Colchester Weather(2023) - Data Visualization

Musaddik Maulavi

2024-04-16

Introduction

The “temp2023.csv” dataset includes daily climatic data gathered from a weather station near Colchester in the year 2023. This dataset contains 365 observations and 18 variables, offering a thorough picture of the weather conditions seen during the year. Each observation is for a single day, and the variables include average temperature, maximum and lowest temperature, wind speed and direction, precipitation, cloud cover, sunlight duration, visibility, and more.

The dataset provides insightful information on the seasonal variations and trends of the weather in the Colchester region. We may learn more about seasonal patterns, weather extremes, and possible relationships between other weather variables by examining this data. Applications such as agriculture, urban planning, energy management, and climate research can all benefit greatly from such insights.

We will thoroughly analyze the “temp2023.csv” file and look at the many weather factors that are recorded all year long in this report. Finding significant patterns, trends, and links in the data is our goal, and we plan to do this using interpretative analysis, data visualization, and descriptive statistics. Furthermore, we’ll make use of sophisticated graphical and statistical approaches to present a thorough summary of the climatic conditions in Colchester.

Dataset

```
data <- read.csv('temp2023.csv')
# head(data)
names(data)
```

```
##  [1] "station_ID"      "Date"            "TemperatureCAvg" "TemperatureCMax"
##  [5] "TemperatureCMin" "TdAvgC"          "HrAvg"           "WindkmhDir"
##  [9] "WindkmhInt"      "WindkmhGust"     "PresslevHp"      "Precmm"
## [13] "TotClOct"        "lowClOct"        "SunD1h"          "VisKm"
## [17] "PreselevHp"      "SnowDepcm"
```

```
str(data)
```

```
## 'data.frame':    365 obs. of  18 variables:
##  $ station_ID     : int  3590 3590 3590 3590 3590 3590 3590 3590 3590 3590 ...
##  $ Date           : chr  "31/12/2023" "30/12/2023" "29/12/2023" "28/12/2023" ...
##  $ TemperatureCAvg: num  8.7 6.6 9.9 9.9 5.8 9.8 12.5 10 9.6 10 ...
##  $ TemperatureCMax: num  10.6 9.7 11.4 11.5 10.6 12.7 14.3 12 10.8 12.6 ...
##  $ TemperatureCMin: num  4.4 4.4 6.9 4 3.9 6.3 9.5 8.4 8.1 8.1 ...
##  $ TdAvgC         : num  7.2 4.2 6 7.5 3.7 7.6 10.1 7 6.5 6.2 ...
##  $ HrAvg          : num  89.6 85.5 77.2 84.6 86.4 86.9 85.3 81.5 81.2 78.2 ...
##  $ WindkmhDir     : chr  "S" "WSW" "SW" "SSW" ...
##  $ WindkmhInt     : num  25 22.7 32.8 32.2 13.2 23.5 34.1 32.7 34.1 37.5 ...
##  $ WindkmhGust    : num  63 50 61.2 70.4 37.1 46.3 72.3 61.2 68.6 77.8 ...
##  $ PresslevHp     : num  999 1007 1004 1003 1016 ...
##  $ Precmm         : num  6.2 0.4 0.8 2.8 2 4.4 0.8 0.8 0 2 ...
##  $ TotClOct       : num  8 4.6 6.5 6.8 4 6.5 7.8 5 8 7.5 ...
##  $ lowClOct       : num  8 6.5 6.7 7.1 6.9 7.4 7.8 6.7 8 7.5 ...
##  $ SunD1h         : num  0 1.1 0.1 0 3.2 0 0 2.9 0 1.4 ...
##  $ VisKm          : num  26.3 48.3 26.7 25.1 30.1 45.8 61.8 72.9 69.4 34.3 ...
##  $ PreselevHp     : logi  NA NA NA NA NA NA ...
##  $ SnowDepcm      : int  NA NA NA NA NA NA NA NA NA NA ...
```

Data Cleaning and pre processing

Before performing Data Visualization I have cleaned the dataset. Firstly I have converted the Date column in Date format. Then I have checked the NA values in each column.

```
# Converting to Date format
data$Date <- as.Date(data$Date, format = "%d/%m/%Y")
```

```
# Counting NA values in each column
na_counts <- colSums(is.na(data))

# Display NA counts
print(na_counts)
```

```
##      station_ID            Date TemperatureCAvg TemperatureCMax TemperatureCMin
##               0               0               0               0               0
##          TdAvgC           HrAvg      WindkmhDir      WindkmhInt     WindkmhGust
##               0               0               0               0               0
##      PresslevHp          Precmm        TotClOct        lowClOct          SunD1h
##               0              27               0              13              82
##           VisKm      PreselevHp       SnowDepcm
##               0             365             364
```

I have separated the date column in Date Month Year separately. So that the month column can be used for Data Visualization easily.

```
final_proj_data <- data[,1:16]
head(final_proj_data)
```

```
##   station_ID       Date TemperatureCAvg TemperatureCMax TemperatureCMin TdAvgC
## 1       3590 2023-12-31             8.7            10.6             4.4    7.2
## 2       3590 2023-12-30             6.6             9.7             4.4    4.2
## 3       3590 2023-12-29             9.9            11.4             6.9    6.0
## 4       3590 2023-12-28             9.9            11.5             4.0    7.5
## 5       3590 2023-12-27             5.8            10.6             3.9    3.7
## 6       3590 2023-12-26             9.8            12.7             6.3    7.6
##   HrAvg WindkmhDir WindkmhInt WindkmhGust PresslevHp Precmm TotClOct lowClOct
## 1  89.6          S       25.0        63.0      999.0    6.2      8.0      8.0
## 2  85.5        WSW       22.7        50.0     1006.9    0.4      4.6      6.5
## 3  77.2         SW       32.8        61.2     1003.6    0.8      6.5      6.7
## 4  84.6        SSW       32.2        70.4     1003.2    2.8      6.8      7.1
## 5  86.4         SW       13.2        37.1     1016.4    2.0      4.0      6.9
## 6  86.9        WSW       23.5        46.3     1006.2    4.4      6.5      7.4
##   SunD1h VisKm
## 1    0.0  26.3
## 2    1.1  48.3
## 3    0.1  26.7
## 4    0.0  25.1
## 5    3.2  30.1
## 6    0.0  45.8
```

```
day <- day(data$Date)
month <- month(data$Date)
year <- year(data$Date)

# Add the split date components as new columns to the existing data frame
final_proj_data$Day <- as.numeric(day)
final_proj_data$Month <- as.numeric(month)
final_proj_data$Year <- as.numeric(year)

final_proj_data <- final_proj_data[,-2]
```

```
str(final_proj_data)
```

```
## 'data.frame':    365 obs. of  18 variables:
##  $ station_ID     : int  3590 3590 3590 3590 3590 3590 3590 3590 3590 3590 ...
##  $ TemperatureCAvg: num  8.7 6.6 9.9 9.9 5.8 9.8 12.5 10 9.6 10 ...
##  $ TemperatureCMax: num  10.6 9.7 11.4 11.5 10.6 12.7 14.3 12 10.8 12.6 ...
##  $ TemperatureCMin: num  4.4 4.4 6.9 4 3.9 6.3 9.5 8.4 8.1 8.1 ...
##  $ TdAvgC         : num  7.2 4.2 6 7.5 3.7 7.6 10.1 7 6.5 6.2 ...
##  $ HrAvg          : num  89.6 85.5 77.2 84.6 86.4 86.9 85.3 81.5 81.2 78.2 ...
##  $ WindkmhDir     : chr  "S" "WSW" "SW" "SSW" ...
##  $ WindkmhInt     : num  25 22.7 32.8 32.2 13.2 23.5 34.1 32.7 34.1 37.5 ...
##  $ WindkmhGust    : num  63 50 61.2 70.4 37.1 46.3 72.3 61.2 68.6 77.8 ...
##  $ PresslevHp     : num  999 1007 1004 1003 1016 ...
##  $ Precmm         : num  6.2 0.4 0.8 2.8 2 4.4 0.8 0.8 0 2 ...
##  $ TotClOct       : num  8 4.6 6.5 6.8 4 6.5 7.8 5 8 7.5 ...
##  $ lowClOct       : num  8 6.5 6.7 7.1 6.9 7.4 7.8 6.7 8 7.5 ...
##  $ SunD1h         : num  0 1.1 0.1 0 3.2 0 0 2.9 0 1.4 ...
##  $ VisKm          : num  26.3 48.3 26.7 25.1 30.1 45.8 61.8 72.9 69.4 34.3 ...
##  $ Day            : num  31 30 29 28 27 26 25 24 23 22 ...
##  $ Month          : num  12 12 12 12 12 12 12 12 12 12 ...
##  $ Year           : num  2023 2023 2023 2023 2023 ...
```

