Intro to Data Science - HW 6

Copyright Jeffrey Stanton 2022, Jeffrey Saltz, and Jasmina Tacheva

```
# Enter your name here: Lily Coyle
```

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this homework by myself, with help from the book and the professor.
```

This module: Data visualization is important because many people can make sense of data more easily when it is presented in graphic form. As a data scientist, you will have to present complex data to decision makers in a form that makes the data interpretable for them. From your experience with Excel and other tools, you know that there are a variety of **common data visualizations** (e.g., pie charts). How many of them can you name?

The most powerful tool for data visualization in R is called **ggplot**. Written by computer/data scientist **Hadley Wickham**, this ** graphics grammar ** tool builds visualizations in layers. This method provides immense flexibility, but takes a bit of practice to master.

Step 1: Make a copy of the data

A. Read the **New York State COVID Testing** dataset we used in HW 3 & 4 from this URL: https://data-science-intro.s3.us-east-2.amazonaws.com/NYS_COVID_Testing_.csv (https://data-science-intro.s3.us-east-2.amazonaws.com/NYS_COVID_Testing_.csv)
into a new dataframe called **df**.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.6     ✓ dplyr   1.0.7
## ✓ tidyr   1.2.0     ✓ stringr 1.4.0
## ✓ readr   2.1.2     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
df <- read_csv("https://data-science-intro.s3.us-east-2.amazonaws.com/NYS_COVID_Testi
ng_.csv")
```

```
## Rows: 7383 Columns: 7
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): TestDate, AgeGroup, AgeCategory
## dbl (4): PositiveCases, TotalTests, Year, id
```

```
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

B. Your dataframe, **df**, contains a so-called **multivariate time series**: a sequence of measurements on COVID tests and results captured repeatedly over time (March 2020 - January 2022). Familiarize yourself with the nature of the time variable **TestDate**.
How often were these measurements taken (in other words, at what frequency were the variables measured)? Put your answer in a comment.

```
#The measurements were taken three times a day. Each age group was taken once a day.
```

C. What is the data type of **TestDate**? Explain in a comment.

```
#The data type of TestDate is character.
```

D. To properly display the **TestDate** values as dates in our plots, we need to convert **TestDate** to date format with the **as.Date()** function. Run the code below and check the data type of the variable again to make sure it is not coded as text anymore:

```
df$TestDate<-as.Date(df$TestDate, format = "%m/%d/%Y")
```

```
str(df)
```

```
## spec_tbl_df [7,383 × 7] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ TestDate     : Date[1:7383], format: "2020-03-04" "2020-03-04" ...
##  $ AgeGroup     : chr [1:7383] "1 to 4" "5 to 19" "< 1" "1 to 4" ...
##  $ PositiveCases: num [1:7383] 0 0 0 0 6 0 1 0 8 2 ...
##  $ TotalTests   : num [1:7383] 1 3 1 7 17 2 10 3 52 18 ...
##  $ AgeCategory  : chr [1:7383] "children" "children" "children" "children" ...
##  $ Year         : num [1:7383] 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020
...
##  $ id           : num [1:7383] 8 12 16 17 22 27 32 37 42 47 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   TestDate = col_character(),
##   ..   AgeGroup = col_character(),
##   ..   PositiveCases = col_double(),
##   ..   TotalTests = col_double(),
##   ..   AgeCategory = col_character(),
##   ..   Year = col_double(),
##   ..   id = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Step 2: Clean up the NAs and create subsets

A. It is always good practice, when you first start working with a dataset, to explore it for missing values. Check the **TotalTests** and **PositiveCases** for missing values. Are there any? What does empty output suggest about the number of missing observations?

Hint: use *is.na()*

```
df[is.na(df$TotalTests),] #there are 22 rows with missing observations
```

```
## # A tibble: 22 × 7
##    TestDate   AgeGroup PositiveCases TotalTests AgeCategory         Year    id
##    <date>     <chr>            <dbl>      <dbl> <chr>              <dbl> <dbl>
##  1 2020-04-22 1 to 4              24         NA children            2020   542
##  2 2020-04-22 5 to 19            289         NA children            2020   547
##  3 2020-04-22 < 1                  4         NA children            2020   552
##  4 2021-12-17 1 to 4             533         NA children            2021  7186
##  5 2021-12-17 5 to 19           3986         NA children            2021  7191
##  6 2021-12-17 < 1                146         NA children            2021  7196
##  7 2020-04-22 35 to 44           992         NA middle-aged_adults  2020   545
##  8 2020-04-22 45 to 54          1089         NA middle-aged_adults  2020   546
##  9 2021-12-17 35 to 44          3330         NA middle-aged_adults  2021  7189
## 10 2021-12-17 45 to 54          2138         NA middle-aged_adults  2021  7190
## # … with 12 more rows
```

```
df[is.na(df$PositiveCases),] #there are 11 rows with 0 positive cases
```

```
## # A tibble: 11 × 7
##    TestDate   AgeGroup PositiveCases TotalTests AgeCategory         Year    id
##    <date>     <chr>            <dbl>      <dbl> <chr>              <dbl> <dbl>
##  1 2021-12-15 1 to 4              NA       9694 children            2021  7164
##  2 2021-12-15 5 to 19             NA      54492 children            2021  7169
##  3 2021-12-15 < 1                 NA       1430 children            2021  7174
##  4 2021-12-15 35 to 44            NA      40170 middle-aged_adults  2021  7167
##  5 2021-12-15 45 to 54            NA      32945 middle-aged_adults  2021  7168
##  6 2021-12-15 55 to 64            NA      29690 older_adults        2021  7170
##  7 2021-12-15 65 to 74            NA      16472 older_adults        2021  7171
##  8 2021-12-15 75 to 84            NA       8168 older_adults        2021  7172
##  9 2021-12-15 85 +                NA       6055 older_adults        2021  7173
## 10 2021-12-15 20 to 24            NA      24679 young_adults        2021  7165
## 11 2021-12-15 25 to 34            NA      54164 young_adults        2021  7166
```

B. There is an R package called **imputeTS** specifically designed to repair missing values in time series data. We will use this instead of the simpler way, **mean substitution**, because it tends to be more accurate.
The **na_interpolation()** function in this package takes advantage of a unique characteristic of time series data: neighboring points in time can be used to guess about a missing value in between.
Use this function on each of the two numeric variables in **df** and don't forget to **update** them by overwriting them with the output of the **na_interpolation()** function.

```
library("imputeTS")
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
df$TotalTests <- na_interpolation(df$TotalTests)
df$PositiveCases <- na_interpolation(df$PositiveCases)
```

