DigData Career Challenge

Sustainable Travel Insights

Authored by:
Anuja Gaikwad

❖ Data Science Approach: (Path A)

Used 3 datasets for analysis: festival_journeys dataset, emissions dataset and congestion factor dataset. Tools used: R Studio and R.

Question: "Can you advise us how different travel choices can affect the carbon footprint?Leverage diverse data sources to recommend best travel options to meet environmentally friendly travel options."

1. Read in data ...Open/import data into your tool of choice ready for the next steps
2. Examine data ...Identify which data is useful to help answer the question
3. Clean up data ...Fix any data issues that may be present (incomplete rows, null values etc)
4. Data analysis ...Write code to analyse the datasets and generate results
5. Visualise the data ...Use graphs/charts to visualise the results and identify trends
6. Interpret and report findings ...Make decisions based on the data and create a report of your findings

R code for the task:

```
# Load required libraries
library(readxl)  # For reading Excel files
library(dplyr)  # For data manipulation
library(ggplot2)  # For data visualization
library(stringr)  # For string manipulation

# Step 1: Read in data
festival_journeys = "C:/Users/anuja/OneDrive/Desktop/DigData/Festival_journeys dataset.xlsx"
emissions = "C:/Users/anuja/OneDrive/Desktop/DigData/Emissions dataset.xlsx"
congestion = "C:/Users/anuja/OneDrive/Desktop/DigData/Congestion Factor dataset.xlsx"

# Load datasets
festival_journeys_file = read_excel(festival_journeys)
emissions_file = read_excel(emissions)
congestion_file = read_excel(congestion)

View(festival_journeys_file)
View(emissions_file)
View(congestion_file)

colnames(festival_journeys_file)
colnames(emissions_file)
colnames(congestion_file)
# Step 2: Examine data
print("Festival Journeys Data Overview:")
print(head(festival_journeys_file))

print("Emissions Data Overview:")
print(head(emissions_file))

print("Congestion Data Overview:")
print(head(congestion_file))

# Step 2: Clean column names (strip whitespace and remove special characters)
names(festival_journeys_file) = str_trim(str_replace_all(names(festival_journeys_file), "\ufeff", ""))
names(emissions_file) = str_trim(str_replace_all(names(emissions_file), "\ufeff", ""))
names(congestion_file) = str_trim(str_replace_all(names(congestion_file), "\ufeff", ""))
```

```r
# Step 3: Clean up data
# Convert columns to numeric, coercing errors to NA
festival_journeys_file = festival_journeys_file %>%
mutate(
   Distance = as.numeric(`Distance`),
   Travel_Time = as.numeric(`Travel Time`)
 )


emissions_file = emissions_file %>%
 mutate(Emissions_per_Mile = as.numeric(`Emissions per Mile`))

