STA9750 Individidual Project Report

AUTHOR
Astrid Barreras

1 Overarching Question

How do 311 service request volume, response times, and resolution rates between 2019-2024 vary across NYC boroughs, and to what extent can variations be explained by differences in socioeconomic status and funding allocations?

2 Specific Question #4

What effect do SES indicators and budgets have on 311 service requests volume, response times, and resolution rates?

3 Loading Libraries

```
#for data manipulation needed (mutate, filter, summarise, arrange)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
#for tidying up dfs (pivot_wider for SES indicators df)
library(tidyr)

#for working with the large 311 service requests df
library(readr)

#for cleaning column names throughout dfs for easy standardization
library(janitor)
```

```
Warning: package 'janitor' was built under R version 4.4.3


Attaching package: 'janitor'
```

```
The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```r
#for creating an agency map for aligning agencies frrom budget to 311 service requests
library(tibble)

#for converting to POSIXct created date and closed date columns
library(lubridate)
```

```

Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```r
#for fetching ACS5 data
library(tidycensus)
```

```
Warning: package 'tidycensus' was built under R version 4.4.3
```

```r
#for running GAM model
library(mgcv)
```

```
Loading required package: nlme


Attaching package: 'nlme'

The following object is masked from 'package:dplyr':

    collapse

This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```r
#for generating visualizations
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.4.3
```

```r
#for adding interactivity to visualizations
library(plotly)
```

```
Warning: package 'plotly' was built under R version 4.4.3


Attaching package: 'plotly'
```

```
The following object is masked from 'package:ggplot2':

    last_plot

The following object is masked from 'package:stats':

    filter

The following object is masked from 'package:graphics':

    layout
```

```r
#for plotting correlations
library(corrplot)
```

```
Warning: package 'corrplot' was built under R version 4.4.3

corrplot 0.95 loaded
```

4 Data Acquisition & Processing

4.1 NYC 311 Service Requests

4.1.1 Source

This data was extracted from NYC Open Data as provided by 311. The full data available for extraction encompasses 2010-Present (September 21, 2025) across all NYC government agencies (145 agencies). Explore the data dictionary to learn more.

This data was extracted as a CSV file using the "Query data" function, filtered by `Created Date` is between `2019 Jan 01 12:00:00 AM` AND `2024 Dec 31 11:59:59 PM`. Original file size was 10.13GB, with ~18.6M rows and 41 columns. The read in file size using `readr` dropped to 1.9GB with ~18.6M rows and 7 columns, which was much more manageable for memory.

```r
#using read_csv function from readr as it handles reading large datasets efficiently and quickly
ser_reqs <- read_csv("data/raw/service_requests_2019_2024.csv",
                     col_select = c("Unique Key", "Created Date",
                                    "Closed Date", "Agency",
                                    "Agency Name", "Borough",
                                    "Status"))
```

```
Rows: 18650012 Columns: 7
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (6): Created Date, Closed Date, Agency, Agency Name, Borough, Status
dbl (1): Unique Key
```

```
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.
```

4.1.2 Data Cleaning

Checking the structure of the data highlights the need to convert `Created Date` and `Closed Date` from `character` type to `POSIXct` type as this will make it easier to calculate response time as the difference in time between `Created Date` and `Closed Date`. For standardization, the column names ought to be cleaned.

```
#checking structure of data
str(ser_reqs)
```

```
tibble [18,650,012 × 7] (S3: tbl_df/tbl/data.frame)
 $ Unique Key  : num [1:18650012] 41310910 41309122 41317342 41318587 41318948 ...
 $ Created Date: chr [1:18650012] "01/01/2019 12:00:00 AM" "01/01/2019 12:00:00 AM" "01/01/2019
12:00:00 AM" "01/01/2019 12:00:00 AM" ...
 $ Closed Date : chr [1:18650012] "01/01/2019 12:00:00 AM" "01/10/2019 12:00:00 AM" "01/08/2019
12:00:00 AM" "01/01/2019 12:00:00 AM" ...
 $ Agency      : chr [1:18650012] "DOHMH" "DOHMH" "DOHMH" "DOHMH" ...
 $ Agency Name : chr [1:18650012] "Department of Health and Mental Hygiene" "Department of Health
and Mental Hygiene" "Department of Health and Mental Hygiene" "Department of Health and Mental
Hygiene" ...
 $ Borough     : chr [1:18650012] "BRONX" "QUEENS" "BROOKLYN" "BRONX" ...
 $ Status      : chr [1:18650012] "Closed" "Closed" "Closed" "Closed" ...
 - attr(*, "spec")=
  .. cols(
  ..   `Unique Key` = col_double(),
  ..   `Created Date` = col_character(),
  ..   `Closed Date` = col_character(),
  ..   Agency = col_character(),
  ..   `Agency Name` = col_character(),
  ..   Borough = col_character(),
  ..   Status = col_character()
  .. )
```

Checking the missing values highlights the need to remove rows with missing values in columns `Closed Date` and `Borough`.

```
#checking missing values
colSums(is.na(ser_reqs))
```

```
  Unique Key Created Date  Closed Date       Agency  Agency Name      Borough 
           0            0       269692            0            0        39315 
      Status 
           0 
```

Checking the unique values highlights the need to filter for `Status == Closed` to calculate resolution rate.

```r
#checking unique values in status to know what value to filter for to compute resolution rate
unique(ser_reqs$Status)
```

```
[1] "Closed"      "In Progress" "Open"        "Pending"     "Started"
[6] "Email Sent"  "Assigned"    "Unspecified" "Cancel"
```

Checking the missing values highlights the need to filter for `borough %in% c("BRONX", "BROOKLYN", "MANHATTAN", "QUEENS", "STATEN ISLAND")` .

```r
#checking unique values in Borough column
unique(ser_reqs$Borough)
```

```
[1] "BRONX"         "QUEENS"        "BROOKLYN"      "MANHATTAN"
[5] "STATEN ISLAND" "Unspecified"   NA
```

Leveraging these insights, 311 service requests data can be cleaned.

```r
#cleaning 311 service requests
ser_reqs_cl <- ser_reqs %>%

  #standardizing column names
  clean_names() %>%

  #converting created and closed dates to POSIXct
  mutate(created_date = mdy_hms(created_date),
         closed_date = mdy_hms(closed_date),

         #extracing the year from created date
         year = year(created_date)) %>%

  #filtering for non-missing values
  filter(!is.na(closed_date), !is.na(borough),

         #filtering for necessary boroughs
         borough %in% c("BRONX", "BROOKLYN",
                        "MANHATTAN", "QUEENS", "STATEN ISLAND"))
```

Rechecking the structure shows `Created Date` and `Closed Date` were converted from `character` type to `POSIXct` type.

```r
#checking the structure of the data
str(ser_reqs_cl)
```

```
tibble [18,286,780 × 8] (S3: tbl_df/tbl/data.frame)
 $ unique_key  : num [1:18286780] 41310910 41309122 41317342 41318587 41318948 ...
 $ created_date: POSIXct[1:18286780], format: "2019-01-01 00:00:00" "2019-01-01 00:00:00" ...
 $ closed_date : POSIXct[1:18286780], format: "2019-01-01 00:00:00" "2019-01-10 00:00:00" ...
 $ agency      : chr [1:18286780] "DOHMH" "DOHMH" "DOHMH" "DOHMH" ...
```

```
 $ agency_name : chr [1:18286780] "Department of Health and Mental Hygiene" "Department of Health and Mental Hygiene" "Department of Health and Mental Hygiene" "Department of Health and Mental Hygiene" ...
 $ borough     : chr [1:18286780] "BRONX" "QUEENS" "BROOKLYN" "BRONX" ...
 $ status      : chr [1:18286780] "Closed" "Closed" "Closed" "Closed" ...
 $ year        : num [1:18286780] 2019 2019 2019 2019 2019 ...
 - attr(*, "spec")=
  .. cols(
  ..   `Unique Key` = col_double(),
  ..   `Created Date` = col_character(),
  ..   `Closed Date` = col_character(),
  ..   Agency = col_character(),
  ..   `Agency Name` = col_character(),
  ..   Borough = col_character(),
  ..   Status = col_character()
  .. )
```

Rechecking for missing values shows rows with missing values were removed.

```
#checking for missing columns
colSums(is.na(ser_reqs_cl))
```

```
  unique_key created_date  closed_date       agency  agency_name      borough 
           0            0            0            0            0            0 
      status         year 
           0            0 
```

Rechecking unique values of `borough` shows only necessary borough information was retained.

```
#checking unique values in borough column
unique(ser_reqs_cl$borough)
```

```
[1] "BRONX"         "QUEENS"        "BROOKLYN"      "MANHATTAN"    
[5] "STATEN ISLAND"
```

When previously calculating mean response times, extreme values (>365 days) and negative values (< 0 days) was encountered at the borough x agency x year level.

Checking prior to calculating outcomes, the proportion of negative and extreme values against the total volume of service requests.

```
#checking service requests with less than 0 days to close
neg_vals <- ser_reqs_cl %>%


  #calculating days to close
  #converting the difference between closed and created date to numeric
  #using the POSIXct function difftime to handle the calc, unit being days
  mutate(days_to_close = as.numeric(difftime(closed_date,
                                             created_date,
```

```
                                             units="days"))) %>%
  #filtering for less than 0 days to close
  filter(days_to_close < 0)


#calcuulating prop of neg_vals in relation to total volume of reqs
nrow(neg_vals) / nrow(ser_reqs_cl)
```

```
[1] 0.002998122
```

```
#checking service requests with greater than 365 days to close
ext_vals <- ser_reqs_cl %>%

  #calculating days to close
  #converting the difference between closed and created date to numeric
  #using the POSIXct function difftime to handle the calc, unit being days
  mutate(days_to_close = as.numeric(difftime(closed_date,
                                             created_date,
                                             units="days"))) %>%

  #filtering for greater than 365 days to close
  filter(days_to_close > 365)

#calculating prop of ext_vals in relation to total volume of reqs
nrow(ext_vals) / nrow(ser_reqs_cl)
```

```
[1] 0.01664049
```

Given that these extreme and negative days to close are merely a marginal fraction of the overall volume of service requests (~1.86%), they will be removed as they are potentially related to data entry issues.

```
#filtering for non outliers
ser_reqs_ro <- ser_reqs_cl %>%

  #calculating days to close
  mutate(days_to_close = as.numeric(difftime(closed_date,
                                             created_date,
                                             units="days"))) %>%

  #filtering for closure days within 0-365 days
  filter(days_to_close >= 0, days_to_close <= 365)
```

Rechecking outliers to see if removal process was successful.

```
#filtering for days to close greater than 365 days
ser_reqs_ro %>%
  filter(days_to_close > 365)
```

```
# A tibble: 0 × 9
# ℹ 9 variables: unique_key <dbl>, created_date <dttm>, closed_date <dttm>,
#   agency <chr>, agency_name <chr>, borough <chr>, status <chr>, year <dbl>,
#   days_to_close <dbl>
```

```
#filtering for days to close less than 0 days
ser_reqs_ro %>%
  filter(days_to_close < 0 )
```

```
# A tibble: 0 × 9
# ℹ 9 variables: unique_key <dbl>, created_date <dttm>, closed_date <dttm>,
#   agency <chr>, agency_name <chr>, borough <chr>, status <chr>, year <dbl>,
#   days_to_close <dbl>
```

Checking the unique values in agencies in case any changes after removal of outliers.

```
#checking the distinct values of agency names
unique(ser_reqs_ro$agency)
```

```
 [1] "DOHMH" "HPD"   "NYPD"  "TLC"   "DOT"   "DEP"   "DSNY"  "DOB"   "DPR"
[10] "DHS"   "DOF"   "DCWP"  "DFTA"  "EDC"   "DOITT" "DOE"   "OSE"   "OTI"
```

