Problem Set 4

Brynn Woolley

Problem 1 - Tidyverse: New Zealand

Use the tidyverse for this problem. In particular, use piping and dplyr as much as you are able. Note: Use of any deprecated functions will result in a point loss.

Install and load the package nzelect. This package contains a number of data tables with information about election results in New Zealand. Use the `nzge` data for this problem.

Note that there are two types of votes cast - a Candidate vote, and a Party vote. Be sure to handle these separately as appropriate. Don't forget that `help()` works on data as well a functions (assuming the data is appropriately documented).

a. Generate a table (which can just be a nicely printed tibble) of vote count (regardless of party) per year/type. Make sure to sort it by vote count.

```r
# load dataset
data("nzge", package = "nzelect")

# summarize total votes per election year and voting type
vote_summary <- nzge %>%
  group_by(election_year, voting_type) %>%
  summarize(total_votes = sum(votes, na.rm = TRUE), .groups = "drop") %>%
  arrange(desc(total_votes))

# print nicely
vote_summary %>%
  knitr::kable(col.names = c("Election Year", "Voting Type", "Total Votes"),
               caption = "Total Votes by Year and Type")
```

Table 1: Total Votes by Year and Type

Election Year	Voting Type	Total Votes
2014	Party	2416479
2014	Candidate	2375493
2008	Party	2356536
2008	Candidate	2325598
2005	Party	2286190
2005	Candidate	2260670
2011	Party	2257336
2011	Candidate	2225766
2002	Party	2040248
2002	Candidate	2022115

b. Focus only on the 2014 election. Report the proportion of votes for each party in the Candidate election. Again, produce a nice table and sort it by percent of vote.

```r
# filter to 2014 election & compute percentages
nzge_2014 <- nzge %>%
  filter(election_year == 2014, voting_type == "Candidate") %>%
  group_by(party) %>%
  summarise(total_votes = sum(votes, na.rm = TRUE), .groups = "drop") %>%
  mutate(percent_of_vote = 100 * total_votes / sum(total_votes)) %>%
  arrange(desc(percent_of_vote))

# print nicely
nzge_2014 %>%
  knitr::kable(
col.names = c("Party", "Total Votes", "Percent of Vote"),
digits = 2,
caption = "2014 Candidate Votes by Party"
  )
```

Table 2: 2014 Candidate Votes by Party

Party	Total Votes	Percent of Vote
National Party	1081787	45.54
Labour Party	801287	33.73
Green Party	165718	6.98
Conservative Party	81075	3.41
New Zealand First Party	73384	3.09
Maori Party	42108	1.77

Party	Total Votes	Percent of Vote
MANA Movement	32333	1.36
Informal Candidate Votes	27886	1.17
ACT New Zealand	27778	1.17
United Future	14722	0.62
Aotearoa Legalise Cannabis Party	4936	0.21
Internet Party	4848	0.20
Democrats for Social Credit	4647	0.20
Ban1080	4448	0.19
Independent	3864	0.16
NZ Independent Coalition	1929	0.08
Focus New Zealand	1797	0.08
Money Free Party	391	0.02
Communist League	135	0.01
Climate Party	116	0.00
Human Rights Party	76	0.00
The Expats	70	0.00
Alliance	59	0.00
NZ Economic Euthenics Party	51	0.00
Patriotic Revolutionary Front	48	0.00

c. Produce a nice table indicating, for each year, which party won the Candidate vote and which party won the Party vote.

```r
# find winning party for candidate vote & party votes per year
winners_by_type <- nzge %>%
  group_by(election_year, voting_type, party) %>%
  summarize(total_votes = sum(votes, na.rm=TRUE), .groups="drop_last") %>%
  slice_max(order_by=total_votes, n=1, with_ties=FALSE) %>%
  ungroup()

# reshape so our candidate & party winners are in the same row
winners <- winners_by_type %>%
  select(election_year, voting_type, party) %>%
  pivot_wider(
    names_from = voting_type,
    values_from = party,
    names_prefix = "Winner_"
  ) %>%
  arrange(election_year)
```

```r
# print nicely
winners %>%
  knitr::kable(
    col.names = c("Election Year", "Candidate Vote Winner", "Party Vote Winner"),
    caption = "Winning Party by Election Year and Vote Type"
  )
```

