London Homicides

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Introduction

On 31 December 2018, Iain Agar published his work on the 10 year history of homicides in London on Kaggle . The data set was named "mmap.xlsx" This is the link to the Kaggle site: https://www.kaggle.com/iainagar/london-homicide-data-20082018 (https://www.kaggle.com/iainagar/london-homicide-data-20082018)

The London homicide dataset

Ager stated that there was no readily available London homicide dataset in the public domain. His mmap dataset was constructed using a number of open resources, most notably the murdermap (http://www.murdermap.co.uk/) website.

I imported his data set mmap.xlsx into R which covered the period from Jan 2008 to Dec 2018, and renamed the data frame it as mmap2018

I contacted the publisher of the murdermap.co.uk website in London who generously agreed to provide additional homicide cases for the period 1 Jan 2019 to 1 November 2020.

I bound the additional 2019 and 2020 cases to the orginal mmap.xlsx data set produced by Iain Agar and named the merged frame set as mmap.

I imported details of the borough population from Iain Agers data set boroughpop and integrated this into the mmap data frame.

I completed a range of data wrangling to ensure that the factor descriptions were consistent across years as between the original 2018 data set and the augmented 2019 and 2020 data sets. For example I before bringing the three data sets back into R. I had to manually define ethnicity for the data in 2019 and 2020 based a review of the murder map people profiles on the murder map website and aligned consistent names for cause of death which changed in the most recent murder map profiles. I extended the ID variable created by Agar in 2018 and applied the same ID methodology to the incremental cases in 2019 and 2020. Finally I needed to create consistency across the period 2000 to 2020 in the status variable which the publisher had made numerous changes to in 2019 and 2020.

The bedrock of the analysis and visualisation work in this RMD is the mmap data frame - which covers the period January 2008 to November 1 2020.

For time series analysis comparing variables by year I filtered to just include the full year data from 2008 to 2019. For sociographic and demographic data that is not comparing variables by year I used all cases through 1 November, 2020.

Victim demographics summary [Do a final update on the overall so whats]

- For the previous 10 years in London males have made up more than 77% of homicide victims
- The proportion of male victims peaks between the ages of 17-24, especially for Black vicitims
- By contrast, those within the grouped category White or White British saw victimisation volumes peak later, and remained highest for all age groups 30+
- During the most recent 3-years (2016-2018) there has been a fall in the total volume of victims aged 12 and under (including infanticide) and aged 35 and over, compared to the corresponding period (2013-2015)
- For those aged between 17-34 there has been a rise in the total volume of victims by almost 50% during the same observed time frame
- Female homicides increased in 2018 when compared with 2017, in contrast the number of male victims saw a reduction.

```
# Install packages
detach("package:sp", unload = TRUE)
install.packages("sp", dependencies=TRUE)
install.packages("dplyr")
install.packages("tidyr")
install.packages("lubridate")
install.packages("ggplot2")
install.packages("sp")
install.packages("vcd")
install.packages("GGally")
install.packages("colourpicker")
install.packages("rgeos")
install.packages("Hmisc",repos = "http://cran.us.r-project.org")
install.packages("gapminder")
install.packages("googleVis")
install.packages("plotly")
install.packages('acepack')
install.packages("cowplot")
install.packages("maptools")
install.packages("ggmap")
install.packages("broom")
install.packages("mapproj")
install.packages("leaflet")
install.packages("ggmosaic")
install.packages("leaflet.extras")
install.packages("tidyverse")
install.packages("reshape")
install.packages("ggridges")
install.packages("rgdal")
```

Import data sets, integrate into a mmap data frame covering cases from 2008 to 2020

Consolidate the mmap data bases from 2018, 2019 and 2020 to update the original Iain Agar 2018 data set to November 2020.

String variables were reformatted as factors.

```
## Loading libraries and reading data

# The libraries used in this rmd are provided in the code below.

library(knitr)
library(readxl)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.3
```

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

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

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

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.0.3
```

```
library(tibble)
library(stringr)
library(lubridate)
```

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

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

```
library(leaflet)
```

```
## Warning: package 'leaflet' was built under R version 4.0.3
```

```
library(leaflet.extras)
```

```
## Warning: package 'leaflet.extras' was built under R version 4.0.3
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.3
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

```
library(reshape)
```

```
## Warning: package 'reshape' was built under R version 4.0.3
```

```
##
## Attaching package: 'reshape'
```

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

```
## The following objects are masked from 'package:tidyr':
##
##     expand, smiths
```

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

```
library(ggridges)
```

```
## Warning: package 'ggridges' was built under R version 4.0.3
```

```
## Import the data - London Homicide data, and London borough population data.

# import original Iain Ager mmap.xlsx into R and save it as mmap18
library(readxl)
mmap18 <- read_excel("C:/Users/david/OneDrive - RMIT University/Data Visualisation and Communication/London Homicides Projec
t/mmap.xlsx")


# clean up the orginal data set and eliminate redundant variables

library(dplyr)
mmap18 = select(mmap18, -6,-7,-8,-13,-14,-16,-17,-18,-19)

# reorder columns
mmap18 <- mmap18[ , c(2, 1, 8, 9,5,4,3,10,6,11,7)]


# update column names
colnames(mmap18) <- c("date", "ID", "latitude", "longitude", "ethnicity", "sex", "age", "ageGroup", "weapon", "borough", "st
atus")

# change strings to factors
mmap18 <- mmap18 %>% mutate_if(is.character, as.factor)

# import the 2019 murder map data into R
mmap19 <- read_excel("C:/Users/david/OneDrive - RMIT University/Data Visualisation and Communication/London Homicides Projec
t/Murdermap2019.xlsx")

mmap19 = select(mmap19, -3,-6)


mmap19$longitude <- as.numeric(mmap19$longitude)


# change strings to factors
mmap19 <- mmap19 %>% mutate_if(is.character, as.factor)

colnames(mmap19) <- c("date", "ID", "latitude", "longitude", "ethnicity", "sex", "age", "ageGroup", "weapon", "borough", "st
atus")

# import the 2019 murder map data into R
mmap20 <- read_excel("C:/Users/david/OneDrive - RMIT University/Data Visualisation and Communication/London Homicides Projec
t/Murdermap2020.xlsx")

mmap20 = select(mmap20, -3,-6)

mmap20 <- mmap20 %>% mutate_if(is.character, as.factor)

colnames(mmap20) <- c("date", "ID", "latitude", "longitude", "ethnicity", "sex", "age", "ageGroup", "weapon", "borough", "st
atus")


##Merge mmap19 and mmap 20 using rbind function
mmap19_20 = rbind(mmap19,mmap20)

#change the levels for "sex" to align with mmap18
mmap19_20 <- mmap19_20 %>%
  mutate(sex = fct_recode(sex,
                          "M" = "Male",
                          "F" = "Female"))

# merge mmap19_20 with mmap18 to create mmap (integrated cases from 2008 to 2020)
mmap = rbind(mmap18, mmap19_20)
```

##Data Wrangling:

- Create an additional weapon used column, reducing levels from 11 to 4 (Knife, Assault, Gun and Other)
- Replicate the date column, broken down into year, date and month
- Align names for counties in borough variable
- Join the borough population data into the mmap data frame

```
# See all levels in weapon variable
levels(mmap$weapon)
```

```
##  [1] "Arson"         "Blunt Object"  "Gun"           "Knife"
##  [5] "Ligature"      "None"          "Other"         "Poison"
##  [9] "Strangulation" "Vehicle"       "Drug"          "Fire"
## [13] "Unknown"
```

```
# Replicate weapon column to enable re-factoring
mmap2 = cbind(mmap, weapon2=rep(mmap$weapon))



# Change levels in Weapon column to include just 4 levels Knife, Gun, Assault and Other
mmap3 <- mmap2 %>%
  mutate(weapon = fct_recode(weapon,
                             "Other" = "Arson",
                             "Other" = "Vehicle",
                             "Other" = "Drug",
                             "Other" = "Poison",
                             "Other" = "Unknown",
                             "Other" = "Ligature",
                             "Other" = "Strangulation",
                             "Other" = "Fire",
                             "Knife" = "Knife",
                             "Gun" = "Gun",
                             "Other" = "Blunt Object",
                             "Assault" = "Assault",
                             "Assault" = "None"))
```

```
## Warning: Problem with `mutate()` input `weapon`.
## i Unknown levels in `f`: Assault
## i Input `weapon` is `fct_recode(...)`.
```

```
## Warning: Unknown levels in `f`: Assault
```

```
# clean up the factor names within borough variable to enable the bind with boroughpop (ensure they are exactly the same!)
mmap3 <- mmap3 %>% mutate(borough = fct_recode(borough,
                             "Westminster" = "City of Westminster",
                             "Kingston upon Thames" = "Kingston",
                             "Kingston upon Thames" = "Kingston-upon-Thames",
                             "Kingston upon Thames" = "Kingston Upon Thames",
                             "Kingston upon Thames" = "Kingston-Upon-Thames",
                             "Richmond upon Thames" = "Richmond"))
```

```
## Warning: Problem with `mutate()` input `borough`.
## i Unknown levels in `f`: Kingston, Kingston Upon Thames
## i Input `borough` is `fct_recode(...)`.
```

```
## Warning: Unknown levels in `f`: Kingston, Kingston Upon Thames
```

```
# Replicate date column
mmap4 = cbind(mmap3, date2=rep(mmap3$date))


# create new data frame mmap5 and separate date variable from mmap4 into year, month and day and save them as integers
mmap5 = separate(mmap4, date2, c("year", "month", "day"), "-") %>%
  mutate_if(is.character, as.integer)

mmap5 <- mmap5 %>% mutate(status = fct_recode(status,
                             "Awaiting Trial" = "Awaiting trial",
                             "Awaiting Trial" = "Awaiting Outcome",
                             "Solved" = "Solved",
                             "Solved" = "Self Defence",
                             "Unsolved" = "Suspect dead",
                             "Unsolved" = "Unsolved"))



mmap5 <- mmap5 %>% mutate(ethnicity = fct_recode(ethnicity,
                             "Other" = "Mixed",
                             "Black" = "Black",
                             "Black" = "Black or Black British",
                             "White" = "White",
                             "White" = "White or White British",
                             "Asian" = "Asian",
                             "Asian" = "Asian or Asian British",
                             "Other" = "Unknown",
                             "Other" = "NA",
                             "Other" = "Any Other Ethnic Appearance"))
```

```
## Warning: Problem with `mutate()` input `ethnicity`.
## i Unknown levels in `f`: NA
## i Input `ethnicity` is `fct_recode(...)`.
```

```
## Warning: Unknown levels in `f`: NA
```

```
levels(mmap5$ethnicity)
```

```
## [1] "Other" "Asian" "Black" "White"
```

```
# read the boroughpop file into R to access borough population data
boroughpop <- read_excel("boroughpop.xlsx")


# change column names in boroughpop
colnames(boroughpop) = c("borough", "population")

# join mmap5 and boroughpop data frames using "borough"
mmap6 <- left_join(mmap5, boroughpop, by = "borough")


# make bourough a factor variable
mmap6$borough <- as.factor(mmap6$borough)


# create new variable for counting homicides by borough and homicides per 100,000 population by borough
mmap6_sum <- mmap6 %>% select(borough, population) %>% group_by(borough) %>% mutate(count = n(), population = population, ho
micide_rate = (count*100000)/11/population) %>% distinct()

# create mmap 7 by filtering mmap5 to just include full year data to end of calendar 2019
mmap7 <- mmap6 %>% filter(year != "2020")


mmap7 <-data.frame(mmap7)

# remove NA cases for ethicity
mmap8 <- mmap7 %>% filter(ethnicity != "NA")

# save mmap6_sum as data frame and name the data frame mmap9
mmap9 <- as.data.frame(mmap6_sum)

# create new data frame mmap10 with just borough and homicide rate variables
mmap10 <- mmap9 %>% select(borough, homicide_rate)

# round the homicide_rate variable to 2 decimal places
mmap10$homicide_rate <- round(mmap10$homicide_rate, 2)

# arrange the data frame based on homicide rate variable descending order
mmap10 <- mmap10 %>% arrange(desc(mmap10$homicide_rate))
```

Data exploration

I have extended Iain's data exploration to include calender 2019 data and YTD calendar 2020 data sourced from the murder map web site publisher in London.

Based on this additional data there were 149 new homicide cases in 2019 and 107 new homicide cases year-to-data at 1 November 2020.

I will include the 2019 data for exploration of the 11 year annual temporal trends from 2008 to 2019 and will include both the 2019 and 2020 cases for the 12 year mix of homicides by demographics such as gender, ethnicity and age. This is because I only have year to data homicide data for the 2020 cases.

key observations about the temporal trends in london homicide for the 11 years from 2008 to 2019:

- In absolute terms the number of homicides was in decline from 2008 to 2014 where it hit a low of 90 cases
- From 2015 to 2019 the trend was upwards in homicides peaking at 149 cases in 2019
- When expressed as homicide cases per 100,000 population the rate of homicides fell from 2008 to 2014 to a low of 1.06 homicides per 100,000 but has steadily increased again from 2015 to 2019 to a peak of 1.66 homicides per 100,000, which is the highest rate since 2008

Data exploration - overall annual trends in homicide cases for London from 2008 to 2019

sources: https://worldpopulationreview.com/world-cities/london-population (https://worldpopulationreview.com/world-cities/london-population) https://www.trustforlondon.org.uk/data/population-over-time/ (https://www.trustforlondon.org.uk/data/population-over-time/)

I also have the borough population data for Greater London in 2016 based on the ONS survey for that year which added to 8.82 million.

But I have not found any single time series data for the population of Greater London. The boroughpop population by borough that we use in this analysis was taken from an OLS survey in 2016, which was the latest OLS survey of population

I have interpolated the trends in London population from this OLS survey in 2017 and other point estimates for years 2010, 2011, 2015, 2018 and 2020.

My estimates for population of greater London for each year from calendar 2008 to 2019 including annual data points from the sources above are:

2008:7.8m, 2009:7.9m, 2010:8.0m, 2011:8.15m, 2012:8.25m, 2013:8.35m, 2014:8.5m, 2015:8.7m, 2016: 8.82m, 2017: 8.85m, 2018: 8.9m, 2019:9.0m

I have used mmap7 for the review of 2008 to 2019 temporal trends in homicide rates in London which excludes the 2020 cases.

```
str(mmap7)
```

```
## 'data.frame':    1477 obs. of  16 variables:
##  $ date      : POSIXct, format: "2008-01-03" "2008-01-06" ...
##  $ ID        : Factor w/ 1584 levels "MM00001","MM00002",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ latitude  : num  51.6 51.5 51.5 51.6 51.5 ...
##  $ longitude : num  -0.064 -0.132 0.179 -0.276 -0.114 ...
##  $ ethnicity : Factor w/ 4 levels "Other","Asian",..: 3 4 1 4 1 4 3 3 4 3 ...
##  $ sex       : Factor w/ 2 levels "F","M": 2 1 2 2 2 2 2 2 2 2 ...
##  $ age       : num  18 44 18 58 25 36 18 51 43 19 ...
##  $ ageGroup  : Factor w/ 10 levels "A. Child 0-6",..: 4 7 4 9 6 7 4 8 7 4 ...
##  $ weapon    : Factor w/ 4 levels "Other","Gun",..: 3 3 3 1 3 1 3 1 3 3 ...
##  $ borough   : Factor w/ 33 levels "Barking and Dagenham",..: 10 22 3 4 22 32 10 19 2 9 ...
##  $ status    : Factor w/ 3 levels "Awaiting Trial",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ weapon2   : Factor w/ 13 levels "Arson","Blunt Object",..: 4 4 4 2 4 8 4 7 4 4 ...
##  $ year      : int  2008 2008 2008 2008 2008 2008 2008 2008 2008 2008 ...
##  $ month     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ day       : int  3 6 7 8 13 17 21 23 23 26 ...
##  $ population: num  332705 324048 246124 329102 324048 ...
```

```
mmap7$year <- as.factor(mmap7$year)

# Number of homicides in London by Year, 2008-2019
p1 <- ggplot(mmap7, aes(x = year)) +
  geom_bar(stat='count', fill = "navyblue", color = "white") +
  geom_label(stat = "count", aes(label = ..count..,y = ..count..), size = 3) +
  labs(x = "Year", y = "Number of Homicides", title = "London homicides, 2008-2019")+
  theme_bw()


# save version to convert to plotly

library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.0.3
```

```
##
## Attaching package: 'plotly'
```

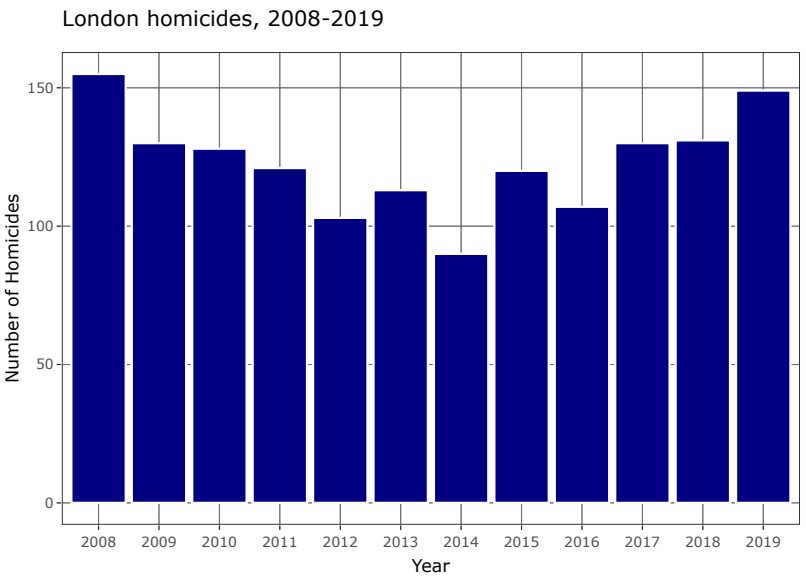
```
## The following object is masked from 'package:reshape':
##
##     rename
```

