EPI590R R Lab 1 - Dataset A

Nicole Luisi

10-07-2024

In the EPI590R Week 3 lab, you learned how to manipulate data and create Epi Curves with Microsoft Excel. In this lab, we will use the same datasets to produce similar visualizations with R.

First, make sure you have the necessary packages installed and loaded.

**readxl** to read Excel files

**dplyr** and **tidyr** to manipulate data

**lubridate** to work with dates

**stringr** to work with strings

**ggplot2** to create data visualizations

**knitr** to help with our images in the document

**RColorBrewer** for colors

Step 1 - Import the data (Dataset\_A.xlsx).

dataA <- read\_excel("data\\Dataset\_A.xlsx")

Step 2 - Explore the data.

The **dplyr::glimpse()** function will show us the number of rows and columns, list the variables and variable types, and give us an idea of the data values. We could also use **str()** for this. The **head()** function will print just a few rows of the dataset, here **n=10** rows is specified.

Next, to explore the data we will use **lapply()** to ask R to return the most appropriate type of output based on variable type. Numeric variables will use **summary()**, factor and character variables with use **table()**, and datetime variables will also use **summary()**.

*Note that we are using results=‘hide’ in the code chunk to suppress all of this output from the knitted Word doc.*.

glimpse(dataA)  
  
head(dataA, n=10)  
  
lapply(dataA, function(x) {  
 if (is.numeric(x)) return(summary(x))  
 if (is.factor(x)) return(table(x))  
 if (is.character(x)) return(table(x))  
 if (inherits(x, "POSIXct") || inherits(x, "POSIXt")) {return(summary(x))}  
})

Step 3 - Clean and prepare data for analysis.

Based on the previous Excel lab, we will be using **pos\_test\_date** for our cases, and we may want to do some grouping by race (**race**) and age (**DOB**) later on, so we will start by cleaning those variables up a bit.

Here we are creating 3 new variables:

The **age** variable is calculated by subtracting the date of birth (**DOB**) from the positive test date (**pos\_test\_date**) and converting that to years; this will give us the age at the time of diagnosis.

From there, the **age\_group** factor variable is created by using the continuous age values to make 10 year age groups, with the exception of the last group which will have all values >=70 years.

Next, a **race\_cat** factor variable is created that collapses values of “Other” and “Multiple Races” into one category.

It looks like we have one missing value for **pos\_test\_date**, so in our last step we will remove this row with **filter()**.

dataA <- dataA %>%  
 mutate(age = as.numeric(difftime(pos\_test\_date, DOB, units = "days")) %/% 365.25,  
 age\_group = as.factor(cut(age,   
 breaks = c(seq(0, 70, by = 10), Inf),   
 right = FALSE,   
 labels = c(paste(seq(0, 60, by = 10), seq(9, 69, by = 10), sep = "-"), "70+"),  
 include.lowest = TRUE)),  
 race\_cat = as.factor(case\_when(  
 race == "American Indian/Alaska Native" ~ "American Indian/Alaska Native",  
 race == "Asian" ~ "Asian",  
 race == "Black" ~ "Black",  
 race == "Native Hawaiian/Other Pacific Islander" ~ "Native Hawaiian/Other Pacific Islander",  
 race == "White" ~ "White",  
 race %in% c("Other", "Multiple Races") ~ "Other or Multiple Races",  
 TRUE ~ "Unknown")))  
  
# Check age variables  
check <- dataA %>% select(pos\_test\_date, DOB, age, age\_group)  
head(check, n=10)

## # A tibble: 10 × 4  
## pos\_test\_date DOB age age\_group  
## <dttm> <dttm> <dbl> <fct>   
## 1 2023-08-03 00:00:00 1997-04-02 00:00:00 26 20-29   
## 2 2022-01-02 00:00:00 1963-07-16 00:00:00 58 50-59   
## 3 2023-03-05 00:00:00 1992-09-17 00:00:00 30 30-39   
## 4 2022-04-26 00:00:00 1986-10-10 00:00:00 35 30-39   
## 5 2022-10-10 00:00:00 1988-08-18 00:00:00 34 30-39   
## 6 2023-04-14 00:00:00 1971-12-08 00:00:00 51 50-59   
## 7 2023-12-08 00:00:00 1997-05-10 00:00:00 26 20-29   
## 8 2023-08-20 00:00:00 1974-11-13 00:00:00 48 40-49   
## 9 2023-05-07 00:00:00 2006-05-16 00:00:00 16 10-19   
## 10 2023-01-01 00:00:00 1937-10-29 00:00:00 85 70+

# Check race variable  
dataA %>% count(race, race\_cat) %>%   
 tidyr::pivot\_wider(names\_from = race\_cat, values\_from = n, values\_fill = list(n = 0))

## # A tibble: 9 × 8  
## race American Indian/Alas…¹ Asian Black Other or Multiple Ra…²  
## <chr> <int> <int> <int> <int>  
## 1 American Indian/Ala… 55 0 0 0  
## 2 Asian 0 374 0 0  
## 3 Black 0 0 901 0  
## 4 Multiple Races 0 0 0 336  
## 5 Native Hawaiian/Oth… 0 0 0 0  
## 6 Other 0 0 0 132  
## 7 Unknown 0 0 0 0  
## 8 White 0 0 0 0  
## 9 <NA> 0 0 0 0  
## # ℹ abbreviated names: ¹​`American Indian/Alaska Native`,  
## # ²​`Other or Multiple Races`  
## # ℹ 3 more variables: `Native Hawaiian/Other Pacific Islander` <int>,  
## # Unknown <int>, White <int>

# Remove row with missing pos\_test\_date value  
dataA <- dataA %>% filter(!is.na(pos\_test\_date))

Step 4 - Plot the case counts by day.

