NYPD_Shootings_Week3Assignment

William Eaton

2025-03-27

Step 1: Install all necessary Packages and Libraries

Install all the necessary libraries. Un-comment whatever packages you need

```r
#install.packages("dplyr")
#install.packages("readr")
#install.packages("lubridate")
#install.packages("ggplot2")
#install.packages("hms")
#install.packages("tidyr")
#install.packages
```

prepare all the libraries.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(readr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(ggplot2)
library(hms)
```

```
##
## Attaching package: 'hms'

## The following object is masked from 'package:lubridate':
##
##     hms
```

```r
library(tidyr)
```

Step 2: Get the Data.

The Data is then Acquired from Link Text.

```r
url_in <- "https://data.cityofnewyork.us/api/views/833y-fsy8/rows.csv?accessType=DOWNLOAD"
nypd_data <- read_csv(url_in, show_col_types = FALSE)
```

```
## `curl` package not installed, falling back to using `url()`
```

```r
summary(nypd_data)
head(nypd_data)
```

Step 3: Clean the Data.

We clean the data by removing data columns we're uninterested in (such as the longitude and latitude information), removing Nan values, and fixing the data formats.

```r
total_incidents <- nrow(nypd_data) #The total number of incidents.
nypd_data <- nypd_data %>%
  select(-INCIDENT_KEY) %>%
  select(-X_COORD_CD) %>%
  select(-Y_COORD_CD) %>%
  select(-Latitude) %>%
  select(-Longitude) %>%
  select(-Lon_Lat) %>%
  select(-LOC_OF_OCCUR_DESC) %>%
  select(-PRECINCT) %>%
  select(-JURISDICTION_CODE) %>%
  select(-LOC_CLASSFCTN_DESC) %>%
  select(-LOCATION_DESC) %>%
  select(-PERP_SEX) %>%
  select(-PERP_RACE) %>%
  select(-VIC_SEX) %>%
  select(-VIC_RACE) %>%
  rename(date = OCCUR_DATE) %>%
  mutate(date = mdy(date)) %>%
  mutate(day_of_week = wday(date, label = TRUE, abbr = FALSE)) %>%
  mutate(week_start = floor_date(date, unit = "week"))
  nypd_data <- nypd_data %>%
  mutate(time = as_hms(OCCUR_TIME)) %>%
  mutate(hour_of_day = hour(time)) %>%
  select(-OCCUR_TIME)
head(nypd_data)
```

```
## # A tibble: 6 x 9
##   date       BORO     STATISTICAL_MURDER_FLAG PERP_AGE_GROUP VIC_AGE_GROUP
##   <date>     <chr>    <lgl>                   <chr>          <chr>
## 1 2021-08-09 BRONX    FALSE                   <NA>           18-24
## 2 2018-04-07 BROOKLYN TRUE                    25-44          25-44
## 3 2022-12-02 BRONX    FALSE                   (null)         25-44
## 4 2006-11-19 BROOKLYN TRUE                    UNKNOWN        18-24
## 5 2010-05-09 BRONX    TRUE                    25-44          <18
## 6 2012-07-22 BRONX    FALSE                   18-24          18-24
## # i 4 more variables: day_of_week <ord>, week_start <date>, time <time>,
## #   hour_of_day <int>
```

Step 4: Basic Statistics.

I wanted to start by getting some basic statistics such as the day with the most and least incidents.

```
#We first group by the date and get a total count of incidents per date
incident_counts <- nypd_data %>%
  group_by(date) %>%
  summarize(incident_count = n())

#We then get the day with the most and least incidents along with the average
most_crimes_day <- incident_counts %>% filter(incident_count == max(incident_count)) %>% slice(1)
least_crimes_day <- incident_counts %>% filter(incident_count == min(incident_count)) %>% slice(1)
average_crimes <- mean(incident_counts$incident_count)

print(paste("Day with the most crimes:", most_crimes_day$date, "with", most_crimes_day$incident_count,
```

```
## [1] "Day with the most crimes: 2020-07-05 with 47 incidents"
```

```
print(paste("Day with the least crimes:", least_crimes_day$date, "with", least_crimes_day$incident_coun
```

```
## [1] "Day with the least crimes: 2006-01-26 with 1 incidents"
```

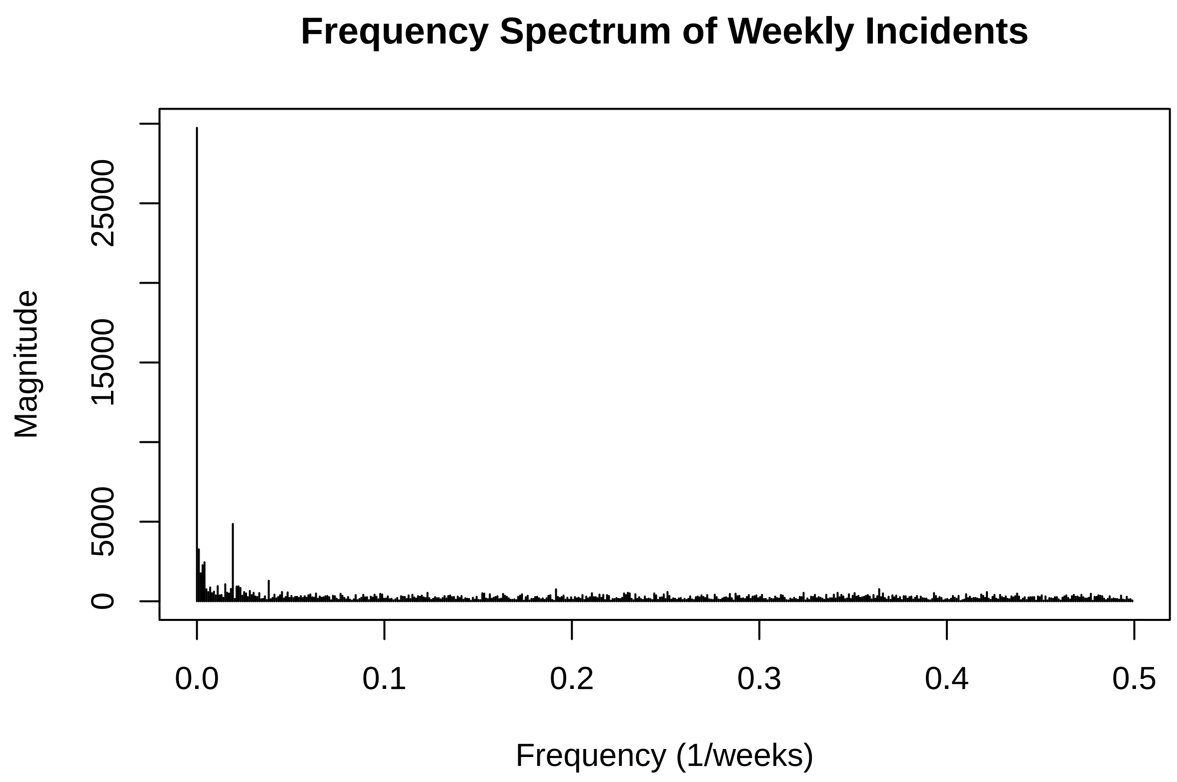
```
print(paste("Average number of incidents per day:", round(average_crimes, 2)))
```

```
## [1] "Average number of incidents per day: 4.63"
```

Step 5: Plot the weekly Data with a fit.

I then plotted the weekly data and did a fourier analysis to get periodic trends.

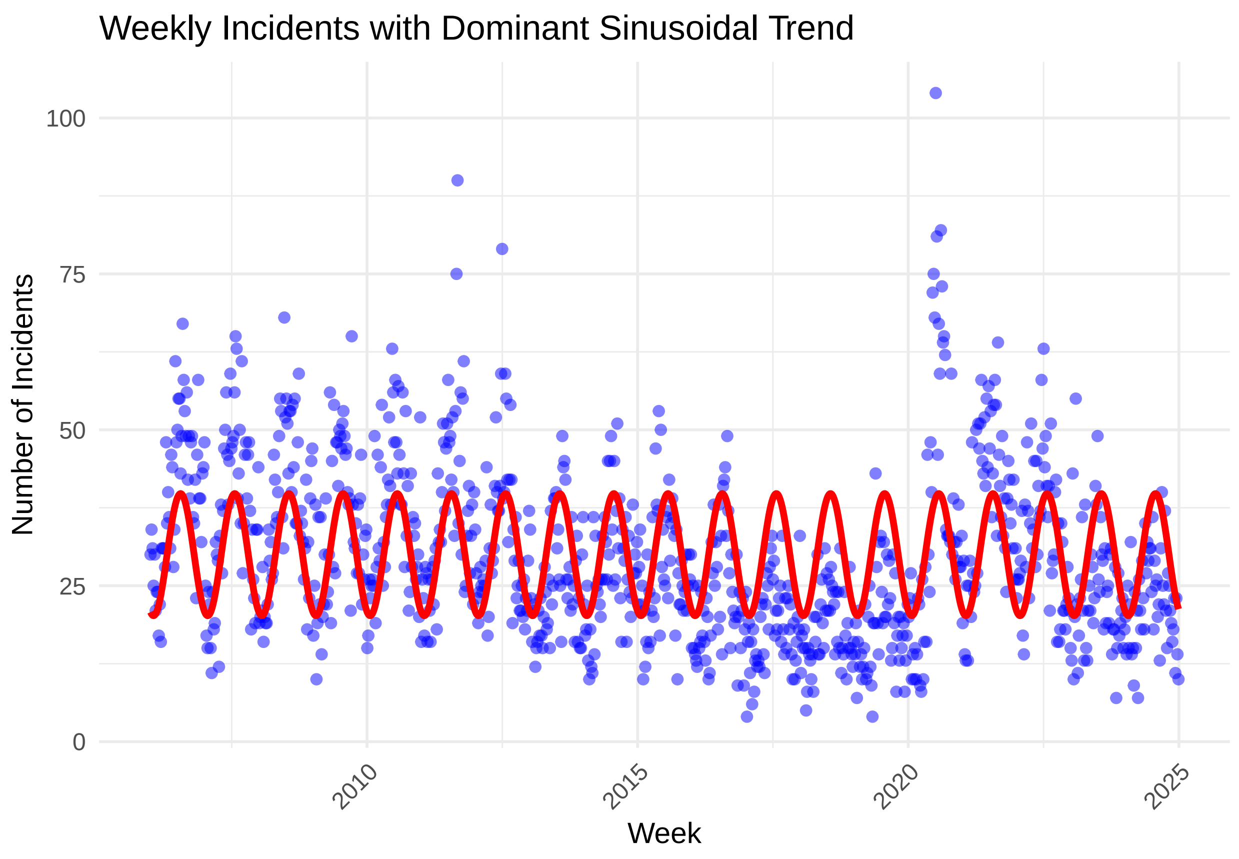
```r
incident_counts_weekly <- nypd_data %>%
  group_by(week_start) %>%
  summarize(incident_count = n(), .groups = "drop")

incident_fft <- fft(incident_counts_weekly$incident_count)

# Get the modulus (magnitude) and frequencies
modulus <- Mod(incident_fft)
n <- length(modulus)
frequencies <- (0:(n - 1)) / n  # Normalized frequency

# Plot spectrum
plot(frequencies[1:(n/2)], modulus[1:(n/2)], type = "h",
     main = "Frequency Spectrum of Weekly Incidents",
     xlab = "Frequency (1/weeks)", ylab = "Magnitude")
```



```r
dominant_index <- which.max(modulus[2:(n/2)]) + 1

# Reconstruct only the dominant frequency
dominant_freq <- incident_fft
dominant_freq[-c(1, dominant_index, n - dominant_index + 2)] <- 0
reconstructed_signal <- Re(fft(dominant_freq, inverse = TRUE) / n)

