Figure 1

Set up the workspace.

```
rm(list=ls(all.names=TRUE))
setwd("/Users/bpetros/Desktop/PHIS")
libs <- c("cowplot", "ggplot2", "lubridate", "scales", "tidyverse")
invisible(lapply(libs, function(x) suppressPackageStartupMessages(library(x, character.only = TRUE))))
options(stringsAsFactors=FALSE, scipen = 999)
theme_set(theme_classic())
```

Read in the cleaned input files.

```
# read in cleaned pt data
pt <- read.csv("cleaned/rsv_patient.csv")
pt$Date = ymd(pt$Discharge_Date)

# read in cleaned tested pt data
ptest <- read.csv("cleaned/tested_patient.csv")
ptest$Date = ymd(ptest$Discharge_Date)
ptest$month = floor_date(ptest$Date, "month")

# read in cleaned testing data
tests <- read.csv("cleaned/tests.csv")
tests$date = ymd(tests$date)

# read in cleaned flu data
flu <- read.csv("cleaned/flu.csv")
flu$date = ymd(flu$date)

# read in cleaned flu testing data
flutest <- read.csv("cleaned/flu_tests.csv")
flutest$date = ymd(flutest$date)
```

This function takes a df with a column named "Date" of type Date and counts the number of rows with a date in each month of the study period.

```
# monthly volume
count_volume <- function(data){
  counts <- data %>%
    group_by(month = floor_date(Date, "month"), .drop = TRUE) %>%
    summarise(count = n(), .groups = "drop") %>%
    ungroup() %>%
    complete(month = seq(as.Date("2013-07-01"), as.Date("2023-06-01"), by = "1 month"), fill = list(cou
  return(counts)}
```

This function takes a df with columns "count" (integer) and "month" (Date) as input. It uses interrupted time series (ITS) analysis to identify trends in volume, considering both linear and log-linear models with the following independent variables:

(i) time (t),
(ii) indicator variables for the intermediate period (I_p) and the post-emergence phase (I_e),
(iii) variables enabling a change in slope for each phase, and
(iv) harmonic terms to model seasonality (H_0, H_p, H_e).

$$Linear : volume = a_o + a_1 * I_p + a_2 * I_e + B_0 * t + B_1 * I_p * t + B_2 * I_e * t + H_0 + H_p + H_e$$

$$Log - Linear : log(volume) = a_o + a_1 * I_p + a_2 * I_e + B_0 * t + B_1 * I_p * t + B_2 * I_e * t + H_0 + H_p + H_e$$

We constructed models with all combinations of 0-2 harmonic terms per phase, selecting either the linear or the log-linear model by comparing transformation-adjusted AICs.

When the argument `include_pandemic` is set to `FALSE`, the function models volumes under the counterfactual scenario in which pandemic-associated disruptions in volume did not occur.

```r
# function to find the best model with or without pandemic predictors
find_best_model <- function(df, include_pandemic = TRUE) {

  if(include_pandemic) {
    # add phase-specific predictors
    df <- df %>%
      mutate(time = 1:nrow(.),
             sc2 = as.numeric(month >= ymd("2020-04-01") & month < ymd("2021-04-01")),
             post = as.numeric(month >= ymd("2021-04-01")),
             post_slope = ifelse(post == 1, cumsum(post), 0),
             sc_slope = ifelse(sc2 == 1, cumsum(sc2), 0))

    # create harmonic terms and append to the data frame
    sin_term <- sin(2*pi*df$time /(12))
    cos_term <- cos(2*pi*df$time /(12))
    df[paste0("harmonic_sin_term")] <- (1-df$sc2-df$post)*sin_term
    df[paste0("harmonic_cos_term")] <- (1-df$sc2-df$post)*cos_term
    df[paste0("pandemic_sin_season")] <- df$sc2*sin_term
    df[paste0("pandemic_cos_season")] <- df$sc2*cos_term
    df[paste0("post_sin_season")] <- df$post*sin_term
    df[paste0("post_cos_season")] <- df$post*cos_term

    # always include these predictors
    fixed_predictors <- c("time", "sc2", "sc_slope", "post", "post_slope")
  } else {
    # if considering counterfactual, only time-dependent parameter is time
    df <- df %>%
      mutate(time = 1:nrow(.))

    # create harmonic terms and append to the data frame
    df[paste0("harmonic_sin_term")] <-  sin(2*pi*df$time /(12))
    df[paste0("harmonic_cos_term")] <- cos(2*pi*df$time /(12))
    fixed_predictors <- c("time")}

  # create a vector of predictor variables
  predictor_vars <- colnames(df)[!colnames(df) %in% c("count", "month")]
```

```r
  # initialize variables to keep track of the best models and their AIC values
  best_model_count <- NULL
  best_aic_count <- Inf
  best_model_log_count <- NULL
  best_aic_log_count <- Inf

  # loop through both possible response variables: count and log(count)
  for (response_var in c("count", "log(count)")) {
    for (i in 1:length(predictor_vars)) {
      combinations <- combn(predictor_vars, i)
      for (j in 1:ncol(combinations)) {
        # include "time" and other fixed predictors in the formula for each model
        formula_str <- paste(response_var, "~", paste(c(fixed_predictors, combinations[, j]), collapse =
        formula <- as.formula(formula_str)
        model <- lm(formula, data = df)
        aic <- AIC(model)

        # select the best log-linear and linear models based on AIC
        if (response_var == "count" && aic < best_aic_count) {
          best_model_count <- model
          best_aic_count <- aic
        } else if (response_var == "log(count)" && aic < best_aic_log_count) {
          best_model_log_count <- model
          best_aic_log_count <- aic}}}}

  # choose the log-linear or linear model based on transformation-adjusted AIC
  sum_log_coefficients <- sum((coef(best_model_log_count))[!is.na(coef(best_model_log_count))])
  if (best_aic_count < (best_aic_log_count + 2 * sum_log_coefficients)) {
    best_model <- best_model_count
    response_var <- "count"
  } else {
    best_model <- best_model_log_count
    response_var <- "log(count)"}
  return(best_model)}
```

This function takes as input a df with the following columns: "date" (Date), "numerator" (integer), and "denominator" (integer). It outputs a data frame with the proportion and with the independent variables that will be used for model fitting.

```r
# prepare plotting data
create_plotting_df <- function(data, column = "", value = "", monthly = FALSE) {
  if (monthly) {
    counts <- data %>%
      group_by(month = floor_date(date, "month")) %>%
      summarize(
        numerator = sum(numerator),
        denominator = sum(denominator),
        plot_ratio = numerator / denominator,
        tot = denominator) %>%
      ungroup() %>%
      mutate(numeric_month = month(month),
             time = 1:nrow(.),
             sc2 = as.numeric(month >= ymd("2020-04-01") & month < ymd("2021-04-01")),
```

```r
            post = as.numeric(month >= ymd("2021-04-01")),
            post_slope = ifelse(post == 1, cumsum(post), 0),
            sc_slope = ifelse(sc2 == 1, cumsum(sc2), 0))
  } else {
    counts = data %>%
      mutate(month = floor_date(Date, "month")) %>%
      group_by(month) %>%
      summarize(countY = sum(!!sym(column) == value),
                tot = n(),  # save the number of data points
                plot_ratio = countY / tot) %>%
      ungroup() %>%
      mutate(numeric_month = month(month),
             time = 1:nrow(.),
             sc2 = as.numeric(month >= ymd("2020-04-01") & month < ymd("2021-04-01")),
             post = as.numeric(month >= ymd("2021-04-01")),
             post_slope = ifelse(post == 1, cumsum(post), 0),
             sc_slope = ifelse(sc2 == 1, cumsum(sc2), 0))}
  return(counts)}
```

This function takes the df that is the output of the function `create_proportion_df` and a string, `ylabel`, as input. It uses interrupted time series (ITS) analysis to identify trends in a proportion over time, considering linear models with the following independent variables:

(i) time (t),
(ii) indicator variables for the intermediate period (I_p) and the post-emergence phase (I_e),
(iii) variables enabling a change in slope for each phase, and
(iv) harmonic terms to model seasonality (H_0, H_p, H_e).

$$proportion = a_o + a_1 * I_p + a_2 * I_e + B_0 * t + B_1 * I_p * t + B_2 * I_e * t + H_0 + H_p + H_e$$

