STATS-506 HW5

Zekai Xu

2025-11-09

Github Repo

Question 1

Part a

```
# -----Define class `waldCI`-----
setClass(
  "waldCI",
  slots = c(mu = "numeric", se = "numeric")
)

# -----Class Validator-----
setValidity(
  "waldCI",
  function(object){
    # Collect error message
    err <- character()

    # Validity Checking
    ## Length
    if (length(object@mu) != 1L)
      err <- c(err, "`mu` must have length 1!")
    if (length(object@se) != 1L)
      err <- c(err, "`se` must have length 1!")
    ## Value boundary
    if (!is.finite(object@mu))
      err <- c(err, "`mu` must be finite!")
```

```
    if (!is.finite(object@se))
      err <- c(err, "`se` must be finite!")
    if (object@se < 0)
      err <- c(err, "`se` must be non-negative!")


    # Output err if not-none
    if (length(err))
      return(err)
    else
      return(TRUE)
  }
)
```

```
Class "waldCI" [in ".GlobalEnv"]

Slots:

Name:       mu      se
Class: numeric numeric
```

```
# -----Customized Constructor-----
## Helper function to compute `z` based on `level`
compute_z <- function(level)
{
  return(qnorm(1 - (1 - level) / 2))
}
waldCI <- function(mean = NULL, sterr = NULL, lower = NULL, upper = NULL, level = 0.95)
{
  # Type 1: `mean` and `stderr` are provided
  if (!is.null(mean) && !is.null(sterr))
    return(new("waldCI", mu = as.numeric(mean), se = as.numeric(sterr)))

  # Type 2: `lower`, `upper` and `level` are provided
  if (!is.null(lower) && !is.null(upper) && !is.null(level))
  {
    # Value check
    if (!is.finite(lower) || !is.finite(upper))
        stop("Need `lower` and `upper` be finite number.")
    if (lower >= upper)
      stop("Need `lower` < `upper`")
    if (level <= 0 || level >= 1)
```

```
      stop("Need `level` between 0 and 1(exclusive)")

    z <- compute_z(level)
    mu <- (lower + upper) / 2
    se <- ((upper - lower) / 2) / z
    return(new("waldCI", mu = mu, se = se))
  }

  stop("Need either (mean, sterr) or (lower, upper, level).")
}
```

```
# -----`show` method-----
show_waldCI <- function(object, level, digits)
{
  z <- compute_z(level)
  lower <- object@mu - z * object@se
  upper <- object@mu + z * object@se

  f <- function(x, d = digits) sprintf(paste0("%.", d, "f"), x)

  cat(
    "waldCI: \n mean = ", f(object@mu), ", \n se = ", f(object@se), "\n",
    " level = ", sprintf("%.3f", level), "  (z = ", sprintf("%.3f", z), ")\n",
    " CI: [", f(lower), ", ", f(upper), "]\n",
    sep = ""
  )
  invisible(object)
}

show_orig <- show
show <- function(object, level = 0.95, digits = 3L)
{
  if (is(object, "waldCI"))
    show_waldCI(object, level, digits)
  else
    show_orig(object)

  invisible(object)
}
```

```r
# -----Accessors-----
## Accessor Generic
setGeneric(
  "lb",
  function(object, level = 0.95) standardGeneric("lb")
)
```

```
[1] "lb"
```

```r
setGeneric(
  "ub",
  function(object, level = 0.95) standardGeneric("ub")
)
```

```
[1] "ub"
```

```r
setGeneric(
  "sterr",
  function(object) standardGeneric("sterr")
)
```

```
[1] "sterr"
```

```r
## Accessor Implementation
setMethod(
  "lb",
  signature("waldCI"),
  function(object, level = 0.95){
    z <- compute_z(level)
    return(object@mu - z * object@se)
  }
)
setMethod(
  "ub",
  signature("waldCI"),
  function(object, level = 0.95){
    z <- compute_z(level)
    return(object@mu + z * object@se)
  }
)
```

```
setMethod(
  "mean",
  signature("waldCI"),
  function(x, ...) {x@mu}
)
setMethod(
  "sterr",
  signature("waldCI"),
  function(object) {object@se}
)
```

```
#-----Setters-----
## Setters Generic
setGeneric(
  "lb<-",
  function(object, level = 0.95, value) standardGeneric("lb<-")
)
```

```
[1] "lb<-"
```

```
setGeneric(
  "ub<-",
  function(object, level = 0.95, value) standardGeneric("ub<-")
)
```

```
[1] "ub<-"
```

```
setGeneric(
  "mean<-",
  function(object, value) standardGeneric("mean<-")
)
```

```
[1] "mean<-"
```

```
setGeneric(
  "sterr<-",
  function(object, value) standardGeneric("sterr<-")
)
```

```
[1] "sterr<-"
```

```r
## Setters Implementation
setReplaceMethod(
  "lb",
  signature(object = "waldCI", value = "numeric"),
  function(object, level = 0.95, value)
  {
    z <- compute_z(level)
    object@mu <- as.numeric(value) + z * object@se
    validObject(object)
    return(object)
  }
)
setReplaceMethod(
  "ub",
  signature(object = "waldCI", value = "numeric"),
  function(object, level = 0.95, value)
  {
    z <- compute_z(level)
    object@mu <- as.numeric(value) - z * object@se
    validObject(object)
    return(object)
  }
)
setReplaceMethod(
  "mean",
  signature(object = "waldCI", value = "numeric"),
  function(object, value)
  {
    object@mu <- value
    validObject(object)
    return(object)
  }
)
setReplaceMethod(
  "sterr",
  signature(object = "waldCI", value = "numeric"),
  function(object, value)
  {
    object@se <- value
    validObject(object)
    return(object)
  }
```

```
)
```

```
#-----Contains Method-----
setGeneric(
  "contains",
  function(object, value, level = 0.95) standardGeneric("contains")
)
```

```
Creating a new generic function for 'contains' in the global environment
```

```
[1] "contains"
```

```
setMethod(
  "contains",
  signature(object = "waldCI", value = "numeric"),
  function(object, value, level = 0.95)
  {
    lower <- lb(object, level)
    upper <- ub(object, level)
    return(value >= lower && value <= upper)
  }
)
```

```
#-----Overlap Method-----
setGeneric(
  "overlap",
  function(ci1, ci2, level = 0.95) standardGeneric("overlap")
)
```

```
[1] "overlap"
```

```
setMethod(
  "overlap",
  signature(ci1 = "waldCI", ci2 = "waldCI"),
  function(ci1, ci2, level = 0.95)
  {
    return(
      max(lb(ci1, level), lb(ci2, level)) <= min(ub(ci1, level), ub(ci2, level))
    )
  }
)
```

```r
#-----as.numeric Method-----
setMethod(
  "as.numeric",
  signature(x = "waldCI"),
  function(x, level = 0.95, ...)
  {
    return(c(lb(x, level), ub(x, level)))
  }
)
```

```r
#-----transform Method-----
setGeneric(
  "transform",
  function(object, ...) standardGeneric("transform")
)
```

```
Creating a new generic function for 'transform' in the global environment
```

```
[1] "transform"
```

```r
setMethod(
  "transform",
  signature(object = "waldCI"),
  function(object, func, level = 0.95, n = 21L, tol = 1e-10, ...)
  {
    lower <- lb(object, level)
    upper <- ub(object, level)

    # Monotonicity checking
    grid <- seq(lower, upper, length.out = max(3L, as.integer(n)))
    y <- vapply(grid, func, numeric(1))

    d <- diff(y)
    inc <- all(d >= -tol) # Approximate monotonic increasing
    dec <- all(d <= tol) # Approximate monotinic decreasing

    if (!inc && !dec)
      stop("func is non-monotonic on [lb, ub].")

    if (inc)
      return(waldCI(lower = func(lower), upper = func(upper), level = level))
```

```
    else
      return(waldCI(lower = func(upper), upper = func(lower), level = level))
  }
)
```

