Vaccine Efficacy

Ron Sielinski

2024-06-05

Introduction

Intuitively, we expect vaccines to reduce a population's rate of viral infection. The analysis described in this paper uses data from the COVID-19 pandemic to explore the relationship between the availability of COVID-19 vaccines and changes in the rates of confirmed cases in the United States (US).

Raw Data

During the pandemic, the Johns Hopkins Coronavirus Resource Center collated data from a variety of authoritative sources. Although the Center ceased collecting and reporting data on March 10, 2023, a static snapshot of the data has been made available by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University on GitHub: https://github.com/CSSEGISandData/COVID-19.

Only the US time series dataset is needed for this report. That dataset contains daily summaries of confirmed cases and deaths, reported at the county level. The US data is contained in two tables, named `time_series_covid19_confirmed_US.csv` and `time_series_covid19_deaths_US.csv`, respectively.

```r
###################
## Load Packages
###################

library(tidyverse)
library(pROC) # used to calculate model performance and generate a ROC curve

###################
## Load the Data
###################

data_url <- str_c('https://raw.githubusercontent.com',
                  '/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/'
)

data_files <- c(
  "time_series_covid19_confirmed_US.csv",
  "time_series_covid19_deaths_US.csv"
)

# import data into separate data frames
confirmed_US <- read_csv(str_c(data_url, data_files[1]))
deaths_US <- read_csv(str_c(data_url, data_files[2]))
```

```r
# clean up the environment
rm(data_url, data_files)
```

Preparing the Data

To simplify analysis, I performed a number of transformations:

- Reshaped the data, pivoting from multiple date columns to a simpler combination of columns, one for date and one for quantity (i.e., cases or deaths)
- Removed unnecessary columns (e.g., `latitude` and `longitude`)
- Converted the `date` column from a string to a date
- Aggregated the daily counts of cases and deaths at the state/territory level
- Reduced the grain from daily to weekly (to reduce period-over-period noise)
- Calculated the number of new cases and deaths per week

```r
##################################
## Tidy and Transform the Data
##################################

# tidy US cases and deaths
US_cases <- confirmed_US |>
  pivot_longer(cols = -c(UID:Combined_Key), names_to = 'date', values_to = 'cases') |>
  # keep Admin2 (secondary geo level) until data is summarized at the state level
  select(Province_State, Admin2, date, cases) |>
  mutate(date = mdy(date))

US_deaths <- deaths_US |>
  pivot_longer(cols = -c(UID:Population), names_to = 'date', values_to = 'deaths') |>
  select(Province_State, Admin2, date, deaths, Population) |>
  mutate(date = mdy(date))

# combine cases and deaths into a single table
US <- US_cases |>
  full_join(US_deaths, by = join_by(Province_State, Admin2, date))

# summarize by date and state/territory
US_by_state <- US |>
  summarize(cases = sum(cases),
            deaths = sum(deaths),
            population = sum(Population),
            .by = c(Province_State, date))

# reduce frequency to weekly
weekly_dat <- US_by_state |>
  mutate(week = floor_date(date, unit = 'week')) |>
  slice_max(date, by = c(Province_State, week)) |>
  # calculate new cases and deaths *after* reducing grain
  group_by(Province_State) |>
  mutate(new_cases = cases - lag(cases),
         new_deaths = deaths - lag(deaths)) |>
  ungroup()
```

```r
# clean up the environment
rm(confirmed_US, deaths_US, US_cases, US_deaths, US, US_by_state)
```

High-level Approach

As suggested in the introduction, an effective vaccine should reduce the rate of viral infection: Before vaccines, we'd expect relatively large numbers of new cases; after vaccines, relatively small numbers of new cases.

To confirm that hypothesis, we'll build an ML model that uses the relative change in case rates to predict whether those results came from a period *before* or *after* the general availability of vaccines. If the hypothesis is correct, the model should do a reasonably good job (better than random) of distinguishing between pre- and post-vaccine time periods.

