))
```

```
# A tibble: 6 x 4
  state mmwr_year mmwr_week deaths
  <chr>     <dbl>     <dbl>  <dbl>
1 AL         2020         1      0
2 AL         2020         2      0
```

```
3 AL          2020          3       0
4 AL          2020          4       0
5 AL          2020          5       0
6 AL          2020          6       0
```

```
save(deaths, file = "./code/deaths.rda")
```

8. (1 point) Repeat this now for vaccination data. Keep the variables `series_complete` and `booster` along with state and date. Display the result and comment appropriately. Hint: only use the rows with `date_type == 'Admin'` to only include vaccine data based on the day it was administered, rather than reported.

```
library(dplyr)
library(lubridate)

vax_clean <- vax_raw %>%
  filter(date_type == "Admin") %>%  # Keep only rows where vaccine data is based on administ
  rename(state = location, series_complete = series_complete_cumulative, booster = booster_cu
  mutate(
    date = as_date(date),
    series_complete = as.numeric(series_complete),
    booster = as.numeric(booster)
  ) %>%
  select(state, date, series_complete, booster) %>%
  arrange(state, date)

print(head(vax_clean))
```

```
  state       date series_complete booster
1    AK 2020-12-13              30       0
2    AK 2020-12-14              30       0
3    AK 2020-12-15              31       0
4    AK 2020-12-16              34       0
5    AK 2020-12-17              41       0
6    AK 2020-12-18              45       0
```

9. (1 point) Now we are ready to join the tables. We will only consider 2020 and 2021 as we don't have population sizes for 2022. However, because we want to guarantee that all dates are included we will create a data frame with all possible weeks. We can use this:

```r
# Load necessary libraries
library(dplyr)
library(lubridate)
library(tidyr)

# Ensure all states use two-letter abbreviations
state_lookup <- data.frame(
  state_full = state.name,
  state_abbr = state.abb
) %>%
  bind_rows(data.frame(state_full = "District of Columbia", state_abbr = "DC"))

convert_state <- function(state_col) {
  ifelse(state_col %in% state.abb, state_col, state_lookup$state_abbr[match(state_col, state
}

# Convert all datasets to use state abbreviations and ensure year/week are numeric
population <- population %>%
  mutate(state = convert_state(state),
         year = as.numeric(year)) %>%
  distinct(state, year, .keep_all = TRUE)

cases_weekly <- cases_weekly %>%
  mutate(state = convert_state(state),
         mmwr_year = as.numeric(mmwr_year),
         mmwr_week = as.numeric(mmwr_week)) %>%
  distinct(state, mmwr_year, mmwr_week, .keep_all = TRUE)

hosp_weekly <- rename(hosp_weekly, state = jurisdiction) %>%
  mutate(state = convert_state(state),
         mmwr_year = as.numeric(mmwr_year),
         mmwr_week = as.numeric(mmwr_week)) %>%
  distinct(state, mmwr_year, mmwr_week, .keep_all = TRUE)

deaths <- deaths %>%
  mutate(state = convert_state(state),
         mmwr_year = as.numeric(mmwr_year),
         mmwr_week = as.numeric(mmwr_week)) %>%
  distinct(state, mmwr_year, mmwr_week, .keep_all = TRUE)

vax_clean <- vax_clean %>%
  mutate(state = convert_state(state), date = as_date(date)) %>%
```

```r
  distinct(state, date, .keep_all = TRUE)

# Create a data frame with all possible weeks in 2020 and 2021
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))

# Ensure all states are represented for each week in 2020 and 2021
dates_and_pop <- crossing(all_dates, data.frame(state = unique(population$state))) %>%
  left_join(population, by = c("state", "mmwr_year" = "year")) %>%
  distinct(state, mmwr_year, mmwr_week, .keep_all = TRUE)  # Remove duplicates

# Merge all datasets while ensuring correct state and date mappings
dat <- dates_and_pop %>%
  left_join(cases_weekly, by = c("state", "mmwr_year", "mmwr_week")) %>%
  left_join(hosp_weekly, by = c("state", "mmwr_year", "mmwr_week")) %>%
  left_join(deaths, by = c("state", "mmwr_year", "mmwr_week")) %>%
  left_join(vax_clean, by = c("state", "date")) %>%
  arrange(state, date)

