MDML Assignment 4

Trent Yu

Due: October 31, 2024. Total Points: 100.

Instructions

1. Fill in the `author` field at the top of this .Rmd file and indicate who you worked with, if anyone. Note that you must turn in your own work (see below).
2. Read the rest of these instructions and the "Grading" section.
3. Starting with the "Assignment" section, complete the rest of this R Markdown file from top to bottom. You will provide R code and some written responses in this Markdown file, and you will fill in the function outlines in the provided `assignment4A.R` and `assignment4B.R` scripts. Do **not** write any code outside of the functions included in `assignment4A.R` and `assignment4B.R`, and do **not** add additional arguments to any function in either script. Do **not** edit this Markdown file except as directed in the instructions. After you have filled in a function in `assignment4A.R` or `assignment4B.R`, you can run the corresponding code block in this Markdown file to progress through the assignment. When you finish the assignment, knit this file to obtain a pdf.
4. You must **only** use the packages loaded in this Markdown file in this assignment. Do not install or load any other packages.
5. I strongly suggest that you create and test your code in a "scratch" R script (don't turn that script in) before putting your answers in this file and in `assignment4A.R` and `assignment4B.R`. That is, you should consider this file and the two R scripts as the place to put your final answers for each question, not where you figure out how to do each question.
6. None of your functions should take more than a few minutes to run. That is, even if you believe your function answers a question correctly, you will lose points if we cannot run it in a reasonable amount of time (under five minutes); if this occurs, you should write more efficient code for your function.
7. Your submission for this assignment should consist only of these four files; do **not** change the file names:
    - `assignment4_workflow.Rmd`
    - `assignment4_workflow.pdf`
    - `assignment4A.R`
    - `assignment4B.R`

Tips

1. Start early! In the past, students have found assignments in this class to be quite time-consuming; there is a reason you are given two weeks for each one.
2. Running code on large datasets can take a long time. You can save time by sampling a small subset of rows to confirm your code works before running it on the whole dataset. Similarly, you may find it helpful to use `dplyr::slice_sample()` to select a random subset of points to plot if your plots are taking a long time to load.

Grading

Submit your assignment files on Brightspace. Assignments submitted after 12:30pm on the due date are considered late. Please see the syllabus for the late policy.

You may discuss this assignment with your professor, course assistant, and classmates, but you must turn in your own work. In particular, *you may not directly copy anyone else's code*; that would constitute plagiarism. You may **not** use ChatGPT or similar generative AI tools when doing your homework unless explicitly instructed to do so.

You will be graded on the following: how accurately you followed instructions; correctness, completeness, and clarity of your code and written answers; creativity (when applicable); and quality of visualizations (when applicable).

Also, note that passing a test (implemented by a test function in the R scripts) does **not** mean that you have done a problem correctly; **failing** a test means that the autograder will not be able to grade that part of your script.

Assignment

Setup

Create a folder on your computer for this assignment and put this R Markdown file, `assignment4A.R`, and `assignment4B.R` in that folder. If you are going to create a "scratch" R script to work in, put that in the folder as well. Make sure this folder is your working directory (either using the `setwd()` command or by going to Session –> Set Working Directory in the RStudio menu). Next, within this folder, create a subdirectory called `data`. Download the zipped `sqf_08_16.csv` file from Brightspace, unzip it, and move the unzipped file into the `data/` directory.

Part A: Model evaluation [60 points]

In this question, you will explore different ways to evaluate the performance of logistic regression models applied to the NYC Stop, Question, and Frisk dataset. First, run the code block below to load the tidyverse and ROCR packages, along with the dataset you will be working with.

setwd("/Users/trentyu/Desktop/Messy Data Machine Learning/Assignment 4") library(tidyverse) library(ROCR) all_sqf_data <- readr::read_csv('data/sqf_08_16.csv')

Question A1: Data cleaning [5 points]

In `assignment4A.R`, complete the `clean_sqf()` function, which takes a single argument, `data`; you will pass `all_sqf_data` to the `data` argument. Inside the function, restrict the data to only stops where the suspected crime is 'cpw', remove all stops in in Precinct 121, and `select` just the following variables:

- whether a weapon was found;
- precinct;
- whether the stop occurred in transit, housing, or neither;
- the ten additional stop circumstances (`additional.*`);
- the ten primary stop circumstances (`stopped.bc.*`);
- subject age, build, sex, height, weight;
- whether the stop occurred inside;

- length of observation period;
- day, month, year, and time of day (`time.period`).