Out of the 18 agencies above, only 15 agencies were identified in the NYC Agency budgets. Three agencies service requests will be removed (EDC, OSE & OTI).

```
ser_reqs_ar <- ser_reqs_ro %>%
  #filter for agencies with budgets
  filter(agency %in% c("DOHMH", "HPD", "NYPD", "TLC", "DOT", "DEP",
                       "DSNY", "DOB", "DPR", "DHS", "DOF", "DCWP",
                       "DFTA", "DOITT", "DOE"))
```

```
ser_reqs_ar %>%
  #check to see if agencies with missing budgets were filtered out
  filter(agency %in% c("OTI", "OSE", "EDC"))
```

```
# A tibble: 0 × 9
# ℹ 9 variables: unique_key <dbl>, created_date <dttm>, closed_date <dttm>,
#   agency <chr>, agency_name <chr>, borough <chr>, status <chr>, year <dbl>,
#   days_to_close <dbl>
```

```
#checking the distinct values of agency names
unique(ser_reqs_ar$agency)
```

```
 [1] "DOHMH" "HPD"   "NYPD"  "TLC"   "DOT"   "DEP"   "DSNY"  "DOB"   "DPR"
[10] "DHS"   "DOF"   "DCWP"  "DFTA"  "DOITT" "DOE"
```

```
#checking the count of agencies left
length(unique(ser_reqs_ar$agency))
```

```
[1] 15
```

Calculation of volume, response times, and resolution rates was performed at the `borough` x `year` level. Volume was calculated by taking the sum of all the service requests. Average response times were calculated using the difference between `closed_date` and `created_date` (previously done above) and then finding the mean (performed below). Resolution rates were calculated using the proportion of closed service requests over volume.

```
ser_reqs_out <- ser_reqs_ar %>%

  #grouping by borough x year for outcome calc
  group_by(borough, year) %>%

  #calculating outcomes
  summarise(

    #summing the volume of service requests
    volume = n(),

    #summing the # of closed reqs
    closed = sum(status == "Closed", na.rm = TRUE),

    #calculating resolution rate based on # of closed reqs & volume
    resolution_rate = closed / volume,

    #calculating mean response time based on days to close
    mean_response = mean(days_to_close, na.rm = TRUE),

    #dropping groups, not necessary, but cleaner for handling later
    .groups = "drop")
```

Checking the first few rows shows that the calculations were performed.

```
#checking first few rows
head(ser_reqs_out)
```

```
# A tibble: 6 × 6
  borough  year volume closed resolution_rate mean_response
  <chr>   <dbl>  <int>  <int>           <dbl>         <dbl>
1 BRONX    2019 439582 439324           0.999          9.04
2 BRONX    2020 597366 597277           1.00           6.79
3 BRONX    2021 650373 650134           1.00           7.02
4 BRONX    2022 723443 723070           0.999          7.25
5 BRONX    2023 618734 618383           0.999          9.30
6 BRONX    2024 731317 730964           1.00           7.98
```

Saving output data to a CSV file.

```
#save to CSV file in project folder path for processed datasets
write.csv(ser_reqs_out, "data/processed/ser_reqs_out.csv")
```

4.2 NYC Expense Budgets

4.2.1 Source

This data was sourced from the NYC Open Data as provided by the Mayor's Office of Management & Budget (OMB). The full data available for extraction encompasses expense agency data by unit of appropriation for the Adopted, Financial Plan and Modified conditions by funding source for agencies across NYC, from FY17 to FY26, last updated July 8, 2025. This data is updated three times per year after publication of the Preliminary, Executive and Adopted Budget, usually in January, April and June respectively.

This data was extracted using the NYC Open Data "Query function". When reading in, columns selected were `Publication Date`, `Fiscal Year`, `Agency Name`, `Total Adopted Budget Amount`, `Unit Appropriation Name`.

Explore the data dictionary to learn more.

```
#reading in the budgets data
budgets <- read_csv("data/raw/nyc_agency_budgets.csv",
                    col_select = c("Publication Date",
                                   "Fiscal Year",
                                   "Agency Name",
                                   "Unit Appropriation Name",
                                   "Total Adopted Budget Amount"))
```

```
Rows: 19094 Columns: 5
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): Agency Name, Unit Appropriation Name
dbl (2): Publication Date, Fiscal Year
num (1): Total Adopted Budget Amount

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.
```

4.2.2 Data Cleaning

Checking the structure highlights the need to standardize column names.

```
#checking structure of data
str(budgets)
```

```
tibble [19,094 × 5] (S3: tbl_df/tbl/data.frame)
 $ Publication Date           : num [1:19094] 20250501 20250501 20250501 20250501 20250501 ...
 $ Fiscal Year                : num [1:19094] 2026 2026 2026 2026 2026 ...
 $ Agency Name                : chr [1:19094] "DEPARTMENT OF FINANCE" "DEPARTMENT OF FINANCE"
"DEPARTMENT OF FINANCE" "DEPARTMENT OF FINANCE" ...
 $ Unit Appropriation Name    : chr [1:19094] "PARKING VIOLATIONS BUREAU OTPS" "LEGAL-OTPS"
"AUDIT-OTPS" "PROPERTY-OTPS" ...
 $ Total Adopted Budget Amount: num [1:19094] 794475 234731 345711 4553322 40849302 ...
 - attr(*, "spec")=
  .. cols(
  ..   `Publication Date` = col_double(),
  ..   `Fiscal Year` = col_double(),
  ..   `Agency Name` = col_character(),
  ..   `Unit Appropriation Name` = col_character(),
  ..   `Total Adopted Budget Amount` = col_number()
  .. )
```

Checking missing values shows no missing values are present in this data.

```
#checking for missing values
colSums(is.na(budgets))
```

```
           Publication Date                 Fiscal Year
                          0                           0
                Agency Name     Unit Appropriation Name
                          0                           0
Total Adopted Budget Amount
                          0
```

Given that these budgets are published in three cycles (preliminary, executive, adopted), the latest `Publication Date` per `Fiscal Year` per `Agency Name` ought to be extracted.

A peek into the `Publication Date` shows there is no standard `Publication Date` for the adopted budget for the June cycle, which is the last `Publication Date` per `Fiscal Year` per `Agency Name.`

```
#filtering for a subset of data to peek
budgets %>%

  #filtering for one year
  filter(`Fiscal Year` == 2024) %>%

  #arranging by publication date
  arrange(desc(`Publication Date`))
```

```
# A tibble: 2,046 × 5
   `Publication Date` `Fiscal Year` `Agency Name`          Unit Appropriation N…¹
                <dbl>         <dbl> <chr>                  <chr>
 1           20230630          2024 DEPARTMENT OF EDUCAT… SE INSTR & SCH LEADER…
 2           20230630          2024 DEPARTMENT OF EDUCAT… CHARTER SCHOOLS
 3           20230630          2024 DEPARTMENT OF EDUCAT… UNIVERSAL PRE-K - PS
```

```
 4           20230630          2024 DEPARTMENT OF EDUCAT… UNIVERSAL PRE-K - OTPS
 5           20230630          2024 DEPARTMENT OF EDUCAT… EARLY CHILDHOOD PROGR…
 6           20230630          2024 DEPARTMENT OF EDUCAT… EARLY CHILDHOOD PROGR…
 7           20230630          2024 DEPARTMENT OF EDUCAT… SCHOOL SUPPORT ORGANI…
 8           20230630          2024 DEPARTMENT OF EDUCAT… SCHOOL SUPPORT ORGANI…
 9           20230630          2024 DEPARTMENT OF EDUCAT… CW SE INSTR & SCHL LE…
10           20230630          2024 DEPARTMENT OF EDUCAT… CW SE INSTR & SCHL LE…
# ℹ 2,036 more rows
# ℹ abbreviated name: ¹`Unit Appropriation Name`
# ℹ 1 more variable: `Total Adopted Budget Amount` <dbl>
```

```
#filtering for a subset of data to peek
budgets %>%

  #filtering for one year
  filter(`Fiscal Year` == 2023) %>%

  #arranging by publication date
  arrange(desc(`Publication Date`))
```

```
# A tibble: 1,982 × 5
   `Publication Date` `Fiscal Year` `Agency Name`         Unit Appropriation N…¹
                <dbl>         <dbl> <chr>                 <chr>
 1           20220614          2023 DEPARTMENT OF EDUCAT… NPS & FIT PMTS - OTPS
 2           20220614          2023 DEPARTMENT OF EDUCAT… CATEGORICAL PROGRAMS …
 3           20220614          2023 DEPARTMENT OF EDUCAT… CATEGORICAL PROGRAMS …
 4           20220614          2023 CITY UNIVERSITY OF N… COMMUNITY COLLEGE-OTPS
 5           20220614          2023 CITY UNIVERSITY OF N… COMMUNITY COLLEGE PS
 6           20220614          2023 CITY UNIVERSITY OF N… HUNTER SCHOOLS-OTPS
 7           20220614          2023 CITY UNIVERSITY OF N… HUNTER SCHOOLS-PS
 8           20220614          2023 CITY UNIVERSITY OF N… SENIOR COLLEGE OTPS
 9           20220614          2023 DEPARTMENT OF INVEST… PERSONAL SERVICES
10           20220614          2023 DEPARTMENT OF INVEST… OTHER THAN PERSONAL S…
# ℹ 1,972 more rows
# ℹ abbreviated name: ¹`Unit Appropriation Name`
# ℹ 1 more variable: `Total Adopted Budget Amount` <dbl>
```

This means that the best way to select is to filter for max `Publication Date` per `Fiscal Year` per `Agency Name`.

```
#filtering for a subset of data to peek
budgets %>%

  #grouping by fiscal year x agency name
  group_by(`Fiscal Year`, `Agency Name`) %>%

  #filter for max publication dates and one year to peek
  filter(`Publication Date` == max(`Publication Date`, na.rm = TRUE,
                                   `Fiscal Year`= 2024))
```

```
# A tibble: 6,607 × 5
# Groups:   Fiscal Year, Agency Name [1,406]
  `Publication Date` `Fiscal Year` `Agency Name`         Unit Appropriation N…¹
               <dbl>         <dbl> <chr>                 <chr>
 1          20250116          2026 LEASE ADJUSTMENT      OTHER THAN PERSONAL S…
 2          20250116          2026 ENERGY ADJUSTMENT     OTHER THAN PERSONAL S…
 3          20250630          2026 PUBLIC ADMINISTRATOR… OTHER THAN PERSONAL S…
 4          20250630          2026 PUBLIC ADMINISTRATOR… PERSONAL SERVICES
 5          20250630          2026 PUBLIC ADMINISTRATOR… OTHER THAN PERSONAL S…
 6          20250630          2026 PUBLIC ADMINISTRATOR… PERSONAL SERVICES
 7          20250630          2026 PUBLIC ADMINISTRATOR… OTHER THAN PERSONAL S…
 8          20250630          2026 PUBLIC ADMINISTRATOR… PERSONAL SERVICES
 9          20250630          2026 PUBLIC ADMINISTRATOR… OTHER THAN PERSONAL S…
10          20250630          2026 PUBLIC ADMINISTRATOR… PERSONAL SERVICES
# ℹ 6,597 more rows
# ℹ abbreviated name: ¹`Unit Appropriation Name`
# ℹ 1 more variable: `Total Adopted Budget Amount` <dbl>
```

Leveraging the insights from about the NYC Agency budget can be cleaned.

```
#cleaning budgets and summarizing total adopted budgets agency x year
budgets_sum <- budgets %>%

  #standardizing column names
  clean_names() %>%

  #grouping
  group_by(agency_name, fiscal_year) %>%

  #filtering for the max pub date per fiscal year x agency
  filter(publication_date == max(publication_date, na.rm = TRUE)) %>%

  #summing budget lines per agency and fiscal year
  summarise(budget = sum(total_adopted_budget_amount,
                         na.rm = TRUE), .groups = "drop")
```

