Analyzing Global Terrorism Trends: Patterns, Lethality, and Forecasting (1970-2017)

Group 14

2025-04-11

This contains the loading of libraries and the two datasets: global terrorism and UN population. Then we process and clean the data by handling outliers and missing values, and performing some feature engineering, which includes renaming columns in terrorism dataset, creating a decade variable, creating a severity index using the nkill and nwound columns.

```
# Load required libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(readr)
library(ggmap)
```

```
## i Google's Terms of Service: <https://mapsplatform.google.com>
##   Stadia Maps' Terms of Service: <https://stadiamaps.com/terms-of-service/>
##   OpenStreetMap's Tile Usage Policy: <https://operations.osmfoundation.org/policies/tiles/>
## i Please cite ggmap if you use it! Use `citation("ggmap")` for details.
```

```
library(rworldmap)
```

```
## Loading required package: sp
## ### Welcome to rworldmap ###
## For a short introduction type :   vignette('rworldmap')
```

```
library(arules)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
##
## Attaching package: 'arules'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arulesViz)
library(ggpubr)
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:arules':
##
##     recode
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(caret)
```

```
## Loading required package: lattice
##
```

```
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
##
## Attaching package: 'forecast'
##
## The following object is masked from 'package:ggpubr':
##
##     gghistogram
```

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(ggfortify)
```

```
## Registered S3 methods overwritten by 'ggfortify':
##   method                 from
##   autoplot.Arima         forecast
##   autoplot.acf           forecast
##   autoplot.ar            forecast
##   autoplot.bats          forecast
##   autoplot.decomposed.ts forecast
##   autoplot.ets           forecast
##   autoplot.forecast      forecast
##   autoplot.stl           forecast
##   autoplot.ts            forecast
##   fitted.ar              forecast
##   fortify.ts             forecast
##   residuals.ar           forecast
```

```
library(naniar) # For missing value visualization

# Load datasets
fulldf <- read_csv("D:\\Stony Brook\\Statistical Computing Project\\globalterrorismdb_0718dist.csv")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 181691 Columns: 135
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (55): approxdate, resolution, country_txt, region_txt, provstate, city, ...
## dbl (75): eventid, iyear, imonth, iday, extended, country, region, latitude,...
## lgl  (5): gsubname3, weaptype4, weaptype4_txt, weapsubtype4, weapsubtype4_txt
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
fullpop <- read_csv("D:\\Stony Brook\\Statistical Computing Project\\UNpopfile.csv")
```

```
## Rows: 371007 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Location, Variant
## dbl (7): LocID, VarID, Time, MidPeriod, PopMale, PopFemale, PopTotal
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Process population data
pop <- fullpop %>%
  select(-MidPeriod, -PopMale, -PopFemale, -VarID) %>%
  filter(Time > 1969 & Variant == 'Medium' & Time < 2017) %>%
  select(-Variant, -LocID)

futurepop <- fullpop %>%
  filter(Time > 2016 & Variant == 'Medium') %>%
  select(-Variant, -LocID, -MidPeriod, -PopMale, -PopFemale, -VarID)

# Select and rename columns in terrorism dataset
df <- fulldf %>%
  select(iyear, imonth, iday, country_txt, region_txt, city, latitude, longitude,
         summary, multiple, attacktype1_txt, targtype1_txt, targsubtype1_txt,
         gname, weaptype1_txt, nkill, nwound, nkillter) %>%
  rename(year = iyear, month = imonth, day = iday, country = country_txt,
         region = region_txt, multiple_attack = multiple, attacktype = attacktype1_txt,
         target_type = targtype1_txt, target_sub_type = targsubtype1_txt,
         group_name = gname, weapon_type = weaptype1_txt)

# Create decade variable
df <- df %>%
  mutate(decade =
           ifelse(year<1980, '70s',
                  ifelse(year < 1990, '80s',
                         ifelse(year < 2000, '90s',
                                ifelse(year < 2010, '2000s', '2010s')))))

df$decade <- factor(df$decade, levels=c("70s", "80s", "90s", "2000s", "2010s"))

# Check for missing values
missing_values <- colSums(is.na(df))
print(missing_values)
```

```
##            year           month             day         country          region
##               0               0               0               0               0
##            city        latitude       longitude         summary multiple_attack
##             434            4556            4557           66129               1
##      attacktype     target_type target_sub_type      group_name     weapon_type
##               0               0           10373               0               0
##           nkill          nwound        nkillter          decade
##           10313           16311           66958               0
```

```
# Handle missing values
df <- df %>%
  # Replace NA in categorical columns with "Unknown"
  mutate(group_name = ifelse(is.na(group_name), "Unknown", group_name)) %>%
  mutate(city = ifelse(is.na(city), "Unknown", city)) %>%
  mutate(target_sub_type = ifelse(is.na(target_sub_type), "Unknown", target_sub_type)) %>%
  mutate(summary = ifelse(is.na(summary), "No summary available", summary)) %>%
  # Replace NA in multiple_attack with 0 (assuming 0 means no multiple attack)
  mutate(multiple_attack = ifelse(is.na(multiple_attack), 0, multiple_attack)) %>%
  # Replace NA in numerical columns with median values
  mutate(
    nkill = ifelse(is.na(nkill), median(nkill, na.rm = TRUE), nkill),
    nwound = ifelse(is.na(nwound), median(nwound, na.rm = TRUE), nwound),
    nkillter = ifelse(is.na(nkillter), median(nkillter, na.rm = TRUE), nkillter)
  )

# Handle missing coordinates using country/region medians
df <- df %>%
  # Create a flag for missing coordinates
  mutate(coords_missing = is.na(latitude) | is.na(longitude))

# Impute missing coordinates with country medians
df <- df %>%
  group_by(country) %>%
  mutate(
    latitude = ifelse(is.na(latitude), median(latitude, na.rm = TRUE), latitude),
    longitude = ifelse(is.na(longitude), median(longitude, na.rm = TRUE), longitude)
  ) %>%
  ungroup()

# For countries with all NA coordinates, use region medians
df <- df %>%
  group_by(region) %>%
  mutate(
    latitude = ifelse(is.na(latitude), median(latitude, na.rm = TRUE), latitude),
    longitude = ifelse(is.na(longitude), median(longitude, na.rm = TRUE), longitude)
  ) %>%
  ungroup()

# If there are still NAs, replace with global medians
global_lat_median <- median(df$latitude, na.rm = TRUE)
global_long_median <- median(df$longitude, na.rm = TRUE)
```

```r
df <- df %>%
  mutate(
    latitude = ifelse(is.na(latitude), global_lat_median, latitude),
    longitude = ifelse(is.na(longitude), global_long_median, longitude)
  )

# Handle outliers using capping method for numerical columns
cap_outliers <- function(x, lower_quantile = 0.05, upper_quantile = 0.95) {
  qnt <- quantile(x, probs = c(lower_quantile, upper_quantile), na.rm = TRUE)
  x[x < qnt[1]] <- qnt[1]
  x[x > qnt[2]] <- qnt[2]
  return(x)
}

# Only apply capping to columns with extreme outliers
df <- df %>%
  mutate(
    nkill_capped = cap_outliers(nkill),
    nwound_capped = cap_outliers(nwound),
    nkillter_capped = cap_outliers(nkillter)
  )

# Create additional features for analysis
df <- df %>%
  # Create date column (handling missing month/day values)
  mutate(
    month = ifelse(month == 0 | is.na(month), 1, month),
    day = ifelse(day == 0 | is.na(day), 1, day),
    date = as.Date(paste(year, month, day, sep = "-"), format = "%Y-%m-%d")
  ) %>%
  # Create severity index
  mutate(
    severity_index = nkill + 0.5 * nwound,
    severity_category = case_when(
      severity_index == 0 ~ "No casualties",
      severity_index <= 5 ~ "Low",
      severity_index <= 20 ~ "Medium",
      severity_index <= 50 ~ "High",
      TRUE ~ "Extreme"
    )
  )

# Create a version of the dataset with both original and capped values for analysis
df_analysis <- df

# Check the result of data cleaning
glimpse(df_analysis)
```

```
## Rows: 181,691
## Columns: 26
## $ year              <dbl> 1970, 1970, 1970, 1970, 1970, 1970, 1970, 1970, 1970~
## $ month             <dbl> 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ day               <dbl> 2, 1, 1, 1, 1, 1, 2, 2, 2, 3, 1, 6, 8, 9, 9, 10, 11,~
```

```
## $ country           <chr> "Dominican Republic", "Mexico", "Philippines", "Gree~
## $ region            <chr> "Central America & Caribbean", "North America", "Sou~
## $ city              <chr> "Santo Domingo", "Mexico city", "Unknown", "Athens",~
## $ latitude          <dbl> 18.456792, 19.371887, 15.478598, 37.997490, 33.58041~
## $ longitude         <dbl> -69.95116, -99.08662, 120.59974, 23.76273, 130.39636~
## $ summary           <chr> "No summary available", "No summary available", "No ~
## $ multiple_attack   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ attacktype        <chr> "Assassination", "Hostage Taking (Kidnapping)", "Ass~
## $ target_type       <chr> "Private Citizens & Property", "Government (Diplomat~
## $ target_sub_type   <chr> "Named Civilian", "Diplomatic Personnel (outside of ~
## $ group_name        <chr> "MANO-D", "23rd of September Communist League", "Unk~
## $ weapon_type       <chr> "Unknown", "Unknown", "Unknown", "Explosives", "Ince~
## $ nkill             <dbl> 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0~
## $ nwound            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ nkillter          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ decade            <fct> 70s, 70s, 70s, 70s, 70s, 70s, 70s, 70s, 70s, 70s, 70~
## $ coords_missing    <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL~
## $ nkill_capped      <dbl> 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0~
## $ nwound_capped     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ nkillter_capped   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ date              <date> 1970-07-02, 1970-01-01, 1970-01-01, 1970-01-01, 197~
## $ severity_index    <dbl> 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.~
## $ severity_category <chr> "Low", "No casualties", "Low", "No casualties", "No ~
```

```
# Verify missing values after cleaning
missing_after <- colSums(is.na(df_analysis))
print(missing_after)
```

```
##              year             month               day           country
##                 0                 0                 0                 0
##            region              city          latitude         longitude
##                 0                 0                 0                 0
##           summary   multiple_attack        attacktype       target_type
##                 0                 0                 0                 0
##   target_sub_type        group_name       weapon_type             nkill
##                 0                 0                 0                 0
##            nwound          nkillter            decade    coords_missing
##                 0                 0                 0                 0
##      nkill_capped     nwound_capped   nkillter_capped              date
##                 0                 0                 0                 0
##    severity_index severity_category
##                 0                 0
```

```
# After fully cleaning the data
nrow(df_analysis)
```

```
## [1] 181691
```

EXPLORATORY DATA ANALYSIS (EDA)

1. Data Overview

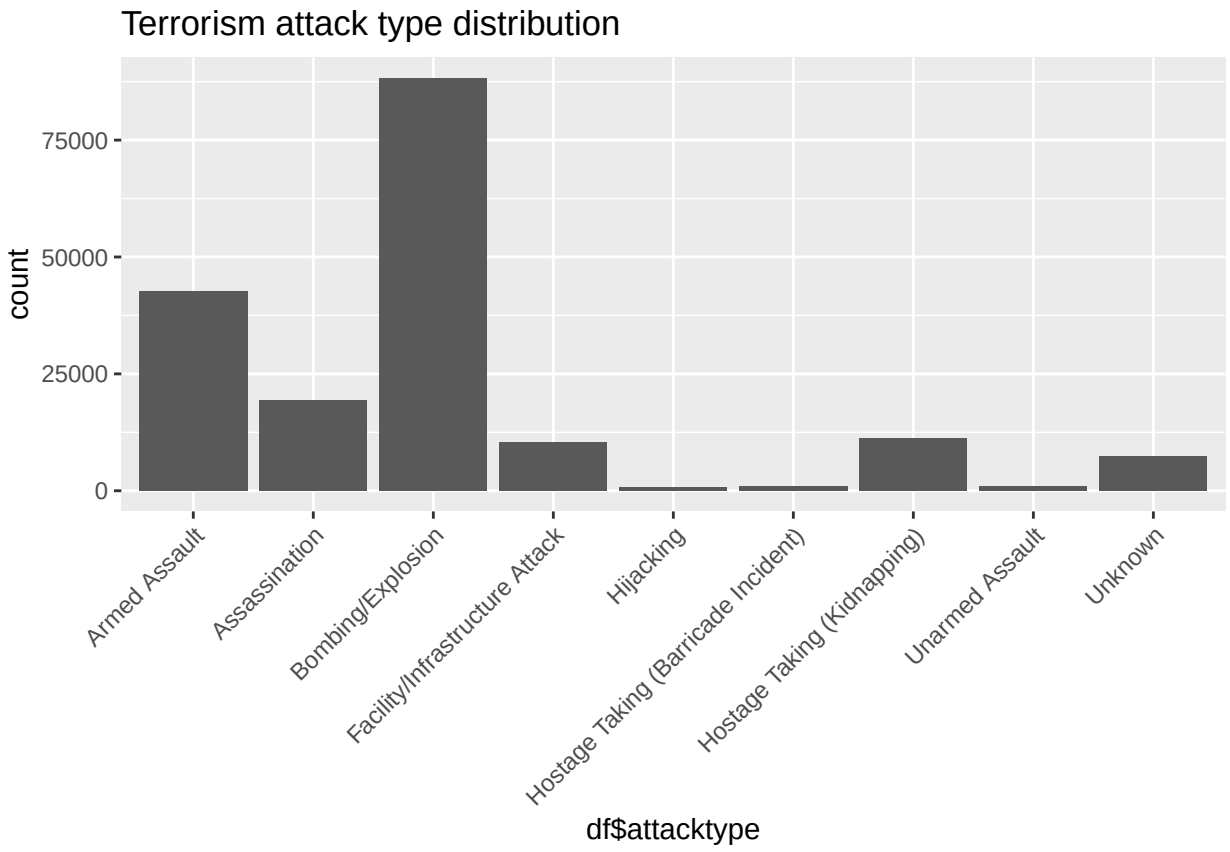
#1.1 Attack type Distribution

Depicting the distribution of various types of attacks, using the ggplot library in the form of a histogram. We can see how much more the count of bombing/explosion is compared to others, with hijacking having the lowest count. Hostage taking and uhnarmed assault also have one of the lowest counts.

```
ggplot(data = df, aes(x = df$attacktype)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_histogram(stat = "count") +
  labs(title='Terrorism attack type distribution')
```

```
## Warning in geom_histogram(stat = "count"): Ignoring unknown parameters:
## `binwidth`, `bins`, and `pad`
```

```
## Warning: Use of `df$attacktype` is discouraged.
## i Use `attacktype` instead.
```

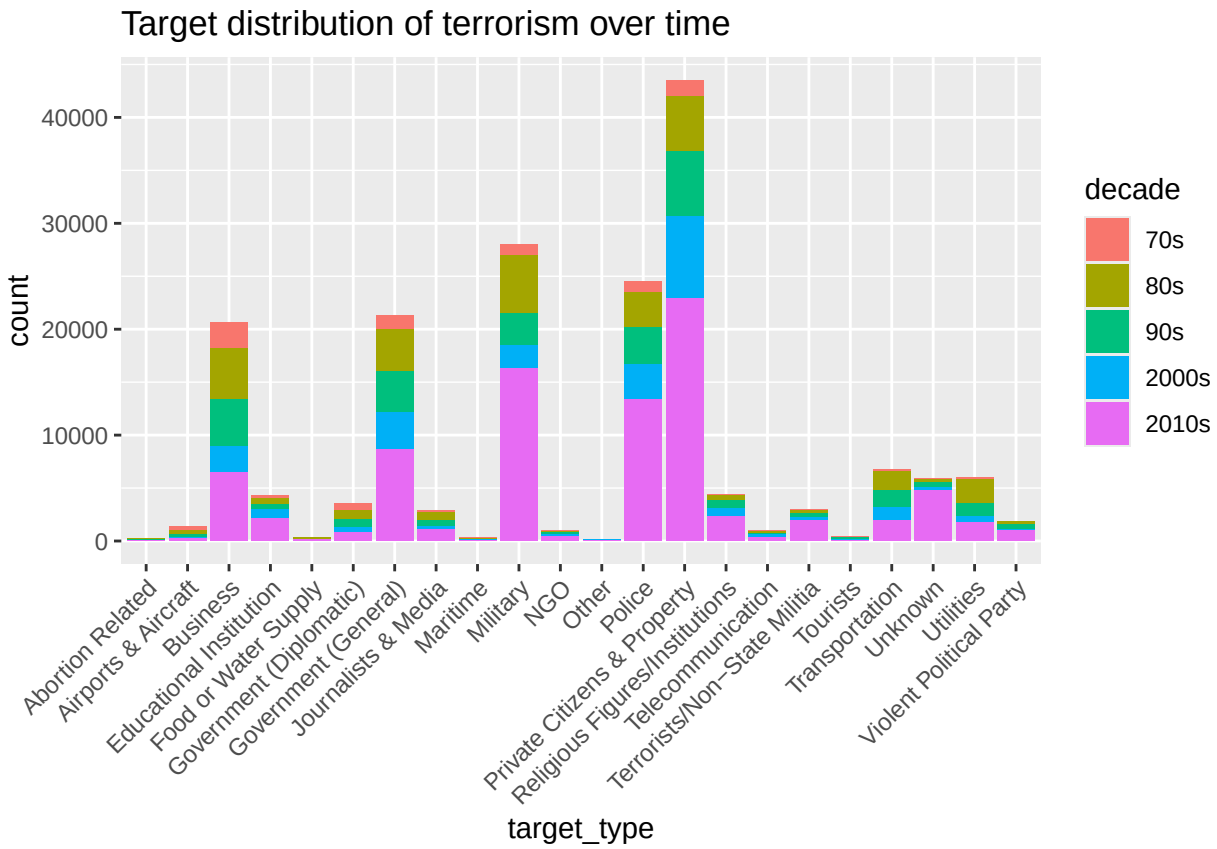


#1.2 Target Distribution

Let's get an idea of what kind of targets terrorists hit. The visualization graph depicts a histogram, containing the target distribution of terrorism. The demographic/group most targeted are the private citizens and property, with the military coming a close second. The lowest count is attributed to the educational institutions, with number of attacks being a mere 4322 compared to the 43511 of private citizens and property.

