**Table of Contents**

[**Python Part** 3](#_Toc183447014)

[**Q1: Implementation of the transport network** 3](#_Toc183447015)

[**Plot:** 4](#_Toc183447016)

[**Q2: Visualizing the network** 4](#_Toc183447017)

[**Plot:** 5](#_Toc183447018)

[**Q4: Displaying attributes for lines** 6](#_Toc183447019)

[**Plot** 7](#_Toc183447020)

[**Q5: Improvements:** 7](#_Toc183447021)

[**Data Source:** 7](#_Toc183447022)

[**R Part** 8](#_Toc183447023)

[**Question # 1** 8](#_Toc183447024)

[**Data pre-processing and descriptive statistics** 8](#_Toc183447025)

[**1.** **Creating two data frames:** 8](#_Toc183447026)

[**2.** **Statistical distribution of these datasets:** 9](#_Toc183447027)

[**Plot:** 9](#_Toc183447028)

[**Plot:** 10](#_Toc183447029)

[**3.** **Description of the workflow and justification of plotting choice:** 10](#_Toc183447030)

[**Workflow description:** 10](#_Toc183447031)

[**Justification:** 11](#_Toc183447032)

[**Question # 2** 11](#_Toc183447033)

[**Data transformation and visualization using ggplot2** 11](#_Toc183447034)

[**1.** **Code for creating a new data frame using package dplyr.** 11](#_Toc183447035)

[**2.** **Which region(s) has the highest total count of offences as well as offence rate per 1000 populations?** 12](#_Toc183447036)

[**Highest total count of offences:** 12](#_Toc183447037)

[**Plot:** 13](#_Toc183447038)

[**Justification:** 13](#_Toc183447039)

[**Offence rate per 1000:** 14](#_Toc183447040)

[**Plot:** 14](#_Toc183447041)

[**Justification:** 14](#_Toc183447042)

[**3.** **What are the top three counties that have the lowest total count of offences?** 14](#_Toc183447043)

[**Plot:** 15](#_Toc183447044)

[**Justification:** 16](#_Toc183447045)

# **Python Part**

# **Q1: Implementation of the transport network**

import networkx as netx

import matplotlib.pyplot as plot

# Create a graph

G = netx.Graph()

# Define updated lines and their stations for New York City

lines = {

    "1": ["South Ferry", "Chambers St", "14 St", "Times Sq-42 St", "59 St-Columbus Circle"],

    "2": ["Flatbush Av-Brooklyn College", "Atlantic Av-Barclays Center", "14 St", "Times Sq-42 St", "96 St"],

    "3": ["New Lots Av", "Atlantic Av-Barclays Center", "14 St", "Times Sq-42 St", "96 St"],

    "4": ["Crown Hts-Utica Av", "Atlantic Av-Barclays Center", "14 St-Union Sq", "Grand Central-42 St", "125 St"],

    "5": ["Flatbush Av-Brooklyn College", "Atlantic Av-Barclays Center", "14 St-Union Sq", "Grand Central-42 St", "125 St"],

    "6": ["Brooklyn Bridge-City Hall", "14 St-Union Sq", "Grand Central-42 St", "59 St", "125 St"],

}

# Add edges and nodes for each line

for line, stations in lines.items():

    for i in range(len(stations) - 1):

        G.add\_edge(stations[i], stations[i + 1], line=line)

# Identify transfer stations (nodes shared by multiple lines)

transfer\_stations = {station for station in G.nodes if sum(station in line\_stations for line\_stations in lines.values()) > 1}

# Adding additional transfer connections for graph connectivity

additional\_transfers = [

    ("14 St", "14 St-Union Sq"),

]

for station1, station2 in additional\_transfers:

    G.add\_edge(station1, station2, line="Transfer")

# Check if graph is connected

is\_connected = netx.is\_connected(G)

print("Graph is connected:", is\_connected)

# Plot for question 1

plot.figure(figsize=(12, 12))

pos = netx.spring\_layout(G, seed=42)  # Set seed for consistent layout

netx.draw(G, pos, with\_labels=True, node\_size=300, node\_color="lightgray")

plot.show()