C. Run the code from A to make sure there is no more missing data:

```
df[is.na(df$TotalTests),]
```

```
## # A tibble: 0 × 7
## # … with 7 variables: TestDate <date>, AgeGroup <chr>, PositiveCases <dbl>,
## #   TotalTests <dbl>, AgeCategory <chr>, Year <dbl>, id <dbl>
```

```
df[is.na(df$PositiveCases),]
```

```
## # A tibble: 0 × 7
## # … with 7 variables: TestDate <date>, AgeGroup <chr>, PositiveCases <dbl>,
## #   TotalTests <dbl>, AgeCategory <chr>, Year <dbl>, id <dbl>
```

D. As we've done before, let's create a new variable which is the ratio of **PositiveCases** to **TotalTests** - save it as an additional variable in **df** called **PositivityRate**:

```
df$PositivityRate <- df$PositiveCases / df$TotalTests
```

E. Create a subset of **df** containing **only the records for children.** Save it in a new dataframe called **dfChildren**. Make sure this new df has **2,010 observations and 6 variables**.

```
dfChildren <- df[df$AgeCategory == "children",]
```

F. Create a subset of **df** containing only the records for **young adults.** Save it in a new dataframe called **dfYA.**

```
dfYA <- df[df$AgeCategory == "young_adults",]
```

G. Using the same logic, create 2 more subsets of **df**: one containing only the records for **middle-aged adults** (call it **dfMA**), and another one with only the data of **older adults** - **dfOA.** After this step, you should have a total of 4 subsets: - dfChildren - dfYA - dfMA - dfOA

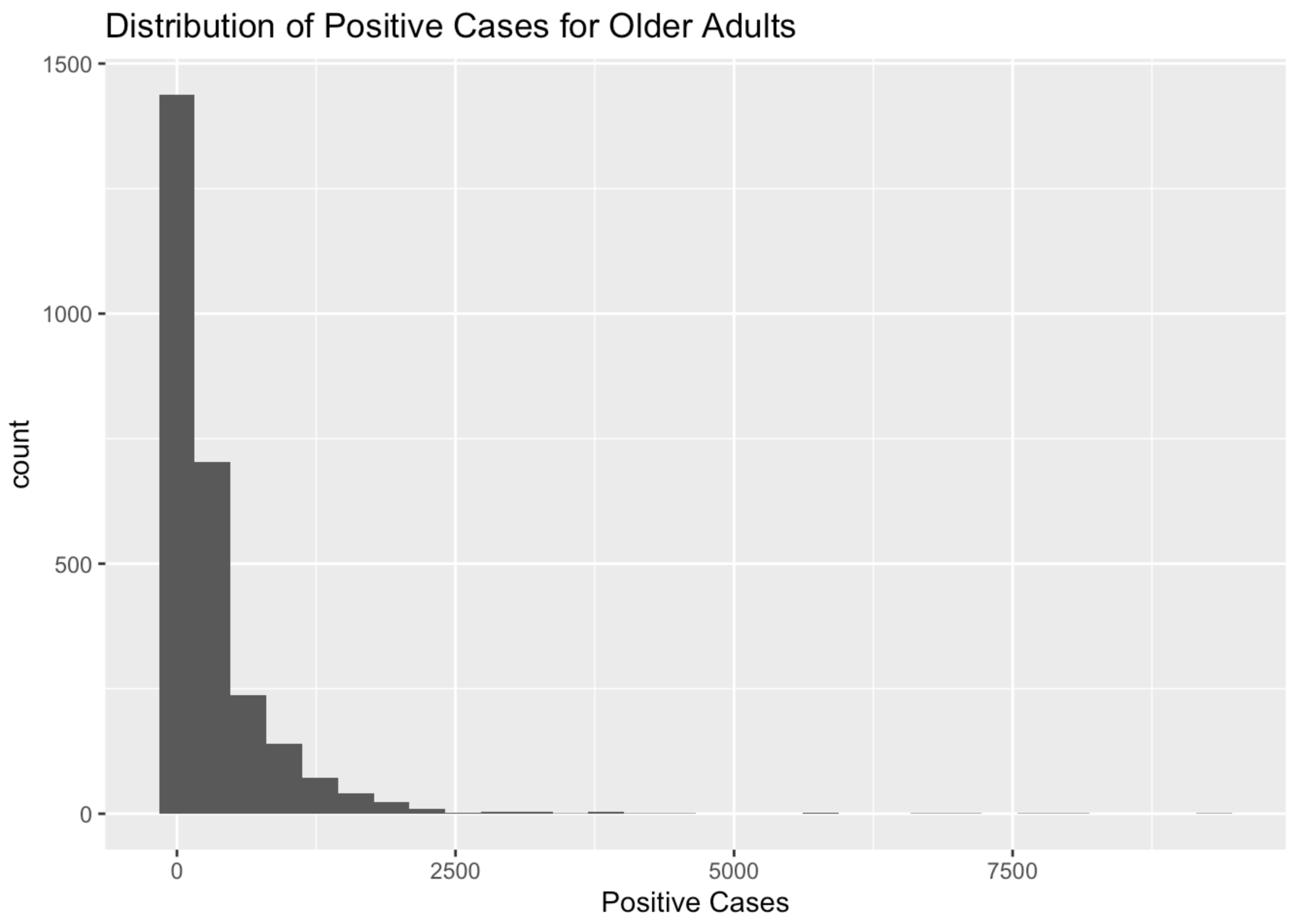
```
dfMA <- df[df$AgeCategory == "middle-aged_adults",]
dfOA <- df[df$AgeCategory == "older_adults",]
```