```
#visual
ggplot(data=df, aes(x=target_type, fill=decade)) +
  geom_histogram(stat='count') +
  theme(axis.text.x= element_text(angle=45, hjust=1)) +
  labs(title='Target distribution of terrorism over time')
```

```
## Warning in geom_histogram(stat = "count"): Ignoring unknown parameters:
## `binwidth`, `bins`, and `pad`
```




```
#table
df %>%
  group_by(target_type) %>%
  summarise(nr_of_attacks = n()) %>%
  arrange(desc(nr_of_attacks)) %>%
  head(n=10)
```

```
## # A tibble: 10 x 2
##    target_type                    nr_of_attacks
##    <chr>                                  <int>
##  1 Private Citizens & Property            43511
##  2 Military                               27984
##  3 Police                                 24506
##  4 Government (General)                   21283
##  5 Business                               20669
##  6 Transportation                          6799
##  7 Utilities                               6023
##  8 Unknown                                 5898
##  9 Religious Figures/Institutions          4440
## 10 Educational Institution                 4322
```

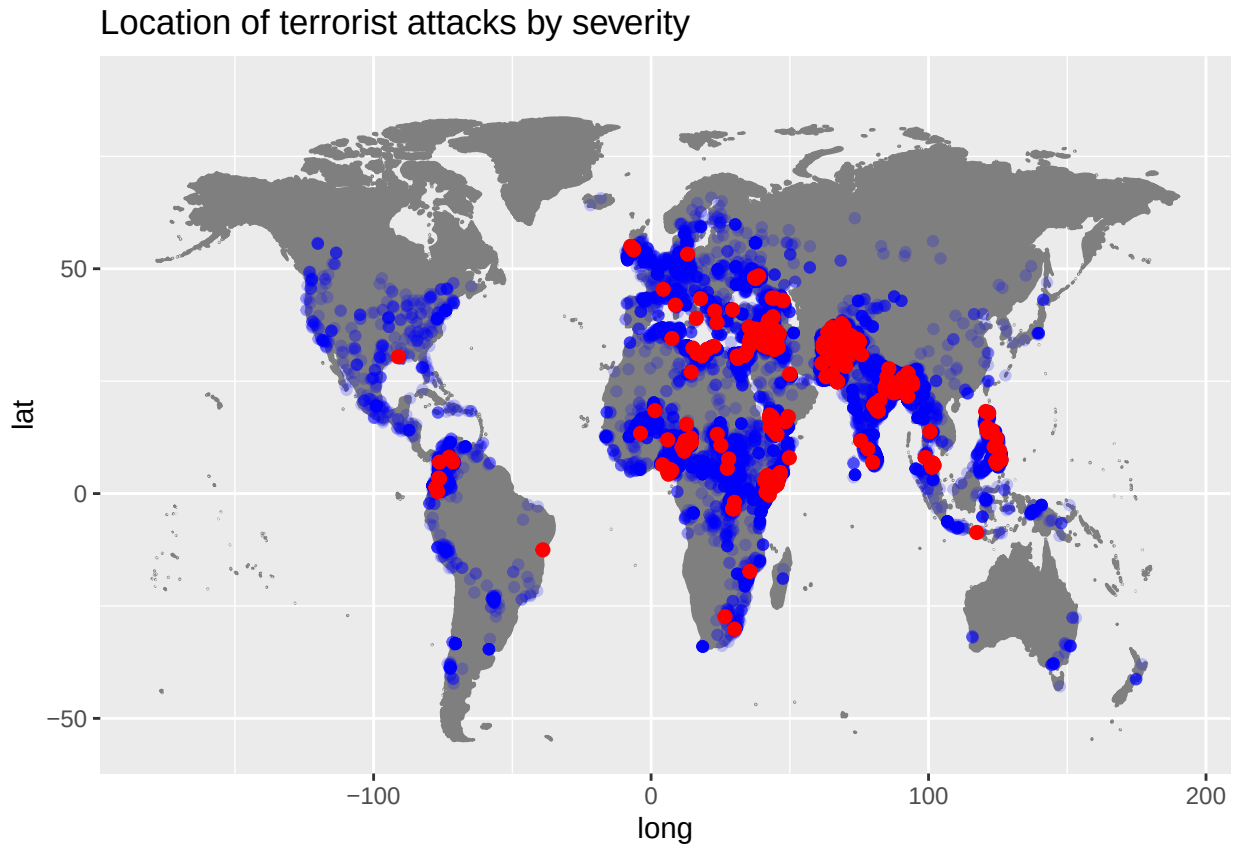
#1.3 location of terrorism (regions/countries/cities)

We want to see where the terrorist attacks happen around the world. For this we'll use the ggplot package. The results signify two colours' dots. They show the parts of the world where terrorism attacks have happened. The red dots mean deadlier killings compared to blue ones. This plot does not include the terror attacks prior to 2007. It only accounts for the attacks over the last decade (2007-2017).

```
# For plotting clarity lets just check out attacks from the last decade and onwards.
df2000 <- df %>%
  filter(year > 2006)

world <- borders("world", colour="gray50", fill="gray50")
worldmap <- ggplot() + world + scale_y_continuous(limits=c(-55, 90))

worldmap +
  geom_point(aes(x=df2000$longitude[df$nkill<51], y=df2000$latitude[df$nkill<51]), col='blue', alpha= 0
  geom_point(aes(x=df2000$longitude[df$nkill>50], y=df2000$latitude[df$nkill>50]), col='red', size=2) +
  labs(title='Location of terrorist attacks by severity')
```

```
## Warning: Removed 81819 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 289 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



1.4 Location distribution

Let's view the top 10 locations for terrorist attacks by region, country and city. The top location for terror attacks is Baghdad in Iraq. It has a count of a whopping 7589, almost 5000 count more than the second most attacked city, Karachi in Pakistan. This analysis is done taking into account the regions, countries and cities mentioned in the dataset.

```
df %>%
  group_by(region) %>%
  summarise( nr_of_attacks = n()) %>%
  arrange(desc(nr_of_attacks)) %>%
  head(n=10)
```

```
## # A tibble: 10 x 2
##    region                       nr_of_attacks
##    <chr>                                <int>
##  1 Middle East & North Africa           50474
##  2 South Asia                           44974
##  3 South America                        18978
##  4 Sub-Saharan Africa                   17550
##  5 Western Europe                       16639
##  6 Southeast Asia                       12485
##  7 Central America & Caribbean          10344
##  8 Eastern Europe                        5144
##  9 North America                         3456
## 10 East Asia                              802
```

```
df %>%
  group_by(country) %>%
  summarise( nr_of_attacks = n()) %>%
  arrange(desc(nr_of_attacks)) %>%
  head(n=10)
```

```
## # A tibble: 10 x 2
##    country        nr_of_attacks
##    <chr>                  <int>
##  1 Iraq                   24636
##  2 Pakistan               14368
##  3 Afghanistan            12731
##  4 India                  11960
##  5 Colombia                8306
##  6 Philippines             6908
##  7 Peru                    6096
##  8 El Salvador             5320
##  9 United Kingdom          5235
## 10 Turkey                  4292
```

```
df %>%
  filter(city != 'Unknown') %>%
  group_by(city) %>%
  summarise( nr_of_attacks = n()) %>%
  arrange(desc(nr_of_attacks)) %>%
  head(n=10)
```

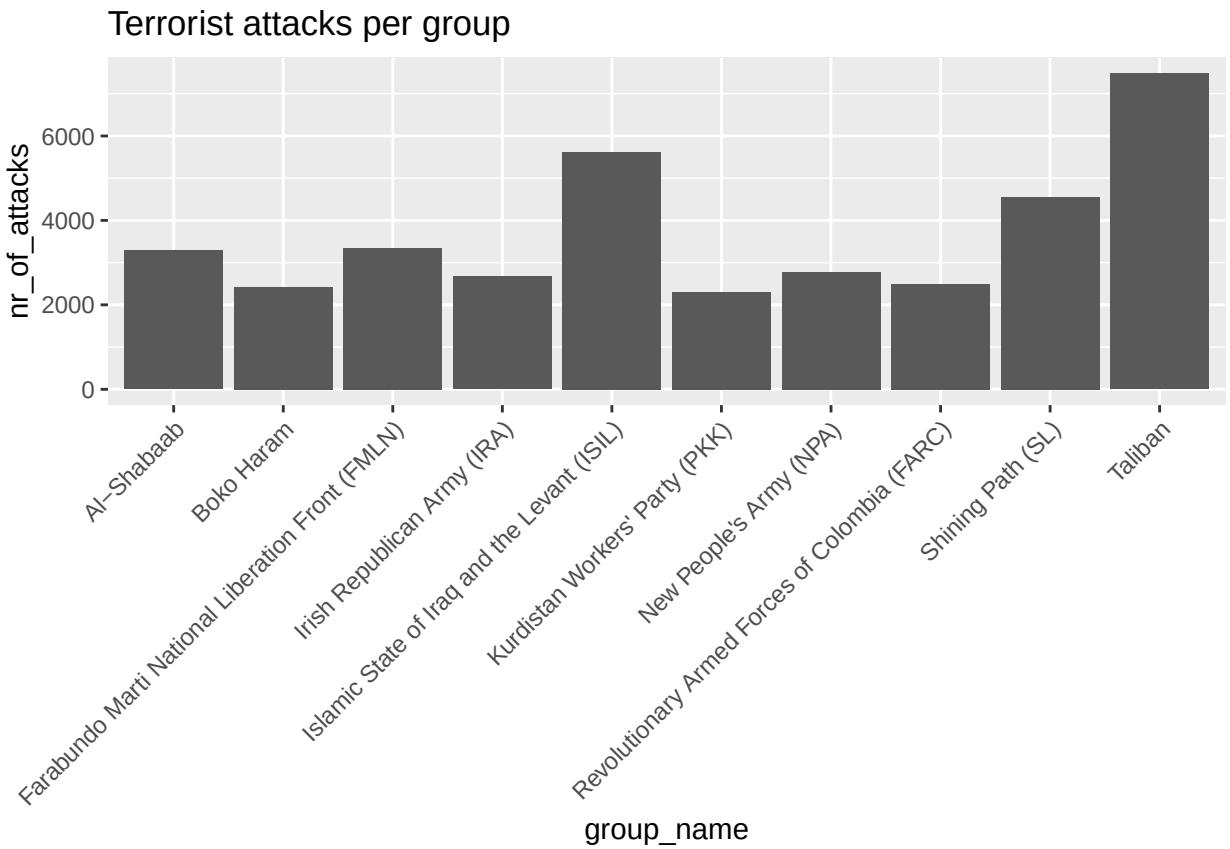
```
## # A tibble: 10 x 2
##    city         nr_of_attacks
##    <chr>                <int>
##  1 Baghdad               7589
##  2 Karachi               2652
##  3 Lima                  2359
##  4 Mosul                 2265
##  5 Belfast               2171
##  6 Santiago              1621
##  7 Mogadishu             1581
##  8 San Salvador          1558
##  9 Istanbul              1048
## 10 Athens                1019
```

1.5 distribution of terrorist groups

We study which terrorist groups are doing these attacks, in the form of a histogram using the ggplot library. The group that committed the highest number of terrorist attacks happens to be the Taliban, with the count topping over 7000 attacks. The lowest attacks are done by the Boko Haram and Kurdistan Workers' Party. However, these are only the top 10 groups accounted for in this plot.

```
#table
top10_groups <- df %>%
  filter(group_name != "Unknown") %>%
  group_by(group_name) %>%
  summarise(nr_of_attacks = n()) %>%
  arrange(desc(nr_of_attacks)) %>%
  head(n=10)

#visual
ggplot(data=top10_groups) +
  stat_summary(aes(x=group_name, y=nr_of_attacks), geom="bar") +
  theme(axis.text.x= element_text(angle=45, hjust=1)) +
  labs(title='Terrorist attacks per group')
```

```
## No summary function supplied, defaulting to `mean_se()`
```



#2. Trends in terorrism ##2.1 Terrorism growth

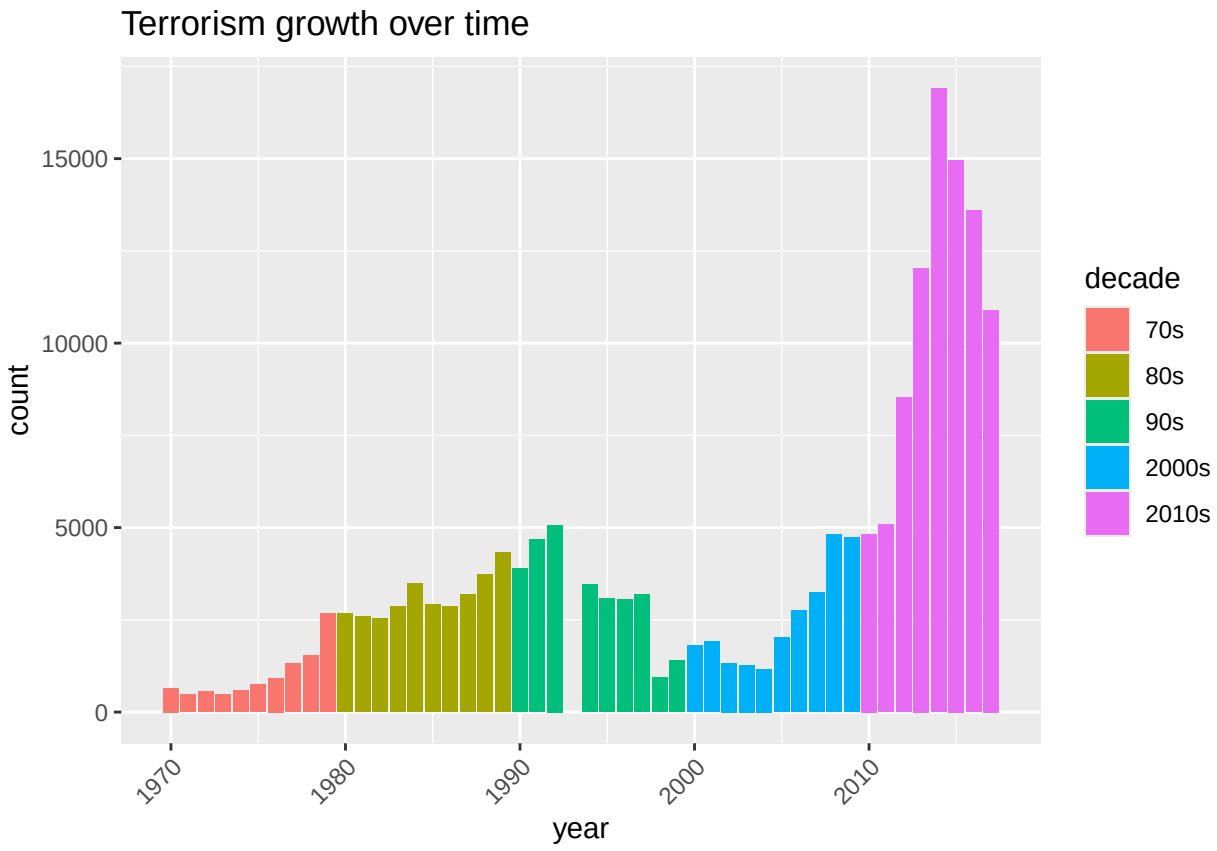
See below for decade breakdown - significant increase since 2010. THe years range froms the 70s to the 2010s. We use ggplot library to plot a histogram of the increase in terrorism attacks over the years. With the lowest count in 1971 and 1973, the highest count is a stark contrast, reaching a peak in the year 2014. Many factors contribute to this statistic, including population growth.

```
#table
df %>%
  group_by(decade) %>%
  summarise(nr_of_attacks = n()) %>%
  arrange(desc(nr_of_attacks)) %>%head(n=10)
```

```
## # A tibble: 5 x 2
##   decade nr_of_attacks
##   <fct>          <int>
## 1 2010s          86815
## 2 80s            31160
## 3 90s            28762
## 4 2000s          25040
## 5 70s             9914
```

```
#visual
ggplot(data=df, aes(x=year, fill=decade)) +
  geom_histogram(stat='count') +
  theme(axis.text.x= element_text(angle=45, hjust=1)) +
  labs(title='Terrorism growth over time')
```

```
## Warning in geom_histogram(stat = "count"): Ignoring unknown parameters:
## `binwidth`, `bins`, and `pad`
```



2.2 Locations of terrorism

This is the histogram plot of the top 20 countries attacked by terrorists. Middle East and North Africa (~27% of total), South Asia (~24%) and South America (11%) are the top three regions in terms of number of attacks. Iraq (~12.9% of total), Pakistan(~8%), Afghanistan (~6.6%), India(~6.4%) and Colombia (4.7%) are the top five countries in terms of number of attacks.

