# Intro to Data Science - HW 6



```
# Enter your name here: Bhavya Shah
```

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

```
# 2. I did this homework with help from the book and the professor and these Internet sources:Go
ogle, javatpoint
```

**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 **ggplot2**. 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 1.3.2 ──
## ✔ ggplot2 3.4.1      ✔ purrr   1.0.1
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.3.0      ✔ stringr 1.5.0
## ✔ readr   2.1.3      ✔ forcats 1.0.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
df = readr::read_csv('https://data-science-intro.s3.us-east-2.amazonaws.com/NYS_COVID_Testing.cs
v')
```

```
## Rows: 7383 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): TestDate, AgeGroup, AgeCategory
## dbl (2): PositiveCases, TotalTests
##
## 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.
```

```
df$TestDate = stringr::str_replace(df$TestDate, '0021$', '2021')

rand_vec = sample(1:nrow(df), 50)
df$PositiveCases[rand_vec] = NA
df$TotalTests[rand_vec] = NA
```

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 tests for different age groups were conducted one time per day on average, and the overall
tests were conducted three times per day on average.
```

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

```
#The datatype of TestData is character, which stores date values.
```

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$TestDate)
```

```
##  Date[1:7383], format: "2020-03-02" "2020-03-03" "2020-03-03" "2020-03-03" "2020-03-03" ...
```

# 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()*

```
tt<-is.na(df$TotalTests)
sum(tt)
```

```
## [1] 50
```

```
pc<- is.na(df$PositiveCases)
sum(pc)
```

```
## [1] 50
```

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:

```
tt<-is.na(df$TotalTests)
sum(tt)
```

```
## [1] 0
```

```
tt<-is.na(df$PositiveCases)
sum(tt)
```

```
## [1] 0
```

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 8 variables**.

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

```
## tibble [2,010 × 6] (S3: tbl_df/tbl/data.frame)
##  $ TestDate      : Date[1:2010], format: "2020-03-04" "2020-03-04" ...
##  $ AgeGroup      : chr [1:2010] "1 to 4" "5 to 19" "< 1" "1 to 4" ...
##  $ PositiveCases : num [1:2010] 0 0 0 0 6 0 1 0 8 2 ...
##  $ TotalTests    : num [1:2010] 1 3 1 7 17 2 10 3 52 18 ...
##  $ AgeCategory   : chr [1:2010] "children" "children" "children" "children" ...
##  $ PositivityRate: num [1:2010] 0 0 0 0 0.353 ...
```

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=="children",]
```

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

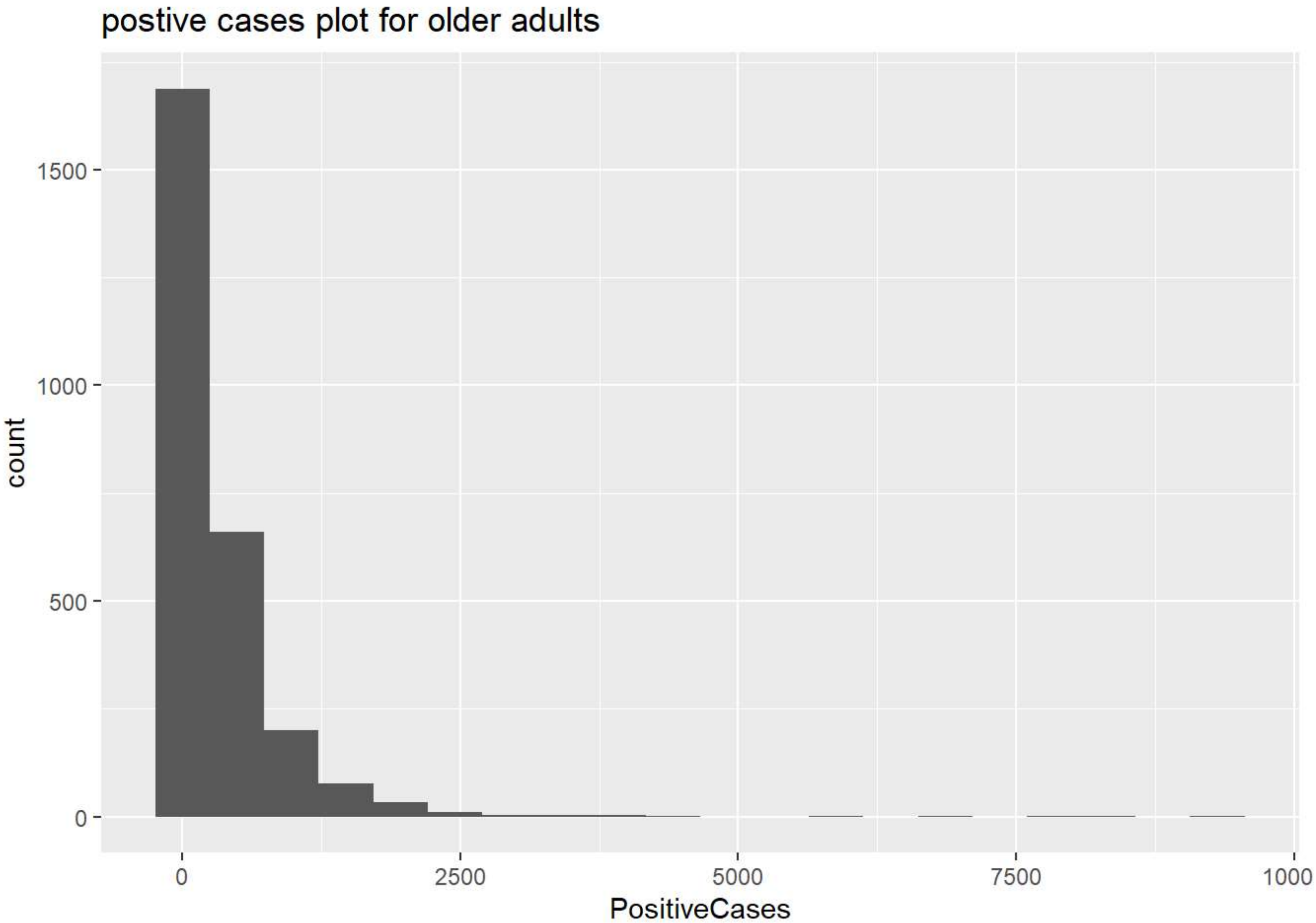
```
dfMA<-df[df$AgeCategory=="middle-aged_adults",]
dfOA<-df[df$AgeCategory=="senior_citizens",]
```