Data Visualization

1. Table

```
# Table
one_way_table <- table(final_proj_data$WindkmhDir)
print(one_way_table)
```

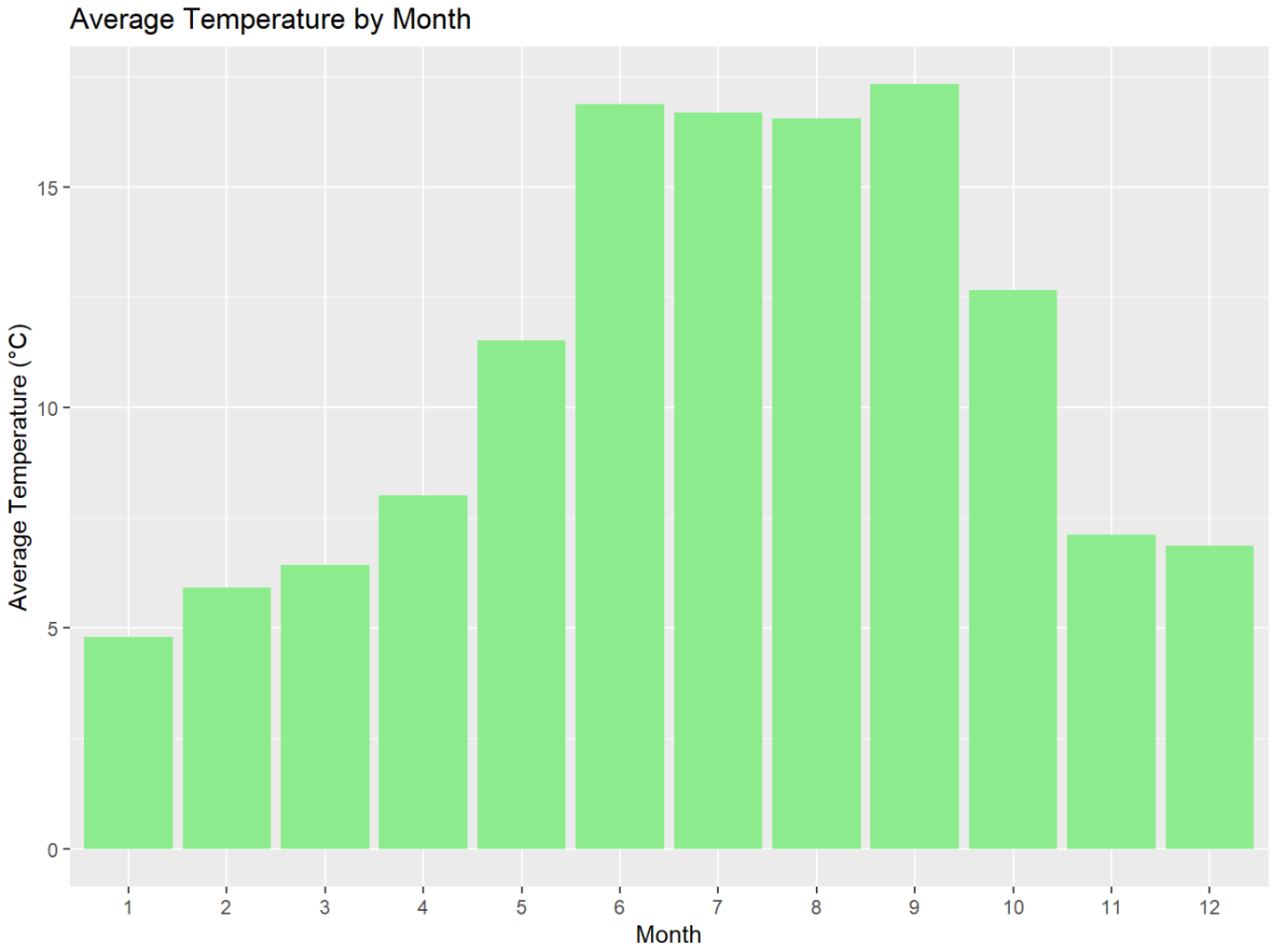
```
##
##   E ENE ESE   N  NE NNE NNW  NW   S  SE SSE SSW  SW   W WNW WSW
##  10  20  11  18  22  15  13  15  26   6  14  41  49  34  13  58
```

The provided one-way table represents the frequency of wind directions observed in the climate data collected from a weather station close to Colchester in 2023. It categorizes the wind direction into 16 different categories, including “E” (East), “ENE” (East-Northeast), “ESE” (East-Southeast), “N” (North), “NE” (Northeast), “NNE” (North-Northeast), “NNW” (North-Northwest), “NW” (Northwest), “S” (South), “SE” (Southeast), “SSE” (South-Southeast), “SSW” (South-Southwest), “SW” (Southwest), “W” (West), “WNW” (West-Northwest), and “WSW” (West-Southwest).

Every category denotes a distinct direction in which the wind is blowing, and the dataset records the related frequency of occurrences. As an illustration, there are 10 observations that have an easterly wind direction and 58 observations that have a west-southwest wind direction.

2. Bar Plot

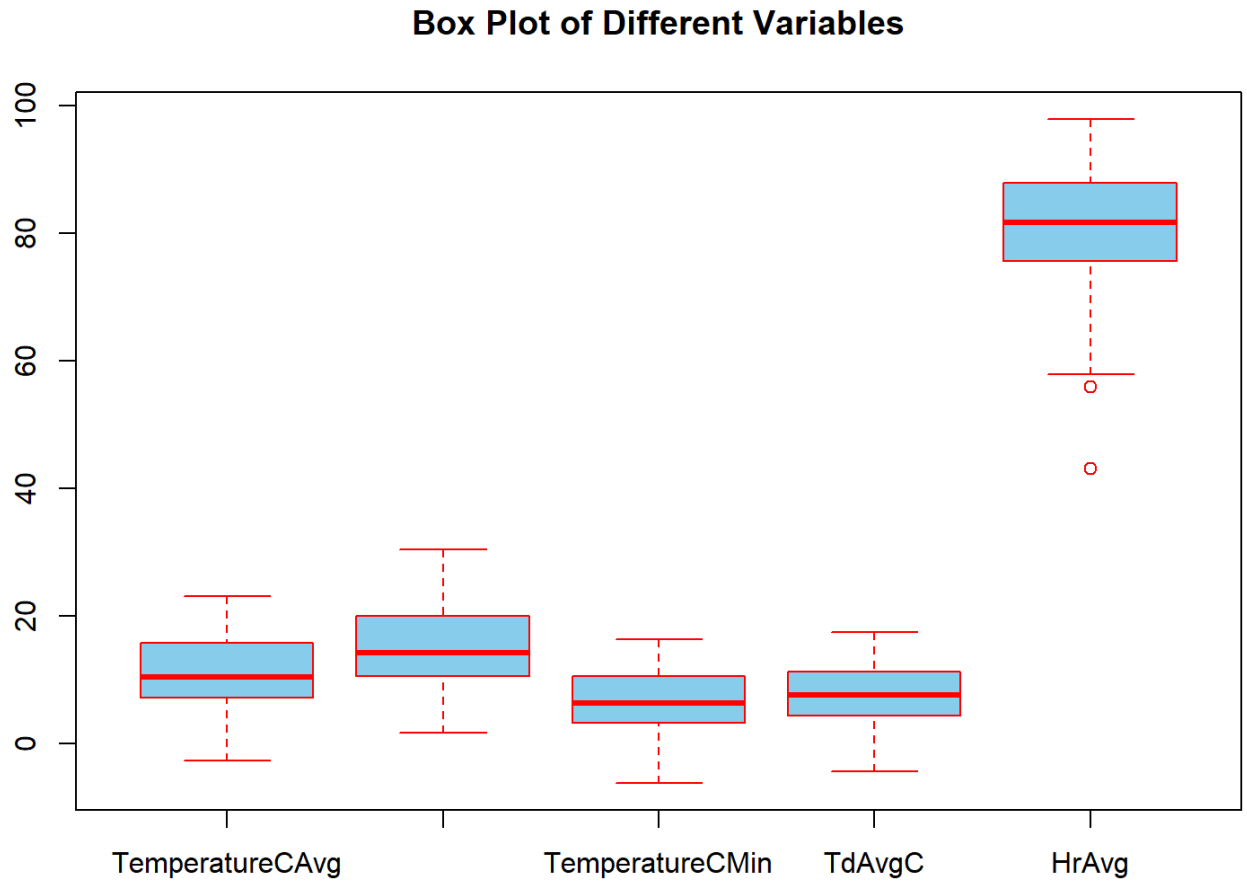
```
# Bar plot of average temperature by month
ggplot(final_proj_data, aes(x = factor(Month), y = TemperatureCAvg)) +
  geom_bar(stat = "summary", fun = "mean", fill = "lightgreen") +
  labs(title = "Average Temperature by Month", x = "Month", y = "Average Temperature (°C)")
```



The average temperature variation throughout the course of the year is depicted in a bar plot. The mean temperature recorded for a given month is represented by each bar. The months are shown on the x-axis, while the average temperature in degrees Celsius is shown on the y-axis. The figure makes it easy to compare annual temperature trends and highlights any patterns or seasonal fluctuations. One potential use for it may be to shed light on seasonal variations in temperature, such as summertime warmth and wintertime cold. The average temperature readings are shown by the light green color fills, which make the information more aesthetically pleasing and understandable.