First, we will use **summarise()** to count the number of cases by day. Our **pos\_test\_date** variable is already by day (exact dates) so we will use this as the grouping variable.

dataA\_cases\_per\_day <- dataA %>%  
 group\_by(pos\_test\_date) %>%  
 summarise(cases = n()) %>%  
 mutate(pos\_test\_date=as.Date(pos\_test\_date))   
  
# Check results  
head(dataA\_cases\_per\_day, n=10)

## # A tibble: 10 × 2  
## pos\_test\_date cases  
## <date> <int>  
## 1 2022-01-01 10  
## 2 2022-01-02 8  
## 3 2022-01-03 5  
## 4 2022-01-04 7  
## 5 2022-01-05 4  
## 6 2022-01-06 2  
## 7 2022-01-07 1  
## 8 2022-01-08 3  
## 9 2022-01-09 3  
## 10 2022-01-10 8

Before we try to plot anything, let’s take a look at the date range for **pos\_test\_date**. We will double check to make sure there are no NAs (there shouldn’t be because of our earlier filter), and then use **range()**. We are including **na.rm=T** so that this step works even if there are NAs, but again we should not have any.

It looks like our date range is **Jan 1, 2022 to December 31, 2023**.

We can also get a range of values for the case counts to inform how we set up our plot axis.

It looks like we have daily counts from **1 to 11**.

# Check to make sure there are no NAs   
table(is.na(dataA\_cases\_per\_day$pos\_test\_date))

##   
## FALSE   
## 718

# Now to get the date range   
range(dataA\_cases\_per\_day$pos\_test\_date, na.rm = TRUE)

## [1] "2022-01-01" "2023-12-31"

# And range of values for the cases  
range(dataA\_cases\_per\_day$cases)

## [1] 1 11

Okay, now let’s get to work on a daily case plot.

First, we are going to use our date range to make a start and end point. We want to make sure our plot is uniform, so we will use the list of dates to enforce consistent 1 day increments, even if we don’t have any cases on some days (we will merge this list of dates with our own data and set the case count for any days without cases to zero).

We will also go ahead and create a variable to store the cumulative sum of cases (**cumulative\_case\_count**) using the **cumsum()** function.

# Dataset of all dates in the range we have  
all\_dates <- data.frame(daterange = seq(as.Date("2022-01-01"), as.Date("2023-12-31"), by = "day"))  
  
# Merge with our data and insert 0's if no cases   
dataA\_cases\_per\_day\_rev <- merge(x=all\_dates,   
 y=dataA\_cases\_per\_day,   
 by.x = "daterange",   
 by.y="pos\_test\_date",   
 all.x=T) %>%  
 mutate(cases = replace\_na(cases, 0)) %>%  
 mutate(cumulative\_case\_count = cumsum(cases))  
  
# Check result  
head(dataA\_cases\_per\_day\_rev, n=10)

## daterange cases cumulative\_case\_count  
## 1 2022-01-01 10 10  
## 2 2022-01-02 8 18  
## 3 2022-01-03 5 23  
## 4 2022-01-04 7 30  
## 5 2022-01-05 4 34  
## 6 2022-01-06 2 36  
## 7 2022-01-07 1 37  
## 8 2022-01-08 3 40  
## 9 2022-01-09 3 43  
## 10 2022-01-10 8 51

Next, let’s make a plot of the cases by day using **ggplot**.

Note that the new date variable in the merged file is called **daterange** and the case count variable is called **cases**; these will become our x and y for the plot.

**geom\_col(fill = "#002060")** is setting the color for our columns. You can swap that out for any hex code or color name in the quotes.

**labs()** allows us to set a main title for the plot (title=), as well as titles for the x (x=) and y (y=) axis. You can edit those labels as needed.