Step 3: Use ggplot to explore the distribution of each variable

Don t forget to install and library the ggplot2 package. Then:
A. Create a histogram for **PositiveCases** in the **dfOA** dataframe. Be sure to add a title and briefly describe what the histogram means in a comment.

```
histOA <- ggplot(dfOA, aes(x=PositiveCases)) + geom_histogram() + ylab("count") + xla
b("Positive Cases") + ggtitle("Distribution of Positive Cases for Older Adults")
histOA
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

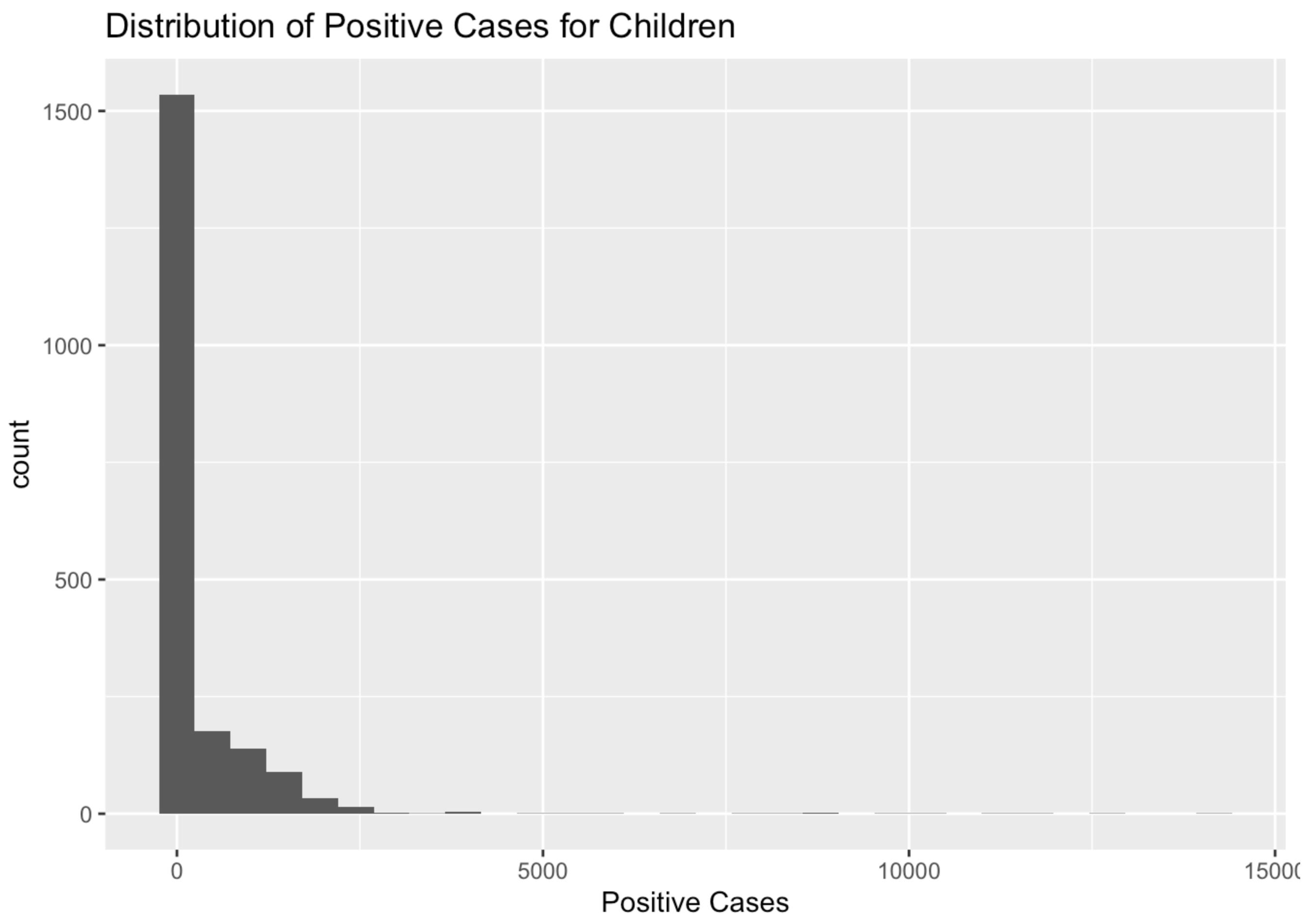


```
#The histogram means that the highest distribution of positive cases came at the star
t
```

B. Create histograms (using **ggplot**) of the **PositiveCases** variable in each of the other three subsets from Step 2G.
   For each histogram, comment on its shape - what information can we glean from it?

```
histChildren <- ggplot(dfChildren, aes(x=PositiveCases)) + geom_histogram() + ylab("c
ount") + xlab("Positive Cases") + ggtitle("Distribution of Positive Cases for Childre
n")
histChildren
```

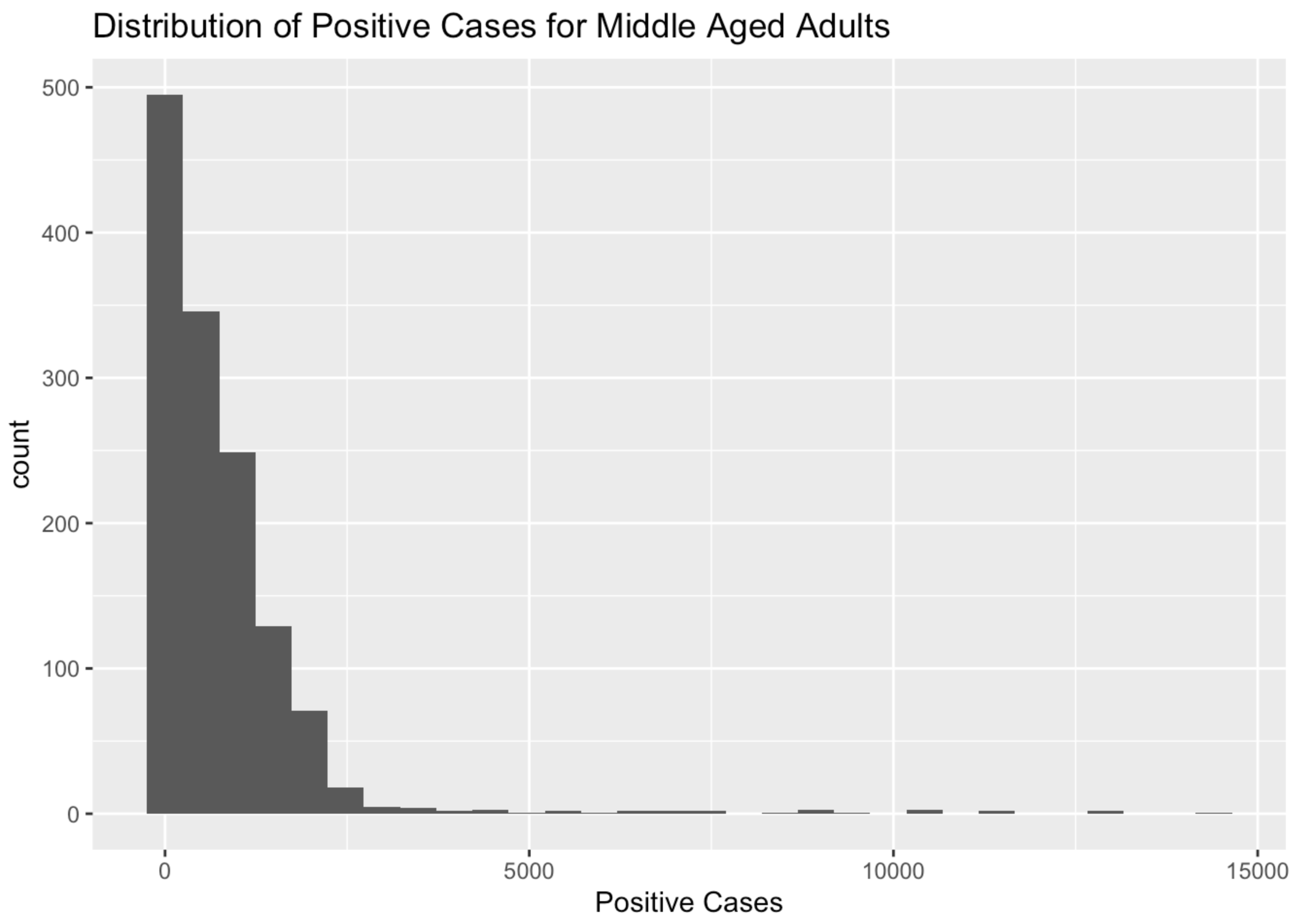
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
#Once again the highest distribution of positive cases came at the start and then beg
an to taper off and stabilize
```

```
histMA <- ggplot(dfMA, aes(x=PositiveCases)) + geom_histogram() + ylab("count") + xla
b("Positive Cases") + ggtitle("Distribution of Positive Cases for Middle Aged Adult
s")
histMA
```