# Add to original dataframe
incident_counts_weekly$sinusoidal_trend <- reconstructed_signal
```

```r
ggplot(incident_counts_weekly, aes(x = week_start)) +
  geom_point(aes(y = incident_count), color = "blue", alpha = 0.5) +
  geom_line(aes(y = sinusoidal_trend), color = "red", size = 1.2) +
  labs(
    title = "Weekly Incidents with Dominant Sinusoidal Trend",
    x = "Week",
    y = "Number of Incidents"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

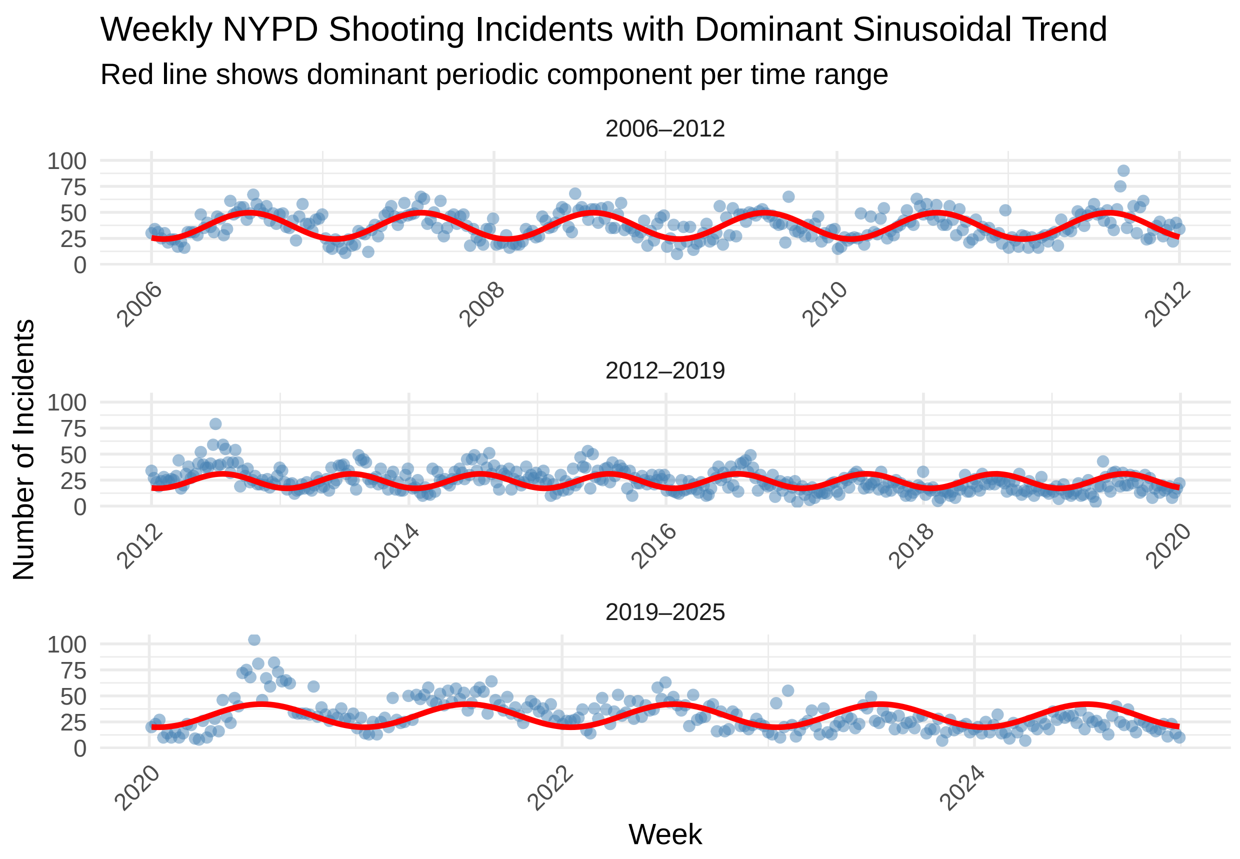


```r
dominant_freq_value <- frequencies[dominant_index]
period_in_weeks <- 1 / dominant_freq_value
print(paste("Estimated period of oscillation:", round(period_in_weeks, 2), "weeks"))
```

```
## [1] "Estimated period of oscillation: 52.21 weeks"
```

I also wanted to see if this periodic pattern was consistent at various “time Periods”. I eye-balled and separetd the data into three eras that looked like they had different average incident counts.

```r
periods <- list(
  "2006-2012" = c(ymd("2006-01-01"), ymd("2012-01-01")),
  "2012-2019" = c(ymd("2012-01-01"), ymd("2019-12-31")),
  "2019-2025" = c(ymd("2019-12-31"), max(nypd_data$week_start, na.rm = TRUE))
)

# Function to process each period
analyze_period <- function(data, start_date, end_date, label) {
  df <- data %>%
    filter(week_start >= start_date, week_start <= end_date) %>%
    group_by(week_start) %>%
    summarize(incident_count = n(), .groups = "drop")

  # Use only the range that actually exists in the data
  data_min <- min(df$week_start)
  data_max <- max(df$week_start)

  df <- df %>%
    complete(week_start = seq.Date(data_min, data_max, by = "week"),
             fill = list(incident_count = 0))

  # FFT
  fft_vals <- fft(df$incident_count)
  modulus <- Mod(fft_vals)
  n <- length(modulus)
  freqs <- (0:(n - 1)) / n

  dominant_idx <- which.max(modulus[2:(n/2)]) + 1
  dominant_freq <- fft_vals
  dominant_freq[-c(1, dominant_idx, n - dominant_idx + 2)] <- 0
  trend <- Re(fft(dominant_freq, inverse = TRUE) / n)
  period_in_weeks <- 1 / freqs[dominant_idx]

  df$sinusoidal_trend <- trend
  df$period_label <- label
  df$estimated_period <- round(period_in_weeks, 1)

  return(df)
}

# Run analysis for all periods
results_list <- lapply(names(periods), function(label) {
  range <- periods[[label]]
  analyze_period(nypd_data, range[1], range[2], label)
})

# Combine all
all_periods_df <- bind_rows(results_list)

# Plot each period
ggplot(all_periods_df, aes(x = week_start)) +
  geom_point(aes(y = incident_count), color = "steelblue", alpha = 0.5) +
  geom_line(aes(y = sinusoidal_trend), color = "red", size = 1) +
  facet_wrap(~ period_label, scales = "free_x", ncol = 1) +
  labs(
    title = "Weekly NYPD Shooting Incidents with Dominant Sinusoidal Trend",
    subtitle = "Red line shows dominant periodic component per time range",
    x = "Week",
    y = "Number of Incidents"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



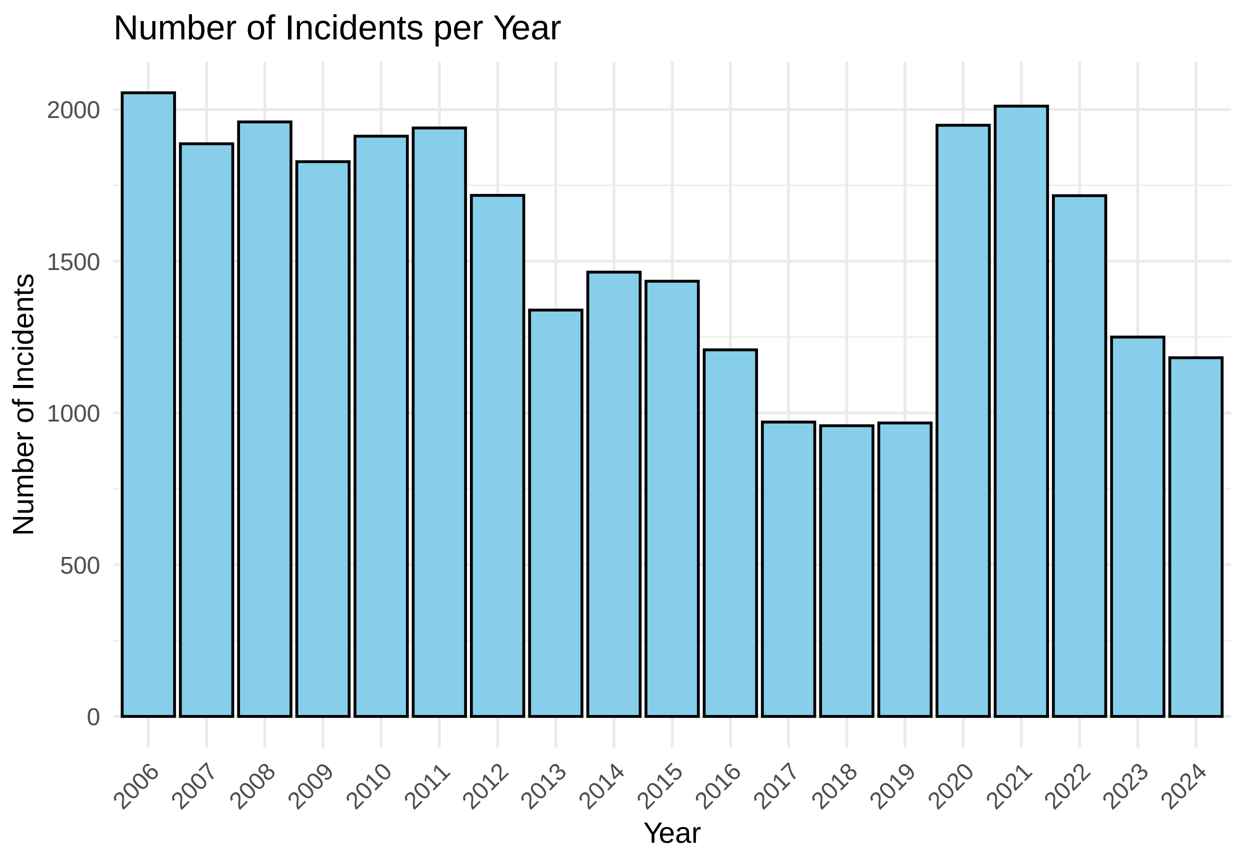
```
all_periods_df %>%
  group_by(period_label) %>%
  summarize(
    Estimated_Period_Weeks = unique(estimated_period),
    Average_Weekly_Incidents = round(mean(incident_count), 2),
    Standard_Deviation = round(sd(incident_count), 2)
  )
```