Table 3: Winning Party by Election Year and Vote Type

Election Year	Candidate Vote Winner	Party Vote Winner
2002	Labour Party	Labour Party
2005	National Party	Labour Party
2008	National Party	National Party
2011	National Party	National Party
2014	National Party	National Party

Problem 2 - Tidyverse: Tennis

Use the tidyverse for this problem. In particular, use piping and `dplyr` as much as you are able. Note: Use of any deprecated functions will result in a point loss.

Use the "ATP Matches" data from 2019 available at https://raw.githubusercontent.com/JeffSackmann/tennis_atp/refs/heads/master/atp_matches_2019.csv.

This data tracks all Tennis matches. This data does not have documentation, so you'll have to explore the data yourself to figure out it's structure. Use it to answer the following questions. Your answers should show both the output from R that allows you to answer it, as well as a written answer.

a. How many tournaments took place in 2019?

```r
# read in data
matches_2019 <- read_csv(
  "https://raw.githubusercontent.com/JeffSackmann/tennis_atp/refs/heads/master/atp_matches_20
)

# count unique tournaments
tournament_count <- matches_2019 %>%
  distinct(tourney_id, tourney_name) %>%
  count(name="n_tournaments")
```

```r
# print nicely
tournament_count %>%
  knitr::kable(
    col.names = c("Number of Tournaments"),
    caption = "Count of ATP Tournaments in 2019"
  )
```

Table 4: Count of ATP Tournaments in 2019

Number of Tournaments
128

b. Did any player win more than one tournament? If so, how many players won more than one tournament, and how many tournaments did the most winning player(s) win?

```r
# find the winner of each tournament
tournament_winners <- matches_2019 %>%
  filter(round=="F") %>%
  select(tourney_id, tourney_name, winner_name)

# count number of tournaments won per player
player_wins <- tournament_winners %>%
  count(winner_name, name="tournaments_won") %>%
  arrange(desc(tournaments_won))

# players with more than 1 win
multi_winners <- player_wins %>%
  filter(tournaments_won>1)

# print nicely
multi_winners %>%
  knitr::kable(
    col.names = c("Player", "Tournaments Won"),
    caption = "Players with more than 1 win in 2019"
  )
```

Table 5: Players with more than 1 win in 2019

Player	Tournaments Won
Dominic Thiem	5

Player	Tournaments Won
Novak Djokovic	5
Daniil Medvedev	4
Rafael Nadal	4
Roger Federer	4
Alex De Minaur	3
Stefanos Tsitsipas	3
Benoit Paire	2
Cristian Garin	2
Jo-Wilfried Tsonga	2
Matteo Berrettini	2
Nick Kyrgios	2

c. Is there any evidence that winners have more aces than losers? (If you address this with a hypothesis test, do not use base R functionality - continue to remain in the Tidyverse.)

```r
# reshape so winners & losers are in one column
aces_long <- matches_2019 %>%
  select(winner_name, loser_name, w_ace, l_ace) %>%
  pivot_longer(
    cols = c(w_ace, l_ace),
    names_to = "result_type",
    values_to = "aces"
  ) %>%
  mutate(
    result = if_else(result_type == "w_ace", "Winner", "Loser")
  ) %>%
  drop_na(aces)

# summarize aces by result type
aces_summary <- aces_long %>%
  group_by(result) %>%
  summarize(
    mean_aces = mean(aces, na.rm=TRUE),
    median_aces = median(aces, na.rm=TRUE),
    sd.aces = sd(aces, na.rm=TRUE),
    n = n()
  )
aces_summary %>%
  knitr::kable(caption="Summary of Aces by Match Result")
```