Part b

```
ci1 <- waldCI(lower = 17.2, upper = 24.7, level = 0.95)
ci2 <- waldCI(mean = 13, sterr = 2.5)
ci3 <- waldCI(lower = 27.43, upper = 39.22, level = 0.75)
```

```
ci1
```

```
An object of class "waldCI"
Slot "mu":
[1] 20.95

Slot "se":
[1] 1.9133
```

```
ci2
```

```
An object of class "waldCI"
Slot "mu":
[1] 13

Slot "se":
[1] 2.5
```

```
ci3
```

```
An object of class "waldCI"
Slot "mu":
[1] 33.325

Slot "se":
[1] 5.12453
```

```r
as.numeric(ci1)
```

```
[1] 17.2 24.7
```

```r
as.numeric(ci2, .8)
```

```
[1]  9.796121 16.203879
```

```r
as.numeric(ci3)
```

```
[1] 23.28111 43.36889
```

```r
show(ci1, .9)
```

```
waldCI:
 mean = 20.950,
 se = 1.913
 level = 0.900  (z = 1.645)
 CI: [17.803, 24.097]
```

```r
show(ci2, .8)
```

```
waldCI:
 mean = 13.000,
 se = 2.500
 level = 0.800  (z = 1.282)
 CI: [9.796, 16.204]
```

```r
show(ci3, .75)
```

```
waldCI:
 mean = 33.325,
 se = 5.125
 level = 0.750  (z = 1.150)
 CI: [27.430, 39.220]
```

```
lb(ci2)
```

```
[1] 8.10009
```

```
ub(ci2, .99)
```

```
[1] 19.43957
```

```
mean(ci1)
```

```
[1] 20.95
```

```
sterr(ci3)
```

```
[1] 5.12453
```

```
lb(ci2) <- 10.5
lb(ci2, .9) <- 10
mean(ci3) <- 34
contains(ci1, 17)
```

```
[1] FALSE
```

```
contains(ci2, 11, .9)
```

```
[1] TRUE
```

```
contains(ci3, 44)
```

```
[1] TRUE
```

```
overlap(ci1, ci2)
```

```
[1] TRUE
```

```r
overlap(ci1, ci2, .99)
```

```
[1] TRUE
```

```r
eci1 <- transform(ci1, exp)
eci1
```

```
An object of class "waldCI"
Slot "mu":
[1] 26686022167

Slot "se":
[1] 13600514831
```

```r
show(eci1, .75)
```

```
waldCI:
 mean = 26686022166.702,
 se = 13600514831.420
 level = 0.750  (z = 1.150)
 CI: [11040678357.584, 42331365975.821]
```

```r
mean(transform(ci2, sqrt))
```

```
[1] 3.697722
```