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

**Don t forget to install and library the ggplot2 package.** Then:

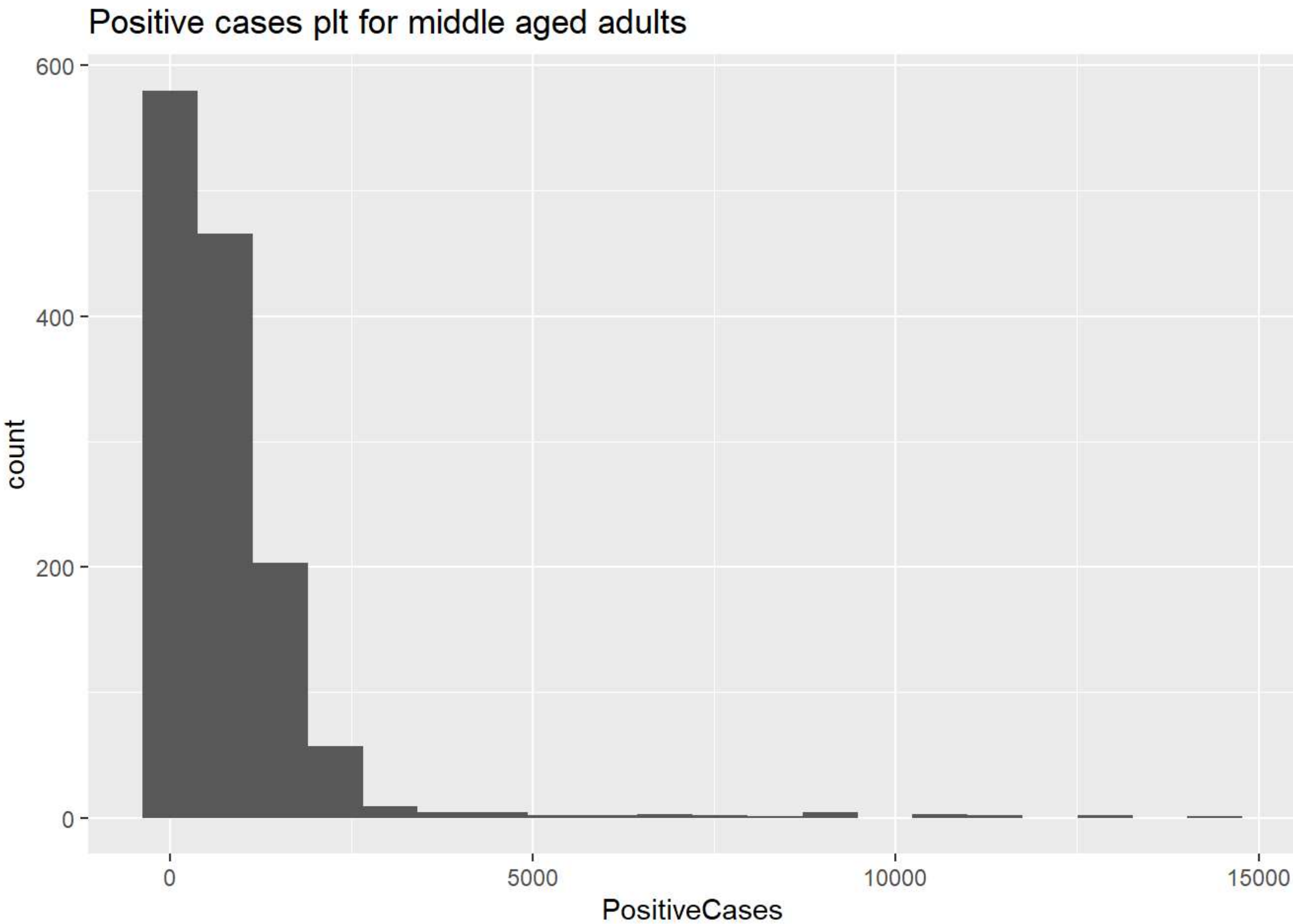
A. Create a histogram for **PositiveCases** in the **dfOA** dataframe (using **ggplot**). Be sure to add a title and briefly describe what the histogram means in a comment.

```
library(ggplot2)
ggplot(dfOA)+aes(x=PositiveCases)+geom_histogram(bins = 20)+ggtitle("postive cases plot for olde
r adults")
```