# Drop rows where critical columns have NA values
festival_journey_file = festival_journeys_file %>% filter(!is.na(Distance) & !is.na(`Travel Time`))

emissions_file = emissions_file %>% filter(!is.na(`Emissions per Mile`))

# Verify data types
print(str(festival_journeys_file))
print(str(emissions_file))

# Step 4: Data analysis
# Merge festival journeys with emissions data on 'Mode of Transport'
data_combined = festival_journeys_file %>%
left_join(emissions_file, by = "Mode of Transport")

# Calculate emissions for each journey
data_combined = data_combined %>%
 mutate(Total_Emissions_kg_CO2 = `Distance` * `Emissions per Mile`)


# Step 5: Visualize the data
# Emissions by mode of transport
ggplot(data_combined, aes(x = `Mode of Transport`, y = `Emissions per Mile`)) +
 geom_bar(stat = "summary", fun = "sum", fill = "green") +
 theme_minimal() +
 labs(
  title = "Total Emissions by Mode of Transport",
  x = "Mode of Transport",
  y = "Total Emissions (kg CO2)"
 ) +
 theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Distribution of travel times by mode of transport
ggplot(data_combined, aes(x = `Mode of Transport`, y = `Travel Time`)) +
geom_boxplot(fill = "lightblue", color = "purple") +
 theme_minimal() +
 labs(
  title = "Distribution of Travel Time by Mode of Transport",
  x = "Mode of Transport",
  y = "Travel Time (mins)"
 ) +
 theme(axis.text.x = element_text(angle = 45, hjust = 1))

data_combined = data_combined %>%
 select(-Origin)
```

```
library(GGally)
ggpairs(data_combined, aes(colour = 'Mode of Transport'))

ggplot(data_combined,aes(x=`Total_Emissions_kg_CO2`,y=`Distance`)) +
  geom_point() +
  geom_smooth(method=lm,se=FALSE)

model = lm(Distance~Total_Emissions_kg_CO2, data=data_combined)
summary(model)
library(ggfortify)
autoplot(model)
# Step 6: Interpret and report findings
# Group data by mode of transport to get average emissions and travel times
summary = data_combined %>%
  group_by(`Mode of Transport`) %>%
  summarise(
    Average_Emissions = mean(`Emissions per Mile`, na.rm = TRUE),
    Average_Travel_Time = mean(`Travel Time`, na.rm = TRUE),
    Average_Distance = mean(`Distance`, na.rm = TRUE)
  )

print("Summary of Travel Modes:")
print(summary)

# Identify the most eco-friendly and time-efficient travel options
emissions_low = summary %>%
  filter(Average_Emissions == min(Average_Emissions, na.rm = TRUE))

time_shortest = summary %>%
  filter(Average_Travel_Time == min(Average_Travel_Time, na.rm = TRUE))

cat("\nThe most eco-friendly and optimal mode of transport is:", emissions_low$`Mode of Transport`,
    "with an average of", emissions_low$Average_Emissions, "kg CO2 per journey.\n")

cat("\nThe fastest mode of transport is:", time_shortest$`Mode of Transport`,
    "with an average travel time of", time_shortest$Average_Travel_Time, "minutes per journey.\n")
```