# **Plot:**

A network of cities with lines and dots

Description automatically generated with medium confidence

# **Q2: Visualizing the network**

# Updated color map for new lines

color\_map = {

    "1": "blue",

    "2": "green",

    "3": "red",

    "4": "orange",

    "5": "yellow",

    "6": "gray",

}

# Draw network

plot.figure(figsize=(12, 12))

pos = netx.spring\_layout(G, seed=42)  # Set seed for consistent layout

# Draw each line with colors

for line, color in color\_map.items():

    edges = [(u, v) for u, v, d in G.edges(data=True) if d["line"] == line]

    netx.draw\_networkx\_edges(G, pos, edgelist=edges, edge\_color=color, width=2, label=line)

# Draw transfer stations in a larger size and different color

netx.draw\_networkx\_nodes(G, pos, nodelist=transfer\_stations, node\_color="lightblue", node\_size=300, label="Transfer Stations")

# Draw other stations

netx.draw\_networkx\_nodes(G, pos, nodelist=set(G.nodes) - transfer\_stations, node\_color="lightgray", node\_size=100)

# Draw labels

netx.draw\_networkx\_labels(G, pos, font\_size=8)

# Plot for question 2

plot.figure(figsize=(12, 12))

pos = netx.spring\_layout(G, seed=42)  # Set seed for consistent layout

netx.draw(G, pos, with\_labels=True, node\_size=300, node\_color="lightgray")

plot.show()

# **Plot:**

A map of a network

Description automatically generated with medium confidence

# **Q4: Displaying attributes for lines**

# Set edge weights (distances in meters) for all pairs

distances = {

    ("South Ferry", "Chambers St"): 2000,

    ("Chambers St", "14 St"): 3000,

    ("14 St", "Times Sq-42 St"): 2000,

    ("Times Sq-42 St", "59 St-Columbus Circle"): 1500,

    ("Flatbush Av-Brooklyn College", "Atlantic Av-Barclays Center"): 5000,

    ("Atlantic Av-Barclays Center", "14 St"): 4000,

    ("14 St-Union Sq", "Grand Central-42 St"): 2000,

    ("Grand Central-42 St", "125 St"): 6000,

    ("Brooklyn Bridge-City Hall", "14 St-Union Sq"): 3000,

    ("Inwood-207 St", "59 St-Columbus Circle"): 8000,

    ("59 St-Columbus Circle", "125 St"): 7000,

    ("Flatbush Av-Brooklyn College", "Atlantic Av-Barclays Center"): 5000,

    ("Atlantic Av-Barclays Center", "14 St"): 4000,

    ("14 St", "Times Sq-42 St"): 2000,

    ("Times Sq-42 St", "96 St"): 5000,

    ("New Lots Av", "Atlantic Av-Barclays Center"): 6000,

    ("Crown Hts-Utica Av", "Atlantic Av-Barclays Center"): 6000,

    ("125 St", "59 St"): 7000,

    ("Grand Central-42 St", "59 St"): 1500,

    ("14 St-Union Sq", "Atlantic Av-Barclays Center"): 4000,

}

# Update graph with distances

for (u, v), distance in distances.items():

    if G.has\_edge(u, v):

        G[u][v]['distance'] = distance

# Draw edges with labels (distances)

edge\_labels = {(u, v): f"{d['distance']}m" for u, v, d in G.edges(data=True) if 'distance' in d}

netx.draw\_networkx\_edge\_labels(G, pos, edge\_labels=edge\_labels, font\_size=8)