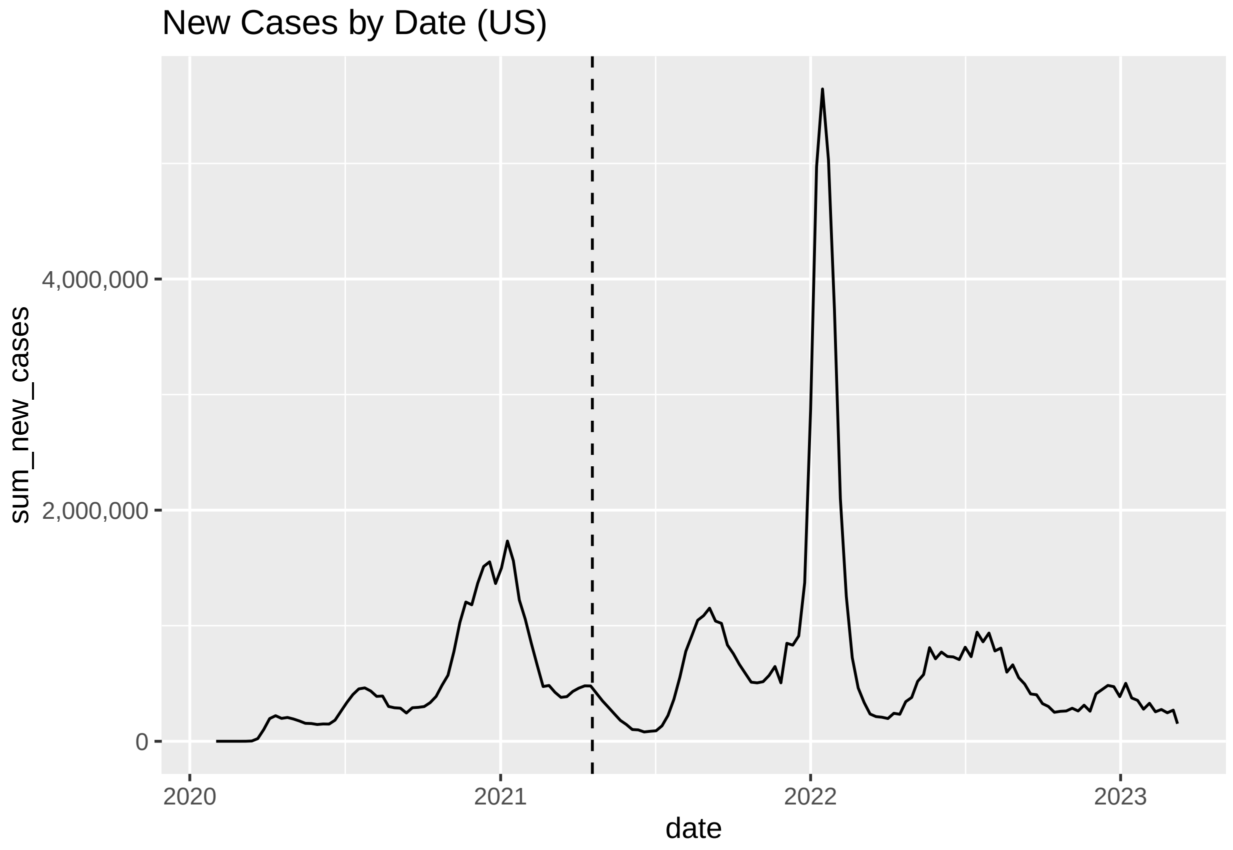
To create the necessary perspective, we first need to establish a specific date for the availability of vaccines. That's complicated by the fact that multiple vaccines were developed: Pfizer-Biontech, Moderna, and others. Choosing a specific date is further complicated by the fact that the initial availability of vaccines was limited because pharmaceutical companies needed time to ramp up their production. Related to that, administration of the vaccines was phased out over several months to ensure that the most vulnerable groups had the opportunity to get inoculated before the rest of the population.