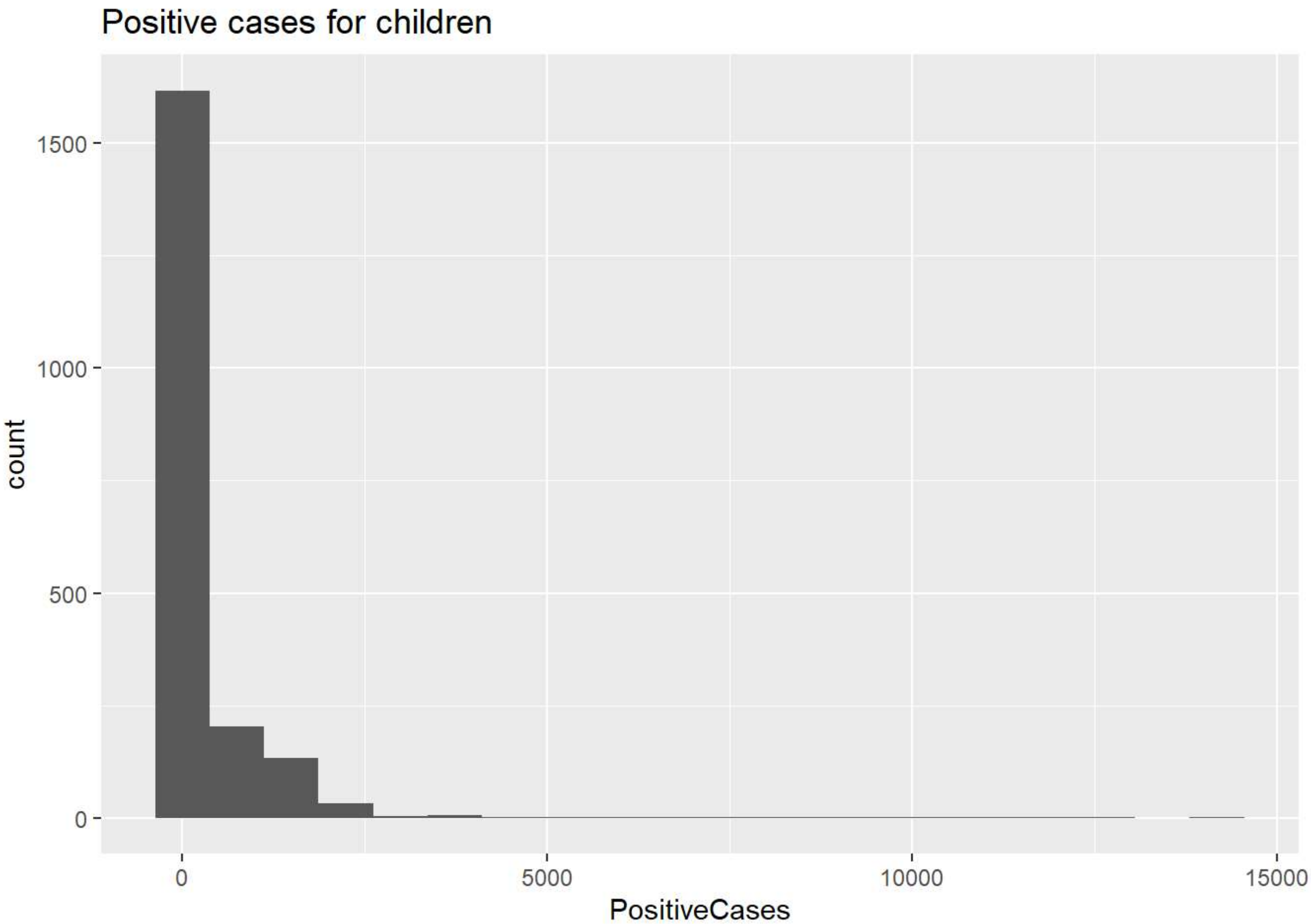



B. Create histograms (using **ggplot**) of the **PositiveCases** variable in each of the other three subsets from Step 2G.
   For each histogram, comment on its shape - what information can we glean from it?

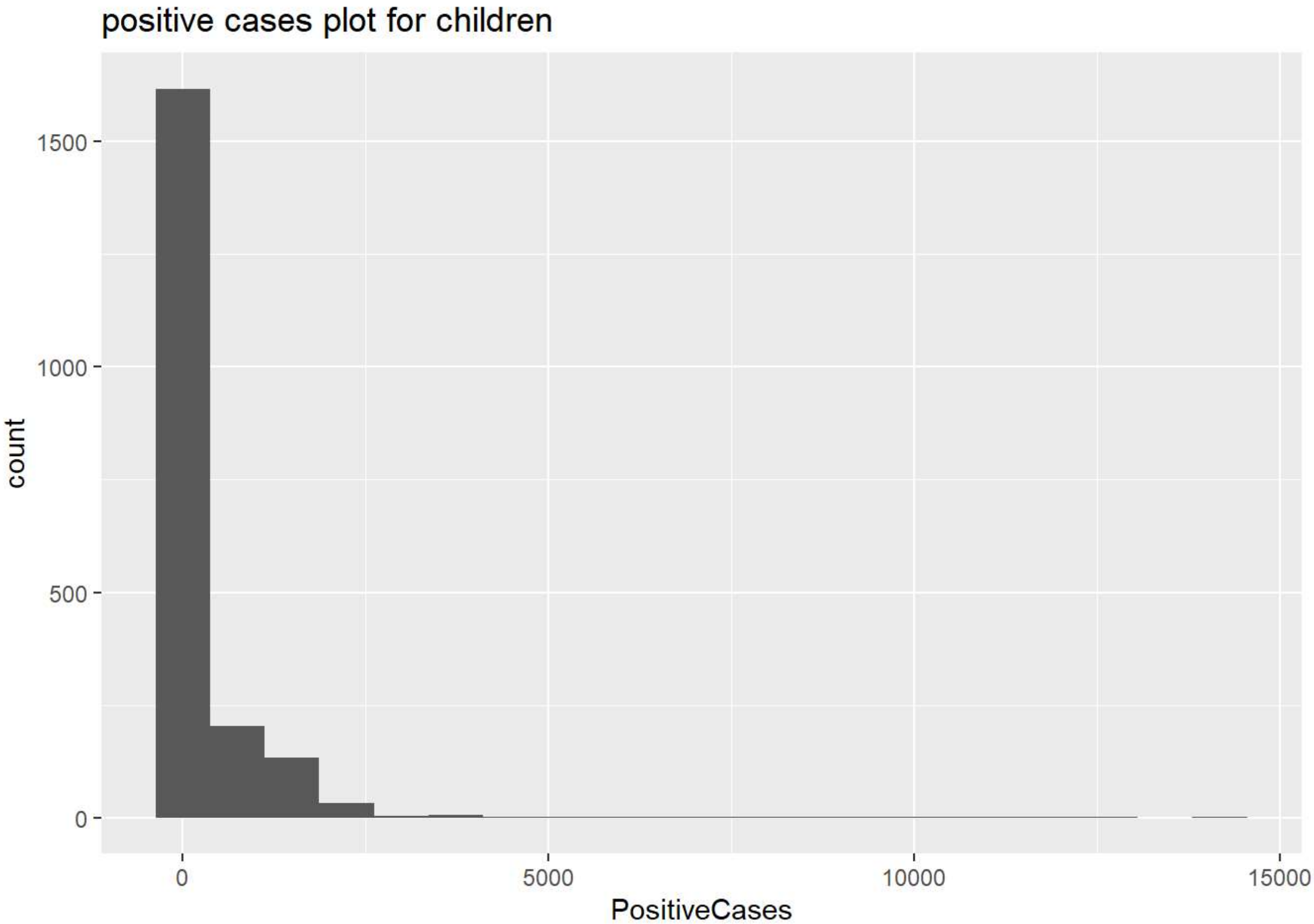
```
ggplot(dfMA)+aes(x=PositiveCases)+geom_histogram(bins=20)+ggtitle("Positive cases plt for middle
aged adults")
```




```
ggplot(dfYA)+aes(x=PositiveCases) + geom_histogram(bins=20)+ggtitle("Positive cases for childre
n")
```