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

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

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
p2 <- ggplot(mmap7, aes(x = year)) +
  geom_bar(stat='count', fill = "navyblue", color = "white") +
  labs(x = "Year", y = "Number of Homicides", title = "London homicides, 2008-2019")+
  theme_bw()
ggplotly(p2)
```



```
# calculate sum borough population based on the 2016 ONS data for boroughs
GreaterLondonPop <- sum(boroughpop$population)
GreaterLondonPop
```

```
## [1] 8825001
```

```
# create vector for Greater London Population for 2008 to 2019
 GreaterLondonPopMillions <- c(7.8, 7.9, 8.0, 8.15, 8.25, 8.35, 8.5, 8.7, 8.82, 8.85, 8.9, 9.0)

# Greater London Population in 100,000s
GreaterLondon <- GreaterLondonPopMillions*10

# Greater London total homicides

LondonHomicides <- c(155, 130, 128, 121, 103, 113, 90, 120, 107, 130, 131, 149)

Year <- c(2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019)

# merge vectors into a data frame

London <- data.frame(Year, LondonHomicides, GreaterLondon)

London
```

```
##    Year LondonHomicides GreaterLondon
## 1  2008             155          78.0
## 2  2009             130          79.0
## 3  2010             128          80.0
## 4  2011             121          81.5
## 5  2012             103          82.5
## 6  2013             113          83.5
## 7  2014              90          85.0
## 8  2015             120          87.0
## 9  2016             107          88.2
## 10 2017             130          88.5
## 11 2018             131          89.0
## 12 2019             149          90.0
```

```
# Number of homicides per 100,000 population, 2008-2019

LondonHomicideRate <- London %>% mutate(Homicide_Rate = LondonHomicides/GreaterLondon)

LondonHomicideRate$Homicide_Rate <- round(LondonHomicideRate$Homicide_Rate,2)

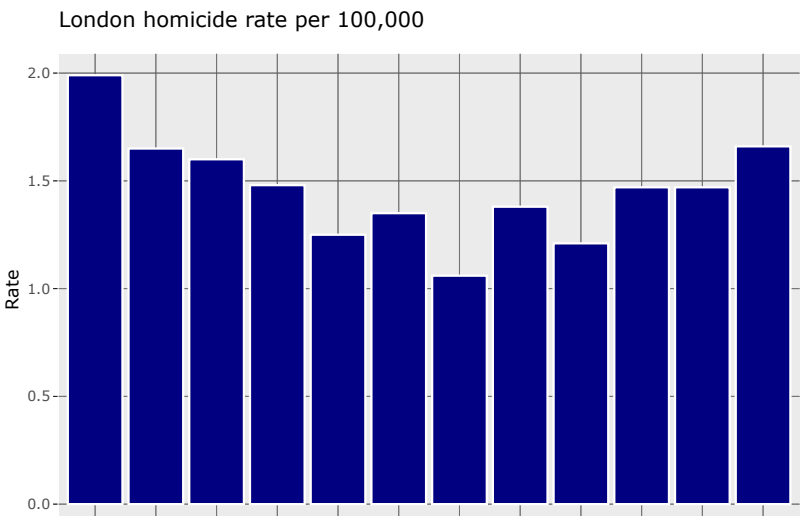
LondonHomicideRate$Year <- as.factor(LondonHomicideRate$Year)

str(LondonHomicideRate)
```

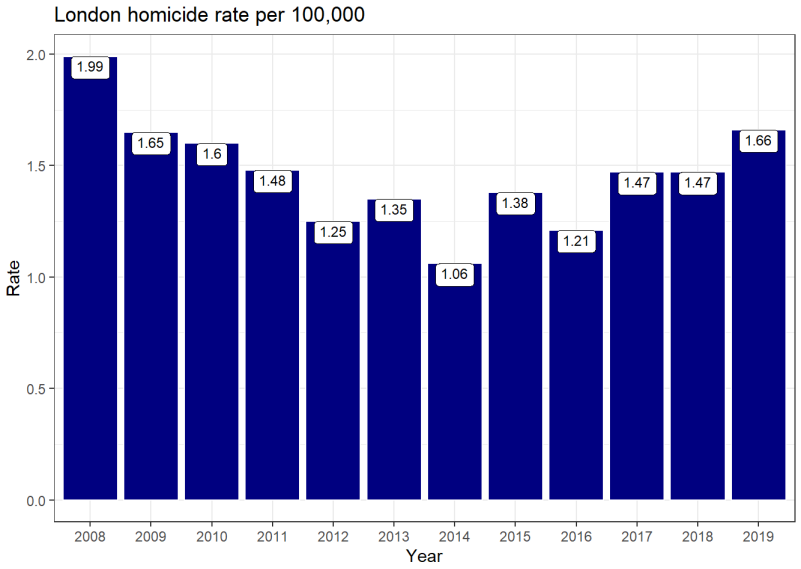
```
## 'data.frame':    12 obs. of  4 variables:
##  $ Year           : Factor w/ 12 levels "2008","2009",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ LondonHomicides: num  155 130 128 121 103 113 90 120 107 130 ...
##  $ GreaterLondon  : num  78 79 80 81.5 82.5 83.5 85 87 88.2 88.5 ...
##  $ Homicide_Rate  : num  1.99 1.65 1.6 1.48 1.25 1.35 1.06 1.38 1.21 1.47 ...
```

```
# chart Number of Homicides in London per 100,000 population, 2008 - 2019
p3 <- ggplot(LondonHomicideRate, aes(x= Year, y=Homicide_Rate)) +
geom_bar(stat="identity", fill="navyblue", color = "white")+
  geom_label(aes(label = Homicide_Rate, vjust = 1),size = 4)+
  labs(x = "Year", y = "Rate", title = "London homicide rate per 100,000")+
  theme_bw()


# create plotly version of the chart
p4 <- ggplot(LondonHomicideRate, aes(x=Year, y=Homicide_Rate)) +
geom_bar(stat="identity", fill="navyblue", color = "white")+
  labs(x = "Year", y = "Rate", title = "London homicide rate per 100,000")
ggplotly(p4)
```




```
# save version for juxtapose chart
p4 <- ggplot(LondonHomicideRate, aes(x= as.factor(Year), y=Homicide_Rate)) +
geom_bar(stat="identity", fill="navyblue", color = "white")+
  geom_label(aes(label = Homicide_Rate, vjust = 1),size = 3)+
  labs(x = "Year", y = "Rate", title = "London homicide rate per 100,000")+
  theme_bw()
p4
```




```
library(cowplot)
```

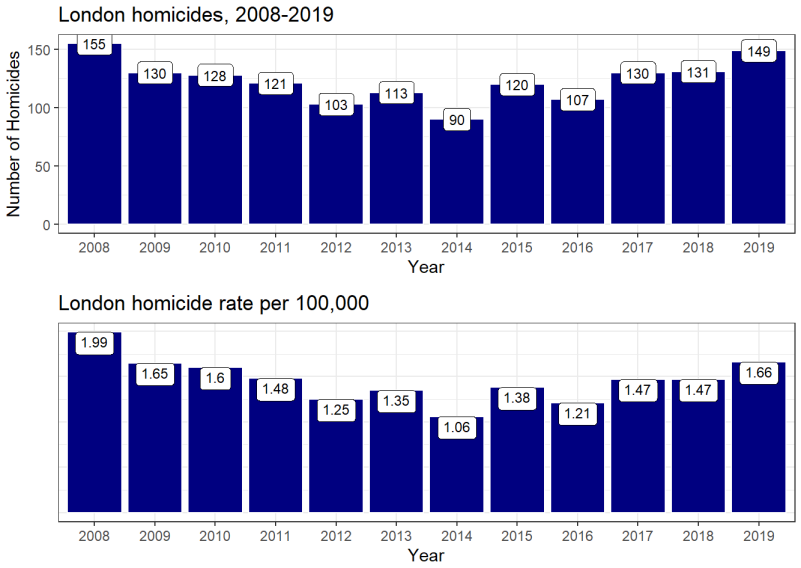
```
## Warning: package 'cowplot' was built under R version 4.0.3
```

```
##
## Attaching package: 'cowplot'
```

```
## The following object is masked from 'package:reshape':
##
##     stamp
```

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

```
# juxtapose the plots of homicides and homicide rate
plot_grid(p1, p4  + theme(axis.title.y=element_blank(),
                                    axis.text.y=element_blank(),
                                    axis.ticks.y = element_blank()), ncol=1, align="v",
          rel_heights = c(2,2))
```



Gender

The proportion table for the homicide cases between 2008 and 2020 has been provided to show an overview of gender breakdown. During the 12 years male victims accounted for over 77% of all recorded homicide cases in London during the past 12 years.

