Figure 3

Brittany A. Petros

2024-02-26

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

```
# 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")]

  # 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.

```
# 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")),
             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.

```
# 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)
        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.

```r
# 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" 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.

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

  # point estimate for the sum
  est_sum <- coef_1 + coef_2
  # standard error for the sum
  se_sum <- sqrt(se_1^2 + se_2^2)
  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_res

  # 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)
    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.

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

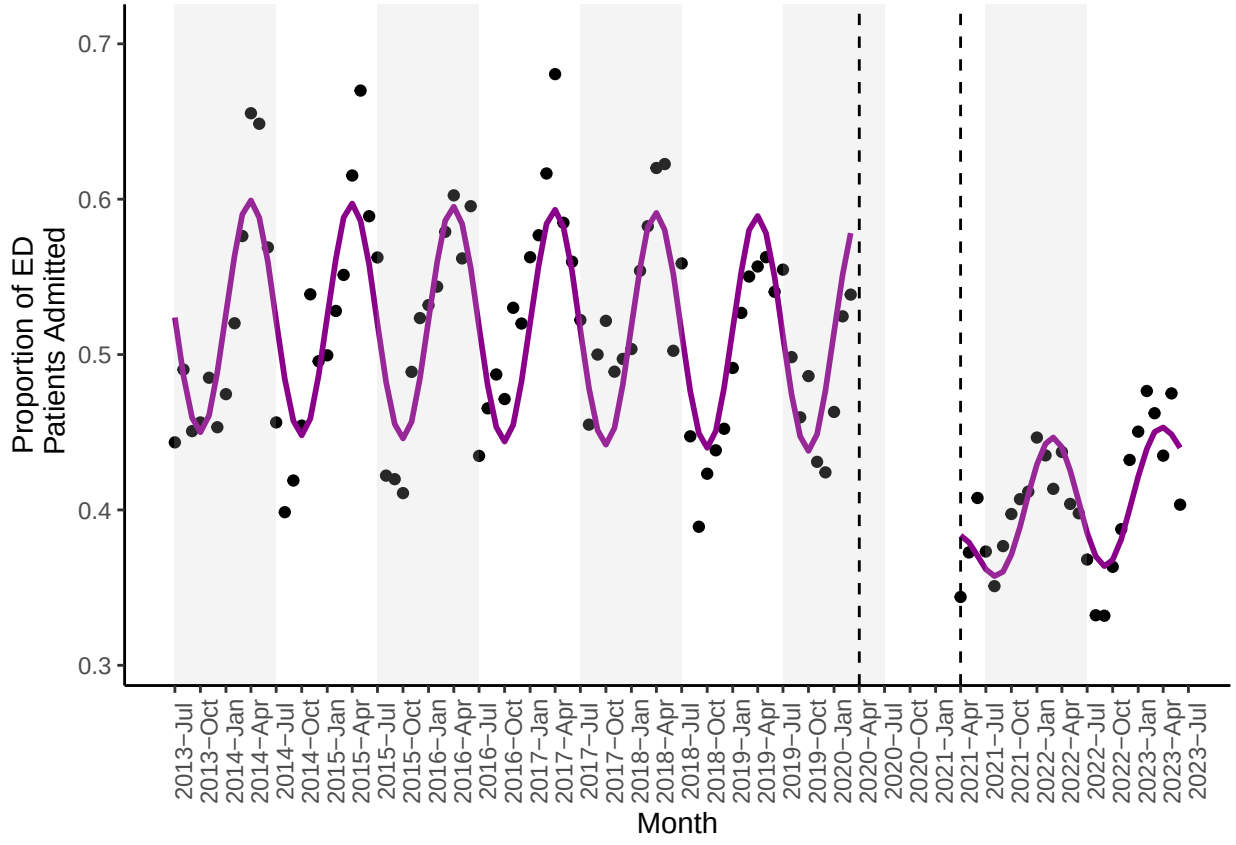
```
# create a df to store gray rectangles
gray_rectangles <- data.frame(
  xmin = as.Date(paste(grey_years, "-07-01", sep = "")),
  xmax = as.Date(paste(grey_years + 1, "-06-30", sep = "")),
  ymin = 1, ymax = Inf)
rm(grey_years)
```

Model the proportion of emergency department patients with a diagnosis of RSV admitted over time (Figure 3A).

```
# filter for cases where pt was seen in ED and has known disposition
ed <- pt %>%
  mutate(ED_Dispo = ifelse(is.na(ED_Dispo), "Unknown", ED_Dispo)) %>%
  filter(ED_entry == 1)

# model proportion of RSV patients admitted from ED
fig3a = generate_and_plot_proportion_model(create_plotting_df(ed, "ED_Dispo",
                                                              "ED_Admission"), "Proportion of ED\n Patients Admitted")

# plot data & report model fits
fig3a$plot
```




```
summary(fig3a$best_model)
```

```
##
```

```
## Call:
## lm(formula = formula, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.14977 -0.02676 -0.00037  0.02887  0.13412
##
## Coefficients:
##                          Estimate Std. Error t value             Pr(>|t|)
## (Intercept)             0.5257539  0.0100963  52.074 < 0.0000000000000002 ***
## time                   -0.0001665  0.0002140  -0.778              0.43824
## sc2                     1.2637396  0.4319845   2.925              0.00420 **
## sc_slope               -0.1506594  0.0464753  -3.242              0.00158 **
## post                   -0.1484711  0.0241703  -6.143  0.00000001409831527 ***
## post_slope              0.0030623  0.0013998   2.188              0.03087 *
## harmonic_1_sin_term    -0.0659284  0.0070313  -9.376  0.00000000000000132 ***
## harmonic_1_cos_term     0.0360309  0.0071098   5.068  0.00000169449925411 ***
## pandemic_1_sin_season   0.2812801  0.0978285   2.875              0.00487 **
## pandemic_1_cos_season   0.0927000  0.0391710   2.367              0.01975 *
## post_1_sin_season      -0.0345301  0.0127536  -2.707              0.00789 **
## pandemic_2_cos_season   0.8914544  0.3354842   2.657              0.00909 **
## post_2_sin_season       0.0220246  0.0156918   1.404              0.16334
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04488 on 107 degrees of freedom
## Multiple R-squared:  0.7666, Adjusted R-squared:  0.7404
## F-statistic: 29.29 on 12 and 107 DF,  p-value: < 0.00000000000000022
```

```
report_prop_coefficients(fig3a$best_model)
```