Next, restrict to complete cases, so no row has any missing information, and convert time of day, and precinct into factors; `clean_sqf()` should return this cleaned tibble.

After completing `clean_sqf()`, uncomment and run the code block below, which saves the output of the function as a tibble called `sqf_data`.
(Note: `clean_sqf()` here is similar, although **not** identical, to the corresponding function in Question B1 of HW 3; you may adapt your code from that question in this assignment.)

source('assignment4A.R') sqf_data <- clean_sqf(data = all_sqf_data) test_clean_sqf()

Question A2: Data splitting [5 points]

In `assignment4A.R`, complete the `split_sqf_data()` function, which takes a single argument, `data`; you will pass `sqf_data` to the `data` argument. Inside the function, restrict the data to only include years between 2013-2015 (inclusive), split the data into `sqf_pre_2015` (data from 2013-2014) and `sqf_2015` (data from 2015), respectively, and remove the `year` column from both `sqf_pre_2015` and `sqf_2015`. Then, randomly shuffle the rows in `sqf_pre_2015` and split it in half, calling the two halves `sqf_pre_train` and `sqf_pre_test`.

The function `split_sqf_data()` should return a named list with three elements:

- `sqf_pre_train`: a tibble of a random half of stops between 2013 and 2014
- `sqf_pre_test`: a tibble with the other half of the stops from 2013 and 2014
- `sqf_2015`: a tibble of stops from 2015.

After completing `split_sqf_data()`, uncomment and run the code block below.

source('assignment4A.R') sqf_split <- split_sqf_data(data = sqf_data) test_split_sqf_data()

Question A3: Modeling [15 points]

In `assignment4A.R`, complete the `logistic_model_1()` function, which takes a single argument, `data`; you will pass the list `sqf_split` to the `data` argument (do not edit the `seed` argument). Inside the function, fit a logistic regression on `sqf_pre_train`, using all features to predict whether a weapon was found or not. Use the `ROCR` package to compute the AUC of this model's predictions on `sqf_pre_test` and on `sqf_2015` (your AUCs should be numbers between 0 and 1).

The function `logistic_model_1()` should return a named list with two elements:

- `auc_pre_test`: a vector with one element, the AUC for predictions on `sqf_pre_test`
- `auc_2015`: a vector with one element, the AUC for predictions on `sqf_2015`

After completing `logistic_model_1()`, uncomment and run the code block below.

source('assignment4A.R') auc_list <- logistic_model_1(data = sqf_split) auc_list test_logistic_model_1()

Question A4: Thinking about AUC [10 points]

5. Write a paragraph answering the following questions:

   i. Why is the AUC on `sqf_pre_test` noticeably higher than the AUC on `sqf_2015`?

   There are three big reasons why the AUC on `sqf_pre_test` noticeably higher than the AUC on `sqf_2015`. Temporal shift can be the first reason. The data from 2015 might have underlying trends that are different from data in 2013 and 2014. Changes in policy, or other social factors could have altered the way stops were conducted or how weapons were found, making the 2015 data different from the data from 2013 and 2014. Second,the model may have overfitting. It could perform better on sqf_pre_test because this data comes from the same time period as the training data(sqf_pre_train). It suggests the model is more finely tuned to patterns from 2013-2014. Third, data distribution might be one of the reasons. The distribution of features or the target variable (found.weapon) may vary between the two datasets, and if the 2015 data has a different distribution of key variables, the model trained on 2013–2014 might not generalize as well to this new data.

   ii. If you were going to use this model to make predictions about stops in 2015 or later, would the AUC on `sqf_pre_test` or `sqf_2015` be a better estimate of performance on unseen data?