```
sex <- table(mmap5$sex) # Victim gender table
addmargins(sex) # Plus totals
```

```
##
##    F    M  Sum
##  369 1215 1584
```

```
round((prop.table(sex)),digits = 2) # Proportion
```

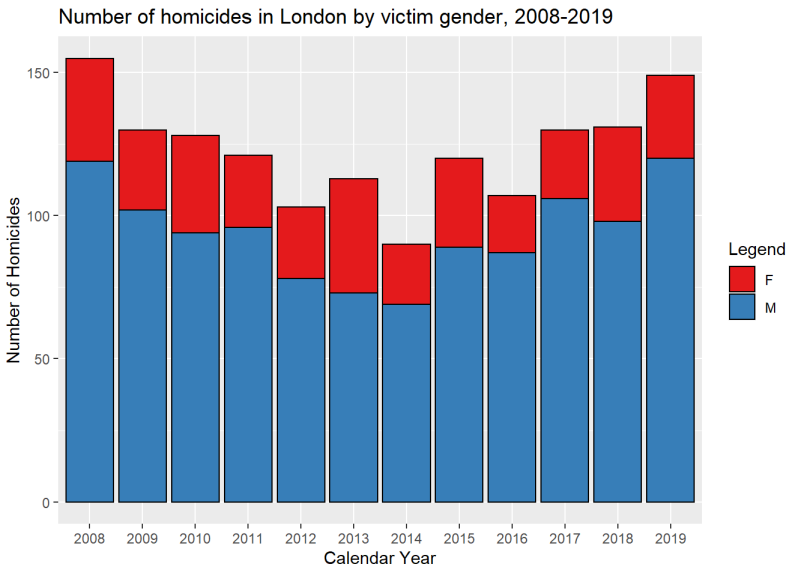
```
##
##    F    M
## 0.23 0.77
```

The bar chart below shows the 12-year trend for male and female homicide victims in London. There has been consecutive increases in the volume of female victims between 2016 and 2019.

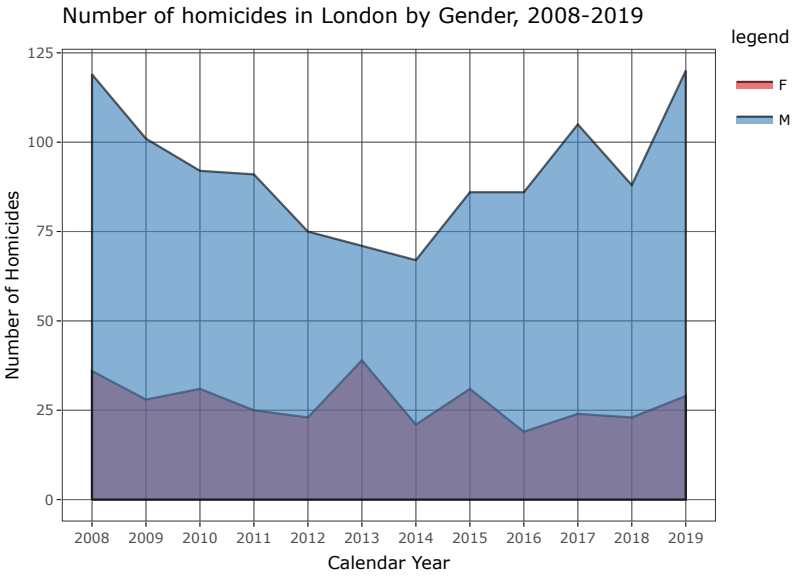
```
# Bar Chart showing Number of homicides in London by victim gender, 2008-2019
ggplot(mmap7, aes(x = year, fill = sex)) +
  geom_bar(stat='count', position = 'dodge', colour = "black") +
  scale_x_discrete(breaks = c(2008:2019)) +
  scale_fill_brewer(palette = "Set1") +
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "Legend", title = "Number of homicides in London by victim gen
der, 2008-2019")
```


```
# Stacked Bar Chart showing Number of homicides in London by victim gender, 2008-2019
ggplot(mmap7, aes(x = year, fill = sex)) +
  geom_bar(stat='count', facets = sex ~., colour = "black") +
  scale_x_discrete(breaks = c(2008:2019)) +
  scale_fill_brewer(palette = "Set1") +
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "Legend", title = "Number of homicides in London by victim gen
der, 2008-2019")
```

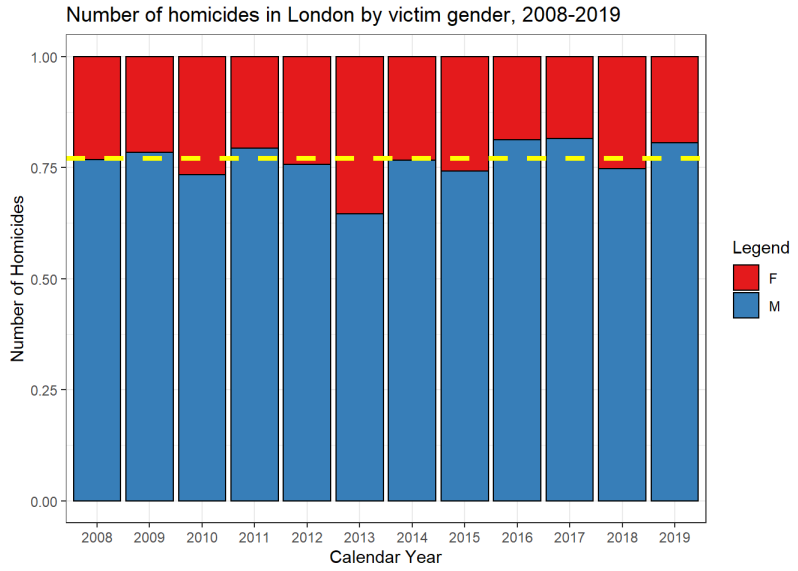
```
## Warning: Ignoring unknown parameters: facets
```



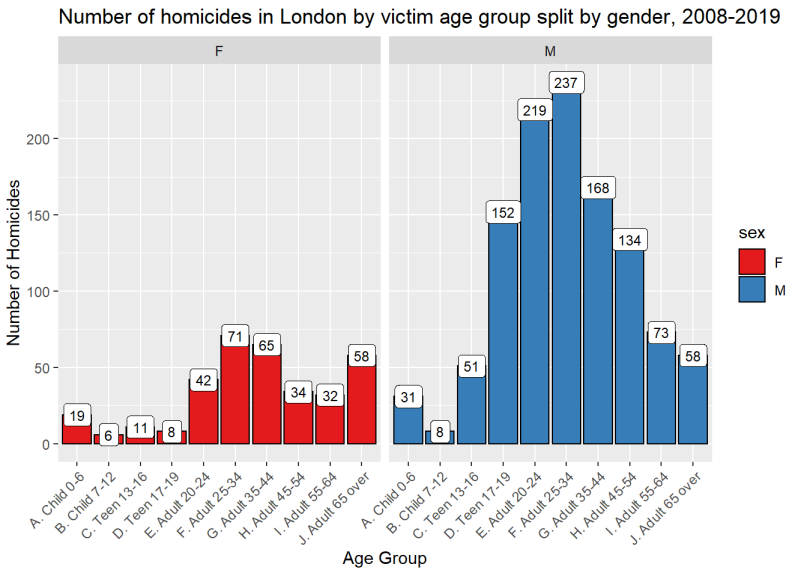
```
p18 <- ggplot(mmap8, aes(x = year, fill = sex)) +
  geom_density(stat='count', alpha = 0.6, colour = "black") +
  scale_x_continuous(breaks = c(2008:2020)) +
  scale_fill_brewer(palette = "Set1") +
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "legend", title = "Number of homicides in London by Gender, 2008-2019")+
   theme_bw()
ggplotly(p18)
```



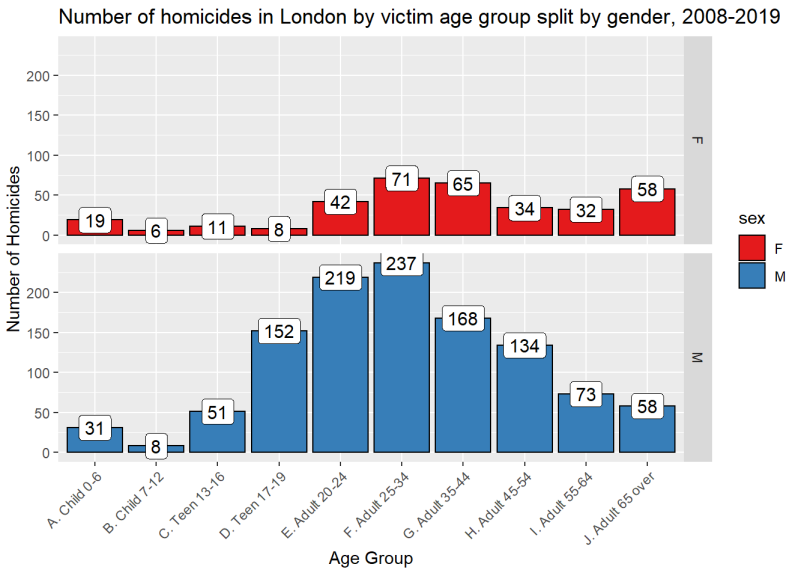

