Assignment 2

Sarosh Farhan (24210969)

Table of contents

Task 1: Manipulation

1.1 Loading the data

```
#loading necessary libraries
library(rio)
library(dplyr)
library(lubridate)
library(scales)
library(tidyr)
library(ggplot2)

#loading the data as tibble using rio package
pedestrian <- import("pedestrian_2023.csv", setclass = "tibble")

#filtering data to get rid of columns with IN and OUT as suffix
pedestrian <- select(pedestrian,
                     -ends_with("IN"), -ends_with("OUT"))

#taking out the dimension to get the number of rows and column
dim(pedestrian)
```

```
[1] 8760   27
```

Here I loaded the data as tibble using the *import()* function from the *rio* packege. It gives *setclass* argument which lets us store the dataframe as tibble.

I also removed the columns which ended with **IN** and **OUT** using the *dplyr select()* command.

I then printed the number of columns and number of rows as asked in the question, which comes out to be 8760 rows and 27 columns.

1.2 Fixing variables to have correct data type

```
check <- class(pedestrian$Time) %in% c("POSIXct", "POSIXt")

if (check == FALSE){
  pedestrian$Time <- dmy_hm(pedestrian$Time)
  class(pedestrian$Time)
  print("Times changed to correct data type")
}else{
  print("Time is already in correct data type")
}
```

```
[1] "Times changed to correct data type"
```

To fix the data type of Time column in the data-set I first checked if the Time column's class belongs to either **POSIXct** or **POSIXt.** After this I checked it with if statement with the condition that Time column didn't belong to correct datatype then I changed the datatype using *lubridate* library's *dmy_hm()* method. Once updated I just output a message stating the update was applied as a feedback.

```
#changing all other coulumns datatypes
for (col in names(pedestrian)){
```

```
  if (col != "Time"){
    if (!is.numeric(pedestrian[[col]])){
      pedestrian[[col]] <- as.numeric(pedestrian[[col]])
    }
  }
}

str(pedestrian, width = 80, strict.width = "wrap")
```

```
tibble [8,760 x 27] (S3: tbl_df/tbl/data.frame)
$ Time : POSIXct[1:8760], format: "2023-01-01 00:00:00" "2023-01-01 01:00:00"
   ...
$ Aston Quay/Fitzgeralds : int [1:8760] 3700 4369 3807 2477 1534 491 391 864
   708 764 ...
$ Baggot st lower/Wilton tce inbound : int [1:8760] 74 100 52 42 11 5 4 14 3 32
   ...
$ Baggot st upper/Mespil rd/Bank : int [1:8760] 4296 1466 1642 967 328 291 178
   1364 2177 3549 ...
$ Capel st/Mary street : int [1:8760] 808 639 546 349 309 190 187 438 570 882
   ...
$ College Green/Bank Of Ireland : int [1:8760] 1325 1209 1151 1368 710 156 99
   108 129 169 ...
$ College Green/Church Lane : int [1:8760] 829 685 589 556 207 67 113 57 93 151
   ...
$ College st/Westmoreland st : int [1:8760] 801 641 481 395 110 44 48 53 74 128
   ...
$ D'olier st/Burgh Quay : int [1:8760] 1714 1411 1219 1225 559 262 178 116 263
   546 ...
$ Dame Street/Londis : int [1:8760] 858 574 664 771 337 100 100 112 113 148 ...
$ Grafton st/Monsoon : int [1:8760] 251 1319 1181 1430 673 128 119 125 134 329
   ...
$ Grafton Street / Nassau Street / Suffolk Street: int [1:8760] 362 289 277 238
   61 26 24 29 31 50 ...
$ Grafton Street/CompuB : int [1:8760] 870 1243 890 740 390 95 30 74 122 168
   ...
$ Grand Canal st upp/Clanwilliam place : int [1:8760] 48 56 31 46 10 1 7 11 8
   20 ...
$ Grand Canal st upp/Clanwilliam place/Google : int [1:8760] 9 87 27 26 12 5 7
   7 11 17 ...
$ Henry Street/Coles Lane/Dunnes : int [1:8760] 1102 836 437 456 294 131 120 94
   181 507 ...
$ Mary st/Jervis st : int [1:8760] 189 81 59 67 30 16 17 14 30 25 ...
$ North Wall Quay/Samuel Beckett bridge East : int [1:8760] 2643 1795 1011 677
   368 728 1330 327 805 876 ...
$ North Wall Quay/Samuel Beckett bridge West : int [1:8760] 43 582 262 188 123
   81 244 106 234 550 ...
$ O'Connell St/Parnell St/AIB : int [1:8760] 548 389 319 215 201 77 63 76 77
   122 ...
$ O'Connell st/Princes st North : int [1:8760] 1408 891 870 722 283 93 121 171
   175 233 ...
$ Phibsborough Rd/Enniskerry Road : int [1:8760] 56 86 63 90 36 8 9 18 19 21
   ...
$ Richmond st south/Portabello Harbour inbound : int [1:8760] 388 151 445 512
   328 97 45 94 178 171 ...
$ Richmond st south/Portabello Harbour outbound : int [1:8760] 201 454 277 496
   230 51 28 38 51 65 ...
$ Talbot st/Guineys : int [1:8760] 2133 1347 751 654 794 703 429 323 459 1584
   ...
$ Westmoreland Street East/Fleet street : int [1:8760] 1011 634 501 445 127 33
   34 39 61 78 ...
```

```
$ Westmoreland Street West/Carrolls : int [1:8760] 1835 1426 1458 1709 786 187
   119 122 156 274 ...
```