Table 6: Summary of Aces by Match Result

result	mean_aces	median_aces	sd.aces	n
Loser	5.792502	4	5.631426	2694
Winner	7.497402	6	6.065966	2694

```r
# observe t statistic
obs_t <- aces_long %>%
  specify(aces ~ result) %>%
  calculate(stat="t")

# permuate distribution
null_dist <- aces_long %>%
  specify(aces ~ result) %>%
  hypothesize(null="independence") %>%
  generate(reps=10000, type="permute") %>%
  calculate(stat="t")

# get p value
p_val <- null_dist %>%
  get_p_value(obs_stat=obs_t, direction="two_sided")

print(paste0("P-value: ", p_val$p_value))
```

```
[1] "P-value: 0"
```

A permutation test (10 000 reps) comparing the number of aces by winners vs. losers yielded p < 0.001, indicating statistically significant evidence that winners hit more aces on average.

d. Identify the player(s) with the highest win-rate. (Note that this is NOT asking for the highest number of wins.) Restrict to players with at least 5 matches.

```r
# create long form of matches df with one row per player per match
player_results <- matches_2019 %>%
  select(winner_name, loser_name) %>%
  pivot_longer(
    cols = c(winner_name, loser_name),
    names_to = "result_type",
    values_to = "player"
  ) %>%
  mutate(result = if_else(result_type == "winner_name", "Win", "Loss"))
```

```r
# compute total matches & wins per player
win_rates <- player_results %>%
  group_by(player) %>%
  summarize(
    matches_played = n(),
    wins = sum(result == "Win"),
    win_rate = wins / matches_played
  ) %>%
  filter(matches_played >= 5) %>%
  arrange(desc(win_rate))

top_win_rate <- max(win_rates$win_rate, na.rm=TRUE)

best_players <- win_rates %>%
  filter(win_rate == top_win_rate)

best_players %>%
  knitr::kable(
    col.names = c("Player", "Matches Played", "Wins", "Win Rate"),
    digits = 3,
    caption = "Player(s) with the Highest Win Rate"
  )
```

Table 7: Player(s) with the Highest Win Rate

Player	Matches Played	Wins	Win Rate
Rafael Nadal	69	60	0.87

Problem 3 - Visualization

Note: This is, intentionally, a very open-ended question. There is no “right” answer. The goal is for you to explore your plotting options, and settle on something reasonable. You can use base R, ggplot, or something else. You’ll likely have to look online for resources on plotting beyond what we covered in class.

Use the NYTimes Covid data (https://raw.githubusercontent.com/nytimes/covid-19-data/refs/heads/master/rolling-averages/us-states.csv).

This lists daily Covid new cases. For each of the following, produce a publication-ready plot which addresses the question. Use your plot to support an argument for your question.

You will be graded on:
1. Is the type of graph & choice of variables appropriate to answer the question?
2. Is the graph clear and easy to interpret?
3. Is the graph publication ready?

   a. How many major and minor spikes in cases were there?

```
# load data
covid_states <- read_csv(
  "https://raw.githubusercontent.com/nytimes/covid-19-data/refs/heads/master/rolling-average
)


# basic structure
glimpse(covid_states)
```

```
Rows: 61,942
Columns: 9
$ date               <date> 2020-01-21, 2020-01-22, 2020-01-23, 2020-01-24, 2~
$ geoid              <chr> "USA-53", "USA-53", "USA-53", "USA-53", "USA-17", ~
$ state              <chr> "Washington", "Washington", "Washington", "Washing~
$ cases              <dbl> 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,~
$ cases_avg          <dbl> 0.14, 0.14, 0.14, 0.14, 0.14, 0.14, 0.14, 0.14, 0.~
$ cases_avg_per_100k <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ deaths             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ deaths_avg         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ deaths_avg_per_100k <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
```