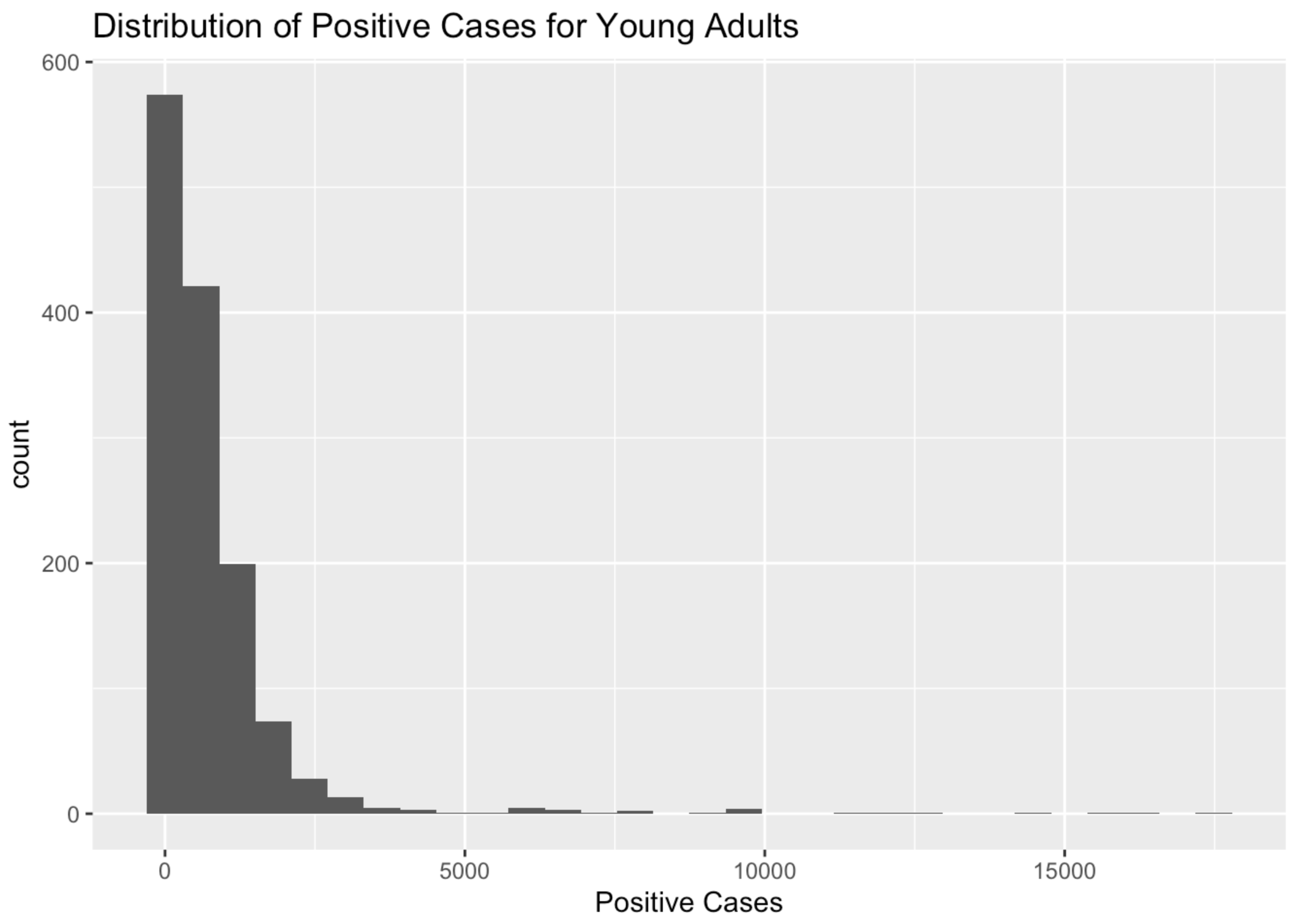
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
#In this case the spike was also at the start but it took longer for it to go down
```

```
histYA <- ggplot(dfYA, aes(x=PositiveCases)) + geom_histogram() + ylab("count") + xla
b("Positive Cases") + ggtitle("Distribution of Positive Cases for Young Adults")
histYA
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