```
## # A tibble: 3 x 4
##   period_label Estimated_Period_Weeks Average_Weekly_Incide~1 Standard_Deviation
##   <chr>                         <dbl>                   <dbl>              <dbl>
## 1 2006-2012                      52.3                    37.0               13.0
## 2 2012-2019                      52.2                    24.1               10.3
## 3 2019-2025                      52.2                    31                 15.4
## # i abbreviated name: 1: Average_Weekly_Incidents
```

Step 6: Plot incident counts over various time resolutions.

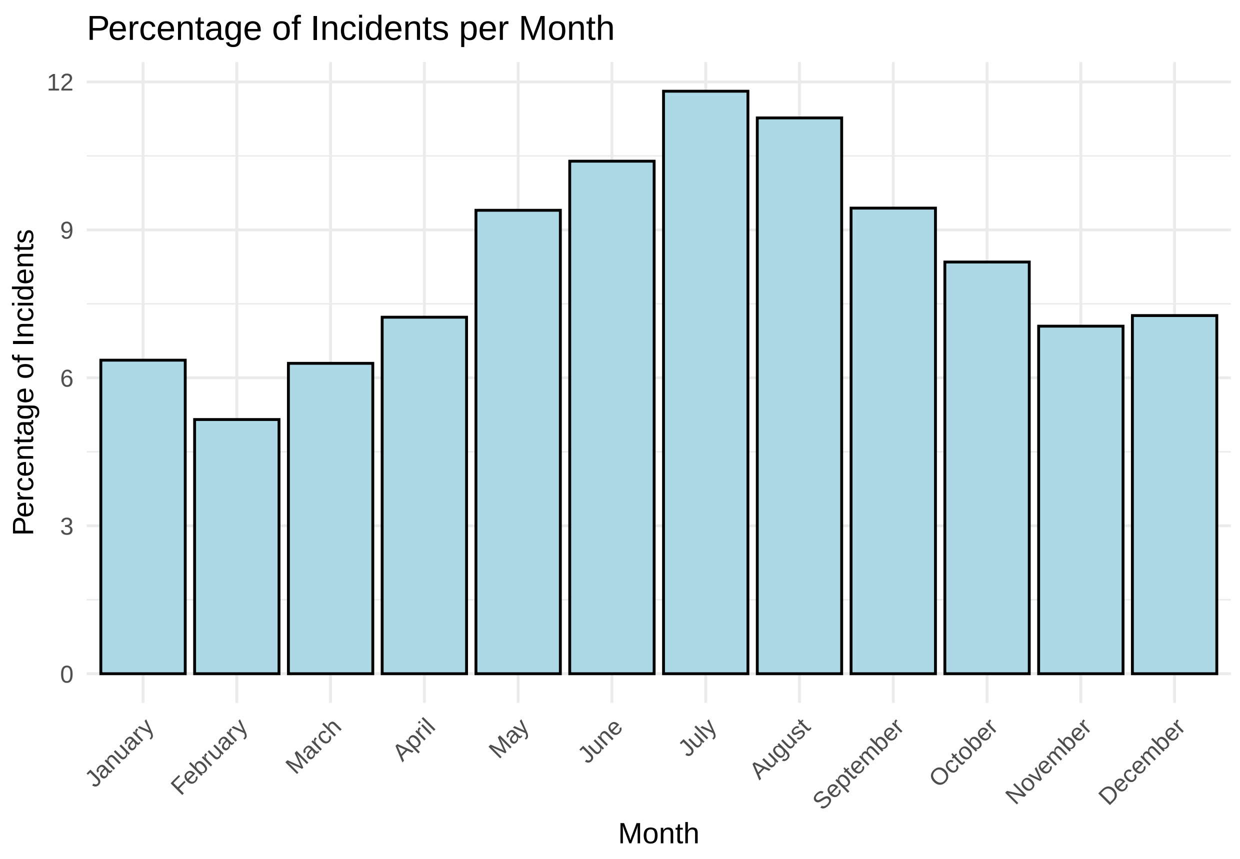
I then started grouping the incident counts into various "time resolutions". I started with grouping them based on the year.

```r
nypd_data <- nypd_data %>%
  mutate(year = year(date)) %>%
  filter(year >= 2000 & year <= 2025)

# Count incidents per year
incident_counts_year <- nypd_data %>%
  group_by(year) %>%
  summarize(incident_count = n())

# Plot number of incidents per year
ggplot(incident_counts_year, aes(x = factor(year), y = incident_count)) +
  geom_col(fill = "skyblue", color = "black") +
  labs(
    title = "Number of Incidents per Year",
    x = "Year",
    y = "Number of Incidents"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



I then grouped them based on the month.

```r
# Add month name to data
nypd_data <- nypd_data %>%
  mutate(month_name = month(date, label = TRUE, abbr = FALSE))

# Group by month and count
incident_counts_month <- nypd_data %>%
  group_by(month_name) %>%
  summarize(incident_count = n()) %>%
  mutate(percentage = (incident_count / total_incidents) * 100)

# Reorder factor levels for correct month order
incident_counts_month$month_name <- factor(incident_counts_month$month_name,
                                           levels = month.name)

# Plot percentage of incidents per month
ggplot(incident_counts_month, aes(x = month_name, y = percentage)) +
  geom_col(fill = "lightblue", color = "black") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE, color = "orange") +
  labs(
    title = "Percentage of Incidents per Month",
    x = "Month",
    y = "Percentage of Incidents"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

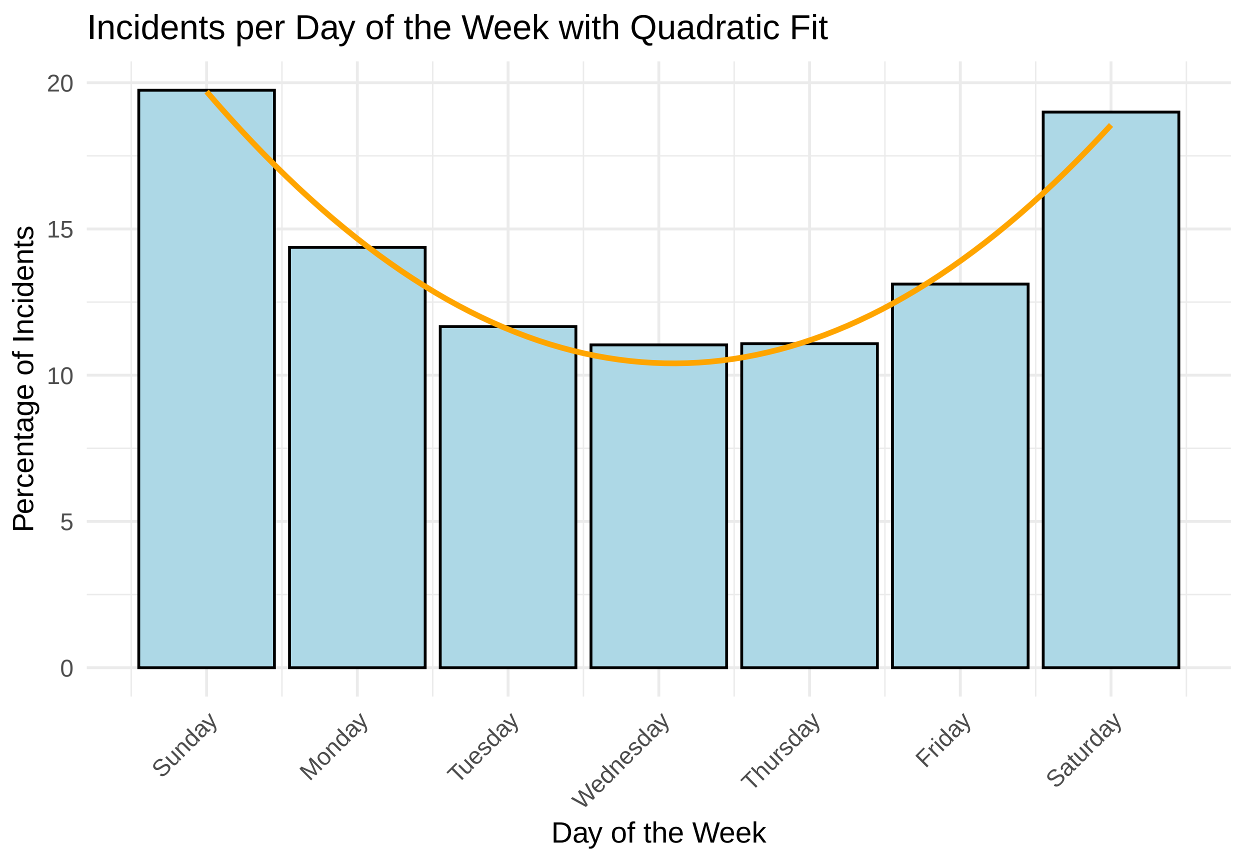


I then grouped them based on the day of the week.

```r
incident_counts_day_of_week <- nypd_data %>%
  count(day_of_week) %>%
  mutate(
    day_of_week_num = as.numeric(factor(day_of_week, levels = c(
      "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
    ))),
    avg_incidents = n / sum(n)
  )

# Fit a quadratic model
quad_model <- lm(avg_incidents ~ poly(day_of_week_num, 2, raw = TRUE), data = incident_counts_day_of_we
summary(quad_model)
```

```
##
## Call:
## lm(formula = avg_incidents ~ poly(day_of_week_num, 2, raw = TRUE),
##     data = incident_counts_day_of_week)
##
## Residuals:
##          1          2          3          4          5          6          7
##  0.0004467 -0.0029730  0.0009077  0.0062390 -0.0011335 -0.0078960  0.0044091
##
## Coefficients:
##                                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                            0.2666372  0.0089474   29.80 7.55e-06
## poly(day_of_week_num, 2, raw = TRUE)1 -0.0793461  0.0051277  -15.47 0.000102
## poly(day_of_week_num, 2, raw = TRUE)2  0.0096802  0.0006264   15.45 0.000102
##
## (Intercept)                           ***
## poly(day_of_week_num, 2, raw = TRUE)1 ***
## poly(day_of_week_num, 2, raw = TRUE)2 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005741 on 4 degrees of freedom
## Multiple R-squared:  0.9837, Adjusted R-squared:  0.9756
## F-statistic: 120.9 on 2 and 4 DF,  p-value: 0.0002647
```

```r
# Plot the data
ggplot(incident_counts_day_of_week, aes(x = day_of_week_num, y = avg_incidents * 100)) +
  geom_col(fill = "lightblue", color = "black") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), color = "orange", se = FALSE) +
  labs(
    title = "Incidents per Day of the Week with Quadratic Fit",
    x = "Day of the Week",
    y = "Percentage of Incidents"
  ) +
  theme_minimal() +
  scale_x_continuous(
    breaks = 1:7,
    labels = c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



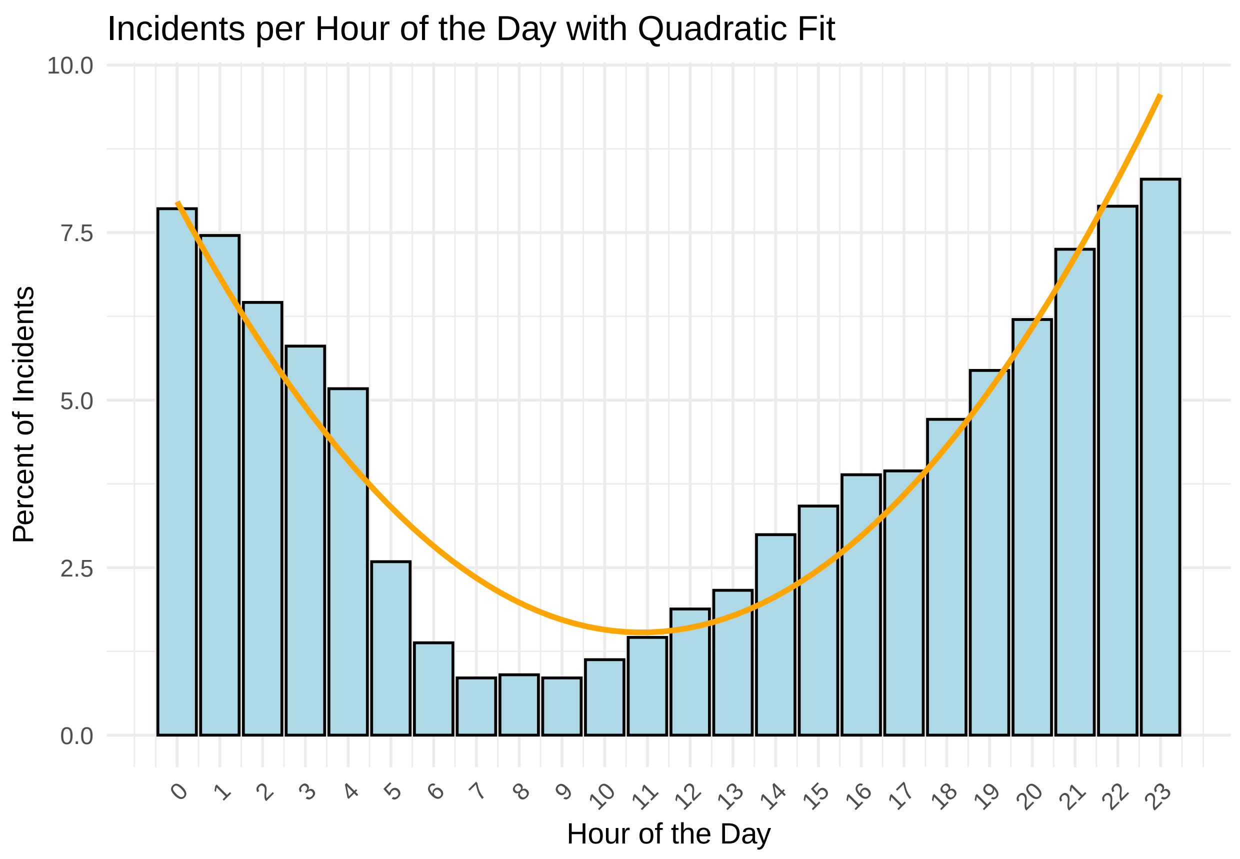
I finally grouped them based on hour of the day.

```
# Count incidents by hour of the day
incident_counts_hourly <- nypd_data %>%
  group_by(hour_of_day) %>%
  summarize(incident_count = n())

# Fit a quadratic model for incident counts by hour of the day
quad_model_hourly <- lm(incident_count ~ poly(hour_of_day, 2, raw = TRUE), data = incident_counts_hourl

# Display summary of the quadratic model
summary(quad_model_hourly)
```

```
##
## Call:
## lm(formula = incident_count ~ poly(hour_of_day, 2, raw = TRUE),
##     data = incident_counts_hourly)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -443.63 -160.45   59.06  187.37  318.96
##
```

```
## Coefficients:
##                                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)                         2368.156    141.318   16.76 1.25e-13 ***
## poly(hour_of_day, 2, raw = TRUE)1 -352.119     28.461  -12.37 4.14e-11 ***
## poly(hour_of_day, 2, raw = TRUE)2   16.210      1.195   13.56 7.37e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 250.3 on 21 degrees of freedom
## Multiple R-squared:  0.9013, Adjusted R-squared:  0.8919
## F-statistic: 95.93 on 2 and 21 DF,  p-value: 2.744e-11
```