Output from console:

```
> colnames(festival_journeys_file)
[1] "Origin"          "Mode of Transport" "Distance"          "Travel Time"
[5] "Age Group"       "Carpooling"        "Weather"           "Peak/NoPeak"
[9] "Day"
> colnames(emissions_file)
[1] "Mode of Transport"  "Emissions per Mile"
> colnames(congestion_file)
[1] "...1"      "Peak time" "Non-peak"
```

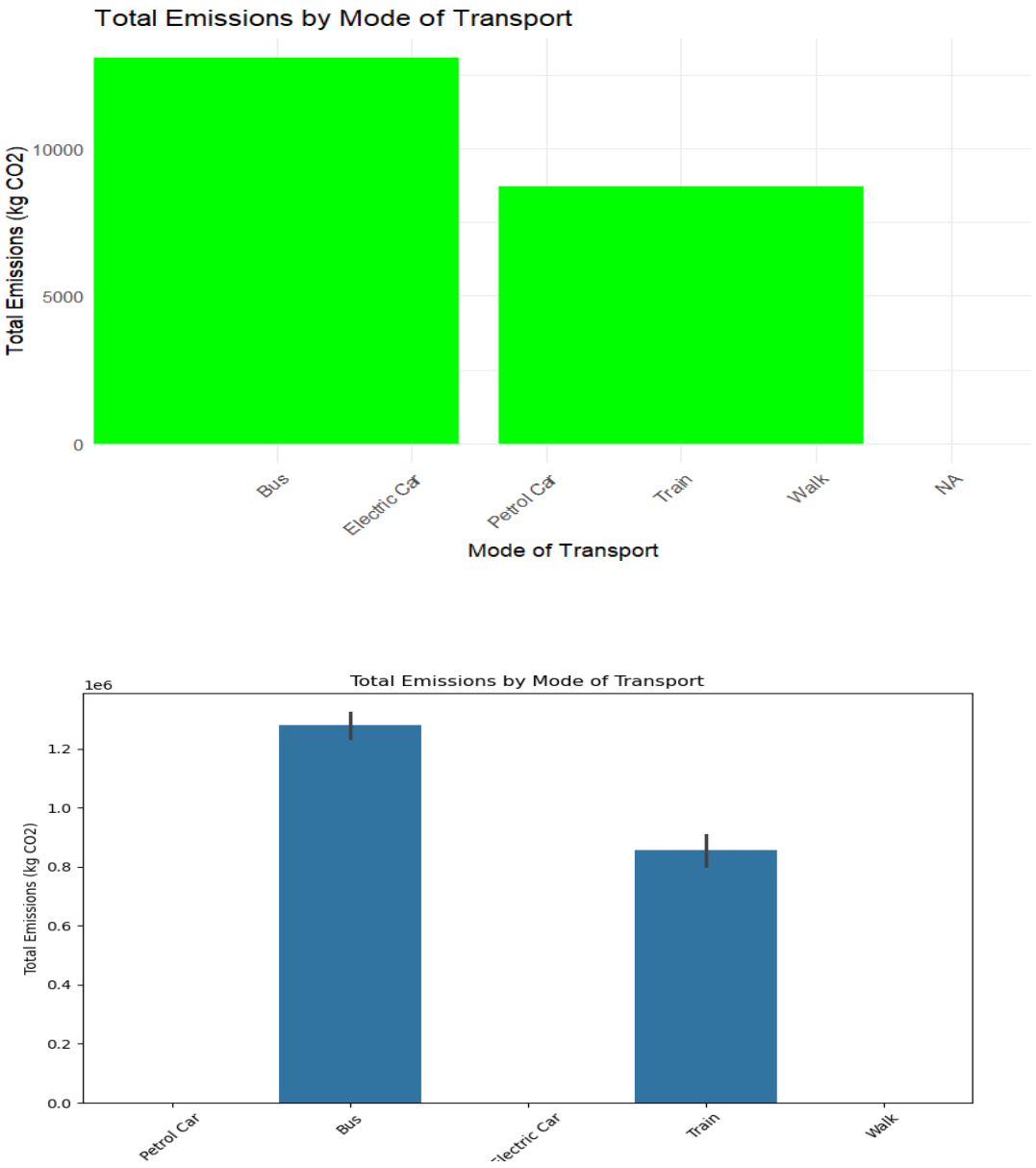

Fig 1: Bar graphs of total emissions by mode of transport

From the bar graphs, it is evident that, bus and train have more emission rate as compared to any other mode of transport. Traveling by walk does not cause any emission which is very logical to think about.