I also checked if other columns are of the *numeric* class or not, if not then I changed every columns as numeric and then printed the structure of the tibble using *str()* command. Seeing the output we can infer that all of the columns are in correct datatype now,

1.3 Loading another data-set and renaming variables

```
#importing weather
weather <- import("weather_2023.txt", setclass = "tibble")

weather <- weather |> rename(precipitation = rain,
                       airTemp = temp,
                       meanWndSpeed = wdsp, cloudAmt = clamt)
dim(weather)
```

```
[1] 8760    5
```

I loaded the weather data-set as tibble using the *import()* command and then changed the column names to something more meaningful so that I don't confuse it.

I then printed the number of rows and columns of this data-set which comes out to be 8760 rows and 5 columns.

1.4 Converting a variable in an ordered factor

```
#using cut to create ordered factor
weather$cloudAmt = factor(weather$cloudAmt,
                       levels = c(0:9),
                       labels = c(
                         "0: No cloud",
                         "1: 1/8 part cloud",
                         "2: 2/8 part cloud",
                         "3: 3/8 part cloud",
                         "4: 4/8 part cloud",
                         "5: 5/8 part cloud",
                         "6: 6/8 part cloud",
                         "7: 7/8 part cloud",
                         "8: Fully clouded",
                         "9: Sky obscured"
                       ),
                       ordered = TRUE)

#to print the levels
levels(weather$cloudAmt)
```

```
 [1] "0: No cloud"       "1: 1/8 part cloud" "2: 2/8 part cloud"
 [4] "3: 3/8 part cloud" "4: 4/8 part cloud" "5: 5/8 part cloud"
 [7] "6: 6/8 part cloud" "7: 7/8 part cloud" "8: Fully clouded"
[10] "9: Sky obscured"
```

```
#to check if the factors are ordered or not
is.ordered(weather$cloudAmt)
```

```
[1] TRUE
```

I uses the cut function to create an ordered factor for cloud amount column, the levels of factors range from 0 to 9 as per the data. I then showed the levels using *levels()* method and then checked if the factors are ordered or not using *is.ordered()* method.