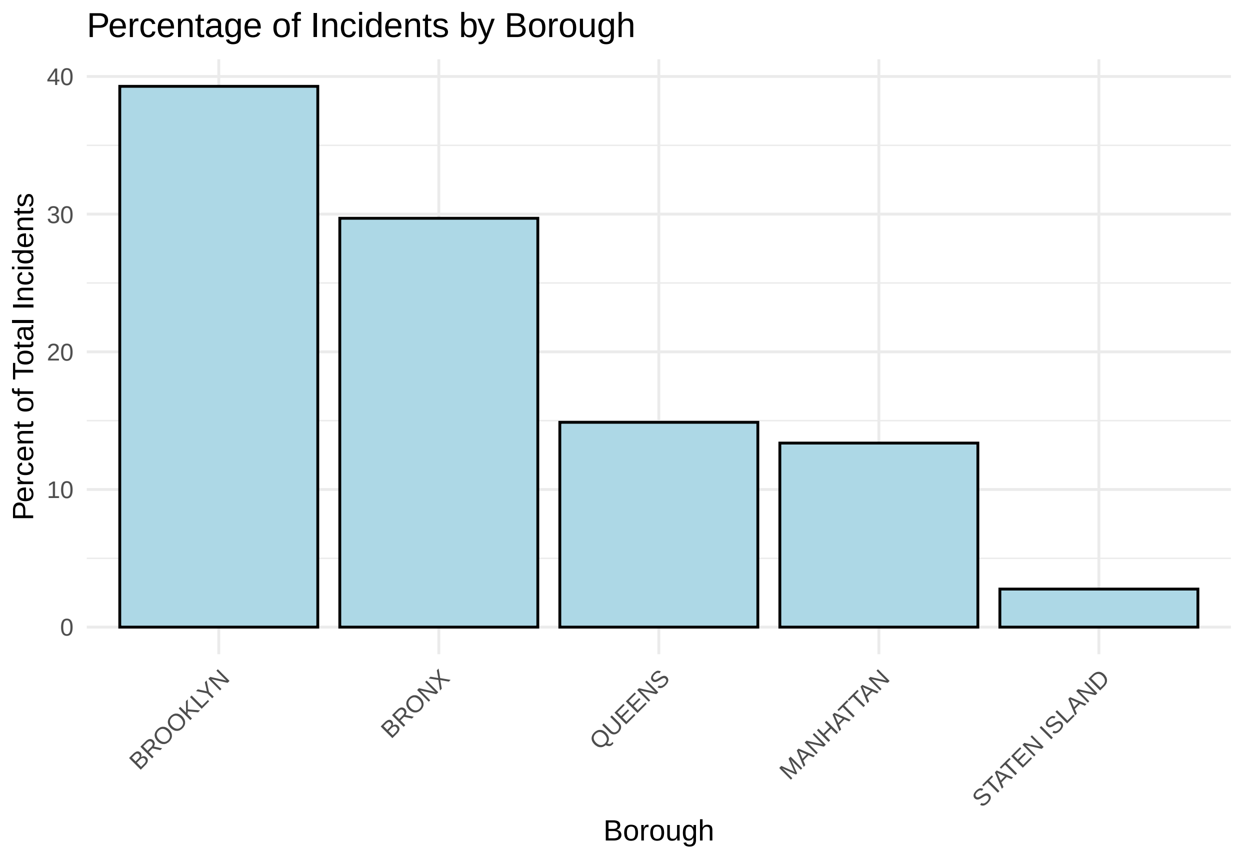
```r
ggplot(incident_counts_hourly, aes(x = hour_of_day, y = (incident_count / total_incidents) * 100)) +
geom_col(fill = "lightblue", color = "black") +
geom_smooth(method = "lm", formula = y ~ poly(x, 2), color = "orange", se = FALSE) +
labs(title = "Incidents per Hour of the Day with Quadratic Fit",
x = "Hour of the Day",
y = "Percent of Incidents") +
theme_minimal() +
scale_x_continuous(breaks = 0:23, labels = 0:23) +
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



For fun, I decided to plot the data based on the location of the incidents.

```r
# Calculate total incidents
total_incidents <- nrow(nypd_data)
```

```r
# Count incidents by borough and calculate percentage
incident_counts_boro <- nypd_data %>%
  group_by(BORO) %>%
  summarize(incident_count = n()) %>%
  mutate(percentage = (incident_count / total_incidents) * 100)

# Plot the percentage of incidents by borough
ggplot(incident_counts_boro, aes(x = reorder(BORO, -percentage), y = percentage)) +
  geom_col(fill = "lightblue", color = "black") +
  labs(title = "Percentage of Incidents by Borough",
       x = "Borough",
       y = "Percent of Total Incidents") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

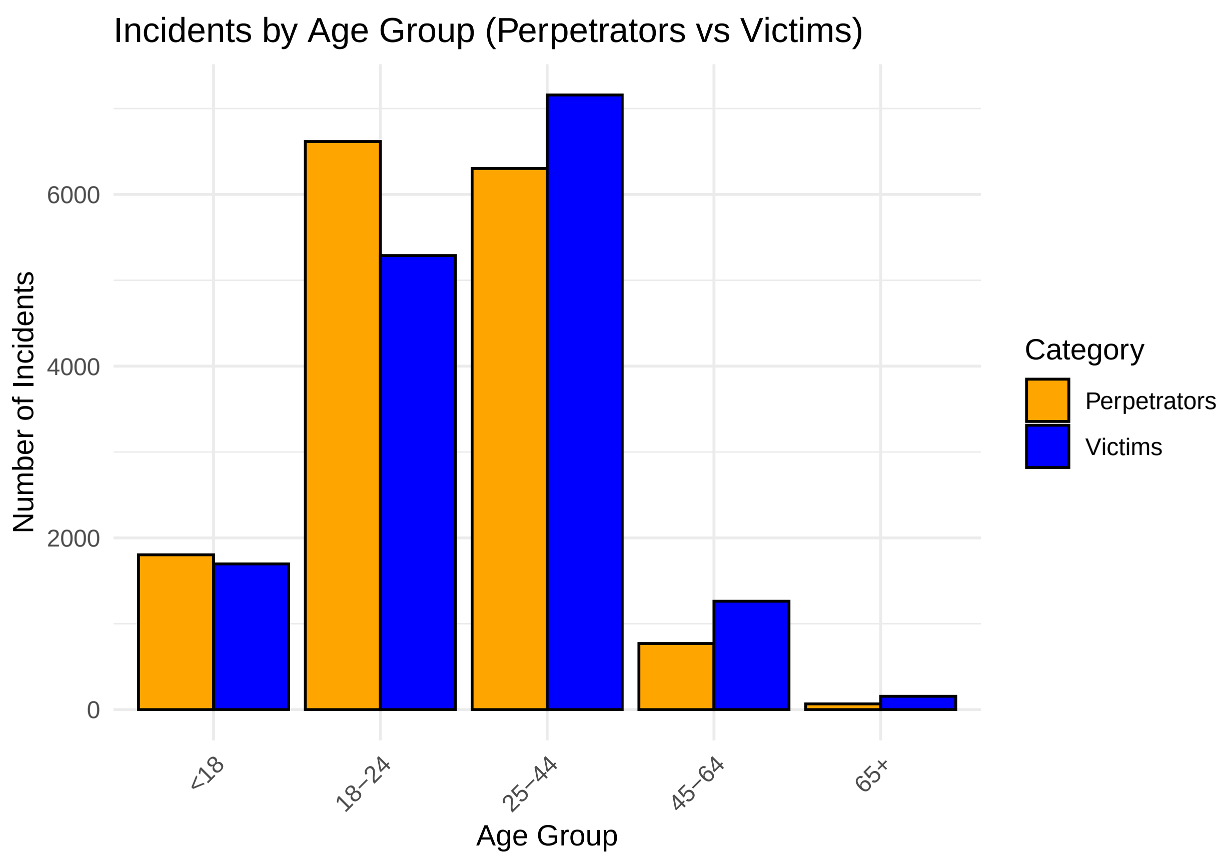


Step 7: Plot incident counts over age groups for both Victims and Perpetrators.