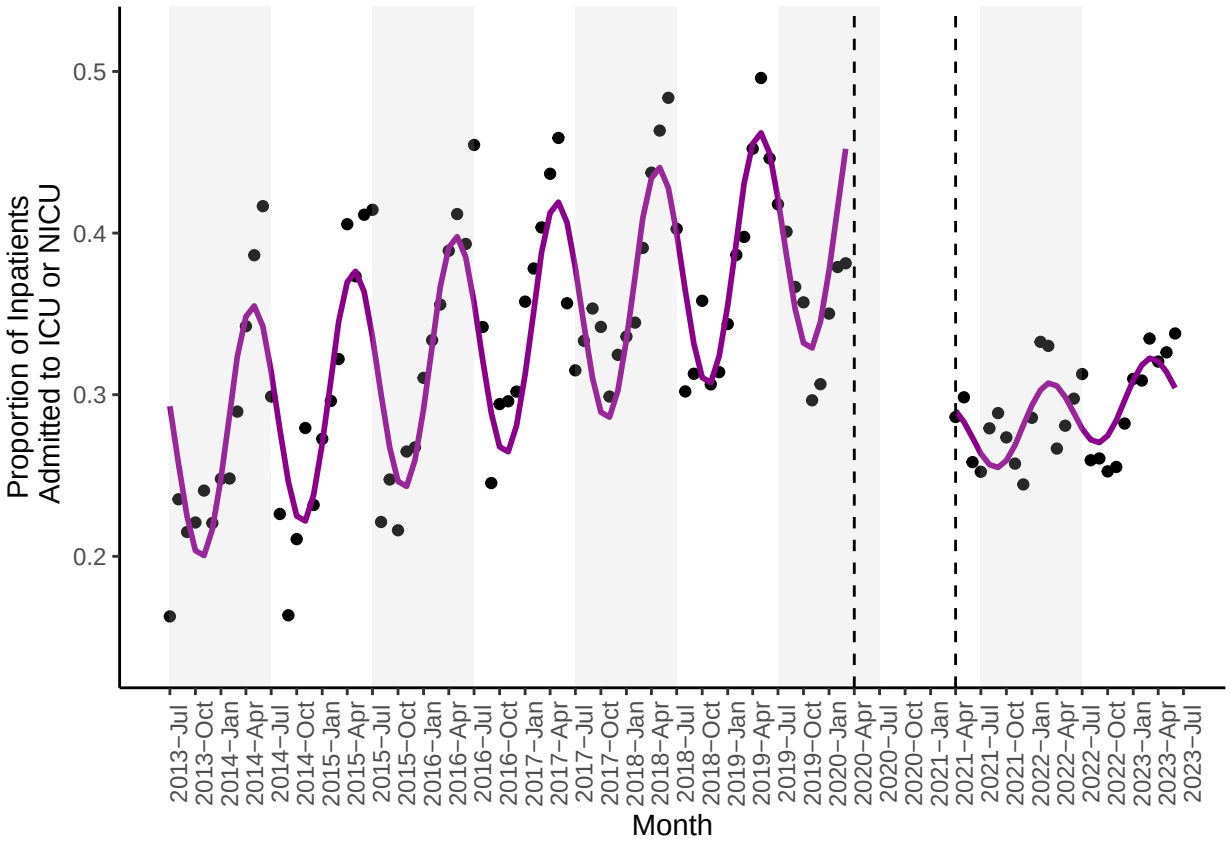
```
## intercept: 0.5258 ( 0.506 - 0.5455 )
## time: -0.002 ( -0.00703 - 0.00304 )
## intercept + post: 0.3773 ( 0.3259 - 0.4286 )
## post_slope 0.03675 ( 0.00383 - 0.06967 )
```

```
rm(ed)
```

Model the proportion of inpatients with a diagnosis of RSV requiring intensive care (Figure 3B).

```
# model proportion of RSV patients admitted to the ICU or NICU
pt$ICU_NICU = ifelse(pt$ICU_Flag == "Y" | pt$NICU_Flag == "Y", "Y", "N")
fig3b = generate_and_plot_proportion_model(create_plotting_df(pt %>%
        filter(Patient_Type_Title == "Inpatient" | Patient_Type_Title == "Obs Unit"),
        "ICU_NICU", "Y"), "Proportion of Inpatients\n Admitted to ICU or NICU")

# plot data and report model fits
fig3b$plot
```




```
summary(fig3b$best_model)
```

```
##
## Call:
## lm(formula = formula, data = data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.132191 -0.018532 -0.001579  0.022598  0.152690
##
## Coefficients:
##                          Estimate Std. Error t value             Pr(>|t|)
## (Intercept)             0.2634159  0.0092436  28.497 < 0.0000000000000002 ***
## time                    0.0017842  0.0001960   9.104  0.00000000000000504 ***
## sc2                    -0.9019558  0.3954980  -2.281               0.0245 *
## sc_slope                0.0894811  0.0425499   2.103               0.0378 *
## post                   -0.1598128  0.0198459  -8.053  0.00000000000115356 ***
## post_slope             -0.0005011  0.0010522  -0.476               0.6349
## harmonic_1_sin_term    -0.0442201  0.0064374  -6.869  0.00000000042941740 ***
## harmonic_1_cos_term     0.0574645  0.0065093   8.828  0.00000000000002123 ***
## pandemic_1_sin_season -0.2347917  0.0895657  -2.621               0.0100 *
## pandemic_1_cos_season  0.0705145  0.0358625   1.966               0.0518 .
## post_1_sin_season     -0.0222373  0.0116356  -1.911               0.0586 .
## pandemic_2_cos_season -0.7517047  0.3071484  -2.447               0.0160 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04109 on 108 degrees of freedom
## Multiple R-squared:  0.782,  Adjusted R-squared:  0.7598
## F-statistic: 35.22 on 11 and 108 DF,  p-value: < 0.00000000000000022
```

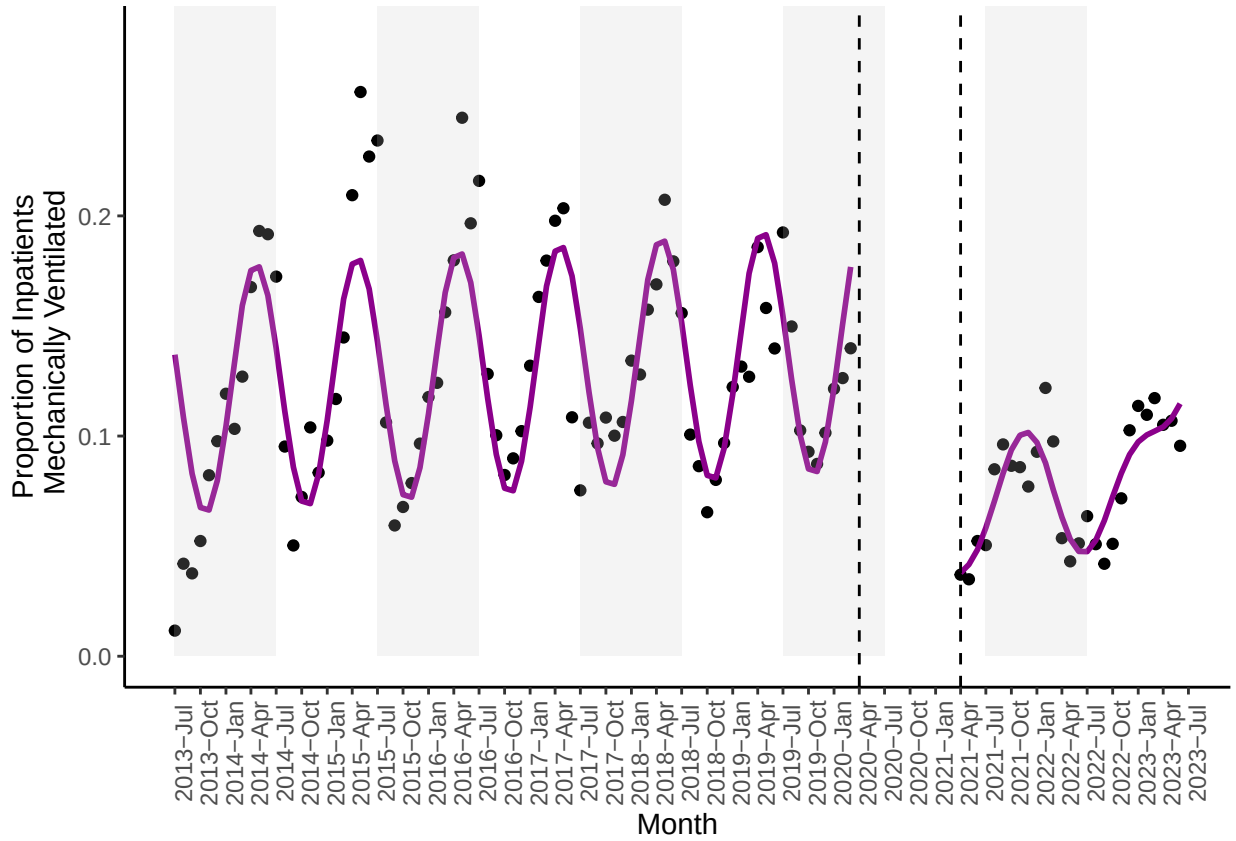
```
report_prop_coefficients(fig3b$best_model)
```

```
## intercept: 0.2634 ( 0.2453 - 0.2815 )
## time: 0.02141 ( 0.0168 - 0.02602 )
## intercept + post: 0.1036 ( 0.0607 - 0.1465 )
## post_slope -0.00601 ( -0.03076 - 0.01873 )
```

