Practicum Assignment: Data Preparation

Coles Mercier

2023-03-03

Instructions

For this practicum, you will select one of the projects below and 'prepare' the data. Brief Description: You will retrieve, load and clean the data. Using R, you will find and fix errors in the data. You will produce visualizations to aid in your identification of errors. Due: End of week 8

The readxl package is a useful package in R for importing data from Excel files. It provides a simple and efficient way to read data from Excel files, including support for reading data from specific worksheets and specifying column and row ranges.

Read in the Excel file

The code snippet *excel_file <- read_excel("presidential_polls_2020.xlsx", sheet = NULL, col_names = TRUE)* was used to import the Excel file named "presidential_polls_2020.xlsx" into R, with the first row of the data being treated as the column names.

However, it's worth noting that running this code can sometimes result in lengthy "warnings" due to issues such as missing values or formatting inconsistencies in the Excel file. As a result, I choose not to display the output of the code, even though it successfully imported the data into R.

Combine Datafiles

By using the **rbind()** function, I was able to combine two data frames together, since both of them share the same column names and data types. The resulting data frame contains all of the rows from both data frames. The code used to accomplish this is shown below:

```
combined_data <- rbind(my_sheet1, my_sheet2)
```

Clean Data

I used the **glimpse()** function to display a compact summary of the data set. When **glimpse()** is used, it displays a preview of the data, including the number of rows and columns, the data types of each column, and a few sample values from each column. This is useful for getting a quick overview of the data and for checking whether the data is correct and variables are in the expected format.See code used immediately below:

```
glimpse(combined_data)
```

```
## Rows: 615
## Columns: 17
## $ cycle                          <dbl> 2020, 2020, 2020, 2020, 2020, 2020, 2020,…
## $ state                          <chr> "Wyoming", "Wyoming", "Wyoming", "WY", "W…
## $ modeldate                      <dbl> 44138, 44138, 44138, 44138, 44138, 44138,…
## $ candidate_name                 <chr> "Joseph R. Biden Jr.", "Joseph R. Biden J…
## $ startdate                      <chr> "44124", "44122", "44121", "44120", "4411…
## $ enddate                        <dbl> 44136, 44135, 44134, 44133, 44132, 44132,…
## $ pollster                       <chr> "SurveyMonkey", "SurveyMonkey", "Survey M…
## $ samplesize                     <dbl> 330, 361, 378, 1, 614, 739, 713, 654, NA,…
## $ population                     <chr> "lv", "lv", "lv", "LV", "lv", "lv", "lv",…
## $ weight                         <chr> "0.16555700000000001", "2.576410000000000…
## $ influence                      <chr> "0.165557", "0.0256049", "0.0132735", "0.…
## $ pct                            <chr> "33", "34", "33", "33", NA, "31", "32", "…
## $ house_adjusted_pct             <dbl> 31.85685, 32.84583, 31.85685, 31.85685, N…
## $ trend_and_house_adjusted_pct <dbl> 31.85630, 32.83956, 31.85175, 31.85221, N…
## $ tracking                       <chr> "T", "T", "T", "T", NA, "T", "T", "T", NA…
## $ poll_id                        <dbl> 72663, 72525, 72424, 72345, 72334, 72242,…
## $ question_id                    <dbl> 136416, 136147, 135950, 135785, 135714, 1…
```