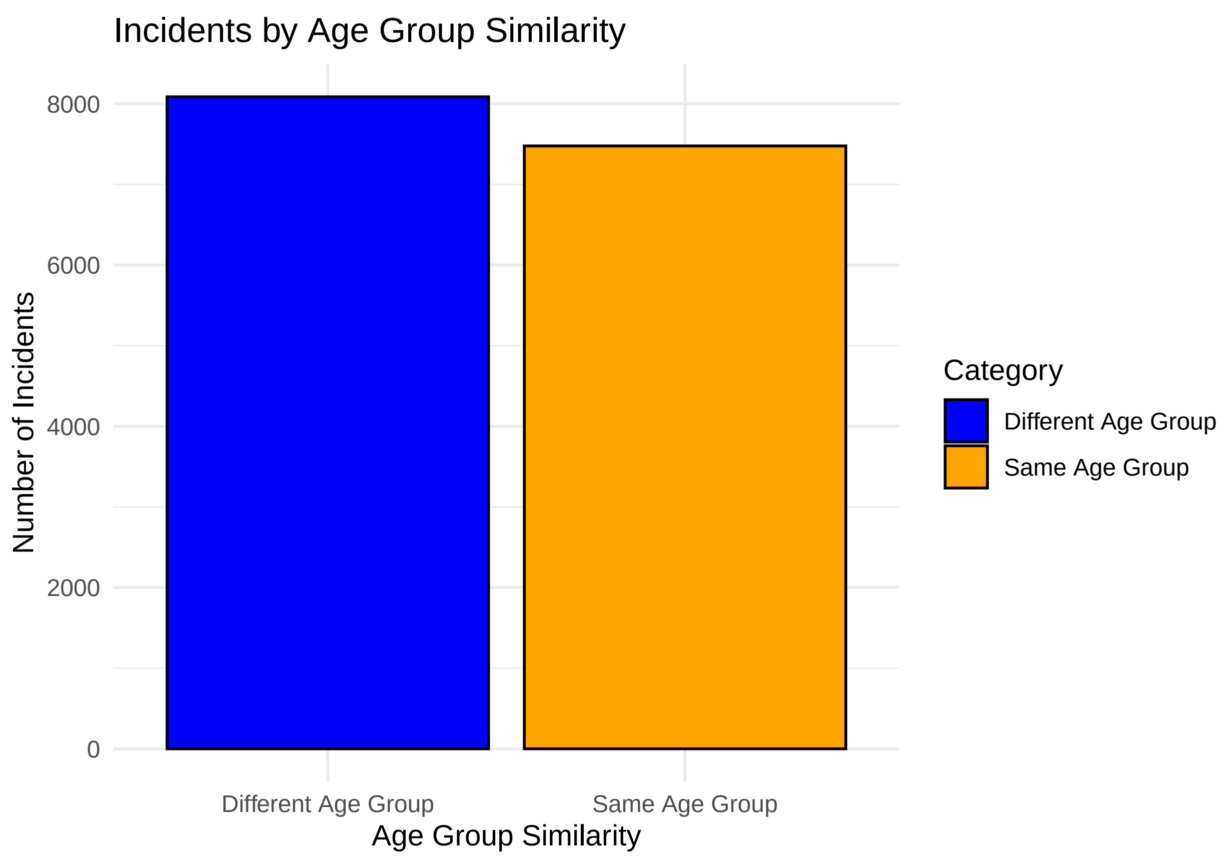
```r
nypd_data_filtered <- nypd_data %>%
  filter(!is.na(PERP_AGE_GROUP), !is.na(VIC_AGE_GROUP)) %>%
  filter(PERP_AGE_GROUP != "(null)", PERP_AGE_GROUP != "UNKNOWN",
         PERP_AGE_GROUP != "1020", PERP_AGE_GROUP != "1028") %>%
  filter(!grepl("^[0-9]{2,3}$", PERP_AGE_GROUP) | as.numeric(PERP_AGE_GROUP) <= 100) %>%
  filter(VIC_AGE_GROUP != "(null)", VIC_AGE_GROUP != "UNKNOWN",
         VIC_AGE_GROUP != "1020", VIC_AGE_GROUP != "1028") %>%
  filter(!grepl("^[0-9]{2,3}$", VIC_AGE_GROUP) | as.numeric(VIC_AGE_GROUP) <= 100)
```

```
## Warning: There was 1 warning in ‘filter()‘.
## i In argument: ‘|...‘.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in ‘filter()‘.
## i In argument: ‘|...‘.
## Caused by warning:
## ! NAs introduced by coercion
```

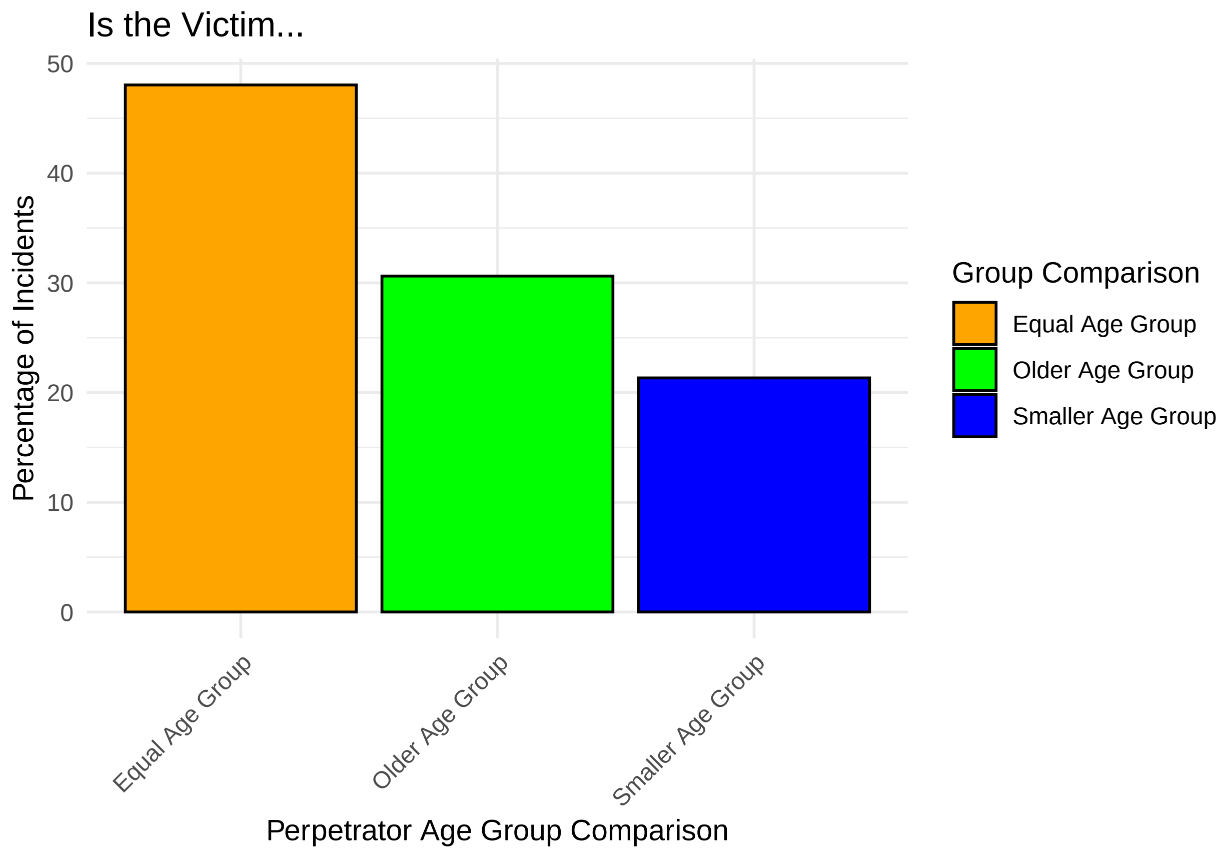
```r
age_order <- c("<18", "18-24", "25-44", "45-64", "65+")

# Create a long-format dataset for plotting and remove rows where Age_Group is NA
incident_counts_age_group <- nypd_data_filtered %>%
  select(PERP_AGE_GROUP, VIC_AGE_GROUP) %>%
  pivot_longer(cols = c(PERP_AGE_GROUP, VIC_AGE_GROUP),
               names_to = "Type", values_to = "Age_Group") %>%
  mutate(Age_Group = factor(Age_Group, levels = age_order)) %>%
  group_by(Type, Age_Group) %>%
  summarize(incident_count = n(), .groups = "drop") %>%
  filter(!is.na(Age_Group))  # Remove NA rows

ggplot(incident_counts_age_group, aes(x = Age_Group, y = incident_count, fill = Type)) +
  geom_col(position = "dodge", color = "black") +
  scale_fill_manual(values = c("PERP_AGE_GROUP" = "orange", "VIC_AGE_GROUP" = "blue"),
                    labels = c("Perpetrators", "Victims")) +
  labs(title = "Incidents by Age Group (Perpetrators vs Victims)",
       x = "Age Group",
       y = "Number of Incidents",
       fill = "Category") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```r
same_age_counts <- nypd_data_filtered %>%
  mutate(Same_Age_Group = ifelse(PERP_AGE_GROUP == VIC_AGE_GROUP, "Same Age Group", "Different Age Grou
  count(Same_Age_Group)  # Count the number of incidents per group

ggplot(same_age_counts, aes(x = Same_Age_Group, y = n, fill = Same_Age_Group)) +
  geom_col(color = "black") +
  scale_fill_manual(values = c("Same Age Group" = "orange", "Different Age Group" = "blue")) +
  labs(title = "Incidents by Age Group Similarity",
       x = "Age Group Similarity",
       y = "Number of Incidents",
       fill = "Category") +
  theme_minimal()
```

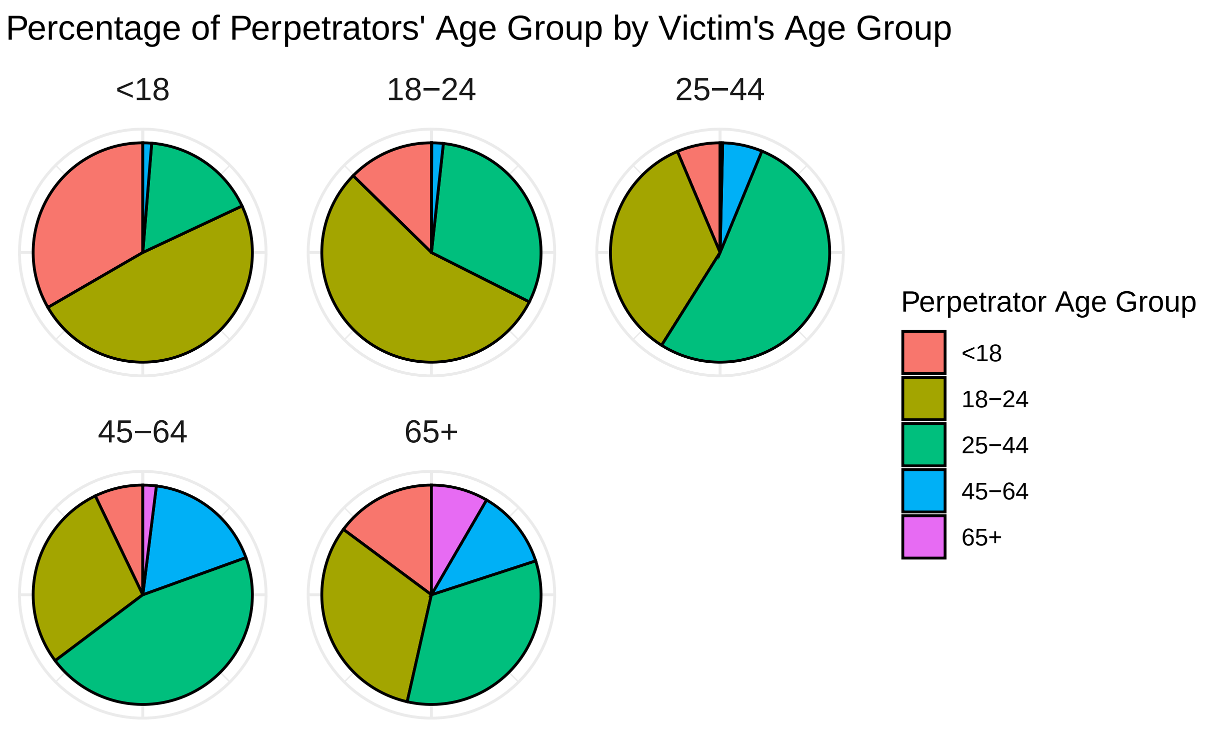


```r
compare_age_groups <- function(perp_age, vic_age) {
  if (vic_age == perp_age) {
    return("Equal Age Group")
  } else if (vic_age < perp_age) {
    return("Smaller Age Group")
  } else {
    return("Older Age Group")
  }
}

# Apply the comparison function
nypd_data_filtered <- nypd_data_filtered %>%
  mutate(
    Age_Comparison = mapply(compare_age_groups, PERP_AGE_GROUP, VIC_AGE_GROUP)
  )

# Calculate the percentage of incidents for each category
age_comparison_counts <- nypd_data_filtered %>%
  count(Age_Comparison) %>%
  mutate(percentage = (n / sum(n)) * 100)

# Plot the bar charts
ggplot(age_comparison_counts, aes(x = Age_Comparison, y = percentage, fill = Age_Comparison)) +
  geom_bar(stat = "identity", color = "black") +
  scale_fill_manual(values = c("Smaller Age Group" = "blue", "Equal Age Group" = "orange", "Older Age G
  labs(
    title = "Is the Victim...",
    x = "Perpetrator Age Group Comparison",
    y = "Percentage of Incidents",
    fill = "Group Comparison"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
# Define valid age group levels
age_order <- c("<18", "18-24", "25-44", "45-64", "65+")

# Clean and filter the dataset
nypd_data_filtered <- nypd_data %>%
  filter(PERP_AGE_GROUP %in% age_order,
         VIC_AGE_GROUP %in% age_order)

# Create a long-format dataset for plotting
incident_counts <- nypd_data_filtered %>%
  select(PERP_AGE_GROUP, VIC_AGE_GROUP) %>%
  mutate(
    PERP_AGE_GROUP = factor(PERP_AGE_GROUP, levels = age_order),
    VIC_AGE_GROUP = factor(VIC_AGE_GROUP, levels = age_order)
  ) %>%
  droplevels() %>%  # Drop any unused factor levels
  group_by(VIC_AGE_GROUP, PERP_AGE_GROUP) %>%
```