Model the proportion of inpatients with a diagnosis of RSV requiring mechanical ventilation (Figure 3C).

```
# model proportion of RSV patients mechanically ventilated
fig3c = generate_and_plot_proportion_model(create_plotting_df(pt %>%
        filter(Patient_Type_Title == "Inpatient" | Patient_Type_Title == "Obs Unit"),
        "Mechanical_Vent_Flag", "Y"), "Proportion of Inpatients\n Mechanically Ventilated")

# plot data and report model fits
fig3c$plot
```



```r
summary(fig3c$best_model)
```

```
##
## Call:
## lm(formula = formula, data = data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.125343 -0.016159  0.001367  0.014919  0.142357
##
## Coefficients:
##                          Estimate Std. Error t value             Pr(>|t|)
## (Intercept)             0.1196838  0.0079602  15.035 < 0.0000000000000002 ***
## time                    0.0002431  0.0001688   1.441              0.15260
## sc2                    -1.8666499  0.2717836  -6.868      0.00000000045939 ***
## sc_slope                0.2307848  0.0308643   7.477      0.00000000002298 ***
## post                   -0.1031011  0.0195035  -5.286      0.00000067385825 ***
## post_slope              0.0025178  0.0011132   2.262              0.02575 *
## harmonic_1_sin_term    -0.0374598  0.0055436  -6.757      0.00000000078522 ***
## harmonic_1_cos_term     0.0413076  0.0056055   7.369      0.00000000003936 ***
## pandemic_1_sin_season -0.3323607  0.0724881  -4.585      0.00001247456685 ***
## post_1_cos_season     -0.0160176  0.0097068  -1.650              0.10188
## pandemic_2_sin_season  0.3016764  0.1044839   2.887              0.00471 **
## pandemic_2_cos_season -1.5553631  0.2028690  -7.667      0.00000000000892 ***
## post_2_sin_season       0.0215676  0.0125396   1.720              0.08836 .
## post_2_cos_season       0.0169830  0.0096063   1.768              0.07995 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03538 on 106 degrees of freedom
## Multiple R-squared:  0.8397, Adjusted R-squared:  0.8201
## F-statistic: 42.72 on 13 and 106 DF,  p-value: < 0.00000000000000022
```

```r
report_prop_coefficients(fig3c$best_model)
```

```
## intercept: 0.1197 ( 0.1041 - 0.1353 )
## time: 0.00292 ( -0.00105 - 0.00689 )
## intercept + post: 0.0166 ( -0.0247 - 0.0579 )
## post_slope 0.03021 ( 0.00403 - 0.0564 )
```

Model the volume of emergency department patients with a diagnosis of RSV admitted over time (Figure 3D).

```r
# filter for patients with ED visit
ed_admit <- pt %>%
  mutate(ED_Dispo = ifelse(is.na(ED_Dispo), "Unknown", ED_Dispo)) %>%
  filter(ED_entry == 1) %>%
  mutate(month = floor_date(Date, "month")) %>%
  group_by(month) %>%
  # count number of admissions
  summarise(numerator = sum(ED_Dispo == "ED_Admission"))
```

```r
# model ED admission volume
ed_vol_model = find_best_model(data.frame("month" = ed_admit$month, "count" = ed_admit$numerator))
summary(ed_vol_model)
```

```
##
## Call:
## lm(formula = formula, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.28405 -0.22563 -0.01697  0.22566  0.93697
##
## Coefficients:
##                       Estimate Std. Error t value             Pr(>|t|)
## (Intercept)           5.839026   0.092262  63.287 < 0.0000000000000002 ***
## time                  0.007529   0.001956   3.849             0.000201 ***
## sc2                  -3.125375   0.393952  -7.933     0.00000000000212 ***
## sc_slope             -0.084295   0.056137  -1.502             0.136115
## post                  0.651809   0.199858   3.261             0.001484 **
## post_slope           -0.014464   0.010575  -1.368             0.174225
## harmonic_sin_term    -0.982143   0.064253 -15.286 < 0.0000000000000002 ***
## harmonic_cos_term    -1.614969   0.064970 -24.857 < 0.0000000000000002 ***
## pandemic_sin_season -0.909531   0.176568  -5.151     0.00000117427955 ***
## pandemic_cos_season -0.504363   0.268081  -1.881             0.062613 .
## post_sin_season       1.136363   0.116490   9.755 < 0.0000000000000002 ***
## post_cos_season      -0.601791   0.111381  -5.403     0.00000039420366 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4101 on 108 degrees of freedom
## Multiple R-squared:  0.9477, Adjusted R-squared:  0.9424
## F-statistic: 178.1 on 11 and 108 DF,  p-value: < 0.00000000000000022
```

```r
process_lm_output(ed_vol_model)
```

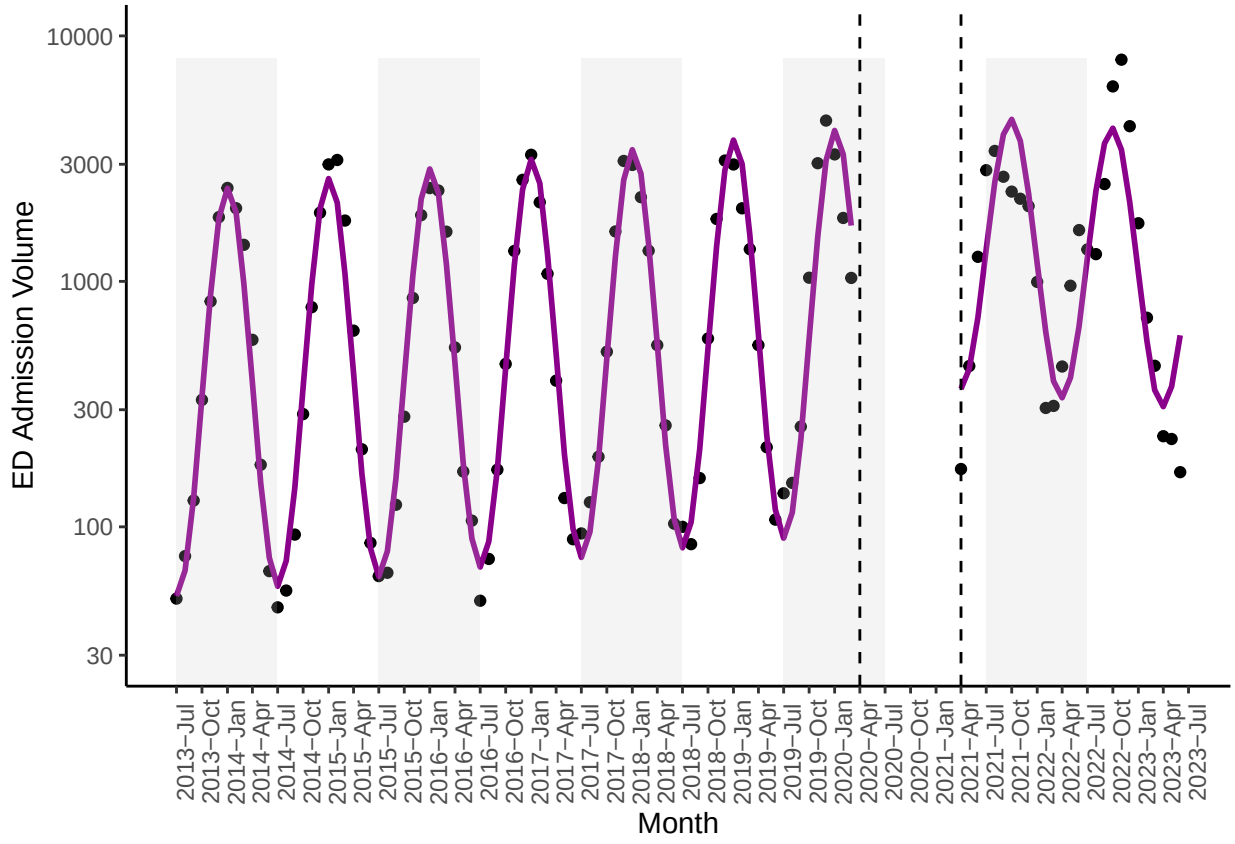
```
## Coefficient: time
## Value: 1.0946
## Exponentiated 95% CI: 1.0448 to 1.1467
## Coefficient: post
## Value: 1.919
## Exponentiated 95% CI: 1.2913 to 2.8518
## Coefficient: post_slope
## Value: 0.8407
## Exponentiated 95% CI: 0.6537 to 1.0811
## Formula: ~ log(count) time + sc2 + sc_slope + post + post_slope + harmonic_sin_term + harmonic_cos_t
```

```r
ed_admit$pred_vol = exp(ed_vol_model$fitted.values)

# plot the data
fig3d <- ggplot(ed_admit, aes(x = month, y = numerator)) +
  geom_point(data = ed_admit %>%filter(month < as.Date("2020-04-01") | month >= as.Date("2021-04-01")),
             aes(x = month, y = numerator), color = "black") +
```

```
  geom_line(data = ed_admit %>% filter(month < as.Date("2020-04-01")),
            aes(x = month, y = pred_vol), color = "darkmagenta", linewidth = 1) +
  geom_line(data = ed_admit %>% filter(month >= as.Date("2021-04-01")),
            aes(x = month, y = pred_vol), color = "darkmagenta", linewidth = 1) +
  ylab("ED Admission Volume") +
  xlab("Month") +
  coord_cartesian(ylim = c(30, 10000)) +
  scale_y_log10(breaks = c(30, 100, 300, 1000, 3000, 10000)) +
  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 = max(ed_admit$numerator) + 100),
    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 = "")
fig3d
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```



Model the volume of inpatients with a diagnosis of RSV requiring intensive care (Figure 3E).