In order to examine the structure of a data object, I utilized the **str()** function. This function provides a comprehensive summary of the object's structure, including details such as its dimensions (i.e., the number of rows and columns), the number of levels for factors, and the class of each column. By using **str()**, it becomes easier to obtain a rapid overview of the data and verify that the data is accurate and that variables are in the anticipated format. The code used to accomplish this is displayed below:

```
str(combined_data$startdate)
```

```
##  chr [1:615] "44124" "44122" "44121" "44120" "44112" "44105" "44104" ...
```

```
str(combined_data)
```

```
## tibble [615 × 17] (S3: tbl_df/tbl/data.frame)
##  $ cycle                       : num [1:615] 2020 2020 2020 2020 2020 2020 2020 2
##  $ state                       : chr [1:615] "Wyoming" "Wyoming" "Wyoming" "WY" .
##  $ modeldate                   : num [1:615] 44138 44138 44138 44138 44138 ...
##  $ candidate_name              : chr [1:615] "Joseph R. Biden Jr." "Joseph R. Bid
##  $ startdate                   : chr [1:615] "44124" "44122" "44121" "44120" ...
##  $ enddate                     : num [1:615] 44136 44135 44134 44133 44132 ...
##  $ pollster                    : chr [1:615] "SurveyMonkey" "SurveyMonkey" "Surve
##  $ samplesize                  : num [1:615] 330 361 378 1 614 739 713 654 NA NA
##  $ population                  : chr [1:615] "lv" "lv" "lv" "LV" ...
##  $ weight                      : chr [1:615] "0.16555700000000001" "2.57641000000
##  $ influence                   : chr [1:615] "0.165557" "0.0256049" "0.0132735" "
##  $ pct                         : chr [1:615] "33" "34" "33" "33" ...
##  $ house_adjusted_pct          : num [1:615] 31.9 32.8 31.9 31.9 NA ...
##  $ trend_and_house_adjusted_pct: num [1:615] 31.9 32.8 31.9 31.9 NA ...
##  $ tracking                    : chr [1:615] "T" "T" "T" "T" ...
##  $ poll_id                     : num [1:615] 72663 72525 72424 72345 72334 ...
##  $ question_id                 : num [1:615] 136416 136147 135950 135785 135714 .
```

I utilized the **complete.cases()** function to identify and remove missing values (NA) from the data set. This function returns a logical vector that indicates which rows in a data frame are complete, meaning that they contain no missing values. By negating this vector using the ! operator, I can obtain a logical vector that is TRUE for incomplete rows (i.e., those that contain missing values) and FALSE for complete rows. This enabled me to easily filter out the incomplete rows. The code used to accomplish this is displayed below:

```
sum(!complete.cases(combined_data))
```

```
## [1] 438
```

In order to remove rows with missing values, I used the **na.omit()** function. This function removes any rows that contain missing values and returns a new data frame with only the complete cases. By doing so, I was able to eliminate missing data and obtain a more accurate representation of the data set. The code used to accomplish this is displayed below:

```
complete_data <- na.omit(combined_data)
```

I used **complete.cases()** function again to assure there are no more missing values (NA). The code used to accomplish this is displayed below:

```
sum(!complete.cases(complete_data))
```

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

```
str(combined_data$modeldate)
```

```
##  num [1:615] 44138 44138 44138 44138 44138 ...
```

I converted the Excel model date column to R dates by using the **as.Date()** function. This function is designed to convert a string or numeric value into a date object in R. By doing so, it becomes easier to manipulate and analyze the dates, such as sorting them by date or calculating the time difference between dates. This conversion was necessary to ensure that the date data is in the appropriate format for subsequent analyses. The code used to accomplish this is displayed below:

```
complete_data$modeldate <- as.Date(complete_data$modeldate, origin = "1899-12-30")
```

```
str(combined_data$startdate)
```

```
##  chr [1:615] "44124" "44122" "44121" "44120" "44112" "44105" "44104" ...
```

In order to convert the start date column to R dates, I first had to convert the column to integers. This was necessary in order to properly convert the data into date format. By doing so, it becomes easier to manipulate and analyze the dates, such as sorting them by date or calculating the time difference between dates. This conversion was necessary to ensure that the date data is in the appropriate format for subsequent analyses. The code used to accomplish this is displayed below:

```
complete_data$startdate <- as.integer(complete_data$startdate)
```

```
## Warning: NAs introduced by coercion
```

```
complete_data$startdate <- as.Date(complete_data$startdate, origin = "1899-12-30")
```

```
str(combined_data$enddate)
```

```
##  num [1:615] 44136 44135 44134 44133 44132 ...
```

Converted the Excel “enddate” column to R dates. The code used to accomplish this is displayed below:

```
complete_data$enddate <- as.Date(complete_data$enddate, origin = "1899-12-30")
```

In order to remove row 145 from the data set due to the "startdate" column having an NA value in one of the fields, I used the following code snippet: **complete_data <- complete_data[-145,]**. It's important to note that running this code repeatedly would continue to delete row 145 each time, which could lead to unintended consequences. Therefore, once I was satisfied with the final data set, I chose not to run this code any further.

In order to remove row 28 from the data set due to the "startdate" column having an NA value in one of the fields, I used the following code snippet: **complete_data <- complete_data[-28,]**. It's important to note that running this code repeatedly would continue to delete row 145 each time, which could lead to unintended consequences. Therefore, once I was satisfied with the final data set, I chose not to run this code any further.