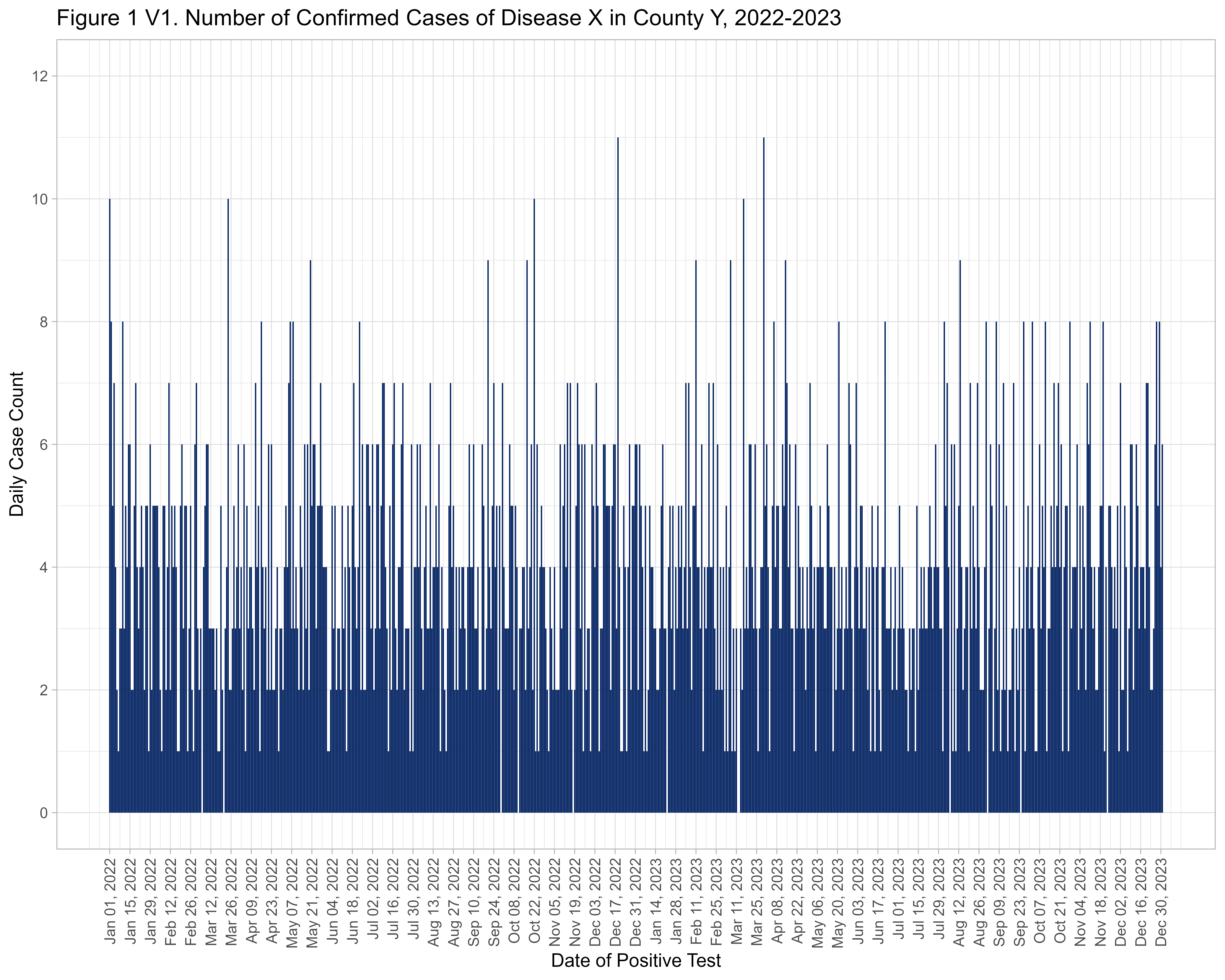
**scale\_x\_date()** is used to change the format of the date labels on the x axis. The **%b** gives an abbreviated month name (i.e., Jan, Feb, Mar), **%d** gives a 2-digit day, and **%Y** gives a 4-digit year. You can try using other date formats here as well. The **breaks** argument is specifying 2 week intervals for the dates between the start and end value we set.

**scale\_y\_continuous()** is creating some specifications for the values on the y axis based on our maximum daily case counts.

**theme\_light()** and **theme()** elements are just customizing the look a bit more. For the x axis text, we are making sure those dates are vertical (angle=90) so they don’t overlap, similar to the way they look in Excel.

Note that we are storing the whole plot in an object called **fig** so we can easily save it as an image file. We are using **theme\_light()** here because this looks best in the png file.

# Provide start and end dates to force plot boundaries  
start\_date <- as.Date("2022-01-01")   
end\_date <- as.Date("2023-12-31")   
  
# Daily plot  
fig1a <- ggplot(dataA\_cases\_per\_day\_rev, aes(x = daterange, y = cases)) +  
 geom\_col(fill = "#002060") +  
 labs(title = "Figure 1 V1. Number of Confirmed Cases of Disease X in County Y, 2022-2023",  
 x = "Date of Positive Test",  
 y = "Daily Case Count") +  
 scale\_x\_date(date\_labels = "%b %d, %Y",   
 breaks = seq(start\_date, end\_date, by="2 weeks")) +  
 scale\_y\_continuous(  
 limits = c(0, 12),   
 breaks = seq(0, 12, by = 2)) +  
 theme\_light() +   
 theme(axis.text.x = element\_text(angle = 90, hjust = 1, vjust = 0.5))  
  