# Display first few rows of the final dataset
print(head(dat))
```

```
# A tibble: 6 x 11
  date       mmwr_year mmwr_week state state_name population total_cases
  <date>         <dbl>     <dbl> <chr> <chr>           <dbl>       <dbl>
1 2020-01-25      2020         4 AK    Alaska         732441           0
2 2020-02-01      2020         5 AK    Alaska         732441           0
3 2020-02-08      2020         6 AK    Alaska         732441           0
4 2020-02-15      2020         7 AK    Alaska         732441           0
5 2020-02-22      2020         8 AK    Alaska         732441           0
6 2020-02-29      2020         9 AK    Alaska         732441           0
# i 4 more variables: total_hospitalizations <dbl>, deaths <dbl>,
#   series_complete <dbl>, booster <dbl>
```

Now join all the tables to create your final table. Make sure it is ordered by date within each state. Call it `dat`. Show a few rows here.

Data visualization: generate some plots

We are now ready to create some figures. For each question below, write code that generates a plot that addresses the question.

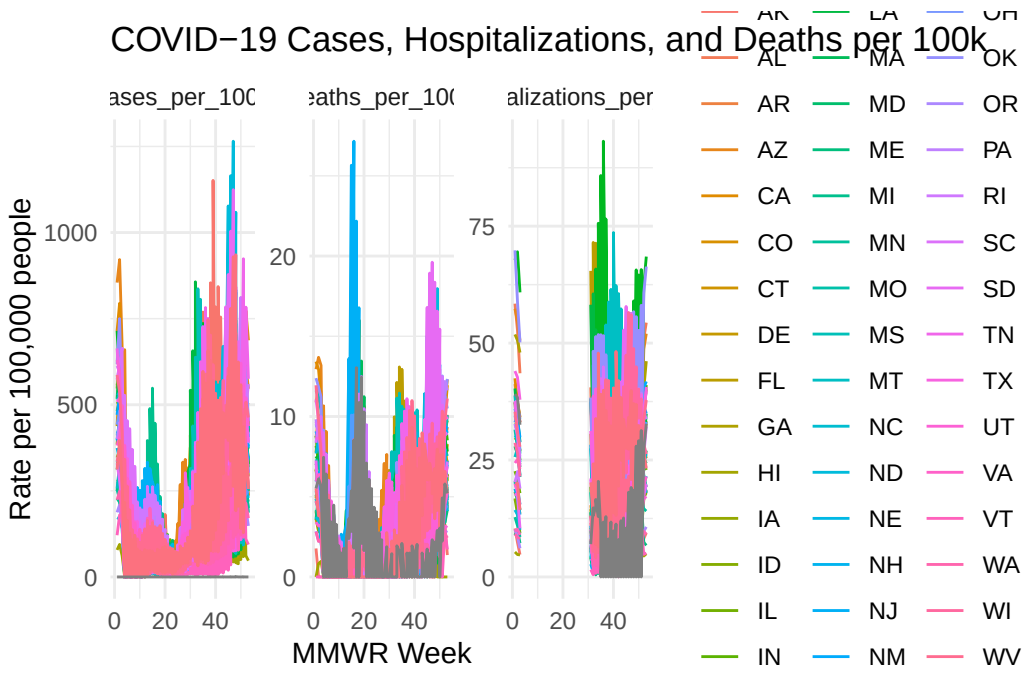
10. (1 point) Plot a trend plot for cases, hospitalizations and deaths for each state. Color by region. Plot rates per 100,000 people. Place the plots on top of each other. Hint: Use `pivot_longer` and `facet_wrap`.

```
library(ggplot2)
library(dplyr)
library(tidyr)


print(head(dat))
```

```
# A tibble: 6 x 11
  date       mmwr_year mmwr_week state state_name population total_cases
  <date>         <dbl>     <dbl> <chr> <chr>           <dbl>       <dbl>
1 2020-01-25      2020         4 AK    Alaska         732441           0
2 2020-02-01      2020         5 AK    Alaska         732441           0
3 2020-02-08      2020         6 AK    Alaska         732441           0
4 2020-02-15      2020         7 AK    Alaska         732441           0
5 2020-02-22      2020         8 AK    Alaska         732441           0
6 2020-02-29      2020         9 AK    Alaska         732441           0
# i 4 more variables: total_hospitalizations <dbl>, deaths <dbl>,
#   series_complete <dbl>, booster <dbl>
```