Replaced all elements in the "cycle" column of the complete_data data frame that are equal to '20' with the value 2020. This step was necessary to standardize the cycle column of the complete_data data frame. By doing this, I ensure that all observations are using the same naming convention, which can avoid issues during analysis or visualization. The code used to accomplish this is displayed below:

```
complete_data$cycle[complete_data$cycle=='20'] <-2020
```

This table displays the contents of the State column prior to cleanup. The code used to accomplish this is displayed below:

```
library(knitr)
state_counts <- as.data.frame(table(combined_data$state))
state_counts <- select(state_counts, State = Var1, Count = Freq)
kable(state_counts, format = "markdown", align = "c", caption = "Initial Name and Co
```

State	Count
National	26
NATl	1
NC	1
North Carolina	118
PA	2
Pennsylvania	125

State	Count
South Carolina	10
Texas	1
Utah	34
Vermont	26
West Virginia	25
WI	1
Wisconsin	66
WY	1
Wyoming	7

Initial Name and Counts of States in the Presidential Polls 2020 Dataset

Changed “WY” to “Wyoming” to standardize the names of states in the state column of the complete_data data frame. By doing this, I ensure that all observations are using the same naming convention, which can avoid issues during analysis or visualization. The code used to accomplish this is displayed below:

```
complete_data$state[complete_data$state=="WY"] <- "Wyoming"
```

Changed “WI” to “Wisconsin” to standardize the names of states in the state column of the complete_data data frame. By doing this, I ensure that all observations are using the same naming convention, which can avoid issues during analysis or visualization. . The code used to accomplish this is displayed below:

```
complete_data$state[complete_data$state=="WI"] <- "Wisconsin"
```

Changed “NATl” to “National” to standardize the names of states in the state column of the complete_data data frame. By doing this, I ensure that all observations are using the same naming convention, which can avoid issues during analysis or visualization. . The code used to accomplish this is displayed below:

```
complete_data$state[complete_data$state=="NATl"] <- "National"
```

I used the **table()** function primarily to display the corrected state names in the state column.

```
table(complete_data$state)
```

```
##
##       National North Carolina   Pennsylvania South Carolina           Utah
##             26              22             24              2             21
##        Vermont  West Virginia      Wisconsin        Wyoming
##             21              20             34              7
```

Changed “Biden” to “Joseph R. Biden Jr” to standardize the names of states in the state column of the complete_data data frame. By doing this, I ensure that all observations are using the same naming convention, which can avoid issues during analysis or visualization. The code used to accomplish this is displayed below:

```
complete_data$candidate_name[complete_data$candidate_name=="Biden"] <- "Joseph R. Bi
```

Changed “Survey Monkey” to “SurveyMonkey” to standardize the names of states in the state column of the complete_data data frame. By doing this, I ensure that all observations are using the same naming convention, which can avoid issues during analysis or visualization. The code used to accomplish this is displayed below:

```
complete_data$pollster[complete_data$pollster=="Survey Monkey"] <- "SurveyMonkey"
```

Changed “Morning Consult” to “MorningConsult” to standardize the names of states in the state column of the complete_data data frame. By doing this, I ensure that all observations are using the same naming convention, which can avoid issues during analysis or visualization. The code used to accomplish this is displayed below:

```
complete_data$pollster[complete_data$pollster=="Morning Consult"] <- "MorningConsult
```

Changed “Morning Con” to “MorningConsult” to standardize the names of states in the state column of the complete_data data frame. By doing this, I ensure that all observations are using the same naming convention, which can avoid issues during analysis or visualization. The code used to accomplish this is displayed below:

```
complete_data$pollster[complete_data$pollster=="Morning Con"] <- "MorningConsult"
```

I filtered the data by removing rows where the value in the sample size column was less than 2. There were 4 rows that had a sample size of only 1, which is considered an outlier and likely a mistake. The code used to accomplish this is displayed below:

```
complete_data <- complete_data[complete_data$samplesize >= 2, ]
```

I used the **summary()** function to generate summary statistics of the data set. These statistics can give me a sense of the distribution of the sample sizes and any potential outliers. The code used to accomplish this is displayed below:

```
summary(complete_data$samplesize)
```