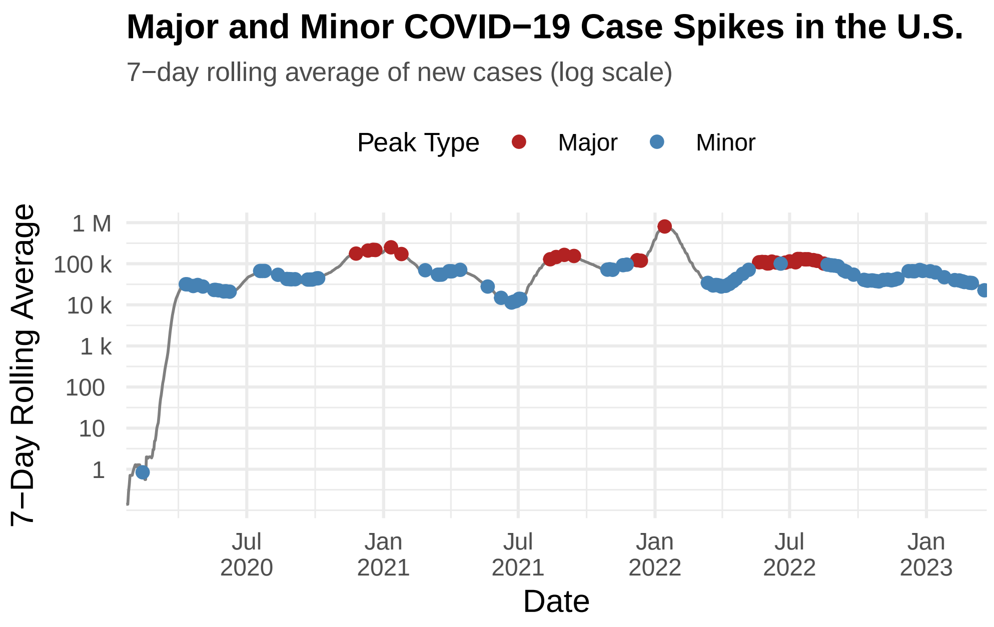
```
# preview first few rows
head(covid_states)
```

```
# A tibble: 6 x 9
  date       geoid  state   cases cases_avg cases_avg_per_100k deaths deaths_avg
  <date>     <chr>  <chr>   <dbl>     <dbl>              <dbl>  <dbl>      <dbl>
1 2020-01-21 USA-53 Washin~     1      0.14                  0      0          0
2 2020-01-22 USA-53 Washin~     0      0.14                  0      0          0
3 2020-01-23 USA-53 Washin~     0      0.14                  0      0          0
4 2020-01-24 USA-53 Washin~     0      0.14                  0      0          0
5 2020-01-24 USA-17 Illino~     1      0.14                  0      0          0
6 2020-01-25 USA-53 Washin~     0      0.14                  0      0          0
# i 1 more variable: deaths_avg_per_100k <dbl>
```

```r
# aggregate by date
covid_us <- covid_states %>%
  group_by(date) %>%
  summarise(cases_avg = sum(cases_avg, na.rm = TRUE))

# look for peaks
covid_us <- covid_us %>%
  arrange(date) %>%
  mutate(
    prev_avg = lag(cases_avg),
    next_avg = lead(cases_avg),
    is_peak = cases_avg > prev_avg & cases_avg > next_avg
  )

# extract peaks and classify by relative size
peaks <- covid_us %>%
  filter(is_peak) %>%
  mutate(
    peak_type = case_when(
      cases_avg >= quantile(cases_avg, 0.75, na.rm = TRUE) ~ "Major",
      TRUE ~ "Minor"
    )
  ) %>%
  arrange(desc(cases_avg))

peaks %>%
  count(peak_type)
```

```
# A tibble: 2 x 2
  peak_type     n
  <chr>     <int>
1 Major        31
2 Minor        92
```

```r
top_peaks <- peaks %>%
  filter(peak_type == "Major") %>%
  slice_max(cases_avg, n = 3)


ggplot(covid_us, aes(x = date, y = cases_avg)) +
  geom_line(color = "gray50", linewidth = 0.5) +
  geom_point(data = peaks, aes(color = peak_type), size = 1.8) +

  # simplified y-axis using built-in SI notation (K, M)
  scale_y_log10(
    labels = scales::label_number(scale_cut = scales::cut_si("")),

    breaks = 10 ^ (0:6)
  ) +

  # regular date breaks and clearer labeling
  scale_x_date(
    date_breaks = "6 months",
    date_labels = "%b\n%Y",
    expand = c(0, 0)
  ) +

  scale_color_manual(values = c("Major" = "firebrick", "Minor" = "steelblue")) +
  labs(
    title = "Major and Minor COVID-19 Case Spikes in the U.S.",
    subtitle = "7-day rolling average of new cases (log scale)",
    x = "Date",
    y = "7-Day Rolling Average",
    color = "Peak Type"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(face = "bold", size = 13),
    plot.subtitle = element_text(size = 10, color = "gray30"),
    axis.title.y = element_text(margin = margin(r = 5)),
    axis.text.y = element_text(size = 9),
    axis.text.x = element_text(size = 9),
    legend.position = "top",
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 9),
    plot.margin = margin(t = 15, r = 15, b = 10, l = 15)
  )
```