   The AUC on sqf_2015 would provide a better estimate of the model's performance on unseen data from 2015 or later because it represents a more recent time period, offering a more accurate reflection of how the model might perform on future data. However, the AUC on sqf_pre_test could be artificially high, as this test set comes from the same time period (2013–2014) as the training data. This potentially lead to over-optimistic performance estimates.

   iii. More generally, when evaluating a model's performance using a simple training/testing split, should you always make the split by shuffling all of your data and splitting randomly?

   For time-dependent data (ex. stop-and-frisk data where policies or external factors may change over time), random shuffling should be avoided to prevent data leakage. However, splitting based on time (training on earlier years and testing on later ones) more accurately simulates real-world conditions. As a result, even though random shuffling works for many datasets, a time-based split is better for time-dependent data to ensure realistic model evaluation.

Question A5: More modeling [15 points]

In `assignment4A.R`, complete the `logistic_model_2()` function, which takes a single argument, `data`; you will pass `sqf_data` to the `data` argument (do not edit the `seed` argument). Inside the function, fit a logistic regression model just on stops from 2008, predicting `found.weapon` as a function of all other predictors except `year`. Then, for each year after 2008, subset the data to stops from that year, use your model to estimate predicted probabilities, and calculate the corresponding AUC (using the ROCR package).

The function `logistic_model_2()` should return a tibble with two columns, `year` and `auc` (there should be 8 rows, with `year` going from 2009 to 2016, and `auc` indicating the AUC for predictions that year).

After completing `logistic_model_2()`, uncomment and run the code block below.

source('assignment4A.R') predictions <- logistic_model_2(data = sqf_data) test_logistic_model_2()

Question A6: Visualizing [10 points]

Using the `predictions` tibble created above, plot AUC scores on the y-axis and the corresponding year on the x-axis. Include your code for the plot where indicated, and below, include the plot itself along with a few sentences describing the pattern you see in the plot, and why that pattern might occur.

```r
setwd("/Users/trentyu/Desktop/Messy Data Machine Learning/Assignment 4")
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(ROCR)
all_sqf_data <- readr::read_csv('data/sqf_08_16.csv')
```

```
## Rows: 3197575 Columns: 86
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (12): city, sector, day, month, location.housing, suspected.crime, iden...
## dbl  (13): year, xcoord, ycoord, time.period, precinct, lat, lon, observatio...
## lgl  (60): frisked, searched, inside, officer.uniform, reason.explained, oth...
## dttm  (1): timestamp
##
## 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.
```

```r
setwd("/Users/trentyu/Desktop/Messy Data Machine Learning/Assignment 4/data")
source('assignment4A.R')
sqf_data <- clean_sqf(data = all_sqf_data)
test_clean_sqf()
```

```
## Test passed
## Test passed
## Test passed
```

```r
source('assignment4A.R')
sqf_split <- split_sqf_data(data = sqf_data)
```

```
## [1] "Filtered sqf_pre_2015:"
## # A tibble: 6 x 34
##   found.weapon precinct location.housing subject.age subject.build subject.sex
##   <lgl>        <fct>    <chr>                  <dbl> <chr>         <chr>
## 1 FALSE        7        housing                   23 thin          male
## 2 FALSE        9        neither                   32 medium        male
## 3 TRUE         13       transit                   37 thin          male
## 4 TRUE         14       transit                   23 thin          male
## 5 FALSE        23       housing                   15 medium        male
## 6 FALSE        23       housing                   18 thin          male
## # i 28 more variables: subject.height <dbl>, subject.weight <dbl>,
```

```
## #   inside <lgl>, observation.period <dbl>, day <chr>, month <chr>, year <dbl>,
## #   time.period <fct>, additional.associating <lgl>,
## #   additional.direction <lgl>, additional.highcrime <lgl>,
## #   additional.time <lgl>, additional.sights <lgl>, additional.other <lgl>,
## #   additional.evasive <lgl>, additional.proximity <lgl>,
## #   additional.report <lgl>, additional.investigation <lgl>, ...
## [1] "Filtered sqf_2015:"
## # A tibble: 6 x 34
##   found.weapon precinct location.housing subject.age subject.build subject.sex
##   <lgl>        <fct>    <chr>                  <dbl> <chr>         <chr>
## 1 FALSE        22       neither                   14 thin          male
## 2 FALSE        67       neither                   25 thin          male
## 3 FALSE        114      neither                   23 thin          male
## 4 FALSE        33       neither                   19 thin          male
## 5 FALSE        40       housing                   20 muscular      male
## 6 FALSE        20       neither                   35 medium        male
## # i 28 more variables: subject.height <dbl>, subject.weight <dbl>,
## #   inside <lgl>, observation.period <dbl>, day <chr>, month <chr>, year <dbl>,
## #   time.period <fct>, additional.associating <lgl>,
## #   additional.direction <lgl>, additional.highcrime <lgl>,
## #   additional.time <lgl>, additional.sights <lgl>, additional.other <lgl>,
## #   additional.evasive <lgl>, additional.proximity <lgl>,
## #   additional.report <lgl>, additional.investigation <lgl>, ...
## [1] "Number of rows in sqf_pre_2015_shuffled: 57984"
```

```
test_split_sqf_data()
```

```
## Test passed
## Test passed
## Test passed
## Test passed
```

```
source('assignment4A.R')
auc_list <- logistic_model_1(data = sqf_split)
auc_list
```

```
## $auc_pre_test
## [1] 0.8102723
##
## $auc_2015
## [1] 0.7527764
```