1.5 Use of skim_without_charts()

```
library(skimr)
skim_without_charts(weather)
```

Table 1: Data summary

Name	weather
Number of rows	8760
Number of columns	5
_____	
Column type frequency:	
factor	1
numeric	3
POSIXct	1
_____	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cloudAmt	0	1	TRUE	9	7: : 3865, 6: : 1338, 8: : 733, 1: : 727

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
precipitation	0	1	0.11	0.50	0.0	0.0	0.0	0.0	10.6
airTemp	0	1	10.74	4.85	-4.3	7.4	10.7	14.3	25.5
meanWndSpeed	0	1	8.99	4.21	0.0	6.0	9.0	11.0	31.0

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
Time	0	1	2023-01-01	2023-12-31 23:00:00	2023-07-02 11:30:00	8760

As we see from the output, we get a summary of the data using the above function. It is an alternative to *summary()* method. Right off the bat we see the number of rows and number of columns and the frequency of the column type of the data.

We also see the mean, standard deviation and percentiles of the numeric variables.

We see from the output summary that there are 8760 rows and 5 columns and the different datatypes that the columns are.

We also see that it gives a nice summary table for the numeric variables complete with mean, standard deviation, 0, 25, 50, 75 100 percentiles as well.

1.6 Checking data type of Time and checking the range of Time

```
#taking bool so that we can check if data type is
#of POSIXt or POXSIXct
bool <- class(weather$Time) %in% c("POSIXct", "POSIXt")

#checking if data type is correct if found false
#datatype is changed to POSIXct
if(bool[1] == TRUE){
  print("Time is in correct data type")
}else{
  print("Time is not in correct data type, so changing it in POSIXct")
```

```
  weather$Time <- dmy_hm(weather$Time)
  class(weather$Time)
  print("Time changed to correct data type")
}
```

```
[1] "Time is in correct data type"
```

In the code snippet above I have checked if the Time column in weather data-set is of *POSIXct* or *POSIXt* and then if we find it to be true then we don't do anything, if found false, I change it to correct time format using *dmy_hm()* and then print the class to double-check if the data-type was converted or not.

```
#checking if range of weather and pedestrian data are same or not
range(weather$Time) - range(pedestrian$Time)
```

```
Time differences in secs
[1] 0 0
```

In the above code snippet I have just used *range()* on Time column of both pedestrian and weather data-set and then subtracted one from the other to see if there is a difference or not, we see from the output that the Time difference is 0 hence the range of Time for both pedestrian and weather data are equal.

1.7 Joining the two datasets

```
#merged datasets for weather and pedestrian
joinedData <- merge(pedestrian, weather, by = "Time")
```

I join both the data-set using *merge()* method, this is a base R method, looking at the help file I found that by default it this gives a *natural join* which is a special case of *inner join()*.

1.8 Adding two new columns for day and month

```
#mutaing and adding column for day and month
joinedData <- mutate(joinedData,
                     day = wday(Time, label = TRUE),
                     month = month(Time, label = TRUE))

#checking if day and month are odered or not
is.ordered(joinedData$day)
```

```
[1] TRUE
```

```
is.ordered(joinedData$month)
```

```
[1] TRUE
```

I added two new columns using *mutate()* method and extracted day from the time format using *wday()* method from *lubridate* library with *label=TRUE* as an argument which gives day of the week as an ordered factor of character strings such as *Sunday* from the given date-time format.

I extracted month from the given Time column using the *month()* method from *lubridate* package and added *label = TRUE* as an argument which gives the month on the year as a character string and it will display an abbreviated version of the label such a *Jan*.

Lastly I checked if the factors are ordered or not using the *is.ordered()* method, which from the output above we see it as TRUE.