It plots the original data points and the model fit.

```r
# plot proportions
generate_and_plot_proportion_model <- function(counts, ylabel) {
  generate_and_compare_models <- function(data, n_harmonics) {
    results <- list()
    # create a vector of predictor variables
    predictor_vars <- c("harmonic_1_sin_term", "harmonic_1_cos_term", "pandemic_1_sin_season",
                        "pandemic_1_cos_season","post_1_sin_season","post_1_cos_season",
                        "harmonic_2_sin_term", "harmonic_2_cos_term", "pandemic_2_sin_season",
                        "pandemic_2_cos_season","post_2_sin_season","post_2_cos_season")
    # always include these predictors
    fixed_predictors <- c("time", "sc2", "sc_slope", "post", "post_slope")
    formula <- paste("plot_ratio", "~", paste(fixed_predictors, collapse = " + "))
    best_model <- lm(as.formula(formula), data)
    best_aic <- AIC(best_model)
    for (i in 1:length(predictor_vars)) {
      combinations <- combn(predictor_vars, i)
      for (j in 1:ncol(combinations)) {
        # fit each model
        formula_str <- paste("plot_ratio", "~", paste(c(fixed_predictors, combinations[, j]), collapse =
        formula <- as.formula(formula_str)
```

```r
      model <- lm(formula, data = data)
      aic <- AIC(model)
      # select the best model based on AIC
      if (aic < best_aic) {
        best_model <- model
        best_aic <- aic}}}
  return(best_model)}

n_harmonics = 2
# create harmonic terms and append to counts
for (n in 1:n_harmonics) {
  sin_term = sin(2*pi*counts$time /(12*n))
  cos_term = cos(2*pi*counts$time /(12*n))
  counts[paste0("harmonic_", n, "_sin_term")] <- (1-counts$sc2-counts$post)*sin_term
  counts[paste0("harmonic_", n, "_cos_term")] <- (1-counts$sc2-counts$post)*cos_term
  counts[paste0("pandemic_", n, "_sin_season")] <- counts$sc2*sin_term
  counts[paste0("pandemic_", n, "_cos_season")] <- counts$sc2*cos_term
  counts[paste0("post_", n, "_sin_season")] <- counts$post*sin_term
  counts[paste0("post_", n, "_cos_season")] <- counts$post*cos_term}

# generate and compare models
best_model <- generate_and_compare_models(counts, n_harmonics)
counts$pred = best_model$fitted.values

counts = counts %>% filter(month < as.Date("2020-04-01") | month >= as.Date("2021-04-01"))

# plot the raw data and the model fit
plot <- ggplot(counts, aes(x = month, y = plot_ratio)) +
  geom_point(data = counts, color = "black") +
  geom_line(data = (counts %>% filter(month < as.Date("2020-04-01"))),
            linewidth = 1, aes(x = month, y = pred), color = "darkmagenta") +
  geom_line(data = (counts %>% filter(month >= as.Date("2021-04-01"))),
            linewidth = 1, aes(x = month, y = pred), color = "darkmagenta") +
  ylab(ylabel) +
  xlab("Month") +
  scale_x_date(labels = scales::date_format("%Y-%b"), breaks = break_dates) +
  geom_rect(data = gray_rectangles,
            aes(xmin = xmin, xmax = xmax, ymin = 0, ymax = 1),
            fill = "grey80", alpha = 0.2, inherit.aes = FALSE) +
  geom_vline(xintercept = as.Date("2020-04-01"), linetype = "dashed", color = "black") +
  geom_vline(xintercept = as.Date("2021-04-01"), linetype = "dashed", color = "black") +
  theme(axis.text.x = element_text(angle = 90), legend.position = "top",
        legend.justification = "left") + labs(color = "") +
  coord_cartesian(ylim = c(max(min(counts$plot_ratio)-0.025, 0), max(counts$plot_ratio) + 0.025))
return(list(plot = plot, best_model = best_model))}
```

This function takes an object of class “lm” from the output of the function `find_best_model` and prints the estimate and the 95% confidence interval for the following variables: a_2, B_0, B_2.

If a linear model was constructed, a_2 (“post”) represents an additive change in the intercept at the start of the post-emergence phase. If a log-linear model was constructed, a_2 (“post”) represents a multiplicative (e.g., fold) change in the intercept at the start of the post-emergence phase.

If a linear model was constructed, B_0 (“time”) represents the slope in the pre-pandemic phase. If a log-linear model was constructed, B_0 ("time") represents an annual percent change in volume during the pre-pandemic phase. Slopes are expressed such that the unit of time is assumed to be years.

If a linear model was constructed, B_2 ("post_slope") represents an additive change in the slope at the start of the post-emergence phase. If a log-linear model was constructed, B_2 ("post_slope") represents a multiplicative change in the slope at the start of the post-emergence phase. Slopes are expressed such that the unit of time is assumed to be years.

```
# print coefficients and their 95% CIs
process_lm_output <- function(model) {
  summary_model <- summary(model)
  formula_terms <- attr(terms(model), "term.labels")
  response_var <- as.character(formula(model)[2])
  # Check if "log" is present in the response variable
  has_log_response_var <- grepl("log", response_var)
  for (term in formula_terms) {
    coefficients <- coef(model)
    ci <- confint(model, level = 0.95)
    if (!grepl("harmonic|season|sc", term)) {
      if (has_log_response_var) {
        if (term %in% c("time", "sc_slope", "post_slope")) {
          coef_value <- round(exp(coefficients[term] * 12), 4)
          ci_coef <- round(exp(ci[term, ] * 12), 4)
        } else {
          coef_value <- round(exp(coefficients[term]), 4)
          ci_coef <- round(exp(ci[term, ]), 4)}
        cat("Coefficient:", term, "\n")
        cat("Value:", coef_value, "\n")
        cat("Exponentiated 95% CI:", ci_coef[1], "to", ci_coef[2], "\n")
      } else {
        if (term %in% c("time", "sc_slope", "post_slope")) {
          coef_value <- round(coefficients[term] * 12, 4)
          ci_coef <- round(ci[term, ] * 12, 4)
        } else {
          coef_value <- round(coefficients[term], 4)
          ci_coef <- round(ci[term, ], 4)}
        cat("Coefficient:", term, "\n")
        cat("Value:", coef_value, "\n")
        cat("95% CI:", ci_coef[1], "to", ci_coef[2], "\n")}}}
  cat("Formula:", as.character(formula(model)), "\n")}
```

This function takes an object of class "lm" from the output of the function `generate_and_plot_proportion_model` and prints the estimate and the 95% confidence interval for the following variables: $a_0, a_0 + a_2, B_0, B_2$.

a_0 ("intercept") represents the average proportion in the pre-pandemic phase. $a_0 + a_2$ ("post intercept") represents the average proportion in the post-emergence phase. B_0 ("time") represents the slope in the pre-pandemic phase. B_2 ("post_slope") represents the additive change in the slope in the post-emergence phase relative to the pre-pandemic phase. Slopes are expressed such that the unit of time is assumed to be years.

```
compute_sum_and_ci <- function(lm_model, coef_name_1, coef_name_2, alpha = 0.05) {
  # extract coefficients and standard errors from the model
  coef_1 <- ifelse(!is.null(coef_name_1), coef(lm_model)[coef_name_1], 0)
  coef_2 <- ifelse(!is.null(coef_name_2), coef(lm_model)[coef_name_2], 0)
  se_1 <- ifelse(!is.null(coef_name_1), sqrt(diag(vcov(lm_model)))[coef_name_1], 0)
```

```r
  se_2 <- ifelse(!is.null(coef_name_2), sqrt(diag(vcov(lm_model)))[coef_name_2], 0)

  # point estimate for sum
  est_sum <- coef_1 + coef_2
  # standard error for the sum
  se_sum <- sqrt(se_1^2 + se_2^2)
  # margin of error for the sum
  margin_of_error <- qnorm(1 - alpha / 2) * se_sum

  # check if coef names contain "slope" or "time" and adjust CI to be annual (vs monthly)
  if (grepl("slope|time", coef_name_1)) {
    ci_lower <- (coef_1 + coef_2 - margin_of_error) * 12
    ci_upper <- (coef_1 + coef_2 + margin_of_error) * 12
    est_sum <- 12*(coef_1 + coef_2)
  } else {
    ci_lower <- coef_1 + coef_2 - margin_of_error
    ci_upper <- coef_1 + coef_2 + margin_of_error}

  result <- list(
    est_sum = est_sum,
    ci_lower = ci_lower,
    ci_upper = ci_upper)
  return(result)}

# get proportion regression results
report_prop_coefficients <- function(lm_model, alpha = 0.05) {
  # extract coefficients and standard errors from the model
  coef_names <- names(coef(lm_model))

  # extract desired coefficients
  intercept_coef <- ifelse("(Intercept)" %in% coef_names, "(Intercept)", NULL)
  time_coef <- ifelse("time" %in% coef_names, "time", NULL)
  post_coef <- ifelse("post" %in% coef_names, "post", NULL)
  post_slope_coef <- ifelse("post_slope" %in% coef_names, "post_slope", NULL)

  # report intercept and 95% CI
  if (!is.null(intercept_coef)) {
    intercept_result <- (compute_sum_and_ci(lm_model, intercept_coef, NULL, alpha))
    cat("intercept:", round(intercept_result$est_sum, 4), "(", round(intercept_result$ci_lower, 4), "-"

  # report time and 95% CI
  if (!is.null(time_coef)) {
    time_result <- compute_sum_and_ci(lm_model, time_coef, NULL, alpha)
    cat("time:", round(time_result$est_sum, 5), "(", round(time_result$ci_lower, 5), "-", round(time_re

  # report intercept + post and 95% CI
  if (!is.null(intercept_coef) && !is.null(post_coef)) {
    intercept_post_result <- (compute_sum_and_ci(lm_model, intercept_coef, post_coef, alpha))
    cat("intercept + post:", round(intercept_post_result$est_sum, 4), "(", round(intercept_post_result$

  # report post_slope and 95% CI
  if (!is.null(post_slope_coef)) {
    time_result <- compute_sum_and_ci(lm_model, post_slope_coef, NULL, alpha)
```

```r
    cat("post_slope", round(time_result$est_sum, 5), "(", round(time_result$ci_lower, 5), "-", round(ti
```

Create objects that will be used for plotting.

```r
# shade every other year
grey_years <- seq(year(min(tests$date)), year(max(tests$date))-1, by = 2)
# sequence of 3-mo date intervals
break_dates <- seq(min(tests$date), max(tests$date)+31, by = "3 months")

# create a df to store gray rectangles
gray_rectangles <- data.frame(
  xmin = as.Date(paste(grey_years, "-07-01", sep = "")),
  xmax = as.Date(paste(grey_years + 1, "-06-30", sep = "")),
  ymin = 1, ymax = Inf)
rm(grey_years)
```

Model testing volume over time (Figure 1A).