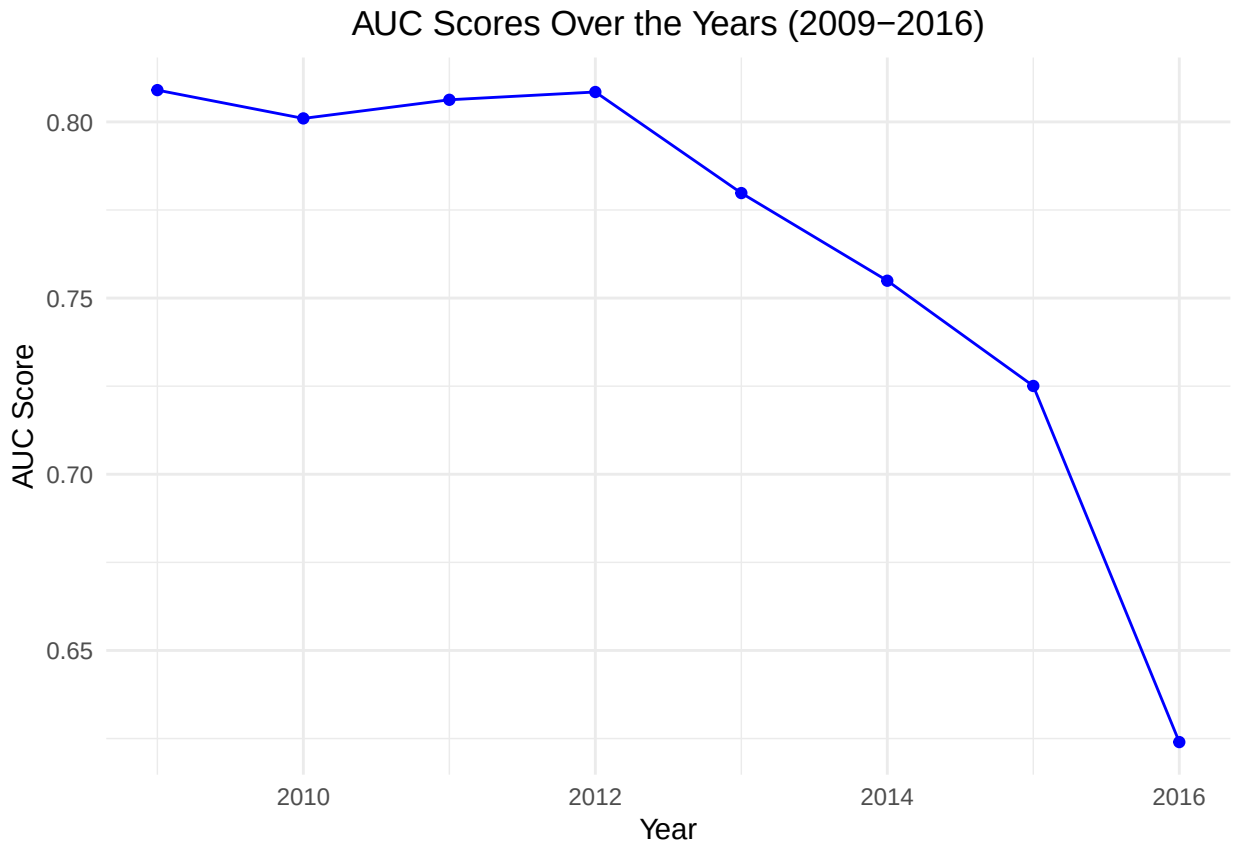
```
test_logistic_model_1()
```

```
## Test passed
## Test passed
## Test passed
## Test passed
```

```
source('assignment4A.R')
predictions <- logistic_model_2(data = sqf_data)
test_logistic_model_2()
```

```
## Test passed
## Test passed
## Test passed
```

```r
library(ggplot2)
library(dplyr)

ggplot(predictions, aes(x = year, y = auc)) +
  geom_line(color = "blue") +
  geom_point(color = "blue") +
  labs(title = "AUC Scores Over the Years (2009-2016)",
       x = "Year",
       y = "AUC Score") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



PLOT AND WRITTEN EXPLANATION GOES HERE. This decline in AUC over time could be due to changes in external factors affecting the data,which the model (trained on 2008 data) was not able to account for. As the years progress, these changes may have altered the patterns and relationships in the data which leads to a reduced predictive power of the model for later years. This shifts causes the model to generalize poorly to the later year.

Part B: Scraping Boston crime data. [40 points]

In this problem, you will collect and examine Boston crime data over the last two years from a community news and information website. You will scrape data about reported incidents (crime type, neighborhood, and time of day) from this site. Doing this problem does *not* involve reading the text of reported articles or comments on articles, but please be aware that some of the headlines may be disturbing. To provide some context, data scraped from this website (and other similar sites) were used as part of a large-scale research project to supplement sources of official crime statistics (which are often flawed) with the goal of compiling and publicly releasing accurate crime statistics for the use of the public, researchers, journalists, advocacy organizations, and public officials.

Question B1: Acquire and clean the data [30 points]