Part c

```r
# Negative standard error
ci4 <- waldCI(mean = 10, sterr = -1)
```

```
Error in validObject(.Object): invalid class "waldCI" object: `se` must be non-negative!
```

```r
# lb > ub
ci5 <- waldCI(lower = 10, upper = 5)
```

```
Error in waldCI(lower = 10, upper = 5): Need `lower` < `upper`
```

```r
# Infinite bound test
ci6 <- waldCI(lower = 10, upper = Inf)
```

```
Error in waldCI(lower = 10, upper = Inf): Need `lower` and `upper` be finite number.
```

```r
# Invalid use of `lb`
ci7 <- waldCI(mean = 10, sterr = 2)
lb(ci7) <- Inf
```

```
Error in validObject(object): invalid class "waldCI" object: `mu` must be finite!
```

```r
# Invalid use of `ub`
ci8 <- waldCI(lower = 10, upper = 20)
ub(ci8) <- Inf
```

```
Error in validObject(object): invalid class "waldCI" object: `mu` must be finite!
```

```r
# Invalid use of `mean`
ci9 <- waldCI(lower = 10, upper = 20)
mean(ci9) <- Inf
```

```
Error in validObject(object): invalid class "waldCI" object: `mu` must be finite!
```

```r
# Invalid use of `sterr`
ci10 <- waldCI(lower = 10, upper = 20)
sterr(ci10) <- -1
```

```
Error in validObject(object): invalid class "waldCI" object: `se` must be non-negative!
```

Question 2

Part a

```r
# Load Dataset
url <- "https://raw.githubusercontent.com/JeffSackmann/tennis_atp/refs/heads/master/atp_matc
tennis <- fread(url)

tourneys <- unique(tennis[ , .(tourney_name) ])
tourneys[ , tourney_name := sub("Davis.*", "Davis Cup", tourney_name) ]
tourneys <- unique(tourneys, by = "tourney_name")

cat("Number of tournaments that took place in 2019: ", dim(tourneys)[1], "\n")
```

```
Number of tournaments that took place in 2019:  69
```

Part b

```r
winners <- tennis[
  round == "F",
  .(n_tournaments = uniqueN(tourney_id)),
  by = .(winner_id)
][ order(-n_tournaments) ]

multi_winners <- winners[ n_tournaments > 1 ]

cat("Number of players who won more than one tournament: ", nrow(multi_winners), "\n")
```

```
Number of players who won more than one tournament:  12
```

```r
cat("Number of tournaments that the most winning player win: ",
    winners[1, n_tournaments], "\n")
```

```
Number of tournaments that the most winning player win:  5
```

Part c

We use Bootstrap to build a 95% confidence interval for the mean difference without assuming normality, and the hypothesis is:

$$H_0 : ace_{winner} - ace_{loser} > 0, \quad H_1 : ace_{winner} - ace_{loser} \leq 0$$

```r
# 1) Per-match ace difference (winner - loser)
ace_diff <- tennis[ , .(diff = as.numeric(w_ace) - as.numeric(l_ace)) ][ !is.na(diff) ]

# 2) Observed mean
obs_mean <- ace_diff[ , mean(diff) ]

# 3) Bootstrap (5000 resamples with replacement)
set.seed(506)
B <- 5000L
n <- nrow(ace_diff)

boot_stat <- replicate(B, {
  idx <- sample.int(n, n, replace = TRUE)
  mean(ace_diff$diff[idx])
})

# 4) One-sided percentile confidence intervals
lower_bound <- unname(quantile(boot_stat, probs = 0.05))

# Output and brief explanation
cat("Observed mean difference (winner - loser): ", obs_mean, "\n")
```

```
Observed mean difference (winner - loser):  1.7049
```

```r
cat("One-sided 95% LOWER CI (percentile): [", lower_bound, ", Inf)\n", sep = "")
```

```
One-sided 95% LOWER CI (percentile): [1.483296, Inf)
```