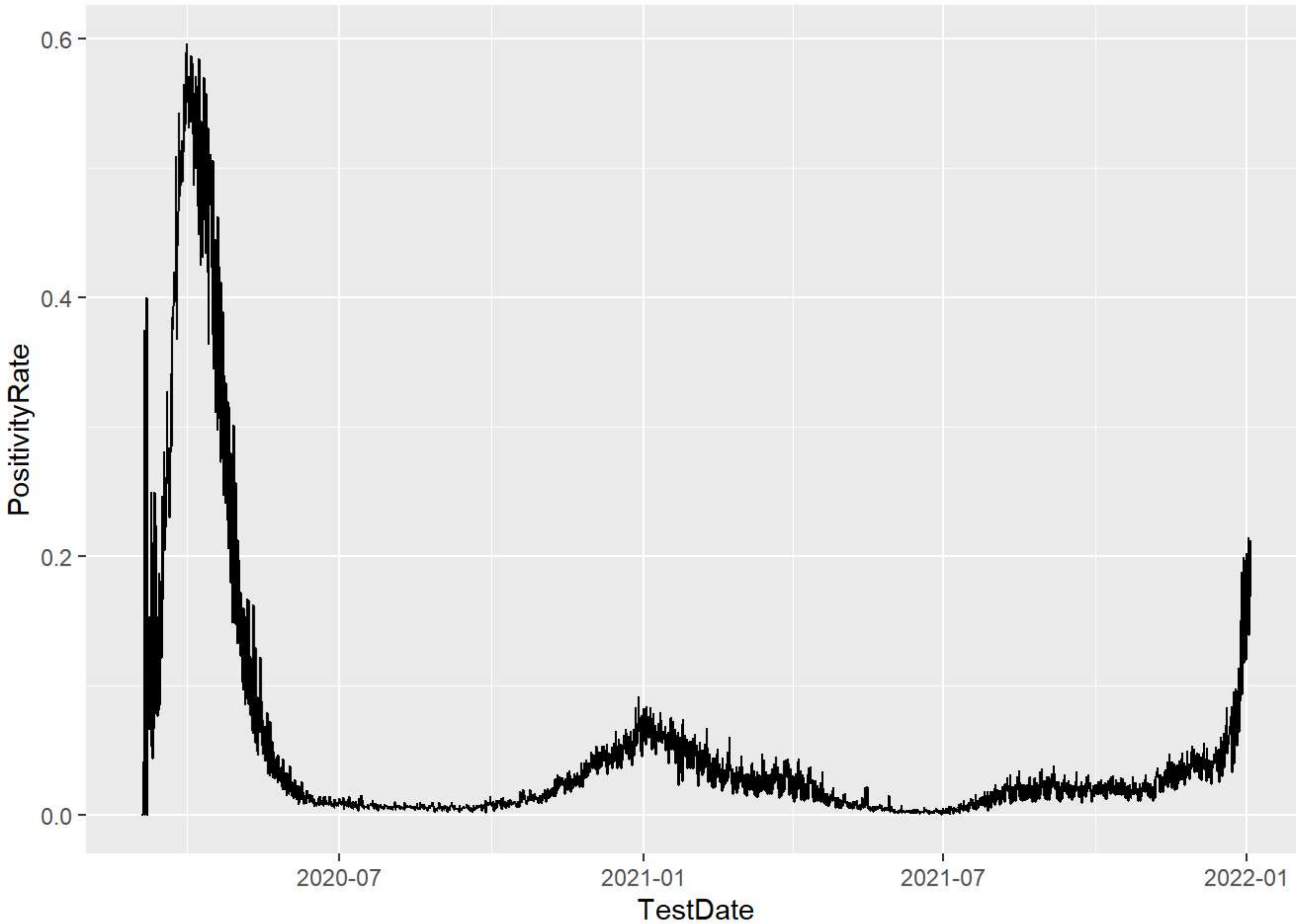

```
ggplot(dfChildren)+aes(x=PositiveCases)+geom_histogram(bins = 20)+ggtitle("positive cases plot f
or children")
```




# Step 4: Explore how the data changes over time

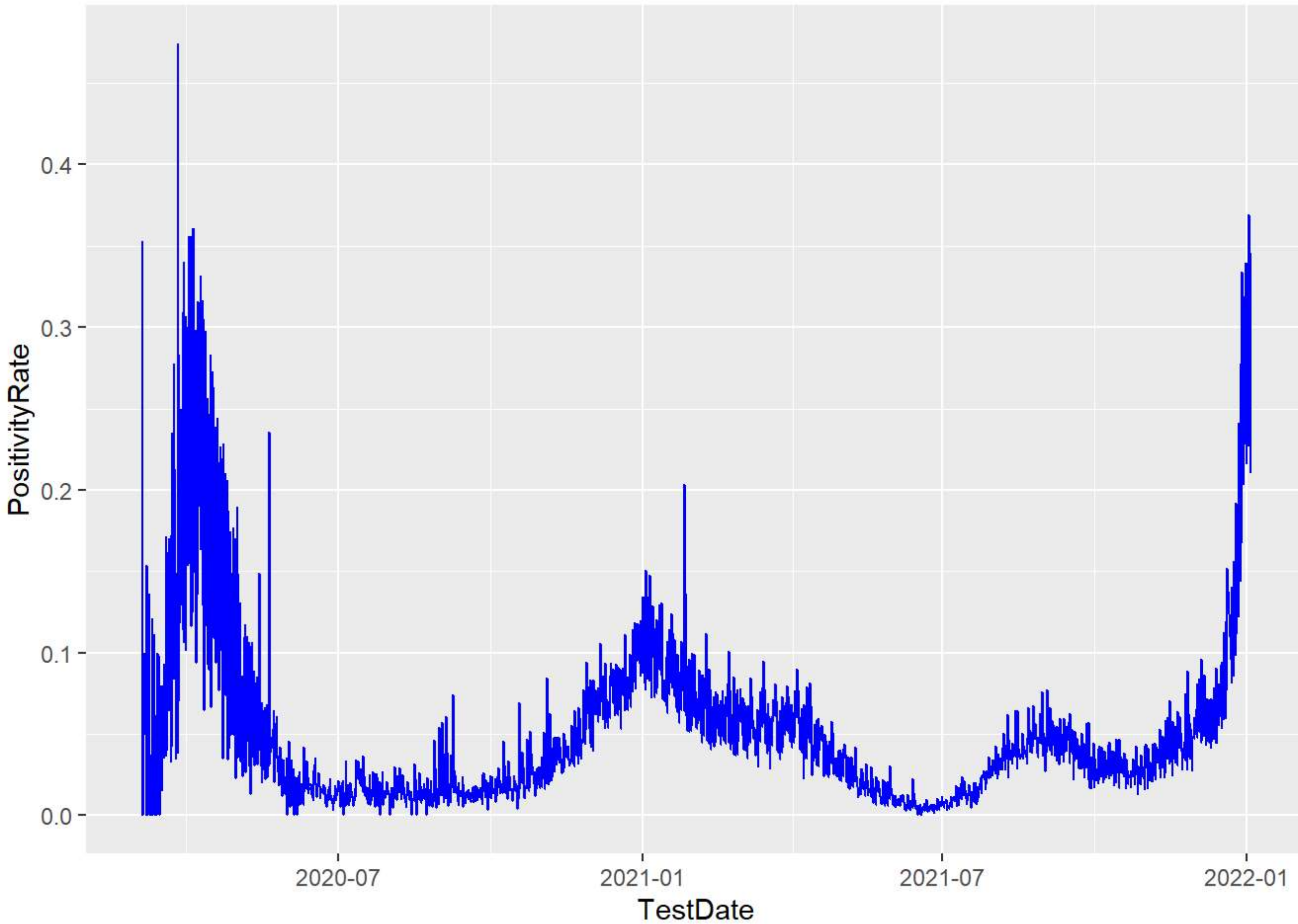
A. These data were collected in a period of almost 2 years. You can thus observe changes over time with the help of a line chart. Let's focus on the **dfOA** subset first:
   Create a **line chart**, with **TestDate** on the X-axis and **PositivityRate** on the Y-axis.

```
LineChartOA <- ggplot(dfOA, aes(x=TestDate, y=PositivityRate))+geom_line()
LineChartOA
```