In `assignment4B.R`, complete the `scrape_data()` function, which has no arguments. Inside the function, scrape the *crime type* and *hour* from all 20 of the "Crime by neighborhood" pages in the drop down menu on Universal Hub (e.g., https://www.universalhub.com/crime/allston.html). Make sure to get **all** data from all neighborhoods; some of the neighborhoods may have multiple pages of data or non-standard urls. We suggest you use a `foreach` loop (as in HW 2 Part A) to loop over urls and combine data from different neighborhoods. You should correct the following issues with the crime type field (do not correct any other possible issues with that field):

- Empty crime types (i.e., " ") should be encoded as `NA`;
- "Stabbin" should be encoded as "Stabbing";
- "Assault with a dangeous weapon" should be encoded as "Assault with a dangerous weapon"; and
- "Assault and battery with a dangeous weapon" should be encoded as "Assault and battery with a dangerous weapon".

The `scrape_data()` function should return a tibble that contains all crimes in all neighborhoods (i.e., each row represents a crime) and has three columns:

1. `crime`: the name of the crime, from the `Type` field on each page. Make sure to get rid of extra whitespace and `"\\n"` in the crime type names.
2. `hour`: the hour as an integer from 0 to 23, from the `Date` field. You might find `lubridate::parse_date_time()` useful.
3. `nbhd`: the complete neighborhood name as a lowercase string, with a dash instead of a space for neighborhood names comprising several words (e.g., `back-bay`).

After completing `scrape_data()`, uncomment and run the code block below.

library(foreach) library(rvest) source('assignment4B.R') crime_data <- scrape_data() test_scrape_data()

Question B2: Visualize [10 points]

Using the `crime_data` tibble created above, make a plot visualizing the total number of crimes in Boston (aggregated across neighborhoods and crime types) on the y-axis (use `geom_line()` in ggplot) and the hour of day on the x-axis. Include your code for the plot where indicated, as well as the plot itself and a few sentences describing the pattern you observe.