# Save plot as png image   
ggsave("output\\Figure 1 V1 Dataset A.png", plot = fig1a, width = 10, height = 8, dpi = 500)  
  
# Print image here  
knitr::include\_graphics("output\\Figure 1 V1 Dataset A.png")



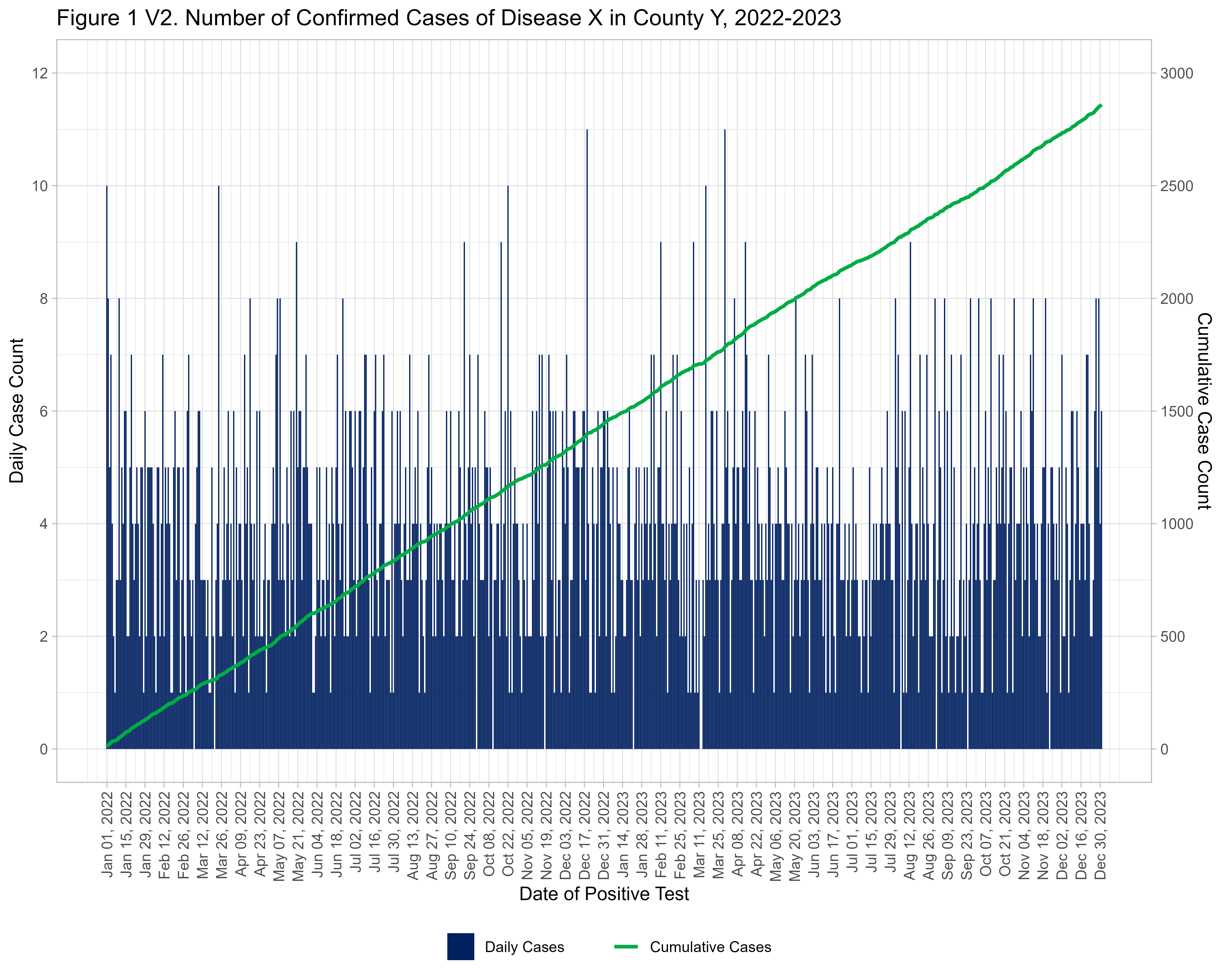
Now let’s fancy this up a bit more like you did in Excel. Here we will copy and paste the same basic plot and add some additional features.

**geom\_line()** is used to overlay a line with our cumulative case count (**cumulative\_case\_count**). Here we are using 12/3000 to rescale the cumulative case count to match the scale we have for daily cases.

**sec.axis()** in **scale\_y\_continuous()** allows us to have a secondary axis on the right side of the plot for the cumulative case count while keeping the scale we have for daily case count on the left. We will use increments of 500 for this one. By reworking the aes settings in this version we can use **scale\_fill\_manual** and **scale\_color\_manual** to set colors for the daily case count columns and the cumulative case total line. Again, feel free to adjust these color hex codes - check out [Color Brewer](https://colorbrewer2.org/) for guidance.

In the **theme()** section we will specify some info for the legend so we can get a key at the bottom with the colors for our columns and line.

# Plot  
fig1b <- ggplot(dataA\_cases\_per\_day\_rev, aes(x = daterange)) +  
 geom\_col(aes(y=cases, fill = "Daily Cases"), show.legend = TRUE) +  
 geom\_line(aes(y = cumulative\_case\_count \* (12/3000), color = "Cumulative Cases"), linewidth = 1) +  
 labs(title = "Figure 1 V2. Number of Confirmed Cases of Disease X in County Y, 2022-2023",  
 x = "Date of Positive Test",  
 y = "Daily Case Count") +  
 scale\_x\_date(date\_labels = "%b %d, %Y",   
 breaks = seq(start\_date, end\_date, by="2 weeks")) +  
 scale\_y\_continuous(  
 limits = c(0, 12),  
 breaks = seq(0, 12, by = 2),  
 sec.axis = sec\_axis(~ .\*(3000/12), name = "Cumulative Case Count", breaks = seq(0, 3000, by = 500))) +  
 scale\_fill\_manual(values = "#002060", labels = "Daily Cases") +   
 scale\_color\_manual(values = "#00AC45", labels = "Cumulative Cases") +   
 theme\_light() +   
 theme(axis.text.x = element\_text(angle = 90, hjust = 1, vjust = 0.5),  
 legend.position = "bottom",  
 legend.title = element\_blank()) +  
 guides(color = guide\_legend(override.aes = list(fill = NA, shape = NA)),  
 fill = guide\_legend(override.aes = list(color = NA)))  
  
# Save plot as png image   
ggsave("output\\Figure 1 V2 Dataset A.png", plot = fig1b, width = 10, height = 8, dpi = 500)  
  
# Print image here  
knitr::include\_graphics("output\\Figure 1 V2 Dataset A.png")



Step 5 - Plot the case counts by month.