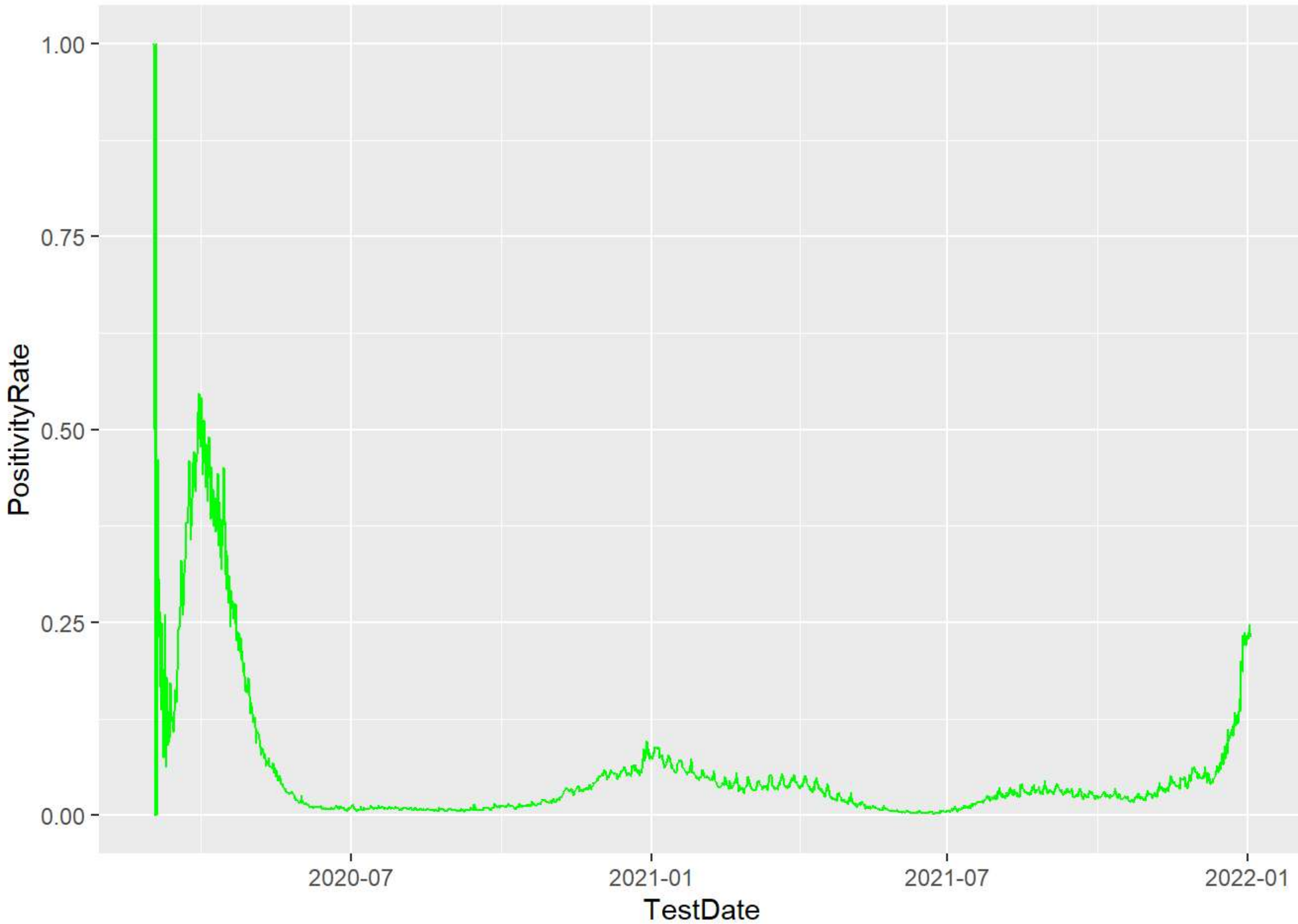



B. Next, create similar graphs for each of the other three subsets. Change the **color** of the line plots (any color you want).