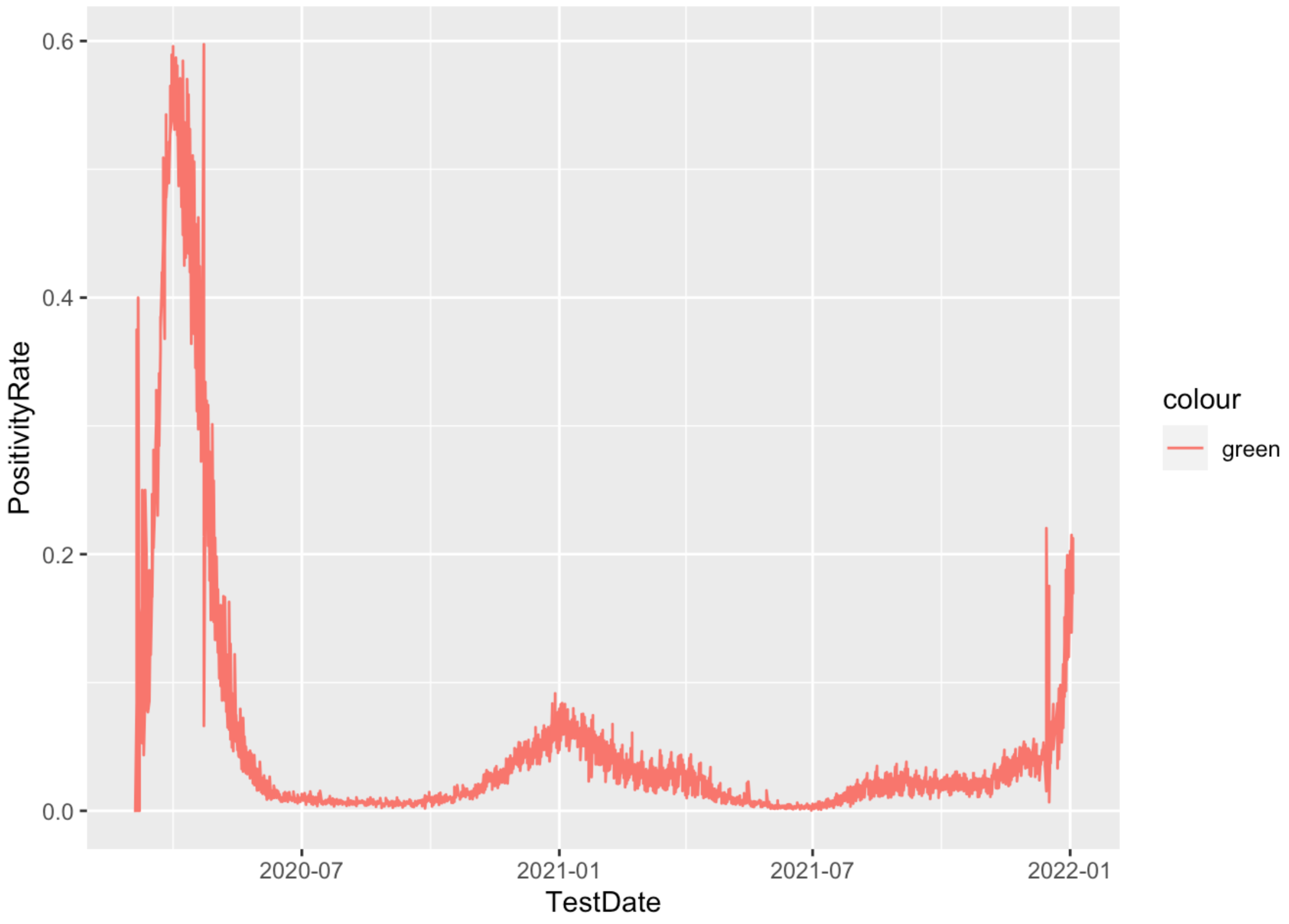


```
#Similar to the one above this one had a large spike right at the beginning and taper
ed off at a slower pace
```

Step 4: Explore how the data changes over time

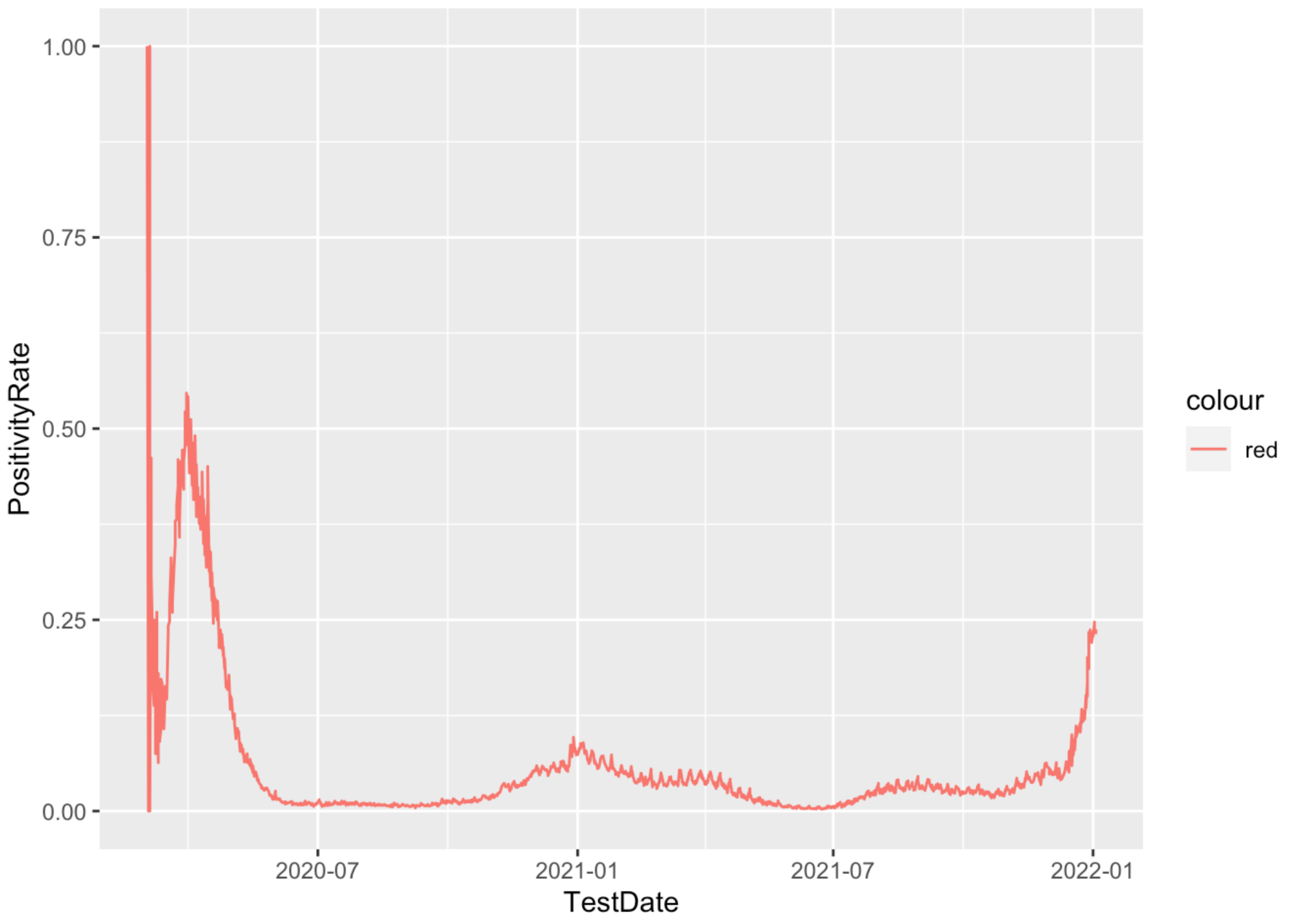
A. These data were collected in a period of almost 2 years. You can thus observe changes over time with the help of a line chart. Let's focus on the **dfOA** subset first:
Create a **line chart**, with **TestDate** on the X-axis and **PositivityRate** on the Y-axis.

```
lineOA <- ggplot(dfOA, aes(x= TestDate, y = PositivityRate)) + geom_line(aes(y=Positi
vityRate, color = "green"))
lineOA
```