```r
# filter for inpatients
icu_admit <- pt %>%
  filter(Patient_Type_Title != "ED Visit") %>%
  mutate(month = floor_date(Date, "month")) %>%
  group_by(month) %>%
  # count number of ICU admissions
  summarise(numerator = sum(ICU_Flag == "Y" | NICU_Flag == "Y"))

# model ICU admission volume
icu_vol_model = find_best_model(data.frame("month" = icu_admit$month, "count" = icu_admit$numerator))
summary(icu_vol_model)
```

```
##
## Call:
## lm(formula = formula, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1081 -0.2638  0.0273  0.2518  0.8593
##
## Coefficients:
##                      Estimate Std. Error t value             Pr(>|t|)
## (Intercept)          4.998914   0.089382  55.927 < 0.0000000000000002 ***
## time                 0.012879   0.001895   6.797  0.00000000059336331 ***
## sc2                 -2.729765   0.263681 -10.353 < 0.0000000000000002 ***
## sc_slope            -0.130875   0.033995  -3.850               0.0002 ***
## post                 0.159720   0.193620   0.825               0.4112
## post_slope          -0.016222   0.010245  -1.583               0.1162
## harmonic_sin_term   -1.167653   0.062247 -18.758 < 0.0000000000000002 ***
## harmonic_cos_term   -1.419142   0.062942 -22.547 < 0.0000000000000002 ***
## pandemic_sin_season -0.895783   0.165706  -5.406  0.00000038384220479 ***
## post_sin_season      1.039067   0.112854   9.207  0.00000000000000275 ***
## post_cos_season     -0.562956   0.107904  -5.217  0.00000087340542245 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3973 on 109 degrees of freedom
## Multiple R-squared:  0.9447, Adjusted R-squared:  0.9396
## F-statistic: 186.2 on 10 and 109 DF,  p-value: < 0.00000000000000022
```

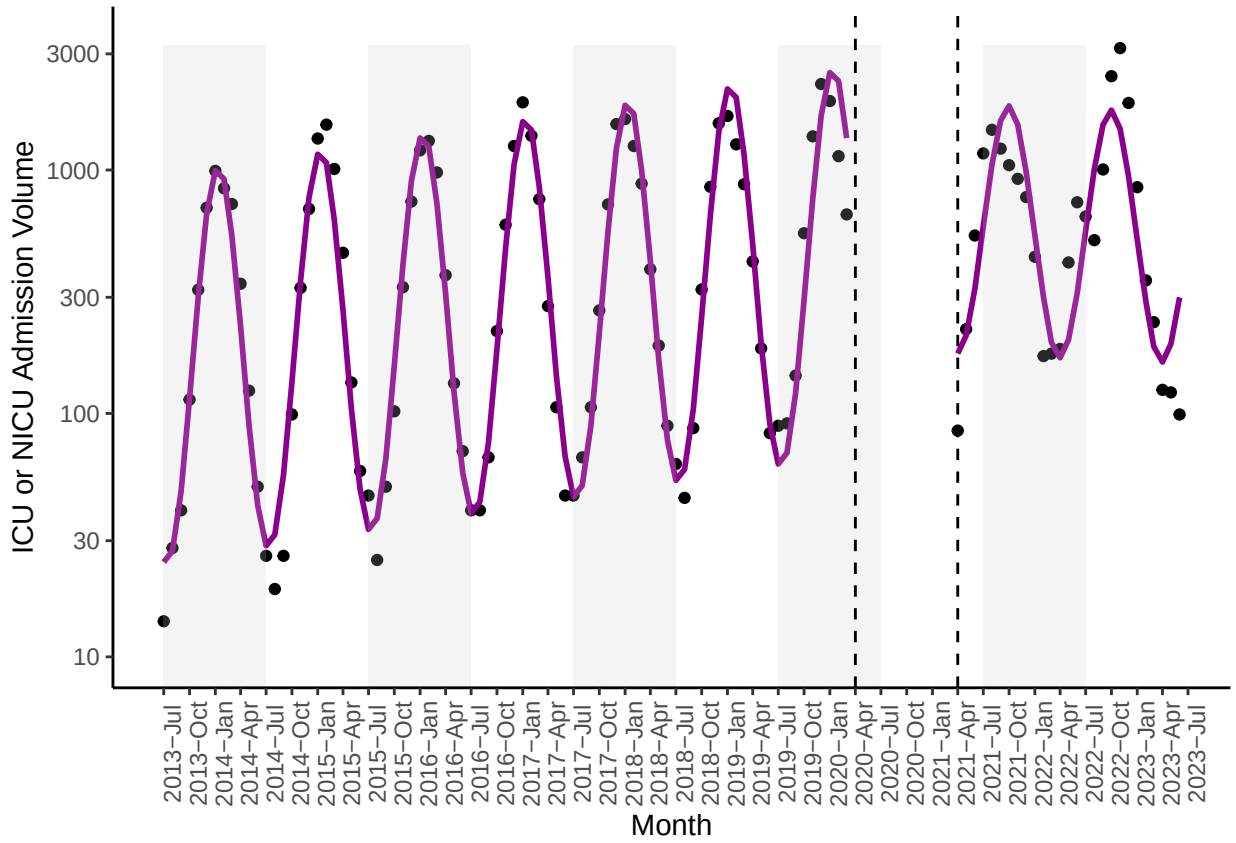
```r
process_lm_output(icu_vol_model)
```

```
## Coefficient: time
## Value: 1.1671
## Exponentiated 95% CI: 1.1157 to 1.2209
## Coefficient: post
## Value: 1.1732
## Exponentiated 95% CI: 0.7993 to 1.722
## Coefficient: post_slope
## Value: 0.8231
## Exponentiated 95% CI: 0.6451 to 1.0502
## Formula: ~ log(count) time + sc2 + sc_slope + post + post_slope + harmonic_sin_term + harmonic_cos_t
```

```r
icu_admit$pred_vol = exp(icu_vol_model$fitted.values)

# plot the data
fig3e <- ggplot(icu_admit, aes(x = month, y = numerator)) +
  geom_point(data = icu_admit %>%filter(month < as.Date("2020-04-01") | month >= as.Date("2021-04-01")),
             aes(x = month, y = numerator), color = "black") +
  geom_line(data = icu_admit %>% filter(month < as.Date("2020-04-01")),
            aes(x = month, y = pred_vol), color = "darkmagenta", linewidth = 1) +
  geom_line(data = icu_admit %>% filter(month >= as.Date("2021-04-01")),
            aes(x = month, y = pred_vol), color = "darkmagenta", linewidth = 1) +
  ylab("ICU or NICU Admission Volume") +
  xlab("Month") +
  coord_cartesian(ylim = c(10, 3500)) +
  scale_y_log10(breaks = c(10, 30, 100, 300, 1000, 3000)) +
  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 = max(icu_admit$numerator) + 100),
    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 = "")
fig3e
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```