1.9 Using dplyr::relocate() to relocate month and day to second and third column

```
#relocating the columns
joinedData <- joinedData |>
              relocate(month, .after = 1) |>
              relocate(day, .after = month)

colnames(joinedData)
```

```
 [1] "Time"
 [2] "month"
 [3] "day"
 [4] "Aston Quay/Fitzgeralds"
 [5] "Baggot st lower/Wilton tce inbound"
 [6] "Baggot st upper/Mespil rd/Bank"
 [7] "Capel st/Mary street"
 [8] "College Green/Bank Of Ireland"
 [9] "College Green/Church Lane"
[10] "College st/Westmoreland st"
[11] "D'olier st/Burgh Quay"
[12] "Dame Street/Londis"
[13] "Grafton st/Monsoon"
[14] "Grafton Street / Nassau Street / Suffolk Street"
[15] "Grafton Street/CompuB"
[16] "Grand Canal st upp/Clanwilliam place"
[17] "Grand Canal st upp/Clanwilliam place/Google"
[18] "Henry Street/Coles Lane/Dunnes"
[19] "Mary st/Jervis st"
[20] "North Wall Quay/Samuel Beckett bridge East"
[21] "North Wall Quay/Samuel Beckett bridge West"
[22] "O'Connell St/Parnell St/AIB"
[23] "O'Connell st/Princes st North"
[24] "Phibsborough Rd/Enniskerry Road"
[25] "Richmond st south/Portabello Harbour inbound"
[26] "Richmond st south/Portabello Harbour outbound"
[27] "Talbot st/Guineys"
[28] "Westmoreland Street East/Fleet street"
[29] "Westmoreland Street West/Carrolls"
[30] "precipitation"
[31] "airTemp"
[32] "meanWndSpeed"
[33] "cloudAmt"
```

As per the task, I used *relocate()* method from *dplyr* library to relocate the *month* column to the second position, the argument *.after = 1* does the relocation of *month* column to the second column. Similarly relocating *day* column but here I provided argument *.after = day* so that it relocates the *day* column after the *month* column which also happens to be the third column.

Task2: Analysis

2.1 Computing months that had the highest and lowest overall pedestrian traffic

I first calculated the total pedestrian count by summing up the row data so that I have the hourly total of the footfall, this was done using *rowSums()* method.

I then created a new data frame to get aggregate of the monthly footfalls, for this I used *aggregate()* method.

I then get the index of maximum value for the footfall based on the month by using *which.max()* method. I also get the minimum using the *which.max()* method and created a table to show the output.

```
#calculted the row sumns to get total footfalls of each
#area in the dataset
joinedData$totalPedestrian <- rowSums(joinedData[, 4:29],
                                      na.rm = TRUE)

#aggregate to get total footfalls mothly from the data
totalMonthly <- aggregate(totalPedestrian ~ month,
                          joinedData, sum)

#getting the maximum pedestrian for month
maxMonth <- totalMonthly[which.max(totalMonthly$totalPedestrian), ]

#geting the minimum pedestrian fro month
minMonth <- totalMonthly[which.min(totalMonthly$totalPedestrian), ]

#combine results into a single data frame for display
result_table <- data.frame(
  Month = c(maxMonth$month, minMonth$month),
  Total_Pedestrian = c(maxMonth$totalPedestrian,
                       minMonth$totalPedestrian),
  Status = c("Highest", "Lowest")
)
#diplay table
knitr::kable(result_table,
      col.names =   c("Month",
                      "Total Pedestrian",
                      "Status"),
      caption = "Monthly Pedestrian Traffic:
      Highest and Lowest Months")
```

Table 5: Monthly Pedestrian Traffic: Highest and Lowest Months

Month	Total Pedestrian	Status
Mar	25002430	Highest
Jun	15128010	Lowest

We see that March saw the most overall pedestrian footfall at 25,002,430 and June saw the lowest overall footfall at 15,128,010.

