Part 1 – Python

The city chosen was Paris and the public transport network chosen was “The Paris Métro”.

Network was imported as nx and matplotlib.pyplot was imported as plt.

**Solution Code for Q1:**

import networkx as nx

import matplotlib.pyplot as plt

#plot of graphical visualization

paris\_metro = nx.Graph()

#5 lines with paris metro stations

routes = {

"Line 1": ["La Défense", "Esplanade de La Défense", "Pont de Neuilly", "Champs-Élysées", "Concorde"],

"Line 2": ["Porte Dauphine", "Charles de Gaulle–Étoile", "Place de Clichy", "Anvers", "Nation"],

"Line 4": ["Porte de Clignancourt", "Gare du Nord", "Châtelet", "Saint-Michel", "Montparnasse"],

"Line 6": ["Charles de Gaulle–Étoile", "Trocadéro", "La Motte-Picquet–Grenelle", "Montparnasse", "Nation"],

"Line 8": ["Balard", "La Motte-Picquet–Grenelle", "Concorde", "Opéra", "Reuilly–Diderot"],

}

#nodes and edges

for route, stops in routes.items():

paris\_metro.add\_nodes\_from(stops)

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

paris\_metro.add\_edge(stops[i], stops[i + 1], route=route)

#intersection stations

intersections = [

("Champs-Élysées", "Charles de Gaulle–Étoile"),

("Montparnasse", "La Motte-Picquet–Grenelle"),

("Concorde", "Opéra"),

("Gare du Nord", "Châtelet"),

]

for stop1, stop2 in intersections:

paris\_metro.add\_edge(stop1, stop2, route="Intersection")

#plot and color

layout\_pos = nx.spring\_layout(paris\_metro, seed=24)

edge\_colors = []

#default colour added

for start, end, adttributes in paris\_metro.edges(data=True):

if adttributes["route"] == "Intersection":

edge\_colors.append("gray")

else:

edge\_colors.append(adttributes["route"])

#graph plotted

nx.draw(paris\_metro, layout\_pos, with\_labels=True, node\_size=550, node\_color="lightgrey", font\_size=7)

nx.draw\_networkx\_edges(paris\_metro, layout\_pos)

plt.title("Paris Métro Network")

plt.show()

A network diagram of a metro network

Description automatically generated

Plot for q1 showing Paris Metro Network

Q2: Codes Added to the above code with explanations

Code added 1-

#nodes and edges

for route, stops in routes.items():

for stop in stops:

#stops to the graph, track the routes it is in

if stop not in paris\_metro.nodes:

paris\_metro.add\_node(stop, routes={route})

else:

paris\_metro.nodes[stop]['routes'].add(route)

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

paris\_metro.add\_edge(stops[i], stops[i + 1], route=route)

This was added so that every node can track which lines/routes it belongs to. The transferring stops that belong to more than a single route was found out with a different colour than the rest of the stops.

Code added 2-

color\_map = {

"Line 1": "red",

"Line 2": "cyan",

"Line 4": "lightblue",

"Line 6": "orange",

"Line 8": "pink",

"Intersection": "gray"

}

#Colors to edges that depends on line

edge\_colors = [color\_map[attributes["route"]] for \_, \_, attributes in paris\_metro.edges(data=True)]

The edges in the graph plot were coloured depending on the line they are on which would allow the viewer or reader more understanding and easier comprehension.

Code added 3-

node\_colors = []

for node, attributes in paris\_metro.nodes(data=True):

if len(attributes["routes"]) > 1: # Multi-line stops

node\_colors.append("yellow")

else:

# Use the color of the line the station primarily belongs to

route = next(iter(attributes["routes"]))

node\_colors.append(color\_map[route])

The intersection stops were coloured as yellow. Normal stops were coloured depending on the line they are on by the use of “color\_map”.

Code added 4-

route\_legend = [plt.Line2D([0], [0], color=color, lw=2, label=route) for route, color in color\_map.items()]

plt.legend(handles=route\_legend, title="Lines", loc="lower right")

This was added to give a legend for more clarity to indicate the lines by color in the lower right corner.

Code added 5-

nx.draw(paris\_metro, layout\_pos, with\_labels=True, node\_size=800, node\_color=node\_colors, font\_size=10, font\_color="black")

nx.draw\_networkx\_edges(paris\_metro, layout\_pos, edge\_color=edge\_colors, width=2)

This was added for better readability and clarity by increasing the size of nodes, size of fonts as well as edge’s thickness.