First, we will use **summarise()** to count the number of cases by month. Here, **as.Date(cut(pos\_test\_date, "month"))** will group the counts by unique values for month.

dataA\_cases\_per\_month <- dataA %>%  
 mutate(month = as.Date(cut(pos\_test\_date, "month"))) %>%  
 group\_by(month) %>%  
 summarise(cases = n())  
  
# Check results  
head(dataA\_cases\_per\_month, n=10)

## # A tibble: 10 × 2  
## month cases  
## <date> <int>  
## 1 2022-01-01 137  
## 2 2022-02-01 103  
## 3 2022-03-01 110  
## 4 2022-04-01 107  
## 5 2022-05-01 144  
## 6 2022-06-01 111  
## 7 2022-07-01 125  
## 8 2022-08-01 120  
## 9 2022-09-01 121  
## 10 2022-10-01 120

Again, before we try to plot anything, let’s take a look at the date range for **month**. We will double check to make sure there are no NAs (no months without any cases).

It looks like our date range is **Jan 2022 to December 2023**.

We can also get a range of values for the case counts to inform how we set up our plot axis.

It looks like we have monthly counts from **100 to 144**.

# Check to make sure there are no NAs   
table(is.na(dataA\_cases\_per\_month$month))

##   
## FALSE   
## 24

# Now to get the date range   
range(dataA\_cases\_per\_month$month, na.rm = TRUE)

## [1] "2022-01-01" "2023-12-01"

# And range of values for the cases  
range(dataA\_cases\_per\_month$cases)

## [1] 100 144

Now let’s make the monthly plot.

Note that the new date variable is called **month** and the case count variable is called **cases**; these will become our x and y for the plot.

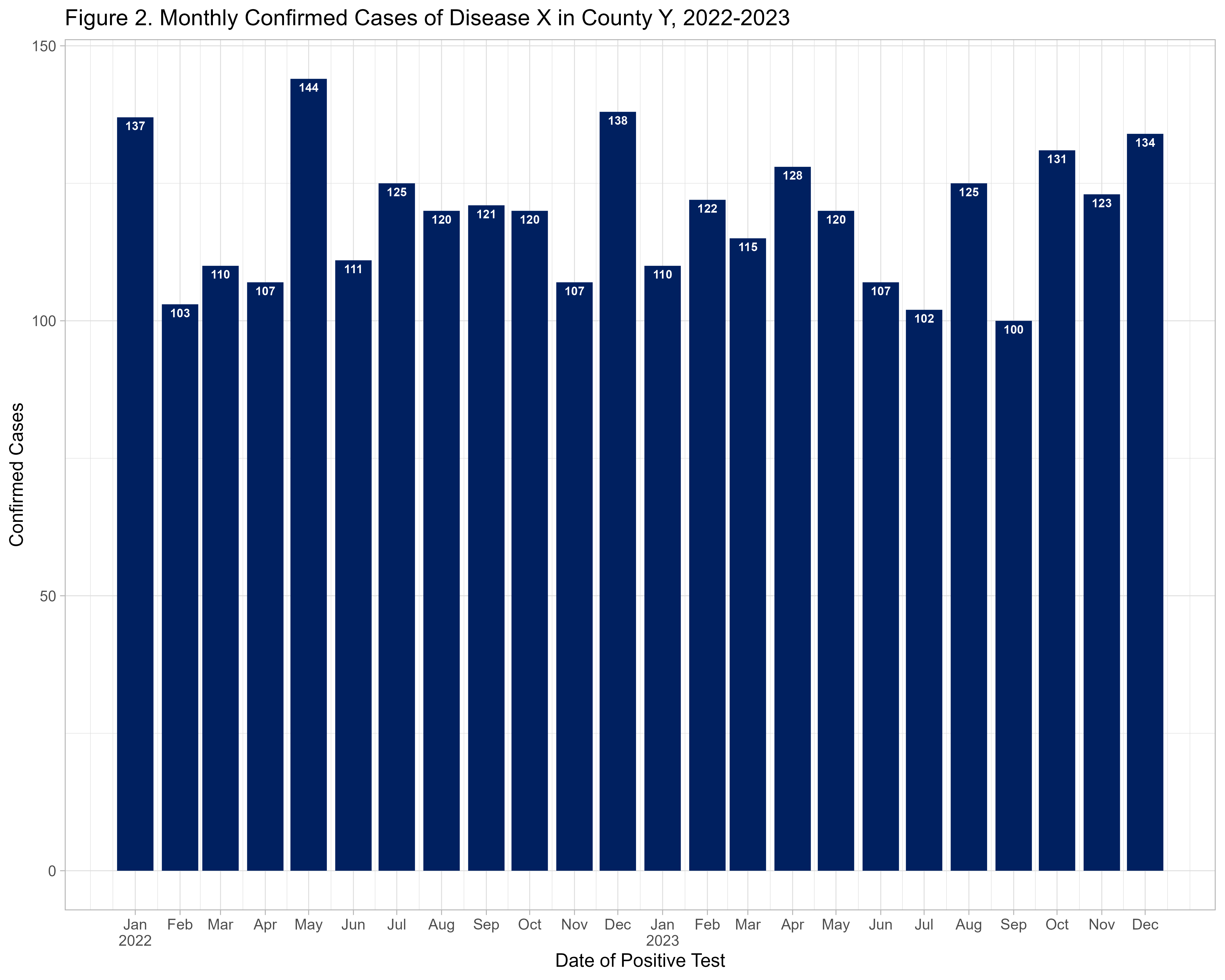
The elements of the monthly plot are similar to the daily plot above, but we have done a few extra things.