3. Box Plot

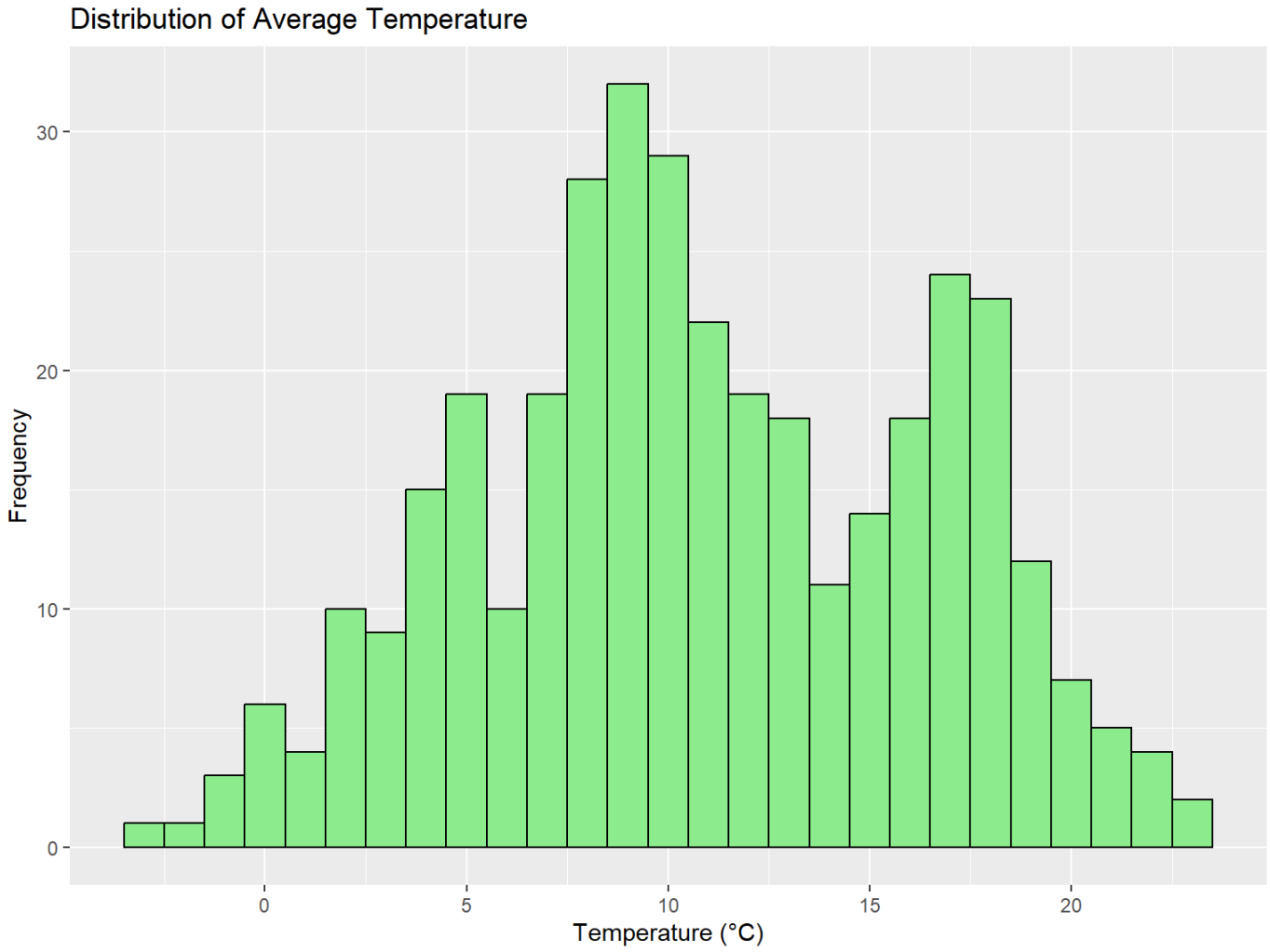
```
boxplot(final_proj_data[, 2:6], col = "skyblue", border = "red",
        main = "Box Plot of Different Variables")
```



The distribution of several variables in the dataset is shown visually by the box plot. A given variable's distribution is shown by each box, and the median value is indicated by the middle line. The box covers the interquartile range (IQR) and goes from the first quartile (Q1) to the third quartile (Q3). The whiskers reach the greatest and lowest values within this range, or 1.5 times the IQR above and below the upper and lower quartiles, respectively. The variables in this figure are identified by varying shades of blue, and for improved visibility, the box edges are highlighted in red. This color scheme provides important information about the distributional properties of the variables and improves the plot's clarity and visual appeal. It also makes it simpler to distinguish between the variables.

4. Histogram

```
# Histogram of average temperature
ggplot(final_proj_data, aes(x = TemperatureCAvg)) +
  geom_histogram(binwidth = 1, fill = "lightgreen", color = "black") +
  labs(title = "Distribution of Average Temperature", x = "Temperature (°C)", y = "Frequenc
y")
```

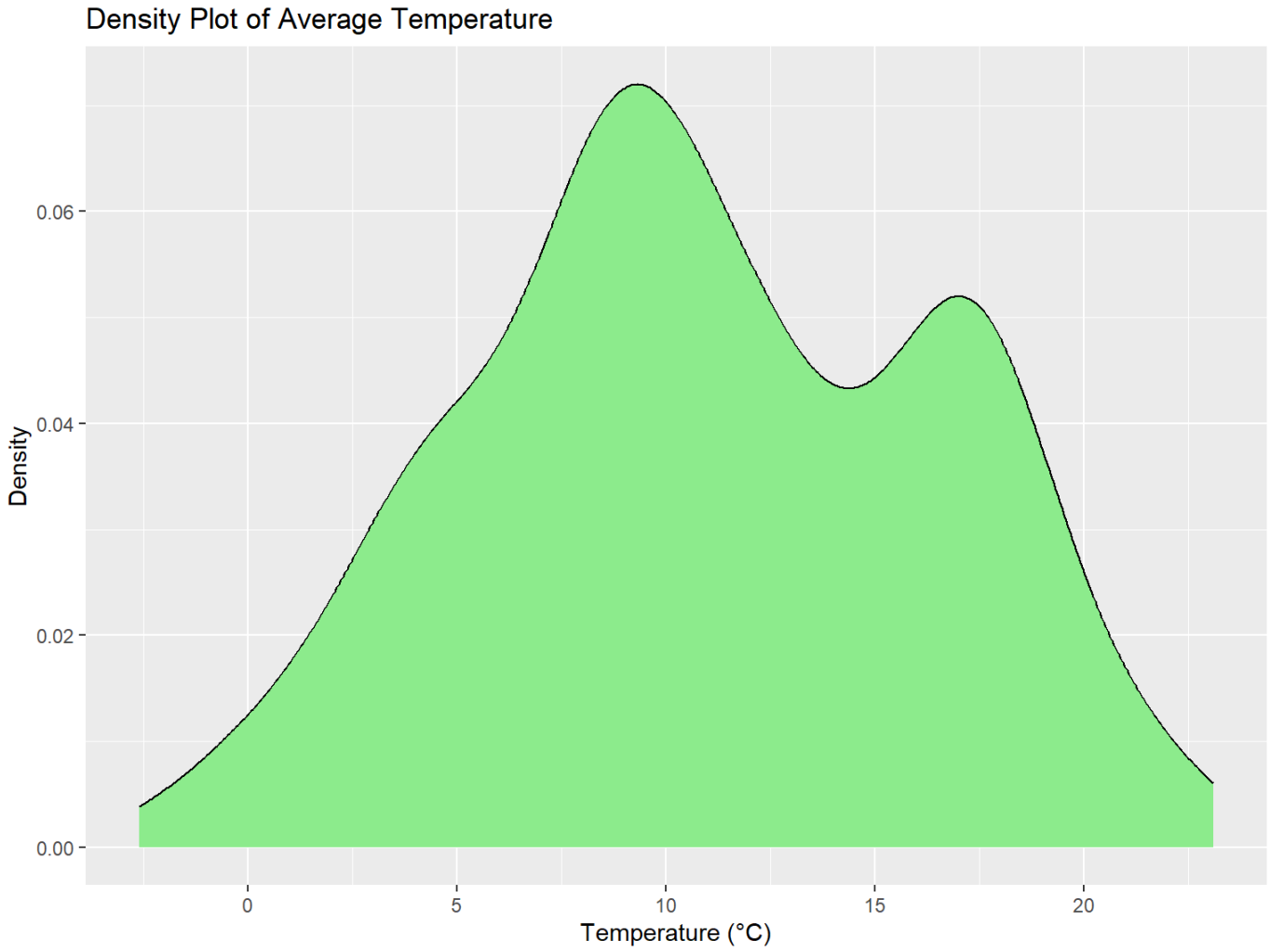


The dataset's average temperature distribution is shown graphically via the histogram. The height of each bar shows the frequency of temperatures falling inside its respective range, and each bar represents a range of temperatures (binwidth = 1) along the x-axis.