```
dat <- dat %>%
  mutate(
    cases_per_100k = (total_cases / population) * 100000,
    hospitalizations_per_100k = (total_hospitalizations / population) * 100000,
    deaths_per_100k = (deaths / population) * 100000
  )


dat_long <- dat %>%
  select(state, mmwr_year, mmwr_week, cases_per_100k, hospitalizations_per_100k, deaths_per_
  pivot_longer(cols = c(cases_per_100k, hospitalizations_per_100k, deaths_per_100k),
               names_to = "metric", values_to = "rate")
```

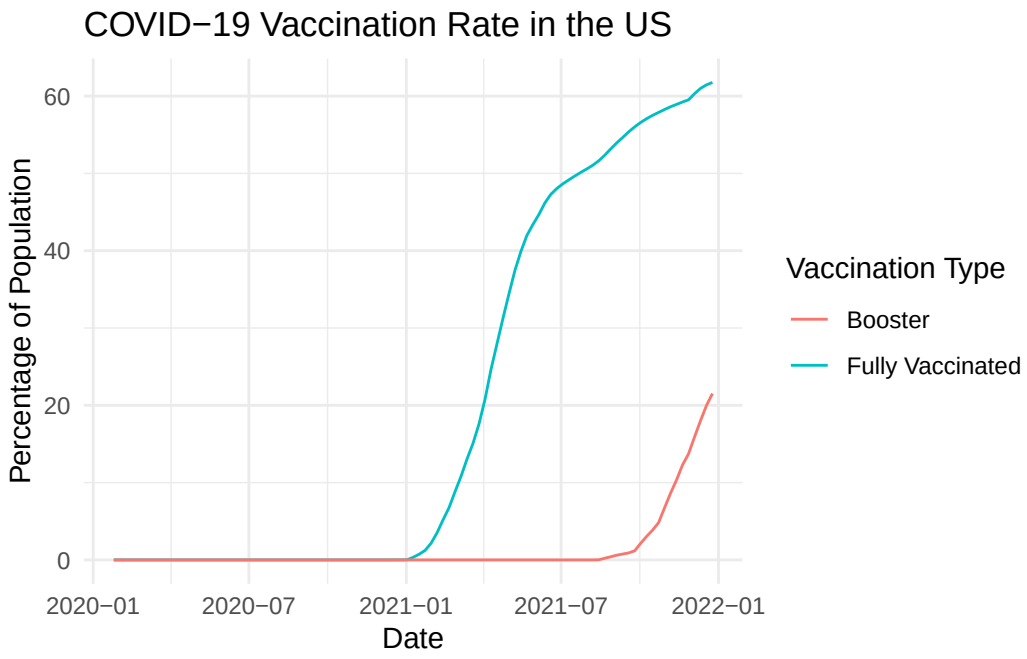
```r
ggplot(dat_long, aes(x = mmwr_week, y = rate, color = state)) +
  geom_line() +
  facet_wrap(~metric, scales = "free_y") +
  labs(title = "COVID-19 Cases, Hospitalizations, and Deaths per 100k",
       x = "MMWR Week",
       y = "Rate per 100,000 people",
       color = "State") +
  theme_minimal()
```



11. (1 point) 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) vaccinated by date. Do the same for the booster. Then plot both percentages.