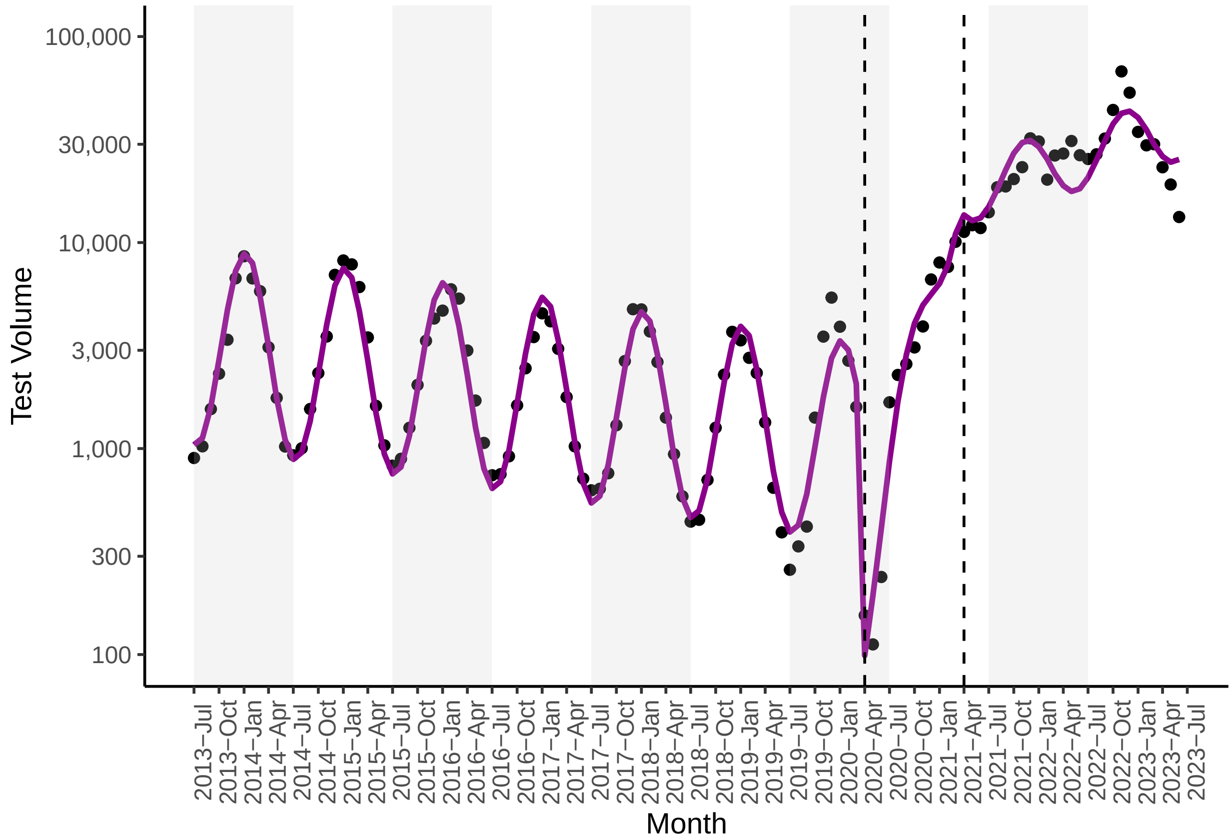
```r
test_counts = data.frame("month" = tests$date,
                         "count" = tests$ED_tests + tests$IP_tests)

# model for test volume
test_model = find_best_model(test_counts)
summary(test_model)
```

```
##
## Call:
## lm(formula = formula, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6437 -0.1515 -0.0159  0.1033  0.6773
##
## Coefficients:
##                      Estimate Std. Error t value             Pr(>|t|)
## (Intercept)          8.071748   0.054419 148.327 < 0.0000000000000002 ***
## time                -0.013535   0.001154 -11.732 < 0.0000000000000002 ***
## sc2                 -2.269989   0.160537 -14.140 < 0.0000000000000002 ***
## sc_slope             0.449810   0.020697  21.733 < 0.0000000000000002 ***
## post                 2.941881   0.117882  24.956 < 0.0000000000000002 ***
## post_slope           0.040795   0.006237   6.540        0.00000000205 ***
## harmonic_sin_term   -0.655425   0.037898 -17.294 < 0.0000000000000002 ***
## harmonic_cos_term   -0.901772   0.038321 -23.532 < 0.0000000000000002 ***
## pandemic_sin_season  0.632491   0.100887   6.269        0.00000000745 ***
## post_sin_season      0.101092   0.068709   1.471                0.144
## post_cos_season     -0.351412   0.065695  -5.349        0.00000049232 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2419 on 109 degrees of freedom
## Multiple R-squared:  0.9723, Adjusted R-squared:  0.9697
## F-statistic: 382.1 on 10 and 109 DF,  p-value: < 0.00000000000000022
```

```r
test_counts$pred = exp(test_model$fitted.values)
process_lm_output(test_model)
```

```
## Coefficient: time
## Value: 0.8501
## Exponentiated 95% CI: 0.8271 to 0.8737
## Coefficient: post
## Value: 18.9515
## Exponentiated 95% CI: 15.0029 to 23.9392
## Coefficient: post_slope
## Value: 1.6316
## Exponentiated 95% CI: 1.4066 to 1.8925
## Formula: ~ log(count) time + sc2 + sc_slope + post + post_slope + harmonic_sin_term + harmonic_cos_te
```

```r
fig1a <- ggplot(test_counts, aes(x = month, y = count)) +
  geom_point(color = "black") +
  geom_line(linewidth = 1, aes(x = month, y = pred), color = "darkmagenta") +
  ylab("Test Volume") +
  xlab("Month") +
  scale_y_continuous(trans = 'log10',
                     breaks = c(100, 300, 1000, 3000, 10000, 30000, 100000), labels = comma) +
  coord_cartesian(ylim = c(99, 100000)) +
  scale_x_date(labels = scales::date_format("%Y-%b"), breaks = break_dates) +
  geom_rect(
    data = gray_rectangles,
    aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
    fill = "grey80", alpha = 0.2, inherit.aes = FALSE) +
  geom_vline(xintercept = as.Date("2020-04-01"), linetype = "dashed", color = "black") +
  geom_vline(xintercept = as.Date("2021-04-01"), linetype = "dashed", color = "black") +
  theme(axis.text.x = element_text(angle = 90), legend.position = "top",
        legend.justification = "left") + labs(color = "")
fig1a
```




```
rm(test_counts, test_model)
```

Model patient volume over time (Figure 1B).

```
tot_counts =  pt %>% count_volume()

# model for patient volume
pt_model = find_best_model(tot_counts)
summary(pt_model)
```