Because the histogram in this plot is filled with a light green color, it stands out visually and is simple to identify from other parts. The bars' black edge makes them easier to see and clearly delineates the borders of each bin.

5. Density Plot

```
# Density plot of average temperature
ggplot(final_proj_data, aes(x = TemperatureCAvg)) +
  geom_density(fill = "lightgreen") +
  labs(title = "Density Plot of Average Temperature", x = "Temperature (°C)", y = "Density")
```



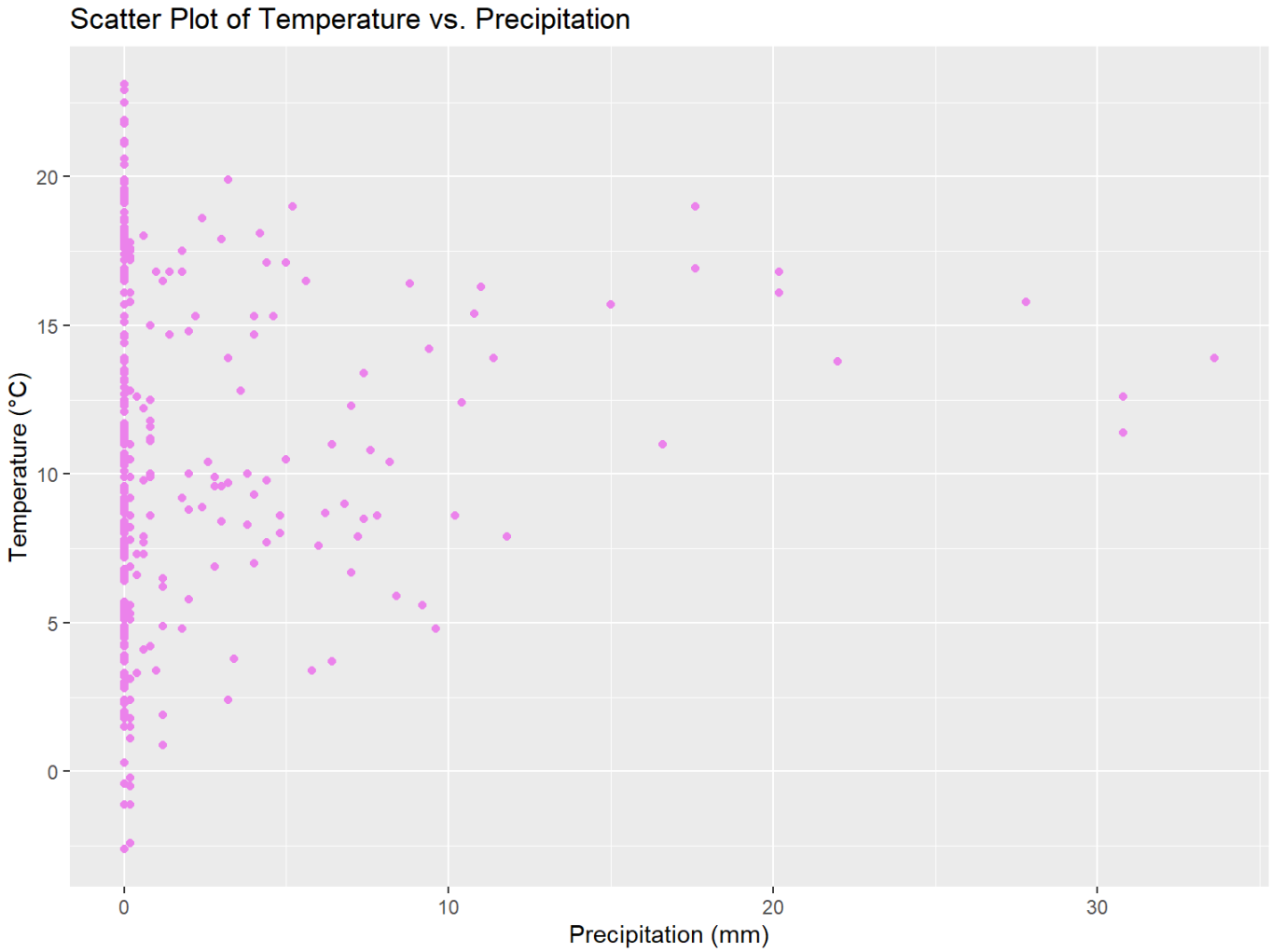
By calculating the probability density function, the density plot makes the distribution of average temperatures in the dataset visually appealing. It offers a smooth curve that depicts the data's distributional form.

Higher peaks denote areas of higher density in this figure, where the probability density is represented by the area under the curve. The curve's fill color is bright green, which enhances its visual attractiveness and distinguishability. The title "Density Plot of Average Temperature" clearly indicates the purpose of the plot, while the labels on the x and y axes provide context for interpreting the data. Overall, this density plot effectively communicates the distributional characteristics of the average temperature variable in the dataset.

6. Scatter Plot

```
ggplot(final_proj_data, aes(x = Precmm, y = TemperatureCAvg)) +
  geom_point(color = "violet") +
  labs(title = "Scatter Plot of Temperature vs. Precipitation", x = "Precipitation (mm)", y =
"Temperature (°C)")
```

```
## Warning: Removed 27 rows containing missing values (`geom_point()`).
```



The relationship between average temperature and precipitation is shown visually by the scatter plot. For each individual dataset observation, the combination of temperature and precipitation is represented as a point on the plot.

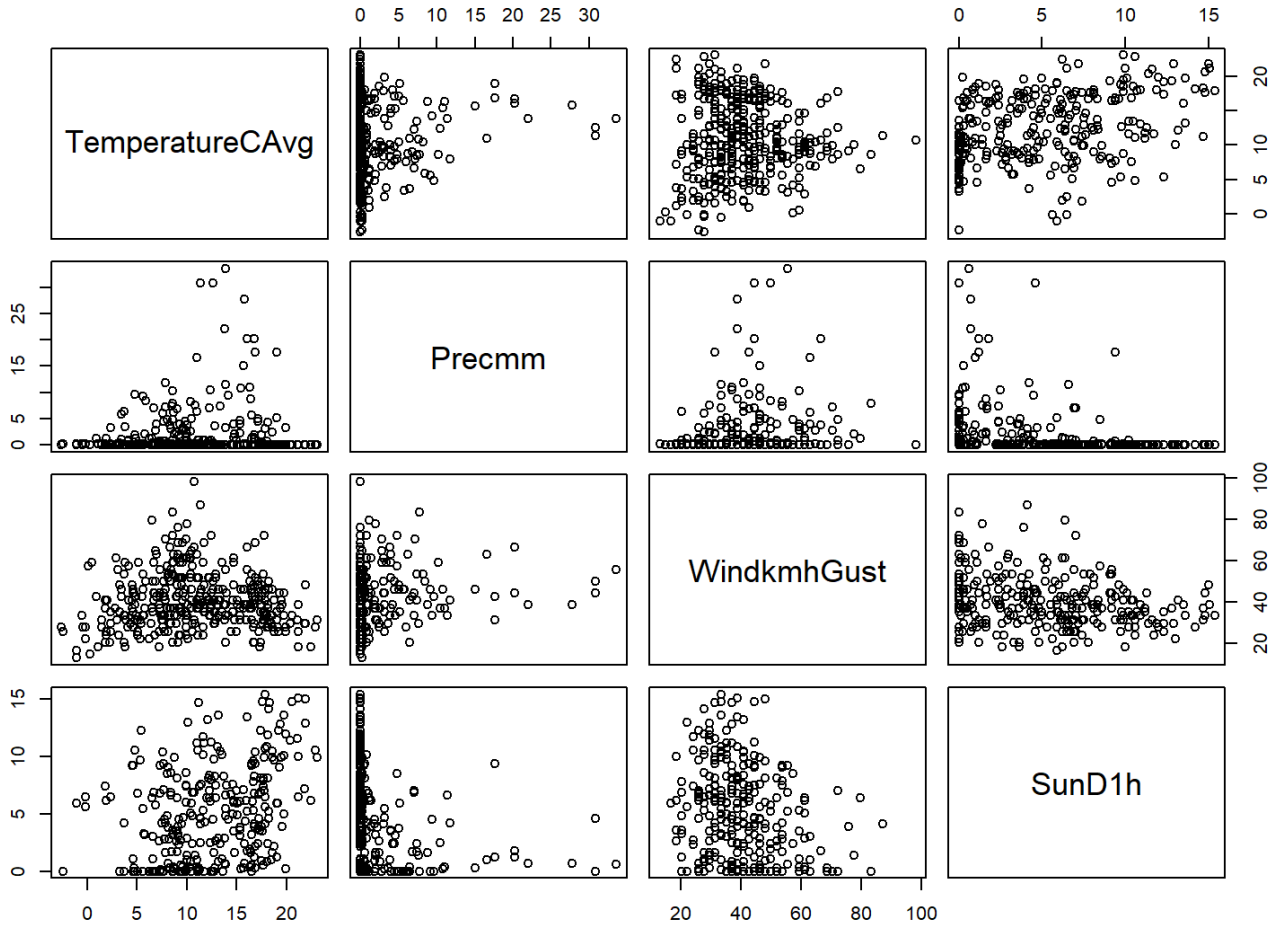
The above graph displays temperature in degrees Celsius (°C) on the y-axis and precipitation in millimeters (mm) on the x-axis. The blue markings on each point contrast sharply with the white background.