For the monthly plot, the **scale\_x\_date()** breaks are 1-month intervals.

The **custom\_labels** function is allowing us to plot the month abbreviations on the x axis but only including the year value with the month when the year changes, otherwise the axis is really busy and repetitive.

We are also using **geom\_text()** to insert a label with the number of cases inside each column; you can adjust the placement, size, and color of these labels as needed.

# Function to help with some custom labels for our x axis  
custom\_labels <- function(dates) {  
 months <- format(dates, "%b")   
 years <- format(dates, "%Y")   
 labels <- ifelse(duplicated(years), months, paste(months, years, sep = "\n"))  
 return(labels)  
}  
  
# Plot  
fig2 <- ggplot(dataA\_cases\_per\_month, aes(x = month, y = cases)) +  
 geom\_col(fill = "#002060") +  
 geom\_text(aes(label = cases), vjust = 1.5, color = "white", size=2.5, fontface="bold") +  
 labs(title = "Figure 2. Monthly Confirmed Cases of Disease X in County Y, 2022-2023",  
 x = "Date of Positive Test",  
 y = "Confirmed Cases") +  
 scale\_x\_date(labels = custom\_labels, breaks = seq(start\_date, end\_date, by = "1 month")) +  
 theme\_light()  
  
# Save plot as png image   
ggsave("output\\Figure 2 Dataset A.png", plot = fig2, width = 10, height = 8, dpi = 500)  
  
# Print image here  
knitr::include\_graphics("output\\Figure 2 Dataset A.png")



Step 6 - Make some stratified weekly plots.

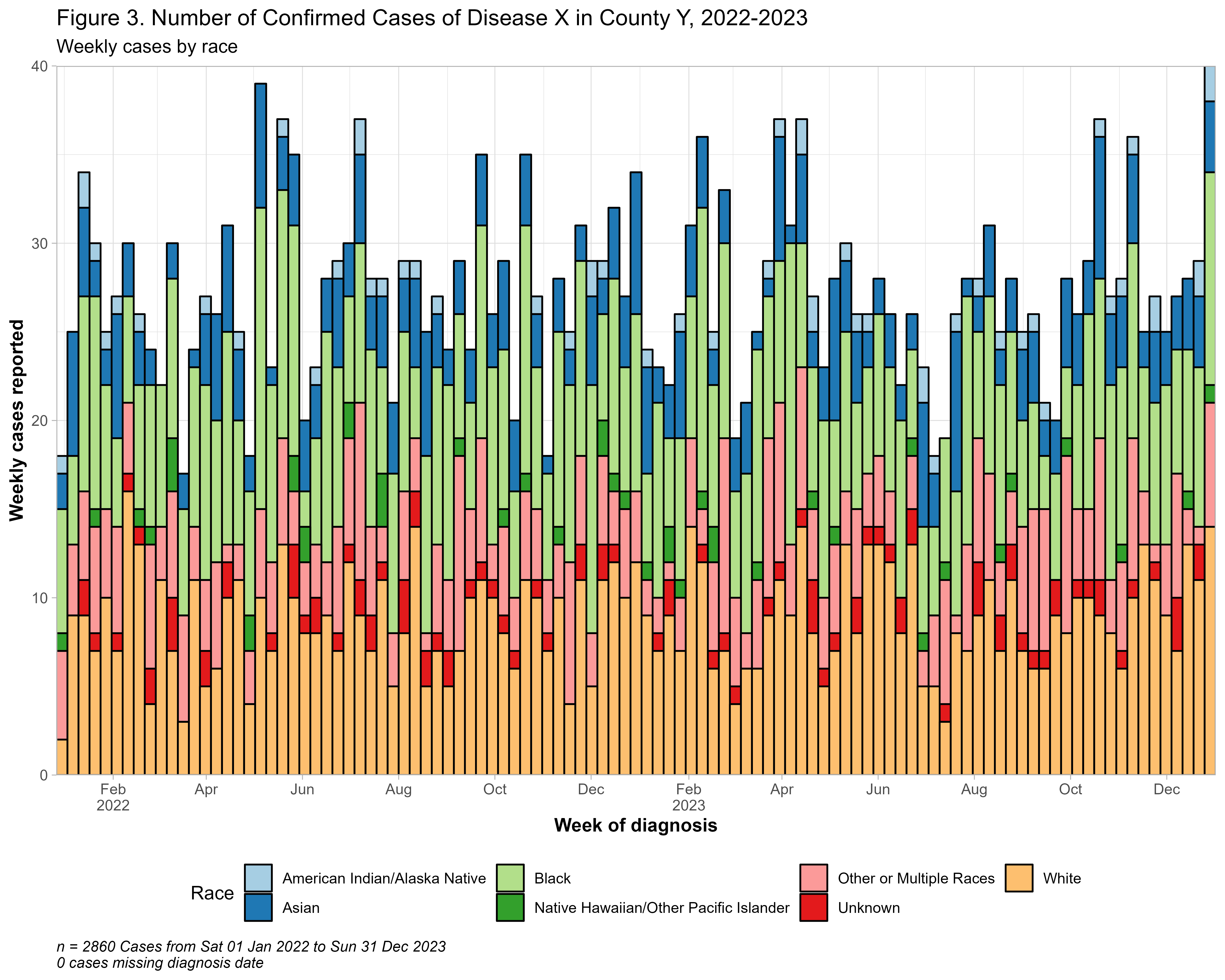
*Figures 3 and 4 below were adapted from Batra, Neale, et al. The Epidemiologist R Handbook. 2021.*