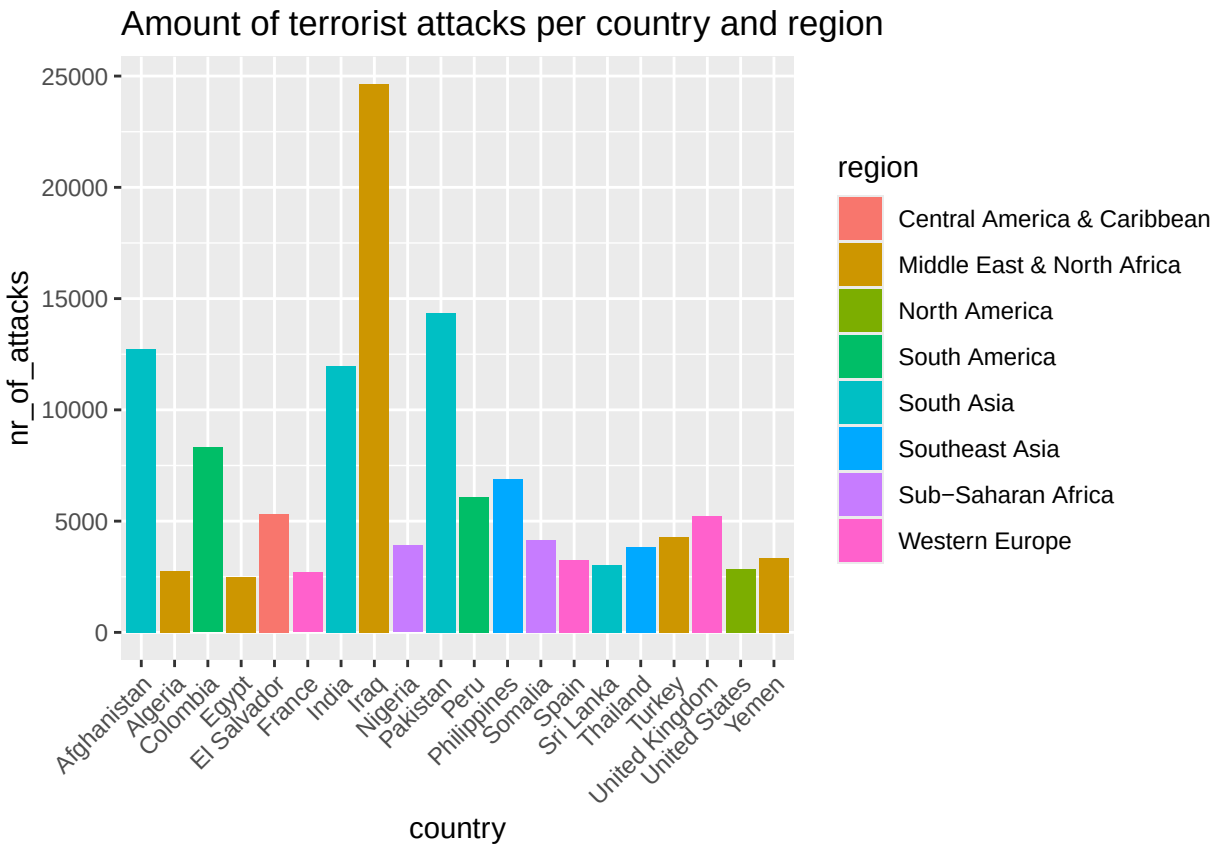
```
#table
top20_countries <- df %>%
  group_by(region, country) %>%
  summarise(nr_of_attacks = n()) %>%
  mutate(percent = nr_of_attacks/sum(nr_of_attacks))%>%
  arrange(desc(nr_of_attacks)) %>%
  head(n=20)
```

```
## `summarise()` has grouped output by 'region'. You can override using the
## `.groups` argument.
```

```
#visual by country
ggplot(data=top20_countries) +
```

```
  stat_summary(aes(x=country, y=nr_of_attacks, fill=region), geom="bar") +
  theme(axis.text.x= element_text(angle=45, hjust=1)) +
  labs(title='Amount of terrorist attacks per country and region')
```

```
## No summary function supplied, defaulting to `mean_se()`
```



2.3 Has terrorism become deadlier?

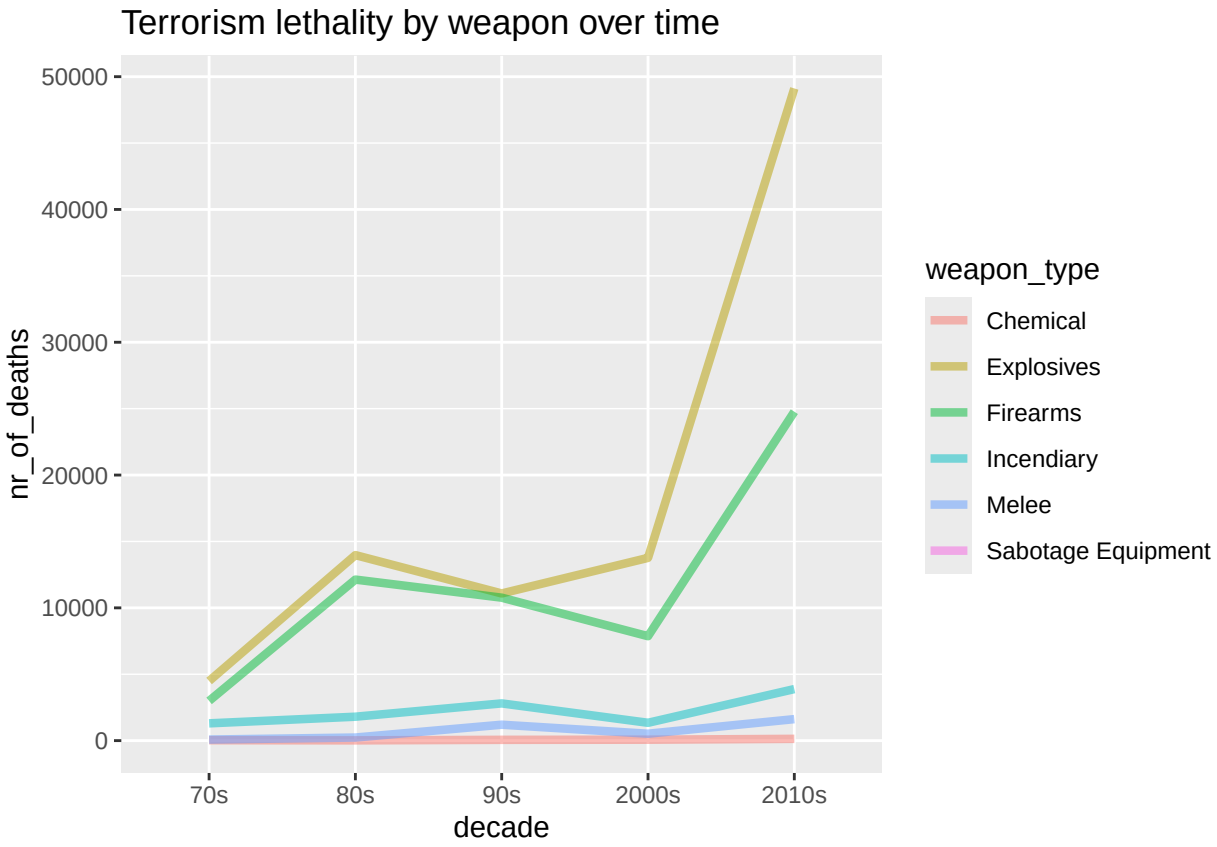
The plot depicts how attacks have become more and more deadly over the years. Explosive attacks have increased exponentially. Over half of all deaths by terrorist attack have occurred during bomb attacks. The next deadliest weapon grouping is "Firearms" responsibile for ~32% of all Terror attack deaths. Lowest is accounted for the sabotage equipment.

```
#table
weapon_lethality <- df %>%
  filter(weapon_type != "Unknown") %>%
  select(decade, weapon_type, nkill)%>%
  group_by(decade,weapon_type)%>%
  summarise(nr_of_deaths = n())%>%
  top_n(n=5, wt=nr_of_deaths) %>%
  mutate(percent_deaths = (nr_of_deaths/sum(nr_of_deaths)*100))
```

```
## `summarise()` has grouped output by 'decade'. You can override using the
## `.groups` argument.
```

```r
#Visual by decade / weapon type
ggplot(data=weapon_lethality, aes(x=decade, y=nr_of_deaths, col=weapon_type, group= weapon_type)) +
  geom_line(size=1.5, alpha=0.5) +
  labs(title='Terrorism lethality by weapon over time')
```

```
## 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.
```



2.4 Activity of groups over time

First we identify the top ten Terror Groups in terms of number of attacks. This plot is different than a histogram as it shows the exact increase of decrease of activity. Here we see how the attacks by Islamic State of Iraq and the Levant (ISIL) have grown exponentially in the 2010s.

```r
top10_groups <- df %>%
  filter(group_name != "Unknown") %>%
  group_by(group_name) %>%
  summarise(nr_of_attacks = n()) %>%
  arrange(desc(nr_of_attacks)) %>%
  head(n=10)
```

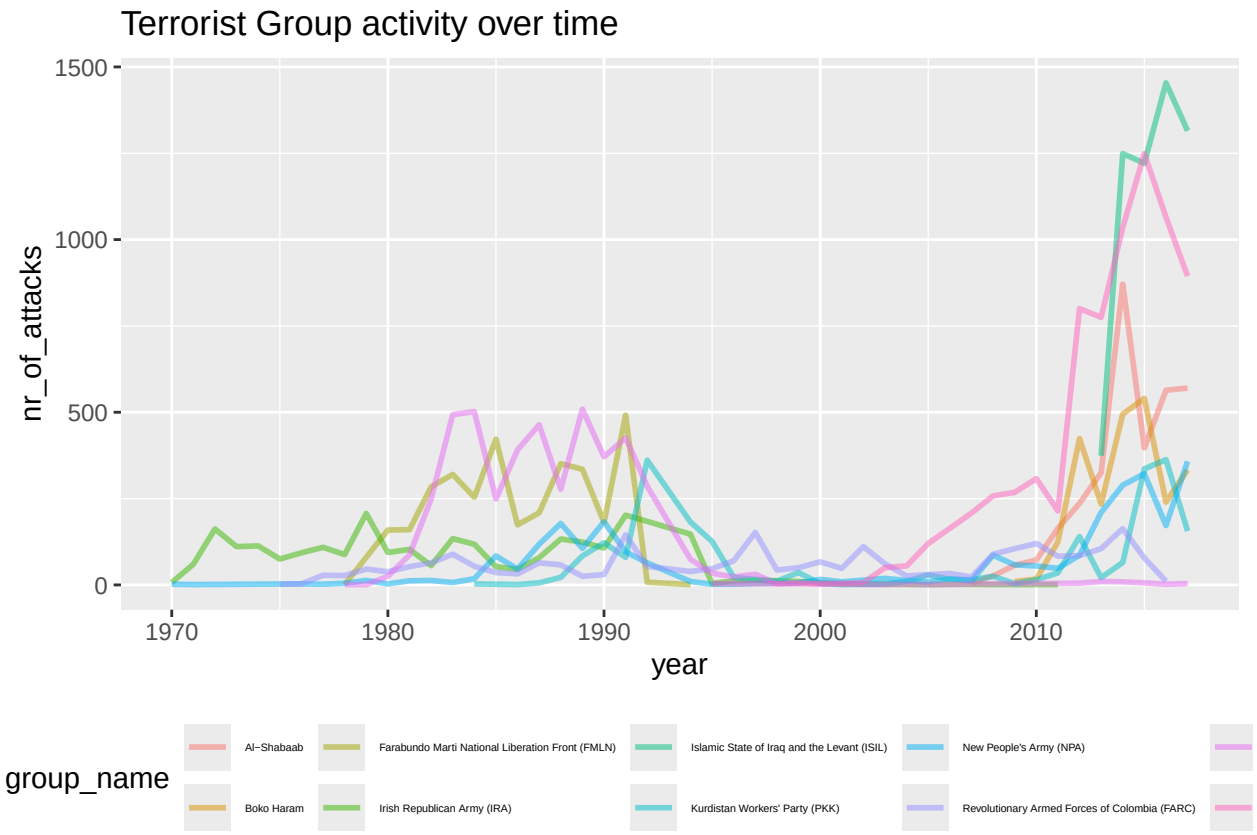
```
top10_groups
```

```
## # A tibble: 10 x 2
##    group_name                                          nr_of_attacks
##    <chr>                                                       <int>
##  1 Taliban                                                      7478
##  2 Islamic State of Iraq and the Levant (ISIL)                  5613
##  3 Shining Path (SL)                                            4555
##  4 Farabundo Marti National Liberation Front (FMLN)             3351
##  5 Al-Shabaab                                                   3288
##  6 New People's Army (NPA)                                      2772
##  7 Irish Republican Army (IRA)                                  2671
##  8 Revolutionary Armed Forces of Colombia (FARC)                2487
##  9 Boko Haram                                                   2418
## 10 Kurdistan Workers' Party (PKK)                               2310
```

```
#table
top10_groups_activity <- df %>%
filter(df$group_name %in% c("Taliban", "Shining Path (SL)", "Islamic State of Iraq and the Levant (ISIL)
select(year, group_name)%>%
group_by(year, group_name) %>%
  summarise(nr_of_attacks = n())%>%
  arrange(desc(nr_of_attacks))%>%
   top_n(n=10, wt=nr_of_attacks)
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
#Visual by Top 10 Terror Group Activity  / decade since 1970
ggplot(data=top10_groups_activity, aes(x=year, y=nr_of_attacks, col=group_name, group= group_name)) +
  geom_line(size=1, alpha=0.5) +
  theme(legend.position="right")+
  labs(title='Terrorist Group activity over time') +
  theme(legend.position="bottom", legend.text=element_text(size=3.5))
```



2.5 Weapon choice over time

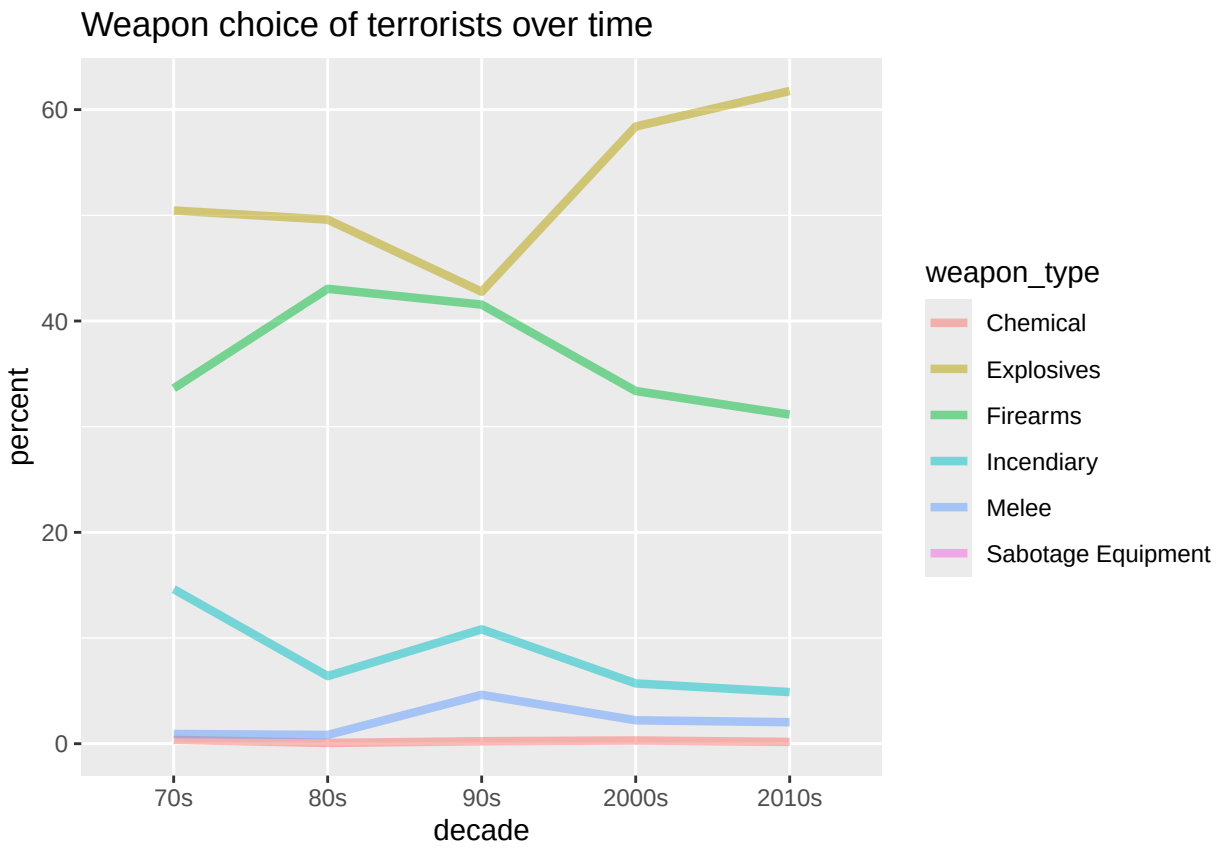
Did technology growth change what weapons terrorist use? Explosives and firearms remain the top weapons throughout the years. The lowest (almost non existent) weapons are chemical weapons, which are probably capable of wiping out the entire human race, are non-arguably the least used.

```
dfweapons <- df %>%
  select(year, weapon_type, decade) %>%
  filter(weapon_type != "Unknown")

#table
top15_weapons <- dfweapons %>%
  group_by(decade, weapon_type) %>%
  summarise(nr_of_attacks = n()) %>%
  top_n(n=5, wt=nr_of_attacks) %>%
  mutate(percent = nr_of_attacks/sum(nr_of_attacks)*100) %>%
  arrange(decade, desc(nr_of_attacks))
```

```
## `summarise()` has grouped output by 'decade'. You can override using the
## `.groups` argument.
```

```
#visual
ggplot(data=top15_weapons, aes(x=decade, y=percent, col=weapon_type, group= weapon_type)) +
```

```
  geom_line(size=1.5, alpha=0.5) +
  labs(title='Weapon choice of terrorists over time')
```