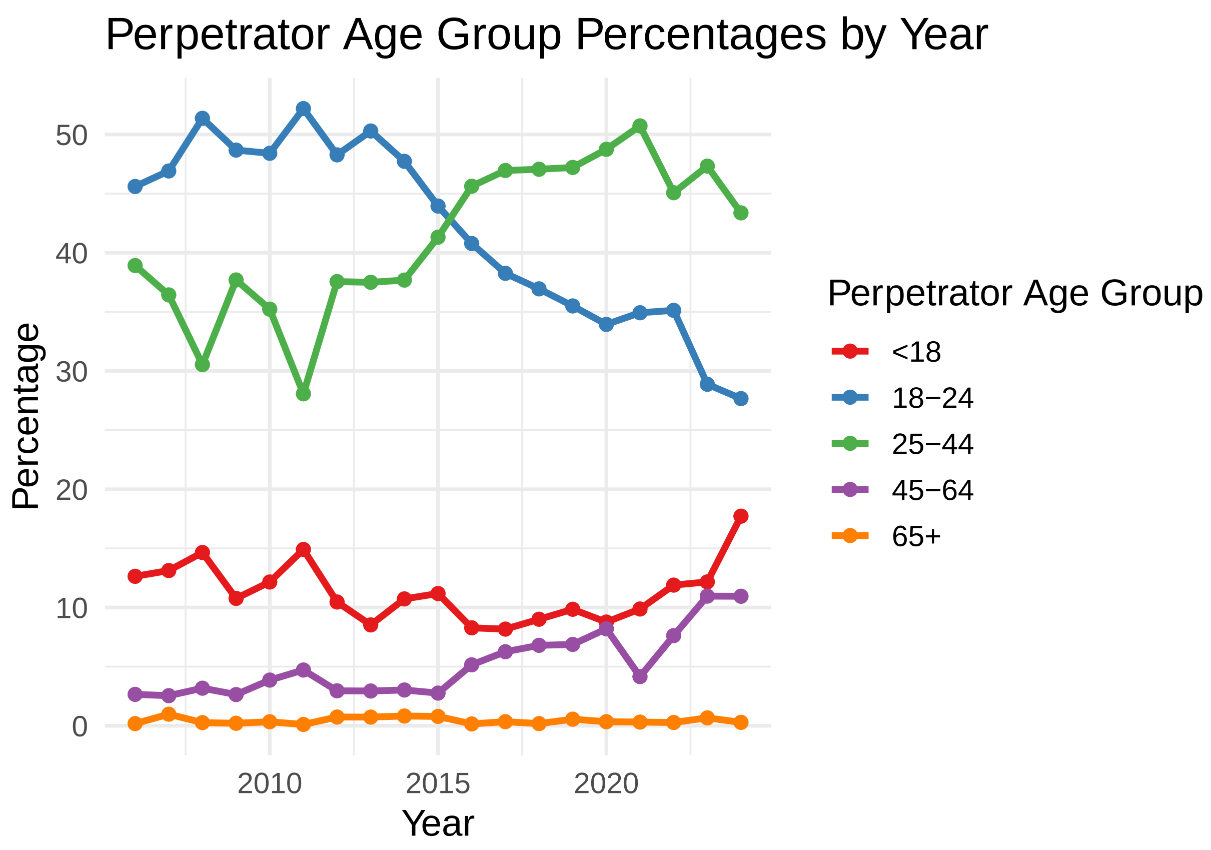
```r
  summarize(incident_count = n(), .groups = "drop") %>%
  group_by(VIC_AGE_GROUP) %>%
  mutate(percentage = (incident_count / sum(incident_count)) * 100)

# Create pie charts for each victim's age group
ggplot(incident_counts, aes(x = "", y = percentage, fill = PERP_AGE_GROUP)) +
  geom_bar(stat = "identity", width = 1, color = "black") +
  facet_wrap(~ VIC_AGE_GROUP) +
  coord_polar(theta = "y") +
  labs(
    title = "Percentage of Perpetrators' Age Group by Victim's Age Group",
    x = NULL, y = NULL, fill = "Perpetrator Age Group"
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_blank(),
    axis.ticks = element_blank(),
    strip.text = element_text(size = 12),
    legend.position = "right"
  )
```



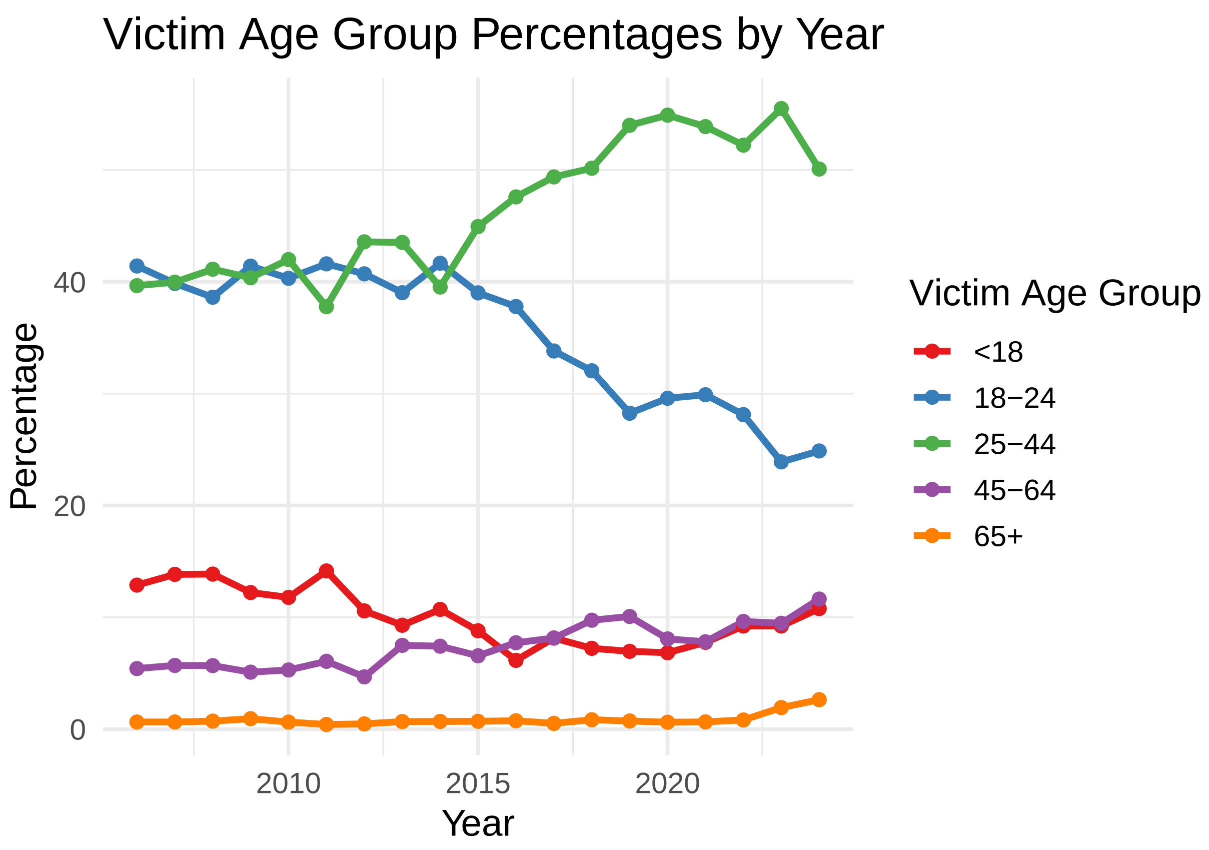
## Step 8: Plot incident counts for each age group over time.

```r
age_order <- c("<18", "18-24", "25-44", "45-64", "65+")

# Calculate yearly percentages
```

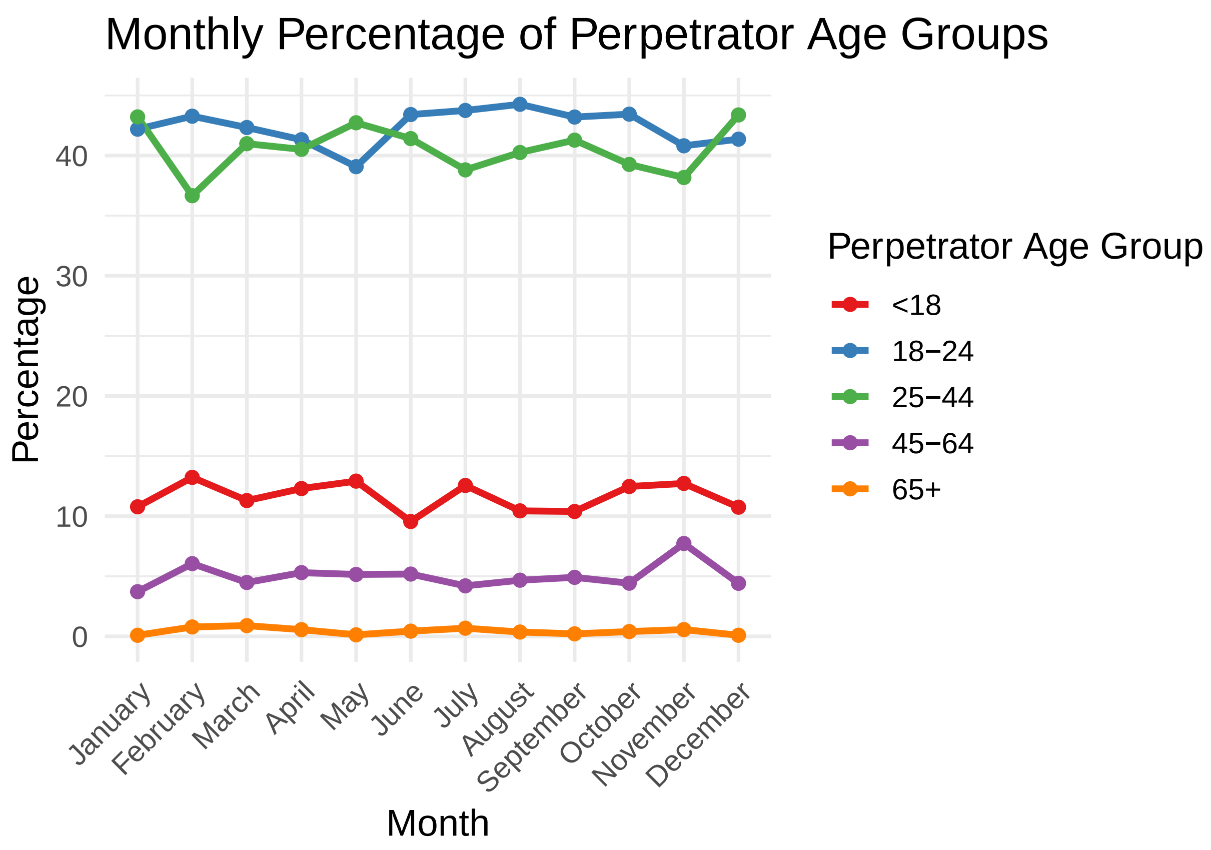
```r
perp_age_by_year <- nypd_data %>%
  filter(!is.na(PERP_AGE_GROUP),
         PERP_AGE_GROUP %in% age_order,
         !is.na(date)) %>%
  mutate(year = year(date),
         PERP_AGE_GROUP = factor(PERP_AGE_GROUP, levels = age_order)) %>%
  group_by(year, PERP_AGE_GROUP) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(year) %>%
  mutate(percentage = 100 * count / sum(count))

vic_age_by_year <- nypd_data %>%
  filter(!is.na(VIC_AGE_GROUP),
         VIC_AGE_GROUP %in% age_order,
         !is.na(date)) %>%
  mutate(year = year(date),
         VIC_AGE_GROUP = factor(VIC_AGE_GROUP, levels = age_order)) %>%
  group_by(year, VIC_AGE_GROUP) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(year) %>%
  mutate(percentage = 100 * count / sum(count))

# Line plot
ggplot(perp_age_by_year, aes(x = year, y = percentage, color = PERP_AGE_GROUP)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  scale_color_brewer(palette = "Set1") +
  labs(title = "Perpetrator Age Group Percentages by Year",
       x = "Year", y = "Percentage", color = "Perpetrator Age Group") +
  theme_minimal(base_size = 14)
```