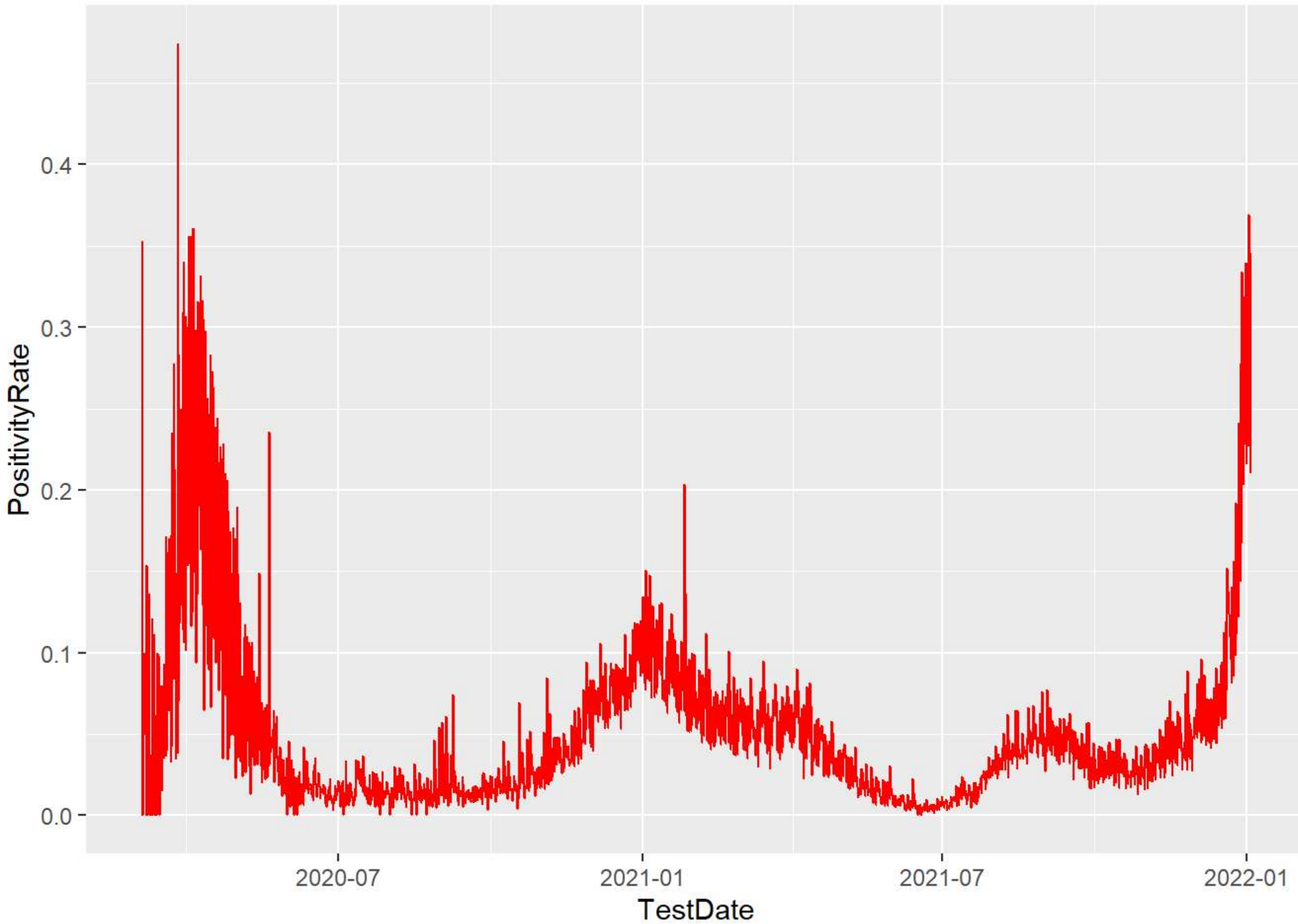
```
#Generated a line chart for dfYA dataframe
#aes x-axis as TestDate and y as PositivityRate with color as blue
LineChartYA <- ggplot(dfYA, aes(x=TestDate, y=PositivityRate))+
  geom_line(color="blue")
LineChartYA
```




```
#Generated a line chart for dfMA dataframe
#aes x-axis as TestDate and y as PositivityRate with color as green
LineChartMA <- ggplot(dfMA, aes(x=TestDate, y=PositivityRate))+
  geom_line(color="green")
LineChartMA
```




```
#Generated a line chart for dfChildren dataframe
#aes x-axis as TestDate and y as PositivityRate with color as red
LineChartChildren <- ggplot(dfChildren, aes(x=TestDate, y=PositivityRate))+
  geom_line(color="red")
LineChartChildren
```



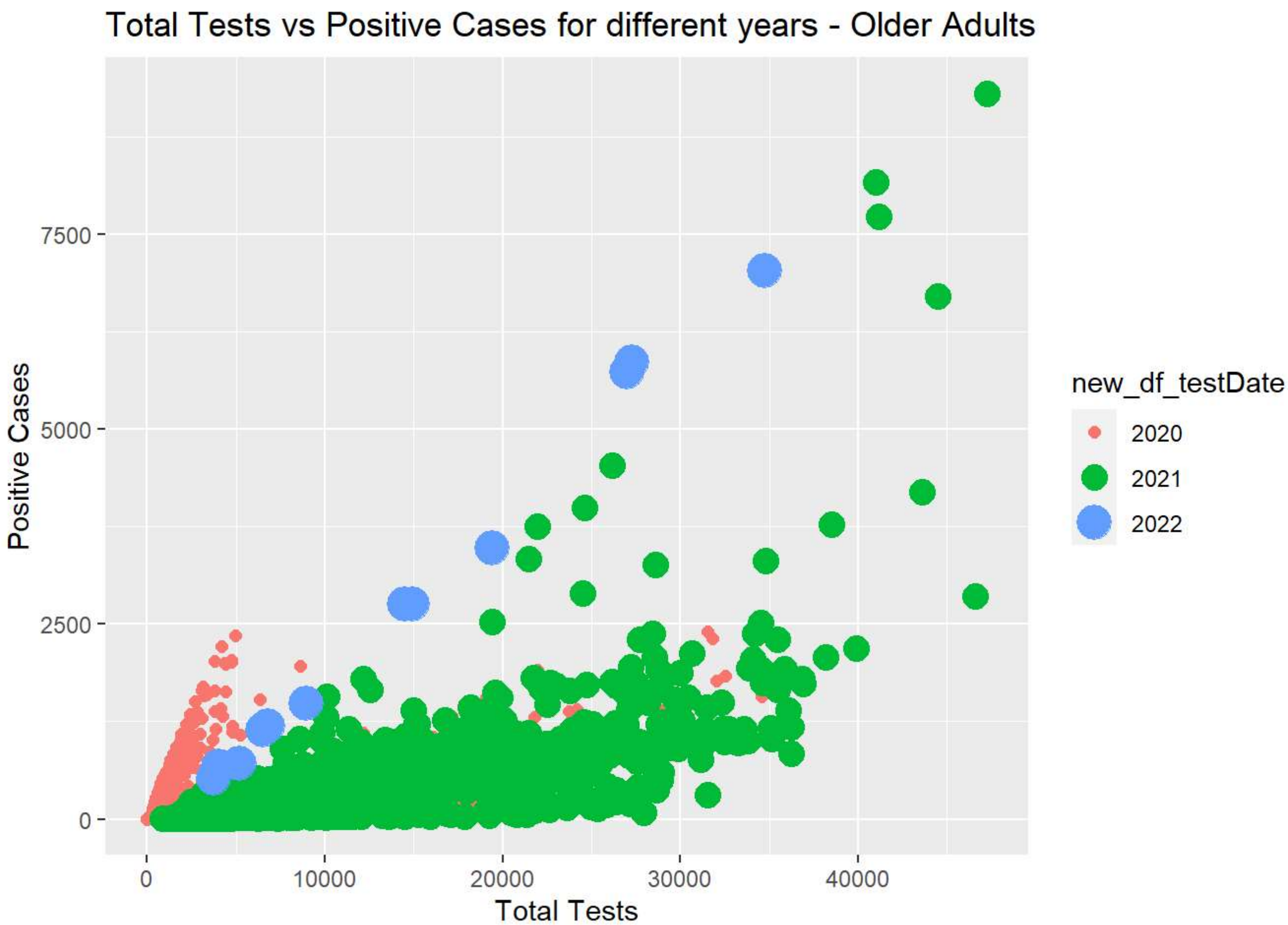

C. In a comment, talk about the insights you got from the line charts you created - can you spot any trends within and between the line charts?

```
#The line chart pattern is similar for all four data frames
#The positivity Rate rises at the start of the year, then gradually declines and slightly rises.
```

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**.

```
new_df_testDate <- format(dfOA$TestDate, format = "%Y")
myplot <- ggplot(dfOA, aes(x=TotalTests, y=PositiveCases)) + geom_point(aes(col=new_df_testDate, size=new_df_testDate))
myplot <- myplot + ggtitle("Total Tests vs Positive Cases for different years - Older Adults") + xlab("Total Tests") + ylab("Positive Cases")
myplot
```