Model the volume of inpatients with a diagnosis of RSV requiring mechanical ventilation (Figure 3F).

```
# filter for inpatients
mv_count <- pt %>%
  filter(Patient_Type_Title != "ED Visit") %>%
  mutate(month = floor_date(Date, "month")) %>%
  group_by(month) %>%
    # count number of patients mech ventilated
  summarise(numerator = sum(Mechanical_Vent_Flag == "Y"))

# model mech ventilation volume
mv_vol_model = find_best_model(data.frame("month" = mv_count$month, "count" = mv_count$numerator + 1)) 
summary(mv_vol_model)
```

```
##
## Call:
## lm(formula = formula, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6320 -0.2637  0.0711  0.2885  0.9379
##
## Coefficients:
##                      Estimate Std. Error t value             Pr(>|t|)
## (Intercept)          4.099576   0.105513  38.854 < 0.0000000000000002 ***
## time                 0.010709   0.002237   4.788      0.0000053965204 ***
```

```
## sc2                 -2.087107   0.450534  -4.633      0.0000101385847 ***
## sc_slope            -0.166124   0.064199  -2.588              0.01099 *
## post                -0.266402   0.228564  -1.166              0.24637
## post_slope          -0.000845   0.012094  -0.070              0.94442
## harmonic_sin_term   -1.338914   0.073482 -18.221 < 0.0000000000000002 ***
## harmonic_cos_term   -1.288257   0.074302 -17.338 < 0.0000000000000002 ***
## pandemic_sin_season -0.615524   0.201928  -3.048              0.00289 **
## pandemic_cos_season  0.501871   0.306585   1.637              0.10455
## post_sin_season      0.991268   0.133221   7.441      0.0000000000255 ***
## post_cos_season     -0.824336   0.127378  -6.472      0.0000000029240 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.469 on 108 degrees of freedom
## Multiple R-squared:  0.9188, Adjusted R-squared:  0.9106
## F-statistic: 111.2 on 11 and 108 DF,  p-value: < 0.00000000000000022
```

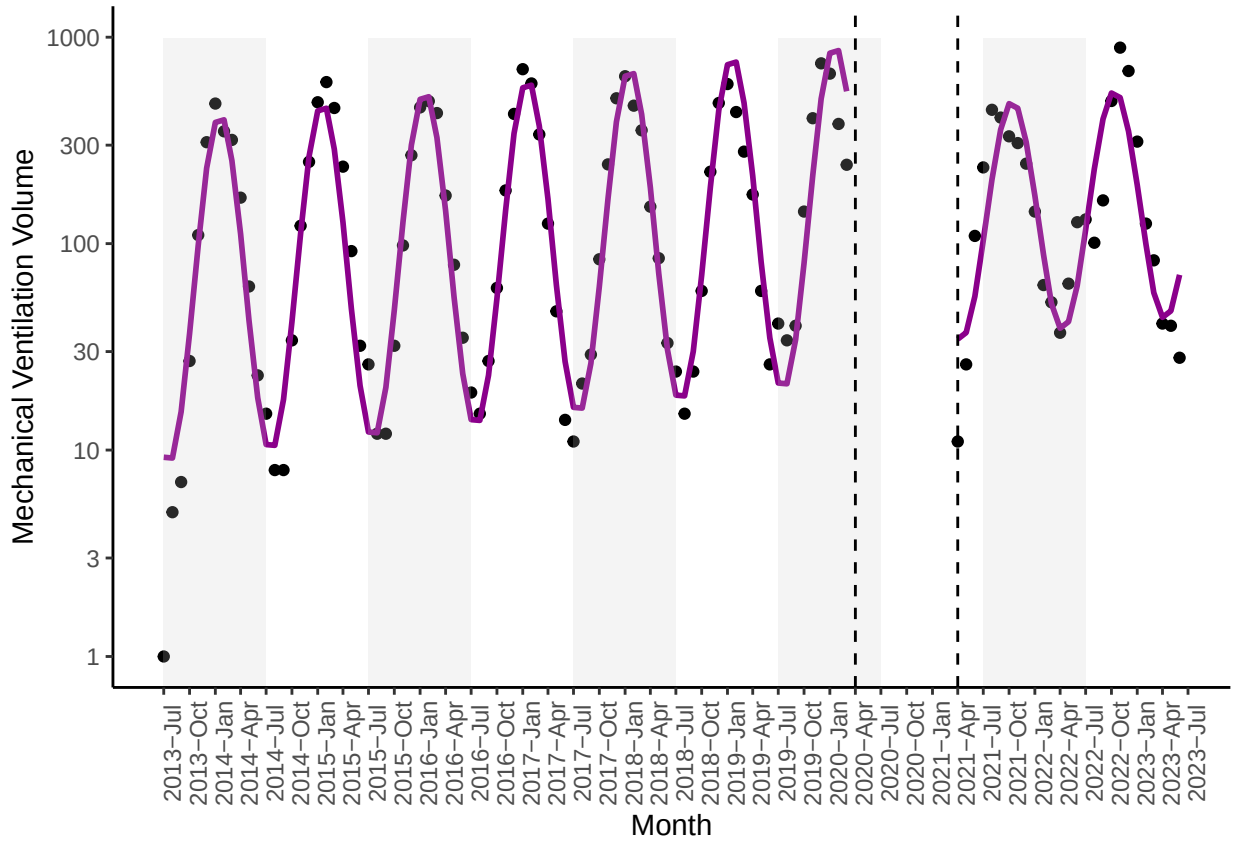
```
process_lm_output(mv_vol_model)
```

```
## Coefficient: time
## Value: 1.1371
## Exponentiated 95% CI: 1.0782 to 1.1993
## Coefficient: post
## Value: 0.7661
## Exponentiated 95% CI: 0.487 to 1.2052
## Coefficient: post_slope
## Value: 0.9899
## Exponentiated 95% CI: 0.7425 to 1.3198
## Formula: ~ log(count) time + sc2 + sc_slope + post + post_slope + harmonic_sin_term + harmonic_cos_t
```

```
mv_count$pred_vol = exp(mv_vol_model$fitted.values) - 1 # re-adjust

fig3f <- ggplot(mv_count, aes(x = month, y = numerator)) +
  geom_point(data = mv_count %>%filter(month < as.Date("2020-04-01") | month >= as.Date("2021-04-01")),
             aes(x = month, y = numerator), color = "black") +
  geom_line(data = mv_count %>% filter(month < as.Date("2020-04-01")),
            aes(x = month, y = pred_vol), color = "darkmagenta", linewidth = 1) +
  geom_line(data = mv_count %>% filter(month >= as.Date("2021-04-01")),
            aes(x = month, y = pred_vol), color = "darkmagenta", linewidth = 1) +
  ylab("Mechanical Ventilation Volume") +
  xlab("Month") +
  coord_cartesian(ylim = c(1, 1000)) +
  scale_y_log10(breaks = c(1, 3, 10, 30, 100, 300, 1000)) +
  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 = max(mv_count$numerator) + 100),
    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 = "")
fig3f
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

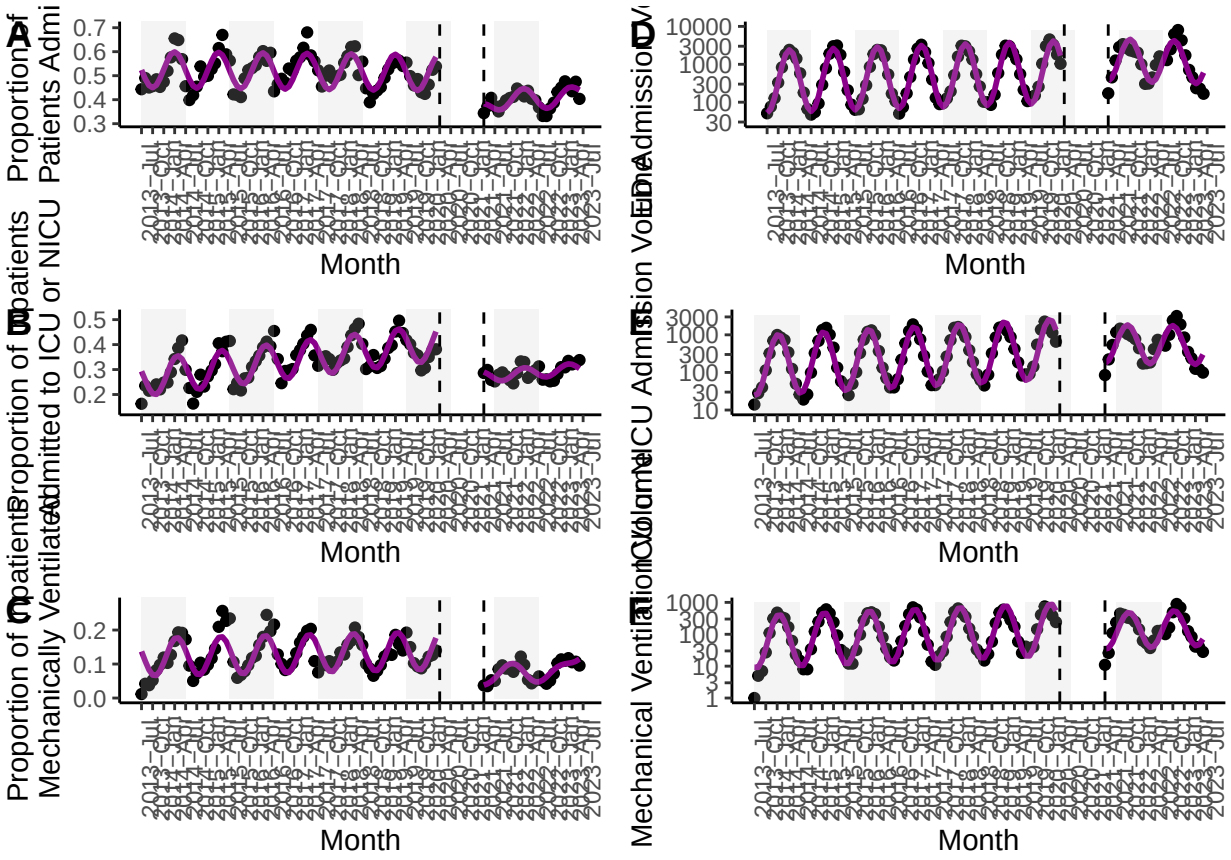


Generate Figure 3.