A map of a network

Description automatically generated

Plot for q2 showing Paris Metro Network with coloured stops and routes

Q3:

Codes added and their explanations:

Code added 1-

lines\_with\_distances = {

"Line 1": [

("La Défense", "Esplanade de La Défense", 0.95),

("Esplanade de La Défense", "Pont de Neuilly", 0.70),

("Pont de Neuilly", "Champs-Élysées", 4.56),

("Champs-Élysées", "Concorde", 0.63),

],

"Line 2": [

("Porte Dauphine", "Charles de Gaulle–Étoile", 1.33),

("Charles de Gaulle–Étoile", "Place de Clichy", 2.62),

("Place de Clichy", "Anvers", 1.2),

("Anvers", "Nation", 5.39),

],

"Line 4": [

("Porte de Clignancourt", "Gare du Nord", 2.19),

("Gare du Nord", "Châtelet", 2.69),

("Châtelet", "Saint-Michel", 0.61),

("Saint-Michel", "Montparnasse", 1.86),

],

"Line 6": [

("Charles de Gaulle–Étoile", "Trocadéro", 1.28),

("Trocadéro", "La Motte-Picquet–Grenelle", 1.86),

("La Motte-Picquet–Grenelle", "Montparnasse", 1.8),

("Montparnasse", "Nation", 6.0),

],

"Line 8": [

("Balard", "La Motte-Picquet–Grenelle", 2.05),

("La Motte-Picquet–Grenelle", "Concorde", 2.58),

("Concorde", "Opéra", 0.89),

("Opéra", "Reuilly–Diderot", 5.54),

],

}

The distances between the stations were added. This was measured in kilometers and they were taken from google maps manually.

Added code 2-

for route, edges in routes\_with\_distances.items():

for stop1, stop2, distance in edges:

# Update or create the node with the correct routes it belongs to

if stop1 not in paris\_metro.nodes:

paris\_metro.add\_node(stop1, routes={route})

else:

paris\_metro.nodes[stop1]['routes'].add(route)

if stop2 not in paris\_metro.nodes:

paris\_metro.add\_node(stop2, routes={route})

else:

paris\_metro.nodes[stop2]['routes'].add(route)

# Add the edge with distance

paris\_metro.add\_edge(stop1, stop2, route=route, distance=distance)

This gives the allowance for the connection of stops and edges/routes with distance to the graph. The routes that connect every stop on a single line also have the distances.

Added code 3-

paris\_metro.add\_edge(stop1, stop2, route=route, distance=distance)

This caused an assignment of “distance” to every edge.

Added code 4-

edgelabels = nx.get\_edge\_attributes(paris\_metro, "distance")

nx.draw\_networkx\_edge\_labels(paris\_metro, pos, edge\_labels=edgelabels, font\_size=8)

On the graph plot, this shows the distances inputted by displaying them on the edges.

Added code 5-

plt.text(0.95, 0.01, 'Distances in km', verticalalignment=top, horizontalalignment='right', transform=plt.gca().transAxes, color='black', fontsize=10)

This shows a legend saying “distances in km” for clarity.

A map of a metro network

Description automatically generated

Plot for q3 showing Paris Metro Network with distances between stops

Q4-

Two improvements that can be suggested. One of them is the spacing out the labels from the nodes in order to increase the visibility and clarity. This is because currently the labels and nodes overlap each other causing a very cluttered visualization plot. Station labels can be moved away from the nodes to one side for all.

The other one is to make the important transfer stations even more conspicuous. This could be done by making them even more highlighted to increases emphasis as only making them coloured as yellow is not indicative enough. Larger node sizes could solve this issue and make them more visible while making them stand out for better identification.

Part 2- R

Q1:  
#libraries firstly loaded

library(tidyr)

library(ggplot2)

library(readxl)

library(dplyr)

#file loaded

file\_path <- "/Users/zuhannoor/Documents/1Studies/fundamentals/cw\_r.xlsx"

data <- read\_excel(file\_path, sheet = "Table 3d", skip = 8)

# columns were renamed

colnames(data) <- c(

"Category", "2012\_2013", "2013\_2014", "2014\_2015",

"2015\_2016", "2016\_2017", "2017\_2018",

"2018\_2019", "2019\_2020", "2020\_2021",

"2021\_2022", "Percentage\_Change"

)

year\_columns <- c("2012\_2013", "2013\_2014", "2014\_2015",

"2015\_2016", "2016\_2017", "2017\_2018",

"2018\_2019", "2019\_2020", "2020\_2021", "2021\_2022")

# Conversion of the columns of years into numeric data to make NA

data[year\_columns] <- lapply(data[year\_columns], function(x) as.numeric(as.character(x)))

# NA values removed

data <- data %>%

filter(!is.na(Category)) %>%

filter\_at(vars(all\_of(year\_columns)), any\_vars(!is.na(.)))