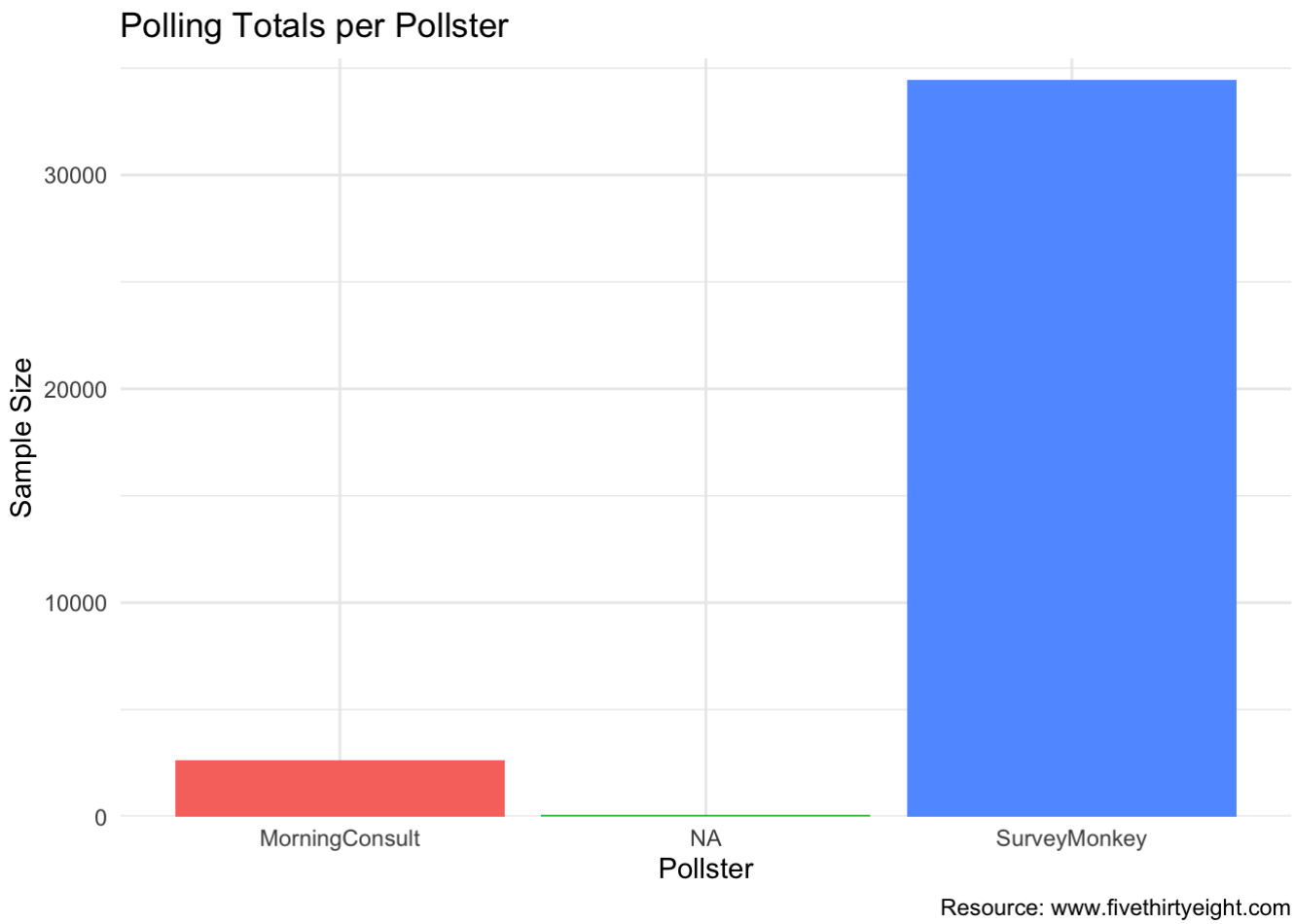
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      99     797    1504    5059    7258   34460
```

Visual-1:

The chart provides a visual representation of the differences in sample sizes across the two pollsters during the 2020 campaign for Joseph R. Biden Jr. It is worth noting that despite efforts to remove it, the chart may still display “NA”, which is not actually present in the data set.