```
fig3 = plot_grid(fig3a$plot, fig3d, fig3b$plot, fig3e, fig3c$plot, fig3f, ncol = 2,
                 labels = c("A", "D", "B", "E", "C", "F"))
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
## Transformation introduced infinite values in continuous y-axis
## Transformation introduced infinite values in continuous y-axis
```

```
fig3
```



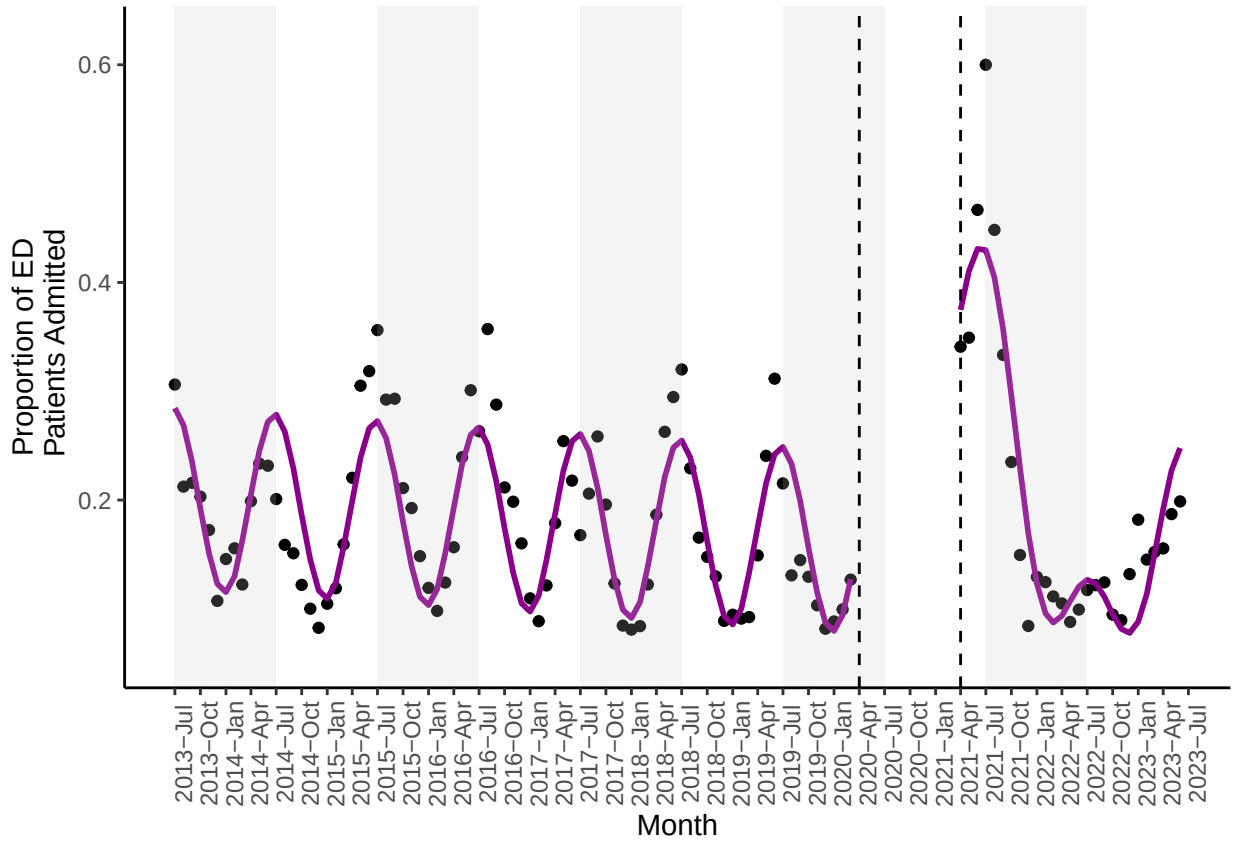

```
ggsave("figs/fig3.pdf", plot = fig3, width = 12, height = 12)
rm(ed_admit, icu_admit, mv_count, fig3)
```

Model the proportion of emergency department patients with a diagnosis of influenza admitted over time (Supplementary Figure 4A).