#2 categories sliced from the excel file

# Bank and credit fraud in industries sub-categories

bank\_fraudulence\_categories <- c(

"Cheque, plastic card and online bank accounts (not PSP) [note 5, 6]",

"Application fraud (excluding mortgages)",

"Mortgage related fraud",

"Mandate fraud",

"Dishonestly retaining a wrongful credit"

)

data\_frame\_banking <- data %>%

filter(Category %in% bank\_fraudulence\_categories)

# Financial investments [note 11] subcategories

financial\_investments\_categories <- c(

"Share/bond sales or boiler room fraud [note 11]",

"Pyramid or Ponzi schemes [note 11]",

"Prime bank guarantees",

"Time shares and holiday club fraud",

"Other financial investment [note 12,13]"

)

data\_frame\_financial\_investments <- data %>%

filter(Category %in% financial\_investments\_categories)

#conversion to long

bank\_Long <- data\_frame\_banking %>%

pivot\_longer(cols = all\_of(year\_columns), names\_to = "Year", values\_to = "Value") %>%

mutate(Year = gsub("\_", "-", Year))

financial\_investments\_long <- data\_frame\_financial\_investments %>%

pivot\_longer(cols = all\_of(year\_columns), names\_to = "Year", values\_to = "Value") %>%

mutate(Year = gsub("\_", "-", Year))

#check data for first category

str(bank\_Long)

head(bank\_Long)

#check data for second category

str(financial\_investments\_long)

head(financial\_investments\_long)

#Visualizations plot were made below

# 1st category Banking Fraud Dot Plot

ggplot(bank\_Long, aes(x = Year, y = Value, color = Category)) +

geom\_point(position = position\_jitter(width = 0.2), size = 2, alpha = 0.8) +

labs(title = "Banking Fraud Subcategories (Dot Plot)",

x = "Year", y = "Value", color = "Subcategory") +

theme\_minimal() +

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

# 2nd category Financial Investments Dot Plot

ggplot(financial\_investments\_long, aes(x = Year, y = Value, color = Category)) +

geom\_point(position = position\_jitter(width = 0.2), size = 2, alpha = 0.8) +

labs(title = "Financial Investments Subcategories (Dot Plot)",

x = "Year", y = "Value", color = "Subcategory") +

theme\_minimal() +

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

#Present Central Tendency Visualizations mean and median

#mean summary values for each year

banking\_mean <- bank\_Long %>%

group\_by(Year) %>%

summarise(Mean\_Value = mean(Value, na.rm = TRUE))

financial\_investments\_mean <- financial\_investments\_long %>%

group\_by(Year) %>%

summarise(Mean\_Value = mean(Value, na.rm = TRUE))

# Banking Fraudulence Central Tendency graphical Plot Mean

ggplot(banking\_mean, aes(x = Year, y = Mean\_Value)) +

geom\_col(fill = "green", alpha = 0.7) +

labs(title = "Central Tendency (Mean) for Banking Fraud",

x = "Year", y = "Mean Value") +

theme\_minimal() +

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

# Finance Investments Central Tendency graphical Plot Mean

ggplot(financial\_investments\_mean, aes(x = Year, y = Mean\_Value)) +

geom\_col(fill = "red", alpha = 0.7) +

labs(title = "Central Tendency (Mean) for Financial Investments",

x = "Year", y = "Mean Value") +

theme\_minimal() +

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

#median for banking fraudulence

banking\_median <- bank\_Long %>%

group\_by(Year) %>%

summarise(Median\_Value = median(Value, na.rm = TRUE))

#median for finance investments

financial\_investments\_median <- financial\_investments\_long %>%

group\_by(Year) %>%

summarise(Median\_Value = median(Value, na.rm = TRUE))