Part d

```r
# Long form: one row per player per match, with win indicator
long_players <- rbindlist(
  list(
    tennis[ , .(player_id = winner_id, player_name = winner_name, win = 1L) ],
    tennis[ , .(player_id = loser_id,  player_name = loser_name,  win = 0L) ]
  ),
  use.names = TRUE
)
```

```r
# Summarise to matches, wins, and win_rate per player
win_rate <- long_players[ ,
  .(matches = .N, wins = sum(win)),
  by = .(player_id, player_name)
][
  , win_rate := wins / matches
][
  matches >= 5
][
  order(-win_rate, -wins, -matches)
]

# Show the table
print(win_rate)
```

```
     player_id     player_name matches  wins  win_rate
         <int>          <char>   <int> <int>     <num>
  1:    104745    Rafael Nadal      69    60 0.8695652
  2:    104925  Novak Djokovic      69    58 0.8405797
  3:    103819   Roger Federer      66    55 0.8333333
  4:    106421 Daniil Medvedev      80    59 0.7375000
  5:    104731  Kevin Anderson      15    11 0.7333333
 ---
163:    106058       Jack Sock       5     1 0.2000000
164:    104810          Zhe Li       7     1 0.1428571
165:    111200      Elias Ymer       8     1 0.1250000
166:    106075   Jozef Kovalik      10     1 0.1000000
167:    105155     Pedro Sousa       9     0 0.0000000
```

```r
# Text output for the top player
cat("The player with the highest win-rate is: ", win_rate$player_name[1], "\n")
```

```
The player with the highest win-rate is:  Rafael Nadal
```