For purposes of this analysis, we will use the date when COVID-19 was deemed generally available to the US population. According to *The New York Times* https://www.nytimes.com/2021/04/19/world/adults-eligible-covid-vaccine.html, on April 19, 2021, "all adults in every U.S. state, Washington, D.C., and Puerto Rico [were] now eligible for a Covid-19 vaccine, meeting the . . . deadline that President Biden set two weeks ago."

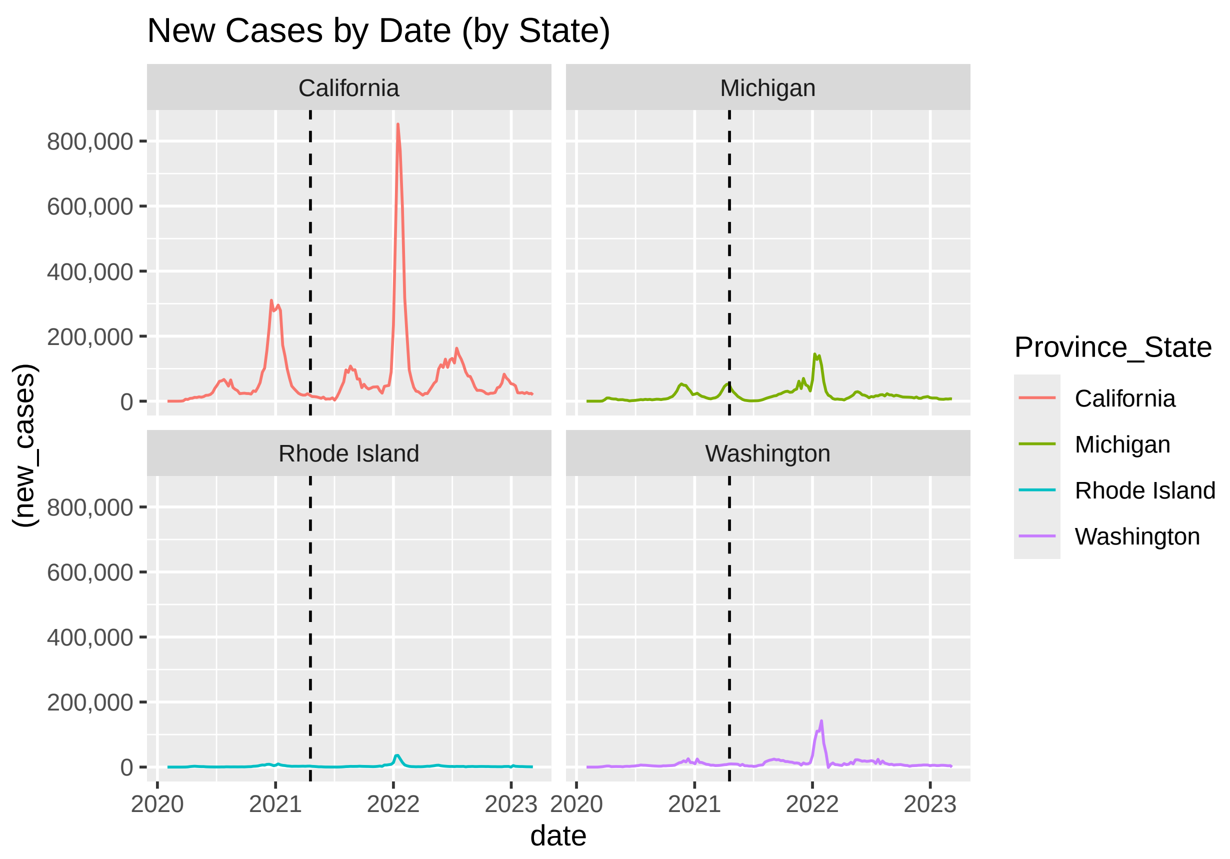
```r
######################
## Augment the Data
######################

# establish vaccine date
vaccine_date <- mdy('04-19-2021')

# identify geographies not included in 'every U.S. state, Washington, D.C., and Puerto Rico'
excluded_geos <- c(
  'American Samoa',
  'Diamond Princess',
  'Grand Princess',
  'Guam',
  'Northern Mariana Islands',
  'Virgin Islands'
)

# remove excluded geographies
weekly_dat <- weekly_dat %>%
  filter(!Province_State %in% excluded_geos)

# identify dates that followed the vaccine date
weekly_dat <- weekly_dat |>
    mutate(post_vaccine = week > vaccine_date)

# clean up the environment
rm(excluded_geos)
```