The figure shows several distinct spikes in U.S. COVID-19 case data between 2020 and 2023. We see major peaks occur in December 2020, January 2021, and January 2022. Smaller waves appeared throughout mid-2021 and late 2022. The log scale highlights both large and moderate increases while preserving proportional differences.

b. For the states with the highest and lowest overall rates per population, what differences do you see in their trajectories over time?

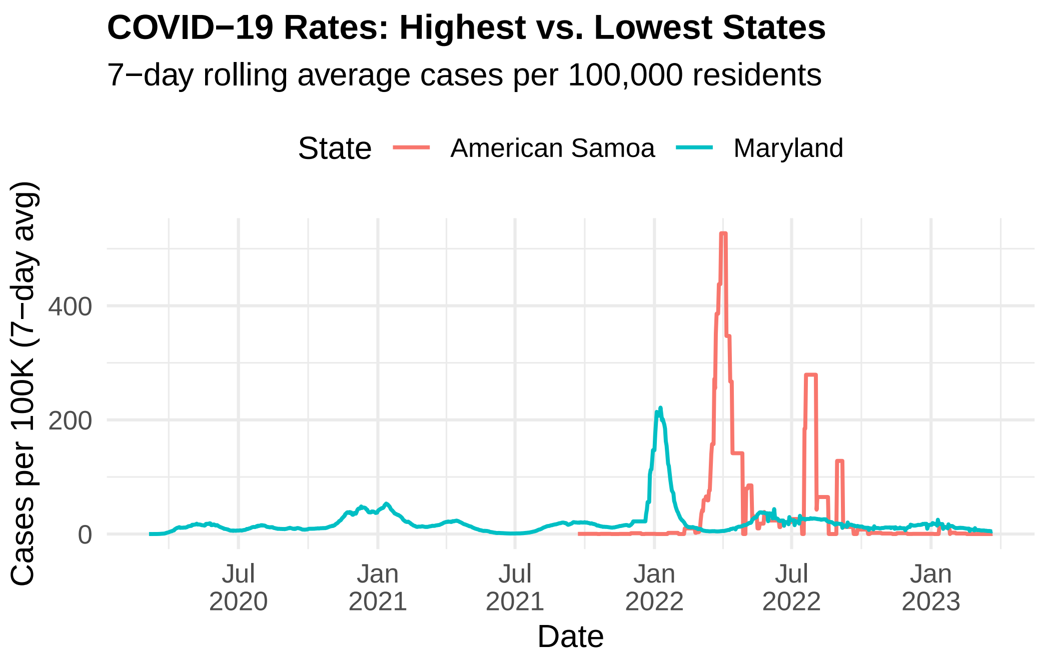
```r
# identify states with highest & lowest rates
state_rates <- covid_states %>%
  group_by(state) %>%
  summarise(
    mean_cases_per_100k = mean(cases_avg_per_100k, na.rm = TRUE)
  ) %>%
  arrange(desc(mean_cases_per_100k))

# extract top and bottom states
top_state <- slice_max(state_rates, mean_cases_per_100k, n = 1)
bottom_state <- slice_min(state_rates, mean_cases_per_100k, n = 1)


top_state$state
```

```
[1] "American Samoa"
```

```r
bottom_state$state
```

```
[1] "Maryland"
```

```r
# plot
covid_states %>%
  filter(state %in% c(top_state$state, bottom_state$state)) %>%
  ggplot(aes(x = date, y = cases_avg_per_100k, color = state)) +
  geom_line(linewidth = 0.7) +
  labs(
    title = "COVID-19 Rates: Highest vs. Lowest States",
    subtitle = "7-day rolling average cases per 100,000 residents",
    x = "Date",
    y = "Cases per 100K (7-day avg)",
    color = "State"
  ) +
  scale_x_date(
    date_breaks = "6 months",
    date_labels = "%b\n%Y"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    legend.position = "top",
    plot.title = element_text(face = "bold", size = 13)
  )
```