```
ggplot(complete_data, aes(x = pollster, y = samplesize, fill = pollster)) +
  geom_bar(position = "dodge", stat = "identity") +
  scale_y_continuous(limits = c(0, max(complete_data$samplesize) + 1000), expand = c
  xlab("Pollster")+
  ylab("Sample Size") + labs(title = "Polling Totals per Pollster", caption = "Resou
  theme_minimal() +
  guides(fill = FALSE)
```

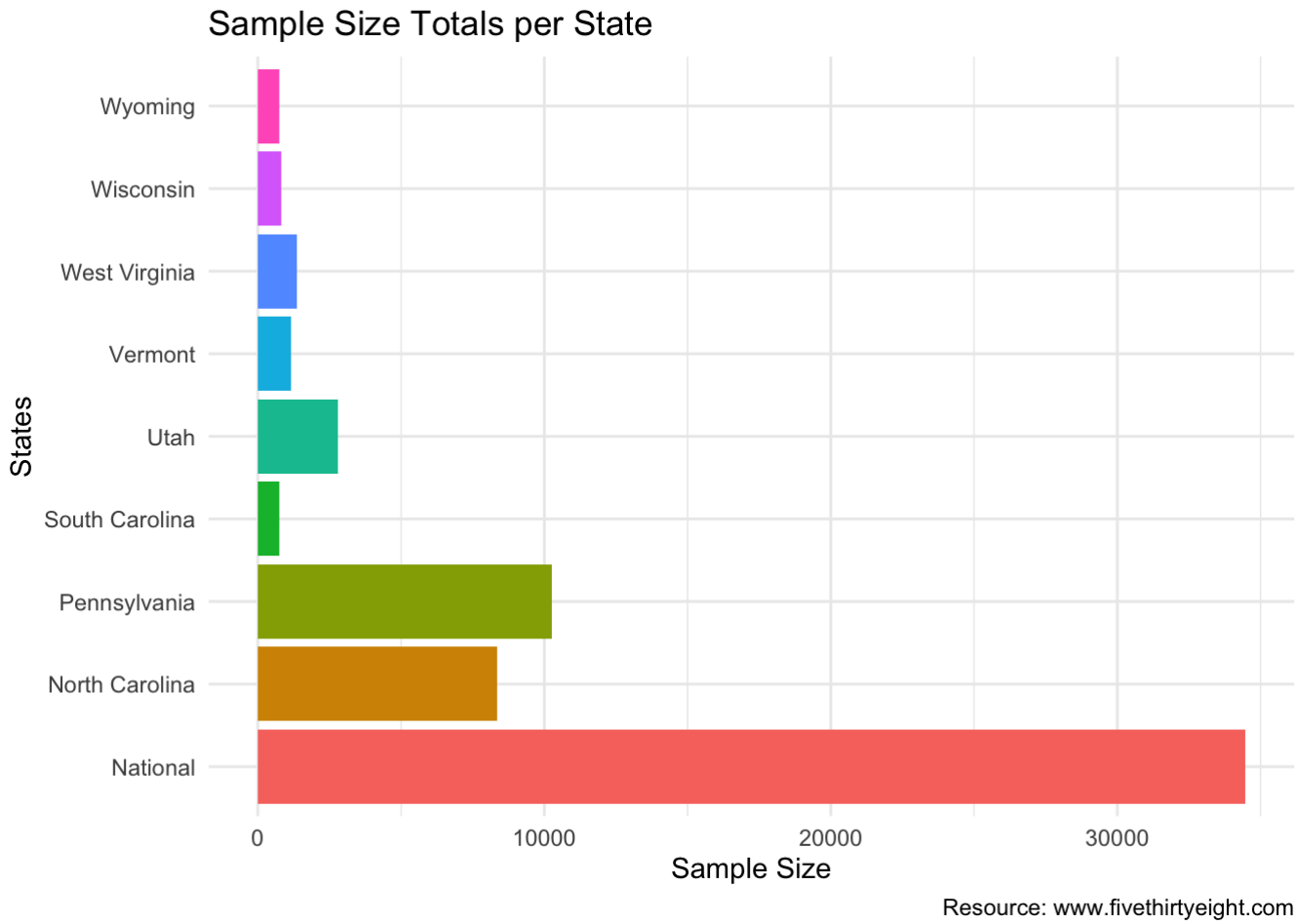
```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" inste
## of ggplot2 3.3.4.
```



Visual-2:

The chart provides a visual representation of the differences in sample sizes across the states during the 2020 campaign for Joseph R. Biden Jr.

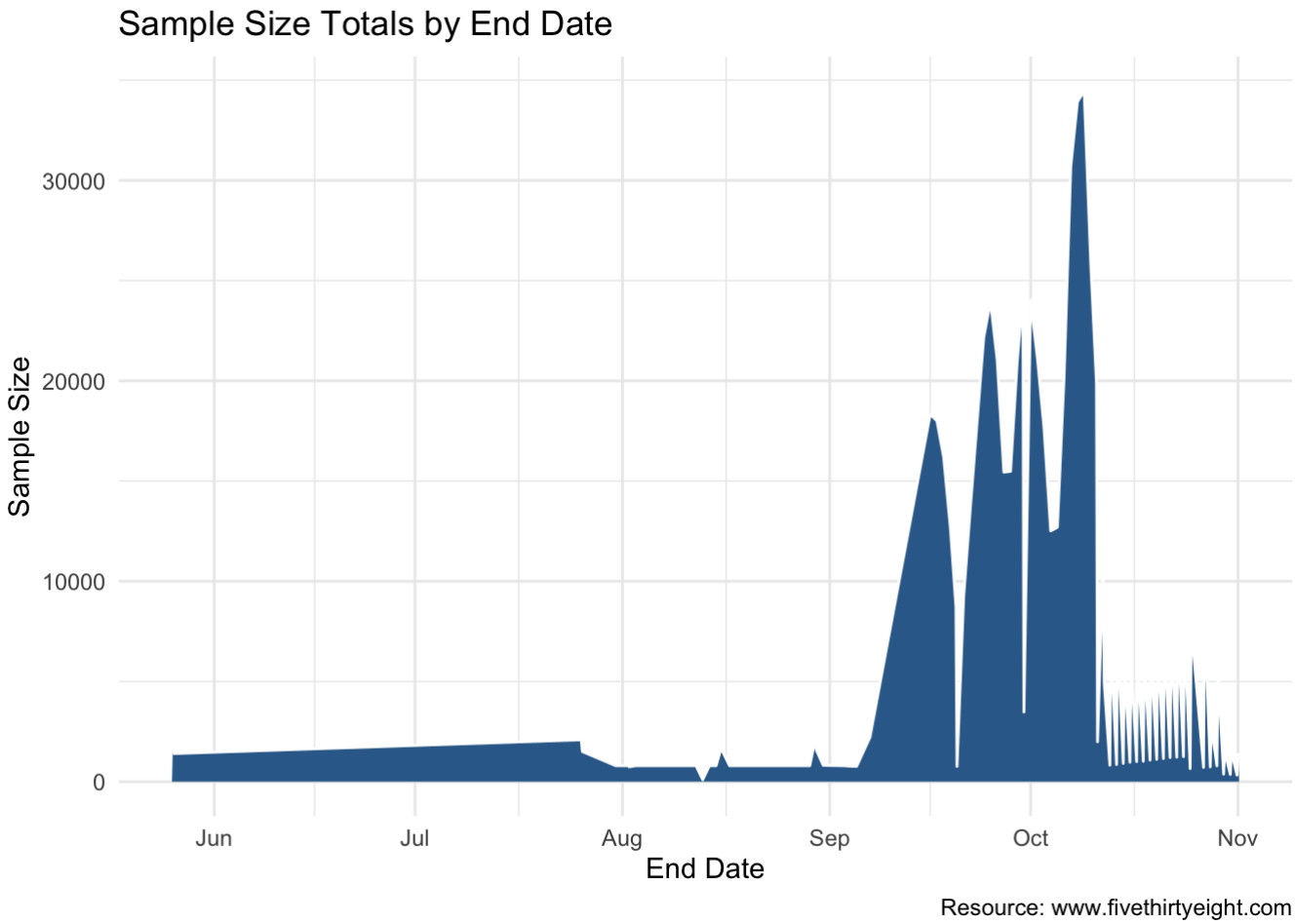
```
ggplot(complete_data, aes(x = samplesize, y = state, fill = state)) +
  geom_bar(position = "dodge", stat = "identity")+
  xlab("Sample Size")+
  ylab("States") + labs(title = "Sample Size Totals per State", caption = "Resource:
  theme_minimal() +
  guides(fill = FALSE)
```



Visual-3:

This chart displays the total sample size for the 2020 Campaign for Joseph R. Biden Jr over time. It is worth noting that the highest sample sizes occurred between approximately mid-September to mid-October.

```
ggplot(complete_data, aes(x = enddate, y = samplesize, fill = enddate)) + geom_area(
  xlab("End Date")+
  ylab("Sample Size") + labs(title = "Sample Size Totals by End Date", caption = "Re
  theme_minimal() +
  guides(fill = FALSE)
```


Sample Size Totals by End Date
30000
20000
10000
0
Sample Size
Jun
Jul
Aug
Sep
Oct
Nov
End Date
Resource: www.fivethirtyeight.com