2.2 Use ggplot() to create a plot displaying three time series of daily pedestrian footfall in three locations of your choice. Add two vertical bars to mark St. Patrick's day and Christmas Day

```
#Select three locations for the plot
locations <- c("Aston Quay/Fitzgeralds",
               "Capel st/Mary street",
               "Baggot st upper/Mespil rd/Bank")

#Prepare the data for plotting
plot_data <- joinedData %>%
  select(Time, all_of(locations)) %>%
  pivot_longer(cols = all_of(locations),
               names_to = "Location",
               values_to = "Footfall") %>%
  mutate(Date = as_date(Time))

#Calculate daily totals
daily_data <- plot_data %>%
  group_by(Date, Location) %>%
  summarize(DailyFootfall = sum(Footfall,
                                na.rm = TRUE), .groups = "drop")

#Create the plot
ggplot(daily_data, aes(x = Date, y = DailyFootfall, color = Location)) +
  geom_line() +
  geom_vline(xintercept = as_date("2023-03-17"),
             linetype = "dashed", color = "darkgreen", linewidth=0.8) +
  geom_vline(xintercept = as_date("2023-12-25"),
             linetype = "dashed", color = "red", linewidth=0.8) +
  labs(title = "Daily Pedestrian Footfall in Dublin",
       x = "Date",
       y = "Daily Footfall") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  scale_color_brewer(palette = "Set1") +
  annotate("text", x = as_date("2023-03-17"),
           y = max(daily_data$DailyFootfall),
           label = "St. Patrick's Day",
           angle = 90, vjust = -0.5, hjust = 0.8, color="darkgreen") +
  annotate("text", x = as_date("2023-12-25"),
           y = max(daily_data$DailyFootfall),
           label = "Christmas Day",
           angle = 90, vjust = -0.5, hjust = 0.8, color="red")+
  #to center align title
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "top")
```

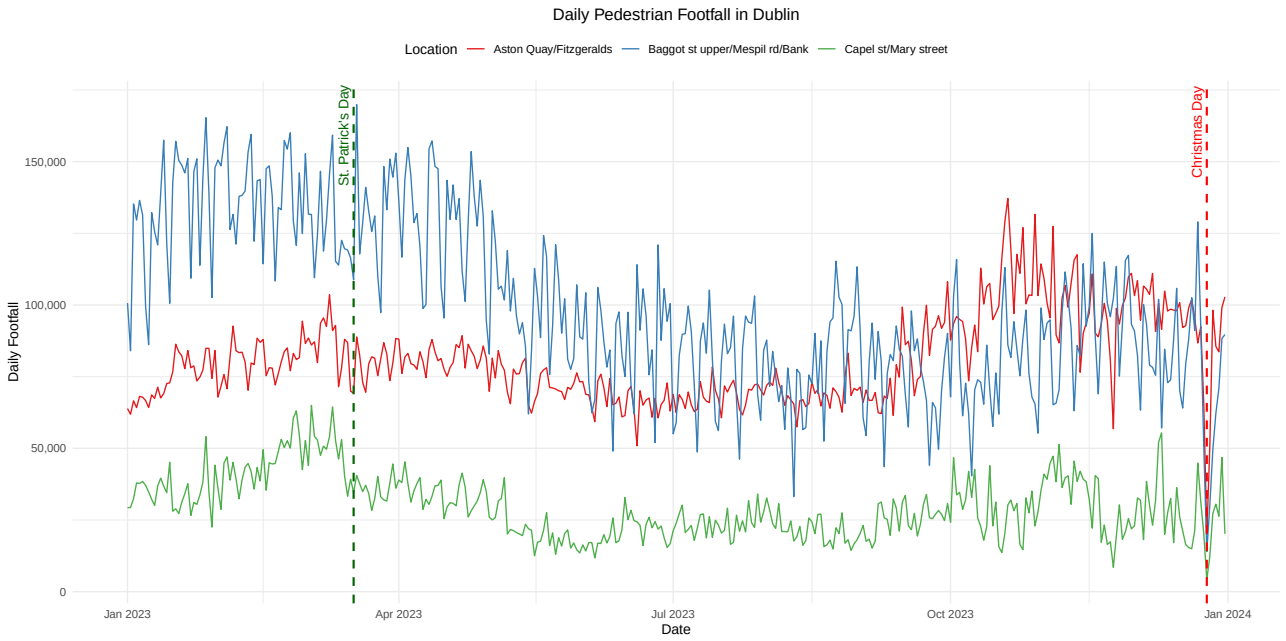

Figure 1: Time series plot for daily pedestrian footfall

I selected "Aston Quay/Fitzgeralds", "Capel st/Mary street", "Baggot st upper/Mespil rd/Bank" as my locations to show a time series plot on daily pedestrian footfall.