The title “Scatter Plot of Temperature vs. Precipitation” clearly indicates the purpose of the plot, while the labels on the x and y axes provide context for interpreting the data.

This scatter plot allows us to visually assess any potential relationship or pattern between temperature and precipitation variables.

7. Pair PLot

```
# Pair plot of selected variables
selected_vars <- c("TemperatureCAvg", "Precmm", "WindkmhGust", "SunD1h")
pairs(final_proj_data[selected_vars])
```



Using scatter plots and diagonal histograms for each pair of variables, the pair plot illustrates the relationships between several variables in a dataset. The histograms display the distribution of each individual variable, while the scatter plots illustrate the connection between two variables.

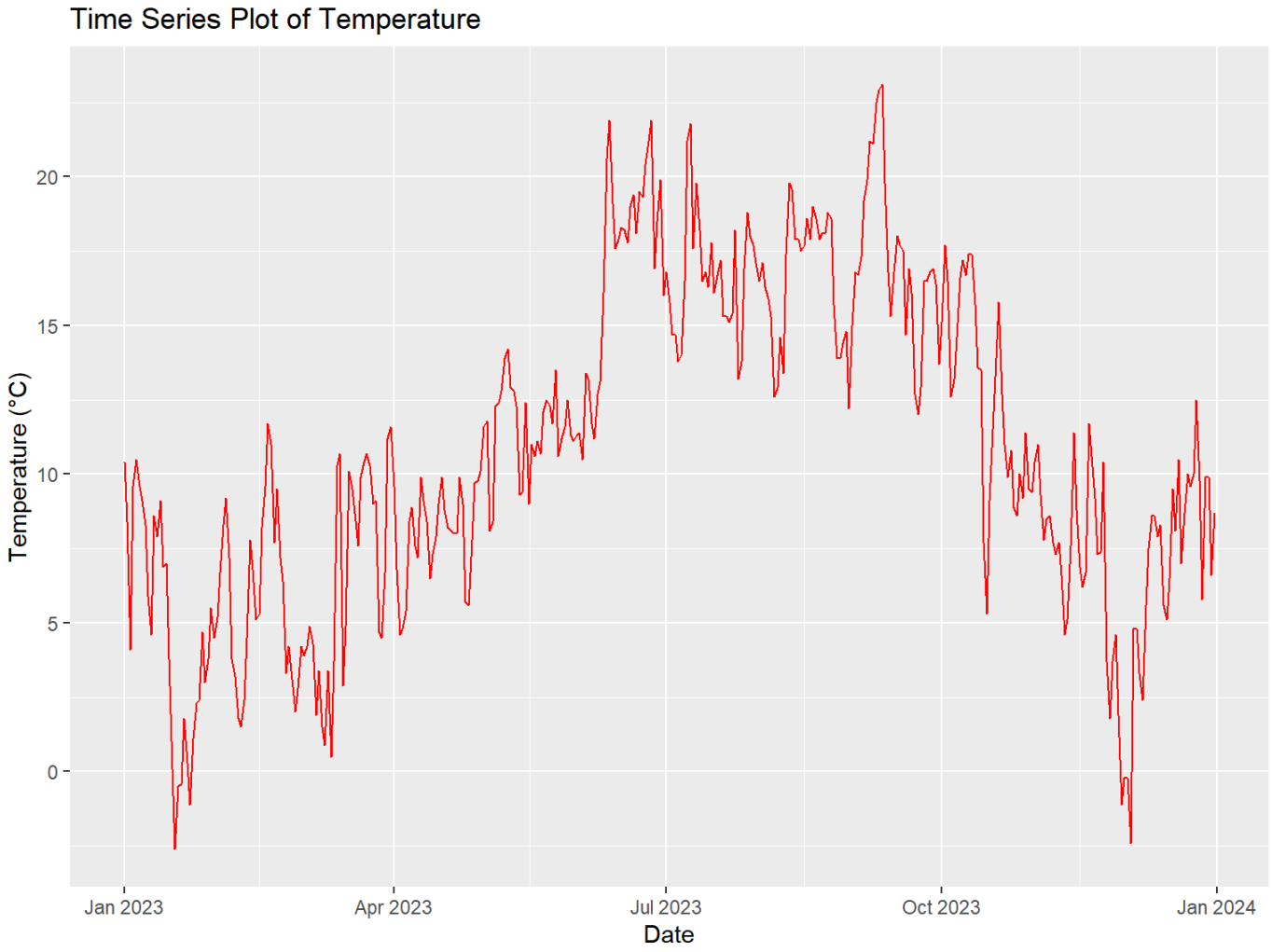
The variables “TemperatureCAvg”, “Precmm”, “WindkmhGust”, and “SunD1h” are the ones that were chosen for this pair plot. The average temperature, precipitation, wind gusts, and hours of sunlight are represented by these variables, in that order.

The pair plot’s diagonal shows the distribution of values for each variable as represented by histograms. Scatter plots that show the associations between pairs of variables are called off-diagonal plots. An observation from the dataset is represented by each point in the scatter plots.

We can visually examine the distributions and interactions between the chosen variables with this pair plot, which offers insights into possible patterns or correlations in the data.

8. Time Series Plot

```
# Time series plot of temperature
ggplot(data, aes(x = Date, y = TemperatureCAvg)) +
  geom_line(color = "red") +
  labs(title = "Time Series Plot of Temperature", x = "Date", y = "Temperature (°C)")
```



The average temperature's fluctuation throughout year is seen in the time series plot. The average temperature measured on a particular day is represented by each point on the plot. The dates are shown on the x-axis, while the matching average temperature values in degrees Celsius are displayed on the y-axis.