# Add legend and title

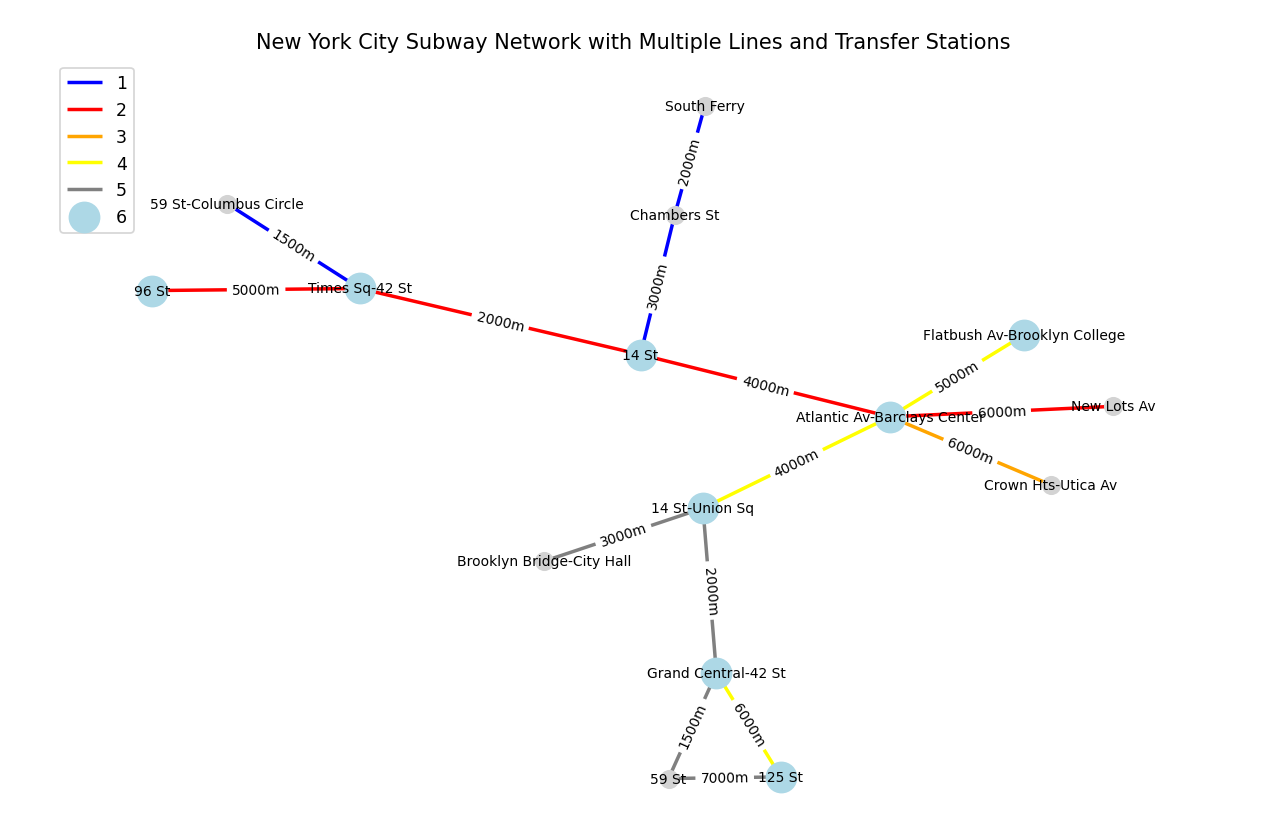
plot.legend(color\_map.keys(), loc='upper left', fontsize=10)

plot.title("New York City Subway Network with Multiple Lines and Transfer Stations")

plot.axis("off")

plot.show()

# **Plot**



# **Q5: Improvements:**

Based on the generated map, here are two suggested improvements:

* Implementing a way to show directions and align each station relative to one another by direction. This will make the map useable by the intended users in real life situations.
* Showing the distance relative to one another based on the length of each edge. If the distance is 3000m the length of the edge should be smaller than the one with 5000m. This will improve the readers sense of distance.

# **Data Source:**

The data used in this plot is taken from the MTA Live website, which provides real-time information about the New York City subway system.

# **R Part**

# **Question # 1**

# **Data pre-processing and descriptive statistics**

# **Creating two data frames:**

To create two data frames using dplyr, we first loaded the required libraries and the dataset, then filter the data to create subsets for “Banking and credit industry fraud” and “All charity fraud”.