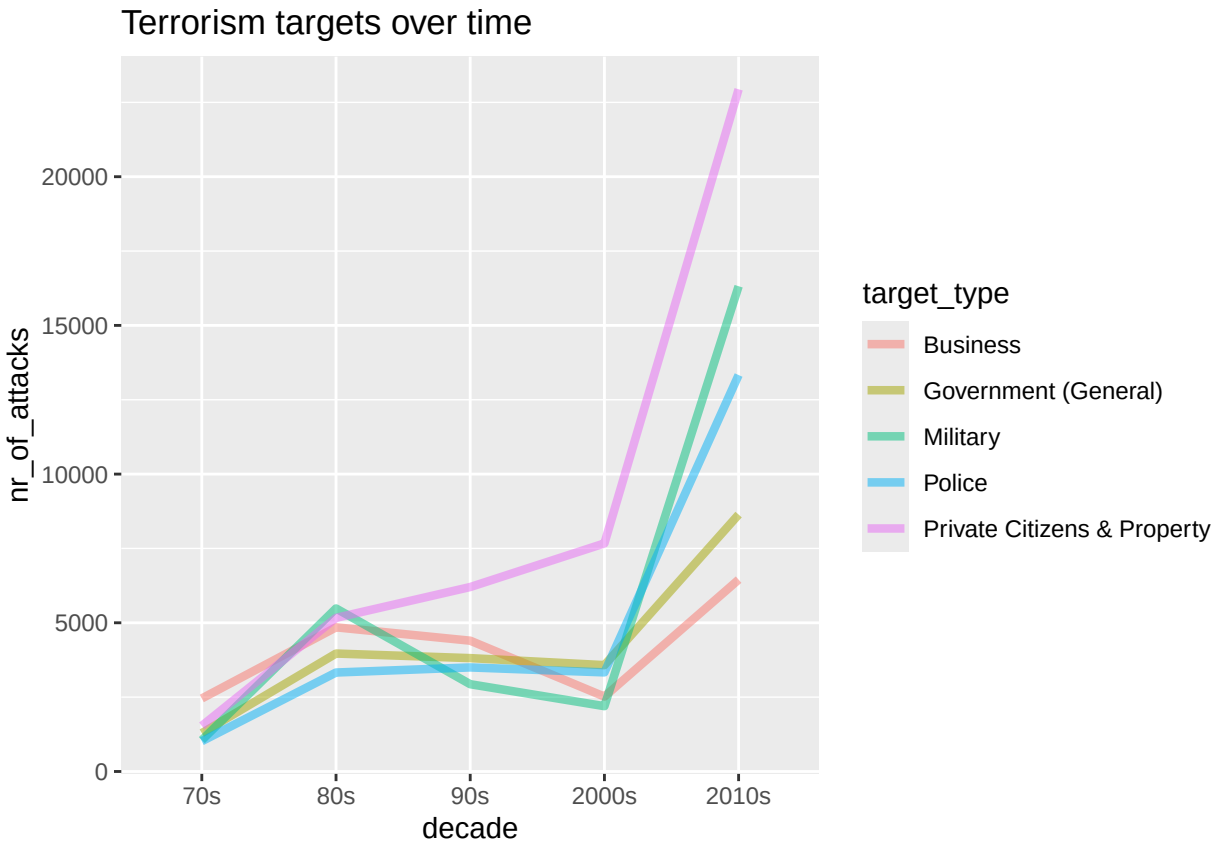
2.6 Target choice over time

Have the targets changed? Have terrorists changed what targets they use? The graph depicts that the top targets in the 70s were businesses, which were overtaken by the private citizens and property by the end (2010s). Military attacks were pretty low too but now they are the 2nd most attacked groups.

```
dftargets <- df %>%
  select(year, target_type, target_sub_type, decade) %>%
  filter(target_type != "Unknown")

#table
dftargetstop <- dftargets %>%
  group_by(decade, target_type) %>%
  summarise(nr_of_attacks = n()) %>%
  top_n(n=5, wt=nr_of_attacks) %>%
  arrange(decade, desc(nr_of_attacks))
```

```
## `summarise()` has grouped output by 'decade'. You can override using the
## `.groups` argument.
```

```r
#visual
ggplot(data=dftargetstop, aes(x=decade, y=nr_of_attacks, col=target_type, group= target_type)) +
  geom_line(size=1.5, alpha=0.5)+
  labs(title='Terrorism targets over time')
```



#RESEARCH QUESTION 1 How does the number of fatalities (nkill) vary by weapon type and target type over different years?

Method: Two-way ANOVA with bootstrap resampling.

```r
head(df)
```

```
## # A tibble: 6 x 26
##    year month   day country           region   city  latitude longitude summary
##   <dbl> <dbl> <dbl> <chr>             <chr>    <chr>    <dbl>     <dbl> <chr>
## 1  1970     7     2 Dominican Republic Central~ Sant~     18.5     -70.0 No sum~
## 2  1970     1     1 Mexico            North A~ Mexi~     19.4     -99.1 No sum~
## 3  1970     1     1 Philippines       Southea~ Unkn~     15.5     121.  No sum~
## 4  1970     1     1 Greece            Western~ Athe~     38.0      23.8 No sum~
## 5  1970     1     1 Japan             East As~ Fuko~     33.6     130.  No sum~
## 6  1970     1     1 United States     North A~ Cairo     37.0     -89.2 1/1/19~
## # i 17 more variables: multiple_attack <dbl>, attacktype <chr>,
## #   target_type <chr>, target_sub_type <chr>, group_name <chr>,
## #   weapon_type <chr>, nkill <dbl>, nwound <dbl>, nkillter <dbl>, decade <fct>,
## #   coords_missing <lgl>, nkill_capped <dbl>, nwound_capped <dbl>,
## #   nkillter_capped <dbl>, date <date>, severity_index <dbl>,
```

```
## #   severity_category <chr>
```

```
colnames(df)
```

```
##  [1] "year"              "month"             "day"
##  [4] "country"           "region"            "city"
##  [7] "latitude"          "longitude"         "summary"
## [10] "multiple_attack"   "attacktype"        "target_type"
## [13] "target_sub_type"   "group_name"        "weapon_type"
## [16] "nkill"             "nwound"            "nkillter"
## [19] "decade"            "coords_missing"    "nkill_capped"
## [22] "nwound_capped"     "nkillter_capped"   "date"
## [25] "severity_index"    "severity_category"
```

checking the data types of each column

```
str(df)
```

```
## tibble [181,691 x 26] (S3: tbl_df/tbl/data.frame)
##  $ year             : num [1:181691] 1970 1970 1970 1970 1970 1970 1970 1970 1970 1970 ...
##  $ month            : num [1:181691] 7 1 1 1 1 1 1 1 1 1 ...
##  $ day              : num [1:181691] 2 1 1 1 1 1 2 2 2 3 ...
##  $ country          : chr [1:181691] "Dominican Republic" "Mexico" "Philippines" "Greece" ...
##  $ region           : chr [1:181691] "Central America & Caribbean" "North America" "Southeast Asia" "
##  $ city             : chr [1:181691] "Santo Domingo" "Mexico city" "Unknown" "Athens" ...
##  $ latitude         : num [1:181691] 18.5 19.4 15.5 38 33.6 ...
##  $ longitude        : num [1:181691] -70 -99.1 120.6 23.8 130.4 ...
##  $ summary          : chr [1:181691] "No summary available" "No summary available" "No summary availa
##  $ multiple_attack  : num [1:181691] 0 0 0 0 0 0 0 0 0 0 ...
##  $ attacktype       : chr [1:181691] "Assassination" "Hostage Taking (Kidnapping)" "Assassination" "
##  $ target_type      : chr [1:181691] "Private Citizens & Property" "Government (Diplomatic)" "Journal
##  $ target_sub_type  : chr [1:181691] "Named Civilian" "Diplomatic Personnel (outside of embassy, cons
##  $ group_name       : chr [1:181691] "MANO-D" "23rd of September Communist League" "Unknown" "Unknow
##  $ weapon_type      : chr [1:181691] "Unknown" "Unknown" "Unknown" "Explosives" ...
##  $ nkill            : num [1:181691] 1 0 1 0 0 0 0 0 0 0 ...
##  $ nwound           : num [1:181691] 0 0 0 0 0 0 0 0 0 0 ...
##  $ nkillter         : num [1:181691] 0 0 0 0 0 0 0 0 0 0 ...
##  $ decade           : Factor w/ 5 levels "70s","80s","90s",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ coords_missing   : logi [1:181691] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ nkill_capped     : num [1:181691] 1 0 1 0 0 0 0 0 0 0 ...
##  $ nwound_capped    : num [1:181691] 0 0 0 0 0 0 0 0 0 0 ...
##  $ nkillter_capped  : num [1:181691] 0 0 0 0 0 0 0 0 0 0 ...
##  $ date             : Date[1:181691], format: "1970-07-02" "1970-01-01" ...
##  $ severity_index   : num [1:181691] 1 0 1 0 0 0 0 0 0 0 ...
##  $ severity_category: chr [1:181691] "Low" "No casualties" "Low" "No casualties" ...
```

We assess the dataset for checking zeros in nkill, total non-missing nkill values, and percentage of zeros in the dataset. We also create a subset pos_val of the main cleaned dataset df, that includes all the positive values and no zeroes. We are removing the zero values so that it does not lead to skewed results or inflated Type 1 errors.

```r
# Check zeros in nkill
sum(df$nkill == 0, na.rm = TRUE)
```

```
## [1] 98462
```

```r
# Total non-missing nkill values
sum(!is.na(df$nkill))
```

```
## [1] 181691
```

```r
# Percentage of zeros
sum(df$nkill == 0, na.rm = TRUE) / sum(!is.na(df$nkill)) * 100
```

```
## [1] 54.19201
```

```r
library(tidyverse)
library(caret)      # for cross-validation
library(boot)       # for bootstrap
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
## The following object is masked from 'package:car':
##
##     logit
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(glmnet)
```

```
## Loaded glmnet 4.1-8
```

```r
df$nkill[is.na(df$nkill)] <- 0

sum(df$nkill == 0)
```

```
## [1] 98462
```

```r
pos_val <- df %>% filter(!is.na(nkill) & nkill > 0)

# Ensure pos_val is numeric
pos_val$pos_val <- as.numeric(pos_val$nkill)
```

There are a lot of rare values present in the weapon_type column which in turn cause the standard error to increase. Here we collapse the rare categories to reduce the error.

```r
table(pos_val$weapon_type)
```

```
##
##                                                                  Biological
##                                                                           6
##                                                                    Chemical
##                                                                          63
##                                                                  Explosives
##                                                                       32868
##                                                                Fake Weapons
##                                                                           1
##                                                                   Firearms
##                                                                       40461
##                                                                 Incendiary
##                                                                         722
##                                                                      Melee
##                                                                       2191
##                                                                      Other
##                                                                         38
##                                                               Radiological
##                                                                           1
##                                                          Sabotage Equipment
##                                                                           9
##                                                                    Unknown
##                                                                       6797
## Vehicle (not to include vehicle-borne explosives, i.e., car or truck bombs)
##                                                                         72
```

```r
# Collapse rare weapon types
weapon_counts <- table(pos_val$weapon_type)
rare_weapons <- names(weapon_counts[weapon_counts < 500])

pos_val$weapon_type_collapsed <- as.character(pos_val$weapon_type)
pos_val$weapon_type_collapsed[pos_val$weapon_type_collapsed %in% rare_weapons] <- "Other"
pos_val$weapon_type_collapsed <- as.factor(pos_val$weapon_type_collapsed)

# Check new levels
table(pos_val$weapon_type_collapsed)
```

```
##
## Explosives   Firearms Incendiary      Melee      Other    Unknown
##      32868      40461        722       2191        190       6797
```

This code performs a two-way ANOVA with bootstrapping to assess how fatalities (pos_val) vary by weapon type, target type, and year. It first converts relevant columns to factors, then defines a function that runs

ANOVA on bootstrap samples and extracts the F-statistics. The boot() function is used to repeat this process 500 times, providing estimates of variability and stability for each effect. Finally, a standard ANOVA is run for comparison, allowing interpretation of both point estimates and their reliability.

```
# Load libraries
library(boot)
library(dplyr)

set.seed(123)

# Convert weapon_type and target_type to factors
pos_val$weapon_type <- as.factor(pos_val$weapon_type)
pos_val$target_type <- as.factor(pos_val$target_type)

# Define the resampling function for bootstrapping
boot_anova <- function(data, indices) {
  boot_data <- data[indices, ]

  # Perform Two-way ANOVA: pos_val ~ weapon_type * target_type + year
  anova_model_boot <- aov(pos_val ~ weapon_type_collapsed * target_type + factor(year), data = boot_data

  # Extract the F-statistic and return it as a numeric vector
  f_stat <- summary(anova_model_boot)[[1]]$`F value`

  # Return only the F-statistic
  return(f_stat)
}

# Perform bootstrapping
results <- boot(data = pos_val, statistic = boot_anova, R = 100)

# results
print(results)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = pos_val, statistic = boot_anova, R = 100)
##
##
## Bootstrap Statistics :
##      original    bias    std. error
## t1* 68.069227 13.295808   42.598149
## t2* 30.906684  1.756934    5.565110
## t3* 12.136068  1.676477    2.255909
## t4*  7.029822  2.803163    4.302777
## WARNING: All values of t5* are NA
```

```
# Alternatively, perform a standard ANOVA without resampling for comparison
anova_model <- aov(pos_val ~ weapon_type * target_type + factor(year), data = pos_val)
```

```r
# Summary of the standard ANOVA
summary(anova_model)
```

```
##                              Df   Sum Sq Mean Sq F value Pr(>F)
## weapon_type                  11   149869   13624   54.19 <2e-16 ***
## target_type                  21   165389    7876   31.32 <2e-16 ***
## factor(year)                 46   143180    3113   12.38 <2e-16 ***
## weapon_type:target_type     110   459173    4174   16.60 <2e-16 ***
## Residuals                83040 20879306     251
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F-statistics show how strongly each factor affects the outcome (pos_val, i.e., fatalities). Higher F-values = stronger evidence that a factor matters.

In this case:

F = 54.19 for weapon_type → very strong effect

F = 31.32 for target_type → strong effect

F = 12.38 for year → moderate but significant effect

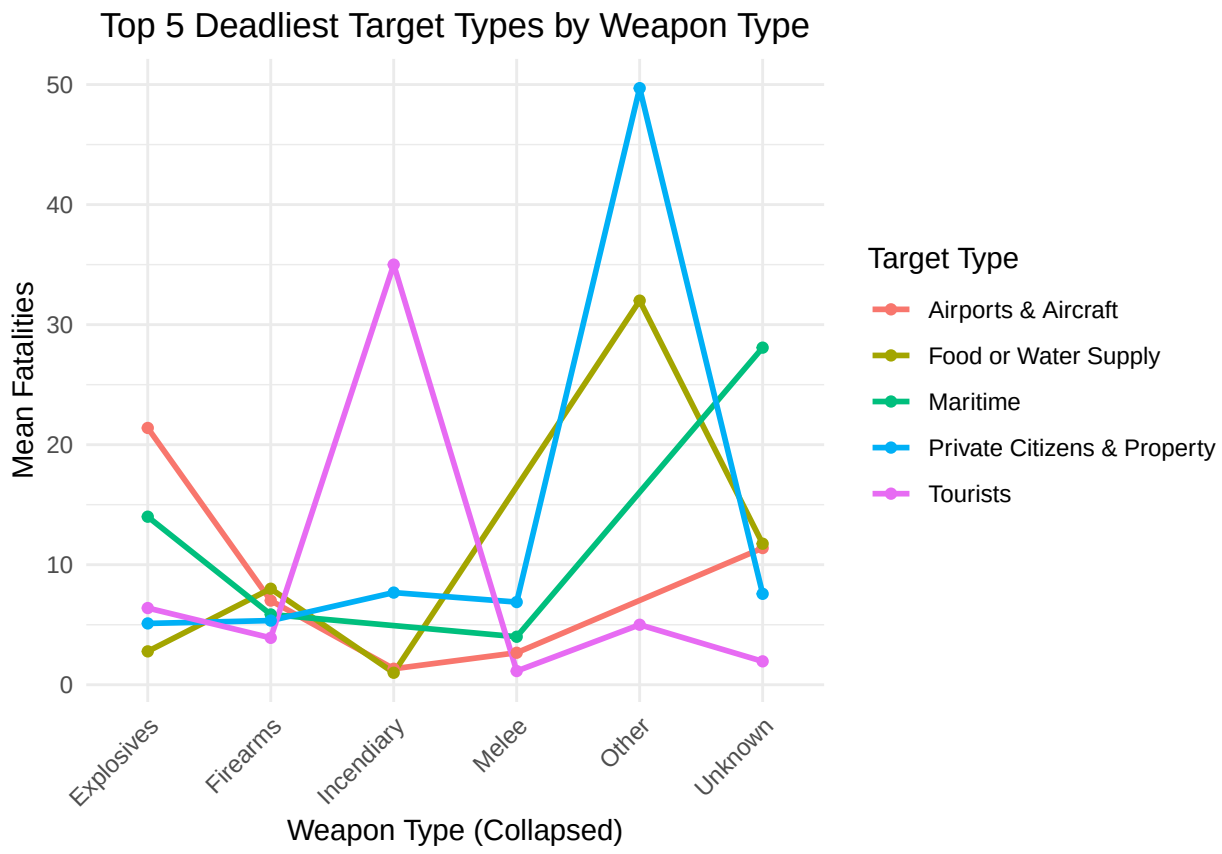
F = 16.60 for weapon_type × target_type interaction → important combined effect

The number of people killed in terrorist attacks is strongly influenced by the type of weapon used, the type of target, and the year. Importantly, some weapon-target combinations are particularly deadly, highlighting the importance of considering interactions. These findings are statistically robust, as confirmed by bootstrapping.

This interaction plot shows how average fatalities vary across combinations of weapon types and target types. Sharp peaks indicate especially deadly combinations, such as attacks on Tourists and Private Citizens & Property using Other weapons. Overall, while most combinations result in moderate fatalities, a few stand out with significantly higher impact.