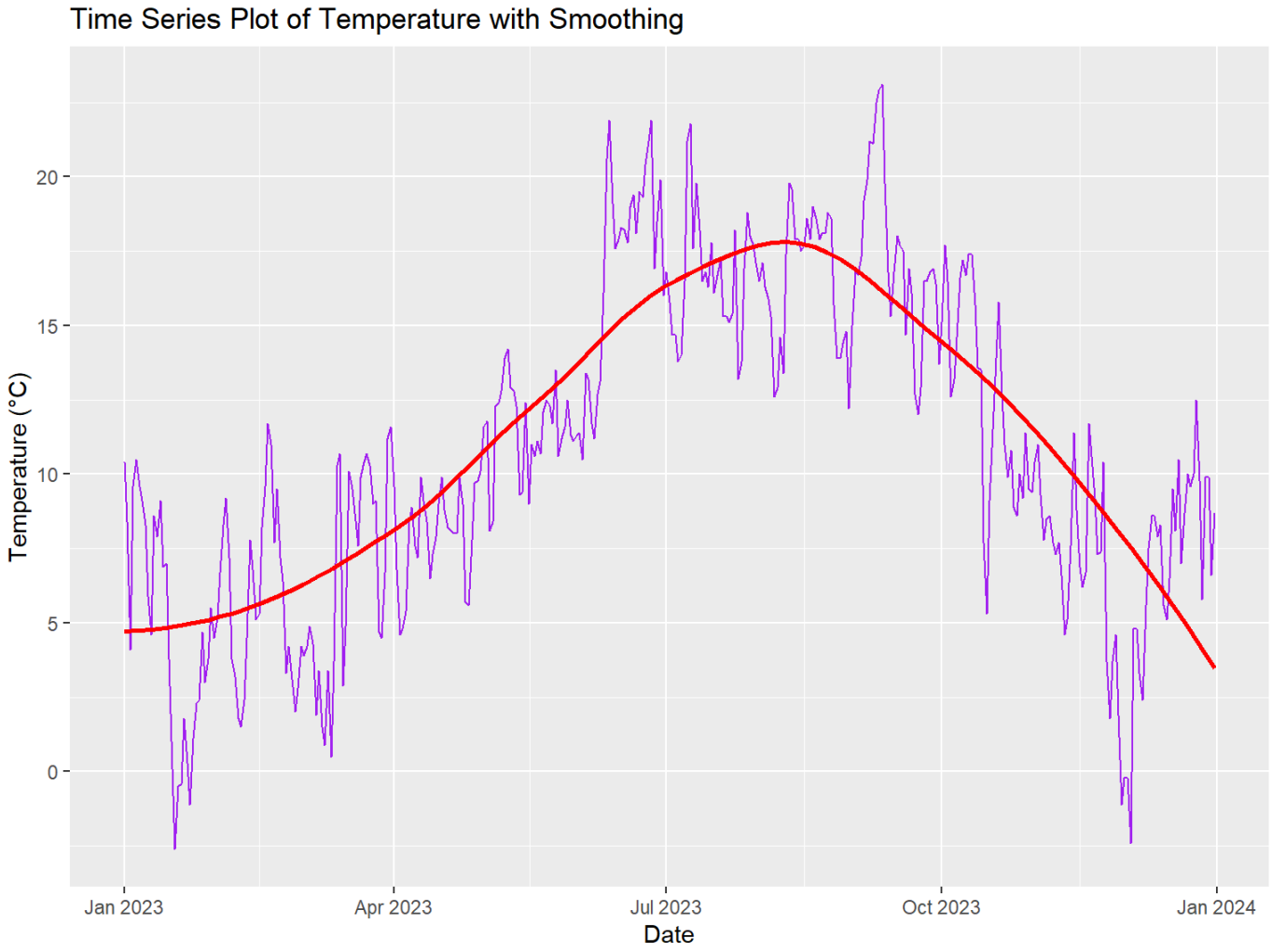
The general trend in temperature variations during the dataset's time period is depicted visually by the red line that connects the dots. We can see any patterns, trends, or seasonality in the changes in temperature over time with this map.

By analyzing this time series plot, we can gain insights into the temperature patterns and fluctuations, which can be valuable for understanding climate dynamics and making informed decisions in various applications, such as agriculture, energy management, and environmental monitoring.

9. Smoothning Plot

```
# Time series plot of temperature with smoothing
ggplot(data, aes(x = Date, y = TemperatureCAvg)) +
  geom_line(color = "purple") +
  geom_smooth(method = "loess", se = FALSE, color = "red") +
  labs(title = "Time Series Plot of Temperature with Smoothing", x = "Date", y = "Temperature
(°C)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Like the previous graph, the time series plot with smoothing shows the change in the average temperature over time. To offer a clearer representation of the underlying trend in temperature changes, a smoothing curve has been applied to this graphic.

The raw data points, or actual recorded average temperature readings on certain days, are shown by the blue line. By minimizing noise and highlighting the underlying pattern, the red smoothing curve—which was produced using the LOESS (Locally Weighted Scatterplot Smoothing) method—makes it easier to see the general trend in temperature changes.

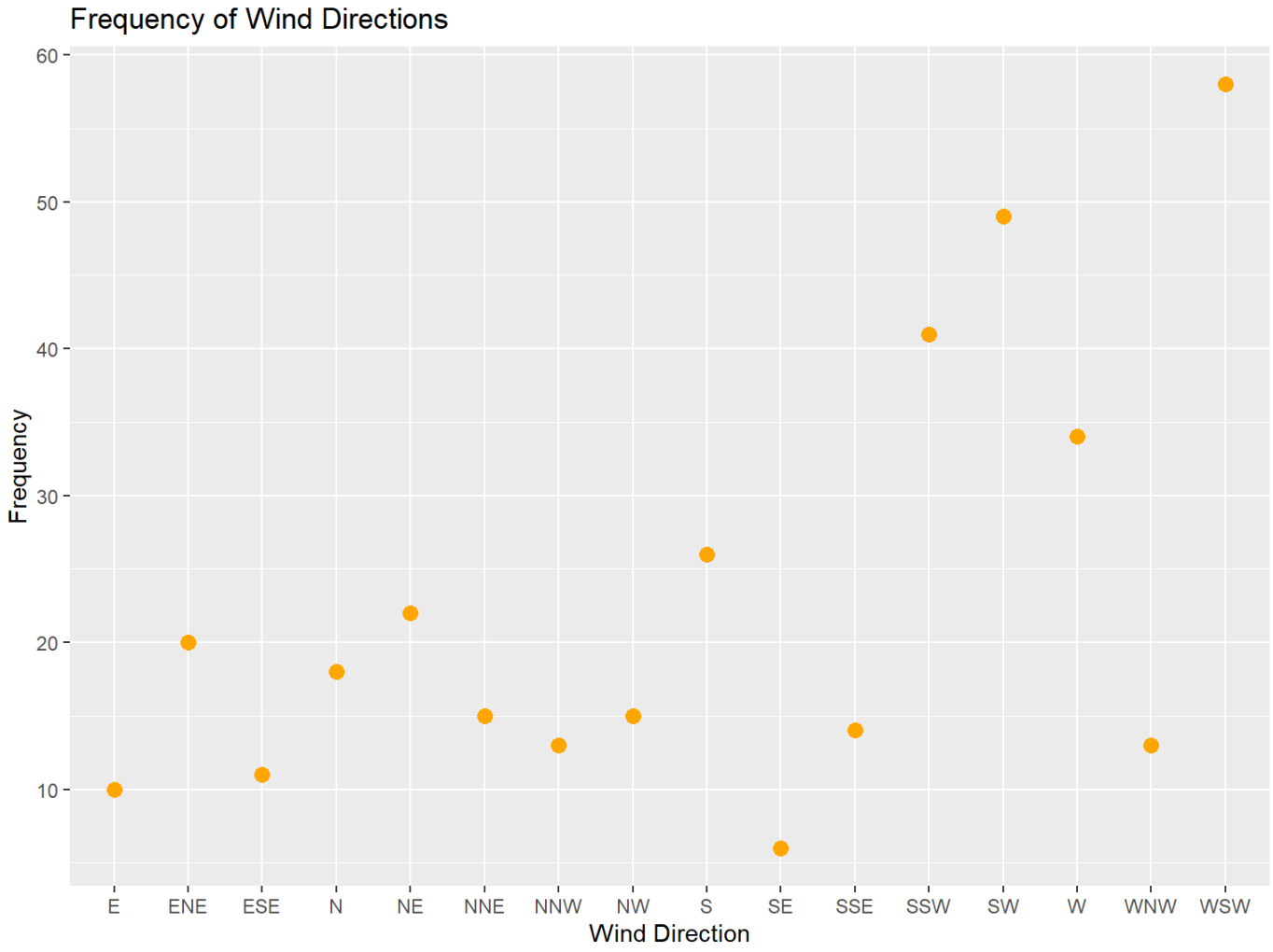
This display allows for a better understanding of the long-term temperature trends while maintaining the individual data points for reference since it combines the raw data points with the smoothed curve. Finding small variations or patterns in temperature over time—which might not be immediately obvious from the raw data alone—can be made easier with the help of this graphic.

10. Dot Plot

```
# Counting the frequency of each wind direction
wind_direction_freq <- table(final_proj_data$WindkmhDir)

# Converting the table to a data frame
wind_direction_df <- data.frame(Direction = names(wind_direction_freq),
                                Frequency = as.numeric(wind_direction_freq))