Exploring the Data

Plotting of the number of new cases by date, however, reveals that the most obvious inflection point in the long-term trend was in the first half of 2022, nearly a full year after the general availability of vaccines (indicated by the dashed vertical line).



A look at the same plot for a sample of individual geographies reveals similar trends.



This new perspective draws attention to the fact that our model will need to accommodate for differences in population sizes. (A decrease of 10,000 cases would represent a much larger change for Rhode Island than California.) To ensure that the changes for each state are comparable, we need to put them on the same relative scale.

We could calculate the change in new cases as a percentage, but the resulting increases and decreases would not be symmetric. If you increase a number by 50% then decrease the result by 50%, you don't end up with the original number: You end up with 3/4 of the original number. Similarly, increases can exceed 100%, but decreases cannot.

A more mathematically robust solution is log returns. By taking the log of the period-over-period percentage change, we overcome the limitations of raw percentages.

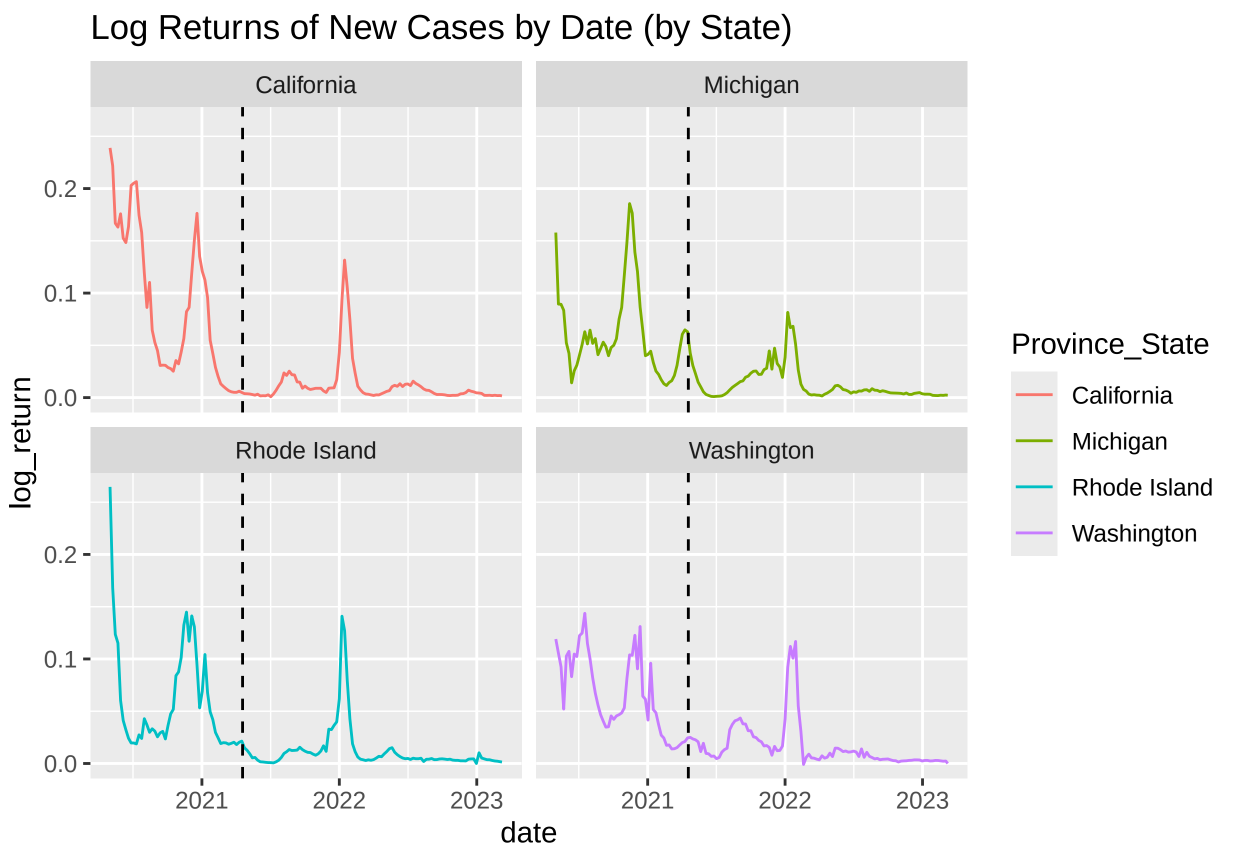
```
###########################
## Calculate Log Returns
###########################

# establish the first week that a state reported cases (in a new data frame)
first_cases <- weekly_dat |>
  filter(cases > 0) |>
  slice_min(n = 1, by = Province_State, order_by = week) |>
  select(Province_State, week, cases) |>
  rename(first_week = week,
         first_cases_cnt = cases)

# starting from the week of a state's first reported case, calculate the log
# return of the increase in cases
```

```r
weekly_dat <- weekly_dat |>
  inner_join(first_cases, by = 'Province_State') |>
  filter(week >= first_week) |>
  group_by(Province_State, first_week) |>
  mutate(log_return = log(cases / lag(cases))) |>
  ungroup() |>
  # get rid of first week (which will have log_return == NA)
  filter(week > first_week)
```