To get to daily data I first had to create another data-frame which consists of daily footfall of these three locations and for that I first created a row wise data for three location using *pivot_longer()* and then summarized the sum of data into another data-frame using *summarize()* method.

I then plot out the graph using *geom_line()* and added vertical lines for Christmas and St. Patrick's day using *geom_vline()*.

Observations from the plot:
From the graph, we see that footfall gradually decreases in summer months and then starts increasing in fall months.

We also see that there's a sharp decline in footfalls on Christmas day in the area that I have chosen, which makes sense as people celebrate this day with family and there may be closure of businesses due to holiday.

We see that there's an increase in footfall on St. Patrick's Day, which again makes sense as it is a holiday and people usually celebrate this day by holding parades throughout the city.

2.3 Create a table displaying the minimum and maximum temperature, the mean daily precipitation amount, and the mean daily wind speed by season.

```
#Define seasons
joinedData <- joinedData %>%
  mutate(Season = case_when(
    month %in% c("Dec", "Jan", "Feb") ~ "Winter",
    month %in% c("Mar", "Apr", "May") ~ "Spring",
    month %in% c("Jun", "Jul", "Aug") ~ "Summer",
    month %in% c("Sep", "Oct", "Nov") ~ "Autumn"
  ))


#Summarize data by season
seasonal_summary <- joinedData %>%
  group_by(Season) %>%
  summarize(
    Min_Temp = min(airTemp, na.rm = TRUE),
    Max_Temp = max(airTemp, na.rm = TRUE),
    Mean_Daily_Precipitation = mean(precipitation, na.rm = TRUE),
    Mean_Daily_Wind_Speed = mean(meanWndSpeed, na.rm = TRUE),
```

```
    .groups = 'drop'
  )

#Display the summary table
knitr::kable(seasonal_summary, format = "html",
          digits = 3,
          col.names =   c("Season",
                          "Min Temperature (°C)",
                          "Max Temperature (°C)",
                          "Mean Daily Precipitation (mm)",
                          "Mean Daily Wind Speed (kt)"),
        caption = "Table showing statistics as per seasons")
```

Table 6: Table showing statistics as per seasons

Season	Min Temperature (°C)	Max Temperature (°C)	Mean Daily Precipitation (mm)	Mean Daily Wind Speed (kt)
Autumn	-2.0	25.5	0.149	8.346
Spring	-4.3	19.4	0.105	8.635
Summer	3.8	24.0	0.144	8.502
Winter	-4.3	13.8	0.058	10.496

For displaying table with the values I first had to create a new column called Season where I divided and labelled the seasons based on month of the year(as per hint given in the question).

I then used the *summarize()* method to calculate the min, max and mean of the temperature, precipitation and meanWnd-Speed columns and then used *knitr::kable()* to display the table. I also changed the column names inside the display table to get proper output and rounded off to three decimal places.

Observations from the table:
We see that min temperature for both spring and winter are -4.3°C which can make sense as it is a transition season from winter to summer and it may have been recorded near ending of winter.

We also see that Autumn has the maximum max temperature which can make sense if it is recorded at the end of summer and since it is a transitioning season it may see some changes.

We see that mean daily precipitation in winter is close to zero suggesting no rains in winter but other seasons may receive showers.

We also see that winters are a lot windier than any other seasons with a mean wind speed of 10.49 knots making Irish winter harsher than other seasons(brace yourselves, winter is coming!).