# Creating a dot plot
ggplot(wind_direction_df, aes(x = Direction, y = Frequency)) +
  geom_point(size = 3, color = "orange") +
  labs(title = "Frequency of Wind Directions", x = "Wind Direction", y = "Frequency")
```



The frequency distribution of the wind directions found in the dataset is depicted in this dot plot. Every dot is a representative of a particular wind direction; the direction is indicated by its location on the x-axis, and the frequency of occurrence is shown by its vertical position.

The many wind directions, including North (N), South (S), East (E), West (W), and their variants, are shown on the x-axis. The frequency of each wind direction is represented by the y-axis, which shows how frequently each direction occurs in the dataset.

For maximum regularity, the dots are uniformly sized and have an orange tint to enhance visibility. Viewers may determine which wind directions are most common and evaluate any patterns or trends in the wind direction data by looking at the distribution of dots along the x-axis.

11. Other Plots

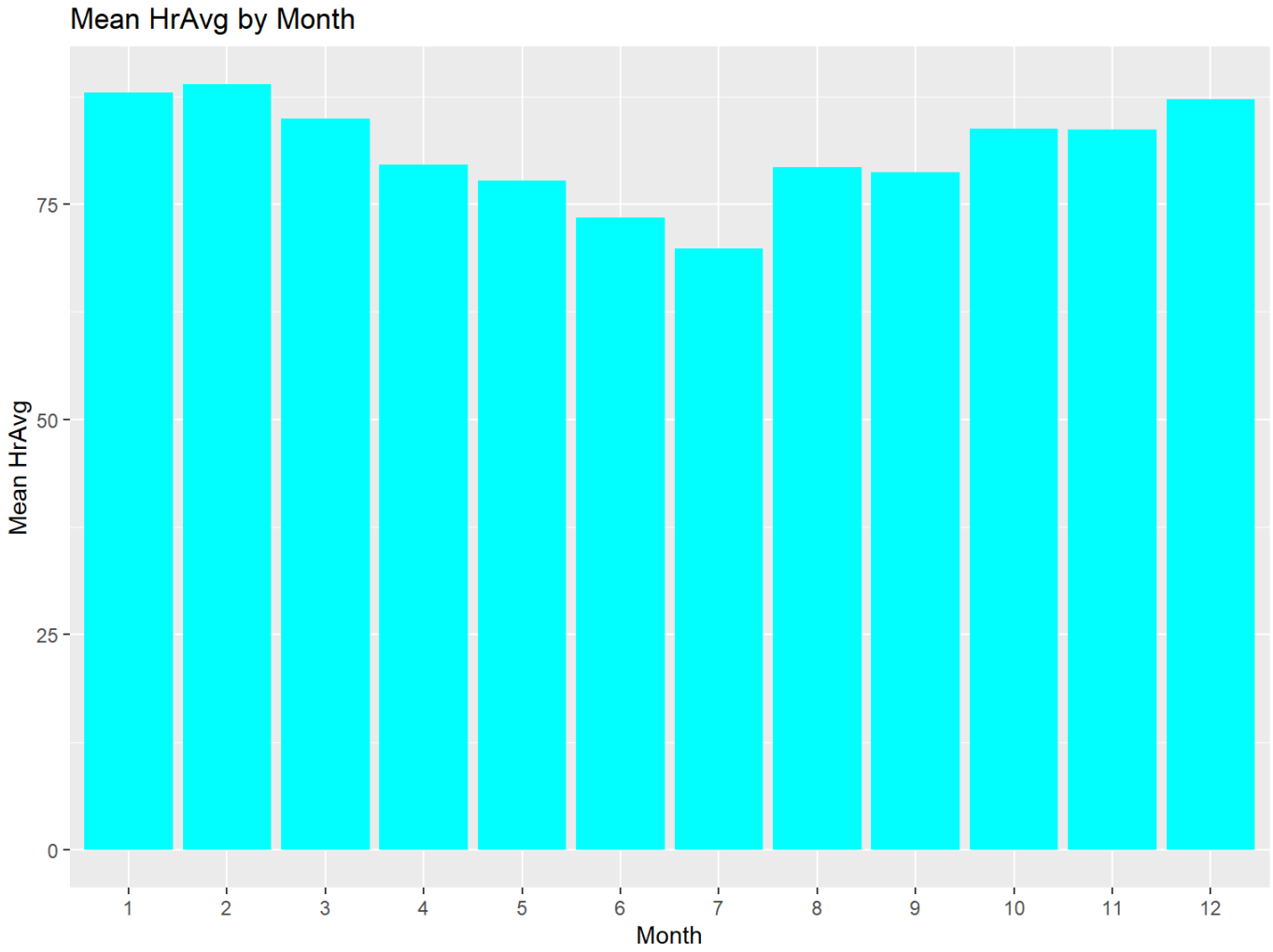
i)

```
bar_plot_table <- final_proj_data[,c(6,17)]
# bar_plot_table

bar_plot_table$Month <- factor(bar_plot_table$Month, levels = unique(bar_plot_table$Month))

mean_hravg <- bar_plot_table %>%
  group_by(Month) %>%
  summarise(mean_HrAvg = mean(HrAvg, na.rm = TRUE))

ggplot(mean_hravg, aes(x = Month, y = mean_HrAvg)) +
  geom_bar(stat = "identity", fill = "cyan") +
  scale_x_discrete(labels = 1:12) +  # Set x-axis labels to integers 1 through 12
  labs(x = "Month", y = "Mean HrAvg", title = "Mean HrAvg by Month")
```



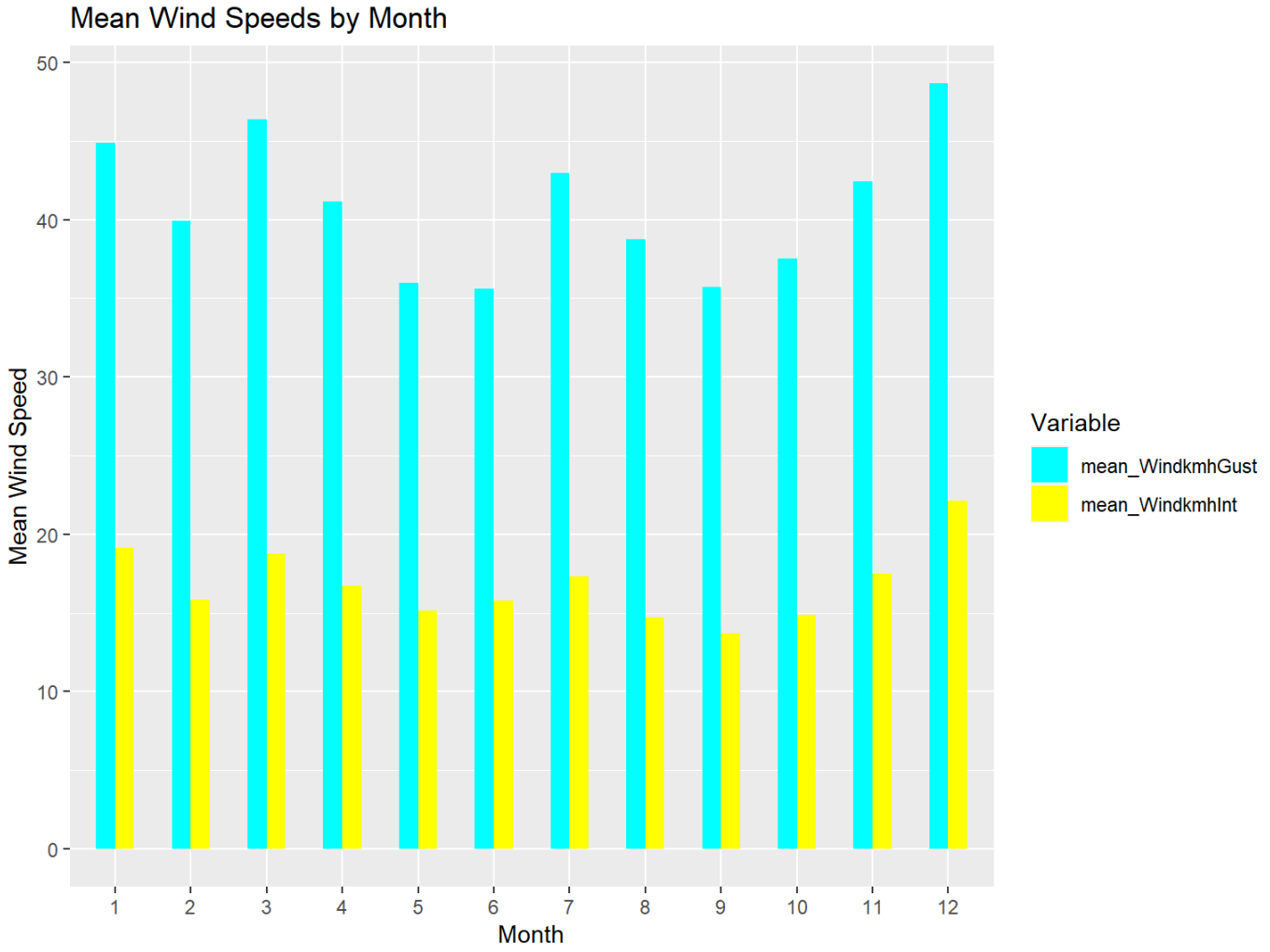
The mean HrAvg (average hour) for every month of the year is displayed in this bar graphic. Every bar shows the average HrAvg value for a particular month, with the months of January through December shown on the x-axis.

The typical hourly rate for each month is shown visually by the cyan-colored bars, which show the HrAvg values. This graphic makes it possible to identify any seasonal patterns or year-over-year changes in HrAvg. For example, depending on the context of the dataset, peaks or dips in the mean HrAvg over various months may reflect times of greater or decreased activity.