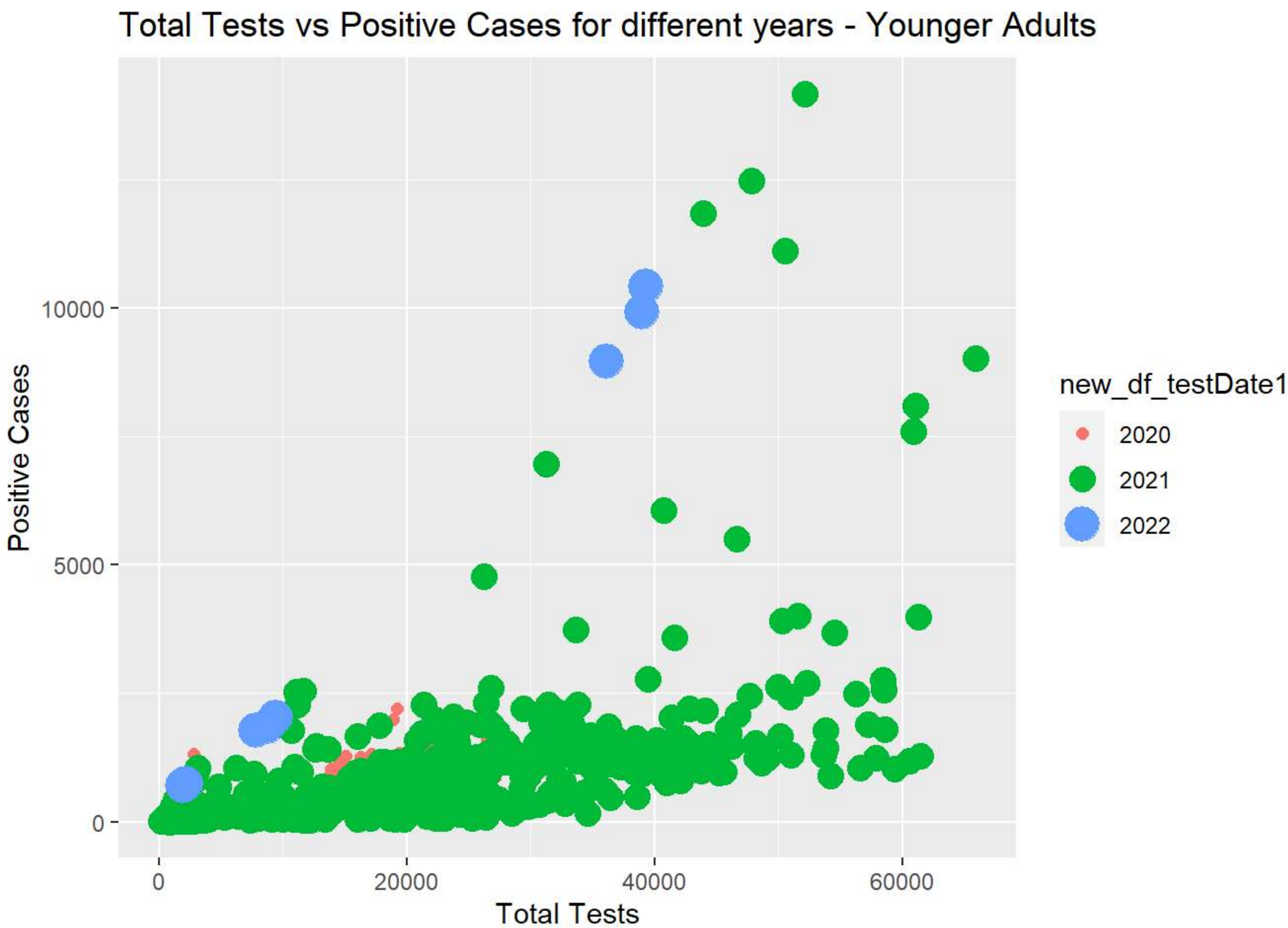
```
## Warning: Using size for a discrete variable is not advised.
```




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

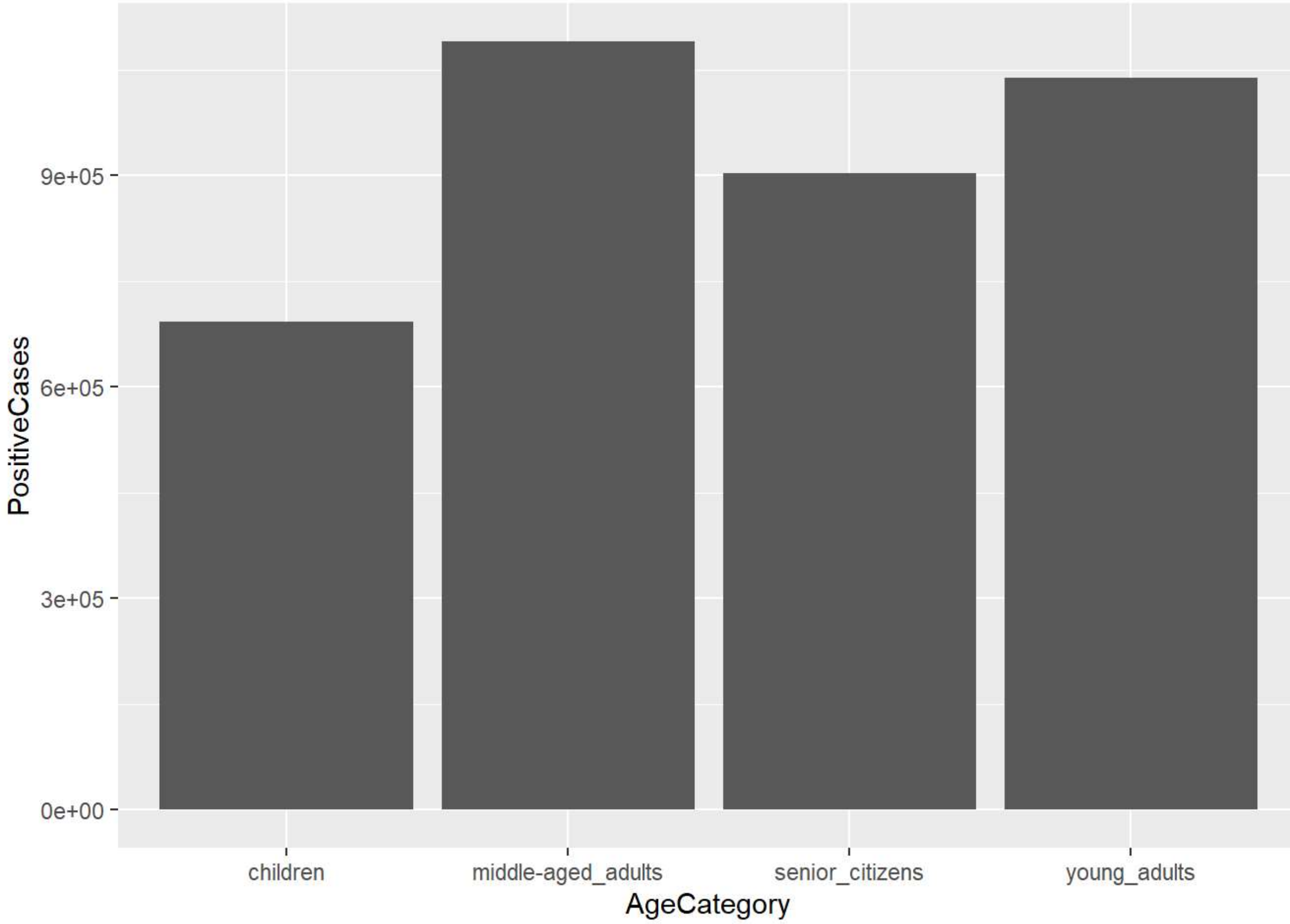
```
new_df_testDate1 <- format(dfYA$TestDate, format = "%Y")
myplot1 <- ggplot(dfYA, aes(x=TotalTests, y=PositiveCases)) + geom_point(aes(col=new_df_testDate
1, size=new_df_testDate1))
myplot1 <- myplot1 + ggtitle("Total Tests vs Positive Cases for different years - Younger Adult
s") + xlab("Total Tests") + ylab("Positive Cases")
myplot1
```

```
## Warning: Using size for a discrete variable is not advised.
```