Checking the first few rows shows calculations have been performed correctly.

```
#checking the first few rows
head(budgets_sum)
```

```
# A tibble: 6 × 3
  agency_name                       fiscal_year     budget
  <chr>                                   <dbl>      <dbl>
1 ADMIN FOR CHILDREN'S SERVICES            2017 2948922092
2 ADMIN FOR CHILDREN'S SERVICES            2018 2977931705
3 ADMIN FOR CHILDREN'S SERVICES            2019 3129344639
4 ADMIN FOR CHILDREN'S SERVICES            2020 2971704535
5 ADMIN FOR CHILDREN'S SERVICES            2021 2690417661
6 ADMIN FOR CHILDREN'S SERVICES            2022 2658462183
```

Cross checking with City Council Budget Reports to ensure the proper budgets were pulled for the fiscal year. This confirmed that it was successful.

```
budgets_sum %>%

  #filtering for one agency to test
  filter(agency_name == "HOUSING PRESERVATION AND DEVELOPMENT")
```

```
# A tibble: 10 × 3
   agency_name                          fiscal_year     budget
   <chr>                                      <dbl>      <dbl>
 1 HOUSING PRESERVATION AND DEVELOPMENT        2017  752992761
 2 HOUSING PRESERVATION AND DEVELOPMENT        2018 1271948186
 3 HOUSING PRESERVATION AND DEVELOPMENT        2019 1145089005
 4 HOUSING PRESERVATION AND DEVELOPMENT        2020 1142480319
 5 HOUSING PRESERVATION AND DEVELOPMENT        2021 1021051162
 6 HOUSING PRESERVATION AND DEVELOPMENT        2022 1055474033
 7 HOUSING PRESERVATION AND DEVELOPMENT        2023 1167710181
 8 HOUSING PRESERVATION AND DEVELOPMENT        2024 1256806980
 9 HOUSING PRESERVATION AND DEVELOPMENT        2025 1413169333
10 HOUSING PRESERVATION AND DEVELOPMENT        2026 1993121341
```

Saving to CSV file.

```
#saving to csv
write.csv(budgets_sum ,"data/processed/budgets_sum.csv")
```

To align the budget data to the 311 and ACS5 data, the budgets were converted from fiscal year to calendar year.

```
budgets_conv <- budgets_sum %>%

#grouping by agency
  group_by(agency_name) %>%

  #arranging is necessary for lag (in what order to compare fiscal years)
  arrange(fiscal_year, .by_group = TRUE) %>%

  #fiscal year to cal year conversion
  #used lag from dplyr for easier calcution
  mutate(

    #takes half of the current years budget
    half_current = budget / 2,

    #takes half of the previous year budget
    half_prev    = lag(budget, default = 0) / 2,

    #adding half the current year and half the prev year budget
```

```
    budget = half_current + half_prev,

    #renaming fiscal year as year
    year = fiscal_year) %>%

  #ungrouping for better handling
  ungroup() %>%

  #selecting only the necessary columns
  select(agency_name, year, budget) %>%

  #filtering only for necessary years
  filter(year %in% c(2019:2024))
```

Check final unique values of `agency_names` for mapping with 311 service requests.

```
#checking the distinct values of agency names
unique(budgets_conv$agency_name)
```

```
  [1] "ADMIN FOR CHILDREN'S SERVICES"
  [2] "BOARD OF CORRECTION"
  [3] "BOARD OF ELECTIONS"
  [4] "BOROUGH PRESIDENT - BROOKLYN"
  [5] "BOROUGH PRESIDENT - MANHATTAN"
  [6] "BOROUGH PRESIDENT - QUEENS"
  [7] "BOROUGH PRESIDENT BRONX"
  [8] "BOROUGH PRESIDENT STATEN ISLAND"
  [9] "BRONX COMMUNITY BOARD #1"
 [10] "BRONX COMMUNITY BOARD #10"
 [11] "BRONX COMMUNITY BOARD #11"
 [12] "BRONX COMMUNITY BOARD #12"
 [13] "BRONX COMMUNITY BOARD #2"
 [14] "BRONX COMMUNITY BOARD #3"
 [15] "BRONX COMMUNITY BOARD #4"
 [16] "BRONX COMMUNITY BOARD #5"
 [17] "BRONX COMMUNITY BOARD #6"
 [18] "BRONX COMMUNITY BOARD #7"
 [19] "BRONX COMMUNITY BOARD #8"
 [20] "BRONX COMMUNITY BOARD #9"
 [21] "BROOKLYN COMMUNITY BOARD #1"
 [22] "BROOKLYN COMMUNITY BOARD #10"
 [23] "BROOKLYN COMMUNITY BOARD #11"
 [24] "BROOKLYN COMMUNITY BOARD #12"
 [25] "BROOKLYN COMMUNITY BOARD #13"
 [26] "BROOKLYN COMMUNITY BOARD #14"
 [27] "BROOKLYN COMMUNITY BOARD #15"
 [28] "BROOKLYN COMMUNITY BOARD #16"
 [29] "BROOKLYN COMMUNITY BOARD #17"
 [30] "BROOKLYN COMMUNITY BOARD #18"
 [31] "BROOKLYN COMMUNITY BOARD #2"
```

[32] "BROOKLYN COMMUNITY BOARD #3"
[33] "BROOKLYN COMMUNITY BOARD #4"
[34] "BROOKLYN COMMUNITY BOARD #5"
[35] "BROOKLYN COMMUNITY BOARD #6"
[36] "BROOKLYN COMMUNITY BOARD #7"
[37] "BROOKLYN COMMUNITY BOARD #8"
[38] "BROOKLYN COMMUNITY BOARD #9"
[39] "BROOKLYN PUBLIC LIBRARY"
[40] "BUSINESS INTEGRITY COMMISSION"
[41] "CAMPAIGN FINANCE BOARD"
[42] "CITY CLERK"
[43] "CITY COUNCIL"
[44] "CITY UNIVERSITY OF NEW YORK"
[45] "CITYWIDE SAVINGS INITIATIVES"
[46] "CIVIL SERVICE COMMISSION"
[47] "CIVILIAN COMPLAINT REVIEW BOARD"
[48] "COMMISSION ON HUMAN RIGHTS"
[49] "COMMISSION ON RACIAL EQUITY"
[50] "CONFLICTS OF INTEREST BOARD"
[51] "CRIMINAL JUSTICE COORDINATOR"
[52] "DEBT SERVICE"
[53] "DEPARTMENT FOR THE AGING"
[54] "DEPARTMENT OF BUILDINGS"
[55] "DEPARTMENT OF CITY PLANNING"
[56] "DEPARTMENT OF CITYWIDE ADMIN SERVICE"
[57] "DEPARTMENT OF CONSUMER AFFAIRS"
[58] "DEPARTMENT OF CORRECTION"
[59] "DEPARTMENT OF CULTURAL AFFAIRS"
[60] "DEPARTMENT OF DESIGN & CONSTRUCTION"
[61] "DEPARTMENT OF EDUCATION"
[62] "DEPARTMENT OF EMERGENCY MANAGEMENT"
[63] "DEPARTMENT OF ENVIRONMENTAL PROTECT."
[64] "DEPARTMENT OF FINANCE"
[65] "DEPARTMENT OF HEALTH AND MENTAL HYGIENE"
[66] "DEPARTMENT OF HOMELESS SERVICES"
[67] "DEPARTMENT OF INFO TECH & TELECOMM"
[68] "DEPARTMENT OF INVESTIGATION"
[69] "DEPARTMENT OF PARKS AND RECREATION"
[70] "DEPARTMENT OF PROBATION"
[71] "DEPARTMENT OF RECORDS & INFORMATION SVS"
[72] "DEPARTMENT OF SANITATION"
[73] "DEPARTMENT OF SMALL BUSINESS SERVICES"
[74] "DEPARTMENT OF SOCIAL SERVICES"
[75] "DEPARTMENT OF TRANSPORTATION"
[76] "DEPARTMENT OF VETERANS' SERVICES"
[77] "DEPARTMENT OF YOUTH & COMMUNITY DEV"
[78] "DEPT OF CONSUMER & WORKER PROTECTION"
[79] "DISTRICT ATTORNEY BRONX COUNTY"
[80] "DISTRICT ATTORNEY KINGS COUNTY"
[81] "DISTRICT ATTORNEY NEW YORK COUNTY"
[82] "DISTRICT ATTORNEY QUEENS COUNTY"

```
 [83] "DISTRICT ATTORNEY RICHMOND COUNTY"
 [84] "DISTRICTING COMMISSION"
 [85] "ENERGY ADJUSTMENT"
 [86] "EQUAL EMPLOYMENT PRACTICES COMMISSION"
 [87] "FINANCIAL INFORMATION SERVICE AGENCY"
 [88] "FIRE DEPARTMENT"
 [89] "HEALTH AND HOSPITALS CORP"
 [90] "HOUSING PRESERVATION AND DEVELOPMENT"
 [91] "INDEPENDENT BUDGET OFFICE"
 [92] "LANDMARKS PRESERVATION COMM."
 [93] "LAW DEPARTMENT"
 [94] "LEASE ADJUSTMENT"
 [95] "MANHATTAN COMMUNITY BOARD #1"
 [96] "MANHATTAN COMMUNITY BOARD #10"
 [97] "MANHATTAN COMMUNITY BOARD #11"
 [98] "MANHATTAN COMMUNITY BOARD #12"
 [99] "MANHATTAN COMMUNITY BOARD #2"
[100] "MANHATTAN COMMUNITY BOARD #3"
[101] "MANHATTAN COMMUNITY BOARD #4"
[102] "MANHATTAN COMMUNITY BOARD #5"
[103] "MANHATTAN COMMUNITY BOARD #6"
[104] "MANHATTAN COMMUNITY BOARD #7"
[105] "MANHATTAN COMMUNITY BOARD #8"
[106] "MANHATTAN COMMUNITY BOARD #9"
[107] "MAYORALTY"
[108] "MISCELLANEOUS"
[109] "NEW YORK PUBLIC LIBRARY"
[110] "NEW YORK RESEARCH LIBRARIES"
[111] "NYC TAXI AND LIMOUSINE COMM"
[112] "OFFICE OF ADMIN TRIALS & HEARINGS"
[113] "OFFICE OF ADMINISTRATIVE TAX APPEALS"
[114] "OFFICE OF COLLECTIVE BARGAINING"
[115] "OFFICE OF PAYROLL ADMINISTRATION"
[116] "OFFICE OF PROSECUTION SPEC NARCO"
[117] "OFFICE OF RACIAL EQUITY"
[118] "OFFICE OF THE ACTUARY"
[119] "OFFICE OF THE COMPTROLLER"
[120] "PENSION CONTRIBUTIONS"
[121] "POLICE DEPARTMENT"
[122] "PUBLIC ADMINISTRATOR- QUEENS COUNTY"
[123] "PUBLIC ADMINISTRATOR-BRONX COUNTY"
[124] "PUBLIC ADMINISTRATOR-KINGS COUNTY"
[125] "PUBLIC ADMINISTRATOR-NEW YORK COUNTY"
[126] "PUBLIC ADMINISTRATOR-RICHMOND COUNTY"
[127] "PUBLIC ADVOCATE"
[128] "QUEENS BOROUGH PUBLIC LIBRARY"
[129] "QUEENS COMMUNITY BOARD #1"
[130] "QUEENS COMMUNITY BOARD #10"
[131] "QUEENS COMMUNITY BOARD #11"
[132] "QUEENS COMMUNITY BOARD #12"
[133] "QUEENS COMMUNITY BOARD #13"
```

```
[134] "QUEENS COMMUNITY BOARD #14"
[135] "QUEENS COMMUNITY BOARD #2"
[136] "QUEENS COMMUNITY BOARD #3"
[137] "QUEENS COMMUNITY BOARD #4"
[138] "QUEENS COMMUNITY BOARD #5"
[139] "QUEENS COMMUNITY BOARD #6"
[140] "QUEENS COMMUNITY BOARD #7"
[141] "QUEENS COMMUNITY BOARD #8"
[142] "QUEENS COMMUNITY BOARD #9"
[143] "STATEN ISLAND COMMUNITY BOARD #1"
[144] "STATEN ISLAND COMMUNITY BOARD #2"
[145] "STATEN ISLAND COMMUNITY BOARD #3"
```