The state with the highest rate (American Samoa) shows sharp, isolated peaks concentrated in early 2022, reflecting intense but brief outbreaks.

The state with the lowest rate (Maryland), by contrast, maintained a much flatter trajectory throughout the pandemic, with smaller fluctuations and lower sustained case rates.

This highlights how outbreak intensity and timing varied dramatically even after population adjustment.

c. Identify, to the best of your ability without a formal test, the first five states to experience Covid in a substantial way.

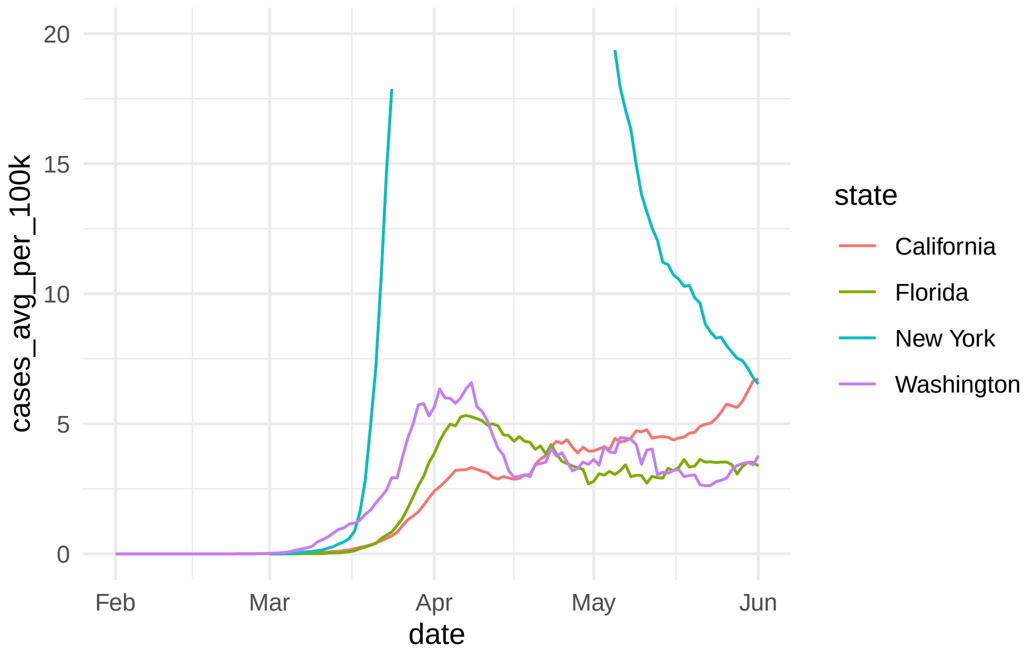
```r
## explore data to inform threshold (c) and consecutive days (k)

# examine distribution of daily rates
summary_stats <- covid_states %>%
  summarise(
    min = min(cases_avg_per_100k, na.rm = TRUE),
    q25 = quantile(cases_avg_per_100k, 0.25, na.rm = TRUE),
    median = median(cases_avg_per_100k, na.rm = TRUE),
    mean = mean(cases_avg_per_100k, na.rm = TRUE),
    q75 = quantile(cases_avg_per_100k, 0.75, na.rm = TRUE),
    p90 = quantile(cases_avg_per_100k, 0.9, na.rm = TRUE),
    p95 = quantile(cases_avg_per_100k, 0.95, na.rm = TRUE),
    max = max(cases_avg_per_100k, na.rm = TRUE)
  )
summary_stats
```

```
# A tibble: 1 x 8
    min   q25 median  mean   q75   p90   p95   max
  <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     0  7.87   15.9  28.0  32.0  60.4  87.8  527.
```

```
# assess persistence of small vs large values
covid_states %>%
  group_by(state) %>%
  mutate(over_1 = cases_avg_per_100k > 1,
         run_id = with(rle(over_1), rep(seq_along(lengths), lengths)),