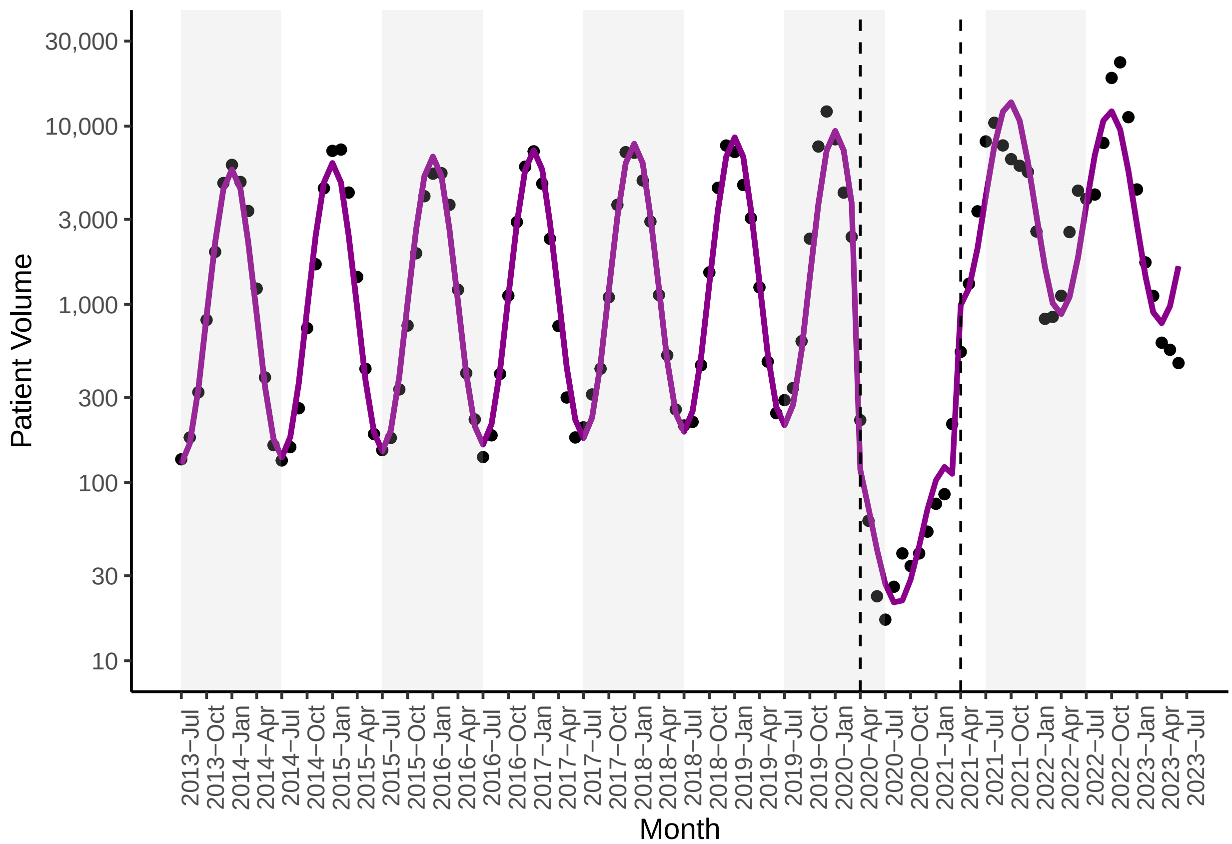
```
##
## Call:
## lm(formula = formula, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.25160 -0.20490 -0.03946  0.16228  0.87022
##
## Coefficients:
##                      Estimate Std. Error t value             Pr(>|t|)
## (Intercept)          6.720280   0.084181  79.832 < 0.0000000000000002 ***
## time                 0.006926   0.001785   3.881              0.00018 ***
## sc2                 -3.087884   0.359445  -8.591   0.0000000000000726 ***
## sc_slope            -0.039625   0.051220  -0.774              0.44084
## post                 0.882913   0.182353   4.842   0.0000043157163574 ***
```

```
## post_slope           -0.016630   0.009648  -1.724              0.08764 .
## harmonic_sin_term    -0.929748   0.058625 -15.859 < 0.0000000000000002 ***
## harmonic_cos_term    -1.635097   0.059280 -27.583 < 0.0000000000000002 ***
## pandemic_sin_season -0.916175   0.161102  -5.687   0.0000001115654224 ***
## pandemic_cos_season -0.358986   0.244600  -1.468              0.14511
## post_sin_season      1.208471   0.106287  11.370 < 0.0000000000000002 ***
## post_cos_season     -0.585209   0.101625  -5.759   0.0000000807411716 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3742 on 108 degrees of freedom
## Multiple R-squared:  0.9544, Adjusted R-squared:  0.9497
## F-statistic: 205.3 on 11 and 108 DF,  p-value: < 0.00000000000000022
```

```r
tot_counts$pred = exp(pt_model$fitted.values)
process_lm_output(pt_model)
```

```
## Coefficient: time
## Value: 1.0867
## Exponentiated 95% CI: 1.0415 to 1.1338
## Coefficient: post
## Value: 2.4179
## Exponentiated 95% CI: 1.6845 to 3.4707
## Coefficient: post_slope
## Value: 0.8191
## Exponentiated 95% CI: 0.6511 to 1.0304
## Formula: ~ log(count) time + sc2 + sc_slope + post + post_slope + harmonic_sin_term + harmonic_cos_te
```

```r
fig1b <- ggplot(tot_counts, aes(x = month, y = count)) +
  geom_point(color = "black") +
  geom_line(aes(x = month, y = pred), color = "darkmagenta", linewidth = 1) +
  geom_line(data = (tot_counts %>% filter(month < as.Date("2020-04-01"))),
            linewidth = 1, aes(x = month, y = pred), color = "darkmagenta") +
  geom_line(data = (tot_counts %>% filter(month >= as.Date("2021-04-01"))),
            linewidth = 1, aes(x = month, y = pred), color = "darkmagenta") +
  ylab("Patient Volume") +
  xlab("Month") +
  scale_y_continuous(trans = 'log10',
                     breaks = c(10, 30, 100, 300, 1000, 3000, 10000, 30000), labels = comma) +
  coord_cartesian(ylim = c(10, 30000)) +
  scale_x_date(labels = scales::date_format("%Y-%b"), breaks = break_dates) +
  geom_rect(
    data = gray_rectangles,
    aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
    fill = "grey80", alpha = 0.2, inherit.aes = FALSE) +
  geom_vline(xintercept = as.Date("2020-04-01"), linetype = "dashed", color = "black") +
  geom_vline(xintercept = as.Date("2021-04-01"), linetype = "dashed", color = "black") +
  theme(axis.text.x = element_text(angle = 90), legend.position = "top",
        legend.justification = "left") + labs(color = "")
fig1b
```



Model the proportion of tests that were positive over time (Figure 1C). For patients who were tested for RSV multiple times and who received an RSV diagnosis, assume that at least one test was positive for RSV.

```
pos_test_counts = pt %>% filter(Num_Tests > 0) %>% count_volume()

# generate df with counts of pos tests and total tests
ratio = data.frame("date" = pos_test_counts$month, "numerator" = pos_test_counts$count,
                   "denominator" = (tests$ED_tests + tests$IP_tests))

# proportion of tests that are positive
ratio_model = generate_and_plot_proportion_model(create_plotting_df(ratio, monthly = TRUE),
                                                  "Test Positivity")
summary(ratio_model$best_model)
```

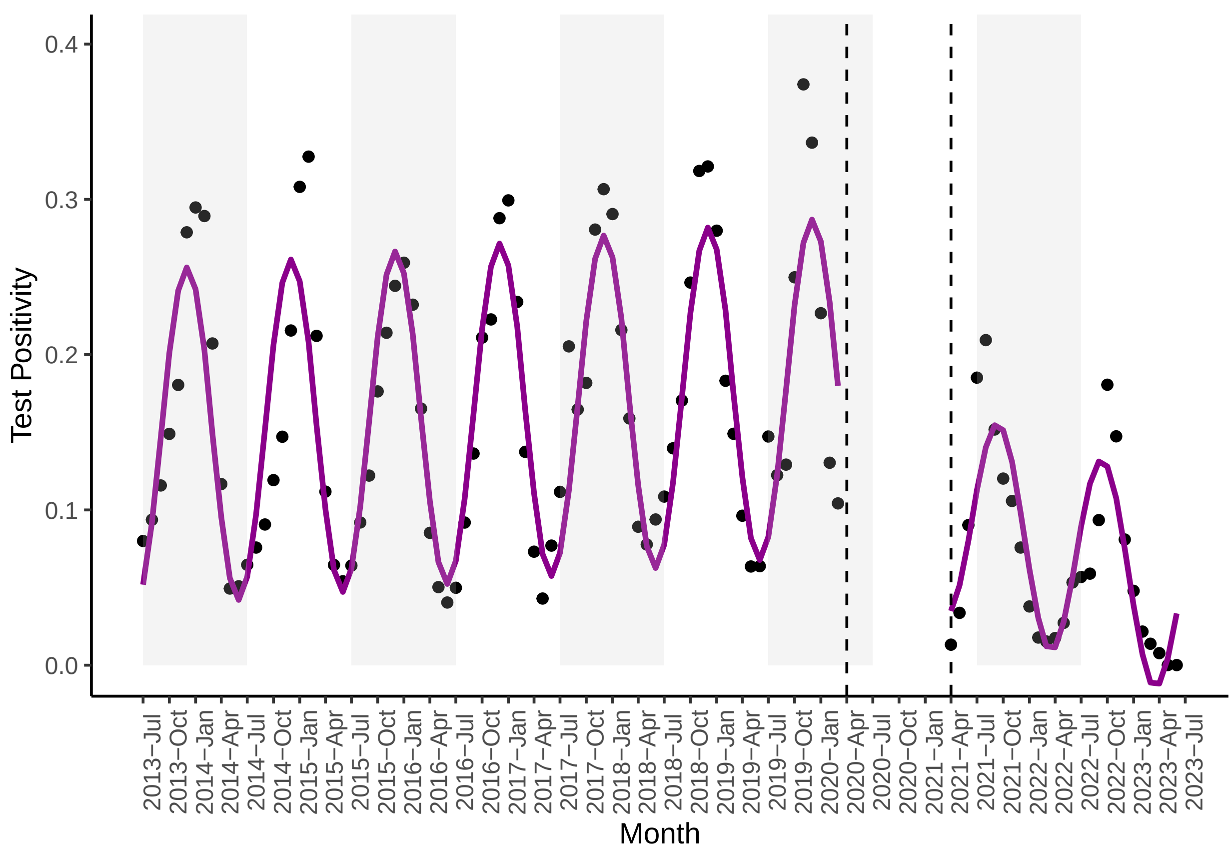
```
##
## Call:
## lm(formula = formula, data = data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.103288 -0.024880 -0.001256  0.019590  0.119555
##
## Coefficients:
##                          Estimate Std. Error t value             Pr(>|t|)
## (Intercept)             0.1452549  0.0089341  16.259 < 0.0000000000000002 ***
## time                    0.0004280  0.0001894   2.259               0.0258 *
```

```
## sc2                      -0.0149828  0.0882723  -0.170             0.8655
## sc_slope                 -0.0186656  0.0100392  -1.859             0.0657 .
## post                     -0.0841494  0.0193994  -4.338         0.0000320591 ***
## post_slope               -0.0023758  0.0010269  -2.314             0.0225 *
## harmonic_1_cos_term      -0.1083531  0.0063084 -17.176 < 0.0000000000000002 ***
## post_1_sin_season         0.0653625  0.0113129   5.778         0.0000000715 ***
## post_1_cos_season        -0.0149646  0.0108167  -1.383             0.1693
## pandemic_2_cos_season     0.0906523  0.0586572   1.545             0.1251
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03983 on 110 degrees of freedom
## Multiple R-squared:  0.8427, Adjusted R-squared:  0.8298
## F-statistic: 65.46 on 9 and 110 DF,  p-value: < 0.00000000000000022
```

```r
report_prop_coefficients(ratio_model$best_model)
```

```
## intercept: 0.1453 ( 0.1277 - 0.1628 )
## time: 0.00514 ( 0.00068 - 0.00959 )
## intercept + post: 0.0611 ( 0.0192 - 0.103 )
## post_slope -0.02851 ( -0.05266 - -0.00436 )
```

```r
fig1c = ratio_model$plot
fig1c
```



```r
rm(pos_test_counts, ratio, ratio_model)
```