```r
vaccination_summary <- dat %>%
  group_by(date) %>%
  summarise(
    total_vaccinated = sum(series_complete, na.rm = TRUE),
    total_boosted = sum(booster, na.rm = TRUE),
    total_population = sum(population, na.rm = TRUE)
  ) %>%
  mutate(
    percent_vaccinated = (total_vaccinated / total_population) * 100,
```

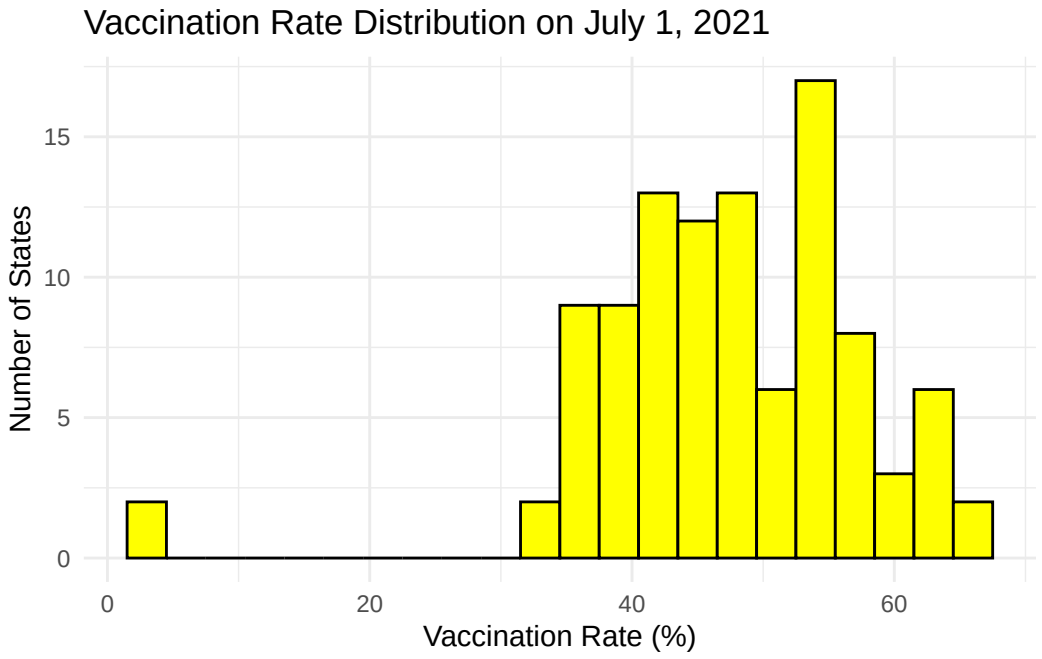
```
    percent_boosted = (total_boosted / total_population) * 100
  )


library(ggplot2)

ggplot(vaccination_summary, aes(x = date)) +
  geom_line(aes(y = percent_vaccinated, color = "Fully Vaccinated")) +
  geom_line(aes(y = percent_boosted, color = "Booster")) +
  labs(
    title = "COVID-19 Vaccination Rate in the US",
    x = "Date",
    y = "Percentage of Population",
    color = "Vaccination Type"
  ) +
  theme_minimal()
```



12. (1 point) Plot the distribution of vaccination rates across states on July 1, 2021.

```
library(ggplot2)
library(dplyr)
```

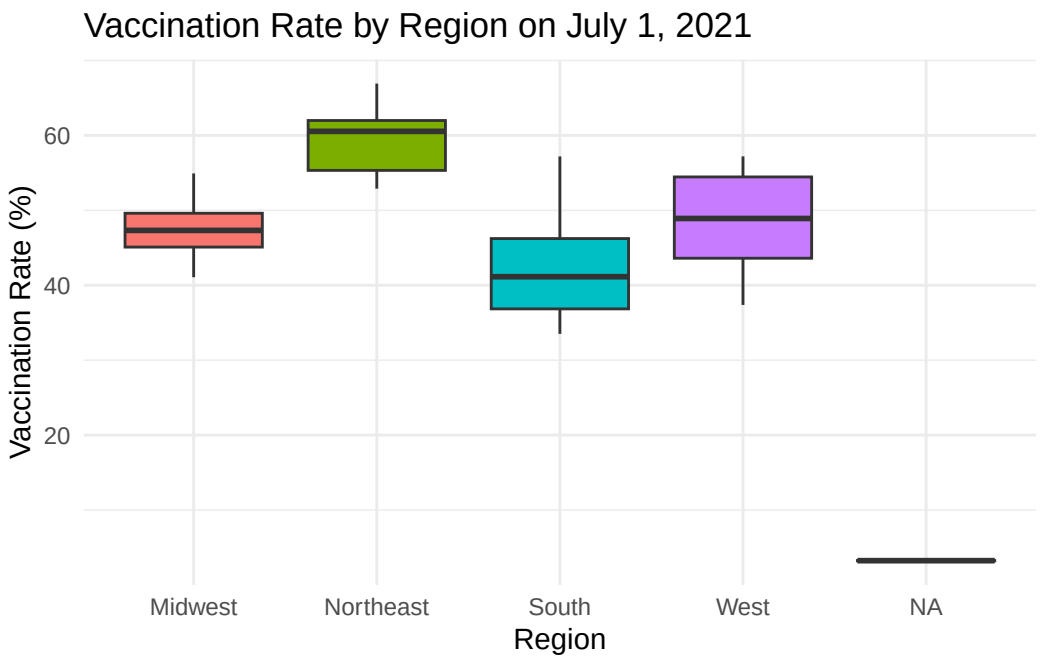
```r
vax_clean <- vax_clean %>%
  mutate(series_complete = as.numeric(series_complete))
population <- population %>%
  mutate(population = as.numeric(population))