First, let’s make a weekly plot stratified by race. We will go back to using the original file here so we have all of our other grouping variables.

Note that our race categories are alphabetical here, we would have to reorder our factor variable to put these in a different order. You can use **forcats::fct\_relevel** for that.

For the weekly breaks, week\_start=1 is based on Monday (7 would be Sunday). Floor and ceiling options are used to create the start and end of the sequence from Mon to Sun. In the histogram, closed = “left” counts cases from the starting point (Mon). The labs dynamic caption allows us to stamp values from out dataset in the text.

# Create breaks by week  
breaks\_by\_week <- seq.Date(  
 from = floor\_date(min(as.Date(dataA$pos\_test\_date, na.rm=T)), "week", week\_start = 1),   
 to = ceiling\_date(max(as.Date(dataA$pos\_test\_date, na.rm=T)), "week", week\_start = 1),   
 by = "week")  
  
# Plot   
fig3 <- ggplot(dataA) +   
 geom\_histogram(  
 mapping = aes(  
 x = as.Date(pos\_test\_date),  
 group = race\_cat,  
 fill = race\_cat),  
 color = "black",  
 breaks = breaks\_by\_week,  
 closed = "left") +  
 scale\_x\_date(  
 expand = c(0,0),   
 date\_breaks = "2 months",   
 date\_minor\_breaks = "1 month",   
 label = scales::label\_date\_short()) +   
 scale\_y\_continuous(expand = c(0,0)) +  
 scale\_fill\_brewer(palette = "Paired", name = "Race") +   
 theme\_light() +  
 theme(plot.caption = element\_text(face = "italic", hjust = 0),   
 axis.title = element\_text(face = "bold"),  
 legend.position = "bottom",  
 strip.text = element\_text(face = "bold", size = 10),  
 strip.background = element\_rect(fill = "grey")) +   
 labs(title = "Figure 3. Number of Confirmed Cases of Disease X in County Y, 2022-2023",  
 subtitle = "Weekly cases by race",  
 fill = "Race",  
 x = "Week of diagnosis",  
 y = "Weekly cases reported",  
 caption = stringr::str\_glue("n = {nrow(dataA)} Cases from {format(min(dataA$pos\_test\_date, na.rm=T), format = '%a %d %b %Y')} to {format(max(dataA$pos\_test\_date, na.rm=T), format = '%a %d %b %Y')}\n{nrow(dataA %>% filter(is.na(pos\_test\_date)))} cases missing diagnosis date"))  
  
