Project 1: D.C. crime

TOTAL: 10pts

PURPOSE

To analyze crime in Washington, D.C. This project introduces you to connecting to data, creating new fields, and conducting trend analysis.

GOAL

Identify crime trends in Washington, DC over behavior, space, and time.

DATA

This project uses real live Washington, D.C. crime data. It’s located here.

METHODS

Step 0: Load and install packages

1. You will need two additional packages to install and load into your script:
    a. install.packages('tidyverse') # run this once, ever
    b. library(tidyverse) # run this every session
    c. install.packages('explore') # run this once, ever
    d. library(explore) # run this every session
    e. install.packages('GGally') # run this once, ever
    f. library(GGally) # run this every session

Step 1: Examine data

1. Take a look at the table - get comfortable with the fields captured, and how it is organized.
2. Look at the 2020 D.C. crime data here.
3. Take a look at the table - examine the fields captured, and how it is organized.

Step 2: Read data

1. In the top left pane in RStudio, add this line of code:
    - a. dc.data2020 <- read.csv("", stringsAsFactors = FALSE)
    - b. In between the quotes, look for the table ID found as part of the URL for the data source.
    - c. It should look something like this:
        - i. dc.data2020 <- read.csv("https://opendata.arcgis.com/datasets/f516e0dd7b614b088ad781b0c4002331_2.csv", stringsAsFactors = FALSE)
    - d. Highlight that line of code and Run it.
    - e. Success!
    - f. In the top right pane, click on the output to view it. Should be a table of all 2020 D.C. crimes to date. Close that table tab after viewing.
2. Repeat this process for 2021.
    - a. From the data website, edit the URL.
        - i. Huh?
        - ii. This is the URL: https://opendata.dc.gov/datasets/crime-incidents-in-2020/data
        - iii. Change the ‘2020’ to ‘2021’
    - b. Click on ‘I want to use this’ at the bottom left
    - c. Click on ‘View API Resources’
    - d. On GeoJSON, click the ‘copy link’ button
    - e. In the top left pane in RStudio, add this line of code:
        - i. dc.data2021 <- read.csv("", stringsAsFactors = FALSE)
        - ii. In between the quotes, paste the link you copied from the data website, and change the .json extension at the end to .csv
        - iii. Highlight that line of code and Run it.
        - iv. More success! A second table added to your environment.
3. Repeat the process from #2 for every year since 2008.
    - a. When complete, you’ll have yearly tables in your Environments pane.
    - b. Like this:
        - i. dc.data2025 <- read.csv("https://opendata.arcgis.com/datasets/74d924ddc3374e3b977e6f002478cb9b_7.csv", stringsAsFactors = FALSE)
        - ii. dc.data2024 <- read.csv("https://opendata.arcgis.com/datasets/c5a9f33ffca546babbd91de1969e742d_6.csv", stringsAsFactors = FALSE)
        - iii. dc.data2023 <- read.csv("https://opendata.arcgis.com/datasets/89561a4f02ba46cca3c42333425d1b87_5.csv", stringsAsFactors = FALSE)
        - iv. dc.data2022 <- read.csv("https://opendata.arcgis.com/datasets/f9cc541fc8c04106a05a1a4f1e7e813c_4.csv", stringsAsFactors = FALSE)
        - v. dc.data2021 <- read.csv("https://opendata.arcgis.com/datasets/619c5bd17ca2411db0689bb0a211783c_3.csv", stringsAsFactors = FALSE)

vi. dc.data2020 <- read.csv("https://opendata.arcgis.com/datasets/f516e0dd7b614b088ad781b0c4002331_2.csv", stringsAsFactors = FALSE)

vii. dc.data2019 <- read.csv("https://opendata.arcgis.com/datasets/f08294e5286141c293e9202fcd3e8b57_1.csv", stringsAsFactors = FALSE)

viii. dc.data2018 <- read.csv("https://opendata.arcgis.com/datasets/38ba41dd74354563bce28a359b59324e_0.csv", stringsAsFactors = FALSE)

ix. dc.data2017 <- read.csv("https://opendata.arcgis.com/datasets/6af5cb8dc38e4bcbac8168b27ee104aa_38.csv", stringsAsFactors = FALSE)

x. dc.data2016 <- read.csv("https://opendata.arcgis.com/datasets/bda20763840448b58f8383bae800a843_26.csv", stringsAsFactors = FALSE)

xi. dc.data2015 <- read.csv("https://opendata.arcgis.com/datasets/35034fcb3b36499c84c94c069ab1a966_27.csv", stringsAsFactors = FALSE)

xii. dc.data2014 <- read.csv("https://opendata.arcgis.com/datasets/6eaf3e9713de44d3aa103622d51053b5_9.csv", stringsAsFactors = FALSE)

xiii. dc.data2013 <- read.csv("https://opendata.arcgis.com/datasets/5fa2e43557f7484d89aac9e1e76158c9_10.csv", stringsAsFactors = FALSE)

xiv. dc.data2012 <- read.csv("https://opendata.arcgis.com/datasets/010ac88c55b1409bb67c9270c8fc18b5_11.csv", stringsAsFactors = FALSE)