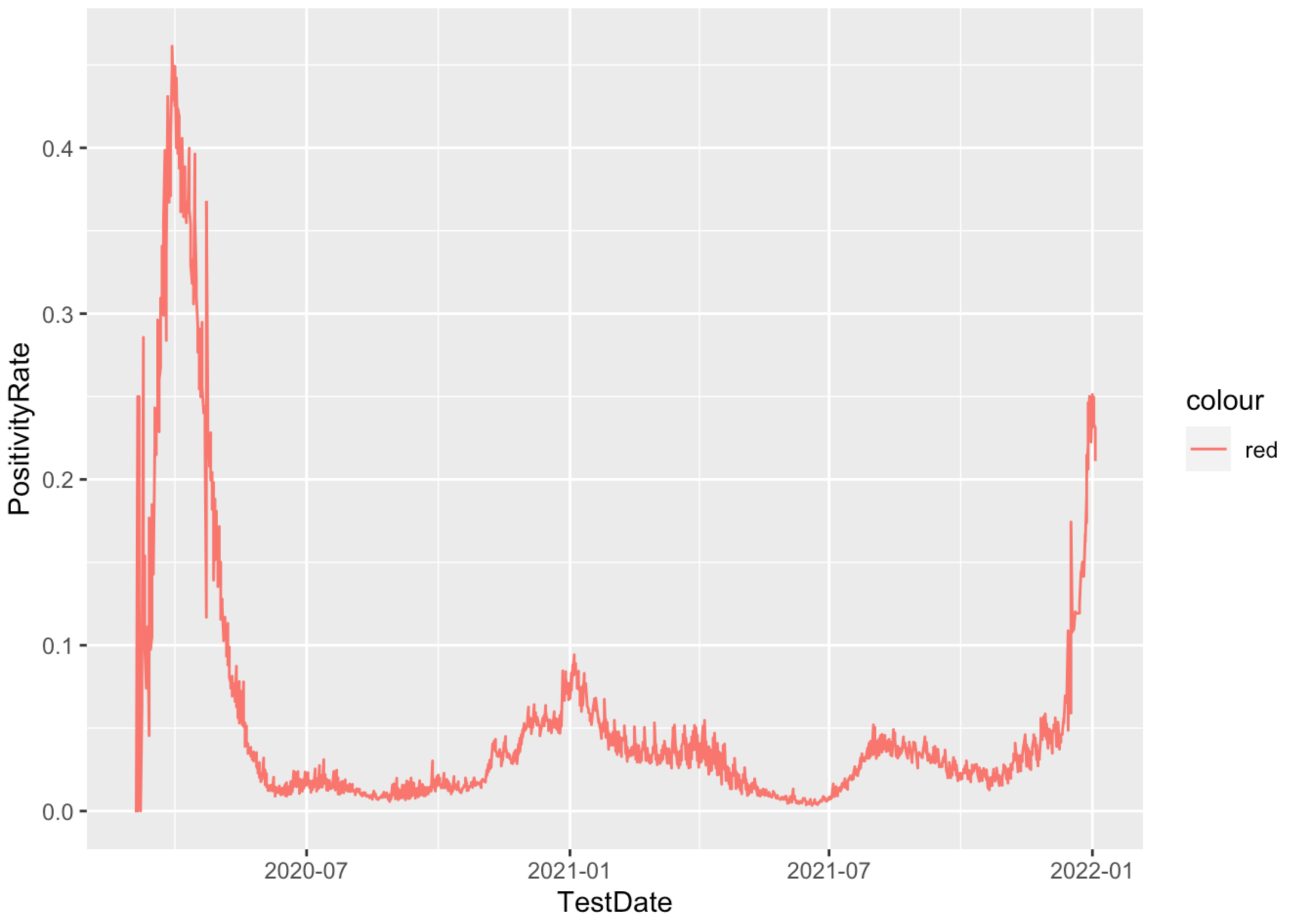


B. Next, create similar graphs for each of the other three subsets. Change the **color** of the line plots (any color you want).