Compare observed patient volumes in the post-emergence phase to (i) the volumes that would be predicted from a model trained only on pre-pandemic data, or (ii) the volumes that would be predicted if testing volume remained unchanged from the pre-pandemic phase to the post-emergence phase (Figure 1D).

```r
# predict post-emergence patient volumes using only pre-pandemic data
pre_pandemic_df = pt %>% filter(Date < as.Date("2020-04-01")) %>% count_volume() %>% filter(month < as.I
pre_pandemic_model = find_best_model(pre_pandemic_df, include_pandemic = FALSE)
#summary(pre_pandemic_model)

# create a prediction df and generate prediction interval for post-emergence volumes
post_emerge_pred <- pt %>% filter(Date < as.Date("2020-04-01")) %>% count_volume() %>%
  complete(month = seq(as.Date("2013-07-01"), as.Date("2023-12-01"), by = "1 month"), fill = list(count
                           harmonic_sin_term = sin(2*pi*time /(12)),
                           harmonic_cos_term = cos(2*pi*time /(12)))

predict <- data.frame(exp(predict(pre_pandemic_model, newdata = post_emerge_pred, interval = "predictio

# add prediction interval to the df with observed patient volumes
tot_counts = tot_counts %>%
  complete(month = seq(as.Date("2013-07-01"), as.Date("2023-12-01"), by = "1 month"),
           fill = list(count = NA, pred = NA))
tot_counts$lwr = predict$lwr
tot_counts$upr = predict$upr
rm(pre_pandemic_df, post_emerge_pred, predict)
tot_counts$group = "Observed Data"

# predict post-emergence testing volumes using only pre-pandemic data
pre_pandemic_df = data.frame("month" = tests$date, "count" = tests$ED_tests + tests$IP_tests) %>% filte
pre_pandemic_model = find_best_model(pre_pandemic_df, include_pandemic = FALSE)
#summary(pre_pandemic_model)
post_emerge_pred = data.frame("month" = tests$date, "count" = tests$ED_tests + tests$IP_tests) %>% muta
post_emerge_pred$pred <- as.integer(exp(predict(pre_pandemic_model, post_emerge_pred, type = "response")

# bootstrap tests in the post-emergence phase according to the volume predicted by pre-pandemic test vo
bootstrap <- data.frame(prct = numeric(),
                        coefficient_post = numeric(),
                        ci_lower = numeric(),
                        ci_upper = numeric())
for (i in 1:50) {
  # randomly select tests in the post-emergence phase according to the volume predicted by the model tr
  ptest_random <- ptest %>%
    filter(month < as.Date("2020-04-01")) %>%  # Keep all rows before 2020-04-01
    bind_rows(
      ptest %>%
        filter(month >= as.Date("2020-04-01")) %>%
        left_join(post_emerge_pred, by = "month") %>%
        group_by(month) %>%
        sample_n(size = first(pred), replace = TRUE))

  # count number of times each testing encounter was selected during bootstrap
  counts <- ptest_random %>%
```

```r
  count(Discharge_ID) %>%
  rename(N = n)
  pt_with_counts <- pt %>%
  left_join(counts, by = "Discharge_ID") %>%
  replace_na(list(N = 0)) # replace NAs with 0s
  rm(counts)

  # total counterfactual patient volume was patients pre-pandemic, pts w/o dx test, and pts selected vi
  pred_pt_vol <- pt_with_counts %>%
  group_by(Discharge_ID) %>%
  uncount(N) %>%
  bind_rows(pt %>% filter(Num_Tests == 0 | Date < as.Date("2020-04-01")))

  # model the volume of patients dx under stable testing regime
  pred_pt_counts <- pred_pt_vol %>% count_volume()
  pt_pred_model <- find_best_model(pred_pt_counts)
  pred_pt_counts$pred = exp(pt_pred_model$fitted.values)
  pred_pt_counts$lwr = NA
  pred_pt_counts$upr = NA
  pred_pt_counts$group = "With Pre-Pandemic Test Volume"
  bootstrap <- bind_rows(bootstrap,
                         data.frame(
                                    prct = 1-(exp(pt_pred_model$coefficients['post']) - 1)/(exp(pt_model
                                    coefficient_post = exp(pt_pred_model$coefficients['post']),
                                    ci_lower =exp(confint(pt_pred_model, level = 0.95)["post",1]),
                                    ci_upper = exp(confint(pt_pred_model, level = 0.95)["post",2])))}

summary(pt_pred_model)
```

```
##
## Call:
## lm(formula = formula, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.01890 -0.20942 -0.04958  0.15119  0.80874
##
## Coefficients:
##                       Estimate Std. Error t value             Pr(>|t|)
## (Intercept)           7.080681   0.081685  86.683 < 0.0000000000000002 ***
## time                  0.003680   0.001732   2.125               0.0359 *
## sc2                  -2.885839   0.348788  -8.274    0.000000000000372 ***
## sc_slope             -0.095051   0.049701  -1.912               0.0585 .
## post                  0.314056   0.176946   1.775               0.0787 .
## post_slope           -0.007539   0.009362  -0.805               0.4224
## harmonic_sin_term    -0.876746   0.056887 -15.412 < 0.0000000000000002 ***
## harmonic_cos_term    -1.632105   0.057522 -28.374 < 0.0000000000000002 ***
## pandemic_sin_season -1.004941   0.156326  -6.429    0.000000003589467 ***
## pandemic_cos_season -0.395241   0.237348  -1.665               0.0988 .
## post_sin_season       1.195966   0.103135  11.596 < 0.0000000000000002 ***
## post_cos_season      -0.467578   0.098612  -4.742    0.000006513655188 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.3631 on 108 degrees of freedom
## Multiple R-squared:  0.9548, Adjusted R-squared:  0.9502
## F-statistic: 207.3 on 11 and 108 DF,  p-value: < 0.00000000000000022
```

```r
process_lm_output(pt_pred_model)
```

```
## Coefficient: time
## Value: 1.0452
## Exponentiated 95% CI: 1.003 to 1.0891
## Coefficient: post
## Value: 1.369
## Exponentiated 95% CI: 0.964 to 1.9441
## Coefficient: post_slope
## Value: 0.9135
## Exponentiated 95% CI: 0.7311 to 1.1414
## Formula: ~ log(count) time + sc2 + sc_slope + post + post_slope + harmonic_sin_term + harmonic_cos_te
```

```r
summary(bootstrap)
```

```
##       prct        coefficient_post    ci_lower         ci_upper
##  Min.   :0.7370   Min.   :1.359    Min.   :0.9509   Min.   :1.933
##  1st Qu.:0.7407   1st Qu.:1.363    1st Qu.:0.9560   1st Qu.:1.942
##  Median :0.7422   Median :1.366    Median :0.9581   Median :1.945
##  Mean   :0.7424   Mean   :1.365    Mean   :0.9581   Mean   :1.945
##  3rd Qu.:0.7440   3rd Qu.:1.368    3rd Qu.:0.9600   3rd Qu.:1.949
##  Max.   :0.7472   Max.   :1.373    Max.   :0.9655   Max.   :1.952
```

```r
cat("Percent inc in patient volume attributable to inc testing volume:")
```

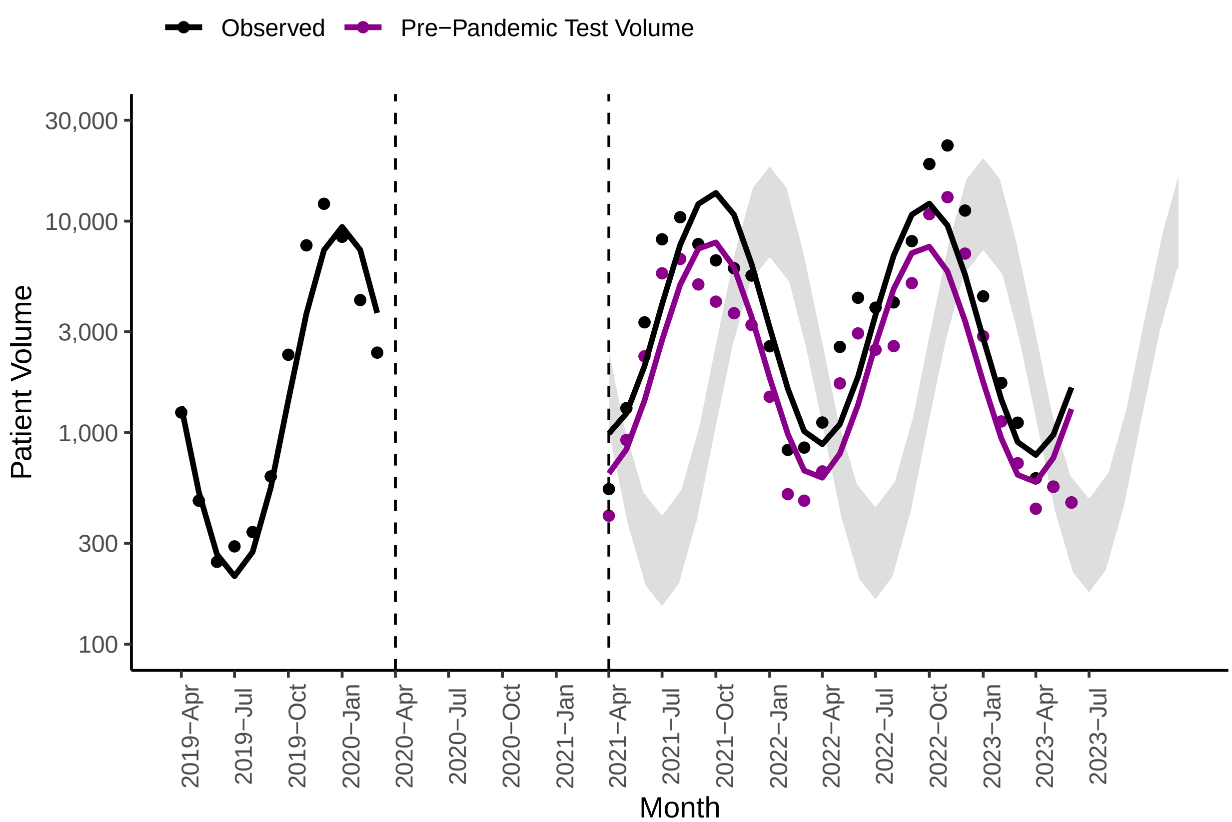
```
## Percent inc in patient volume attributable to inc testing volume:
```

```r
print(median(bootstrap$prct))
```

```
## [1] 0.7422096
```