Question 3

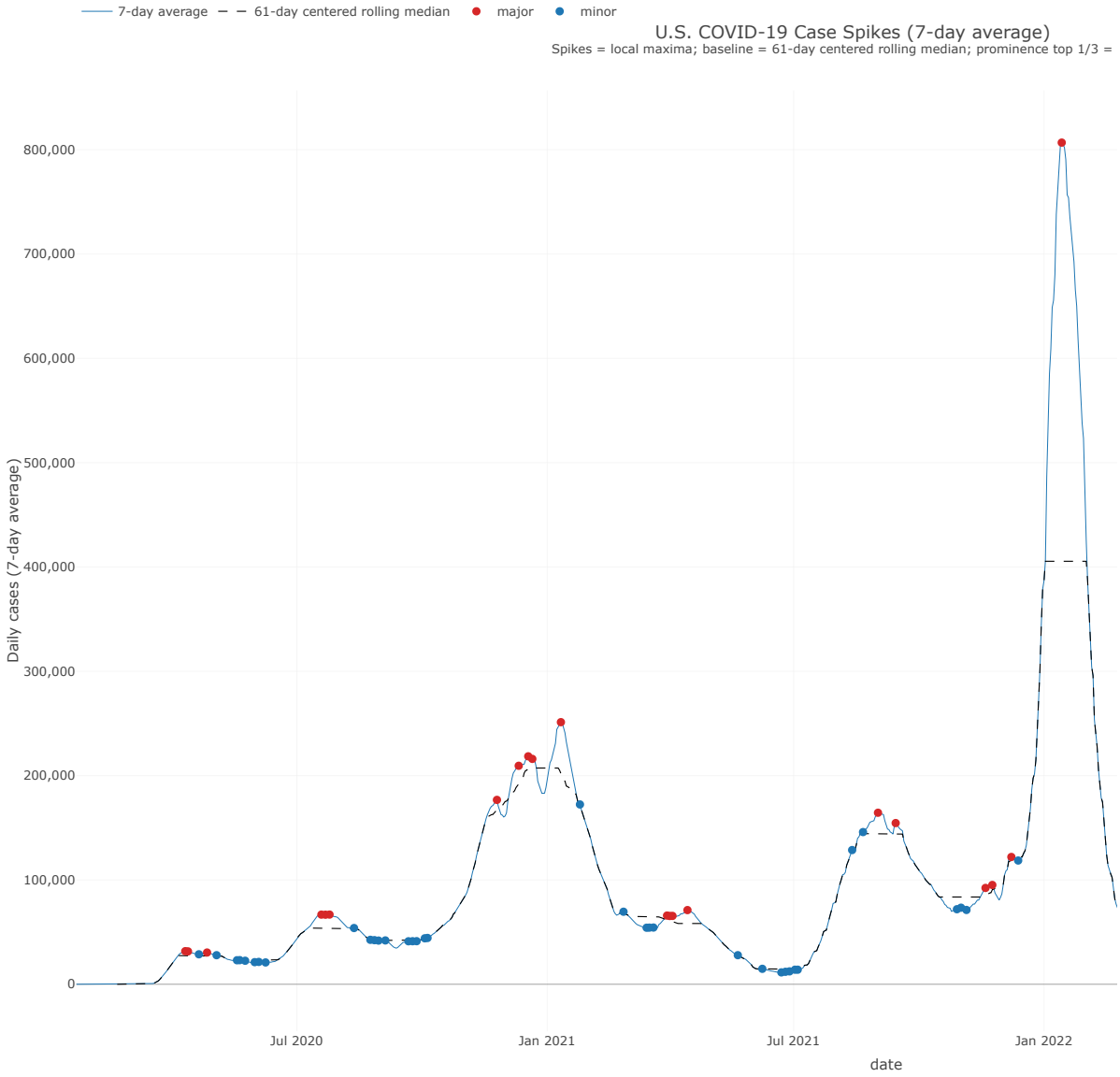
Part a

```r
# Load Dataset
covid <- read_csv(
  "https://raw.githubusercontent.com/nytimes/covid-19-data/refs/heads/master/rolling-averages
  show_col_types = FALSE
) %>%
  arrange(date) %>%
  mutate(
    base_med = zoo::rollmedian(cases_avg, k = 61, fill = NA, align = "center")
  )

# Identify Spikes
k <- 61  # centered rolling window length (~±30 days)

covid_peaks <- covid %>%
  mutate(
    # local maxima
    is_peak  = cases_avg > dplyr::lag(cases_avg) & cases_avg > dplyr::lead(cases_avg),
    # prominence above baseline
    prominence = pmax(cases_avg - base_med, 0)
  ) %>%
  filter(is_peak) %>%
  mutate(
    thresh = quantile(prominence, probs = 2/3, na.rm = TRUE),
    spike_type = if_else(prominence >= thresh, "major", "minor")
  )

peaks_major <- dplyr::filter(covid_peaks, spike_type == "major")
peaks_minor <- dplyr::filter(covid_peaks, spike_type == "minor")

# --- Plotly ---
p <- plot_ly()

# Main cases-average line
p <- p %>%
  add_lines(
    data = covid,
    x = ~date, y = ~cases_avg,
    name = "7-day average",
    line = list(color = "gray40", width = 0.9),
    hoverinfo = "text",
    text = ~paste0(
      "Date: ", date,
```

```
      "<br>Cases (7-day avg): ", format(round(cases_avg), big.mark = ",")
    )
  )

# 61-day centered rolling median (baseline), dashed
p <- p %>%
  add_lines(
    data = covid,
    x = ~date, y = ~base_med,
    name = "61-day centered rolling median",
    line = list(color = "black", dash = "dash", width = 1),
    hoverinfo = "text",
    text = ~paste0(
      "Date: ", date,
      "<br>Baseline (61-day med): ",
      ifelse(is.na(base_med), "NA", format(round(base_med), big.mark = ","))
    )
  )

# Spike points: major (red)
p <- p %>%
  add_markers(
    data = peaks_major,
    x = ~date, y = ~cases_avg,
    name = "major",
    marker = list(size = 8, color = "#d62728"),
    hoverinfo = "text",
    text = ~paste0(
      "Spike: major",
      "<br>Date: ", date,
      "<br>Cases (7-day avg): ", format(round(cases_avg), big.mark = ","),
      "<br>Prominence: ", round(prominence, 1)
    )
  )

# Spike points: minor (blue)
p <- p %>%
  add_markers(
    data = peaks_minor,
    x = ~date, y = ~cases_avg,
    name = "minor",
    marker = list(size = 8, color = "#1f77b4"),
```

```
    hoverinfo = "text",
    text = ~paste0(
      "Spike: minor",
      "<br>Date: ", date,
      "<br>Cases (7-day avg): ", format(round(cases_avg), big.mark = ","),
      "<br>Prominence: ", round(prominence, 1)
    )
  )

# Layout (title + subtitle, legend on top, minimal theme)
p <- p %>%
  layout(
    title = list(
      text = paste0(
        "U.S. COVID-19 Case Spikes (7-day average)",
        "<br><sup>Spikes = local maxima; baseline = 61-day centered rolling median; prominenc
      )
    ),
    xaxis = list(title = NULL),
    yaxis = list(title = "Daily cases (7-day average)", tickformat = ","),
    legend = list(orientation = "h", x = 0, y = 1.1, xanchor = "left"),
    margin = list(t = 90),
    template = "plotly_white",
    annotations = list(
      list(
        text = "Source: NYTimes COVID-19 rolling averages",
        showarrow = FALSE, xref = "paper", yref = "paper",
        x = 0, y = -0.18, xanchor = "left", yanchor = "top",
        font = list(size = 11, color = "gray40")
      )
    ),
    hovermode = "x unified"
  )

p
```

file:////private/var/folders/4c/lcrp_5b50zlb0jbyp80h68j80000gn/T/RtmpNYeUlx/file15618146f30/w



There're roughly 5 major spikes:

1. Spring 2020
2. Winter 2020 - Spring 2021
3. Summer 2021
4. Winter 2021 - Spring 2022
5. Summer 2022

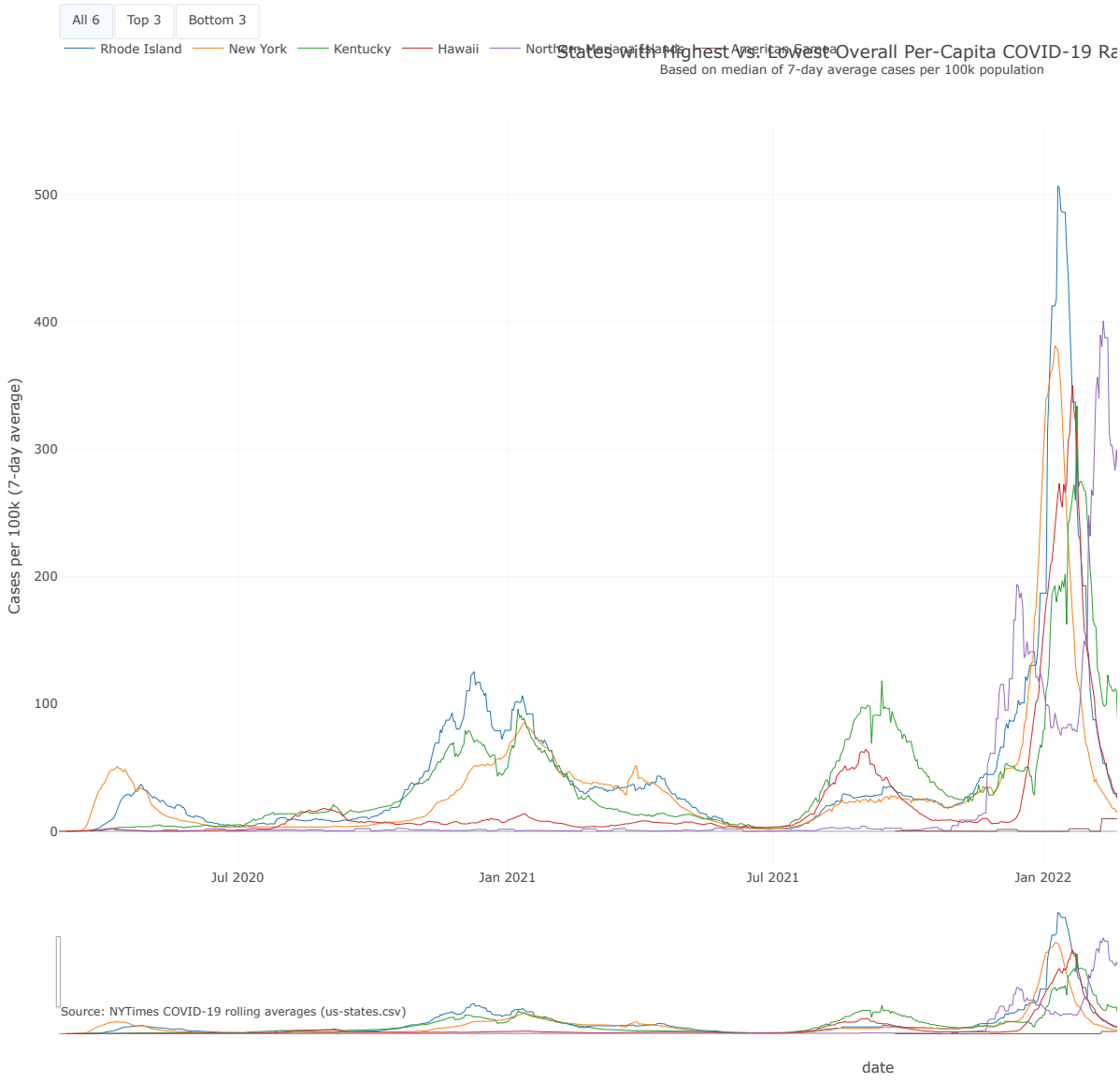
Part b

```r
# Load Dataset
covid_states <- read_csv(
  "https://raw.githubusercontent.com/nytimes/covid-19-data/refs/heads/master/rolling-averages
  show_col_types = FALSE
)

# Compute overall (median) per-capita rate per state
state_rate <- covid_states %>%
  group_by(state) %>%
  summarise(
    overall_rate = median(cases_avg_per_100k, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  arrange(desc(overall_rate))

# Pick top and bottom 3 states (preserve order, drop any accidental dups)
top_states    <- state_rate %>% slice_head(n = 3) %>% pull(state)
bottom_states <- state_rate %>% slice_tail(n = 3) %>% pull(state)
sel_states    <- unique(c(top_states, bottom_states))

compare_states <- covid_states %>%
  filter(state %in% sel_states) %>%
  arrange(state, date)

# Build a single interactive chart with one trace per state
p <- plot_ly()
for (st in sel_states) {
  df <- dplyr::filter(compare_states, state == st)
  p <- p %>%
    add_lines(
      data = df,
      x = ~date, y = ~cases_avg_per_100k,
      name = st,
      line = list(width = 0.9),
      hoverinfo = "text",
      text = ~paste0(
        "State: ", st,
        "<br>Date: ", date,
        "<br>Cases per 100k (7-day avg): ", round(cases_avg_per_100k, 2)
      )
```

```
    )
}

# Visibility masks for the buttons (one element per trace, same order as sel_states)
vis_all    <- rep(TRUE,  length(sel_states))
vis_top    <- sel_states %in% top_states
vis_bottom <- sel_states %in% bottom_states

p <- p %>%
  layout(
    title = list(
      text = paste0(
        "States with Highest vs. Lowest Overall Per-Capita COVID-19 Rates",
        "<br><sup>Based on median of 7-day average cases per 100k population</sup>"
      )
    ),
    xaxis = list(title = NULL, rangeslider = list(visible = TRUE)),
    yaxis = list(title = "Cases per 100k (7-day average)"),
    template = "plotly_white",
    legend = list(orientation = "h", x = 0, y = 1.08, xanchor = "left"),
    margin = list(t = 110),
    hovermode = "x unified",
    updatemenus = list(list(
      type = "buttons",
      direction = "right",
      x = 0, y = 1.16, xanchor = "left",
      buttons = list(
        list(label = "All 6",    method = "update", args = list(list(visible = vis_all))),
        list(label = "Top 3",    method = "update", args = list(list(visible = vis_top))),
        list(label = "Bottom 3", method = "update", args = list(list(visible = vis_bottom)))
      )
    )),
    annotations = list(
      list(
        text = "Source: NYTimes COVID-19 rolling averages (us-states.csv)",
        showarrow = FALSE, xref = "paper", yref = "paper",
        x = 0, y = -0.18, xanchor = "left", yanchor = "top",
        font = list(size = 11, color = "gray40")
      )
    )
  )
```

```
p
```

```
file:////private/var/folders/4c/lcrp_5b50zlb0jbyp80h68j80000gn/T/RtmpNYeUlx/file15612ce67e86/
```