```
# count influenza patients admitted to the ED
flu_ed = flu %>%
  mutate(numerator = flu$ED_admit, denominator = flu$ED_admit+flu$ED_dc)

# model proportion of influenza patients admitted from ED
sf4a = generate_and_plot_proportion_model(create_plotting_df(flu_ed, monthly = TRUE),
                                          "Proportion of ED\n Patients Admitted")

# plot data & report model fits
sf4a$plot
```




```
summary(sf4a$best_model)
```

```
##
## Call:
## lm(formula = formula, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.47105 -0.02725 -0.00254  0.03059  0.29554
##
## Coefficients:
##                      Estimate Std. Error t value             Pr(>|t|)
## (Intercept)          0.2019090  0.0174442  11.575 < 0.0000000000000002 ***
## time                -0.0004957  0.0003698  -1.340             0.182943
## sc2                  0.3756599  0.0506995   7.410      0.0000000000287 ***
## sc_slope            -0.0160973  0.0064944  -2.479             0.014722 *
## post                 0.1352645  0.0382509   3.536             0.000597 ***
## post_slope          -0.0071293  0.0020205  -3.528             0.000613 ***
## harmonic_1_sin_term  0.0346006  0.0121485   2.848             0.005257 **
## harmonic_1_cos_term  0.0759762  0.0122841   6.185      0.0000000110809 ***
## post_1_sin_season    0.0347577  0.0221380   1.570             0.119304
## post_1_cos_season    0.0531457  0.0212203   2.504             0.013744 *
## post_2_cos_season    0.1094551  0.0209379   5.228      0.0000008349609 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.07754 on 109 degrees of freedom
## Multiple R-squared:  0.6805, Adjusted R-squared:  0.6512
## F-statistic: 23.22 on 10 and 109 DF,  p-value: < 0.00000000000000022
```

```
report_prop_coefficients(sf4a$best_model)
```

```
## intercept: 0.2019 ( 0.1677 - 0.2361 )
## time: -0.00595 ( -0.01465 - 0.00275 )
## intercept + post: 0.3372 ( 0.2548 - 0.4196 )
## post_slope -0.08555 ( -0.13307 - -0.03803 )
```

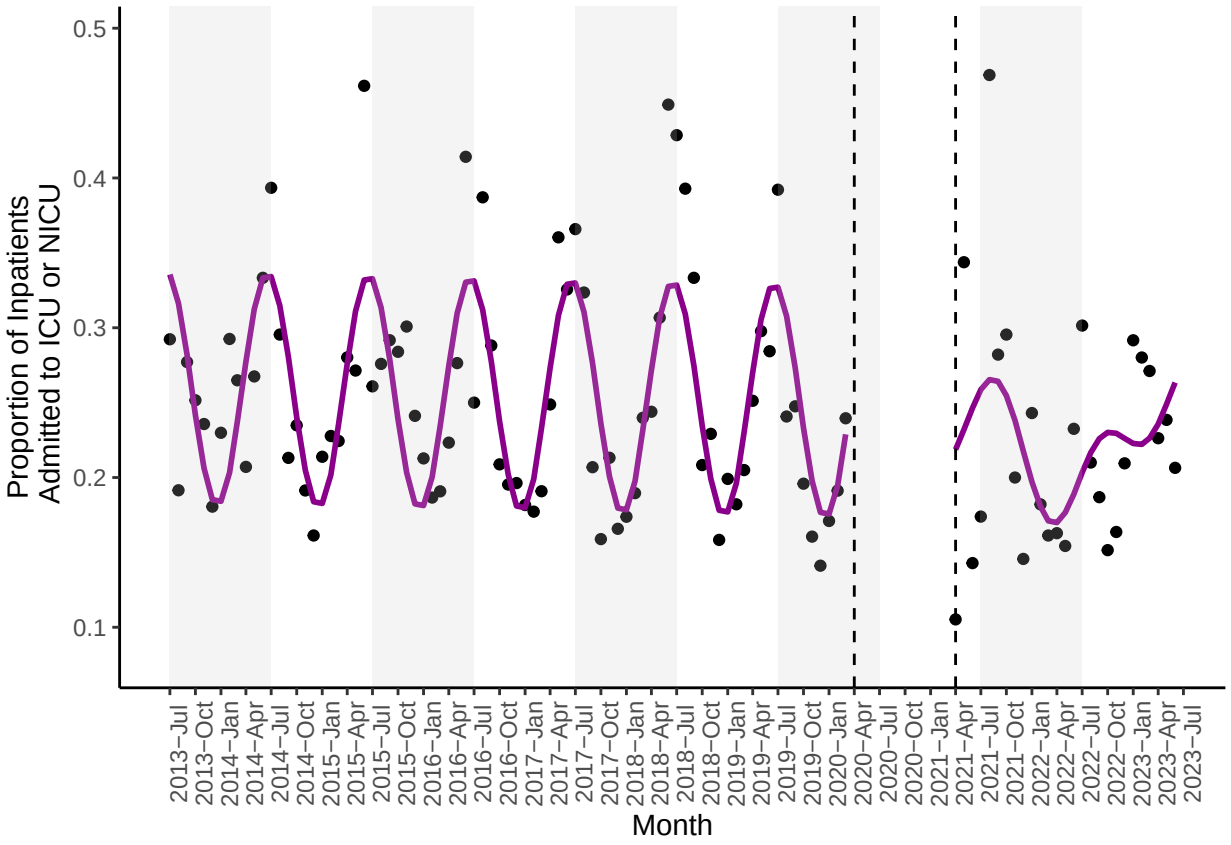
```
rm(flu_ed)
```

Model the proportion of inpatients with a diagnosis of influenza requiring intensive care (Supplementary Figure 4B).

```
# count influenza inpatients admitted to the ICU or NICU
flu_icu = flu %>%
  mutate(numerator = flu$ICU + flu$NICU, denominator = flu$IP)

# model proportion of influenza inpatients receiving intensive care
sf4b = generate_and_plot_proportion_model(create_plotting_df(flu_icu,
        monthly = TRUE), "Proportion of Inpatients\n Admitted to ICU or NICU")

# plot data & report model fits
sf4b$plot
```




```
summary(sf4b$best_model)
```

```
##
## Call:
## lm(formula = formula, data = data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.138099 -0.037242 -0.005986  0.031240  0.236980
##
## Coefficients:
##                          Estimate Std. Error t value             Pr(>|t|)
## (Intercept)             0.2603176  0.0140996  18.463 < 0.0000000000000002 ***
## time                   -0.0001180  0.0002989  -0.395              0.69368
## sc2                     0.3137917  0.0575413   5.453       0.000000311384 ***
## sc_slope               -0.0273482  0.0081348  -3.362              0.00107 **
## post                   -0.0380658  0.0307254  -1.239              0.21804
## post_slope              0.0008254  0.0016255   0.508              0.61262
## harmonic_1_sin_term     0.0219131  0.0098192   2.232              0.02768 *
## harmonic_1_cos_term     0.0743820  0.0099289   7.492       0.000000000019 ***
## pandemic_1_cos_season   0.1752674  0.0396870   4.416       0.000023763388 ***
## post_1_sin_season       0.0253287  0.0178565   1.418              0.15891
## post_2_cos_season       0.0330572  0.0167948   1.968              0.05157 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.06267 on 109 degrees of freedom
## Multiple R-squared:  0.7012, Adjusted R-squared:  0.6738
## F-statistic: 25.58 on 10 and 109 DF,  p-value: < 0.00000000000000022
```