```
# plot code here
library(foreach)
```

```
##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
library(rvest)
```

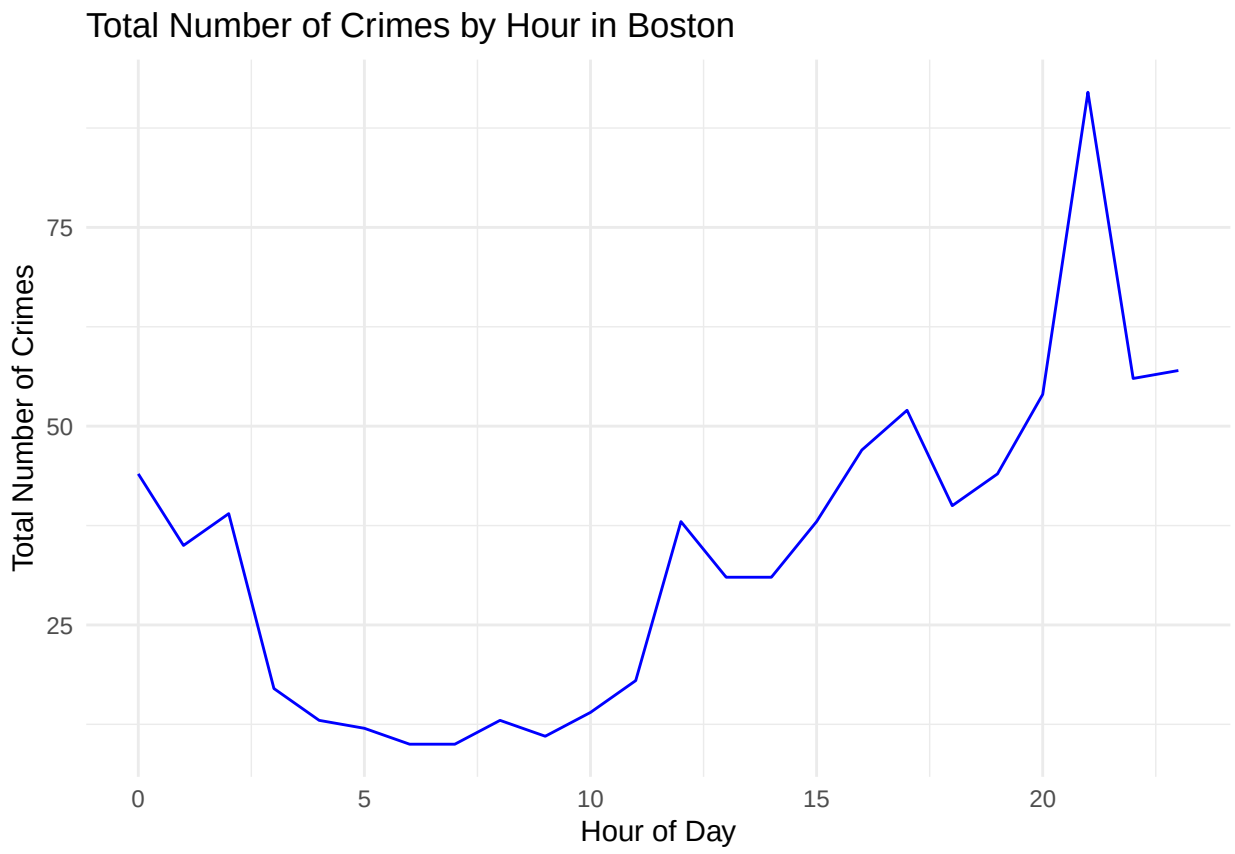
```
##
## Attaching package: 'rvest'

## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
source('assignment4B.R')
crime_data <- scrape_data()
```

```
## Scraping page 1 of https://www.universalhub.com/crime/allston.html

## New names:
## Scraping page 1 of https://www.universalhub.com/crime/back-bay.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/beacon-hill.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/brighton.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/charlestown.html
## New names:
## Scraping page 1 of https://www.universalhub.com/newcrime/9967
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/dorchester.html
## Scraping page 2 of https://www.universalhub.com/crime/dorchester.html?page=1
## New names:
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/downtown.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/east-boston.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/fenway.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/hyde-park.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/jamaica-plain.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/mattapan.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/mission-hill
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/north-end.html
## New names:
```

```
## Scraping page 1 of https://www.universalhub.com/crime/roslindale.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/roxbury.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/south-boston.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/south-end.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/west-roxbury.html
## New names:
## * `` -> `...2`
```

```r
library(ggplot2)
library(dplyr)

crime_by_hour <- crime_data %>%
  group_by(hour) %>%
  summarise(total_crimes = n())

ggplot(crime_by_hour, aes(x = hour, y = total_crimes)) +
  geom_line(color = "blue") +
  labs(title = "Total Number of Crimes by Hour in Boston",
       x = "Hour of Day",
       y = "Total Number of Crimes") +
  theme_minimal()
```



PLOT AND WRITTEN EXPLANATION GOES HERE. he graph shows a trend in the total number of crimes by hour of the day in Boston. Crime rates are generally lower during early morning hours and the daytime, with a noticeable rise as evening approaches. The peak in crime incidents occurs around 21:00 (9 PM), which may indicate an increase in criminal activity during nighttime hours when visibility is lower and fewer people are likely around to witness incidents. This pattern suggests that criminal activity may be more frequent in the evening due to reduced oversight and fewer potential witnesses compared to daytime hours

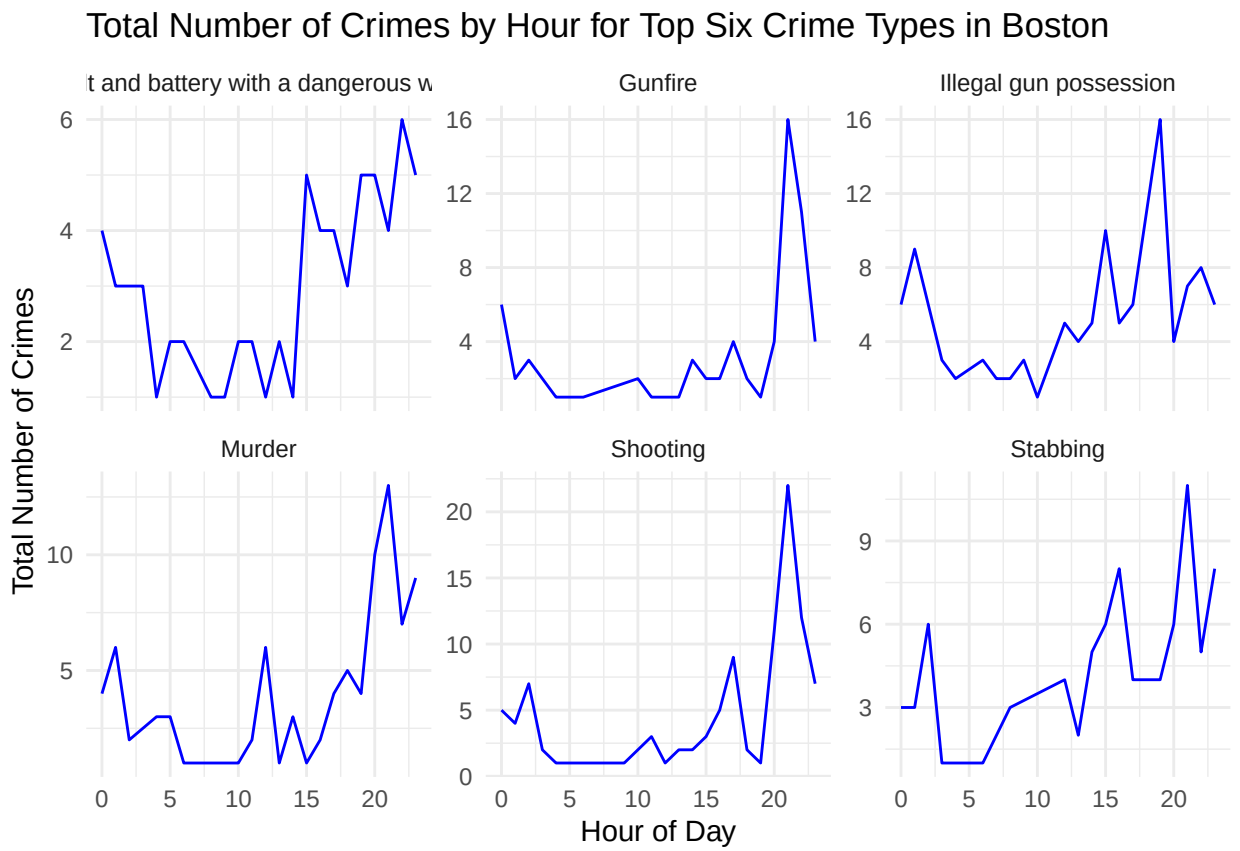
Next, using the `crime_data` tibble, for each of the six most common types of crime in Boston, separately plot the total number of crimes (aggregated across neighborhoods) on the y-axis (use `geom_line()` in ggplot) and the hour of day on the x-axis using ggplot's `facet_wrap()` command; your plot should have six facets (subplots), one for each crime type. Include your code for the plot where indicated, as well as the plot itself and a few sentences describing the pattern you observe.