High-rate areas such as `American Samoa`, `Hawaii`, and `Kentucky` show sharp, concentrated peaks, indicating intense but relatively short-lived outbreaks. In contrast, low-rate areas like `New York`, `Northern Mariana Islands`, and `Rhode Island` exhibit lower, broader, or more irregular curves, suggesting more prolonged but less severe transmission.

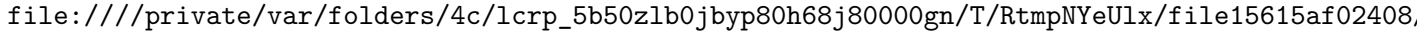
Overall, the trajectories demonstrate that per-capita intensity and outbreak duration varied greatly across regions, reflecting differences in timing, containment policies, and population density.

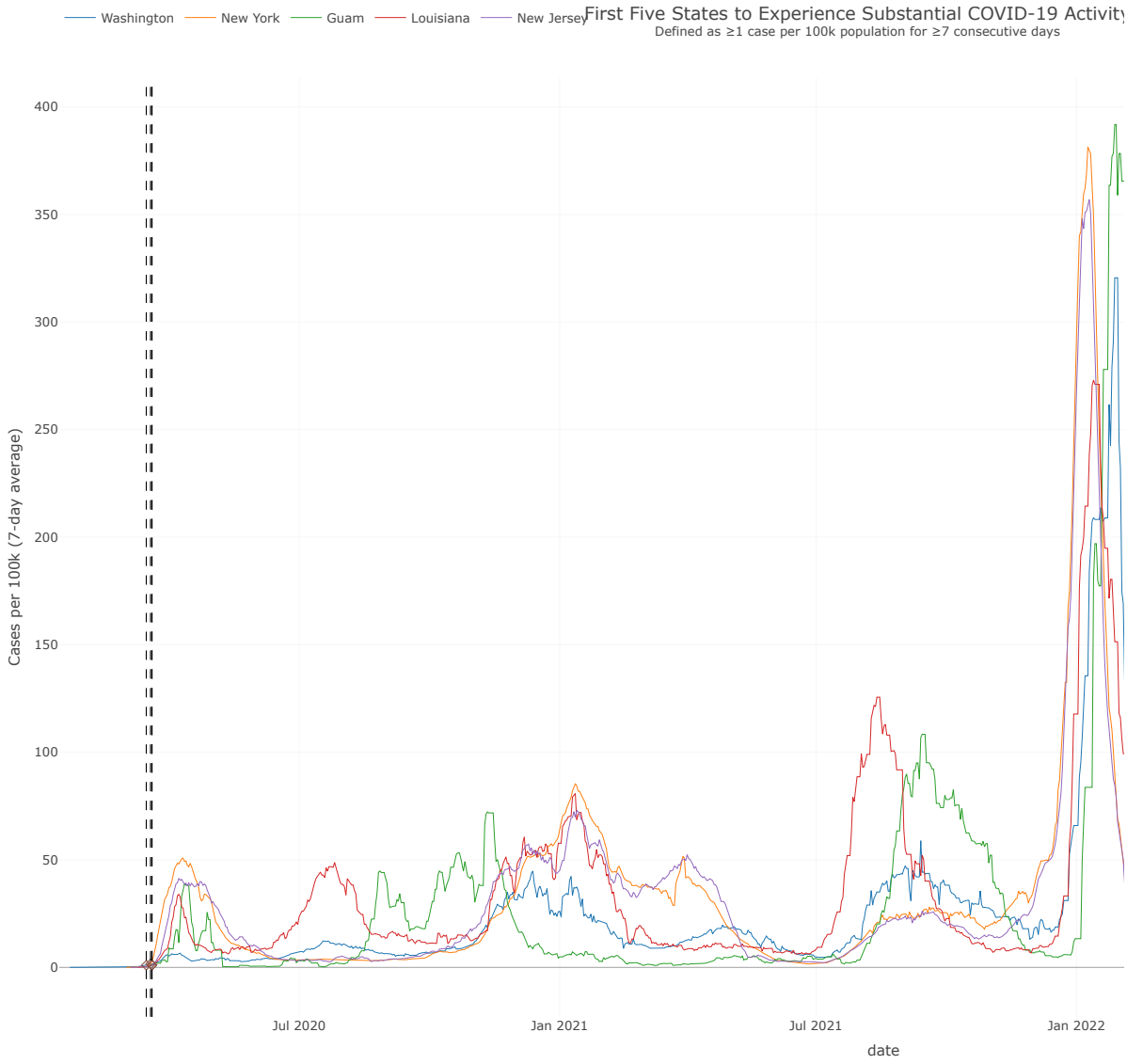
Part c

```
# ---- Load data ----
covid_states <- read_csv(
  "https://raw.githubusercontent.com/nytimes/covid-19-data/refs/heads/master/rolling-averages
  show_col_types = FALSE
)

# ---- Define "substantial" period ----
threshold <- 1.0   # cases per 100k
min_days  <- 7     # consecutive days

# For each state, find the first sustained >= threshold run
first_substantial <- covid_states %>%
  arrange(state, date) %>%
  group_by(state) %>%
  mutate(
    above = cases_avg_per_100k >= threshold,
    run   = data.table::rleid(above)
  ) %>%
  group_by(state, run, .add = TRUE) %>%
  summarise(
    start_date = first(date),
    end_date   = last(date),
    days       = n(),
    above      = first(above),
    .groups    = "drop_last"
  ) %>%
  ungroup() %>%
  filter(above, days >= min_days) %>%
  group_by(state) %>%
  summarise(first_substantial_date = min(start_date), .groups = "drop") %>%
  arrange(first_substantial_date)

# First 5 states and their time series
first5 <- first_substantial %>% slice_head(n = 5)
early_states <- covid_states %>%
```

```r
  filter(state %in% first5$state) %>%
  arrange(state, date)

# ---- Build plotly (single panel; all five states) ----
p <- plot_ly()

# add one trace per state
state_order <- first5$state
for (st in state_order) {
  df <- dplyr::filter(early_states, state == st)
  p <- p %>%
    add_lines(
      data = df,
      x = ~date, y = ~cases_avg_per_100k,
      name = st,
      line = list(width = 0.9),
      hoverinfo = "text",
      text = ~paste0(