```r
library(dplyr)
library(ggplot2)

plot_data <- pos_val %>%
  group_by(weapon_type_collapsed, target_type) %>%
  summarise(mean_fatalities = mean(pos_val), .groups = "drop")

# Get top 5 target types by average fatalities
top_targets <- plot_data %>%
  group_by(target_type) %>%
  summarise(avg_fatalities = mean(mean_fatalities, na.rm = TRUE)) %>%
  arrange(desc(avg_fatalities)) %>%
  slice_head(n = 5)

# Filter plot_data for only those target types
plot_top5 <- plot_data %>%
  filter(target_type %in% top_targets$target_type)

# Plot only the top 5 target types
ggplot(plot_top5, aes(x = weapon_type_collapsed, y = mean_fatalities, color = target_type, group = targ
  geom_line(size = 1) +
  geom_point() +
```

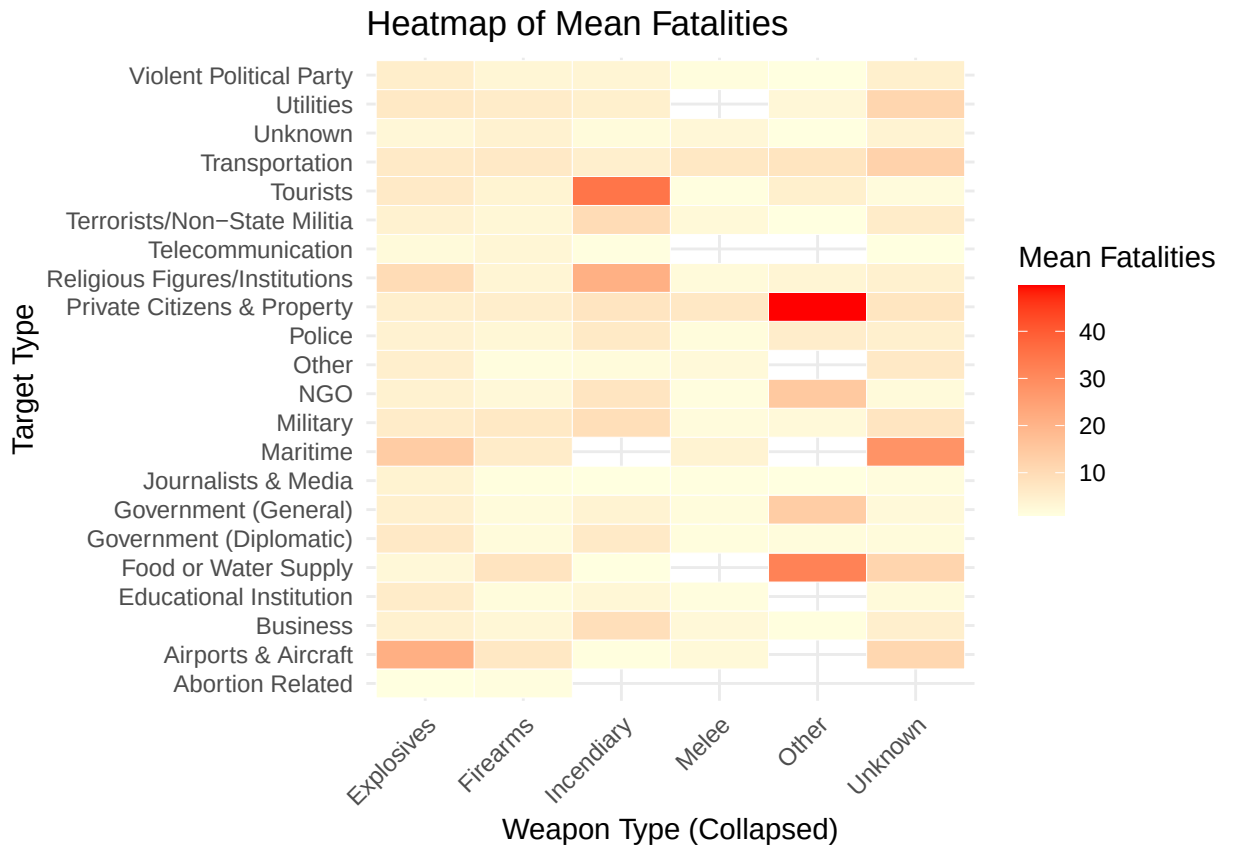
```
  theme_minimal() +
  scale_x_discrete(expand = expansion(mult = c(0.1, 0.1))) +
  labs(title = "Top 5 Deadliest Target Types by Weapon Type",
       x = "Weapon Type (Collapsed)",
       y = "Mean Fatalities",
       color = "Target Type") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))
```




```
heat_data <- pos_val %>%
  group_by(weapon_type_collapsed, target_type) %>%
  summarise(mean_fatalities = mean(pos_val))
```

```
## `summarise()` has grouped output by 'weapon_type_collapsed'. You can override
## using the `.groups` argument.
```

```
ggplot(heat_data, aes(x = weapon_type_collapsed, y = target_type, fill = mean_fatalities)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "lightyellow", high = "red") +
  labs(title = "Heatmap of Mean Fatalities",
       x = "Weapon Type (Collapsed)",
       y = "Target Type",
       fill = "Mean Fatalities") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



This heatmap shows the average number of fatalities across different combinations of weapon types and target types. Darker red cells indicate combinations that resulted in higher mean fatalities per incident, such as attacks on Private Citizens & Property with Other weapons. Most combinations result in lower fatalities (lighter shades), highlighting that only specific pairings lead to especially deadly outcomes.

Research Question 2:

Is there a significant difference in the number of casualties caused by different terrorist groups across regions over time? Method: Two-way ANOVA (Analysis of multi-factor experiments), Random Forest, Logistic Regression

This code builds classification models to predict whether a terrorist attack is part of a coordinated (multiple) attack using features like attack type, target type, region, group name, and casualty counts. It uses both logistic regression and random forest models without applying hyperparameter tuning or SMOTE. The data is preprocessed, split into training and testing sets, and evaluated using confusion matrices, ROC curves, and AUC scores. Cross-validation is performed to assess the generalizability of each model. The results help identify whether specific attack patterns or group characteristics are associated with coordinated attacks.

```
# Load packages
library(tidyverse)
library(caret)
library(randomForest)
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ROCR)

# 1. Prepare the dataset --------------------------------------------

# Select relevant columns
df_model <- df %>%
  dplyr::select(multiple_attack, attacktype, target_type, group_name, region, nkill, nwound)

# Convert categorical predictors to factors
df_model <- df_model %>%
  mutate(across(c(attacktype, target_type, group_name, region), as.factor),
         multiple_attack = factor(multiple_attack, levels = c(0, 1), labels = c("No", "Yes")))  # relab

# simplify group_name by keeping only top frequent groups
top_groups <- names(sort(table(df_model$group_name), decreasing = TRUE)[1:20])
df_model$group_name <- ifelse(df_model$group_name %in% top_groups, df_model$group_name, "Other")
df_model$group_name <- as.factor(df_model$group_name)

# 2. Train/Test Split -----------------------------------------------

set.seed(123)
trainIndex <- createDataPartition(df_model$multiple_attack, p = 0.8, list = FALSE)
train <- df_model[trainIndex, ]
test <- df_model[-trainIndex, ]
```

```r
# 3. Logistic Regression ------------------------------------------

log_model <- glm(multiple_attack ~ ., data = train, family = "binomial")
summary(log_model)
```

```
##
## Call:
## glm(formula = multiple_attack ~ ., family = "binomial", data = train)
##
## Coefficients:
##                                                Estimate Std. Error z value
## (Intercept)                                  -0.9792613  0.3036699  -3.225
## attacktypeAssassination                      -1.1682893  0.0528128 -22.121
## attacktypeBombing/Explosion                   0.5877995  0.0229328  25.631
## attacktypeFacility/Infrastructure Attack      1.0375138  0.0351078  29.552
## attacktypeHijacking                          -0.0560058  0.1750169  -0.320
## attacktypeHostage Taking (Barricade Incident) -0.0055168  0.1213374  -0.045
## attacktypeHostage Taking (Kidnapping)        -0.3791426  0.0438375  -8.649
## attacktypeUnarmed Assault                     0.7597301  0.0968161   7.847
## attacktypeUnknown                             0.3783101  0.0410902   9.207
## target_typeAirports & Aircraft               -0.8514155  0.2347801  -3.626
## target_typeBusiness                           0.3203257  0.1929534   1.660
## target_typeEducational Institution            0.3452791  0.1979770   1.744
## target_typeFood or Water Supply               0.1874465  0.2600715   0.721
## target_typeGovernment (Diplomatic)           -0.5709446  0.2069934  -2.758
## target_typeGovernment (General)               0.1855199  0.1940012   0.956
## target_typeJournalists & Media               -0.4019794  0.2127962  -1.889
## target_typeMaritime                          -0.3298408  0.2878258  -1.146
## target_typeMilitary                          -0.1704300  0.1941781  -0.878
## target_typeNGO                               -0.8633726  0.2651921  -3.256
## target_typeOther                              1.2040385  0.2891331   4.164
## target_typePolice                             0.1609464  0.1941122   0.829
## target_typePrivate Citizens & Property        0.3826284  0.1932728   1.980
## target_typeReligious Figures/Institutions     0.3509588  0.1977800   1.774
## target_typeTelecommunication                  0.9285659  0.2086178   4.451
## target_typeTerrorists/Non-State Militia      -0.2819554  0.2077765  -1.357
## target_typeTourists                          -0.0226358  0.2634711  -0.086
## target_typeTransportation                     0.2689123  0.1965057   1.368
## target_typeUnknown                            0.1613890  0.1980840   0.815
## target_typeUtilities                          1.5003505  0.1951919   7.687
## target_typeViolent Political Party            0.2663641  0.2141276   1.244
## group_name1393                               -0.8139352  0.1158283  -7.027
## group_name1483                               -1.7363254  0.1169314 -14.849
## group_name1534                               -0.4961226  0.0898165  -5.524
## group_name165                                -1.4189939  0.1386088 -10.237
## group_name177                                -1.4543724  0.1039247 -13.994
## group_name1799                               -1.7230609  0.1110336 -15.518
## group_name1865                               -2.4163647  0.1535990 -15.732
## group_name1960                               -1.6485857  0.1208069 -13.646
## group_name2200                               -1.4943197  0.1197299 -12.481
## group_name2266                               -2.2537318  0.1129791 -19.948
## group_name2389                               -3.9407262  0.3029526 -13.008
## group_name2714                               -1.7161224  0.1134859 -15.122
```

```
## group_name2946                                  -1.1638918  0.0982231 -11.849
## group_name3138                                  -1.0260415  0.0915653 -11.206
## group_name3164                                  -1.0417140  0.1179149  -8.834
## group_name3408                                  -2.2113270  0.0823846 -26.841
## group_name618                                   -1.8601650  0.1227063 -15.159
## group_name705                                   -0.2751909  0.0994552  -2.767
## group_name890                                   -1.6078022  0.1152751 -13.948
## group_nameOther                                 -1.2030057  0.0815373 -14.754
## regionCentral America & Caribbean               -0.8457727  0.2287240  -3.698
## regionCentral Asia                               0.0253303  0.2892489   0.088
## regionEast Asia                                  0.2541011  0.2473758   1.027
## regionEastern Europe                             0.2940935  0.2245988   1.309
## regionMiddle East & North Africa                 0.2098679  0.2201064   0.953
## regionNorth America                             -0.1419972  0.2265125  -0.627
## regionSouth America                              0.0784721  0.2219261   0.354
## regionSouth Asia                                 0.0485481  0.2203364   0.220
## regionSoutheast Asia                             0.6948135  0.2213628   3.139
## regionSub-Saharan Africa                         0.4044536  0.2214038   1.827
## regionWestern Europe                            -0.0278885  0.2213313  -0.126
## nkill                                           -0.0066295  0.0012085  -5.486
## nwound                                           0.0013959  0.0002581   5.409
##                                                Pr(>|z|)
## (Intercept)                                    0.001261 **
## attacktypeAssassination                         < 2e-16 ***
## attacktypeBombing/Explosion                     < 2e-16 ***
## attacktypeFacility/Infrastructure Attack        < 2e-16 ***
## attacktypeHijacking                            0.748967
## attacktypeHostage Taking (Barricade Incident)  0.963735
## attacktypeHostage Taking (Kidnapping)           < 2e-16 ***
## attacktypeUnarmed Assault                      4.26e-15 ***
## attacktypeUnknown                               < 2e-16 ***
## target_typeAirports & Aircraft                 0.000287 ***
## target_typeBusiness                            0.096891 .
## target_typeEducational Institution             0.081153 .
## target_typeFood or Water Supply                0.471063
## target_typeGovernment (Diplomatic)             0.005811 **
## target_typeGovernment (General)                0.338930
## target_typeJournalists & Media                 0.058887 .
## target_typeMaritime                            0.251806
## target_typeMilitary                            0.380107
## target_typeNGO                                 0.001131 **
## target_typeOther                               3.12e-05 ***
## target_typePolice                              0.407025
## target_typePrivate Citizens & Property         0.047734 *
## target_typeReligious Figures/Institutions      0.075982 .
## target_typeTelecommunication                   8.55e-06 ***
## target_typeTerrorists/Non-State Militia        0.174777
## target_typeTourists                            0.931535
## target_typeTransportation                      0.171165
## target_typeUnknown                             0.415215
## target_typeUtilities                           1.51e-14 ***
## target_typeViolent Political Party             0.213518
## group_name1393                                 2.11e-12 ***
## group_name1483                                  < 2e-16 ***
```

```
## group_name1534                                3.32e-08 ***
## group_name165                                  < 2e-16 ***
## group_name177                                  < 2e-16 ***
## group_name1799                                 < 2e-16 ***
## group_name1865                                 < 2e-16 ***
## group_name1960                                 < 2e-16 ***
## group_name2200                                 < 2e-16 ***
## group_name2266                                 < 2e-16 ***
## group_name2389                                 < 2e-16 ***
## group_name2714                                 < 2e-16 ***
## group_name2946                                 < 2e-16 ***
## group_name3138                                 < 2e-16 ***
## group_name3164                                 < 2e-16 ***
## group_name3408                                 < 2e-16 ***
## group_name618                                  < 2e-16 ***
## group_name705                                 0.005658 **
## group_name890                                  < 2e-16 ***
## group_nameOther                                < 2e-16 ***
## regionCentral America & Caribbean             0.000217 ***
## regionCentral Asia                            0.930216
## regionEast Asia                               0.304333
## regionEastern Europe                          0.190393
## regionMiddle East & North Africa              0.340345
## regionNorth America                           0.530735
## regionSouth America                           0.723642
## regionSouth Asia                              0.825609
## regionSoutheast Asia                          0.001696 **
## regionSub-Saharan Africa                      0.067735 .
## regionWestern Europe                          0.899729
## nkill                                         4.11e-08 ***
## nwound                                        6.35e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 116545  on 145353  degrees of freedom
## Residual deviance: 103528  on 145291  degrees of freedom
## AIC: 103654
##
## Number of Fisher Scoring iterations: 6
```

```r
# Predict on test set
log_probs <- predict(log_model, newdata = test, type = "response")
log_preds <- ifelse(log_probs > 0.5, "Yes", "No")  # match factor levels
log_preds <- factor(log_preds, levels = levels(test$multiple_attack))  # ensure matching levels

# Confusion Matrix
confusionMatrix(log_preds, test$multiple_attack)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
```

```
##        No  31094  4647
##        Yes   237   359
##
##               Accuracy : 0.8656
##                 95% CI : (0.862, 0.8691)
##    No Information Rate : 0.8622
##    P-Value [Acc > NIR] : 0.03188
##
##                  Kappa : 0.1018
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.99244
##             Specificity : 0.07171
##          Pos Pred Value : 0.86998
##          Neg Pred Value : 0.60235
##              Prevalence : 0.86223
##          Detection Rate : 0.85571
##    Detection Prevalence : 0.98360
##       Balanced Accuracy : 0.53207
##
##        'Positive' Class : No
##
```

```
# ROC Curve
roc_log <- roc(response = test$multiple_attack, predictor = log_probs, levels = c("No", "Yes"))
```

```
## Setting direction: controls < cases
```

```
plot(roc_log, col = "blue", main = "ROC Curve - Logistic Regression")
auc(roc_log)
```

```
## Area under the curve: 0.7256
```

```
# 4. Random Forest ----------------------------------------------------

set.seed(123)
rf_model <- randomForest(multiple_attack ~ ., data = train, ntree = 100, importance = TRUE)
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = multiple_attack ~ ., data = train, ntree = 100,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 12.66%
## Confusion matrix:
##         No  Yes class.error
## No  124129 1199 0.009566896
## Yes  17202 2824 0.858983322
```