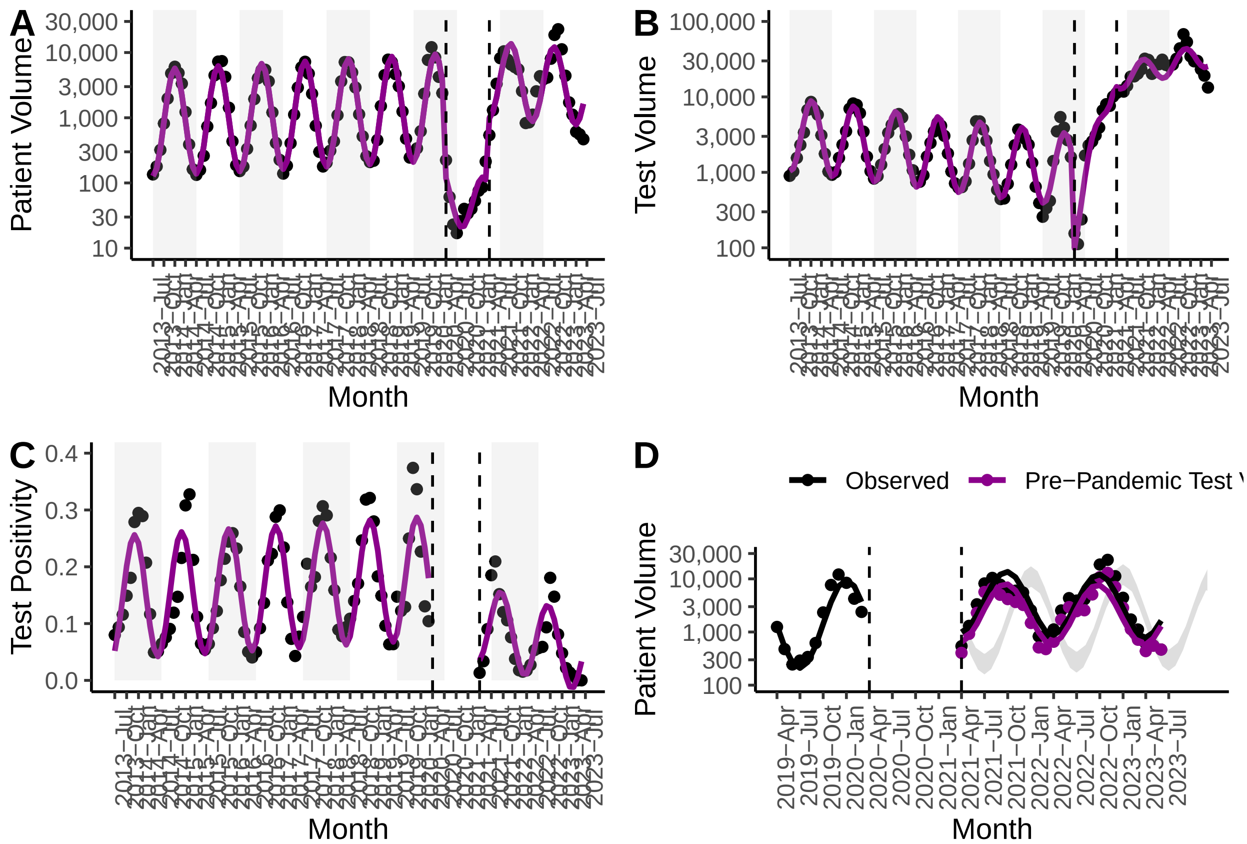
```r
# create plotting df
plot_pt = rbind(tot_counts, pred_pt_counts %>% filter(month >= as.Date("2020-04-01")))
rm(pred_pt_counts, pred_pt_vol)

# plot the observed and modeled counts
fig1d <- ggplot(plot_pt %>% filter(month >= as.Date("2019-04-01")),
                aes(x = month, y = count, color = group)) +
  geom_ribbon(data = plot_pt %>% filter(month >= as.Date("2021-04-01") & group == "Observed Data"),
              aes(x = month, ymin = lwr, ymax = upr), fill = "lightgrey", color = "white", alpha = 0.75
  geom_point(data = plot_pt %>% filter(month >= as.Date("2019-04-01")) %>%
               filter(month < as.Date("2020-04-01") | month >= as.Date("2021-04-01")),
             aes(x = month, y = count, color = group), na.rm = TRUE) +
  geom_line(data = plot_pt %>% filter(month >= as.Date("2019-04-01")) %>%
              filter(month < as.Date("2020-04-01")),
```

```
             aes(x = month, y = pred, color = group), linewidth = 1, na.rm = TRUE) +
  geom_line(data = plot_pt %>% filter(month >= as.Date("2021-04-01")),
             aes(x = month, y = pred, color = group), linewidth = 1, na.rm = TRUE) +
  ylab("Patient Volume") +
  xlab("Month") +
  scale_color_manual(values = c("black", "darkmagenta"),
                     breaks = c("Observed Data", "With Pre-Pandemic Test Volume"),
                     labels = c("Observed", "Pre-Pandemic Test Volume")) +
  scale_y_continuous(trans = "log10", breaks = c(100, 300, 1000, 3000, 10000, 30000), labels = comma) +
  coord_cartesian(ylim = c(100, 30000)) +
  geom_vline(xintercept = as.Date("2020-04-01"), linetype = "dashed", color = "black") +
  geom_vline(xintercept = as.Date("2021-04-01"), linetype = "dashed", color = "black") +
  scale_x_date(labels = scales::date_format("%Y-%b"), breaks = break_dates) +
  theme(axis.text.x = element_text(angle = 90), legend.position = "top",
        legend.justification = "left") + labs(color = "")
fig1d
```



Arrange figure panels (Figure 1).

```
fig1 = plot_grid(fig1b, fig1a, fig1c, fig1d, nrow = 2, labels = c("A", "B", "C", "D"))
fig1
```




```r
ggsave("figs/fig1.pdf", plot = fig1, width = 12, height = 9)
```

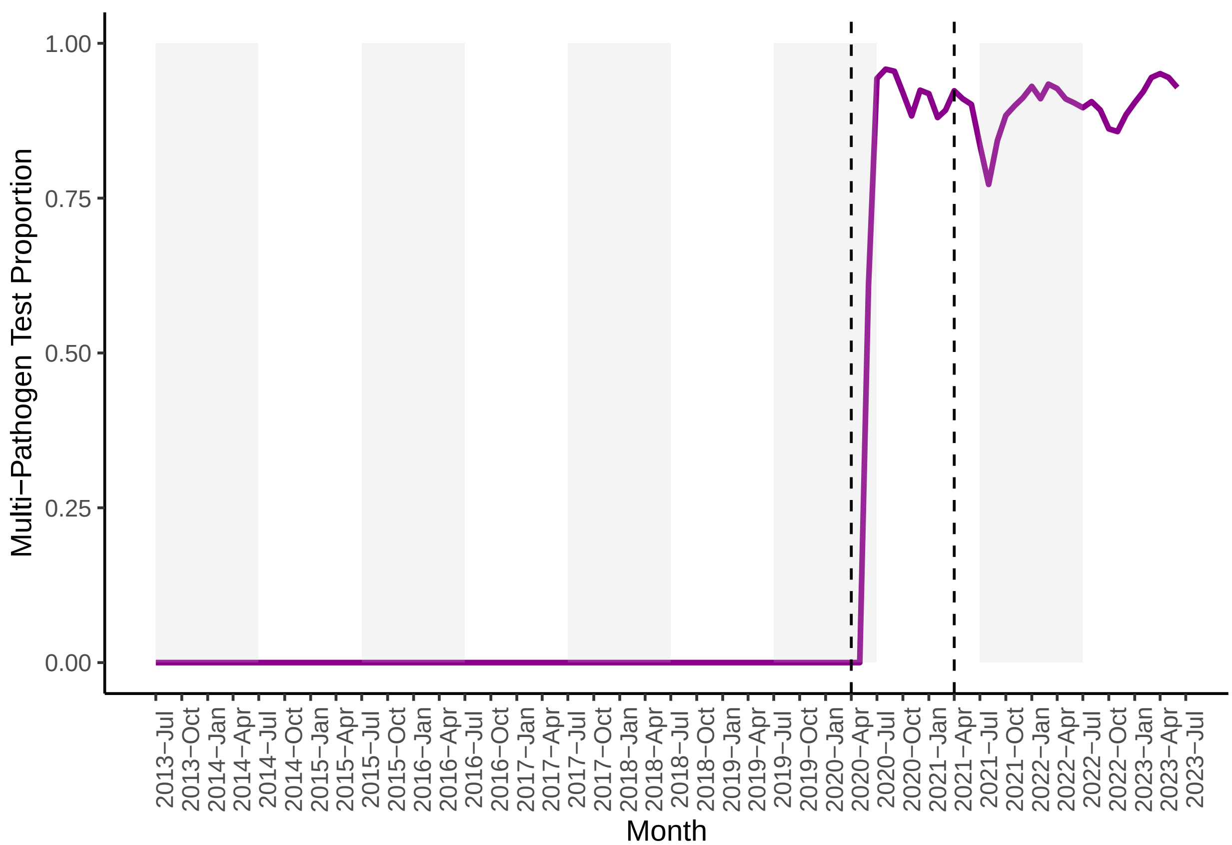
Plot the fraction of RSV tests that were SARS-CoV-2 multi-pathogen tests over time (Supplementary Figure 1).

```r
# calculate the fraction of multi tests relative to the total tests
tests$tot_multi = tests$ED_tests_multi + tests$IP_tests_multi
tests$tot = tests$ED_tests + tests$IP_tests
tests$non_multi = tests$tot - tests$tot_multi
tests$fraction <- tests$tot_multi / tests$tot

# significantly more multi tests in post-emergence phase (as expected)
tests_pre = tests %>% filter(date < as.Date("2020-04-01"))
tests_post = tests %>% filter(date >= as.Date("2021-04-01"))
fisher.test(rbind(cbind(sum(tests_pre$tot_multi), sum(tests_pre$non_multi)),
                  cbind(sum(tests_post$tot_multi), sum(tests_post$non_multi))))
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  rbind(cbind(sum(tests_pre$tot_multi), sum(tests_pre$non_multi)), cbind(sum(tests_post$tot_mul
## p-value < 0.00000000000000022
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.0000000000 0.0001190186
```

```
## sample estimates:
## odds ratio
##          0
```

```r
# plot the data
sf1a <- ggplot(tests, aes(x = date, y = fraction)) +
  geom_line(linewidth = 1, col = "darkmagenta") +
  ylab("Multi-Pathogen Test Proportion") +
  xlab("Month") +
  coord_cartesian(ylim = c(0, 1)) +
  scale_x_date(labels = scales::date_format("%Y-%b"), breaks = break_dates) +
  geom_rect(
    data = gray_rectangles,
    aes(xmin = xmin, xmax = xmax, ymin = 0, ymax = 1),
    fill = "grey80", alpha = 0.2, inherit.aes = FALSE) +
  geom_vline(xintercept = as.Date("2020-04-01"), linetype = "dashed", color = "black") +
  geom_vline(xintercept = as.Date("2021-04-01"), linetype = "dashed", color = "black") +
  theme(axis.text.x = element_text(angle = 90), legend.position = "top",
        legend.justification = "left") + labs(color = "")
sf1a
```




```r
ggsave("figs/suppfig1.pdf", plot = sf1a, width = 6, height = 5)
```