xv. dc.data2011 <- read.csv("https://opendata.arcgis.com/datasets/9d5485ffae914c5f97047a7dd86e115b_35.csv", stringsAsFactors = FALSE)

xvi. dc.data2010 <- read.csv("https://opendata.arcgis.com/datasets/fdacfbdda7654e06a161352247d3a2f0_34.csv", stringsAsFactors = FALSE)

xvii. dc.data2009 <- read.csv("https://opendata.arcgis.com/datasets/73cd2f2858714cd1a7e2859f8e6e4de4_33.csv", stringsAsFactors = FALSE)

xviii. dc.data2008 <- read.csv("https://opendata.arcgis.com/datasets/180d56a1551c4e76ac2175e63dc0dce9_32.csv", stringsAsFactors = FALSE)

Step 3: Merge data

Combine all of the yearly tables into one with this:

- data.temp <- rbind(dc.data2008, dc.data2009, dc.data2010, dc.data2011, dc.data2012, dc.data2013, dc.data2014, dc.data2015, dc.data2016, dc.data2017, dc.data2018, dc.data2019, dc.data2020, dc.data2021, dc.data2022, dc.data2023, dc.data2024, dc.data2025)

Step 4: Clean data

Parse the 'REPORT_DAT' field into separate data and time fields:

- dc.data <- separate(data.temp, REPORT_DAT, into = c("date", "time"), sep = " ")

Step 5: Create data

Create six usable fields (columns) in your data:

1. Format the date column
   a. dc.data$date <- as.Date(dc.data$date, format = "%Y/%m/%d")
2. Create an hour of day column
   a. dc.data$hour <- substr(dc.data$time, 0, 2)
   b. dc.data$hour <- as.numeric(dc.data$hour)
3. Create day of week
   a. dc.data$dow <- weekdays(dc.data$date)
4. Create week of year
   a. dc.data$week <- format(as.Date(dc.data$date), "%U")
   b. dc.data$week <- as.numeric(dc.data$week)
5. Create month of year
   a. dc.data$months <- months(dc.data$date)
6. Create year
   a. dc.data$year <- substr(dc.data$date, 0, 4)
7. Seasons?

Step 6: Analyze data

1. Explore your data.
   a. explore(dc.data)
2. You'll get a new pop-up window.
   a. Change the 'variable' to different fields to analyze them.
   b. Also use the 'target' menu to add complexity.
   c. Try different variables.
   d. Try combinations of variables.
   e. Take some time and get loose. Identify trends, which are common themes across the data.
   f. Close this window when complete (your RStudio session stays 'active' until you close it)

HERE IS THE SCRIPT WE BUILT IN CLASS ON 22 JANUARY 2025

Step 7: Visualize data

1. Here's a (relatively) quick visualization for the numeric fields in these data:
   a. data.new <- select(dc.data, hour, DISTRICT, WARD, week, year)
   b. ggpairs(data.new) + theme_bw()
      i. But this gives you an error, because there are more than 15 unique year values… so we have to adjust the 'cardinality' parameter, like this:
         1. ggpairs(data.new, cardinality_threshold = xx) + theme_bw()

Step 8: Intro to graphing data

Bar graphs:

- Follow these iterative steps to make progressively better bar graphs:
  - ggplot()
  - ggplot(dc.data, aes(SHIFT))
  - ggplot(dc.data, aes(SHIFT)) + geom_bar(stat = "count")
  - ggplot(dc.data, aes(SHIFT)) + geom_bar(stat = "count", aes(fill = OFFENSE))
  - ggplot(dc.data, aes(SHIFT)) + geom_bar(stat = "count", aes(fill = OFFENSE), position = position_stack(reverse = TRUE))
  - ggplot(dc.data, aes(SHIFT)) + geom_bar(stat = "count", aes(fill = OFFENSE), position = position_stack(reverse = TRUE)) + theme_classic()
  - ggplot(dc.data, aes(SHIFT)) + geom_bar(stat = "count", aes(fill = OFFENSE), position = position_stack(reverse = TRUE)) + theme_classic() + theme(legend.position = "bottom")
  - ggplot(dc.data, aes(SHIFT)) + geom_bar(stat = "count", aes(fill = OFFENSE), position = position_stack(reverse = TRUE)) + theme_classic() + facet_wrap(~DISTRICT)

Formatting:

- Make your graph an object, and then you can customize the formatting:
  - ggplot(dc.data, aes(SHIFT)) + geom_bar(stat = "count", aes(fill = OFFENSE), position = position_stack(reverse = TRUE)) + theme_classic() +
  - labs(
  - title = "Your title here",
  - x = "x axis label here",
  - y = "y axis label here"
  -) +
  - theme(
  - panel.grid.minor = element_blank(),
  - panel.grid.major.y = element_blank(),
  - panel.grid.major.x = element_line(),
  - axis.ticks = element_blank()
  -)

Better graphs:

These aren’t mandatory, but a little cleaner and better than the examples above.