To support with selecting only budgets that are aligned with agencies that are in the 311 data, an agency map was created.

```
#agency map to help with joining service requests with budgets
agency_map <- tribble(
  ~agency_abbr, ~agency_name,
  "DOHMH", "DEPARTMENT OF HEALTH AND MENTAL HYGIENE",
  "HPD",   "HOUSING PRESERVATION AND DEVELOPMENT",
  "NYPD",  "POLICE DEPARTMENT",
  "TLC",   "NYC TAXI AND LIMOUSINE COMM",
  "DOT",   "DEPARTMENT OF TRANSPORTATION",
  "DEP",   "DEPARTMENT OF ENVIRONMENTAL PROTECT.",
  "DSNY",  "DEPARTMENT OF SANITATION",
  "DOB",   "DEPARTMENT OF BUILDINGS",
  "DPR",   "DEPARTMENT OF PARKS AND RECREATION",
  "DHS",   "DEPARTMENT OF HOMELESS SERVICES",
  "DOF",   "DEPARTMENT OF FINANCE",
  "DCWP",  "DEPT OF CONSUMER & WORKER PROTECTION",
  "DFTA",  "DEPARTMENT FOR THE AGING",
  "DOITT", "DEPARTMENT OF INFO TECH & TELECOMM",
  "DOE",   "DEPARTMENT OF EDUCATION")
```

Testing to see if the filtering with an agency map works. It works as 15 agencies are retained, coinciding with the number of agencies in the 311 service requests summary.

```
budgets_filt <- budgets_conv %>%

  #filtering for budget agencies in the agency map
  filter(agency_name %in% agency_map$agency_name)

unique(budgets_filt$agency_name)
```

```
[1] "DEPARTMENT FOR THE AGING"
[2] "DEPARTMENT OF BUILDINGS"
[3] "DEPARTMENT OF EDUCATION"
[4] "DEPARTMENT OF ENVIRONMENTAL PROTECT."
[5] "DEPARTMENT OF FINANCE"
```

```
 [6] "DEPARTMENT OF HEALTH AND MENTAL HYGIENE"
 [7] "DEPARTMENT OF HOMELESS SERVICES"
 [8] "DEPARTMENT OF INFO TECH & TELECOMM"
 [9] "DEPARTMENT OF PARKS AND RECREATION"
[10] "DEPARTMENT OF SANITATION"
[11] "DEPARTMENT OF TRANSPORTATION"
[12] "DEPT OF CONSUMER & WORKER PROTECTION"
[13] "HOUSING PRESERVATION AND DEVELOPMENT"
[14] "NYC TAXI AND LIMOUSINE COMM"
[15] "POLICE DEPARTMENT"
```

```
length(unique(budgets_filt$agency_name))
```

```
[1] 15
```

Final summarizing of budget x year can be accomplished by filtering for the agency names in the agency map.

```
budgets_agg <- budgets_conv %>%

  #filtering for budget agencies in the agency map
  filter(agency_name %in% agency_map$agency_name) %>%

  #grouping by year
  group_by(year) %>%

  #summing budgets
  summarise(budget = sum(budget),
            .groups = "drop")
```

Checking final structure.

```
#check structure
str(budgets_agg)
```

```
tibble [6 × 2] (S3: tbl_df/tbl/data.frame)
 $ year  : num [1:6] 2019 2020 2021 2022 2023 ...
 $ budget: num [1:6] 3.94e+10 4.14e+10 4.33e+10 4.42e+10 4.70e+10 ...
```

```
#see first few rows
head(budgets_agg)
```

```
# A tibble: 6 × 2
   year      budget
  <dbl>       <dbl>
1  2019 39448458298.
2  2020 41399018536
3  2021 43309639214
```

```
4   2022 44190185299
5   2023 46982820611
6   2024 49976149672
```

Saving to CSV file for reference.

```
#save to CSV file in project folder path for processed datasets
write.csv(budgets_agg, "data/processed/budgets_agg.csv")
```

4.3 ACS Socioeconomic Indicators

4.3.1 Source

This dataset was sourced from the R package `tidycensus` which wraps around the Census API. Specifically, data obtained from the American Community Survey (5-year 2023 release) for NYC boroughs (Bronx, Queens, Manhattan, Brooklyn, and Staten Island).

To explore the variables available per ACS survey type, the `load_variables` function can be leveraged.

```
#viewing the specific variables in AC5
view(load_variables(2023, "acs5", cache = TRUE))
```

SES Indicator	ACS5 Code
Total Pop	B01003_001
Median Income	B19013_001
Poverty	B17001_002
Unemployment	B23025_005

```
#setting variables

#defining the geography to be counties for borough selection
geography <- "county"

#setting the state as the NYS code
state <- 36

#selecting the necessary counties
county <- c("Bronx", "Kings", "Richmond", "Queens", "New York")


#defining the SES indicators as noted above
variables <- c(
  total_pop      = "B01003_001",
  median_income  = "B19013_001",
```

```
  poverty        = "B17010_002",
  unemployment   = "B23025_005")
```

Fetching primary ACS data (ACS5) which will be used for LM and GAM (smoothing) models.

```
#getting AC5 data while passing the predefined args above
acs5 <- get_acs(
  geography = geography,
  variables = variables,
  state = state,
  county = county,
  year = 2023,
  survey = "acs5",
  geometry = FALSE)
```

```
Getting data from the 2019-2023 5-year ACS
```

4.3.2 Data Cleaning

Checking structure. Columns are not standardized and clean. Borough names include county suffix and state name which will need to be cleaned to match the 311 service requests data.

```
#checking structure
str(acs5)
```

```
tibble [20 × 5] (S3: tbl_df/tbl/data.frame)
 $ GEOID   : chr [1:20] "36005" "36005" "36005" "36005" ...
 $ NAME    : chr [1:20] "Bronx County, New York" "Bronx County, New York" "Bronx County, New
York" "Bronx County, New York" ...
 $ variable: chr [1:20] "total_pop" "poverty" "median_income" "unemployment" ...
 $ estimate: num [1:20] 1419250 73028 49036 71936 2646306 ...
 $ moe     : num [1:20] NA 2745 872 2730 NA ...
```

Checking the first few rows. The format of the dataset is not appropriate for merging with 311 service requests and NYC Agency budgets. This will need to be pivoted wide for proper merging later.

```
#checking first few rows
head(acs5)
```

```
# A tibble: 6 × 5
  GEOID NAME                   variable      estimate   moe
  <chr> <chr>                  <chr>            <dbl> <dbl>
1 36005 Bronx County, New York total_pop      1419250    NA
2 36005 Bronx County, New York poverty          73028  2745
3 36005 Bronx County, New York median_income    49036   872
4 36005 Bronx County, New York unemployment     71936  2730
5 36047 Kings County, New York total_pop      2646306    NA
6 36047 Kings County, New York poverty          89471  2617
```

Cleaning the ACS5 datasets based on insights above.

```
acs_cl <- acs5 %>%

  #pivoting wider for proper merging later
  #the variable names become columns
  #values from the estimate and moe to be the row values of the variable names
  pivot_wider(names_from = variable,
              values_from = c(estimate, moe)) %>%

  #renaming the borough names
  mutate(
    borough = case_when(
      NAME == "Bronx County, New York"    ~ "BRONX",
      NAME == "Kings County, New York"    ~ "BROOKLYN",
      NAME == "New York County, New York" ~ "MANHATTAN",
      NAME == "Queens County, New York"   ~ "QUEENS",
      NAME == "Richmond County, New York" ~ "STATEN ISLAND")) %>%

  #ungrouping
  ungroup() %>%

  #standardizing the column names
  clean_names() %>%

  #removing unnecessary columns
  select(-geoid, -name)
```

Rechecking structure to ensure cleaning was successful.

```
#checking the structure
str(acs_cl)
```

```
tibble [5 × 9] (S3: tbl_df/tbl/data.frame)
 $ estimate_total_pop    : num [1:5] 1419250 2646306 1627788 2330124 492734
 $ estimate_poverty      : num [1:5] 73028 89471 38610 51767 9935
 $ estimate_median_income: num [1:5] 49036 78548 104553 84961 98290
 $ estimate_unemployment : num [1:5] 71936 100919 66581 85295 13201
 $ moe_total_pop         : num [1:5] NA NA NA NA NA
 $ moe_poverty           : num [1:5] 2745 2617 2523 1876 830
 $ moe_median_income     : num [1:5] 872 1052 1777 880 2957
 $ moe_unemployment      : num [1:5] 2730 3082 3110 2847 1214
 $ borough               : chr [1:5] "BRONX" "BROOKLYN" "MANHATTAN" "QUEENS" ...
```

Saving to CSV file.

```
#save to csv
write_csv(acs_cl, "data/processed/acs_cl.csv")
```

4.4 Merging Datasets

Combined 311 outcome data with borough-level SES indicators borough x year and then joined budgets to years.

```r
#reading in the service requests outcome
ser_reqs <- read.csv("data/processed/ser_reqs_out.csv")

#reading the agg budgets
agency_budgets <- read.csv("data/processed/budgets_agg.csv")

#reading the acs5 data
acs5_data <- read.csv("data/processed/acs_cl.csv")
```

```r
merged_datasets <- ser_reqs %>%

  #using left join to merge the acs5 dataset first by borough
  #this will map the acs5 to all years x borough
  left_join(acs5_data, by = c("borough")) %>%

  #using left join to merge with budgets by year
  #this will map the yearly budgets to all boroughs based on the year
  left_join(agency_budgets, by = "year")
```

```r
#checking the first few rows
head(merged_datasets)
```

```
  X.x borough year volume closed resolution_rate mean_response
1   1   BRONX 2019 439582 439324       0.9994131      9.037979
2   2   BRONX 2020 597366 597277       0.9998510      6.789537
3   3   BRONX 2021 650373 650134       0.9996325      7.015387
4   4   BRONX 2022 723443 723070       0.9994844      7.248069
5   5   BRONX 2023 618734 618383       0.9994327      9.304014
6   6   BRONX 2024 731317 730964       0.9995173      7.982609
  estimate_total_pop estimate_poverty estimate_median_income
1            1419250            73028                  49036
2            1419250            73028                  49036
3            1419250            73028                  49036
4            1419250            73028                  49036
5            1419250            73028                  49036
6            1419250            73028                  49036
  estimate_unemployment moe_total_pop moe_poverty moe_median_income
1                 71936            NA        2745               872
2                 71936            NA        2745               872
3                 71936            NA        2745               872
4                 71936            NA        2745               872
5                 71936            NA        2745               872
6                 71936            NA        2745               872
  moe_unemployment X.y      budget
```

```
1              2730   1 39448458298
2              2730   2 41399018536
3              2730   3 43309639214
4              2730   4 44190185299
5              2730   5 46982820611
6              2730   6 49976149672
```

```
#writing to csv
write_csv(merged_datasets, "data/processed/merged_datasets.csv")
```

5 Exploratory Data Analysis (EDA)

Correlations: Visualizing the correlations between SES indicators and budgets with 311 service requests outcomes