vax_july1 <- vax_clean %>%
  filter(date == as.Date("2021-07-01")) %>%
  left_join(population, by = "state") %>%
  mutate(vax_rate = (series_complete / population) * 100)


ggplot(data = vax_july1, mapping = aes(x = vax_rate)) +
  geom_histogram(binwidth = 3, color = "black", fill = "yellow", alpha = 1.0) +
  labs(title = "Vaccination Rate Distribution on July 1, 2021",
       x = "Vaccination Rate (%)",
       y = "Number of States") +
  theme_minimal()
```



13. (1 point) Is there a difference across region? Generate a plot of your choice.

```r
library(ggplot2)
library(dplyr)
```

```r
vax_clean <- vax_clean %>%
  mutate(series_complete = as.numeric(series_complete))

population <- population %>%
  mutate(population = as.numeric(population))

state_region <- data.frame(
  state = c("AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "FL", "GA",
            "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME", "MD",
            "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH", "NJ",
            "NM", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI", "SC",
            "SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI", "WY"),
  region = c("South", "West", "West", "South", "West", "West", "Northeast", "South", "South"
             "West", "West", "Midwest", "Midwest", "Midwest", "Midwest", "South", "South", "
             "Northeast", "Midwest", "Midwest", "South", "Midwest", "West", "Midwest", "West
             "West", "Northeast", "South", "Midwest", "Midwest", "South", "West", "Northeast
             "Midwest", "South", "South", "West", "Northeast", "South", "West", "South", "Mi
)

vax_region <- vax_clean %>%
  filter(date == as.Date("2021-07-01")) %>%
  left_join(population, by = "state") %>%
  left_join(state_region, by = "state") %>%
  mutate(vax_rate = (series_complete / population) * 100)


ggplot(vax_region, aes(x = region, y = vax_rate, fill = region)) +
  geom_boxplot(outlier.color = "red", outlier.shape = 16) +
  labs(title = "Vaccination Rate by Region on July 1, 2021",
       x = "Region",
       y = "Vaccination Rate (%)") +
  theme_minimal() +
  theme(legend.position = "none")
```



Discuss what the plot shows.

The map shows strong geographical variations in immunization rates, with the Northeast and West leading in vaccine uptake, while the South lags behind.The presence of the NA category suggests some missing or unclassified data that should be investigated further.

14. (1 point) Using the previous figures, identify a time period that meets the following criteria:

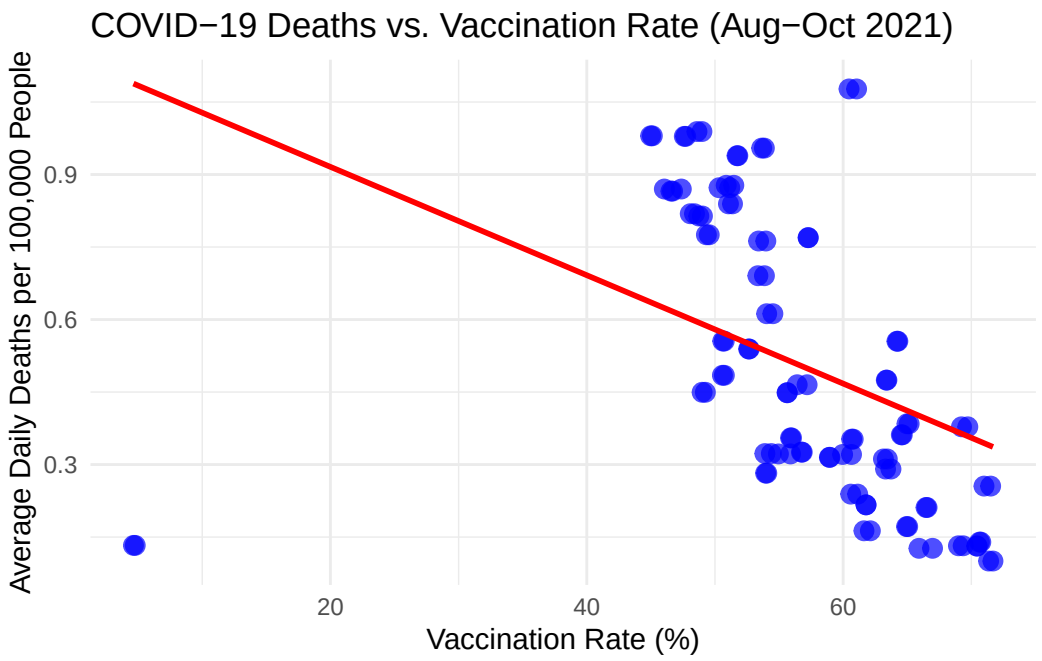
- A significant COVID-19 wave occurred across the United States.
- A sufficient number of people had been vaccinated.

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.