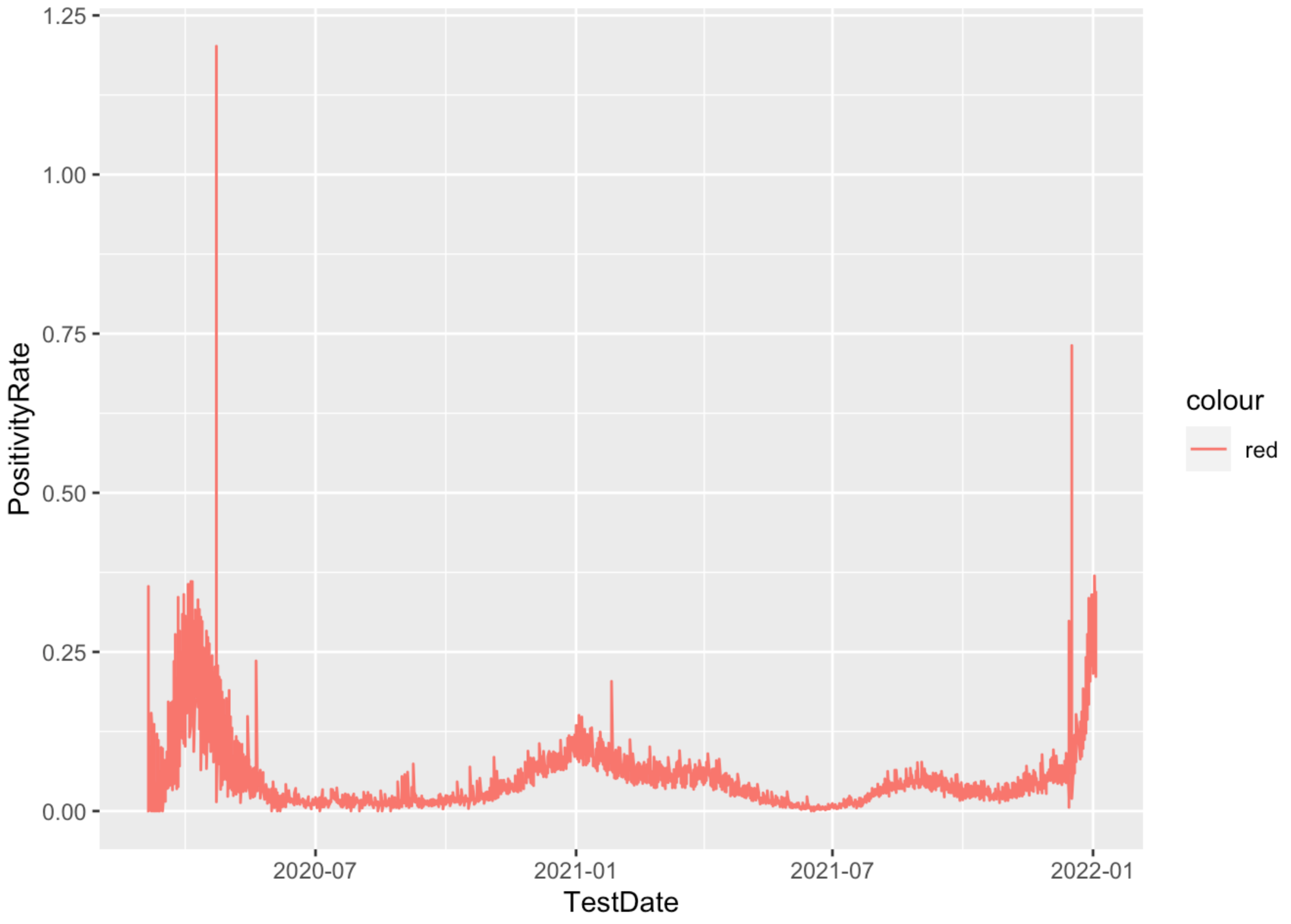
```
lineMA <- ggplot(dfMA, aes(x= TestDate, y = PositivityRate)) + geom_line(aes(y=Positi
vityRate, color = "red"))
lineMA
```



```
lineYA <- ggplot(dfYA, aes(x= TestDate, y = PositivityRate)) + geom_line(aes(y=Positi
vityRate, color = "red"))
lineYA
```



```
lineChildren <- ggplot(dfChildren, aes(x= TestDate, y = PositivityRate)) + geom_line
(aes(y=PositivityRate, color = "red"))
lineChildren
```