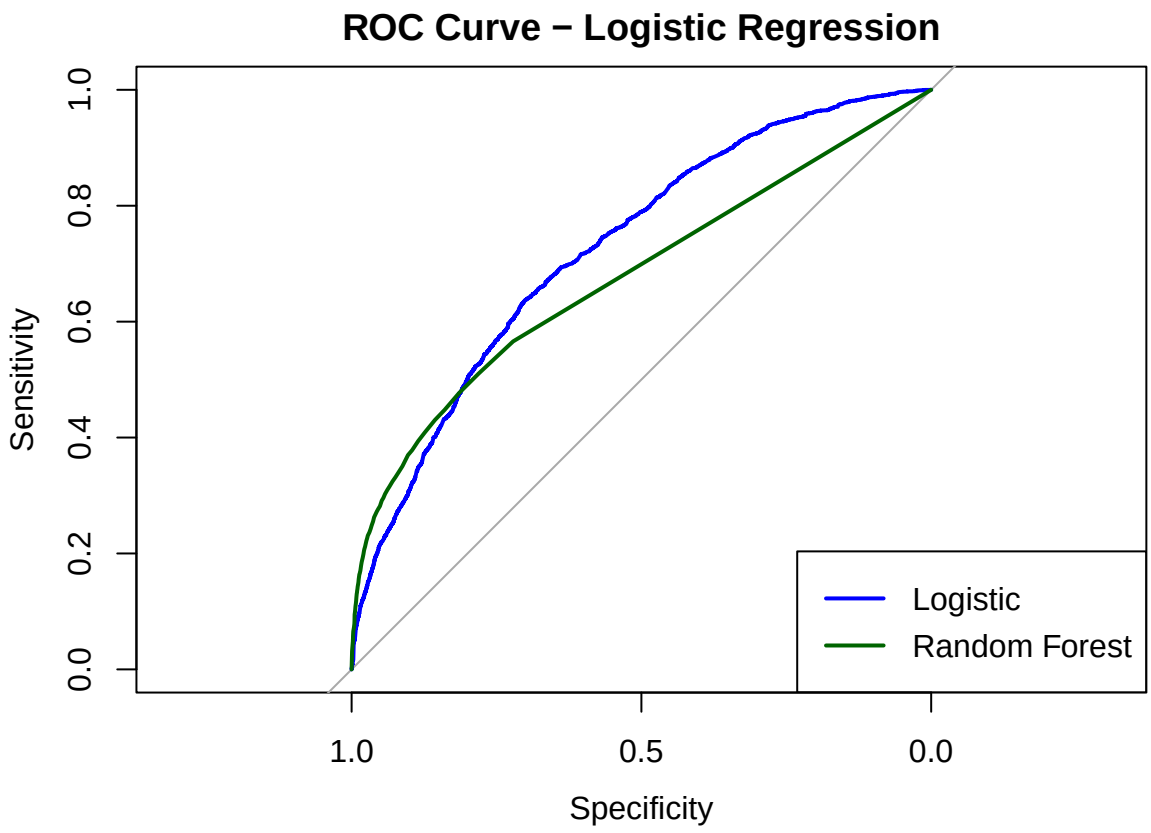
```r
# Predict and evaluate
rf_preds <- predict(rf_model, newdata = test)
confusionMatrix(rf_preds, test$multiple_attack)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  31028  4325
##        Yes   303   681
##
##                Accuracy : 0.8726
##                  95% CI : (0.8692, 0.876)
##     No Information Rate : 0.8622
##     P-Value [Acc > NIR] : 3.183e-09
##
##                   Kappa : 0.1908
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9903
##             Specificity : 0.1360
##          Pos Pred Value : 0.8777
##          Neg Pred Value : 0.6921
##              Prevalence : 0.8622
##          Detection Rate : 0.8539
##    Detection Prevalence : 0.9729
##       Balanced Accuracy : 0.5632
##
##        'Positive' Class : No
##
```

```r
# ROC for RF
rf_probs <- predict(rf_model, newdata = test, type = "prob")[, "Yes"]
roc_rf <- roc(response = test$multiple_attack, predictor = rf_probs, levels = c("No", "Yes"))
```

```
## Setting direction: controls < cases
```

```r
plot(roc_rf, col = "darkgreen", add = TRUE)
legend("bottomright", legend = c("Logistic", "Random Forest"), col = c("blue", "darkgreen"), lwd = 2)
```




```
# AUC
auc(roc_rf)
```

```
## Area under the curve: 0.6764
```

```
# 5. Cross-validation (Logistic) -----------------------------------

set.seed(123)
cv_ctrl <- trainControl(method = "cv", number = 5, classProbs = TRUE, summaryFunction = twoClassSummary
log_cv <- train(multiple_attack ~ ., data = train, method = "glm", family = "binomial", trControl = cv_
log_cv
```

```
## Generalized Linear Model
##
## 145354 samples
##      6 predictor
##      2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 116283, 116283, 116283, 116284, 116283
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7317483  0.9930263  0.07330462
```

```r
# 6. Cross-validation (Random Forest - Using ranger) ----------------

# cv control
cv_ctrl <- trainControl(
  method = "cv",
  number = 3,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = TRUE
)

# Train using ranger with fewer trees for speed
library(doParallel)
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```r
cl <- makePSOCKcluster(parallel::detectCores() - 1)
registerDoParallel(cl)

# run the ranger model
set.seed(123)
rf_cv <- train(
  multiple_attack ~ .,
  data = train,
  method = "ranger",
  trControl = cv_ctrl,
  metric = "ROC",
  num.trees = 100
)
```

```
## Growing trees.. Progress: 33%. Estimated remaining time: 1 minute, 2 seconds.
## Growing trees.. Progress: 61%. Estimated remaining time: 40 seconds.
## Growing trees.. Progress: 89%. Estimated remaining time: 11 seconds.
```

```r
stopCluster(cl)
```

This section uses two-way ANOVA to examine how total casualties vary by terrorist group and region, including their interaction. A new casualties variable is created by combining killed and wounded counts. An interaction plot visualizes patterns, and Tukey's HSD tests identify significant group and region differences. This adds statistical evidence to support differences in attack severity across groups and locations.

```
rf_cv
```

```
## Random Forest
##
## 145354 samples
##      6 predictor
##      2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 96902, 96903, 96903
## Resampling results across tuning parameters:
##
##   mtry  splitrule   ROC        Sens       Spec
##    2    gini        0.7604532  1.0000000  0.0000000
##    2    extratrees  0.7436508  1.0000000  0.0000000
##   32    gini        0.8028868  0.9752489  0.2249575
##   32    extratrees  0.8010017  0.9827333  0.1780186
##   62    gini        0.7893181  0.9660730  0.2499751
##   62    extratrees  0.7888223  0.9646607  0.2503246
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 32, splitrule = gini
##  and min.node.size = 1.
```

```
# Create a new column for total casualties
df_model$casualties <- df_model$nkill + df_model$nwound

# Run two-way ANOVA
anova_result <- aov(casualties ~ group_name * region, data = df_model)
summary(anova_result)
```

```
##                       Df    Sum Sq Mean Sq F value   Pr(>F)
## group_name            20   1012007   50600   30.94  < 2e-16 ***
## region                11    623034   56639   34.63  < 2e-16 ***
## group_name:region     35    179132    5118    3.13 1.32e-09 ***
## Residuals         181624 297018960    1635
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
tukey_group <- TukeyHSD(anova_result, which = "group_name")
tukey_region <- TukeyHSD(anova_result, which = "region")
```

WITH HYPERPARAMETER AND TUNING

This code builds classification models to predict whether a terrorist attack is part of a coordinated (multiple) attack, using SMOTE to handle class imbalance and hyperparameter tuning to optimize model performance. First, categorical predictors are converted to numeric, and the data is split into training and test sets. SMOTE (Synthetic Minority Oversampling Technique) is applied using the themis package to balance the classes in the training data. A logistic regression model is trained on the balanced data, and its performance is evaluated using a confusion matrix and ROC curve on the original test set.

Next, a random forest model is trained on the SMOTE-balanced data with 3-fold cross-validation and hyperparameter tuning for the mtry value (number of variables tried at each split). The training is accelerated using parallel processing. The tuned random forest is then evaluated on the test set using accuracy metrics and AUC from the ROC curve. This approach improves model fairness and predictive power by addressing class imbalance and optimizing model settings. Overall, the code ensures a more robust and accurate prediction of multiple attacks using modern ML best practices.

```
# === Random Forest with SMOTE + Hyperparameter Tuning to Handle Imbalance ===

# Load libraries
library(tidyverse)
library(caret)
library(pROC)
library(themis)
```

```
## Loading required package: recipes
```

```
##
## Attaching package: 'recipes'
```

```
## The following object is masked from 'package:arules':
##
##     discretize
```

```
## The following object is masked from 'package:Matrix':
##
##     update
```

```
## The following object is masked from 'package:stringr':
##
##     fixed
```

```
## The following object is masked from 'package:stats':
##
##     step
```

```
library(recipes)
library(doParallel)

# Step 1: Prepare the dataset

# Convert categorical variables to numeric for SMOTE
df_model <- df_model %>%
  mutate(across(c(attacktype, target_type, group_name, region), ~ as.integer(as.factor(.))))

# Create training and test sets
set.seed(123)
trainIndex <- createDataPartition(df_model$multiple_attack, p = 0.8, list = FALSE)
train_set <- df_model[trainIndex, ]
test_set <- df_model[-trainIndex, ]

# SMOTE using themis + recipes
```

```r
rec <- recipe(multiple_attack ~ ., data = train_set) %>%
  step_smote(multiple_attack, over_ratio = 1.0)

rec_prep <- prep(rec, verbose = FALSE)
train_balanced <- juice(rec_prep)

# Check class balance before and after
cat("Original class distribution:\n")
```

```
## Original class distribution:
```

```r
print(table(train_set$multiple_attack))
```

```
##
##     No    Yes
## 125328  20026
```

```r
cat("Balanced class distribution after SMOTE:\n")
```

```
## Balanced class distribution after SMOTE:
```

```r
print(table(train_balanced$multiple_attack))
```

```
##
##     No    Yes
## 125328 125328
```

```r
# Logistic Regression on SMOTE data
log_model_smote <- glm(multiple_attack ~ ., data = train_balanced, family = "binomial")
summary(log_model_smote)
```

```
##
## Call:
## glm(formula = multiple_attack ~ ., family = "binomial", data = train_balanced)
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.6731084  0.0209417 -32.142  < 2e-16 ***
## attacktype   0.0725592  0.0021636  33.536  < 2e-16 ***
## target_type  0.0376003  0.0007755  48.484  < 2e-16 ***
## group_name  -0.0163161  0.0007576 -21.537  < 2e-16 ***
## region       0.0336284  0.0016548  20.322  < 2e-16 ***
## nkill       -0.0032941  0.0005053  -6.519 7.10e-11 ***
## nwound       0.0022247  0.0003238   6.871 6.39e-12 ***
## casualties          NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 347483  on 250655  degrees of freedom
## Residual deviance: 343208  on 250649  degrees of freedom
## AIC: 343222
##
## Number of Fisher Scoring iterations: 5
```

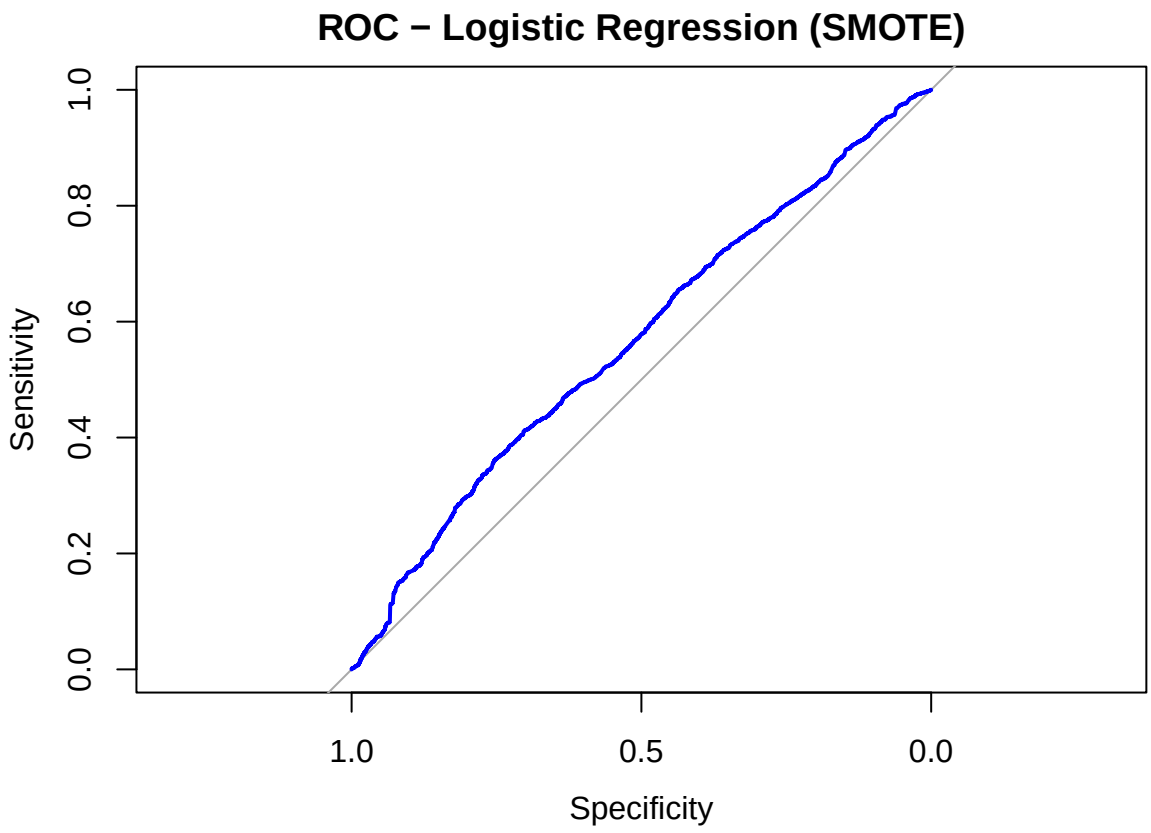
```r
# Predict and evaluate on test set
log_probs_smote <- predict(log_model_smote, newdata = test_set, type = "response")
log_preds_smote <- ifelse(log_probs_smote > 0.5, "Yes", "No")
log_preds_smote <- factor(log_preds_smote, levels = c("No", "Yes"))
confusionMatrix(log_preds_smote, factor(test_set$multiple_attack, levels = c("No", "Yes")))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  18281  2496
##        Yes 13050  2510
##
##                Accuracy : 0.5722
##                  95% CI : (0.5671, 0.5773)
##     No Information Rate : 0.8622
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.045
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5835
##             Specificity : 0.5014
##          Pos Pred Value : 0.8799
##          Neg Pred Value : 0.1613
##              Prevalence : 0.8622
##          Detection Rate : 0.5031
##    Detection Prevalence : 0.5718
##       Balanced Accuracy : 0.5424
##
##        'Positive' Class : No
##
```

```r
# ROC
roc_log_smote <- roc(response = factor(test_set$multiple_attack, levels = c("No", "Yes")),
                     predictor = log_probs_smote, levels = c("No", "Yes"))
```

```
## Setting direction: controls < cases
```

```r
plot(roc_log_smote, col = "blue", main = "ROC - Logistic Regression (SMOTE)")
```




```r
auc(roc_log_smote)
```

```
## Area under the curve: 0.5664
```

```r
# Tuned Random Forest on SMOTE data
# Enable parallel processing to speed up training
cl <- makePSOCKcluster(parallel::detectCores() - 1)
registerDoParallel(cl)

cv_ctrl <- trainControl(
  method = "cv",
  number = 3,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = TRUE
)

set.seed(123)
rf_model_smote <- train(
  multiple_attack ~ .,
  data = train_balanced,
  method = "rf",
  trControl = cv_ctrl,
  metric = "ROC",
  tuneGrid = expand.grid(mtry = c(2, 4, 6)),
```

```
  ntree = 100
)

stopCluster(cl)
print(rf_model_smote)
```

```
## Random Forest
##
## 250656 samples
##      7 predictor
##      2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 167104, 167104, 167104
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##   2     0.8536418  0.7322546  0.8551242
##   4     0.8627039  0.7901267  0.8490361
##   6     0.8524446  0.7911161  0.8485494
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