- First, some data prep to group crime by ‘Person’ and ‘Property’:
  - dc.data$TYPE <- case_when(
  - dc.data$OFFENSE %in%
  - c(
  - "ARSON",
  - "BURGLARY",
  - "MOTOR VEHICLE THEFT",
  - "THEFT F/AUTO",
  - "THEFT/OTHER") ~ "Property",
  - dc.data$OFFENSE %in%
  - c("ASSAULT W/DANGEROUS WEAPON", "HOMICIDE", "ROBBERY", "SEX ABUSE")
  - ~ "Person")

- Second, a cleaner crimes by hour graph:
  - dc.data %>% ggplot() +
  - geom_line(aes(x = xxxx), stat = "count", group = 1, color = "blue", size = 1) +
  - labs(title = "DC Crimes by Hour Reported \n2008 - 202x", x = "Hour Reported", y = "Number of Crimes (Thousands)", fill = "Type of Crime") +
  - theme(plot.title = element_text(hjust = 0.5, size = 14),
  - axis.text.x = element_text(color = "black", size = 10),
  - axis.text.y = element_text(color = "black", size = 10),
  - axis.ticks.y = element_blank(),
  - panel.grid.major.x = element_blank(),
  - panel.grid.major.y = element_line(color = "gray"),
  - panel.background = element_blank()) +
  - scale_y_continuous(limits = c(0, xxxxx), breaks = c(0, 5000,10000,15000,20000,25000,30000,35000), labels = c(0,5,10,15,20,25,30,35))

- Third, graph crime type by district:
  - dc.data %>% filter(DISTRICT != "NA") %>%
  - ggplot() +
  - geom_bar(aes(x = as.factor(xxx), fill = TYPE), stat = "count") +
  - labs(title = "Crimes by DC Police District \n2010 - 202x", x = "District", y = "Number of Crimes (Thousands)", fill = "Type of Crime") +
  - theme(plot.title = element_text(hjust = 0.5, size = 14),
  - axis.text.x = element_text(color = "black", size = 10),
  - axis.text.y = element_text(color = "black", size = 10),
  - axis.ticks.y = element_blank(),
  - panel.grid.major.x = element_blank(),
  - panel.grid.major.y = element_line(color = "gray"),
  - panel.background = element_blank()) +
  - scale_y_continuous(limits = c(0, 125000), breaks = c(0, 25000,50000,75000,100000,125000), labels = c(0,25,50,75,100,125))

- Fourth, graph a crime type as a percentage of weapon type/method:

```
dc.data %>% filter(TYPE == "xxx") %>%
  ggplot() +
  geom_bar(aes(x = OFFENSE, fill = METHOD), position = "fill") +
  labs(title = "DC xxx Crimes by Method \n2010 - 202x", x = "Crime", y = "Proportion of Crime", fill = "Method") +
  theme(plot.title = element_text(hjust = 0.5, size = 14),
        axis.text.x = element_text(color = "black", size = 10),
        axis.text.y = element_text(color = "black", size = 10),
        axis.ticks.y = element_blank(),
        panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_line(color = "gray"),
        panel.background = element_blank()) +
  scale_x_discrete(labels = c("Assault", "Homicide", "Robbery", "Sex Abuse")) +
  scale_y_continuous(labels = c("0%", "25%", "50%", "75%", "100%")) +
  scale_fill_manual(values = c("#1b9e77", "#d95f02","#7570b3"), labels = c("Gun", "Knife", "Other"))
```

HERE IS THE SCRIPT WE CREATED IN CLASS ON 29 JANUARY 2025

ANALYSIS

When conducting your analysis, consider this perspective:

1. State a finding
   a. This is the new thing that you've identified from the data
2. Provide your evidence
   a. This is typically a statistical reference
3. Add the context
   a. This is the integration of the stat(s) and the finding
   b. Compare your finding to the norm/average, or a similar measure in the data

FORMAT

Submit your script as a usable HTML file, via RMarkdown. You create this output as a .Rmd file in RStudio, but only submit the .html file.

Use this file as a template.

HERE IS THE TEMPLATE WE CREATED IN CLASS ON 29 JANUARY 2025

HERE IS THE TEMPLATE WE FURTHER REFINED IN CLASS ON 05 FEBRUARY 2025

SUBMISSION

Once your analysis is complete, please submit your project as either an HTML file, via Canvas.

GRADES

Analysis (4pts)

- Identify four unique analytic findings (*1pt each*)
  - Each finding should be 3-4 sentences, and describe something *specific* about the data.
  - *Each find should include a specific statistical reference*

Visuals (4pts)

- Include two visuals (*2pts each*)
  - These visuals should be thoughtful and logical (make sense to someone that has never seen them before); somehow, someway referenced in the text; and analytically meaningful (provide useful, relevant, and actionable insights)

Grammar (1pt)

- Error-free writing. No typos, run-on sentences, or sentence fragments. Proper punctuation. Real words.

File format (1pt)

- Project, including all analysis and visualizations, created in RMarkdown and submitted as an HTML file via Canvas. File is neatly and cleanly formatted, hiding unnecessary code as appropriate.

Please email me with any questions.