Model influenza testing volume and patient volume over time (Supplementary Figure 2).

```r
# monthly pt volume
flu_counts = data.frame("month" = flu$date, "count" = flu$ED_dc + flu$IP)

# model for influenza pt volume
flu_model = find_best_model(flu_counts)
summary(flu_model)
```

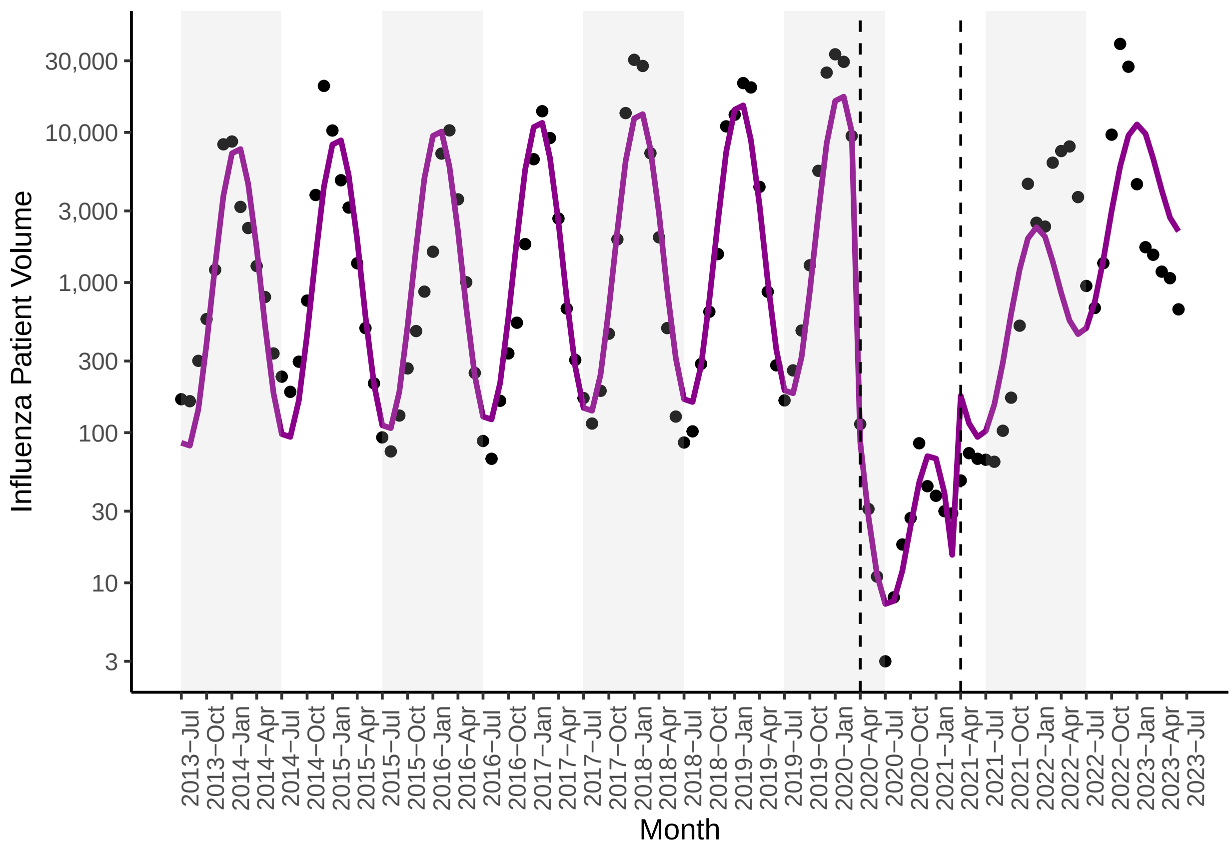
```
##
## Call:
## lm(formula = formula, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.77723 -0.46875 -0.05912  0.52473  2.66858
##
## Coefficients:
##                       Estimate Std. Error t value             Pr(>|t|)
## (Intercept)           6.624829   0.189498  34.960 < 0.0000000000000002 ***
## time                  0.011162   0.004017   2.779             0.006441 **
## sc2                  -2.742392   0.809142  -3.389             0.000979 ***
## sc_slope             -0.253135   0.115299  -2.195             0.030270 *
## post                 -2.389755   0.410492  -5.822         0.0000000606 ***
## post_slope            0.120302   0.021719   5.539         0.0000002163 ***
## harmonic_sin_term    -1.694623   0.131970 -12.841 < 0.0000000000000002 ***
## harmonic_cos_term    -1.546502   0.133443 -11.589 < 0.0000000000000002 ***
## pandemic_sin_season -0.847663   0.362655  -2.337             0.021263 *
## pandemic_cos_season -1.636278   0.550615  -2.972             0.003651 **
## post_sin_season      -0.380193   0.239260  -1.589             0.114976
## post_cos_season      -1.128513   0.228766  -4.933         0.0000029536 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8423 on 108 degrees of freedom
## Multiple R-squared:  0.8556, Adjusted R-squared:  0.8409
## F-statistic: 58.19 on 11 and 108 DF,  p-value: < 0.00000000000000022
```

```r
flu_counts$pred = exp(flu_model$fitted.values)
process_lm_output(flu_model)
```

```
## Coefficient: time
## Value: 1.1433
## Exponentiated 95% CI: 1.0391 to 1.258
## Coefficient: post
## Value: 0.0917
## Exponentiated 95% CI: 0.0406 to 0.2068
## Coefficient: post_slope
## Value: 4.236
## Exponentiated 95% CI: 2.5269 to 7.1011
## Formula: ~ log(count) time + sc2 + sc_slope + post + post_slope + harmonic_sin_term + harmonic_cos_t
```

```r
# plot influenza pt volume
sf2a <- ggplot(flu_counts, aes(x = month, y = count)) +
```

```
  geom_point(color = "black") +
  geom_line(linewidth = 1, aes(x = month, y = pred), color = "darkmagenta") +
  ylab("Influenza Patient Volume") +
  xlab("Month") +
  scale_y_continuous(trans = 'log10',
                     breaks = c(3, 10, 30, 100, 300, 1000, 3000, 10000, 30000), labels = comma) +
  coord_cartesian(ylim = c(3, 40000)) +
  scale_x_date(labels = scales::date_format("%Y-%b"), breaks = break_dates) +
  geom_rect(
    data = gray_rectangles,
    aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
    fill = "grey80", alpha = 0.2, inherit.aes = FALSE) +
  geom_vline(xintercept = as.Date("2020-04-01"), linetype = "dashed", color = "black") +
  geom_vline(xintercept = as.Date("2021-04-01"), linetype = "dashed", color = "black") +
  theme(axis.text.x = element_text(angle = 90), legend.position = "top",
        legend.justification = "left") + labs(color = "")
sf2a
```




```
rm(flu, flu_counts, flu_model)

# monthly test volume
flutest_counts = data.frame("month" = flutest$date, "count" = flutest$ED_tests + flutest$IP_tests)

# model for influenza test volume
flutest_model = find_best_model(flutest_counts)
```

```r
summary(flutest_model)
```