```
# Predict and evaluate
rf_preds_smote <- predict(rf_model_smote, newdata = test_set)
confusionMatrix(rf_preds_smote, factor(test_set$multiple_attack, levels = c("No", "Yes")))
```

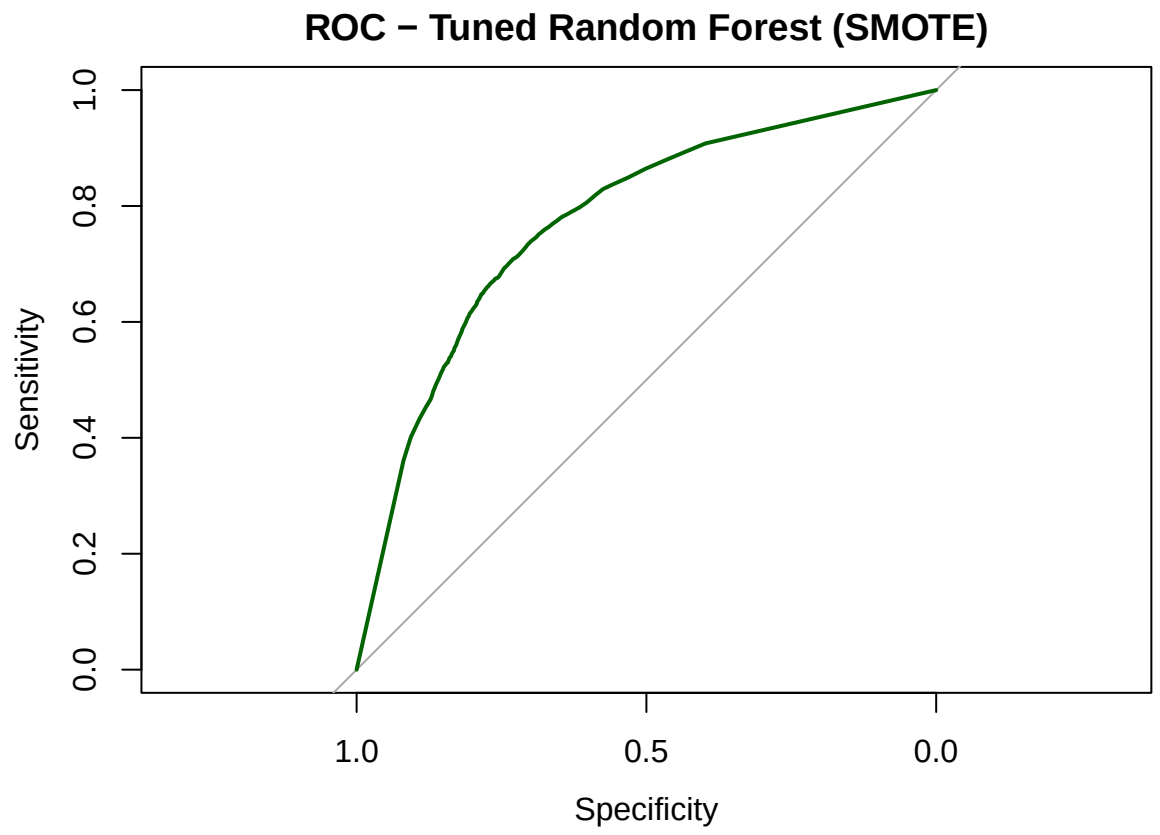
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  24861  1851
##        Yes  6470  3155
##
##                Accuracy : 0.771
##                  95% CI : (0.7666, 0.7753)
##     No Information Rate : 0.8622
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3054
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7935
##             Specificity : 0.6302
##          Pos Pred Value : 0.9307
##          Neg Pred Value : 0.3278
##              Prevalence : 0.8622
##          Detection Rate : 0.6842
##    Detection Prevalence : 0.7351
##       Balanced Accuracy : 0.7119
```

```
##
##        'Positive' Class : No
##
```

```r
# ROC
rf_probs_smote <- predict(rf_model_smote, newdata = test_set, type = "prob")[, "Yes"]
roc_rf_smote <- roc(response = factor(test_set$multiple_attack, levels = c("No", "Yes")),
                    predictor = rf_probs_smote, levels = c("No", "Yes"))
```

```
## Setting direction: controls < cases
```

```r
plot(roc_rf_smote, col = "darkgreen", main = "ROC - Tuned Random Forest (SMOTE)")
```




```r
auc(roc_rf_smote)
```

```
## Area under the curve: 0.7739
```

#RESEARCH QUESTION 3:

How Can We Forecast Future Trends in the Frequency and Severity of Global Terrorist Attacks?

Method: Time-series forecasting

The code performs data preprocessing and feature engineering on the terrorism dataset along with associated population data. It starts by loading the needed libraries and reading in the terrorism and population CSV files. Then, it filters and cleans the population data, and selects relevant columns from the terrorism dataset while renaming them for clarity. It handles missing values through imputation, fixes geographical coordinates by imputing medians at various geographic levels, and caps outliers for several key numeric variables. Finally, it creates new features such as a severity index and date fields, categorizes severity into bins, and prepares the cleaned dataset (df_analysis) for further analysis.

```
# Load required libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(readr)
library(ggmap)
```

```
## i Google's Terms of Service: <https://mapsplatform.google.com>
##   Stadia Maps' Terms of Service: <https://stadiamaps.com/terms-of-service/>
##   OpenStreetMap's Tile Usage Policy: <https://operations.osmfoundation.org/policies/tiles/>
## i Please cite ggmap if you use it! Use `citation("ggmap")` for details.
```

```
library(rworldmap)
```

```
## Loading required package: sp
## ### Welcome to rworldmap ###
## For a short introduction type :   vignette('rworldmap')
```

```
library(arules)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
##
## Attaching package: 'arules'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arulesViz)
library(ggpubr)
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:arules':
##
##     recode
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
##
## Attaching package: 'forecast'
##
## The following object is masked from 'package:ggpubr':
##
##     gghistogram
```

```r
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(ggfortify)
```

```
## Registered S3 methods overwritten by 'ggfortify':
##   method                 from
##   autoplot.Arima         forecast
##   autoplot.acf           forecast
##   autoplot.ar            forecast
##   autoplot.bats          forecast
##   autoplot.decomposed.ts forecast
##   autoplot.ets           forecast
##   autoplot.forecast      forecast
##   autoplot.stl           forecast
##   autoplot.ts            forecast
##   fitted.ar              forecast
##   fortify.ts             forecast
##   residuals.ar           forecast
```

```r
library(naniar)

# Load datasets
fulldf <- read_csv("D:\\Stony Brook\\Statistical Computing Project\\globalterrorismdb_0718dist.csv")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 181691 Columns: 135
## -- Column specification --------------------------------------------------------
```

```
## Delimiter: ","
## chr (55): approxdate, resolution, country_txt, region_txt, provstate, city, ...
## dbl (75): eventid, iyear, imonth, iday, extended, country, region, latitude,...
## lgl  (5): gsubname3, weaptype4, weaptype4_txt, weapsubtype4, weapsubtype4_txt
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
fullpop <- read_csv("D:\\Stony Brook\\Statistical Computing Project\\UNpopfile.csv")
```

```
## Rows: 371007 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Location, Variant
## dbl (7): LocID, VarID, Time, MidPeriod, PopMale, PopFemale, PopTotal
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Process population data
pop <- fullpop %>%
  select(-MidPeriod, -PopMale, -PopFemale, -VarID) %>%
  filter(Time > 1969 & Variant == 'Medium' & Time < 2017) %>%
  select(-Variant, -LocID)

futurepop <- fullpop %>%
  filter(Time > 2016 & Variant == 'Medium') %>%
  select(-Variant, -LocID, -MidPeriod, -PopMale, -PopFemale, -VarID)

# ------------------------------------------------------------------------------
# 1. SELECT AND RENAME COLUMNS
#    Including property damage and hostage columns for extended severity analysis
# ------------------------------------------------------------------------------
df <- fulldf %>%
  select(
    # Date/time info
    iyear, imonth, iday,
    # Geographic info
    country_txt, region_txt, city, latitude, longitude,
    # Incident descriptions
    summary, multiple,
    # Attack/target info
    attacktype1_txt, targtype1_txt, targsubtype1_txt, gname, weaptype1_txt,
    # Casualties
    nkill, nwound, nkillter,
    # Property damage columns
    property, propextent, propextent_txt, propvalue, propcomment,
    # Hostage/kidnapping columns
    ishostkid, nhostkid
  ) %>%
  rename(
    year = iyear,
    month = imonth,
```

```
    day = iday,
    country = country_txt,
    region = region_txt,
    multiple_attack = multiple,
    attacktype = attacktype1_txt,
    target_type = targtype1_txt,
    target_sub_type = targsubtype1_txt,
    group_name = gname,
    weapon_type = weaptype1_txt
  )

# ------------------------------------------------------------------------------
# 2. CREATE A DECADE VARIABLE
# ------------------------------------------------------------------------------
df <- df %>%
  mutate(
    decade = case_when(
      year < 1980 ~ "70s",
      year < 1990 ~ "80s",
      year < 2000 ~ "90s",
      year < 2010 ~ "2000s",
      TRUE        ~ "2010s"
    )
  )

df$decade <- factor(df$decade, levels = c("70s", "80s", "90s", "2000s", "2010s"))

# ------------------------------------------------------------------------------
# 3. CHECK FOR MISSING VALUES
# ------------------------------------------------------------------------------
missing_values <- colSums(is.na(df))
print(missing_values)
```

```
##            year           month             day         country          region
##               0               0               0               0               0
##            city        latitude       longitude         summary multiple_attack
##             434            4556            4557           66129               1
##      attacktype     target_type target_sub_type      group_name     weapon_type
##               0               0           10373               0               0
##           nkill          nwound        nkillter        property      propextent
##           10313           16311           66958               0          117626
##  propextent_txt       propvalue     propcomment       ishostkid        nhostkid
##          117626          142702          123732             178          168119
##          decade
##               0
```

```
# ------------------------------------------------------------------------------
# 4. HANDLE MISSING VALUES
#    Replace NA in categorical columns, median-impute numeric columns, etc.
# ------------------------------------------------------------------------------
df <- df %>%
  # Replace NA in categorical columns with "Unknown" or suitable placeholders
  mutate(
```

```
    group_name      = ifelse(is.na(group_name), "Unknown", group_name),
    city            = ifelse(is.na(city), "Unknown", city),
    target_sub_type = ifelse(is.na(target_sub_type), "Unknown", target_sub_type),
    summary         = ifelse(is.na(summary), "No summary available", summary)
  ) %>%
  # Replace NA in multiple_attack with 0
  mutate(multiple_attack = ifelse(is.na(multiple_attack), 0, multiple_attack)) %>%
  # Replace NA in numeric columns with medians (or zeros if more appropriate)
  mutate(
    nkill     = ifelse(is.na(nkill), median(nkill, na.rm = TRUE), nkill),
    nwound    = ifelse(is.na(nwound), median(nwound, na.rm = TRUE), nwound),
    nkillter  = ifelse(is.na(nkillter), median(nkillter, na.rm = TRUE), nkillter),
    # For property damage, set propvalue=0 if missing
    propvalue = ifelse(is.na(propvalue), 0, propvalue),
    # For property indicator (0 or 1), if missing, assume no property damage
    property  = ifelse(is.na(property), 0, property),
    # For hostage columns, set ishostkid=0 if missing, likewise for nhostkid
    ishostkid = ifelse(is.na(ishostkid), 0, ishostkid),
    nhostkid  = ifelse(is.na(nhostkid), 0, nhostkid)
  )

# ------------------------------------------------------------------------------
# 5. HANDLE MISSING COORDINATES (latitude, longitude)
# ------------------------------------------------------------------------------
df <- df %>%
  mutate(coords_missing = is.na(latitude) | is.na(longitude))

df <- df %>%
  group_by(country) %>%
  mutate(
    latitude  = ifelse(is.na(latitude), median(latitude, na.rm = TRUE), latitude),
    longitude = ifelse(is.na(longitude), median(longitude, na.rm = TRUE), longitude)
  ) %>%
  ungroup()

df <- df %>%
  group_by(region) %>%
  mutate(
    latitude  = ifelse(is.na(latitude), median(latitude, na.rm = TRUE), latitude),
    longitude = ifelse(is.na(longitude), median(longitude, na.rm = TRUE), longitude)
  ) %>%
  ungroup()

global_lat_median  <- median(df$latitude, na.rm = TRUE)
global_long_median <- median(df$longitude, na.rm = TRUE)

df <- df %>%
  mutate(
    latitude  = ifelse(is.na(latitude), global_lat_median, latitude),
    longitude = ifelse(is.na(longitude), global_long_median, longitude)
  )

# ------------------------------------------------------------------------------
```

```r
# 6. HANDLE OUTLIERS VIA CAPPING
# ------------------------------------------------------------------------------
cap_outliers <- function(x, lower_quantile = 0.05, upper_quantile = 0.95) {
  qnt <- quantile(x, probs = c(lower_quantile, upper_quantile), na.rm = TRUE)
  x[x < qnt[1]] <- qnt[1]
  x[x > qnt[2]] <- qnt[2]
  return(x)
}

df <- df %>%
  mutate(
    nkill_capped    = cap_outliers(nkill),
    nwound_capped   = cap_outliers(nwound),
    nkillter_capped = cap_outliers(nkillter),
    propvalue_capped = cap_outliers(propvalue)
  )

# ------------------------------------------------------------------------------
# 7. CREATE ADDITIONAL FEATURES (DATE, SEVERITY INDEX, ETC.)
# ------------------------------------------------------------------------------
df <- df %>%
  mutate(
    # Replace 0 or NA month/day with 1 for minimal date construction
    month = ifelse(month == 0 | is.na(month), 1, month),
    day   = ifelse(day   == 0 | is.na(day),   1, day),
    date  = as.Date(paste(year, month, day, sep = "-"), format = "%Y-%m-%d")
  ) %>%
  # Create severity metrics
  mutate(
    # Simple casualty-based severity
    severity_index = nkill + 0.5 * nwound,

    composite_severity = nkill + 0.5 * nwound + nhostkid + (propvalue / 1e6),

    # Categorize the composite severity into bins
    composite_severity_cat = case_when(
      composite_severity == 0            ~ "No impact",
      composite_severity <= 5            ~ "Low",
      composite_severity <= 20           ~ "Medium",
      composite_severity <= 50           ~ "High",
      TRUE                               ~ "Extreme"
    )
  )

# ------------------------------------------------------------------------------
# 8. PREP FINAL DATASET FOR ANALYSIS
# ------------------------------------------------------------------------------
df_analysis <- df

# Examine structure and missingness post-cleaning

missing_after <- colSums(is.na(df_analysis))
```

```r
# Number of rows retained
nrow(df_analysis)
```

```
## [1] 181691
```