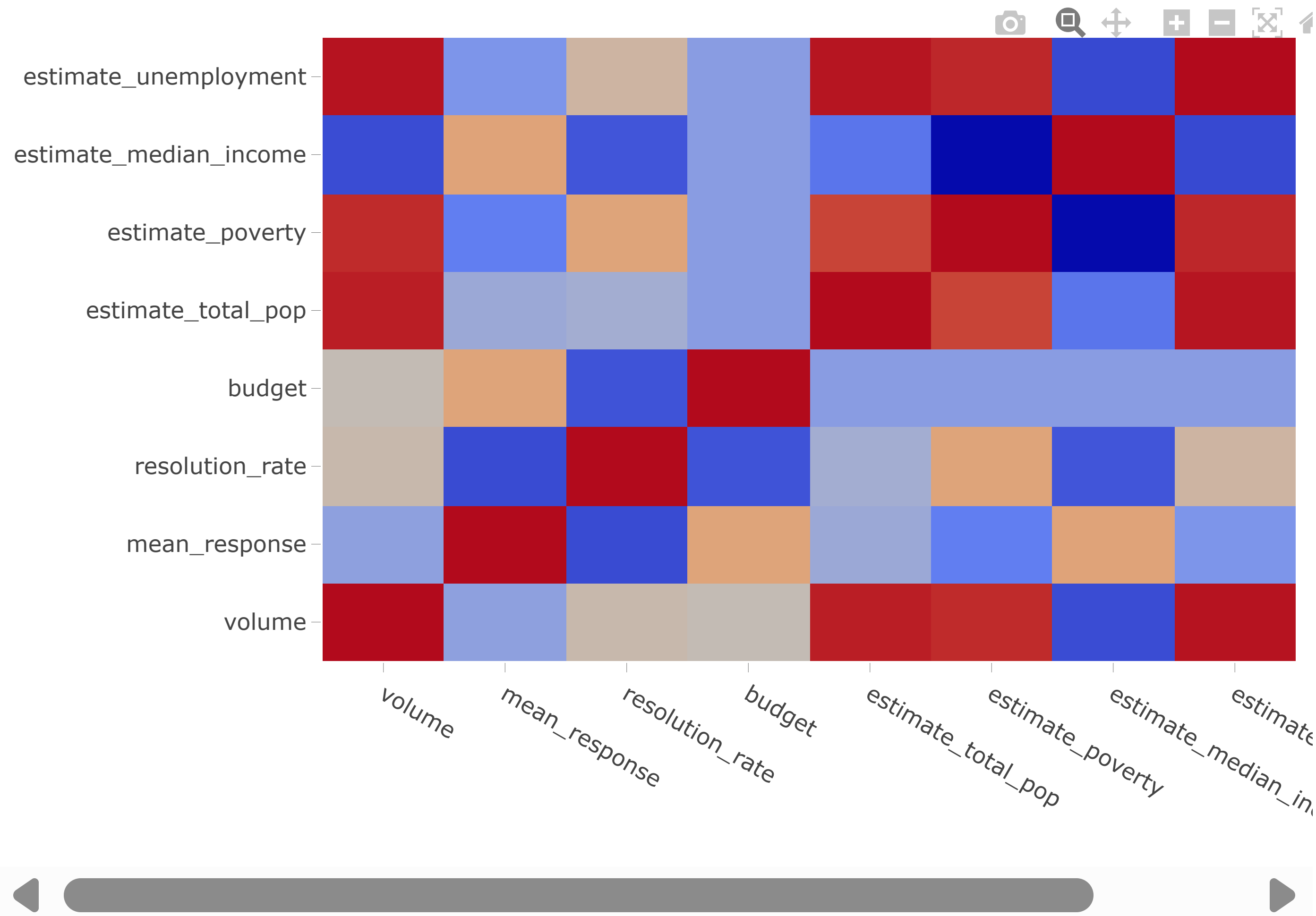
```
#reading in my saved file of final merged datasets
merged_df <- read.csv("data/processed/merged_datasets.csv")
```

5.1 Correlation Heatmap

Plotting the correlations in a heatmap for ease of visually reading.

```
#define a variable to hold the columns to pass for correlation
corr_df <- merged_df %>%

  #select the columns
  select(volume, mean_response, resolution_rate,
         budget, starts_with("estimate_"))

#define a variable to hold the correlations
corr_mat <- cor(corr_df, use = "pairwise.complete.obs")
```

```
#plot the correlations interactively with plot_ly
#setting x as the column names (the variables)
plot_ly(x = colnames(corr_mat),

        #setting y as the column names (the variables)
        y = rownames(corr_mat),

        #setting z correlations calculated
        z = corr_mat,

        #setting plot type as heatmap
        type = "heatmap",

        #setting the color scheme
        colorscale = "RdYlBu")
```



Based on the correlation matrix:

Predictor	Volume	Response Time	Resolution Rate
Population	+0.93	+0.05	+0.08
Median Income	-0.37	+0.37	-0.33
Poverty	+0.88	-0.14	+0.36
Unemployment	+0.96	-0.37	+0.24
Budgets	+0.19	+0.37	-0.34

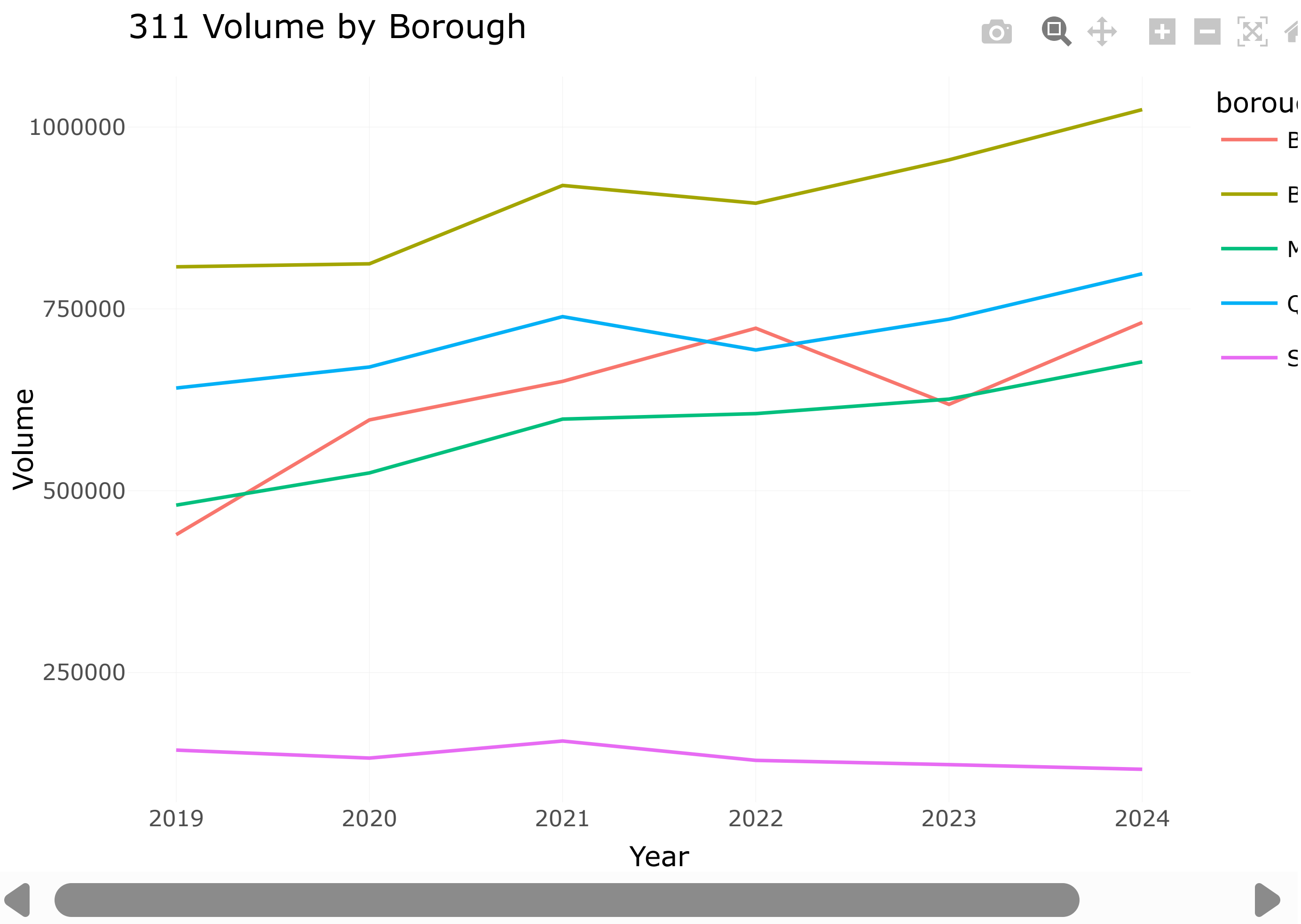
5.2 Trends

```
#creating a function to generate trend plots for the three outcomes
plot_outcome <- function(df, outcome, title, ylab) {

  #setting args to pass (df, outcome)
  #x defined as year
  #color defined based on borough (will show diff colors for each)
  ggplotly(ggplot(df, aes(x = year,
                          y = .data[[outcome]],
```

```
                        color = borough)) +

             #plot line chart
             geom_line() +

             #setting the args to pass
             #default arg for year
             labs(title = title, x = "Year", y = ylab) +

             #setting theme as minimal
             theme_minimal())}
```

5.2.1 Volume

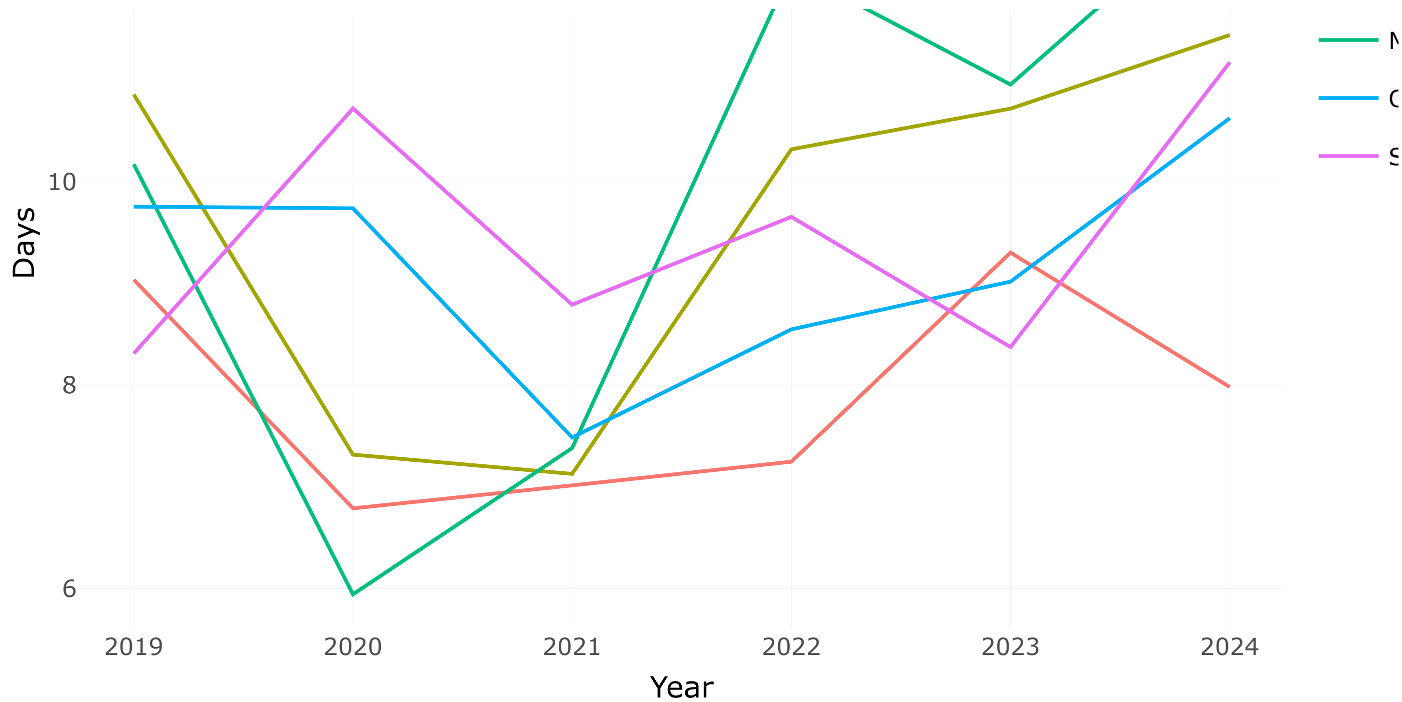
```
#calling the function
#passing the merged dataset as the df
#passing the outcome as volume
plot_outcome(merged_df, "volume",

             #setting the title
             title = "311 Volume by Borough",

             #setting the ylab (axis title)
             ylab  = "Volume")
```



Trends in volume per borough show that Brooklyn has consistently had the highest volume of 311 service requests, while Staten Island has had the least. Bronx, Queens, and Manhattan have similar volume throughout the years. Uniformly, there was an increase in volume after 2020, which coincided with COVID-19.