```
library(foreach)
library(rvest)
source('assignment4B.R')
crime_data <- scrape_data()
```

```
## Scraping page 1 of https://www.universalhub.com/crime/allston.html

## New names:
## Scraping page 1 of https://www.universalhub.com/crime/back-bay.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/beacon-hill.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/brighton.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/charlestown.html
## New names:
## Scraping page 1 of https://www.universalhub.com/newcrime/9967
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/dorchester.html
## Scraping page 2 of https://www.universalhub.com/crime/dorchester.html?page=1
## New names:
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/downtown.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/east-boston.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/fenway.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/hyde-park.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/jamaica-plain.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/mattapan.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/mission-hill
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/north-end.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/roslindale.html
## New names:
```

```
## Scraping page 1 of https://www.universalhub.com/crime/roxbury.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/south-boston.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/south-end.html
## New names:
## Scraping page 1 of https://www.universalhub.com/crime/west-roxbury.html
## New names:
## * `` -> `...2`
```

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)

top_crimes <- crime_data %>%
  count(crime, sort = TRUE) %>%
  top_n(6, n) %>%
  pull(crime)

top_crime_data <- crime_data %>%
  filter(crime %in% top_crimes)

crime_by_hour_type <- top_crime_data %>%
  group_by(hour, crime) %>%
  summarise(total_crimes = n(), .groups = "drop")

ggplot(crime_by_hour_type, aes(x = hour, y = total_crimes)) +
  geom_line(color = "blue") +
  facet_wrap(~ crime, scales = "free_y") +
  labs(title = "Total Number of Crimes by Hour for Top Six Crime Types in Boston",
       x = "Hour of Day",
       y = "Total Number of Crimes") +
  theme_minimal()
```



PLOT AND WRITTEN EXPLANATION GOES HERE. While each crime type has slight variations, they all generally follow the broader pattern of increased occurrences in the late evening, particularly around 21:00 (9 PM). This suggests that factors associated with nighttime, such as reduced visibility and fewer bystanders, may contribute to a rise in crime across various types.