This code performs a time series analysis on terrorist attack data using monthly counts. It begins by loading libraries and reading a pre-cleaned terrorism dataset. The data is aggregated to compute the number of attacks per month, then converted into a time series object. The time series is split into training (80%) and test (20%) sets for forecasting purposes. Finally, it visualizes the original series with the train-test boundary and applies a 3-month moving average to smooth trends for better interpretability.

```r
# ------------------------------------------------------------------------------
# 1. Load Required Libraries
# ------------------------------------------------------------------------------
library(tidyverse)
library(forecast)
library(lubridate)
library(ggfortify)

# ------------------------------------------------------------------------------
# 2. Load the Dataset
# ------------------------------------------------------------------------------
# Adjust the file path as needed.
fulldf <- read_csv("D:\\Stony Brook\\Statistical Computing Project\\globalterrorismdb_0718dist.csv")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## Rows: 181691 Columns: 135
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (55): approxdate, resolution, country_txt, region_txt, provstate, city, ...
## dbl (75): eventid, iyear, imonth, iday, extended, country, region, latitude,...
## lgl  (5): gsubname3, weaptype4, weaptype4_txt, weapsubtype4, weapsubtype4_txt
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

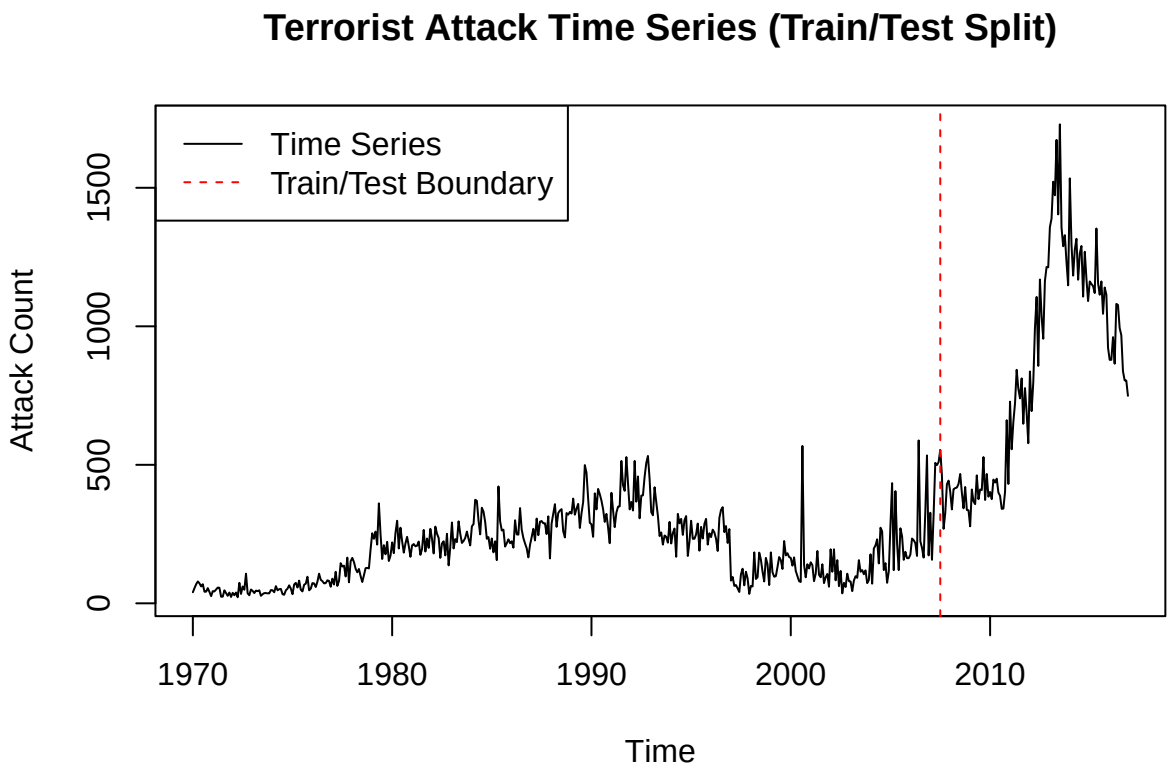
```r
# ------------------------------------------------------------------------------
# 3. Data Preprocessing
#    We assume that the dataset has been cleaned and processed into 'df_analysis'.
#    For our purposes, we only need the 'year' column to compute annual terrorist attack counts.
# ------------------------------------------------------------------------------
# Here we create an aggregated data frame counting the number of terrorist attack events per year.
df_attacks_per_month <- df_analysis %>%
  filter(!is.na(year)) %>%
  group_by(year, month) %>%
  summarise(
    terrorist_attacks_count = n()
  ) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

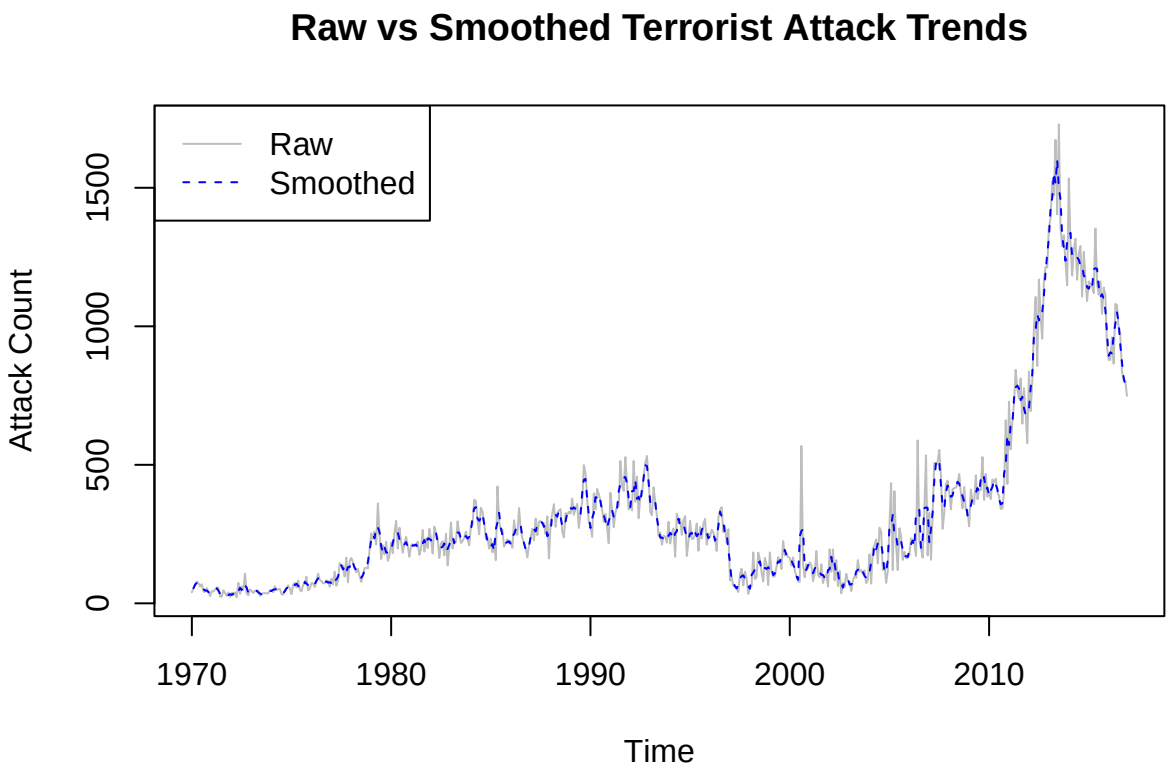
```
print(df_attacks_per_month)
```

```
## # A tibble: 564 x 3
##     year month terrorist_attacks_count
##    <dbl> <dbl>                   <int>
##  1  1970     1                      40
##  2  1970     2                      55
##  3  1970     3                      69
##  4  1970     4                      79
##  5  1970     5                      72
##  6  1970     6                      61
##  7  1970     7                      69
##  8  1970     8                      42
##  9  1970     9                      42
## 10  1970    10                      55
## # i 554 more rows
```

```
# ------------------------------------------------------------------------------
# 4. Create a Time Series Object for Terrorist Attacks (Monthly)
t_attack <- ts(df_attacks_per_month$terrorist_attacks_count,
               start = c(min(df_attacks_per_month$year), min(df_attacks_per_month$month)),
               frequency = 12)




# ------------------------------------------------------------------------------
# 5. Train-Test Split (80% Train, 20% Test)
# ------------------------------------------------------------------------------
n_total <- length(t_attack)
n_train <- floor(0.8 * n_total)
train_ts <- window(t_attack, end = time(t_attack)[n_train + 1])
test_ts  <- window(t_attack, start = time(t_attack)[n_train + 1])

# ------------------------------------------------------------------------------
# 6. Visualize the Raw Series and Train/Test Boundary
# ------------------------------------------------------------------------------
plot(t_attack, main = "Terrorist Attack Time Series (Train/Test Split)", ylab = "Attack Count")
abline(v = time(t_attack)[n_train], col = "red", lty = 2)
legend("topleft", legend = c("Time Series", "Train/Test Boundary"),
       col = c("black", "red"), lty = c(1,2))
```




```r
# ------------------------------------------------------------------------------
# 7. Smooth the Time Series Using a Moving Average
# ------------------------------------------------------------------------------
t_attack_trend <- stats::filter(t_attack, rep(1/3, 3), sides = 2)
ts.plot(t_attack, t_attack_trend, col = c("grey", "blue"), lty = 1:2,
        main = "Raw vs Smoothed Terrorist Attack Trends",
        ylab = "Attack Count")
legend("topleft", legend = c("Raw", "Smoothed"), col = c("grey", "blue"), lty = 1:2)
```



This code fits a Triple Exponential Smoothing (ETS) model to the training data to forecast terrorist attack counts. It generates predictions for both the test period and an additional 10 months into the future. Model performance is evaluated using Mean Absolute Percentage Error (MAPE) and autocorrelation of residuals (ACF1). Actual and predicted values are then organized into data frames and visualized using ggplot2, with color-coded lines distinguishing between train, test, test predictions, and future forecasts.

```r
# 8. Triple Exponential Smoothing (ETS) Model Fitting on Training Data
# ------------------------------------------------------------------------------

best_model <- ets(train_ts, model = "MMM")
summary(best_model)
```

```
## ETS(M,Md,M)
##
## Call:
## ets(y = train_ts, model = "MMM")
##
##   Smoothing parameters:
##     alpha = 0.1501
##     beta  = 0.0085
##     gamma = 1e-04
##     phi   = 0.9652
##
##   Initial states:
##     l = 63.8481
##     b = 0.9803
```

```
##     s = 0.8223 0.9079 1.0049 0.9972 1.1625 1.0153
##           1.0271 1.0865 0.9991 1.0622 0.9278 0.9872
##
##   sigma:  0.3676
##
##      AIC     AICc      BIC
## 6465.804 6467.383 6539.850
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE    ACF1
## Training set -0.162014 65.65959 44.34827 -9.857393 27.39066 0.6811854 0.18477
```

```
  # Forecast using the ETS model
  h <- length(test_ts) + 10  # forecast horizon: test period plus extra 10 time periods
  fcast <- forecast(best_model, h = h, level = c(80, 95))

  # ------------------------------------------------------------------------------
  # 10. Evaluate Model Accuracy on the Test Data
  # ------------------------------------------------------------------------------
  # Extract the predictions corresponding to the test period
  pred_test <- fcast$mean[1:length(test_ts)]
  model_accuracy <- accuracy(pred_test, test_ts)
  print(model_accuracy)
```

```
##                ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set 107.2987 324.2633 262.4695 -4.263991 34.61861 0.8600661  2.247084
```

```
  cat("\nETS Model Performance on Test Data:\n")
```

```
##
## ETS Model Performance on Test Data:
```

```
  cat(sprintf("\nMean Absolute Percentage Error (MAPE): %.2f%%", model_accuracy["Test set", "MAPE"]))
```

```
##
## Mean Absolute Percentage Error (MAPE): 34.62%
```

```
  cat(sprintf("\nAutocorrelation of Residuals (ACF1): %.4f\n", model_accuracy["Test set", "ACF1"]))
```

```
##
## Autocorrelation of Residuals (ACF1): 0.8601
```

```
  # ------------------------------------------------------------------------------
# 11. Combine Data for Plotting: Actual vs. Predicted Terrorist Attack Counts
# ------------------------------------------------------------------------------
# Prepare a data frame for the full actual series (train and test periods)
df_actual <- tibble(
  Year = as.numeric(time(t_attack)),
  Attack_Count = as.numeric(t_attack),
  Type = ifelse(as.numeric(time(t_attack)) <= max(as.numeric(time(train_ts))), "Train", "Test")
)
```

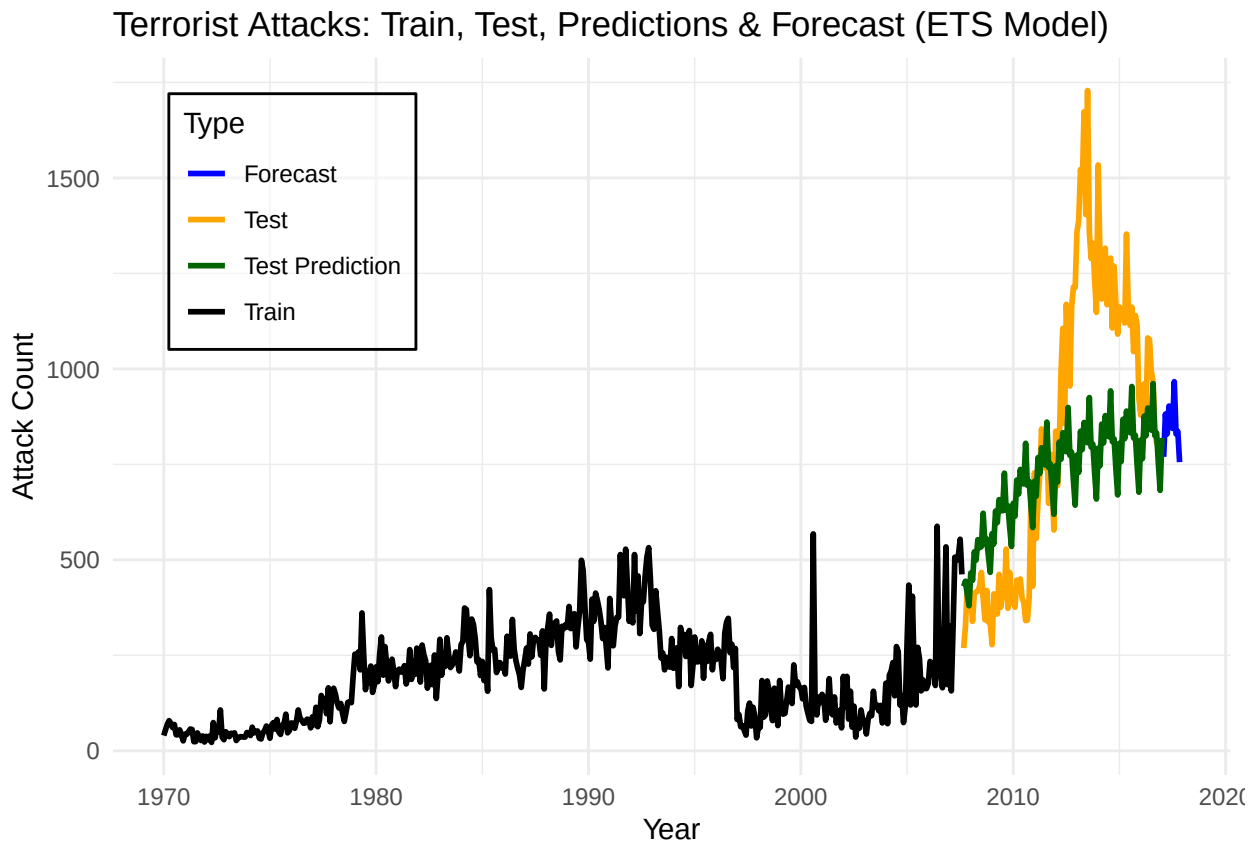
```r
# Prepare a data frame for predictions (test period predictions and extended forecast)
forecast_years <- as.numeric(time(fcast$mean))
df_pred <- tibble(
  Year = forecast_years,
  Prediction = as.numeric(fcast$mean),
  Type = c(rep("Test Prediction", length(test_ts)), rep("Forecast", 10))
)
```

```r
  # ------------------------------------------------------------------------------
  # 12. Plot the Actual vs. Forecasted Terrorist Attack Counts using ggplot2
  # ------------------------------------------------------------------------------
  # Adjust the plotting area directly in your interactive session
  options(repr.plot.width = 30, repr.plot.height = 6)

  # Recreate your plot with adjusted dimensions
  p <- ggplot() +
    geom_line(data = df_actual,
              aes(x = Year, y = Attack_Count, color = Type),
              size = 1) +
    geom_line(data = df_pred,
              aes(x = Year, y = Prediction, color = Type),
              size = 1) +
    labs(title = "Terrorist Attacks: Train, Test, Predictions & Forecast (ETS Model)",
         x = "Year",
         y = "Attack Count") +
    theme_minimal() +
    scale_color_manual(values = c("Train" = "black",
                                  "Test" = "orange",
                                  "Test Prediction" = "darkgreen",
                                  "Forecast" = "blue")) +
    theme(legend.position = c(0.05, 0.95),
          legend.justification = c("left", "top"),
          legend.background = element_rect(fill = alpha("white", 0.5)))
```

```
## 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.
```

```
## Warning: A numeric ‘legend.position‘ argument in ‘theme()‘ was deprecated in ggplot2
## 3.5.0.
## i Please use the ‘legend.position.inside‘ argument of ‘theme()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```r
# Display the adjusted plot
print(p)
```



This code fits an ETS model (MMM) to the training data up to December 2016 and forecasts terrorist attacks for the next 48 months. The forecast is generated with 90% and 95% confidence intervals. The resulting plot shows the forecasted values starting in 2017, with the original time series overlaid in black for context. The x-axis is limited to start from 2010 to provide historical perspective alongside the forecast.

```r
library(forecast)

# Define the training data as all data up to December 2016
train_ts <- window(t_attack, end = c(2016, 12))

# Manually specify the forecast horizon (e.g., 48 months after 2016)
forecast_horizon <- 48

# Fit the ETS model to the training data.
best_model <- ets(train_ts, model = "MMM")


# Generate the forecast starting from 2017 (after 2016) fo  r the specified horizon.
fcast <- forecast(best_model, h = forecast_horizon, level = c(90, 95))

# Determine the ending time from the forecast object for the x-axis limit.
end_time <- time(fcast$mean)[length(fcast$mean)]
```

```r
# Plot the forecast with the x-axis starting at 2010
plot(fcast,
     main = "Forecast of Global Terrorist Attacks (ETS) Starting after 2016",
     xlab = "Year",
     ylab = "Attack Count",
     xlim = c(2010, end_time))

# Overlay the full original time series (displayed in black)
lines(t_attack, col = "black")
```