5.2.2 Response Time

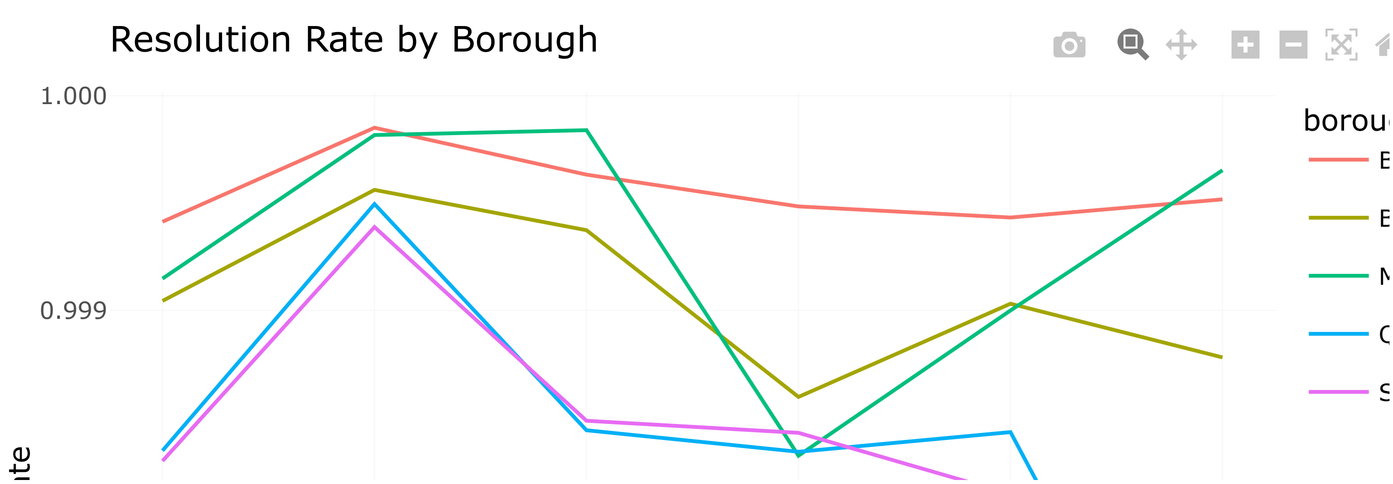
```
#calling the function
#passing the merged dataset as the df
#passing the outcome as mean response
plot_outcome(merged_df, "mean_response",

             #setting the title
             title = "Response Times by Borough",

             #setting the ylab (axis title)
             ylab  = "Days")
```

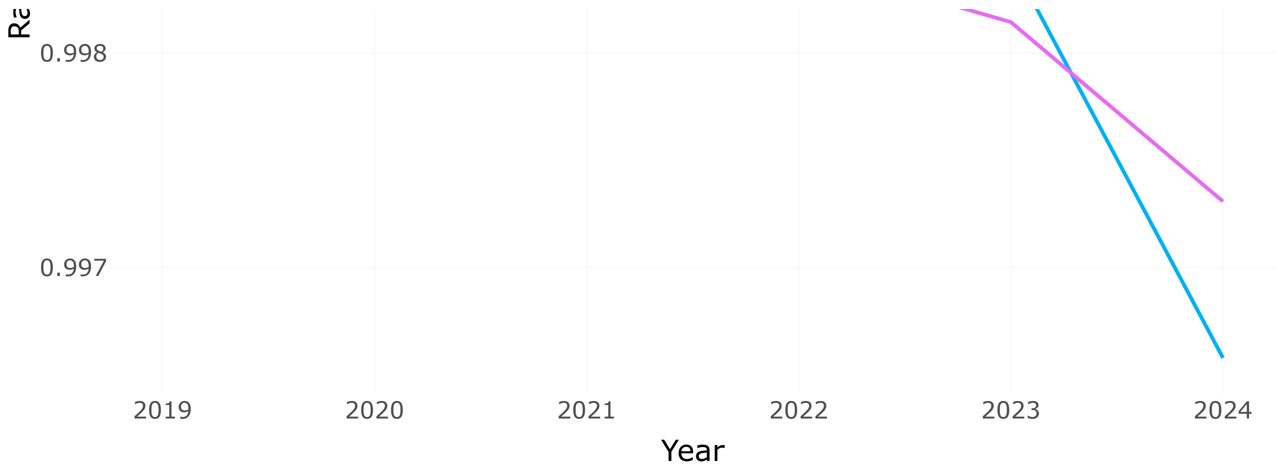


Trends in response times per borough show that Manhattan has higher response times throughout the years, while Staten Island has had lower response times throughout the years. Bronx, Brooklyn, and Queens have had similar response times throughout the years. Uniformly, there was an increase in response times after 2020, which coincided with COVID-19.

5.2.3 Resolution Rate

```
#calling the function
#passing the merged dataset as the df
#passing the outcome as resolution rate
plot_outcome(merged_df, "resolution_rate",

          #setting the title
          title = "Resolution Rate by Borough",

          #setting the ylab (axis title)
          ylab  = "Rate")
```





Trends in resolution rates show that resolution rates have been similar throughout the boroughs up until 2021. Uniformly, resolution rates have decreased since 2020, which coincided with COVID-19.

6 Models

Tested the effects of budgets and borough-level SES indicators (ACS5) on predicting 311 outcomes (LM + GAM)

Defining the arguments to be passed to the LM and GAM model functions.

```
#defining variable to hold budget as a string to pass
budget <- "budget"

#defining variable to hold volume as a string to pass
volume <- "volume"

#defining variable to hold mean_response as a string to pass
mean_response <- "mean_response"

#defining variable to hold resolution_rate as a string to pass
resolution_rate <- "resolution_rate"

#defining a vector of strings to hold SES vars to pass
vars <- c(
  "estimate_total_pop", "estimate_poverty",
  "estimate_median_income", "estimate_unemployment")
```

Creating function to calculate RMSE per model. The RMSE is used to evaluate the models predictive accuracy.

```
#defining rmse function
#arg to pass is the model once performed
model_rmse <- function(model) {

  #extracting the residuals from model
```

```
  #squaring each residual
  #averaging the squared residuals
  #taking the square root of the mean of squared residuals
  sqrt(mean(model$residuals^2,

            #ignoring any missing values
            na.rm = TRUE))}
```

6.1 Linear Models

```
#defining a function to run the subsequent models
#args to pass are:
#the outcome (predefined above)
#ses_vars  (predefined above)
#budget  (predefined above)
#the dataset
run_lm_model <- function(outcome, vars, budget, df) {

  #combining args passed into formula string
  formula_str <- paste(outcome, "~",
                       paste(c(vars, budget), collapse = " + "))

  #converting the string into a formula to pass
  #setting data to the df passed
  lm(as.formula(formula_str), data = df)}
```

6.1.1 Volume

```
#running model and storing in variable
#passing predefined args
lm_volume <- run_lm_model(outcome = volume,
                          vars = vars,
                          budget = budget,
                          df = merged_df)

#checking model summary
summary(lm_volume)
```

```
Call:
lm(formula = as.formula(formula_str), data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-118953  -20206    1890   29351   97035

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)            -7.587e+05  1.534e+05  -4.945 4.77e-05 ***
estimate_total_pop     -1.792e-02  6.290e-02  -0.285 0.778162    
estimate_poverty        2.356e+00  1.204e+00   1.956 0.062153 .  
estimate_median_income  1.496e+00  1.019e+00   1.468 0.155115    
estimate_unemployment   7.404e+00  1.860e+00   3.981 0.000553 ***
budget                  1.431e-05  2.630e-06   5.442 1.36e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 50010 on 24 degrees of freedom
Multiple R-squared:  0.971, Adjusted R-squared:  0.9649 
F-statistic: 160.6 on 5 and 24 DF,  p-value: < 2.2e-16
```

```
#getting rmse
model_rmse(lm_volume)
```

```
[1] 44734.1
```

6.1.1.0.1 Model Summary

The linear model (volume ~ SES + budget) explains 97.1% of the variance in volume in NYC boroughs between 2019-2024. The model is statistically significant overall (F-test $p < 0.001$), indicative that the predictors in the model do explain collectively the variations in volume.

The linear model under-predicted volume by 118953 service requests and over-predicted by 97035 service requests. On average, the linear model is about 44734 (RMSE) service requests from the true total volume.

- **Total Population (Negative, Not Statistically Significant p <0.78)**
    - For every one-unit increase in total population, holding all other predictors constant, the model predicts an decreases by ~1.792e-02 service requests in volume.
- **Poverty (Positive, Marginally Statistically Significant p = 0.06)**
    - For every one-unit increase in poverty, holding all other predictors constant, the model predicts an increases by ~2.356e+00 service requests in volume.
- **Median Income (Positive, Not Statistically Significant p = 0.16)**
    - For every one-unit increase in median income, holding all other predictors constant, the model predicts an increases by ~ 1.496e+00 service requests in volume.
- **Unemployment (Positive, Highly Statistically Significant p <0.001)**
    - For every one-unit increase in unemployment, holding all other predictors constant, the model predicts a increases by ~ 7.404e+00 service requests in volume.
- **Budget (Positive, Highly Statistically Significant p <0.001)**
    - For every one-unit increase in budget, holding all other predictors constant, the model predicts an increases by ~ 1.431e-05 service requests in volume.

6.1.2 Response Time

```
#running model and storing in variable
#passing predefined args
lm_response_time <- run_lm_model(outcome = mean_response,
                                 vars = vars,
                                 budget = budget,
                                 df = merged_df)


#checking model summary
summary(lm_response_time)
```

```
Call:
lm(formula = as.formula(formula_str), data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.4423 -0.7068  0.2693  1.0182  2.1921 

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)  
(Intercept)            -3.415e+00  4.931e+00  -0.692   0.4953  
estimate_total_pop      4.114e-07  2.022e-06   0.203   0.8405  
estimate_poverty        2.461e-05  3.871e-05   0.636   0.5309  
estimate_median_income  4.783e-05  3.276e-05   1.460   0.1573  
estimate_unemployment  -2.021e-05  5.978e-05  -0.338   0.7382  
budget                  1.818e-10  8.453e-11   2.150   0.0418 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.608 on 24 degrees of freedom
Multiple R-squared:  0.3012,	Adjusted R-squared:  0.1557 
F-statistic: 2.069 on 5 and 24 DF,  p-value: 0.1047
```

```
#getting rmse
model_rmse(lm_response_time)
```

```
[1] 1.437795
```

6.1.2.0.1 Model Summary

The linear model (response_time ~ SES + budget) explains 30.1% of the variance in average response times in NYC boroughs between 2019-2024. The model is not statistically significant overall (F-test p = 0.105), indicative that the predictors in the model do not collectively explain the variations in response time.