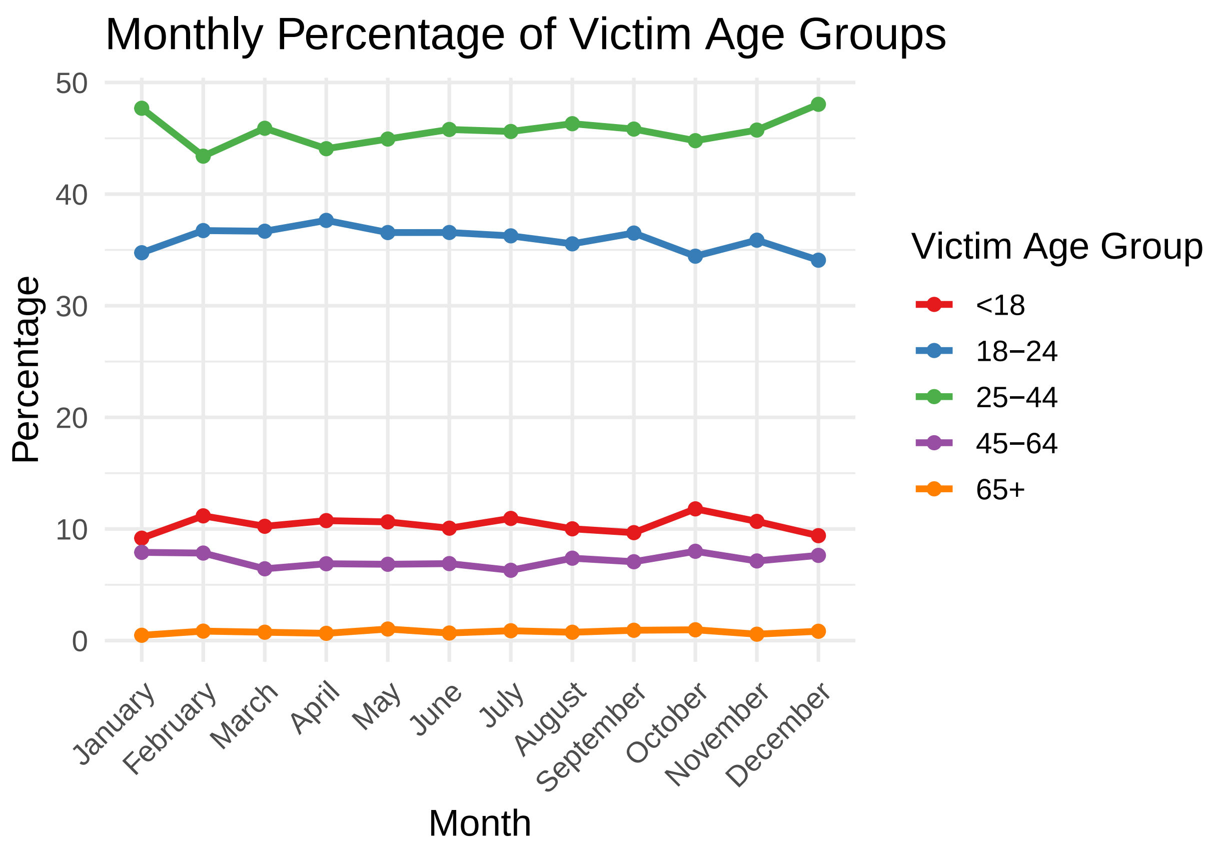
```r
ggplot(vic_age_by_year, aes(x = year, y = percentage, color = VIC_AGE_GROUP)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  scale_color_brewer(palette = "Set1") +
  labs(title = "Victim Age Group Percentages by Year",
       x = "Year", y = "Percentage", color = "Victim Age Group") +
  theme_minimal(base_size = 14)
```



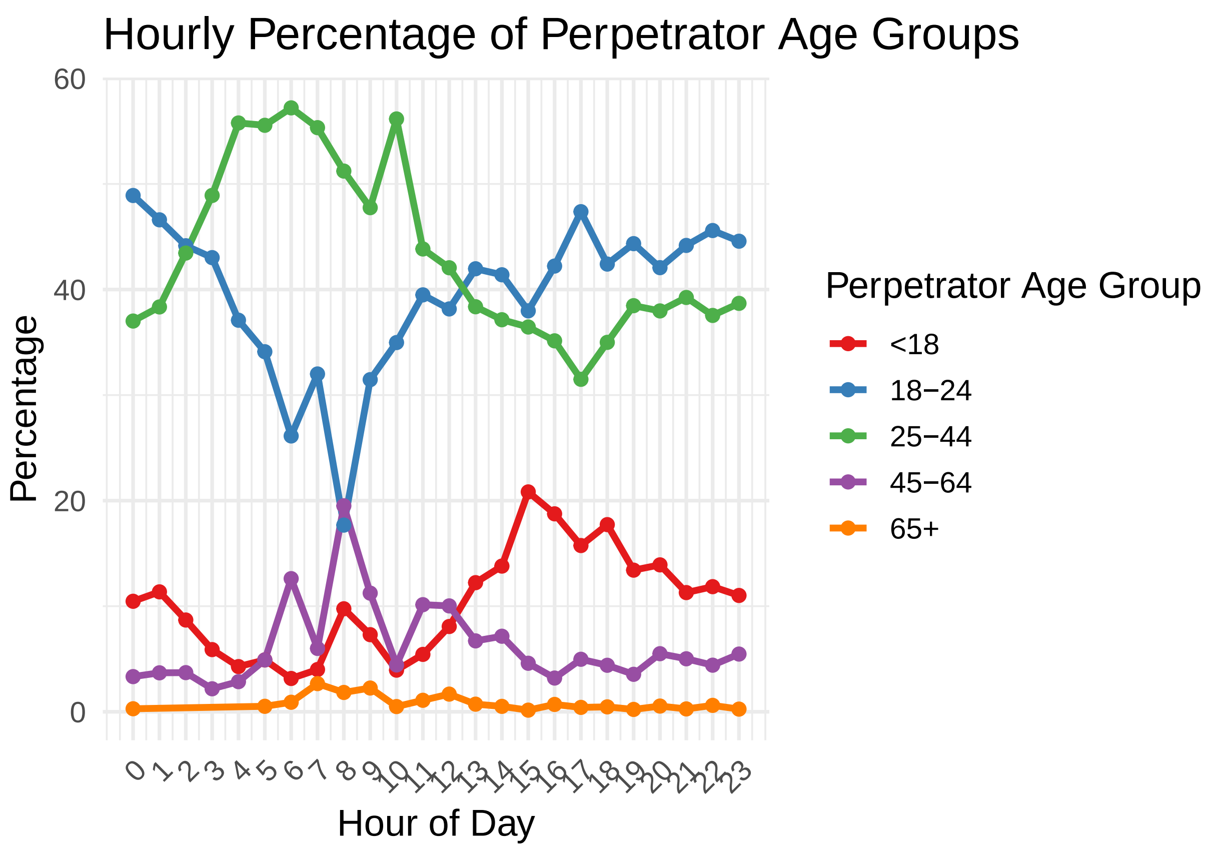
```
perp_monthly <- nypd_data %>%
  filter(!is.na(PERP_AGE_GROUP), PERP_AGE_GROUP %in% age_order) %>%
  mutate(PERP_AGE_GROUP = factor(PERP_AGE_GROUP, levels = age_order)) %>%
  group_by(month_name, PERP_AGE_GROUP) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(month_name) %>%
  mutate(percentage = 100 * count / sum(count)) %>%
  ungroup() %>%
  mutate(month_name = factor(month_name, levels = month.name))

# Plot
ggplot(perp_monthly, aes(x = month_name, y = percentage, color = PERP_AGE_GROUP, group = PERP_AGE_GROUP
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  scale_color_brewer(palette = "Set1") +
  labs(
    title = "Monthly Percentage of Perpetrator Age Groups",
    x = "Month",
    y = "Percentage",
    color = "Perpetrator Age Group"
  ) +
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



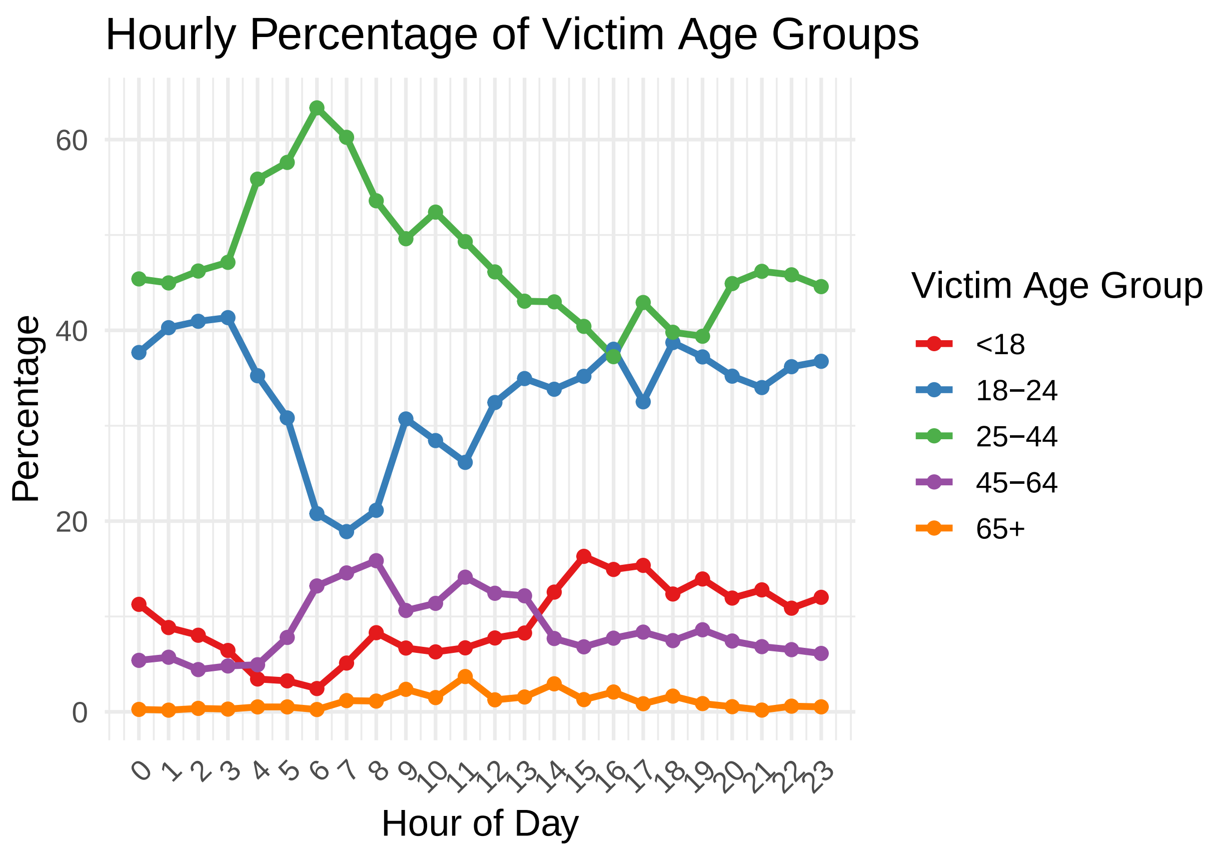
```
victim_monthly <- nypd_data %>%
  filter(!is.na(VIC_AGE_GROUP), VIC_AGE_GROUP %in% age_order) %>%
  mutate(VIC_AGE_GROUP = factor(VIC_AGE_GROUP, levels = age_order)) %>%
  group_by(month_name, VIC_AGE_GROUP) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(month_name) %>%
  mutate(percentage = 100 * count / sum(count)) %>%
  ungroup() %>%
  mutate(month_name = factor(month_name, levels = month.name))