# Save plot as png image   
ggsave("output\\Figure 3 Dataset A.png", plot = fig3, width = 10, height = 8, dpi = 500)  
  
# Print image here  
knitr::include\_graphics("output\\Figure 3 Dataset A.png")

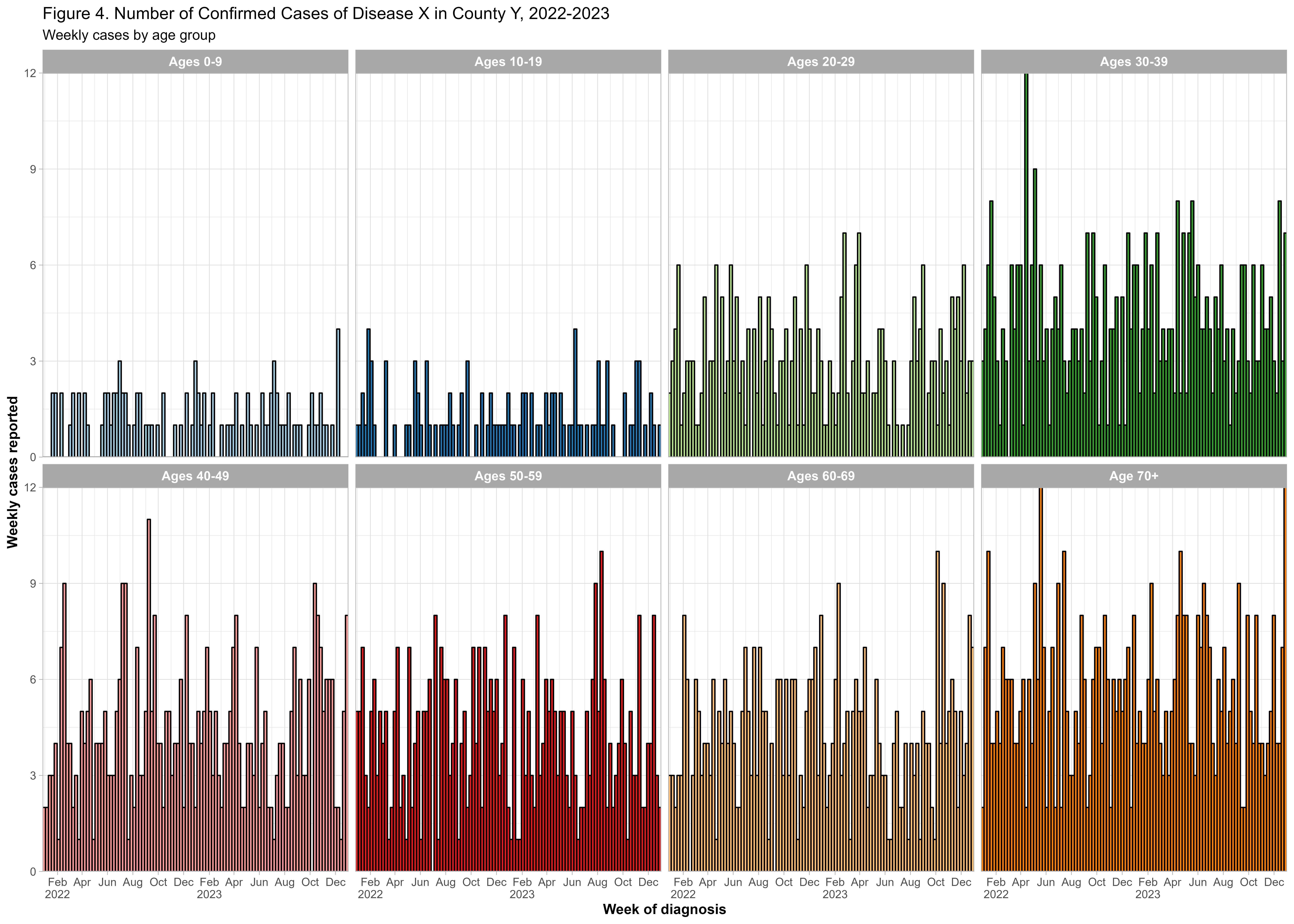


Now let’s make one more stratified by age group. We will use facets for this to get mini plots for each age group.

Here just to make the facets simpler, we will exclude the records with NA values for age.

The **gghighlight::gghighlight()** option is commented out, but this will put an outline of the full epidemic in the background behind the age group plots. You may need to rescale your image if you want to use this.

# Get rid of records without age  
dataA\_rev <- dataA %>% filter(!is.na(age\_group))  
  
# Labels for facets  
my\_labels <- as\_labeller(c(  
 "0-9" = "Ages 0-9",  
 "10-19" = "Ages 10-19",  
 "20-29" = "Ages 20-29",  
 "30-39" = "Ages 30-39",  
 "40-49" = "Ages 40-49",  
 "50-59" = "Ages 50-59",  
 "60-69" = "Ages 60-69",  
 "70+" = "Age 70+"))  
  
# Create breaks by week   
breaks\_by\_week <- seq.Date(  
 from = floor\_date(min(as.Date(dataA\_rev$pos\_test\_date, na.rm=T)), "week", week\_start = 1),   
 to = ceiling\_date(max(as.Date(dataA\_rev$pos\_test\_date, na.rm=T)), "week", week\_start = 1),   
 by = "week")  
  
# Plot   
fig4 <- ggplot(dataA\_rev) +   
 geom\_histogram(  
 mapping = aes(  
 x = as.Date(pos\_test\_date),  
 group = age\_group,  
 fill = age\_group),  
 color = "black",  
 breaks = breaks\_by\_week,  
 closed = "left") +  
 #gghighlight::gghighlight() +  
 scale\_x\_date(  
 expand = c(0,0),   
 date\_breaks = "2 months",   
 date\_minor\_breaks = "1 month",   
 label = scales::label\_date\_short()) +   
 scale\_y\_continuous(expand = c(0,0)) +  
 scale\_fill\_brewer(palette = "Paired", name = "age\_group") +   
 theme\_light() +  
 theme(plot.caption = element\_text(face = "italic", hjust = 0),   
 axis.title = element\_text(face = "bold"),  
 legend.position = "none",  
 strip.text = element\_text(face = "bold", size = 10),  
 strip.background = element\_rect(fill = "darkgray")) +   
 facet\_wrap(~age\_group,   
 ncol = 4,   
 strip.position = "top",  
 labeller = my\_labels) +   
 labs(title = "Figure 4. Number of Confirmed Cases of Disease X in County Y, 2022-2023",  
 subtitle = "Weekly cases by age group",  
 fill = "Age Group",  
 x = "Week of diagnosis",  
 y = "Weekly cases reported")  
  
# Save plot as png image   
ggsave("output\\Figure 4 Dataset A.png", plot = fig4, width = 14, height = 10, dpi = 500)  
  
# Print image here  
knitr::include\_graphics("output\\Figure 4 Dataset A.png")



This is the end of R Lab 1. R Lab 2 will pick up from here and work on some tables and figures similar to those you created in your Week 4 lab.