The linear model under-predicted response times by 3.4 days and over-predicted by 2.2 days. On average, the linear model is about 1.4 days from the true response time (RMSE).

- **Total Population (Positive, Not Statistically Significant p = 0.84)**
  - For every one-unit increase in total population, holding all other predictors constant, the model predicts response times increases by ~ 4.114e-07 days.
- **Poverty (Positive, Not Statistically Significant p =0.53)**
  - For every one-unit increase in poverty, holding all other predictors constant, the model predicts response times increases by ~ 2.461e-05 days.
- **Median Income (Positive, Not Statistically Significant p = 0.16)**
  - For every one-unit increase in median income, holding all other predictors constant,the model predicts average times increases by ~ 4.783e-05 days.
- **Unemployment (Negative, Not Statistically Significant p =0.73)**
  - For every one-unit increase in unemployment, holding all other predictors constant, the model predicts response times decreases by ~2.021e-05 days.
- **Budget (Positive, Statistically Significant p <0.05)**
  - For every one-unit increase in budget, holding all other predictors constant,the model predicts response times increases by ~ 1.818e-10 days.

6.1.3 Resolution Rate

```r
#running model and storing in variable
#passing predefined args
lm_resolution_rate <- run_lm_model(outcome = resolution_rate,
                                   vars = vars,
                                   budget = budget,
                                   df = merged_df)

#checking model summary
summary(lm_resolution_rate)
```

```

Call:
lm(formula = as.formula(formula_str), data = df)

Residuals:
       Min         1Q     Median         3Q        Max 
-1.269e-03 -3.492e-04  7.906e-05  2.768e-04  1.014e-03 

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)             9.994e-01  1.682e-03 594.121  < 2e-16 ***
estimate_total_pop     -2.812e-09  6.896e-10  -4.078 0.000432 ***
estimate_poverty        2.337e-08  1.320e-08   1.770 0.089400 .  
```

```
estimate_median_income  2.638e-08  1.118e-08   2.361 0.026707 *
estimate_unemployment   6.202e-08  2.039e-08   3.042 0.005619 **
budget                 -7.387e-14  2.884e-14  -2.562 0.017121 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0005483 on 24 degrees of freedom
Multiple R-squared:  0.5778,	Adjusted R-squared:  0.4899
F-statistic:  6.57 on 5 and 24 DF,  p-value: 0.0005544
```

```r
#getting rmse
model_rmse(lm_resolution_rate)
```

```
[1] 0.0004904417
```

6.1.3.0.1 Model Summary

The linear model (resolution_rate ~ SES + budget) explains 57.8% of the variance in average response times in NYC boroughs between 2019-2024. The model is statistically significant overall (F-test $p < 0.001$), indicative that the predictors in the model do collectively explain the variations in resolution rates.

The linear model under-predicted resolution rates by ~ 1.269e-03 and over-predicted by ~ 1.014e-03. On average, the linear model is about 0.0005 (RMSE) from the true resolution rate.

- **Total Population (Negative, Highly Statistically Significant p <0.001)**
  - For every one-unit increase in total population, holding all other predictors constant, the model predicts resolution rates decreases by ~ -2.812e-09.
- **Poverty (Positive, Marginally Statistically Significant p = 0.09)**
  - For every one-unit increase in poverty, holding all other predictors constant, the model predicts resolution rates increases by ~2.337e-08.
- **Median Income (Positive, Statistically Significant p <0.05)**
  - For every one-unit increase in median income, holding all other predictors constant, the model predicts resolution rates increases by ~2.638e-08.
- **Unemployment (Positive, Very Statistically Significant p <0.01)**
  - For every one-unit increase in unemployment, holding all other predictors constant, the model predicts resolution rates increases by ~6.202e-08.
- **Budget (Negative, Statistically Significant p <0.05)**
  - For every one-unit increase in budget, holding all other predictors constant, the model predicts resolution rates decreases by ~7.387e-14.

6.2 GAM

```r
#creating function to run GAM model
#taking in args for outcome, ses vars, budget, and df
#these are predefined above
run_gam_model <- function(outcome, vars, budget, df) {

  #building smooth terms for SES + budget
  smooth_terms <- paste0("s(", c(vars, budget), ", k = 4)")

  #combining args passed into formula string
  formula_str <- paste0(outcome, " ~ ",
                        paste(smooth_terms, collapse = " + "))

  #converting the string into a formula to pass
  #setting data to the df passed
  gam(as.formula(formula_str), data = df)}
```

6.2.1 Volume

```r
#running model and storing in variable
#passing predefined args
gam_volume <- run_gam_model(outcome = volume,
                            vars = vars,
                            budget = budget,
                            df = merged_df)

#checking model summary
summary(gam_volume)
```

```
Family: gaussian
Link function: identity

Formula:
volume ~ s(estimate_total_pop, k = 4) + s(estimate_poverty, k = 4) +
    s(estimate_median_income, k = 4) + s(estimate_unemployment,
    k = 4) + s(budget, k = 4)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   592231       8795   67.33   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                            edf Ref.df     F  p-value
s(estimate_total_pop)     1.000  1.000 0.006    0.939
s(estimate_poverty)       1.000  1.000 0.674    0.420
s(estimate_median_income) 1.000  1.000 1.246    0.276
s(estimate_unemployment)  1.000  1.000 0.950    0.340
```

```
s(budget)                 2.047  2.424 14.197 6.32e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.967   Deviance explained = 97.4%
GCV = 3.0333e+09  Scale est. = 2.3207e+09  n = 30
```

```r
#getting rmse
model_rmse(gam_volume)
```

```
[1] 42137.56
```

6.2.1.0.1 Model Summary

The GAM model (volume ~ s(SES) + s(budget)) explains 97.4% of the variance in volume in NYC boroughs between 2019-2024. On average, the GAM model is about 42138 (RMSE) service requests from the true total volume.

- **Total Population (Not Statistically Significant, No Nonlinear Effects)**
  - Estimated effect is not nonlinear (edf ~ 1), and not statistically significant ($p = 0.939$).
- **Poverty (Not Statistically Significant, No Nonlinear Effects)**
  - Estimated effect is not nonlinear (edf ~ 1), and not statistically significant ($p = 0.420$).
- **Median Income (Not Statistically Significant, No Nonlinear Effects)**
  - Estimated effect is not nonlinear (edf ~ 1), and not statistically significant ($p = 0.276$).
- **Unemployment (Not Statistically Significant, No Nonlinear Effects)**
  - Estimated effect is not nonlinear (edf ~ 1), and weakly statistically significant ($p = 0.340$).
- **Budget (Highly Statistically Significant, Nonlinear Effects)**
  - Estimated effect is nonlinear (edf ~ 2.047), and highly statistically significant ($p < 0.001$).

6.2.2 Response Time

```r
#running model and storing in variable
#passing predefined args
gam_mean_response <- run_gam_model(outcome = mean_response,
                                   vars = vars,
                                   budget = budget,
                                   df = merged_df)


#checking model summary
summary(gam_mean_response)
```

```
Family: gaussian
Link function: identity

Formula:
mean_response ~ s(estimate_total_pop, k = 4) + s(estimate_poverty,
    k = 4) + s(estimate_median_income, k = 4) + s(estimate_unemployment,
    k = 4) + s(budget, k = 4)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.2256     0.2693   34.25   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                            edf Ref.df     F p-value
s(estimate_total_pop)     1.000  1.000 0.001  0.9785
s(estimate_poverty)       1.000  1.000 0.007  0.9349
s(estimate_median_income) 1.000  1.000 0.247  0.6239
s(estimate_unemployment)  1.000  1.000 0.001  0.9729
s(budget)                 2.225  2.593 4.256  0.0306 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.289   Deviance explained = 44.2%
GCV = 2.8668  Scale est. = 2.1763    n = 30
```

```
#getting rmse
model_rmse(gam_mean_response)
```

```
[1] 1.285358
```

6.2.2.0.1 Model Summary

The GAM model (response_time~ s(SES) + s(budget)) explains 44.2% of the variance in response time in NYC boroughs between 2019-2024. On average, the GAM model is about 1.3 days (RMSE) from the true average response time.

- **Total Population (Not Statistically Significant, No Nonlinear Effects)**
  - Estimated effect is not nonlinear (edf ~ 1), and not statistically significant (p = 0.979).
- **Poverty (Not Statistically Significant, No Nonlinear Effects)**
  - Estimated effect is not nonlinear (edf ~ 1), and not statistically significant (p = 0.935).
- **Median Income (Not Statistically Significant, No Nonlinear Effects)**
  - Estimated effect is not nonlinear (edf ~ 1), and not statistically significant (p = 0.624).
- **Unemployment (Not Statistically Significant, No Nonlinear Effects)**

- Estimated effect is not nonlinear (edf ~ 1), and not statistically significant (p = 0.973).

- **Budget (Statistically Significant, Nonlinear Effects)**
  - Estimated effect is nonlinear (edf ~ 2.225), and highly statistically significant (p < 0.05).

6.2.3 Resolution Rate

```
#running model and storing in variable
#passing predefined args
gam_resolution_rate <- run_gam_model(outcome = resolution_rate,
                                     vars = vars,
                                     budget = budget,
                                     df = merged_df)


#checking model summary
summary(gam_resolution_rate)
```

```

Family: gaussian
Link function: identity

Formula:
resolution_rate ~ s(estimate_total_pop, k = 4) + s(estimate_poverty,
    k = 4) + s(estimate_median_income, k = 4) + s(estimate_unemployment,
    k = 4) + s(budget, k = 4)

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 9.989e-01  9.976e-05   10013   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                           edf Ref.df     F p-value
s(estimate_total_pop)     1.00  1.000 6.088  0.0211 *
s(estimate_poverty)       1.00  1.000 1.638  0.2128
s(estimate_median_income) 1.00  1.000 4.628  0.0417 *
s(estimate_unemployment)  1.00  1.000 2.859  0.1038
s(budget)                 1.15  1.282 5.688  0.0258 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.494   Deviance explained = 58.3%
GCV = 3.7552e-07  Scale est. = 2.9854e-07  n = 30
```

```
#getting rmse
model_rmse(gam_resolution_rate)
```

```
[1] 0.0004871734
```

6.2.3.0.1 Model Summary

The GAM model (volume ~ s(SES) + s(budget)) explains 58.3% of the variance in resolution rates in NYC boroughs between 2019-2024. On average, the GAM model is about 0.005 (RMSE percentage points from the resolution rates.

- **Total Population (Statistically Significant, No Nonlinear Effects)**
  - Estimated effect is not nonlinear (edf ~ 1), and is statistically significant ($p < 0.05$)
- **Poverty (Not Statistically Significant, No Nonlinear Effects)**
  - Estimated effect is not nonlinear (edf ~ 1), and not statistically significant ($p = 0.212$).
- **Median Income (Statistically Significant, No Nonlinear Effects)**
  - Estimated effect is not nonlinear (edf ~ 1), and is statistically significant ($p < 0.05$)
- **Unemployment (Not Statistically Significant, No Nonlinear Effects)**
  - Estimated effect is not nonlinear (edf ~ 1), and not statistically significant ($p = 0.104$).
- **Budget (Statistically Significant, Slight Nonlinear Effects)**
  - Estimated effect is slightly nonlinear (edf ~ 1.15), and is statistically significant ($p < 0.05$).