# Plot
ggplot(victim_monthly, aes(x = month_name, y = percentage, color = VIC_AGE_GROUP, group = VIC_AGE_GROUP
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  scale_color_brewer(palette = "Set1") +
  labs(
    title = "Monthly Percentage of Victim Age Groups",
    x = "Month",
    y = "Percentage",
    color = "Victim Age Group"
  ) +
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



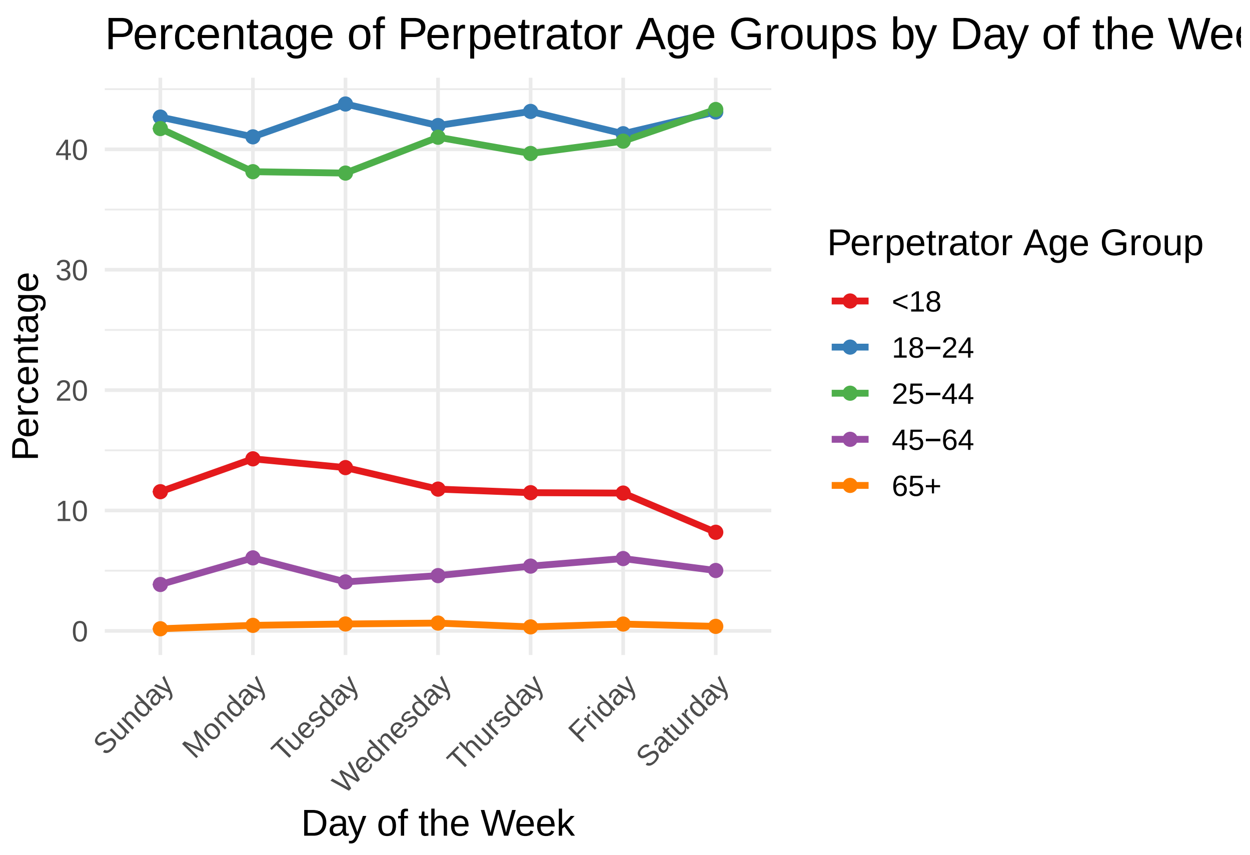
```
perp_hourly <- nypd_data %>%
  filter(!is.na(PERP_AGE_GROUP), PERP_AGE_GROUP %in% age_order) %>%
  mutate(PERP_AGE_GROUP = factor(PERP_AGE_GROUP, levels = age_order)) %>%
  group_by(hour_of_day, PERP_AGE_GROUP) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(hour_of_day) %>%
  mutate(percentage = 100 * count / sum(count)) %>%
  ungroup()

# Plot
ggplot(perp_hourly, aes(x = hour_of_day, y = percentage, color = PERP_AGE_GROUP, group = PERP_AGE_GROUP)
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  scale_color_brewer(palette = "Set1") +
  labs(
    title = "Hourly Percentage of Perpetrator Age Groups",
    x = "Hour of Day",
    y = "Percentage",
    color = "Perpetrator Age Group"
  ) +
  theme_minimal(base_size = 14) +
  scale_x_continuous(breaks = 0:23) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



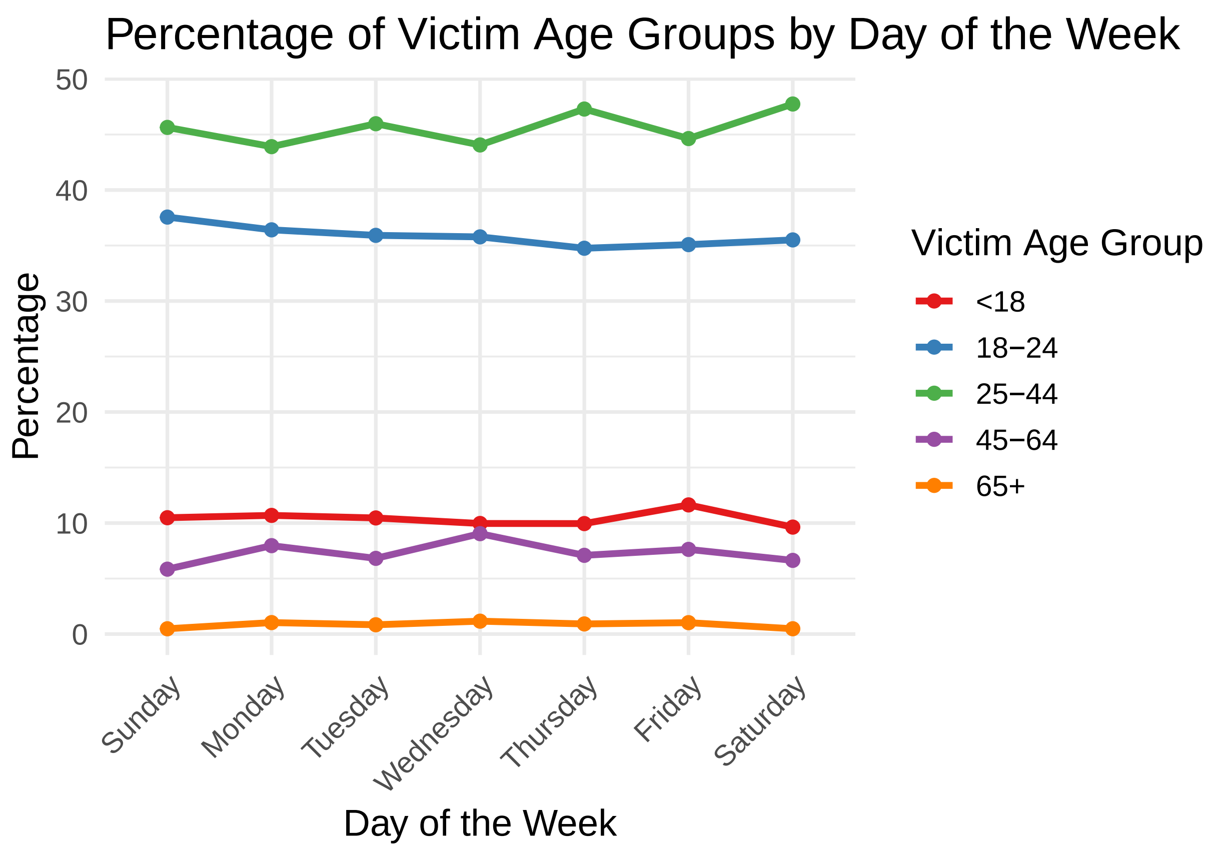
```r
victim_hourly <- nypd_data %>%
  filter(!is.na(VIC_AGE_GROUP), VIC_AGE_GROUP %in% age_order) %>%
  mutate(VIC_AGE_GROUP = factor(VIC_AGE_GROUP, levels = age_order)) %>%
  group_by(hour_of_day, VIC_AGE_GROUP) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(hour_of_day) %>%
  mutate(percentage = 100 * count / sum(count)) %>%
  ungroup()

# Plot
ggplot(victim_hourly, aes(x = hour_of_day, y = percentage, color = VIC_AGE_GROUP, group = VIC_AGE_GROUP)
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  scale_color_brewer(palette = "Set1") +
  labs(
    title = "Hourly Percentage of Victim Age Groups",
    x = "Hour of Day",
    y = "Percentage",
    color = "Victim Age Group"
  ) +
  theme_minimal(base_size = 14) +
  scale_x_continuous(breaks = 0:23) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```r
perp_day_week <- nypd_data %>%
  filter(!is.na(PERP_AGE_GROUP), PERP_AGE_GROUP %in% age_order) %>%
  mutate(PERP_AGE_GROUP = factor(PERP_AGE_GROUP, levels = age_order)) %>%
  group_by(day_of_week, PERP_AGE_GROUP) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(day_of_week) %>%
  mutate(percentage = 100 * count / sum(count)) %>%
  ungroup()

# Plot for perpetrators' age groups by day of the week
ggplot(perp_day_week, aes(x = day_of_week, y = percentage, color = PERP_AGE_GROUP, group = PERP_AGE_GROU
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  scale_color_brewer(palette = "Set1") +
  labs(
    title = "Percentage of Perpetrator Age Groups by Day of the Week",
    x = "Day of the Week",
    y = "Percentage",
    color = "Perpetrator Age Group"
  ) +
  theme_minimal(base_size = 14) +
  scale_x_discrete(limits = c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturda
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
# Create data for victim age groups by day of the week
victim_day_week <- nypd_data %>%
  filter(!is.na(VIC_AGE_GROUP), VIC_AGE_GROUP %in% age_order) %>%
  mutate(VIC_AGE_GROUP = factor(VIC_AGE_GROUP, levels = age_order)) %>%
  group_by(day_of_week, VIC_AGE_GROUP) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(day_of_week) %>%
  mutate(percentage = 100 * count / sum(count)) %>%
  ungroup()

# Plot for victims' age groups by day of the week
ggplot(victim_day_week, aes(x = day_of_week, y = percentage, color = VIC_AGE_GROUP, group = VIC_AGE_GROU
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  scale_color_brewer(palette = "Set1") +
  labs(
    title = "Percentage of Victim Age Groups by Day of the Week",
    x = "Day of the Week",
    y = "Percentage",
    color = "Victim Age Group"
  ) +
  theme_minimal(base_size = 14) +
  scale_x_discrete(limits = c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturda
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```


Percentage of Victim Age Groups by Day of the Week
Percentage
50
40
30
20
10
0
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of the Week
Victim Age Group
<18
18−24
25−44
45−64
65+