# Import required libraries

library(readxl)

library(dplyr)

library(ggplot2)

library(tidyr)

library(scales)

# Load the specified sheet from the Excel file, skipping the initial 8 rows

fraud\_dataset <- read\_excel("D:/Code/HF/24-11-2024/cw\_r.xlsx", sheet = "Table 3d", skip = 8)

# Examine the structure of the loaded data for understanding its format

str(fraud\_dataset)

# Update column names for better readability and usage

colnames(fraud\_dataset) <- c("Fraud\_Type", "Year\_2012\_2013", "Year\_2013\_2014", "Year\_2014\_2015", "Year\_2015\_2016",

                             "Year\_2016\_2017", "Year\_2017\_2018", "Year\_2018\_2019", "Year\_2019\_2020", "Year\_2020\_2021",

                             "Year\_2021\_2022", "Percent\_Change")

# Create subsets of data for different types of fraud

# Extract data related to fraud in the banking and credit industry

banking\_fraud\_data <- fraud\_dataset %>%

  filter(Fraud\_Type == "Banking and credit industry fraud") %>%

  select(-Fraud\_Type, -Percent\_Change) %>%

  pivot\_longer(cols = starts\_with("Year"), names\_to = "Year", values\_to = "Fraud\_Count")

# Extract data related to fraud in charities

charity\_fraud\_data <- fraud\_dataset %>%

  filter(Fraud\_Type == "All charity fraud") %>%

  select(-Fraud\_Type, -Percent\_Change) %>%

  pivot\_longer(cols = starts\_with("Year"), names\_to = "Year", values\_to = "Fraud\_Count")

# Check the structure of the subsets for accuracy

str(banking\_fraud\_data)

str(charity\_fraud\_data)

# Preview the first few entries in each subset

head(banking\_fraud\_data)

head(charity\_fraud\_data)

# Arrange data by year to ensure proper order for visualization

banking\_fraud\_data <- banking\_fraud\_data[order(banking\_fraud\_data$Year), ]

charity\_fraud\_data <- charity\_fraud\_data[order(charity\_fraud\_data$Year), ]

# **Statistical distribution of these datasets:**

To visualize the dispersion and central tendency of the two data frames, ggplot2 was used to create bar charts.

# Visualize yearly counts of charity fraud using a bar chart with theme\_bw

ggplot(charity\_fraud\_data, aes(x = Year, y = Fraud\_Count, fill = Year)) +

  geom\_bar(stat = "identity") +

  labs(title = "Yearly Charity Fraud Counts",

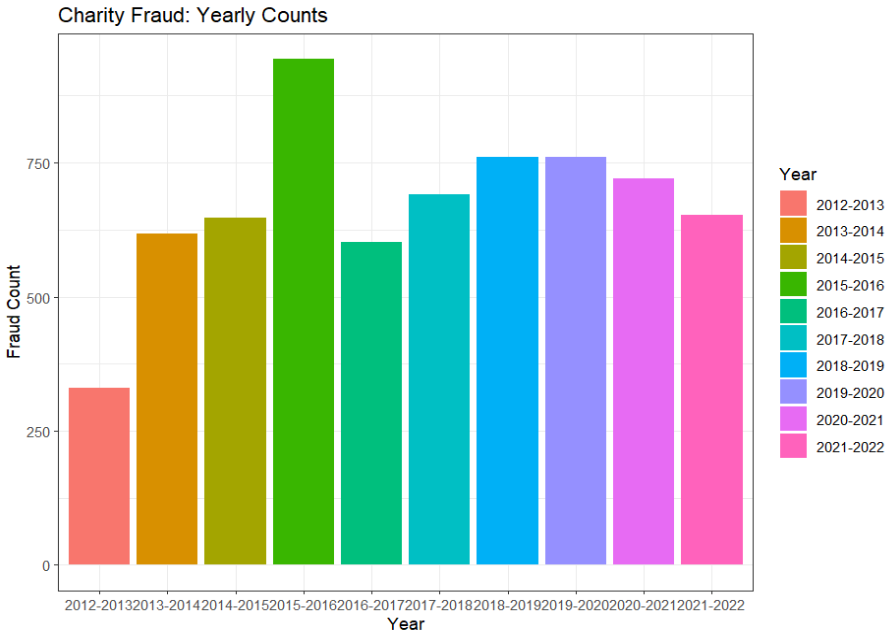
       x = "Year",

       y = "Number of Cases") +

  scale\_y\_continuous(labels = comma) +

  theme\_bw()