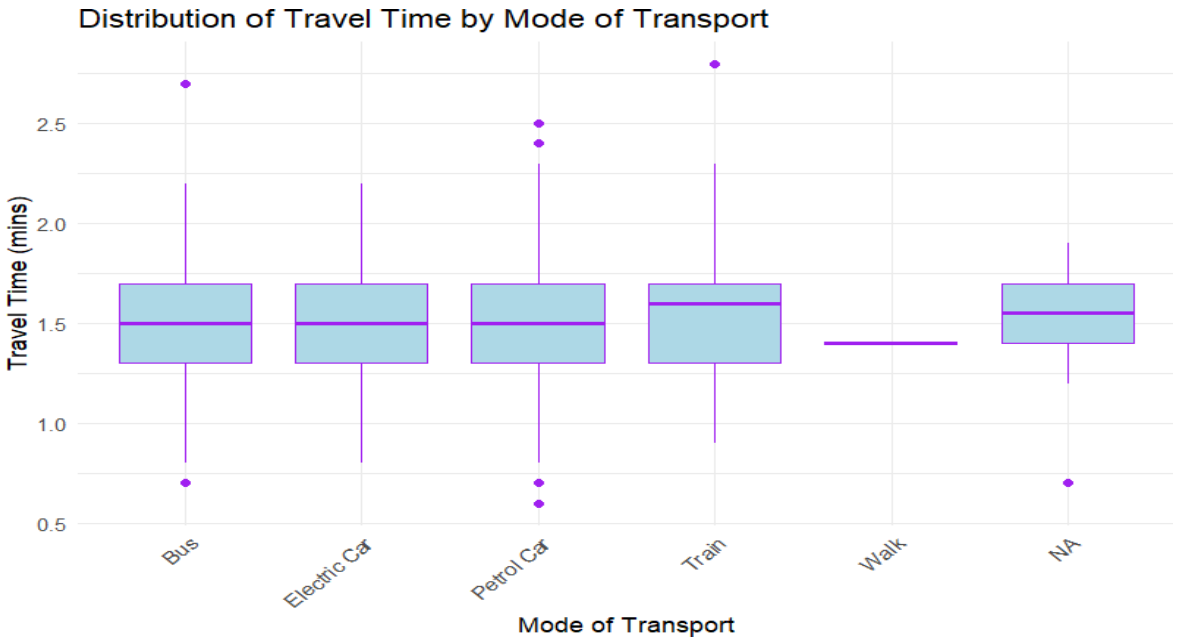

Fig 2: Boxplot of distribution of travel time by mode of transport

As can be seen from the boxplot, Bus, Electric cars and Petrol cars have the same travel time but bus and petrol cars have some points which are straight outliers and need some further investigation. Whereas, travel time by walk is N/A.

Fig 3: Scatter matrix using ggpairs()

```
> model = lm(Distance~Total_Emissions_kg_CO2, data=data_combined)
> summary(model)

Call:
lm(formula = Distance ~ Total_Emissions_kg_CO2, data = data_combined)

Residuals:
   Min     1Q  Median     3Q    Max
-32.169 -14.558   2.515   9.673  28.291

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         1.516e+01  1.621e+00  9.356  <2e-16 ***
Total_Emissions_kg_CO2 1.745e-02  3.199e-04  54.562  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 12.02 on 449 degrees of freedom
  (1049 observations deleted due to missingness)
Multiple R-squared:  0.8689,        Adjusted R-squared:  0.8687
F-statistic:  2977 on 1 and 449 DF,  p-value: < 2.2e-16
```

As can be seen from the scatter matrix, the correlation coefficient between Distance and Total_Emissions_kg_CO2 is quite high of 0.932. Thus, it appears that it is a very good model for predicting the best mode of transport according to distance and rate of carbon emissions. The R-squared value of 0.8689 is quite high which suggests that it is a very optimal model again.

```
Summary of Travel Modes:
   Mode of Transport  Total Emissions (kg CO2)  Travel Time     Distance
0                Bus                4303.421477     1.504027    97.805034
1       Electric Car                        NaN     1.515484    97.007742
2          Petrol Car                       NaN     1.493957   101.082668
3              Train                5611.184314     1.554902    98.441830
4               Walk                        NaN     1.400000    76.100000
```

From the summary table, it can be seen that bus is the most eco-friendly mode of transport as the total emissions by bus are the lowest as compared to other modes of transport. The fastest mode of transport is by walk by comparing the travel times of other modes of transport. Thus, for festival goers, bus is an optimal solution and the fastest and efficient way is by walk.

```
The most eco-friendly and optimal mode of transport is: Bus with an average of 44
kg CO2 per journey.

The fastest mode of transport is: Walk with an average travel time of 1.4 minutes
per journey.
```

Python code for the same task:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 1: Read in data
festival_journeys_path = r"C:\Users\anuja\OneDrive\Desktop\DigData\Festival_journeys dataset.xlsx"
emissions_path = r"C:\Users\anuja\OneDrive\Desktop\DigData\Emissions dataset.xlsx"
congestion_path = r"C:\Users\anuja\OneDrive\Desktop\DigData\Congestion Factor dataset.xlsx"