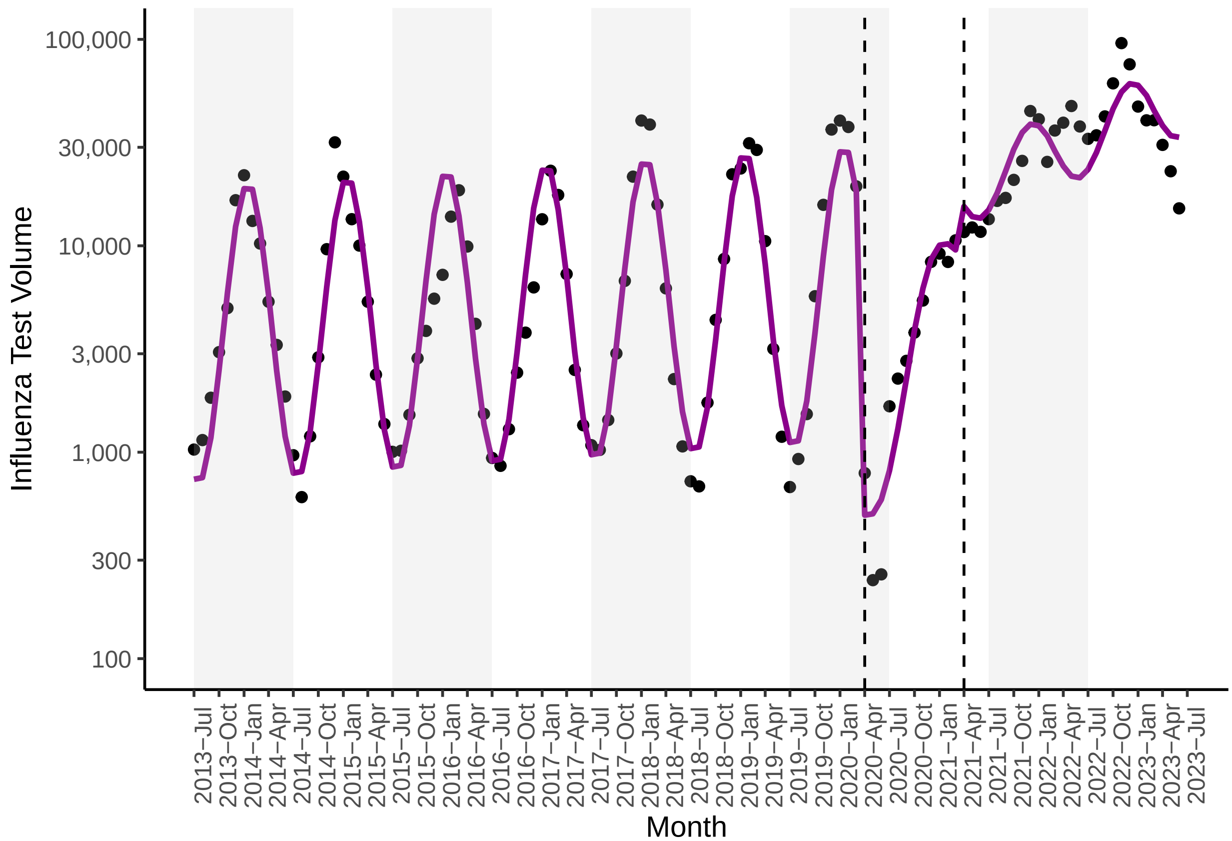
```
##
## Call:
## lm(formula = formula, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.09822 -0.20665 -0.00144  0.24443  0.87079
##
## Coefficients:
##                      Estimate Std. Error t value             Pr(>|t|)
## (Intercept)          8.204001   0.086460  94.888 < 0.0000000000000002 ***
## time                 0.005704   0.001833   3.112             0.002369 **
## sc2                 -2.392726   0.352849  -6.781       0.000000000622 ***
## sc_slope             0.235244   0.049884   4.716       0.000007107211 ***
## post                 1.082453   0.185997   5.820       0.000000059005 ***
## post_slope           0.031902   0.009723   3.281             0.001386 **
## harmonic_sin_term   -1.152114   0.060213 -19.134 < 0.0000000000000002 ***
## harmonic_cos_term   -1.186741   0.060885 -19.492 < 0.0000000000000002 ***
## pandemic_cos_season -0.616894   0.243365  -2.535             0.012655 *
## post_cos_season     -0.411636   0.104061  -3.956             0.000135 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3843 on 110 degrees of freedom
## Multiple R-squared:  0.9316, Adjusted R-squared:  0.926
## F-statistic: 166.4 on 9 and 110 DF,  p-value: < 0.00000000000000022
```

```r
flutest_counts$pred = exp(flutest_model$fitted.values)
process_lm_output(flutest_model)
```

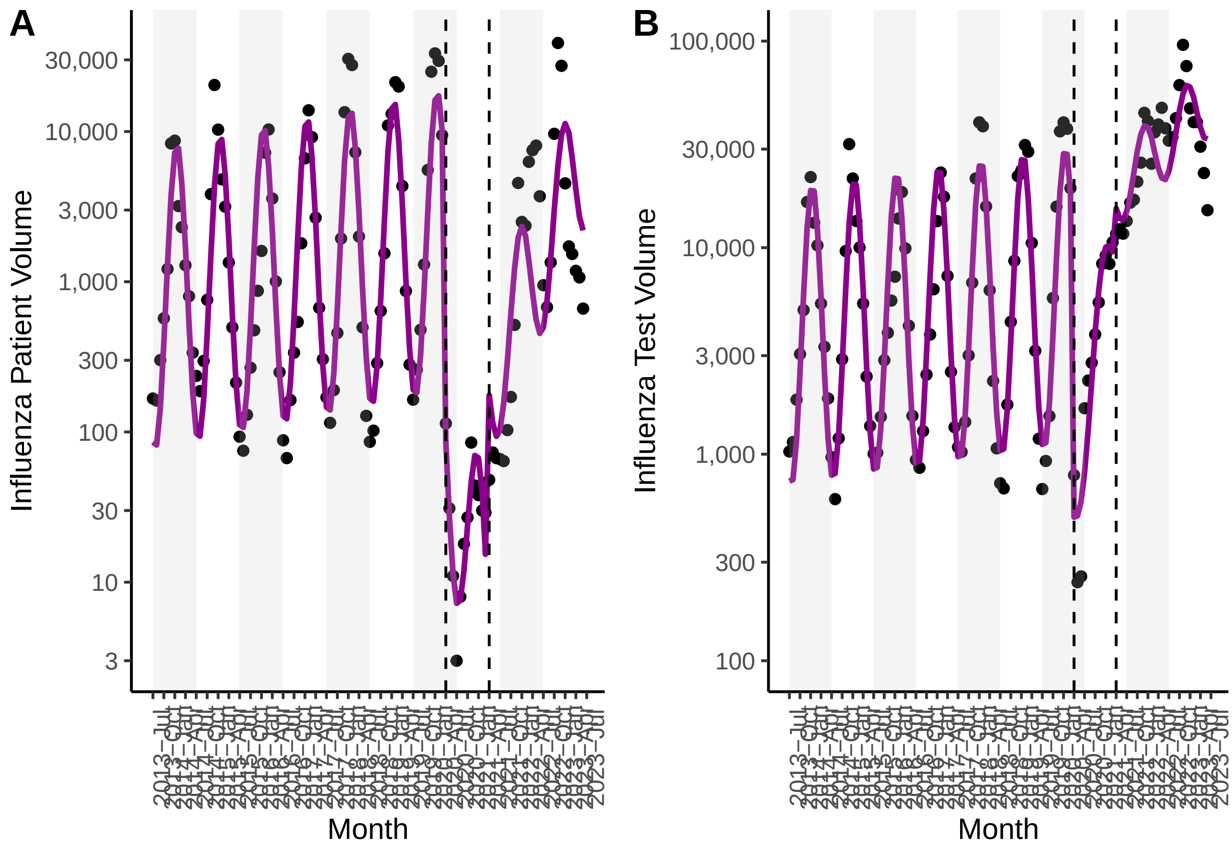
```
## Coefficient: time
## Value: 1.0708
## Exponentiated 95% CI: 1.0252 to 1.1186
## Coefficient: post
## Value: 2.9519
## Exponentiated 95% CI: 2.0418 to 4.2676
## Coefficient: post_slope
## Value: 1.4664
## Exponentiated 95% CI: 1.1637 to 1.8479
## Formula: ~ log(count) time + sc2 + sc_slope + post + post_slope + harmonic_sin_term + harmonic_cos_te
```

```r
# plot influenza test volume
sf2b <- ggplot(flutest_counts, aes(x = month, y = count)) +
  geom_point(color = "black") +
  geom_line(linewidth = 1, aes(x = month, y = pred), color = "darkmagenta") +
  ylab("Influenza Test Volume") +
  xlab("Month") +
  scale_y_continuous(trans = 'log10',
                     breaks = c(100, 300, 1000, 3000, 10000, 30000, 100000), labels = comma) +
  coord_cartesian(ylim = c(100, 100000)) +
```

```
  scale_x_date(labels = scales::date_format("%Y-%b"), breaks = break_dates) +
  geom_rect(
    data = gray_rectangles,
    aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
    fill = "grey80", alpha = 0.2, inherit.aes = FALSE) +
  geom_vline(xintercept = as.Date("2020-04-01"), linetype = "dashed", color = "black") +
  geom_vline(xintercept = as.Date("2021-04-01"), linetype = "dashed", color = "black") +
  theme(axis.text.x = element_text(angle = 90), legend.position = "top",
        legend.justification = "left") + labs(color = "")
sf2b
```




```
rm(flutest, flutest_counts, flutest_model)

sf2 = plot_grid(sf2a, sf2b, nrow = 1, labels = c("A", "B"))
sf2
```




```
ggsave("figs/suppfig2.pdf", plot = sf2, width = 12, height = 5)
```