# **Plot:**



# Visualize yearly counts of banking fraud using a bar chart with theme\_bw

ggplot(banking\_fraud\_data, aes(x = Year, y = Fraud\_Count, fill = Year)) +

  geom\_bar(stat = "identity") +

  labs(title = "Yearly Banking Fraud Counts",

       x = "Year",

       y = "Number of Cases") +

  scale\_y\_continuous(labels = comma) +

  theme\_bw()

# **Plot:**

A graph of colorful bars

Description automatically generated

# **Description of the workflow and justification of plotting choice:**

# **Workflow description:**

1. **Data loading and preparation**

First loaded the dataset from the specified excel sheet. Then examined the structure of the dataset to understand its format and updated column names for easier understanding.

1. **Data subsetting**

Secondly created subsets of the data for “Banking and credit industry fraud” and “All charity fraud”.

1. **Data examination**

Then used str() and head() to examine the structure and preview the first few rows of the subsets.

1. **Data visualization**

Used ggplot2 to create bar charts for visualizing the yearly counts of frauds.

# **Justification:**

Bar chart is chosen for its effectiveness in comparing quantities across different categories. And ggplot2 package is used for its flexibility and powerful optimization options.

# **Question # 2**

# **Data transformation and visualization using ggplot2**

# **Code for creating a new data frame using package dplyr.**

# Import necessary libraries

library(readxl)

library(ggplot2)

library(dplyr)

library(tidyr)

# Load the dataset from the "Table 5" sheet, skipping the first 9 rows

crime\_records <- read\_excel("D:/Code/HF/24-11-2024/cw\_r.xlsx", sheet = "Table 5", skip = 9)

# Examine the structure of the dataset to understand the column names and types and rename the columns

str(crime\_records)

colnames(crime\_records) <- c("Region\_Code", "Region\_Name", "Offence\_Count", "Rate\_Per\_1000", "Change\_Percentage")

# Convert the "Rate\_Per\_1000" column to numeric to enable proper calculations

crime\_records$Rate\_Per\_1000 <- as.numeric(crime\_records$Rate\_Per\_1000)

# Filter out rows where "Offence\_Count" or "Rate\_Per\_1000" have missing values

crime\_records <- crime\_records %>%

  filter(!is.na(Offence\_Count) & !is.na(Rate\_Per\_1000))

# **Which region(s) has the highest total count of offences as well as offence rate per 1000 populations?**

# **Highest total count of offences:**

# Aggregate the data by region to calculate total offences and average rate per 1000 population

regional\_summary <- crime\_records %>%

  group\_by(Region\_Name) %>%

  summarize(

    total\_offences = sum(Offence\_Count, na.rm = TRUE),

    avg\_rate\_per\_1000 = mean(Rate\_Per\_1000, na.rm = TRUE)

  )

# Display the aggregated regional data for verification

print(regional\_summary)

# Create a lollipop plot showing total offences by region, using a classic theme

ggplot(regional\_summary, aes(x = reorder(Region\_Name, total\_offences), y = total\_offences)) +