6.3 Overall Findings

- **Volume**
  - Budgets have positive highly statistically significant linear and nonlinear effects on volume. Given that adopted agency budgets are informed by previous year performance, this could signal that increased demand in previous years drive the need for increases in funding.
  - Unemployment has positive highly statistically significant linear effects on volume. This could be due to periods of economic hardship, suggesting that as economic strain rises, so does the demand for reporting public service issues.
- **Response Time**
  - Budgets have positive statistically significant linear and statistically significant nonlinear effects on response times. Given that budgets are also statistically significant with volume, potentially in response to previous year performance, workloads potentially have increased as well, signalling an increase in response times (more workload means less capacity).
- **Resolution Rate**
  - Budgets have negative statistically significant linear effects on resolution rates. This could be that although budgets are increasing, there may be allocation issues corresponding to the ability to resolve service requests.

- Unemployment has positive very statistically significant linear effects on resolution rates. This could be due to the capacity of individuals who file service complaints to engage with the process of resolving their service requests.
- Median income has positive statistically significant linear effects on resolution rates. This could be due to individuals who are more affluent having higher political and social capital leading to higher expectations of service delivery.
- Total population has negative very statistically significant linear effects on resolution rates. This could be due to the growth of demand in service requests outpacing the capacity to address them.

7 Data Quality & Limitations

7.1 311 Service Requests

- **Recording Accuracy**: In terms of the recording accuracy of this dataset, it has standardized fields for date/time, location, service category, resolution status, agency, etc.
- **Conceptual Suitability:** In terms of conceptual suitability, this dataset is overall fit and relevant to the overarching question and specific questions in providing the information needed to determine 311 service request volume, response times, and resolution rates.
- **Sampling Accuracy**: In terms of the sampling accuracy of this dataset, it represents the population of residents who 1) are willing to submit service requests, 2) have knowledge of how to submit service requests, and 3) have access to phone/mobile/internet to submit service requests. This introduces a sampling bias whereby residents with greater access and awareness of 311 are represented in the dataset. Furthermore, seasonal factors as inclement weather can lead to spikes in service request volume which introduces a sampling bias whereby the magnitude of service requests would be impacted by external factors.
- **Limitations:** Missing values in this dataset were not predefined as NANs and therefore a deeper dive at missing values needed to be conducted. This identified the need to remove rows with “Unspecified” and “NA” boroughs. Some service requests had less than 0 or greater than 365 days response time, which would inflate analysis and modeling. Rows with these two types of issues with response times were removed.

7.2 Budget Data

- **Recording Accuracy**: In terms of the recording accuracy of this dataset, it is high due to the legal mandates tied to financial processes and reporting. Financial budgets for each agency is reported yearly as preliminary budget, executive budget, and adopted budget. The former two, preliminary and executive budgets, are proposals that are not finalized contingent on state and federal allocations as well as council negotiations. The latter, adopted budget, reflects the finalized budget amount after state and federal allocations have been determined and negotiations are final.

- **Sampling Accuracy**: In terms of the sampling accuracy of this dataset, no sampling bias is of concern as budgets are not sampled but rather determined through financial processes.
- **Conceptual Suitability:** In terms of conceptual suitability, this dataset is overall fit and relevant to the overarching question and specific question in providing the information needed for yearly city-wide agency-specific budgets.
- **Limitations:** The expense budgets for NYC run on a fiscal year (July 1 - June 30) which does not coincide with the calendar year. Therefore, a transformation of the budgets was performed, whereby fiscal years were split in two and the total for a year was the summation of half the previous year and half the current year's budgets.

7.3 ACS Socioeconomic Indicators

- **ACS5**: These indicators were released in 2023 and represent the 5-year average between 2019-2023 and were used as the proxy estimates for borough-level estimates for 2019-2024.
- **Conceptual Suitability:** In terms of conceptual suitability, this dataset is overall fit and relevant to the overarching question and specific question in providing the information needed to for socio-economic indicators
- **Recording Accuracy**: In terms of the recording accuracy of this dataset, it is conducted yearly by the U.S. Census Bureau and disseminated by the NYC Department of Planning. Data is collected via mail, online, and Computer Assisted Interviewing (CAPI). This multimodal approach to data collection increases recording accuracy but does introduce recording risks such as interviewer error, mistakes with data entry, and misunderstanding from interviewee, leading to nonsampling errors.
- **Sampling Accuracy**: In terms of the sampling accuracy of this dataset, in conducting this survey, the Census draws from the Master Address File to ensure representation. Annual sample size target is 3.5 million housing units, whereby sampling rates are stratified by geography. Margin of errors are reported with a 90% confidence interval (U.S. Census Bureau (2022).
- **Limitations**: A limitation with ACS5 data is that given its average across time, it does not provide year-to-year insights, meaning that annual variation is not as explicit and cannot be determined. Another limitation is the imputation of the estimates for 2024 was based on the 5-year average between 2019-2024, albeit using the statistical method of last observation carried forward (LOCF).

8 Dashboard

The dashboard developed was hosted using shinyapps.io and can be found here. At the moment it has a run time of 8hrs, and will be refreshed at 8am and 6pm everyday.

8.1 Flexdashboard Code

Trends

Inputs

.sidebar

```
{r}
#this will be in a sidebar (.sidebar)
#creating an input widget
#inputId set to borough for selecting later (input$borough)
selectInput(inputId = "borough",

            #label displayed (above the dropdown for boroughs)
            label = "Select Borough",

            #getting the unique values of the borough for dropdown
            choices = unique(merged_sum$borough),

            #default borough
            selected = "BRONX")

#displays the selected borough
#will show the selected boroughs name through concatenation (label +
input)
renderText({paste("Current Borough:",
                  input$borough)})

#creating a download button to download to download the merged dataset
downloadButton("downBtn", "Download CSV")
```

```
{r}
#defininign function to create plots for trends
#passing arguments for the dataset, borough input, outcome, title,
ylab (axis title)
plotly_trend <- function(df,
```

```{r}
#definign function to create plots for trends
#passing arguments for the dataset, borough input, outcome, title, ylab (axis title)
plotly_trend <- function(df,
                         borough_input,
                         outcome,
                         title,
                         ylab) {
  #for interactivity
  ggplotly(df %>%

             #filtering for the borough from the input
             filter(borough %in% borough_input) %>%

             #setting x-axis as year column
             ggplot(aes(x = year,

                        #setting y-axis as outcome variable
                        y = .data[[outcome]],

                        #setting the color lines by borough
                        color = borough)) +

             #drawing this as a trend line
             geom_line() +

             #adding points for each year
             geom_point() +

             #setting the labs
             labs(

               #title as the title to be passed
               title = title,

               #x-axis lable as year
               x = "Year",

               #y-axis label to be passed
               y = ylab) +

             #setting theme as minimal
             theme_minimal())}
```

311 Volume by Agency

```{r}
#wrapping plot in renderPlotly to render plot using shiny app
renderPlotly({

  #setting df as the read in df
  plotly_trend(df = merged_sum,

               #setting the input as the borough selected
               borough_input = input$borough,

               #setting outcome outcome
               outcome = "volume",

               #setting the prefix for title
               #suffix is the borough selected
               title = paste("311 Volume -", input$borough),

               #setting the y-axis title
               ylab = "Total Requests")})
```

Mean Response Time (Days)

```{r}
#wrapping plot in renderPlotly to render plot using shiny app
renderPlotly({

  #setting df as the read in df
  plotly_trend(df = merged_sum,

               #setting the input as the borough selected
               borough_input = input$borough,

               #setting outcome as mean response
               outcome = "mean_response",

               #setting the prefix for title
               #suffix is the borough selected
               title = paste("Mean Response Time -", input$borough),

               #setting the y-axis title
               ylab = "Response Time (Days)")})
```

Resolution Rate

```{r}
#wrapping plot in renderPlotly to render plot using shiny app
renderPlotly({

  #setting the input as the borough selected
  plotly_trend(df = merged_sum,

               #setting the input as the borough selected
               borough_input = input$borough,

               #setting outcome as resolution rate
               outcome = "resolution_rate",

               #setting the prefix for title
               #suffix is the borough selected
               title = paste("Resolution Rate -", input$borough),

               #setting the y-axis title
               ylab = "Resolution Rate")})
```

```{r}
#referring to the selected borough from plot
#producing a text string to display the output
output$selected_borough <- renderText({



  #setting the suffix for the label text
  #referring to the selected borough from plot
  #converting vector to string
  paste("Current Borough:", paste(input$borough, collapse = ", "))})



  #converting the logic of the download buttong for the server
  output$downBtn <- downloadHandler(


  #setting the file name downloaded
  filename = function() {
    "data/processed/merged_acs5.csv"},


  #writing the content of the merged dataset read in to file
  content = function(file) {
    write.csv(merged_sum, file, row.names = FALSE)})
```

```r
{r}
#referring to the selected borough from plot
#producing a text string to display the output
output$selected_borough <- renderText({



  #setting the suffix for the label text
  #referring to the selected borough from plot
  #converting vector to string
  paste("Current Borough:", paste(input$borough, collapse = ", "))})



#converting the logic of the download buttong for the server
output$downBtn <- downloadHandler(


  #setting the file name downloaded
  filename = function() {
    "data/processed/merged_acs5.csv"},


  #writing the content of the merged dataset read in to file
  content = function(file) {
    write.csv(merged_sum, file, row.names = FALSE)})
```



```r
{r}
#referring to the selected borough from plot
#producing a text string to display the output
output$selected_borough <- renderText({

#setting the suffix for the label text
#referring to the selected borough from plot
#converting vector to string
paste("Current Borough:", paste(input$borough, collapse = ", "))})

#converting the logic of the download buttong for the server
output$downBtn <- downloadHandler(

  #setting the file name downloaded
  filename = function() {
    "data/processed/merged_acs5.csv"},

  #writing the content of the merged dataset read in to file
  content = function(file) {
    write.csv(merged_sum, file, row.names = FALSE)})
```

Correlations

```
{r}
#wrapping plot in renderPlotly to render plot using shiny app
renderPlotly({

  #define a variable to hold the columns to pass for correlation
  corr_mat <- merged_sum %>%

    #select the columns
    select(volume, mean_response, resolution_rate,
           budget, starts_with("estimate_")) %>%

    #computing correlations ensuring pairs use of only non-missing
values
    cor(use = "pairwise.complete.obs")

#plot the correlations interactively with plot_ly
  plot_ly(

    #setting x as the column names (the variables)
    x = colnames(corr_mat),


    #setting y as the column names (the variables)
    y = rownames(corr_mat),

    #setting z correlations calculated
    z = corr_mat,

    #setting plot type as heatmap
    type = "heatmap",

    #setting the color scheme
    colorscale = "RdYlBu")})
```

9 README

To run this qmd successfully,

1. Create a root folder and store this qmd file in the root folder.
2. Create a /data folder in root folder.
3. Create /raw folder in /data folder. Add the datasets from the submission file here (/data/raw).
   - The 311 Service Requests and Agency Budgets files have been shared in the submission of the assignment as truncated versions with only the necessary columns needed for file size management.
4. Create a /processed folder in /data folder to store processed data.

5. Run qmd file with CTRL + ALT + R to have the entire markdown file process or individually run each code chunk.

10 References

1. **Quarto Documentation**: Leveraged this reference to identify OMPL (Outline Processor Markup Language).

2. **flexdashboard Documentation** and **shiny Documentation:** Leveraged these references to learn to how to develop a flexdashboard.

3. **shinyapps.io Documentation**: Leveraged this reference to learn how to publish a flexdashboard.

4. **American Community Survey (5-year)**: Leveraged this reference to learn how to use tidycensus.

5. ***Analyzing US Census Data: Methods, Maps, and Models in R***: Leveraged this reference as a comprehensive documentation supplementing R documentation on tidycensus.

6. **Linear Regression In R**: Cheatsheet was leveraged as a reference for linear modeling.

7. **GAM Application Using R**: Leveraged this reference for GAM modeling.