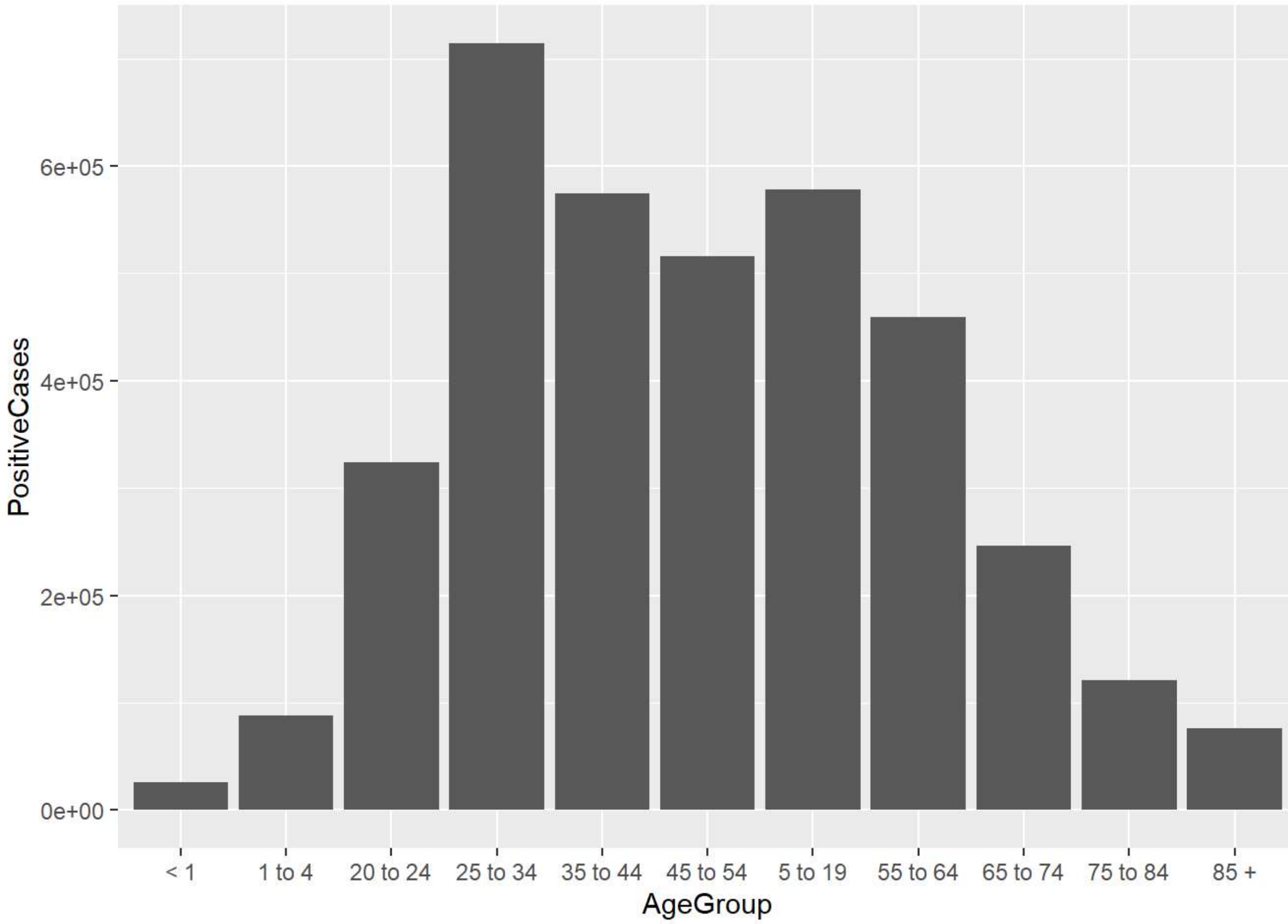



F. Create two barcharts (**using ggplot**) that you think would be interesting by exploring any attribute in two of the dataframes that you have already created via a barchart.

```
#used age group variable in the Old Age adults dataset to plot the bar graph 1
BarChartAC = ggplot(df)
BarChartAC<-BarChartAC +aes(x=AgeCategory, y=PositiveCases)
BarChartAC = BarChartAC + geom_col()
BarChartAC
```




```
#Generated a barchart for df dataframe for AgeGroup #Used aes x-axis as AgeGroup and y-axis as P
ositiveCaes
BarChartAG = ggplot(df)
BarChartAG<-BarChartAG +aes(x=AgeGroup, y=PositiveCases)
BarChartAG = BarChartAG + geom_col()
BarChartAG
```




G.

Interpret these visualizations, what insight do they provide?

```
#In the form of two BarCharts, these visualizations depict the relationship between Positive Cas
es and AgeCategory and Positive Cases and AgeGroup. Children, middle-aged adults, older adults,
and younger adults are all included in the AgeCategory. The age group 25 to 34 has the highest n
umber of positive cases.
```