# Load datasets
festival_journeys = pd.read_excel(festival_journeys_path)
emissions = pd.read_excel(emissions_path)
congestion = pd.read_excel(congestion_path)

# Step 2: Examine data
print("Festival Journeys Data Overview:\n", festival_journeys.head(), "\n")
print("Emissions Data Overview:\n", emissions.head(), "\n")
print("Congestion Data Overview:\n", congestion.head(), "\n")

# Step 2: Clean column names (strip whitespace and remove special characters)
festival_journeys.columns = festival_journeys.columns.str.strip().str.replace('\ufeff', '')
emissions.columns = emissions.columns.str.strip().str.replace('\ufeff', '')
congestion.columns = congestion.columns.str.strip().str.replace('\ufeff', '')

# Print column names to verify
print("Festival Journeys Columns:", festival_journeys.columns)
print("Emissions Columns:", emissions.columns)
print("Congestion Columns:", congestion.columns)



# Step 3: Clean up data
# Convert columns to numeric, coercing errors to NaN
festival_journeys['Distance'] = pd.to_numeric(festival_journeys['Distance'], errors='coerce')
festival_journeys['Travel Time'] = pd.to_numeric(festival_journeys['Travel Time'], errors='coerce')
emissions['Emissions per Mile'] = pd.to_numeric(emissions['Emissions per Mile'], errors='coerce')
#congestion['Congestion Factor'] = pd.to_numeric(congestion['Congestion Factor'], errors='coerce')

# Drop rows where critical columns have NaN values
festival_journeys.dropna(subset=['Distance', 'Travel Time'], inplace=True)
emissions.dropna(subset=['Emissions per Mile'], inplace=True)
#congestion.dropna(subset=['Congestion Factor'], inplace=True)

# Verify data types
print(festival_journeys.dtypes)
print(emissions.dtypes)
#print(congestion.dtypes)


# Step 4: Data analysis
# Merge festival journeys with emissions data on mode of transport
combined_data = pd.merge(festival_journeys, emissions, on='Mode of Transport', how='left')

# Calculate emissions for each journey
```

```
combined_data['Total Emissions (kg CO2)'] = combined_data['Distance'] * combined_data['Emissions per Mile']


# Step 5: Visualize the data
# Emissions by mode of transport
plt.figure(figsize=(10, 6))
sns.barplot(x='Mode of Transport', y='Total Emissions (kg CO2)', data=combined_data, estimator=sum)
plt.title('Total Emissions by Mode of Transport')
plt.xticks(rotation=45)
plt.ylabel('Total Emissions (kg CO2)')
plt.xlabel('Mode of Transport')
plt.show()

# Distribution of travel times by mode of transport
plt.figure(figsize=(10, 6))
sns.boxplot(x='Mode of Transport', y='Travel Time', data=combined_data)
plt.title('Distribution of Adjusted Travel Time by Mode of Transport')
plt.xticks(rotation=45)
plt.ylabel('Adjusted Travel Time (mins)')
plt.xlabel('Mode of Transport')
plt.show()

# Step 6: Interpret and report findings
# Group data by mode of transport to get average emissions and travel times
summary = combined_data.groupby('Mode of Transport').agg({
    'Total Emissions (kg CO2)': 'mean',
    'Travel Time': 'mean',
    'Distance': 'mean'
}).reset_index()

print("Summary of Travel Modes:\n", summary)

# Identify the most eco-friendly and time-efficient travel options
lowest_emissions = summary.loc[summary['Total Emissions (kg CO2)'].idxmin()]
shortest_time = summary.loc[summary['Travel Time'].idxmin()]

print("\nThe most eco-friendly mode of transport is:", lowest_emissions['Mode of Transport'],
      "with an average of", lowest_emissions['Total Emissions (kg CO2)'], "kg CO2 per journey.")

print("\nThe fastest mode of transport is:", shortest_time['Mode of Transport'],
      "with an average travel time of", shortest_time['Travel Time'], "minutes per journey.")
```