```
report_prop_coefficients(sf4b$best_model)
```

```
## intercept: 0.2603 ( 0.2327 - 0.288 )
## time: -0.00142 ( -0.00845 - 0.00561 )
## intercept + post: 0.2223 ( 0.156 - 0.2885 )
## post_slope 0.0099 ( -0.02833 - 0.04814 )
```

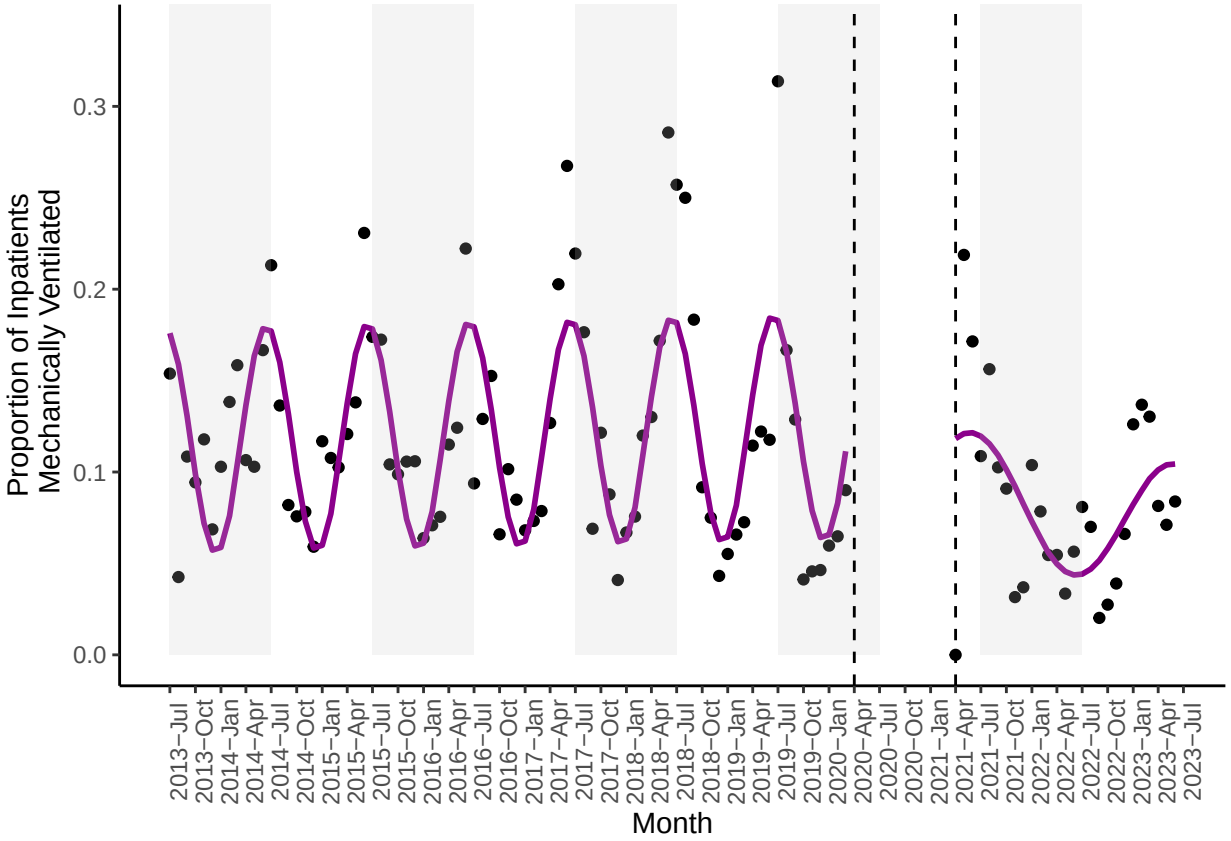
```
rm(flu_icu)
```

Model the proportion of inpatients with a diagnosis of influenza requiring mechanical ventilation (Supplementary Figure 4C).

```
# count influenza inpatients mech ventilated
flu_vent = flu %>%
  mutate(numerator = flu$mech_vent, denominator = flu$IP)

# model proportion of influenza inpatients receiving mech ventilation
sf4c = generate_and_plot_proportion_model(create_plotting_df(flu_vent,
        monthly = TRUE), "Proportion of Inpatients\n Mechanically Ventilated")

# plot data & report model fits
sf4c$plot
```




```
summary(sf4c$best_model)
```

```
##
## Call:
## lm(formula = formula, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.13415 -0.02388 -0.00442  0.03060  0.13084
##
## Coefficients:
##                           Estimate  Std. Error t value             Pr(>|t|)
## (Intercept)            0.11697980  0.01050514  11.135 < 0.0000000000000002 ***
## time                   0.00009605  0.00022271   0.431             0.667122
## sc2                    0.42129417  0.06587187   6.396     0.00000000419549 ***
## sc_slope              -0.09941622  0.02920333  -3.404             0.000932 ***
## post                  -0.03691026  0.02271333  -1.625             0.107067
## post_slope            -0.00080850  0.00118667  -0.681             0.497131
## harmonic_1_sin_term    0.01338648  0.00731598   1.830             0.070044 .
## harmonic_1_cos_term    0.06027369  0.00739765   8.148      0.00000000000071 ***
## pandemic_1_sin_season -0.22682835  0.08170747  -2.776             0.006486 **
## pandemic_1_cos_season  0.30491522  0.06130945   4.973     0.00000249460495 ***
## pandemic_2_sin_season -0.69888148  0.27043866  -2.584             0.011094 *
## post_2_cos_season      0.03462750  0.01243745   2.784             0.006338 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04669 on 108 degrees of freedom
## Multiple R-squared:  0.7909, Adjusted R-squared:  0.7696
## F-statistic: 37.13 on 11 and 108 DF,  p-value: < 0.00000000000000022
```

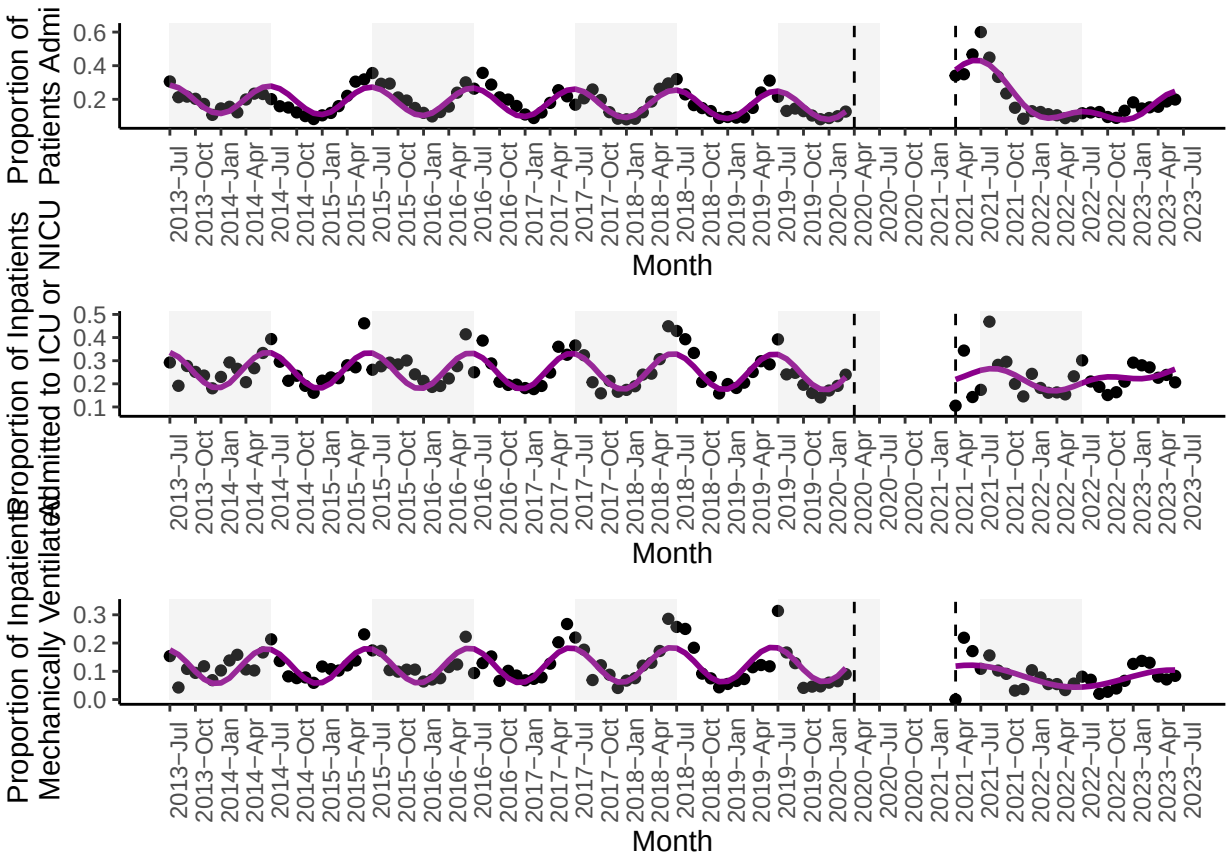
```
report_prop_coefficients(sf4c$best_model)
```

```
## intercept: 0.117 ( 0.0964 - 0.1376 )
## time: 0.00115 ( -0.00409 - 0.00639 )
## intercept + post: 0.0801 ( 0.031 - 0.1291 )
## post_slope -0.0097 ( -0.03761 - 0.01821 )
```

```
rm(flu_vent)
```

Generate Supplementary Figure 4.

```
sf4 <- plot_grid(sf4a$plot, sf4b$plot, sf4c$plot, ncol = 1)
sf4
```




```
ggsave("figs/suppfig4.pdf", plot = sf4, width = 6, height = 9)
```