A plot of the log returns (with the first few months of the pandemic removed) illustrates that using log returns serves its intended purpose:

- The y-axis values no longer show the number of new cases but the *relative change* in new cases
- The y-axis values for each geography are now on a comparable scale

More importantly, we can now see a difference in the y-axis values before and after the vaccine. After the vaccine, the y-axis values are generally smaller in value (often near-zero), and the trend lines are generally smoother (with the notable exception of a spike near the start of 2022).



Both visual indicators suggest that log returns capture a useful relationship between the change in case rates and the general availability of vaccines. For our model, we'll use the value itself to capture the high and low differences between the pre- and post-vaccine time periods. Similarly, we will use a sliding window of seven weeks (the current week ± 3 weeks) to capture the difference in trends between the two periods.

```
############################
## Create Sliding Window
############################

# for each week, establish a sliding window of ±3 weeks, which we'll use to
# create context, reducing the likelihood that a one-off increase or
# decrease will get mistaken for a longer term trend
weekly_dat <- weekly_dat |>
  group_by(Province_State, first_week) |>
  mutate(lag_3 = lag(log_return, 3),
         lag_2 = lag(log_return, 2),
         lag_1 = lag(log_return, 1),
         lead_1 = lead(log_return, 1),
         lead_2 = lead(log_return, 2),
         lead_3 = lead(log_return, 3),
  ) |>
  ungroup()

# filter out weeks with missing lag or lead data. the missing data is a
# potential source of data leakage: missing lag values reveals that the data is
# from early in the pandemic, and missing lead values reveal that the data is
# from late in the pandemic
weekly_dat <- weekly_dat |>
  filter(!if_any(c(lag_3, lag_2, lag_1, lead_1, lead_2, lead_3), is.na))
```