```
# Proportion Bar Chart showing Number of homicides in London by victim gender, 2008-2019 and mean male proportion at 77% in red
ggplot(mmap7, aes(x = year, fill = sex)) +
  geom_bar(stat='count', position = 'fill', colour = "black") +
  geom_hline(yintercept = 0.77, colour = "yellow",linetype="dashed", size =1.5) +
  scale_x_discrete(breaks = c(2008:2019)) +
  scale_fill_brewer(palette = "Set1") +
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "Legend", title = "Number of homicides in London by victim gender, 2008-2019")+
theme_bw()
```



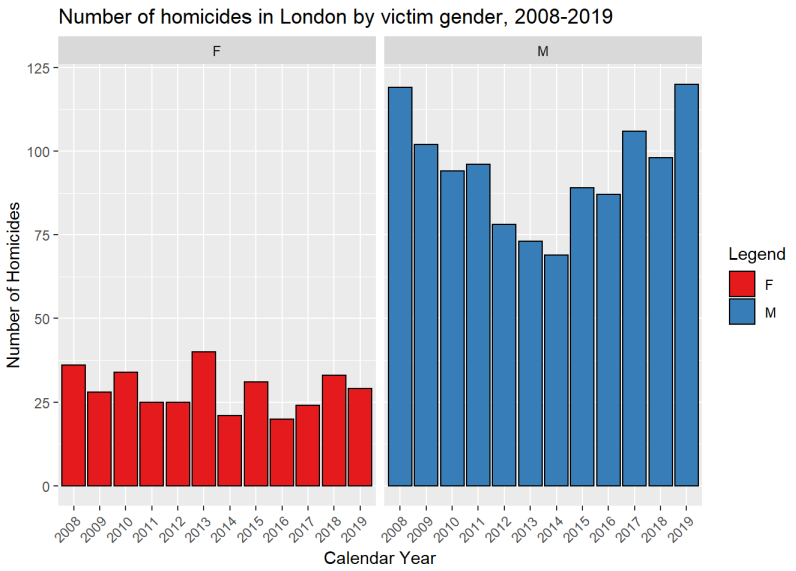

```
# Number of homicides in London by victim age group split by gender, 2008-2019 - facet vertical
ggplot(mmap7, aes(x = ageGroup, fill = sex)) +
  geom_bar(stat='count', color = "black") +
    scale_fill_brewer(palette = "Set1") +
  facet_grid(.~sex)+
  geom_label(stat = "count", aes(label = ..count..,y = ..count..), fill = "white", size = 3) +
  labs(x = "Age Group", y = "Number of Homicides", title = "Number of homicides in London by victim age group split by gender, 2008-2019")+
  theme(axis.text.x = element_text(angle = 45, hjust=1))
```




```
# Number of homicides in London by victim age group split by gender, 2008-2019 - facet horizontal
ggplot(mmap7, aes(x = ageGroup, fill = sex)) +
  geom_bar(stat='count', color = "black") +
    scale_fill_brewer(palette = "Set1") +
  facet_grid(sex~.)+
  geom_label(stat = "count", aes(label = ..count..,y = ..count..), fill = "white", size = 4) +
  labs(x = "Age Group", y = "Number of Homicides", title = "Number of homicides in London by victim age group split by gende
r, 2008-2019")+
  theme(axis.text.x = element_text(angle = 45, hjust=1))
```

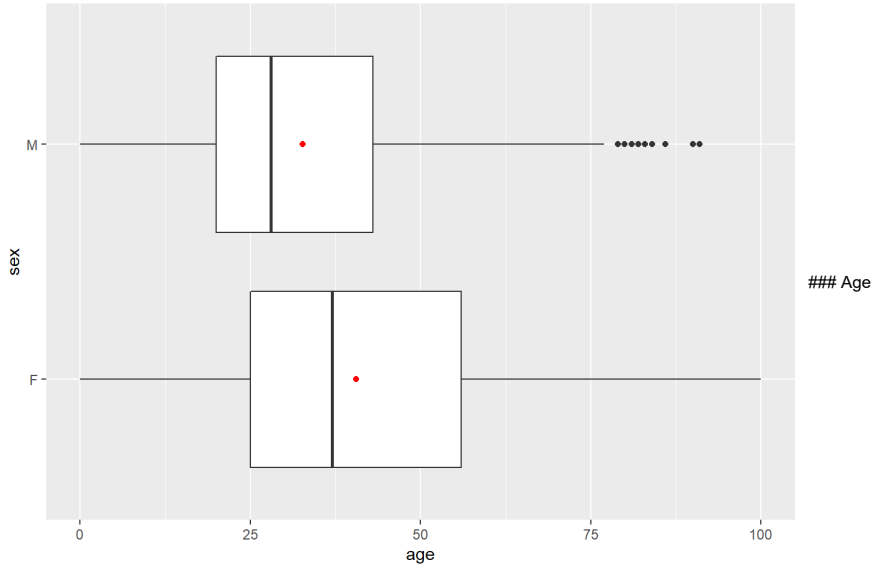



```
# Number of homicides in London by gender for each year between 2008-2019 - facet vertical
ggplot(mmap7, aes(x = year, fill = sex)) +
  geom_bar(stat='count', colour = "black") +
   facet_grid(.~sex)+
  scale_x_discrete(breaks = c(2008:2019)) +
  scale_fill_brewer(palette = "Set1") +
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "Legend", title = "Number of homicides in London by victim gen
der, 2008-2019")+
  theme(axis.text.x = element_text(angle = 45, hjust=1))
```




```
# boxplot of age of victim by gender of victim
qplot (x = age, y = sex, data = mmap5, geom = "boxplot") +
  stat_summary(fun.y="mean", colour="red", geom="point")
```

```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```



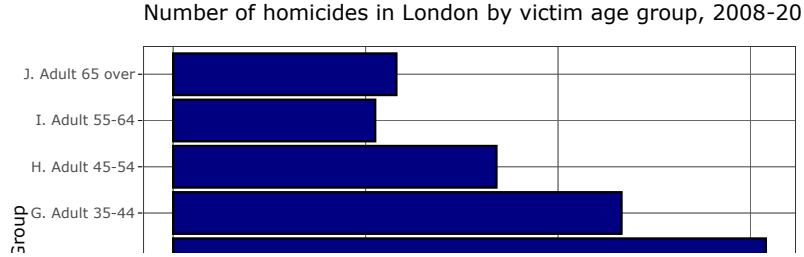
Age

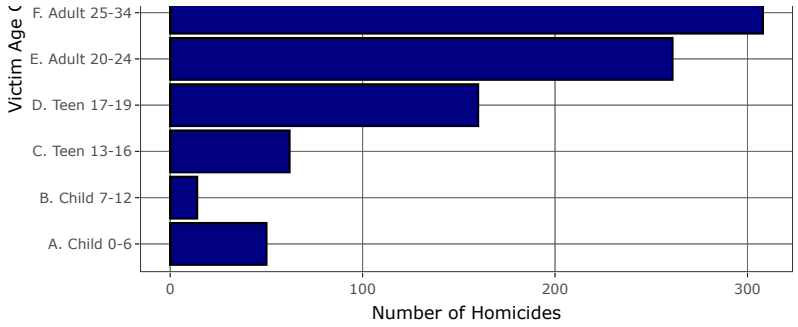
The age charts show a pronounced acute age profile for male victims, with a significant spike in the 17 to 25 age range, with the volume of victims declining with age from 26 onwards. The profile for female victims appears more diffused although totals are visibly higher on the second chart from ages 20 to 40.

The data table provide an annual breakdown for each grouped age.

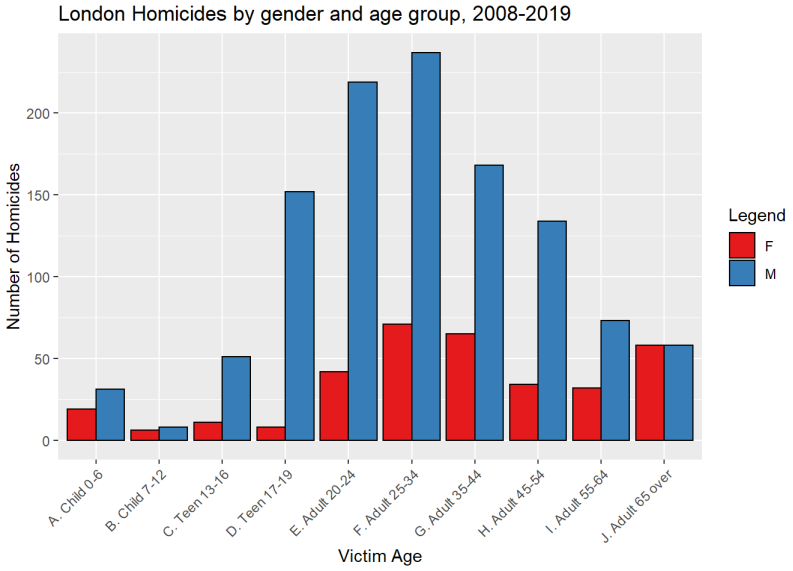
```
# Number of homicides in London by victim age group, 2008-2019
p5 <- ggplot(mmap7, aes(x = ageGroup)) +
  geom_bar(stat='count', fill = "navyblue", color = "black") +
  geom_label(stat = "count", aes(label = ..count..,y = ..count..)) +
  labs(x = "Victim Age Group", y = "Number of Homicides", title = "Number of homicides in London by victim age group, 2008-2
019")+
  coord_flip()+
  theme_bw()
ggplotly(p5)
```

```
## Warning in geom2trace.default(dots[[1L]][[1L]], dots[[2L]][[1L]], dots[[3L]][[1L]]): geom_GeomLabel() has yet to be imple
mented in plotly.
##   If you'd like to see this geom implemented,
##   Please open an issue with your example code at
##   https://github.com/ropensci/plotly/issues
```