#Bank Fraudulence Median Central Tendency graphical Plot

ggplot(banking\_median, aes(x = Year, y = Median\_Value)) +

geom\_col(fill = "green", alpha = 0.7) +

labs(title = "Central Tendency Median for Banking Fraud",

x = "Year", y = "Median Value") +

theme\_minimal() +

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

#Finance Investments Median Central Tendency graphical Plot

ggplot(financial\_investments\_median, aes(x = Year, y = Median\_Value)) +

geom\_col(fill = "orange", alpha = 0.7) +

labs(title = "Central Tendency Median for Financial Investments",

x = "Year", y = "Median Value") +

theme\_minimal() +

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

The various libraries mentioned above in the code were loaded to be used to handle the data manipulation and for visualization purposes. Extra library tidyr was used in order to reshape the data and convert them to long so that it could be used for the plots created with ggplot2. The names of the columns in the given excel file were standardized while year columns were subject to conversion to numeric, while rows that had absent values were taken out. Two subsets were chosen from the dataset, changed to the long type.

A graph with colorful dots and numbers

Description automatically generated

Plot showing Dispersion of banking fraud

A graph with colorful dots

Description automatically generated

Plot showing dispersion of financial investments

The above 2 dot plots represent the two categories of fraudulence – banking and financial investment related. It shows how each of the sub-categories of these two categories had increased or decreased over the passage of years in the x axis. Every sub-category within the main categories were shown with colour as well as legend defining them to make them distinguished. Dot plots were chosen for this due to them being very suitable for their ability to highlight overlapping points within the sub-categories.

In the banking fraud plot, it is observed that cheque frauds have been regularly higher than the rest as time went on, peaking at the end while others remained along the same level with minor variance in application fraud. In the financial investment type fraudulence, there is a steady increase in ponzi schemes and share/bond sales type fraud within the given year ranges. While on the other hand, “Other financial investment [note12,13]” seems to have a sharp rise in value along the years being the highest among all the other sub-categories within every year ranges.

A graph of green bars

Description automatically generated

Plot for median central tendency for banking fraud showing an increase until a peak in 2017-2018 then decreasing

A graph of a number of years

Description automatically generated with medium confidence

Plot for median central tendency for financial investment showing a decrease until 2017-2018 except for 2014-15 after which it increased sharply.

A graph of a number of green bars

Description automatically generated

Plot for mean central tendency for banking frauds shows a gradual increase over the years except for 2012-2013, 2017-2018 and 2020-2021.

A graph of a growing graph

Description automatically generated with medium confidence

Plot for mean central tendency for financial investments shows variance between 2012 to 2018. After which there is a sharp rise.

The calculation of the central tendencies were done prior to plotting the data in order to highlight the trends in the two categories. It was done for mean and median for bank and financial industry related fraudulence. The barcharts were chosen as they are one of the best choices for the assessment of visual trends, being quite simple to interpret and distinguish from each year ranges’ bars.

**Q2:**

#libraries firstly loaded

library(dplyr)

library(readxl)

library(ggplot2)

#file loaded

file\_path <- "/Users/zuhannoor/Documents/1Studies/fundamentals/cw\_r.xlsx" # Adjust this path

data <- read\_excel(file\_path, sheet = "Table 5", skip = 9)

# Renamed the columns

colnames(data) <- c("Area\_Code", "Name\_of\_Area", "Offence\_Total", "Rate\_Per\_1000", "Change\_From\_Prev\_Year")

#rows removed with missing

data <- data %>%

filter(!is.na(Name\_of\_Area) & Name\_of\_Area != "")

#missing data for "City of London" handled

city\_of\_london\_impute <- median(data$Offence\_Total, na.rm = TRUE)

data <- data %>%

mutate(Offence\_Total = ifelse(Name\_of\_Area == "City of London" & is.na(Offence\_Total),

city\_of\_london\_impute,

Offence\_Total))

# removed the particular row Unknown [note 15]

data <- data %>%

filter(Name\_of\_Area != "Unknown [note 15]")

# Regions table was created with exclusion

# these were removed- 'ENGLAND AND WALES [note 13]' and 'ENGLAND' as well as 'WALES'

regions <- c(

"North East", "North West", "Yorkshire and The Humber", "East Midlands",

"West Midlands", "East", "South East", "South West", "London [note 14]"

)

# Counties under WALES excluded according to question

wales\_counties <- c("Dyfed-Powys", "Gwent", "North Wales", "South Wales")

# Filtered the regions wanted

data\_of\_region <- data %>%

filter(Name\_of\_Area %in% regions)