Training the Model

To train the model, we need to separate our data into subsets for training and testing. We also need to ensure that the training set is equally balanced between the "before" and "after" periods. (If there are more cases of one than the other, our model might simply learn a preference for the predominant class.)

```
######################################
## Separate Train and Test Subsets
######################################

# establish a rough percentage size for the training dataset
pct <- 0.8

# ensure balance in the training data
min_class_size <- min(table(weekly_dat$post_vaccine))

# ensure balance in the training data by taking an equal number of T and F
# start by finding the size of smallest class (T or F)
min_class_size <- min(table(weekly_dat$post_vaccine))

# take a random sample of that many records from each class
set.seed(99)
idx_T <- sample(which(weekly_dat$post_vaccine == T), size = pct * min_class_size)
idx_F <- sample(which(weekly_dat$post_vaccine == F), size = pct * min_class_size)

# combine the samples from each class into a single vector
train_idx <- c(idx_T, idx_F)
```

```r
# use all remaining rows for test
n <- nrow(weekly_dat)
test_idx <- which(!(1:n %in% train_idx))

# need to ensure the test dataset is balanced, too
min_test_size <- table(weekly_dat$post_vaccine[test_idx]) |>
  min()

test_idx <- data.frame(test_idx = test_idx,
                       post_vaccine = weekly_dat$post_vaccine[test_idx]) |>
  slice_sample(n = min_test_size, by = post_vaccine) |>
  select(test_idx) |>
  unlist()

# actual size of the training set
# length(train_idx) / n

# clean up the environment
rm(pct, min_class_size, idx_T, idx_F, n)

# ensure training and test datasets are balanced
table(weekly_dat$post_vaccine[train_idx])
```

```
##
## FALSE  TRUE
##  2330  2330
```

```r
table(weekly_dat$post_vaccine[test_idx])
```

```
##
## FALSE  TRUE
##   583   583
```

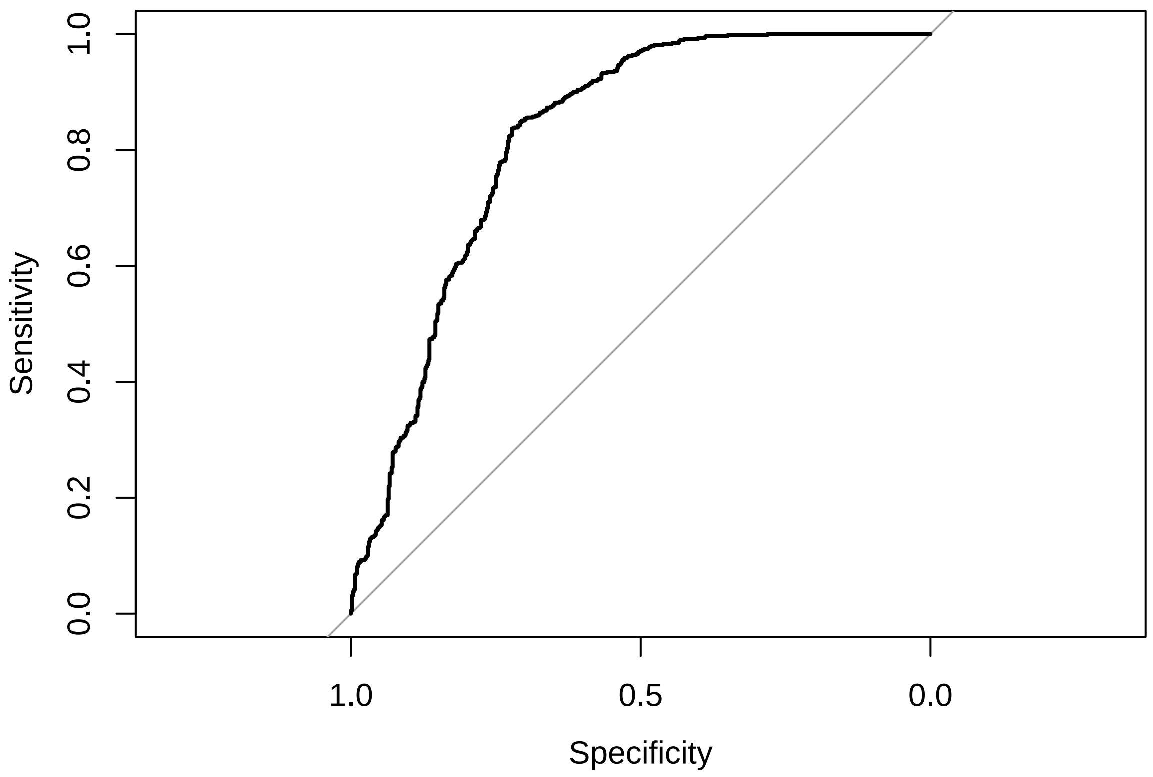
Once we have our train and test datasets, we can fit the model and evaluate its performance.

```r
#########################
## Fit & Eval the Model
#########################

# fit/train the model
model_fit <- lm(
  post_vaccine ~ log_return + Province_State +
    lag_3 + lag_2 + lag_1 +
    lead_1 + lead_2 + lead_3,
  data = weekly_dat[train_idx, ]
)

# summary(model_fit)

# use the model to generate predictions for the test data
test_results <- weekly_dat[test_idx, ]
test_results$prediction <- predict(model_fit, newdata = weekly_dat[test_idx, ])
```