         run_len = ave(over_1, run_id, FUN = length)) %>%
  ungroup() %>%
  filter(over_1) %>%
  summarise(mean_run = mean(run_len), median_run = median(run_len), max_run = max(run_len))
```

```
# A tibble: 1 x 3
  mean_run median_run max_run
     <dbl>      <dbl>   <int>
1     980.       1092    1103
```

```
# visualize early trajectories
ggplot(covid_states %>%
         filter(state %in% c("Washington", "California", "New York", "Florida")),
       aes(x = date, y = cases_avg_per_100k, color = state)) +
  geom_line() +
  scale_x_date(limits = as.Date(c("2020-02-01","2020-06-01"))) +
  scale_y_continuous(limits = c(0,20)) +
  theme_minimal()
```



```
# quantify baseline noise (pre-March 2020)
baseline_sd <- covid_states %>%
  filter(date < as.Date("2020-03-01")) %>%
  summarise(sd_base = sd(cases_avg_per_100k, na.rm = TRUE))
baseline_sd
```

```
# A tibble: 1 x 1
  sd_base
    <dbl>
1  0.0131
```

Exploring the distribution showed that most values fall below 30 cases per 100k, with a long right tail. The pre-March baseline standard deviation (0.013) indicates almost no activity before community spread began. A threshold of 1 case per 100k is roughly 15× above baseline variation and aligns with visible outbreak onset in early states (e.g., Washington, California). Because outbreaks persist once they start, requiring 3 consecutive days above the threshold filters noise without excluding true signals. So, I define "substantial activity" as `cases_avg_per_100k > 1` for at least 3 consecutive days.

```
# apply rule to identify first states
c <- 1
k <- 3
```