```
wind_data <- final_proj_data %>%
  select(WindkmhInt, WindkmhGust, Month) %>%
  group_by(Month) %>%
  summarise(mean_WindkmhInt = mean(WindkmhInt, na.rm = TRUE),
            mean_WindkmhGust = mean(WindkmhGust, na.rm = TRUE))


# Expanding the data frame to have separate rows for each variable
mean_wind_expanded <- wind_data %>%
  pivot_longer(cols = c(mean_WindkmhInt, mean_WindkmhGust),
               names_to = "Variable",
               values_to = "Mean_Value")


# Creating a bar plot with increased gap between bars for each month
ggplot(mean_wind_expanded, aes(x = factor(Month), y = Mean_Value, fill = Variable)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.5), width = 0.5) +
  labs(x = "Month", y = "Mean Wind Speed", fill = "Variable") +
  scale_x_discrete(labels = 1:12) +
  scale_fill_manual(values = c("cyan", "yellow")) +
  ggtitle("Mean Wind Speeds by Month")
```



The average wind speed (measured in kilometers per hour) for each month is shown in this bar plot, which distinguishes between mean_WindkmhInt (average wind speed) and mean_WindkmhGust (average gust wind speed). A set of bars is used to symbolize each month; each pair of bars shows the mean value of the corresponding wind speed variable.

The months of the year are represented by the labels on the x-axis, which run from January to December. The mean wind speed is represented on the y-axis, giving a numerical representation of the typical wind conditions for each month.

The bars are color-coded to distinguish between the two wind speed variables: mean_WindkmhInt is represented in cyan, while mean_WindkmhGust is depicted in yellow. By comparing the heights of the bars within each month, one can discern the differences in average wind speeds and gust speeds throughout the year.

Importing Crime Dataset for visualising Map.

```
# Importing the libraries
crime <- read.csv('crime23.csv')


str(crime)
```

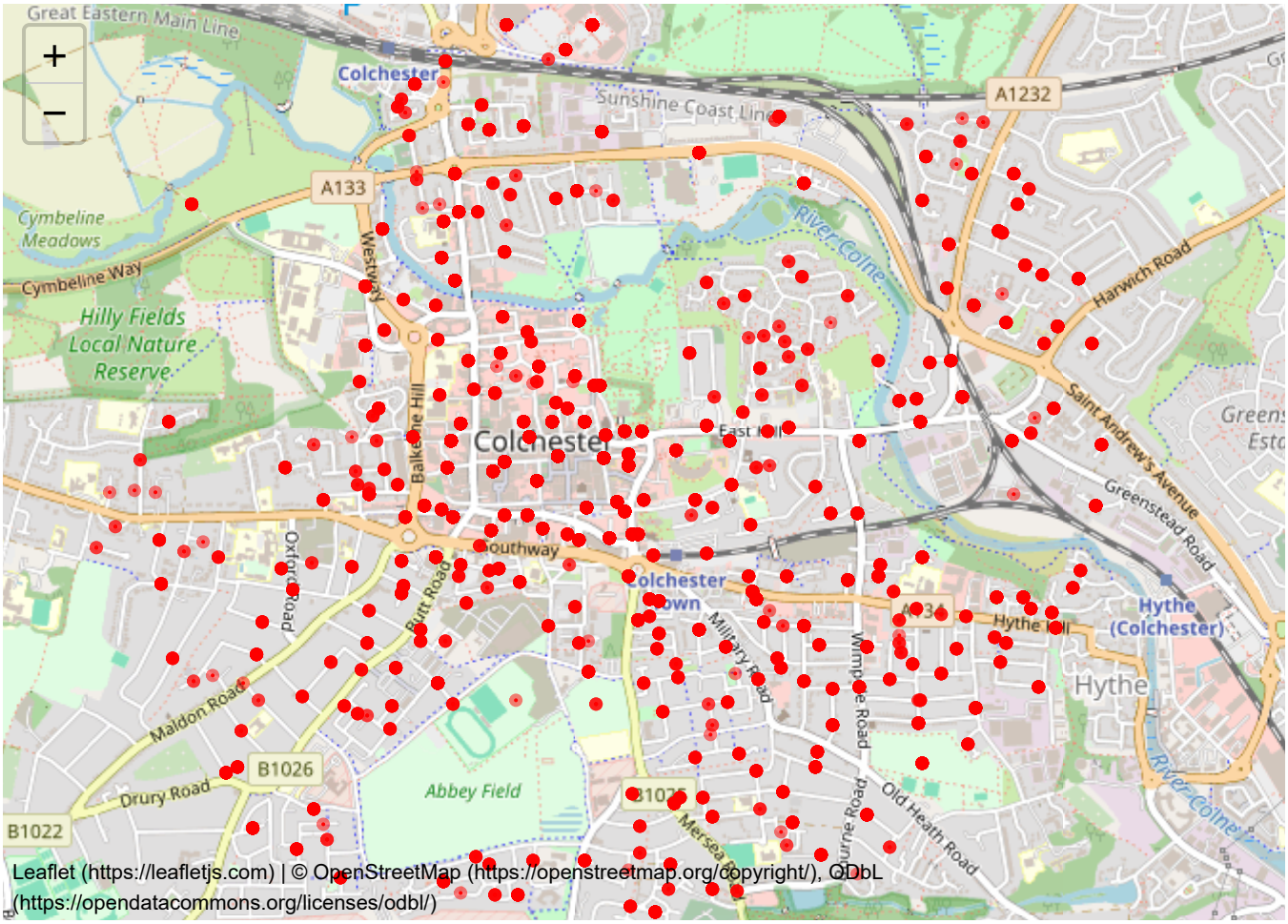
```
## 'data.frame':    6878 obs. of  12 variables:
##  $ category        : chr  "anti-social-behaviour" "anti-social-behaviour" "anti-social-beh
aviour" "anti-social-behaviour" ...
##  $ persistent_id   : chr  "" "" "" "" ...
##  $ date            : chr  "2023-01" "2023-01" "2023-01" "2023-01" ...
##  $ lat             : num  51.9 51.9 51.9 51.9 51.9 ...
##  $ long            : num  0.909 0.902 0.898 0.902 0.895 ...
##  $ street_id       : int  2153366 2153173 2153077 2153186 2153012 2153379 2153105 2153541
2152937 2153107 ...
##  $ street_name     : chr  "On or near Military Road" "On or near " "On or near Culver Stre
et West" "On or near Ryegate Road" ...
##  $ context         : logi  NA NA NA NA NA NA ...
##  $ id              : int  107596596 107596646 107595950 107595953 107595979 107595985 1075
96603 107596291 107596305 107596453 ...
##  $ location_type   : chr  "Force" "Force" "Force" "Force" ...
##  $ location_subtype: chr  "" "" "" "" ...
##  $ outcome_status  : chr  NA NA NA NA ...
```

```
loc<- data.frame(
  inc_type = crime$category,
  street = crime$street_name,
  latitude = as.numeric(crime$lat),
  longitude = as.numeric(crime$long)
)

# Creating a leaflet map with circle markers.
map<- leaflet(loc) %>%
  addTiles() %>%
  addCircleMarkers(radius = 0.5, color = "red", fillOpacity = 1,
                   popup = paste0("Street: ", loc$street,
                                  "<br>Incident Type: ", loc$inc_type)) %>%
  setView(lng = 0.9040, lat = 51.8891, zoom = 14)
```

```
## Assuming "longitude" and "latitude" are longitude and latitude, respectively
```

```
map
```



Using the crime data, we created a leaflet map, marking each occurrence's location with a circle marker. The sort of occurrence is indicated by the color of each marker, and further information is provided by the popup, which includes the street name and incident category. Finding hotspots or trends is made easier by the map's visual depiction of the geographical distribution of occurrences in the region.

Conclusion

This project has offered a thorough examination of the climatic data that was gathered in 2023 from a weather station near Colchester. We have learned a great deal about the seasonal variations in the weather using exploratory data analysis and visualization tools. This project has demonstrated the importance of climate data analysis in understanding local weather patterns, assessing climate variability, and informing decision-making processes across various sectors. The insights gained from this analysis can contribute to improved weather forecasting, climate resilience planning, and environmental management strategies in the Colchester area and beyond