```r
library(ggplot2)
library(dplyr)

# Define the analysis period (Delta wave: August 1 - October 31, 2021)
analysis_period <- dat %>%
  filter(date >= as.Date("2021-08-01") & date <= as.Date("2021-10-31"))

# Calculate average daily deaths per 100,000 people
death_stats <- analysis_period %>%
  group_by(state) %>%
  summarise(
    daily_deaths_per_100k = mean(deaths / population * 100000 / 7, na.rm = TRUE)  # Convert w
  )

# Extract the vaccination rate on the last day of the period (October 31, 2021)
vaccination_summary <- vax_clean %>%
  filter(date == as.Date("2021-10-31")) %>%
  select(state, series_complete) %>%
  left_join(population, by = "state") %>%
  mutate(vaccination_rate = (series_complete / population) * 100) %>%
  select(state, vaccination_rate)

# Merge datasets: deaths and vaccination rates
merged_data <- death_stats %>%
  left_join(vaccination_summary, by = "state")

# Scatter plot: Vaccination Rate vs. Daily Deaths per 100,000
ggplot(merged_data, aes(x = vaccination_rate, y = daily_deaths_per_100k)) +
  geom_point(size = 3, color = "blue", alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +  # Add linear regression trend line
  labs(
    title = "COVID-19 Deaths vs. Vaccination Rate (Aug-Oct 2021)",
    x = "Vaccination Rate (%)",
    y = "Average Daily Deaths per 100,000 People"
  ) +
  theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
```



15. (1 point) Repeat the exercise for the booster.

```
library(ggplot2)
library(dplyr)

# Define the analysis period (Delta wave: August 1 - October 31, 2021)
analysis_period <- dat %>%
  filter(date >= as.Date("2021-08-01") & date <= as.Date("2021-10-31"))

# Calculate average daily deaths per 100,000 people
death_stats <- analysis_period %>%
  group_by(state) %>%
  summarise(
    daily_deaths_per_100k = mean(deaths / population * 100000 / 7, na.rm = TRUE)  # Convert w
  )

# Extract the booster vaccination rate on the last day of the period (October 31, 2021)
booster_summary <- vax_clean %>%
  filter(date == as.Date("2021-10-31")) %>%
  select(state, booster) %>%
  left_join(population, by = "state") %>%
  mutate(booster_rate = (booster / population) * 100) %>%
  select(state, booster_rate)
```

```r
# Merge datasets: deaths and booster vaccination rates
merged_booster_data <- death_stats %>%
  left_join(booster_summary, by = "state")

# Scatter plot: Booster Vaccination Rate vs. Daily Deaths per 100,000
ggplot(merged_booster_data, aes(x = booster_rate, y = daily_deaths_per_100k)) +
  geom_point(size = 3, color = "purple", alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "darkorange") +  # Add linear regression tre
  labs(
    title = "COVID-19 Deaths vs. Booster Vaccination Rate (Aug-Oct 2021)",
    x = "Booster Vaccination Rate (%)",
    y = "Average Daily Deaths per 100,000 People"
  ) +
  theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
```