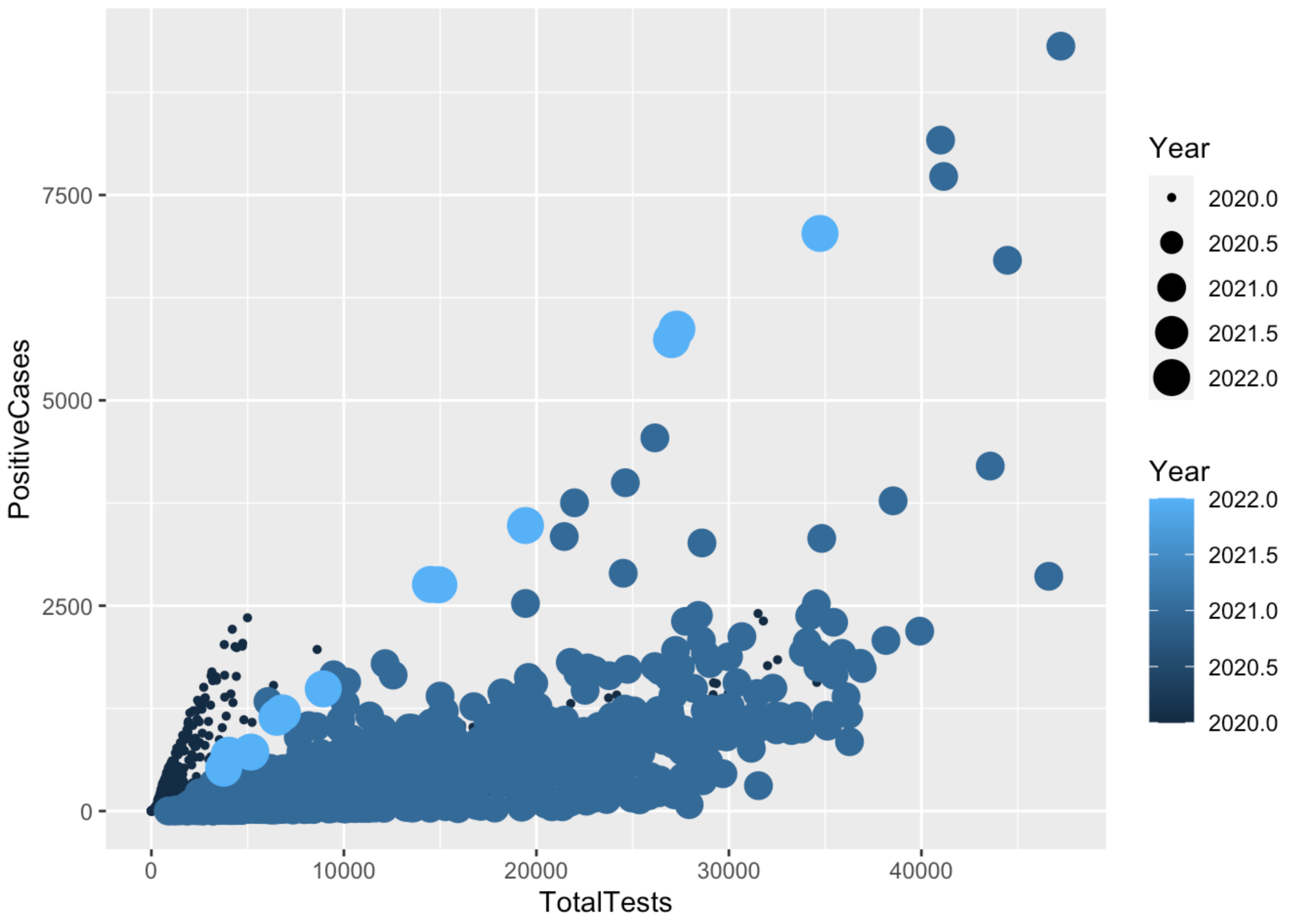


C. In a comment, talk about the insights you got from the line charts you created - can you spot any trends within and between the line charts?

```
#The biggest spike across all age groups came at the start of the pandemic and didn't
rise dramatically again until the start of omicron
```

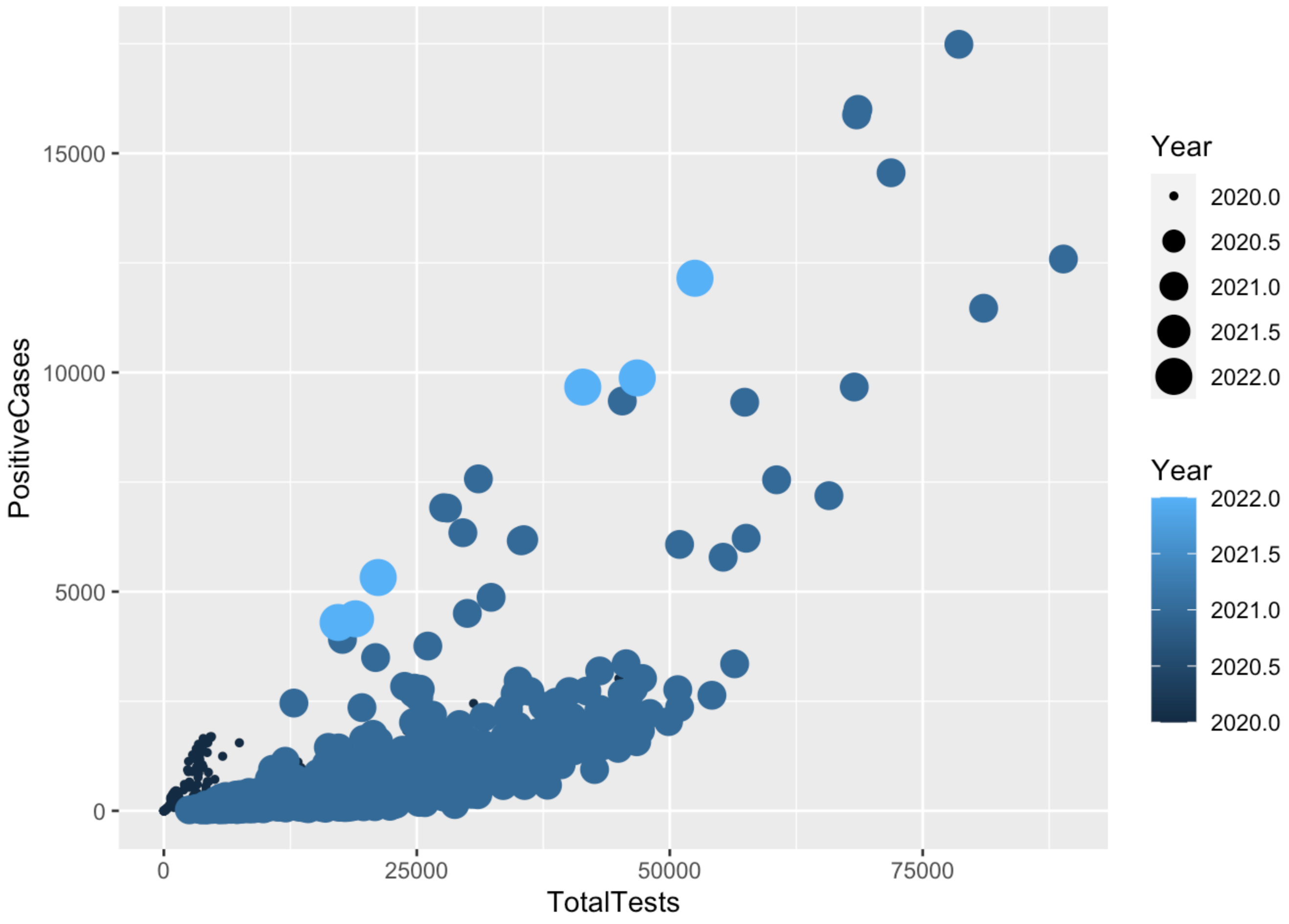
D. Finally, using the **dfOA** subset, create a **scatter plot**, showing **TotalTests** on the x axis, **PositiveCases** on the y axis, and having the **color and size** of the point represent **Year**.

```
scatterOA <- ggplot(dfOA, aes(x = TotalTests, y = PositiveCases, col = Year, size = Y
ear)) + geom_point()
scatterOA
```



E. Create a similar scatter plot for the **dfYA** subset.

```
scatterYA <- ggplot(dfYA, aes(x = TotalTests, y = PositiveCases, col = Year, size = Y
ear)) + geom_point()
scatterYA
```



F. Interpret these visualizations what insight do they provide?

```
#These visualizations show the number of positive covid tests in relation to the year
in which they were taken in. In some instances, closest to 2021, the cases were the h
ighest. However, it is evident the highest number of positive cases was in 2020 and i
n 2022 they are getting lower.
```