        "State: ", st,
        "<br>Date: ", date,
        "<br>Cases per 100k (7-day avg): ", round(cases_avg_per_100k, 2)
      )
    )
}

# dashed vertical lines at each state's first substantial date
vlines <- lapply(seq_len(nrow(first5)), function(i) {
  list(
    type = "line",
    xref = "x", yref = "paper",
    x0 = first5$first_substantial_date[i],
    x1 = first5$first_substantial_date[i],
    y0 = 0, y1 = 1,
    line = list(dash = "dash", width = 1)
  )
})

# optional markers exactly on that day
mark_points <- early_states %>%
  inner_join(first5, by = "state") %>%
  filter(date == first_substantial_date)
```

```r
p <- p %>%
  add_markers(
    data = mark_points,
    x = ~date, y = ~cases_avg_per_100k,
    name = "first substantial day",
    marker = list(size = 7, symbol = "diamond-open"),
    hoverinfo = "text",
    text = ~paste0(
      "State: ", state,
      "<br>First   ", threshold, "/100k for   ", min_days, " days: ", date,
      "<br>Cases per 100k (that day): ", round(cases_avg_per_100k, 2)
    ),
    showlegend = FALSE
  )

p <- p %>%
  layout(
    title = list(
      text = paste0(
        "First Five States to Experience Substantial COVID-19 Activity",
        "<br><sup>Defined as  ", threshold, " case per 100k population for  ", min_days, " co
      )
    ),
    xaxis = list(title = NULL),
    yaxis = list(title = "Cases per 100k (7-day average)"),
    template = "plotly_white",
    legend = list(orientation = "h", x = 0, y = 1.08, xanchor = "left"),
    margin = list(t = 110),
    shapes = vlines,
    annotations = list(
      list(
        text = "Source: NYTimes COVID-19 rolling averages (us-states.csv)",
        showarrow = FALSE, xref = "paper", yref = "paper",
        x = 0, y = -0.16, xanchor = "left", yanchor = "top",
        font = list(size = 11, color = "gray40")
      )
    )
  )

p
```





The first five states (or territories) to experience substantial COVID-19 activity were `Guam`, `Louisiana`, `New Jersey`, `New York`, and `Washington`, with outbreaks emerging around March 2020, marking the start of widespread community transmission in the United States.