  geom\_segment(aes(xend = Region\_Name, yend = 0), color = "black") +

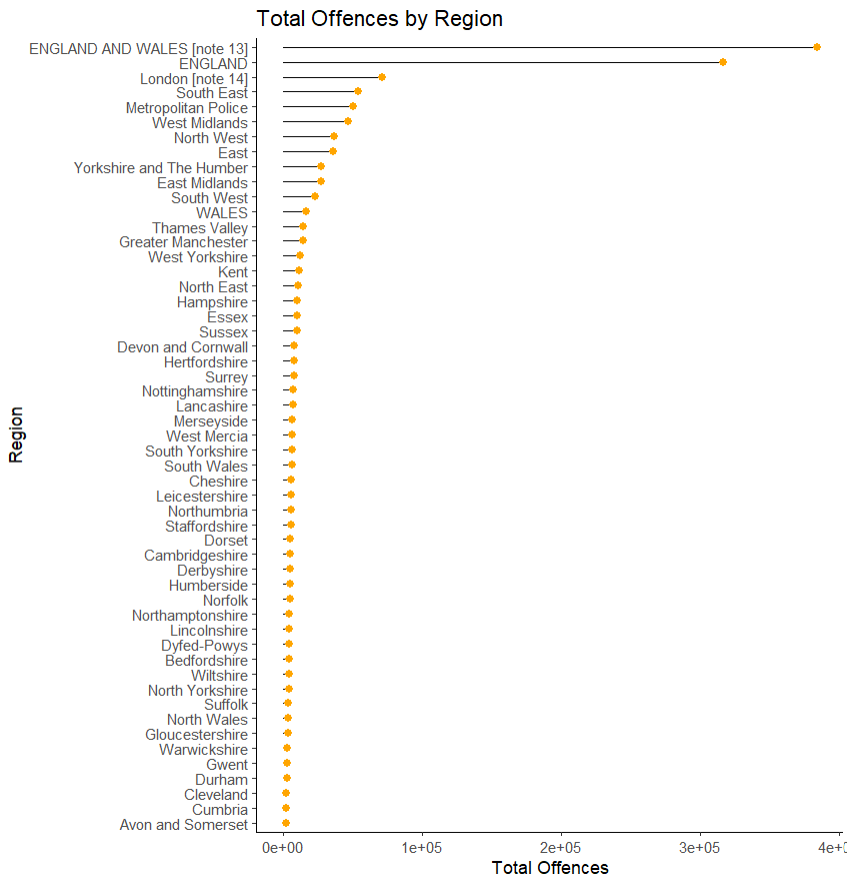
  geom\_point(color = "orange", size = 2) +

  coord\_flip() +

  labs(title = "Total Offences by Region", x = "Region", y = "Total Offences") +

  theme\_classic()

# **Plot:**



# **Justification:**

The lollipop plot is chosen for its clarity in displaying the total offences by region. The horizontal lines and points makes it easy to compare the values across the different regions.

# **Offence rate per 1000:**

# Generate a scatter plot to visualize the rate per 1,000 population by region

ggplot(regional\_summary, aes(x = reorder(Region\_Name, avg\_rate\_per\_1000), y = avg\_rate\_per\_1000)) +