Task3: Creativity

3.1 Monthly plot for pedestrian footfall at Aston Quay

I thought of plotting a monthly plot for pedestrian footfall at Aston Quay as it is one of the busiest streets(near City Center) in Dublin and boy I wasn't wrong!

To plot the graph I first selected month and Aston Quay column and then did a *summarize()* on it getting the sum for daily data and consolidating it for 1 month.

I then plotted the data using *geom_line()* and added points using *geom_point()* to make the y-axis numbers look readable I added commas to it to make sense of the numbers there big using *scale_y_continuous()* method.

```
#filter and summarize data for O'Connell Street
monthly_footfall <- joinedData %>%
  select(month, `Aston Quay/Fitzgeralds`) %>%
  rename(Pedestrian_Count = `Aston Quay/Fitzgeralds`) %>%
  group_by(month) %>%
  summarize(Total_Pedestrian = sum(Pedestrian_Count,
                                   na.rm = TRUE),
                                  .groups = 'drop') %>%
  # Remove any rows where footfall is zero
  filter(Total_Pedestrian > 0)

#create a line plot for monthly footfall
ggplot(monthly_footfall, aes(x = month, y = Total_Pedestrian, group=1)) +
  geom_line(color = "violet", linewidth = 0.8) +
  geom_point(color = "blue", size = 1.8) +
  labs(title = "Monthly Pedestrian Footfall on Aston Quay (2023)",
       x = "Month",
       y = "Total Pedestrian Count") +
  scale_y_continuous(labels = comma)+
  theme_minimal() +
  #rotate x-axis labels for better readability
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+
  #to center align title
  theme(plot.title = element_text(hjust = 0.5))
```

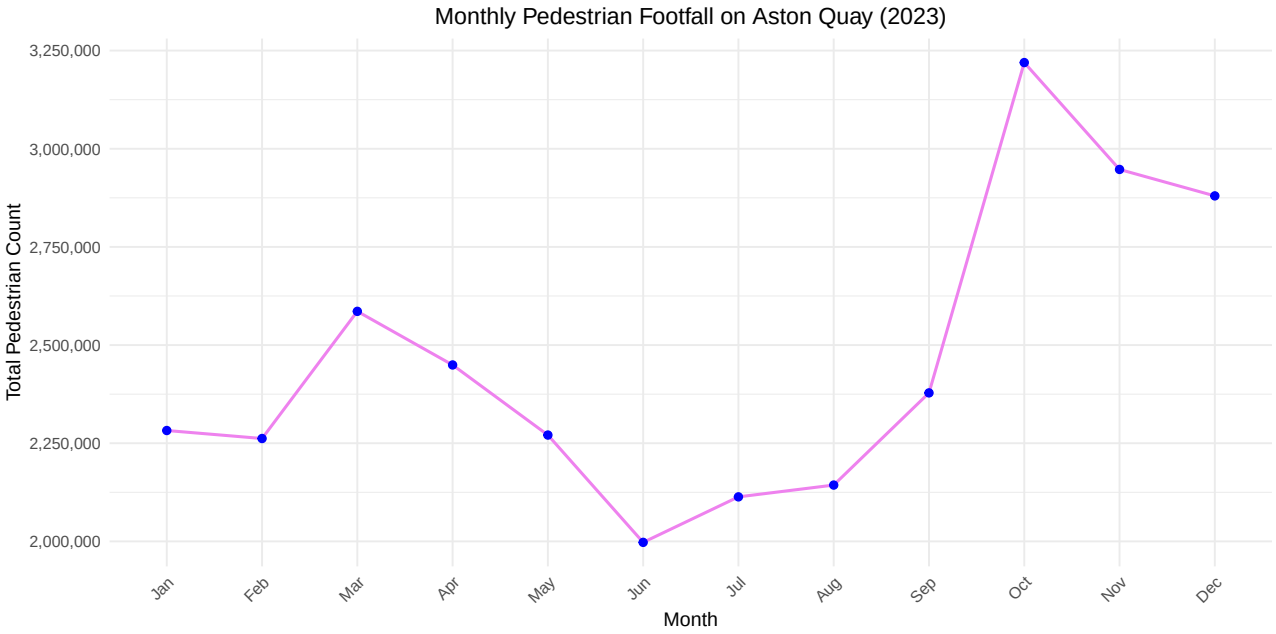

Figure 2: Lineplot for monthly pedestrian footfall at Aston Quay

Observations from the plot:

We see that there is a gradual decline in footfall form March to June in the spring-summer months and then the footfall rises during summer-autumn months.

We see that there is a sharp increase in number of pedestrian at Aston Quay from September to October, this maybe due to the fact that October has Halloween and most of the people celebrate it near the city center(fake Halloween parade news was a nice touch this year). This is just a speculation as other factors may be at play.

The footfall then gradually decreases with the onset of winters.

3.2 Table showing average footfall per location

To create this table I used Pedestrian data-set directly as working with that data-set is easier in this context.

I first calculated the average footfall per location using *summarize()* and the took the mean inside this method.

I then transformed the table using *pivot_longer()* so that I get all rows as my location and footfall as the columns.

I then used *knitr::kable()* to display the table.

```
# Summarize average daily footfall by location
average_footfall_by_location <- pedestrian %>%
  select(-1) %>%
  summarise(across(everything(), ~ mean(.x, na.rm = TRUE)))

# Transform the table to display it row-wise
average_footfall_rowwise <- average_footfall_by_location %>%
  pivot_longer(cols = everything(),
               names_to = "Location",
               values_to = "Average Footfall")

# Display the average footfall table in row-wise format
knitr::kable(average_footfall_rowwise, format = "html",
             caption = "Average footfall based on Location")
```

Table 7: Average footfall based on Location

Location	Average Footfall
Aston Quay/Fitzgeralds	3371.34445

Location	Average Footfall
Baggot st lower/Wilton tce inbound	124.53876
Baggot st upper/Mespil rd/Bank	4159.84302
Capel st/Mary street	1235.84188
College Green/Bank Of Ireland	545.74451
College Green/Church Lane	590.91998
College st/Westmoreland st	602.38577
D'olier st/Burgh Quay	941.58020
Dame Street/Londis	449.77372
Grafton st/Monsoon	1714.57141
Grafton Street / Nassau Street / Suffolk Street	250.19991
Grafton Street/CompuB	1886.63750
Grand Canal st upp/Clanwilliam place	157.28896
Grand Canal st upp/Clanwilliam place/Google	86.40256
Henry Street/Coles Lane/Dunnes	4322.96149
Mary st/Jervis st	447.10389
North Wall Quay/Samuel Beckett bridge East	2121.82738
North Wall Quay/Samuel Beckett bridge West	1579.82521
O'Connell St/Parnell St/AIB	450.66714
O'Connell st/Princes st North	710.05126
Phibsborough Rd/Enniskerry Road	95.79050
Richmond st south/Portabello Harbour inbound	753.05697
Richmond st south/Portabello Harbour outbound	226.69905
Talbot st/Guineys	1288.51879
Westmoreland Street East/Fleet street	1335.53039
Westmoreland Street West/Carrolls	663.33230

Observations from the table:

We see that Henry Street see the most footfall on average at 4322.96 ~ 4323 followed by Baggot Street at 4159.84 ~ 4160 followed by Aston Quay at 3371.34 ~ 3371. This maybe due to the fact that all the above places mentioned are shopping areas and have loads of restaurants as well.

We see that the lowest footfall is at Grand Canal street with an average footfall of 86, this maybe due to the fact that it is a residential/office space area and may not have too much leisurely places to offer.

Overall Dublin looks a bit busy throughout the year, people do move quite a lot inside the city which indicates active lifestyle which is always a good indication healthwise and is even better for businesses to gauge where to open a restaurant or a shop or an office space for more profitability.