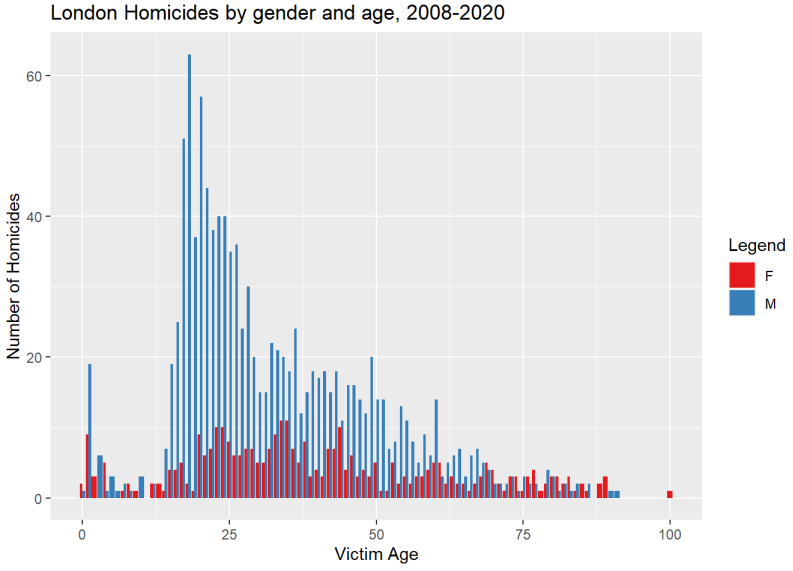





```
# histogram by ageGroup and gender
ggplot(mmap7, aes(x = ageGroup, fill = sex)) +
  geom_bar(position = "dodge", colour = "black") +
  ggtitle("London Homicides by gender and age group, 2008-2019") +
  scale_x_discrete("Victim Age") +
  scale_y_continuous("Number of Homicides") +
  labs(fill = "Legend") +
  scale_fill_brewer(palette = "Set1")+
  theme(axis.text.x = element_text(angle = 45, hjust=1))
```




```
# bar chart by age and gender - Iain Agar original chart extended to include 2019 data
ggplot(mmap7, aes(x = age, fill = sex)) +
  geom_bar(position = "dodge") +
  ggtitle("London Homicides by gender and age, 2008-2020") +
  scale_x_continuous("Victim Age")+
  scale_y_continuous("Number of Homicides") +
  labs(fill = "Legend") +
  scale_fill_brewer(palette = "Set1")
```



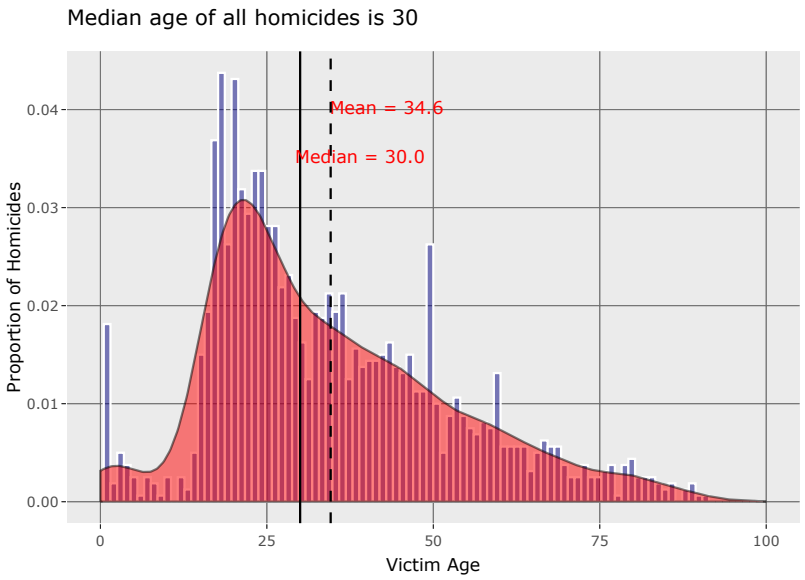

```
# calculate mean and median age for all homicides in London 2008-2020
meanAge <- mean(mmap5$age)
meanAge
```

```
## [1] 34.54987
```

```
medianAge <- median(mmap5$age)
medianAge
```

```
## [1] 30
```

```
# new bar chart with density overlay and box plot showing distribution of age across all data 2008 to 2020
p6 <- ggplot(mmap5, aes(x = age))+
  geom_density(fill = "red", alpha = 0.5)+
  geom_histogram(colour = "white", fill = "navyblue", aes(age, ..density..), alpha = 0.5, bins = 100)+
  labs(x = "Victim Age", y = "Proportion of Homicides", title = "Median age of all homicides is 30", subtitle ="Distribution
of homicides in
London by victim age 2008-2020" )+
  geom_vline(xintercept= median(mmap7$age)) +
  annotate("text",label = "Mean = 34.6",x = 43, y = 0.04, colour = "red") +
  geom_vline(xintercept= mean(mmap7$age),linetype=2) +
  annotate("text",label = "Median = 30.0",x = 39, y = 0.035, colour = "red")+
  scale_x_continuous(limits = c(0, 100))
ggplotly(p6)
```



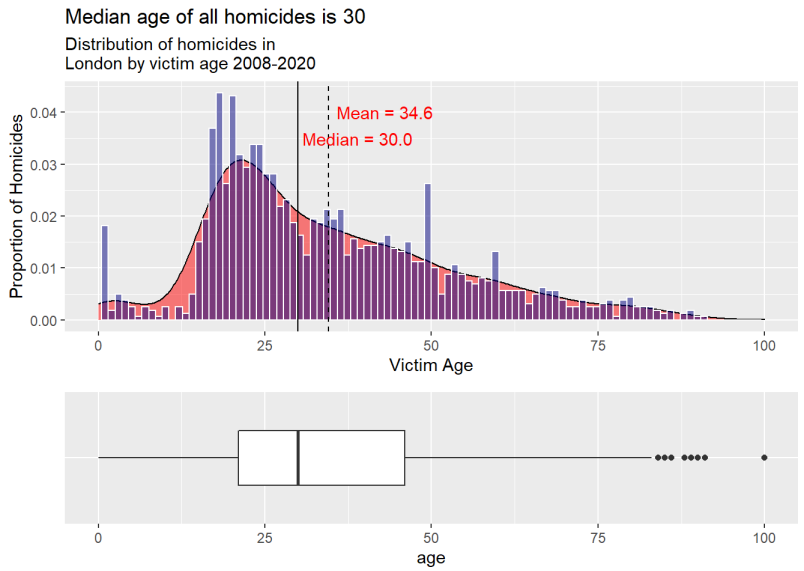

```
# create box plot for age
p7 <- ggplot(mmap5, aes(x = factor(1), y = age)) +
  geom_boxplot(width = .50) + scale_y_continuous(limits = c(0, 100))


# installing cowplot package
library(cowplot)

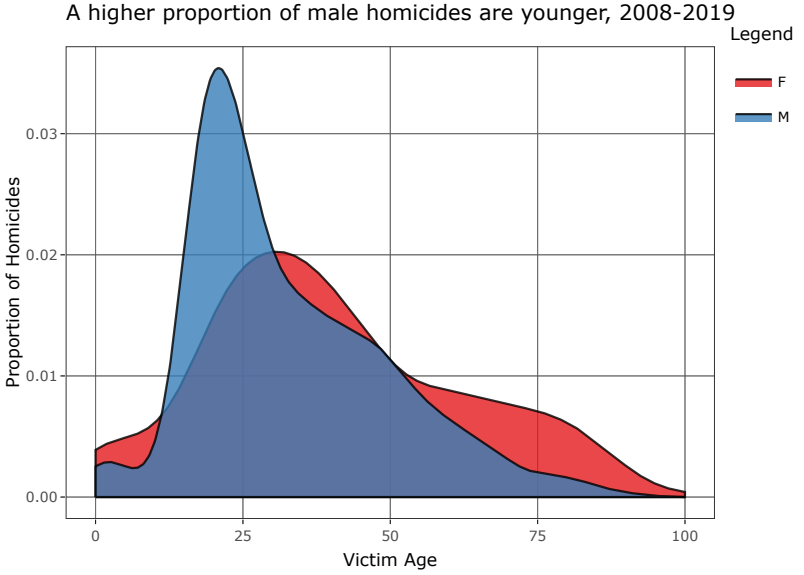
# juxtapose the histogram, density and boxplot for age across all data 2008 to 2020
p8 <- plot_grid(p6, p7 + coord_flip() + theme(axis.title.y=element_blank(),
                                              axis.text.y=element_blank(),
                                              axis.ticks.y = element_blank()), ncol=1, align="v",
          rel_heights = c(2,1))
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

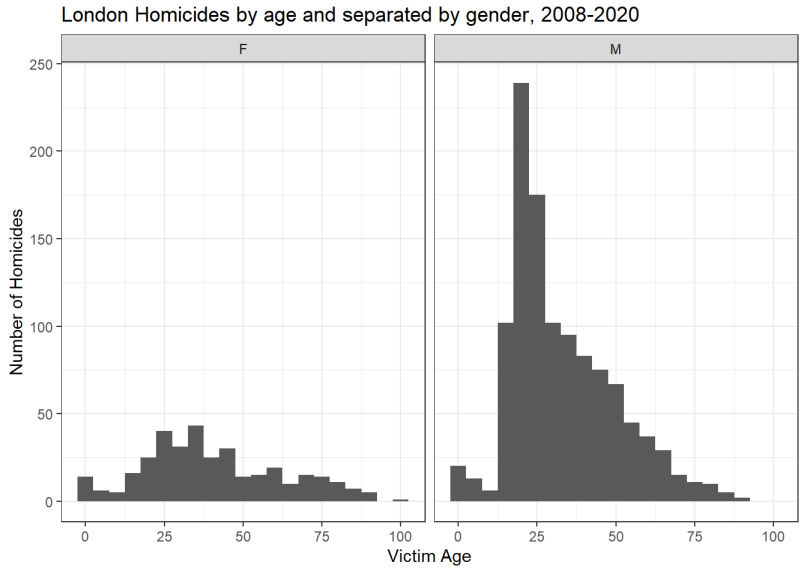
```
p8
```