# Filtered the counties wanted and excluded WALES's counties

data\_of\_county <- data %>%

filter(!Name\_of\_Area %in% c(regions, "ENGLAND AND WALES [note 13]", "ENGLAND", "WALES")) %>%

filter(!Name\_of\_Area %in% wales\_counties)

#Summarization

summary\_of\_region <- data\_of\_region %>%

group\_by(Name\_of\_Area) %>%

summarise(

Offence\_Total = sum(Offence\_Total, na.rm = TRUE),

#Per\_1000\_AVG = mean(Rate\_Per\_1000, na.rm = TRUE)

)

summary\_of\_county <- data\_of\_county %>%

group\_by(Name\_of\_Area) %>%

summarise(

Offence\_Total = sum(Offence\_Total, na.rm = TRUE),

#er\_1000\_AVG = mean(Rate\_Per\_1000, na.rm = TRUE)

) %>%

arrange(Offence\_Total)

# Check data\_of\_region

str(data\_of\_region)

head(data\_of\_region)

# Check data\_of\_county

str(data\_of\_county)

head(data\_of\_county)

# Check summary\_of\_region

str(summary\_of\_region)

head(summary\_of\_region)

# Check summary\_of\_county

str(summary\_of\_county)

head(summary\_of\_county)

#graphical Plot for regions to find the highest count of offences

ggplot(summary\_of\_region, aes(x = reorder(Name\_of\_Area, Offence\_Total), y = Offence\_Total)) +

geom\_col(fill = "cyan") +

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

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 30, hjust = 1))

# Rate\_Per\_1000 had been checked to be numeric and missing values excluded by NA

data\_of\_region <- data\_of\_region %>%

mutate(Rate\_Per\_1000 = as.numeric(Rate\_Per\_1000)) %>%

filter(!is.na(Rate\_Per\_1000))

#total offence rate for every region in england found

summary\_region\_rate <- data\_of\_region %>%

group\_by(Name\_of\_Area) %>%

summarise(

Total\_Offence\_Rate = sum(Rate\_Per\_1000, na.rm = TRUE)

) %>%

arrange(desc(Total\_Offence\_Rate))

#checking of summary

if (nrow(summary\_region\_rate) == 0) {

stop("No valid data found for Rate\_Per\_1000 in data\_of\_region.")

}

# Identification of regions for the highest offence rate in england

region\_High <- summary\_region\_rate %>%

filter(Total\_Offence\_Rate == max(Total\_Offence\_Rate))

#graphical Plot of offence rate for all regions

ggplot(summary\_region\_rate, aes(x = reorder(Name\_of\_Area, Total\_Offence\_Rate), y = Total\_Offence\_Rate)) +

geom\_col(fill = "darkblue") +

labs(title = "Total Offence Rate per 1000 Population in England",

x = "Region",

y = "Total Offence Rate per 1000 Population in england") +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 30, hjust = 1))

#top three counties with lowest count of offences found

county\_Low <- summary\_of\_county %>%

arrange(Offence\_Total) %>%

head(3)

# graphical Plot of the data

ggplot(county\_Low, aes(x = reorder(Name\_of\_Area, Offence\_Total), y = Offence\_Total)) +

geom\_col(fill = "orange") +

labs(title = "Top 3 Counties with Lowest Total Offences in England from data",

x = "County",

y = "Total Offences") +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 30, hjust = 1))

Process:-  
Data set was loaded from the local computer, and libraries were loaded. Column names were renamed for the purpose of clarity. Absent values were taken out as well as Unknown [note15]’s row. The region of England was required so Wales was taken out along with its counties. There was a split of two subsets from the dataset into data\_of\_region and data\_of\_county. Summaries were generated to find the total and total rate of offences as well as the lowest counts as per the specification. The plots were created for the purpose of easy visualization of finding the region with highest offences alongside the rate per thousand people. There was also a plot among the counties to find the safest three counties.

A graph of blue squares

Description automatically generated

The plot above shows that London [note 14] has the highest total offences

A graph of a number of people

Description automatically generated

The plot above indicates that West Midlands has the highest total offence per 1000 amongst the regions

A graph with orange squares

Description automatically generated

The plot above shows that Avon and Somerset, Cumbria and Cleveland have the lowest total offences amongst all the counties.

The bar chart was chosen as it is easy to understand and provides clarity and for these categorical data. This is why the plot is uncluttered and easy to comprehend when seen without much effort.