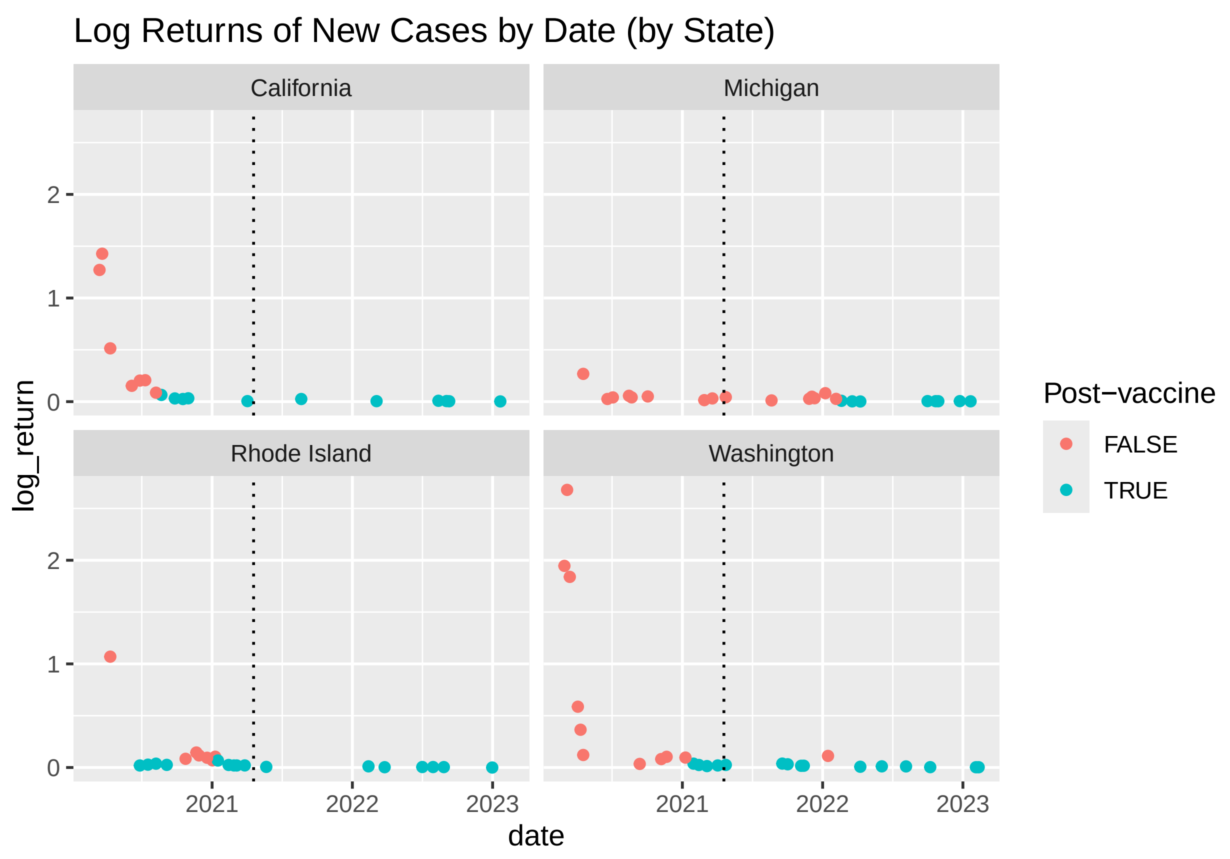
```r
# build and plot the ROC
roc_score <- roc(response = as.numeric(test_results$post_vaccine),
                 predictor = test_results$prediction)

# find the optimal threshold and return relevant performance metrics
roc_coords <- coords(roc_score, x = 'best', transpose = F,
  ret = c(
    'threshold',
    'accuracy',
    'tp',
    'fp',
    'tn',
    'fn',
    'precision',
    'recall'
  )
)

# calculate an F1 score
roc_coords$f1 <- with(roc_coords, 2 * (precision * recall) / (precision + recall))

print(roc_coords)
```

```
##           threshold  accuracy  tp  fp  tn fn precision    recall        f1
## threshold 0.5490978 0.7795883 488 162 421 95 0.7507692 0.8370497 0.7915653
```