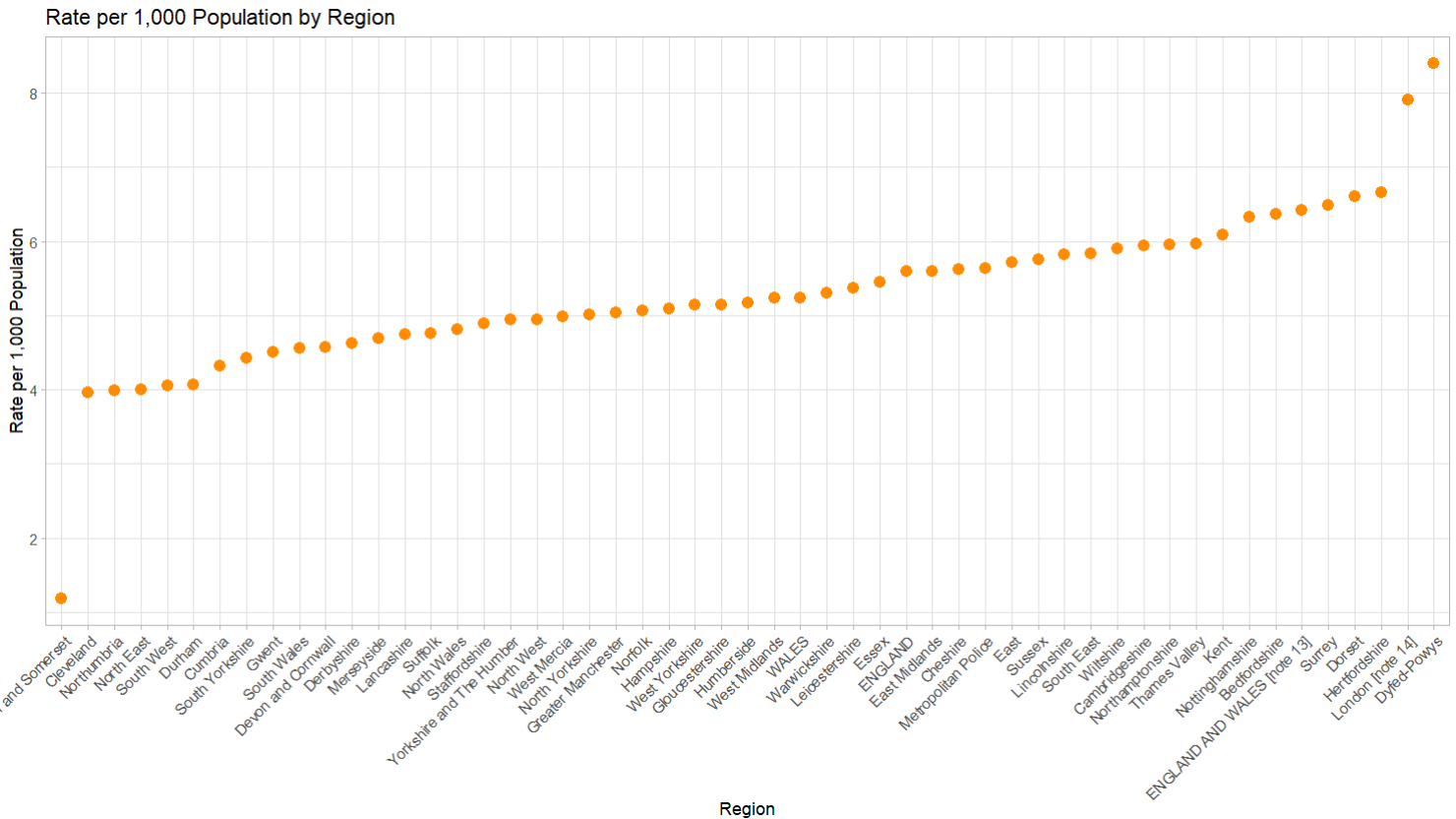
  geom\_point(color = "darkorange", size = 3) +

  labs(title = "Rate per 1,000 Population by Region", x = "Region", y = "Rate per 1,000 Population") +

  theme\_light() +

  theme(axis.text.x = element\_text(angle = 45, hjust = 1))

# **Plot:**



# **Justification:**

The scatter plot it used to visualize the rate per 1000 population by region because it effectively shows the distribution and allows for easy comparison of rates across regions. The use of orange color and size enhances the visual appeal of the plot.

# **What are the top three counties that have the lowest total count of offences?**

# Identify the three counties with the lowest total crimes

low\_offence\_regions <- regional\_summary %>%

  arrange(total\_offences) %>%

  slice(1:3)

# Create a lollipop chart for regions with the lowest total offences, using a minimal theme

ggplot(low\_offence\_regions, aes(x = reorder(Region\_Name, total\_offences), y = total\_offences)) +

  geom\_segment(aes(xend = Region\_Name, yend = 0), color = "lightgreen") +

  geom\_point(color = "darkgreen", size = 4) +

  coord\_flip() +

  labs(title = "Regions with the Lowest Total Offences", x = "Region", y = "Offence Count") +

  theme\_classic()

# **Plot:**

A graph with green lines

Description automatically generated

# **Justification:**

The lollipop plot is again chosen for its simplicity and effectiveness in highlighting the regions with the lowest total offences. The classic theme enhances the visual appeal and makes it easier to understand the plot.