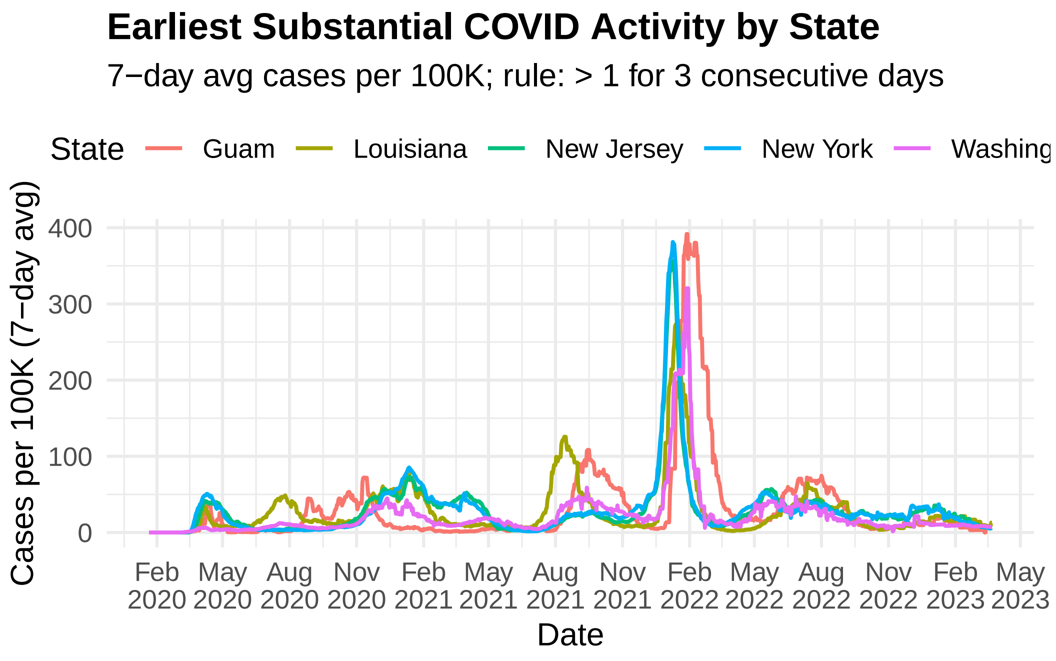
```r
#' Identify first date of sustained COVID activity
#' @param df State-level data frame
#' @param c Threshold for cases_avg_per_100k
#' @param k Minimum consecutive days
#' @return Tibble with first_date

first_sustained <- function(df, c, k) {
  df <- dplyr::arrange(df, date)
  over <- df$cases_avg_per_100k > c
  r <- rle(over)
  run_id <- rep(seq_along(r$lengths), r$lengths)
  run_len <- ave(over, run_id, FUN = length)
  hit <- over & (run_len >= k)
  if (!any(hit, na.rm = TRUE)) return(dplyr::tibble(first_date = as.Date(NA)))
  dplyr::tibble(first_date = min(df$date[hit], na.rm = TRUE))
}

first_dates <- covid_states %>%
  group_by(state) %>%
  group_modify(~ first_sustained(.x, c = c, k = k)) %>%
  ungroup() %>%
  arrange(first_date)

first_five <- slice_head(first_dates, n = 5)

first_five %>%
  knitr::kable(
    col.names = c("State", "First Substantial Case Date"),
    caption = sprintf(
      "First five states by rule: cases_avg_per_100k > %s for %s consecutive days.",
      c, k
    )
  )
```

Table 8: First five states by rule: cases_avg_per_100k > 1 for 3 consecutive days.

State	First Substantial Case Date
Washington	2020-03-16
New York	2020-03-18
Guam	2020-03-19
Louisiana	2020-03-19

State	First Substantial Case Date
New Jersey	2020-03-19

```r
# visualize early trajectories
covid_states %>%
  filter(state %in% first_five$state) %>%
  ggplot(aes(x = date, y = cases_avg_per_100k, color = state)) +
  geom_line(linewidth = 0.7) +
  scale_x_date(date_breaks = "3 months", date_labels = "%b\n%Y") +
  labs(
    title = "Earliest Substantial COVID Activity by State",
    subtitle = sprintf("7-day avg cases per 100K; rule: > %s for %s consecutive days", c, k)
    x = "Date", y = "Cases per 100K (7-day avg)", color = "State"
  ) +
  theme_minimal(base_size = 12) +
  theme(legend.position = "top", plot.title = element_text(face = "bold"))
```



GitHub Link

- Repo: https://github.com/brynnwoolley/STATS-506#