The resulting model has an accuracy of 77.96% and an F1 score of 79.16%. Plotting the ROC curve provides visual confirmation that the model performs better than random guessing.



To get an intuition for what the model is doing, we can plot the test results. (Note that the data in these plots are somewhat sparse because we're looking at only test data.)



Roughly speaking, the model looks for low-value log returns to identify post-vaccine results (the blue-green points), which is consistent with our primary hypothesis. However, the red and blue-green points can't be separated by a horizontal line on any of the state's plots, which would have indicated that the model uses a simple low-value threshold. The model doesn't do that, however, because it also considers the sliding window of ± 3 weeks.

Summary

To explore the relationship between the availability of COVID-19 vaccines and the change in the rate of confirmed cases in the US, this analysis uses a simple linear model to predict whether a calendar week came *before* or *after* the general availability of vaccines based on the period-over-period log returns of new cases.

There are multiple sources of bias in this analysis. Adjusting for them is beyond the scope of this paper, so it's important that we identify the primary sources:

- The emphasis placed on vaccines in the design of the model implies that vaccines were the only way to reduce the infection rates of COVID-19. However, multiple measures were used throughout the pandemic: stay-at-home orders, mask mandates, and more.
- The model likewise fails to account for factors that undermine the efficacy of vaccines, e.g., vaccine reluctance, waning immunity, relaxation of public health measures, new variants, etc. (In fact, the performance of the model degrades near the start of 2022, which aligns with arrival of the Omicron variant.)
- This analysis uses a specific date for the availability of vaccines. For the reasons discussed in the paper (multiple vaccines, phased roll outs, production restrictions), using a specific date is an approximation. Many people received their shots before and after that date.

- Vaccines take time before they are effective, so they can't have an immediate effect on case rates, which is not accounted for in this analysis.

Despite these limitations, the model was able to accurately predict 77.96% of test results, well above random (50%), which is consistent with the hypothesis that vaccines helped reduce infection rates. Of course, we cannot presume a causal relationship between vaccines and the reduced rates, but the model's performance supports the possibility that vaccines helped reduce the spread of COVID-19 in the United States during the pandemic.