```
# density chart by age and gender
p9 <- ggplot(mmap7, aes(x = age, fill = sex)) +
  geom_density(alpha = 0.8)+
  ggtitle("A higher proportion of male homicides are younger, 2008-2019")+
  scale_fill_brewer(palette = "Set1")+
    labs(fill = "Legend")+
  scale_x_continuous("Victim Age")+
  scale_y_continuous("Proportion of Homicides")+
 theme_bw()
ggplotly(p9)
```




```
# histogram chart by age and gender - Iain Agar original chart extended to include 2019 data
ggplot(mmap7, aes(x = age)) +
  geom_histogram(binwidth = 5) +
  facet_wrap(~ sex) +
  ggtitle("London Homicides by age and separated by gender, 2008-2020") +
  scale_x_continuous("Victim Age") +
  scale_y_continuous("Number of Homicides") +
  labs(fill = "Legend") +
  theme_bw()
```




```
# jitter box plots for victims by gender
p10 <- ggplot(data = mmap5, aes(x = sex, y = age)) + geom_boxplot(outlier.shape = NA) + geom_jitter(alpha = 1/5) +
  ylab("age of victim") +
  ggtitle("Men lose their lives to homicide at a younger age than women") +
  stat_summary(fun.y="mean", colour="red", geom="point",shape = 16, size = 4) +
  theme_minimal()
```

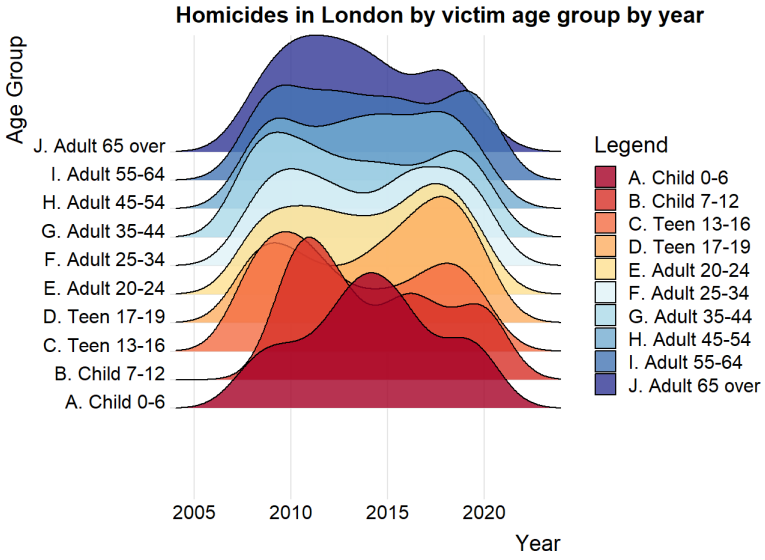
```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```

```
ggplotly(p10)
```


```
# ridgeline density chart for homicides by year and ageGroup
p11 <- ggplot(mmap5, aes(x = year, y = ageGroup, fill = ageGroup)) +
  geom_density_ridges(scale = 5, alpha = 0.8) +
   scale_fill_brewer(palette = "RdYlBu") +
  scale_y_discrete(expand = c(0, 3)) +     # will generally have to set the `expand` option
  scale_x_continuous(expand = c(0, 0)) +   # for both axes to remove unneeded padding
  coord_cartesian(clip = "off") + # to avoid clipping of the very top of the top ridgeline
  labs(x = "Year", y = "Age Group", fill = "Legend", title = "Homicides in London by victim age group by year")+
  theme_ridges()

p11
```

```
## Picking joint bandwidth of 1.32
```



Ethnicity and homicide

Defining people by ethnicity is highly subjective and can be deceptive so great care is needed here. The original mmap database created by Iain Ager inlcuded this variable and we have retained this variable and his ratings for the cases in 2019 and 2020.

The British ethnicity rating system includes these categories: (source https://en.wikipedia.org/wiki/Ethnic_groups_in_the_United_Kingdom (https://en.wikipedia.org/wiki/Ethnic_groups_in_the_United_Kingdom))

- White or White British
- Asian or Asian British (including Indian, Pakistani, Bangladeshi, Chinese, Other Asian)
- Black or Black British
- Mixed or Multiple
- Gypsy or Traveller
- Other Ethnic

We have treated arabic middle eastern people including Egyptians and Lebanese as white. Iranians, Afghans, Armenians, Turks and other non-arab middle eastern races as West Asian and part of the Asian ethic category.

People from the African Continent are black, as well as Brazilians people of African descent. Polynesians, Papuans and Melanesians are also regarded as black.

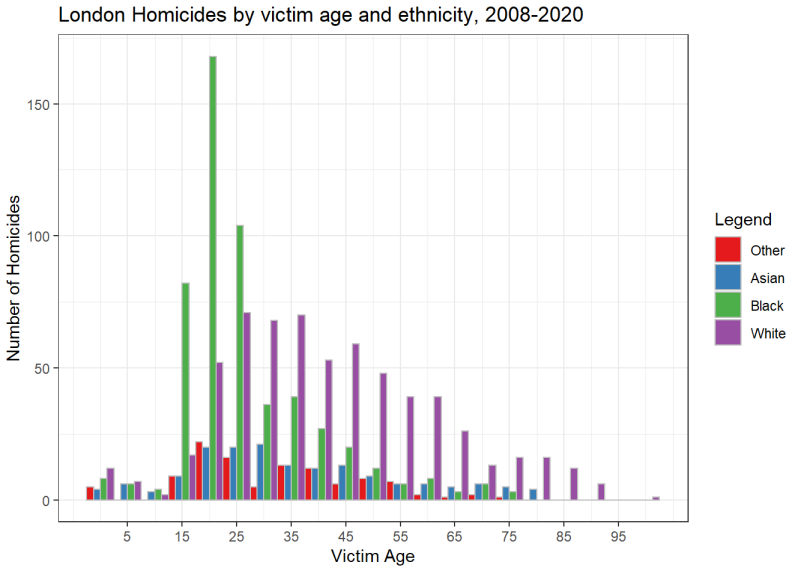
We have used these categories but simplified the levels in the variable ethnicity to include: * White * Asian * Black * Mixed * Other Ethic (including other, gypsy, traveller and unknown)

Key insights from the data:

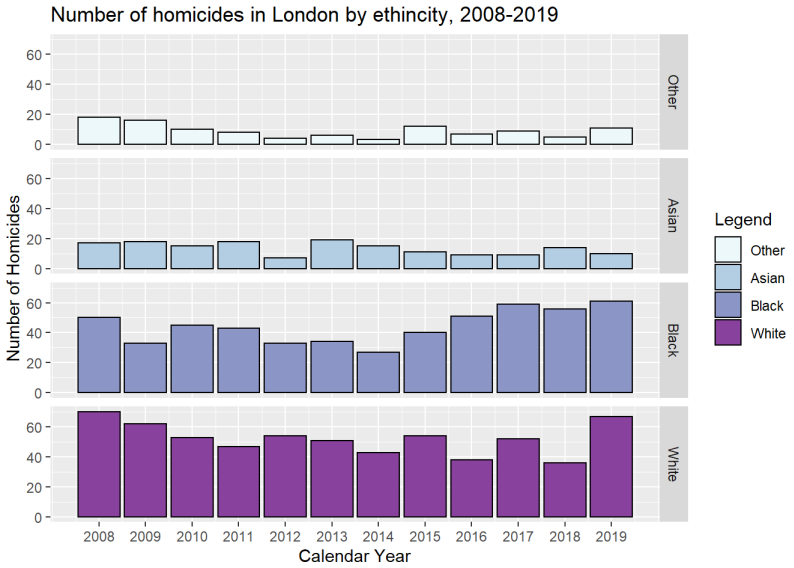
The volume of young Black homicide victims in London is worryingly high during the period observed. Between the ages of 11 and 20 more than two-thirds of homicide victims in London were Black, despite accounting for just fewer than 1 in 5 residents within this age group. White are overcrepresented as homicide victims in London when considering the 50 and over age groups.

````

```r
# histogram of homicides by ethicity of victim - original Iain Agar chart extended to include 2019 data and different levels
ggplot(mmap8,  aes(x = age, fill = ethnicity)) +
  geom_histogram(binwidth = 5, position = "dodge", colour="grey")+
  ggtitle("London Homicides by victim age and ethnicity, 2008-2020")+
  scale_x_continuous("Victim Age", breaks = c(5,15,25,35,45,55,65,75,85,95))+
  scale_y_continuous("Number of Homicides")+
  labels(fill = "Legend")+
  scale_fill_brewer(palette = "Set1")+
  theme_bw()
```
```

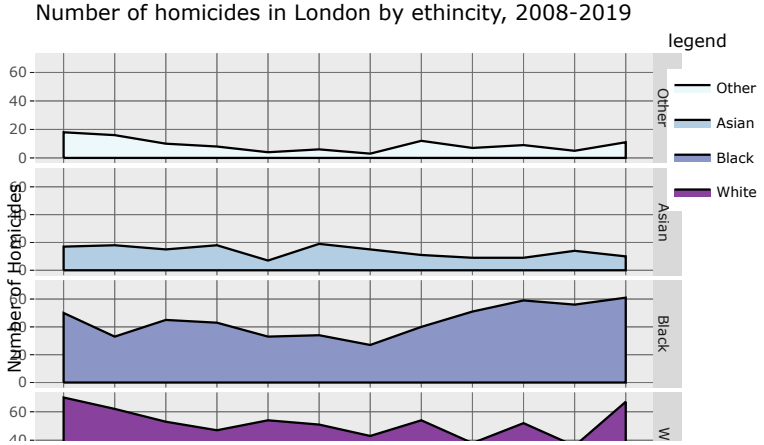



```
# bar chart showing Number of homicides in London by ethnicity, 2008-2019
ggplot(mmap8, aes(x = year, fill = ethnicity)) +
  geom_bar(stat='count', colour = "black") +
  scale_x_continuous(breaks = c(2008:2019)) +
  scale_fill_brewer(palette = "BuPu") +
  facet_grid(ethnicity~.)+
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "Legend", title = "Number of homicides in London by ethincity,
2008-2019")
```



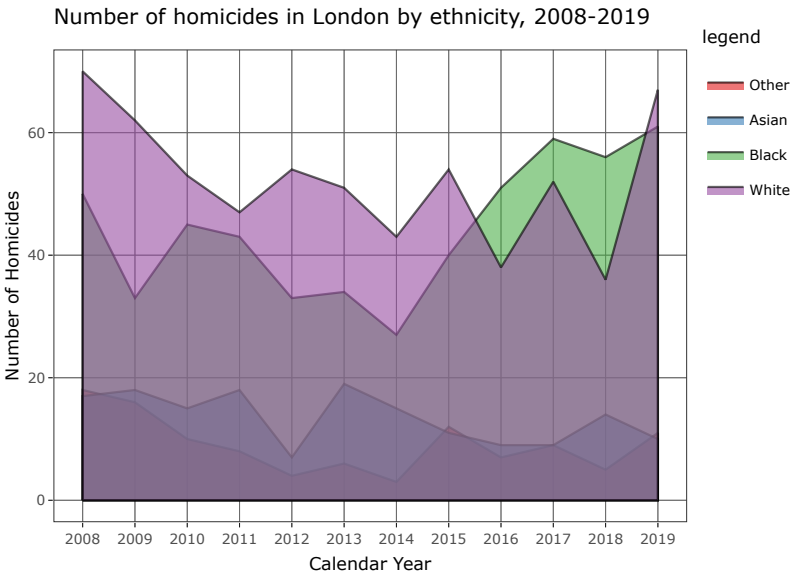

```
# density chart showing Number of homicides in London by ethnicity, 2008-2019
 p15 <- ggplot(mmap8, aes(x = year, fill = ethnicity)) +
  geom_density(stat='count') +
  scale_x_continuous(breaks = c(2008:2020)) +
  scale_fill_brewer(palette = "BuPu") +
  facet_grid(ethnicity~.)+
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "legend", title = "Number of homicides in London by ethincity,
2008-2019")

ggplotly(p15)
```

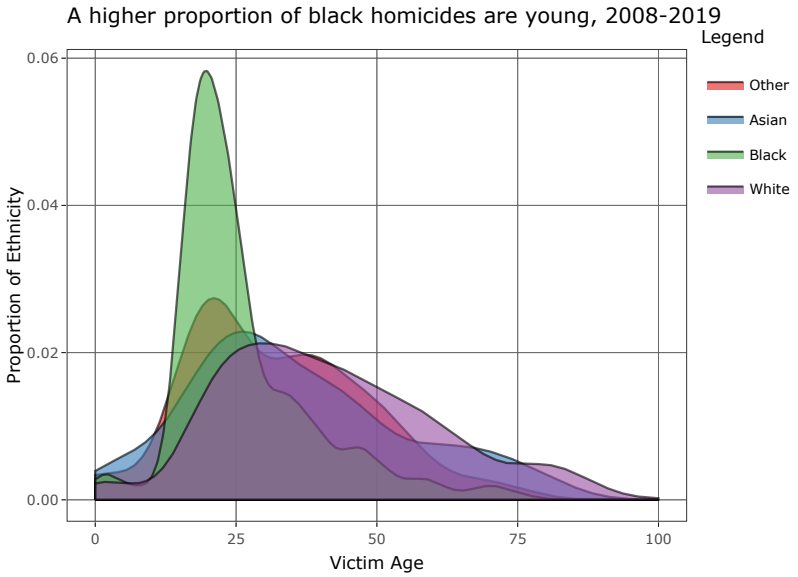



```
# density chart showing Number of homicides in London by ethnicity, 2008-2019
 p16 <- ggplot(mmap8, aes(x = year, fill = ethnicity)) +
  geom_density(stat='count', alpha = 0.6, colour = "black") +
  scale_x_continuous(breaks = c(2008:2020)) +
  scale_fill_brewer(palette = "Set1") +
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "legend", title = "Number of homicides in London by ethnicity,
2008-2019")+
   theme_bw()

ggplotly(p16)
```

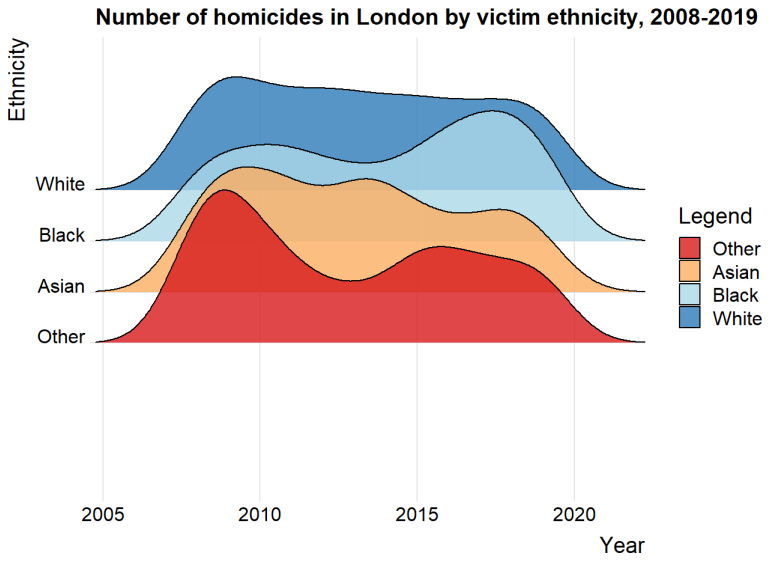



```
# density chart by age and ethnicity
p13 <- ggplot(mmap8, aes(x = age, fill = ethnicity)) +
  geom_density(alpha = 0.6)+
  ggtitle("A higher proportion of black homicides are young, 2008-2019")+
  scale_fill_brewer(palette = "Set1")+
    labs(fill = "Legend")+
  scale_x_continuous("Victim Age")+
  scale_y_continuous("Proportion of Ethnicity")+
 theme_bw()
ggplotly(p13)
```



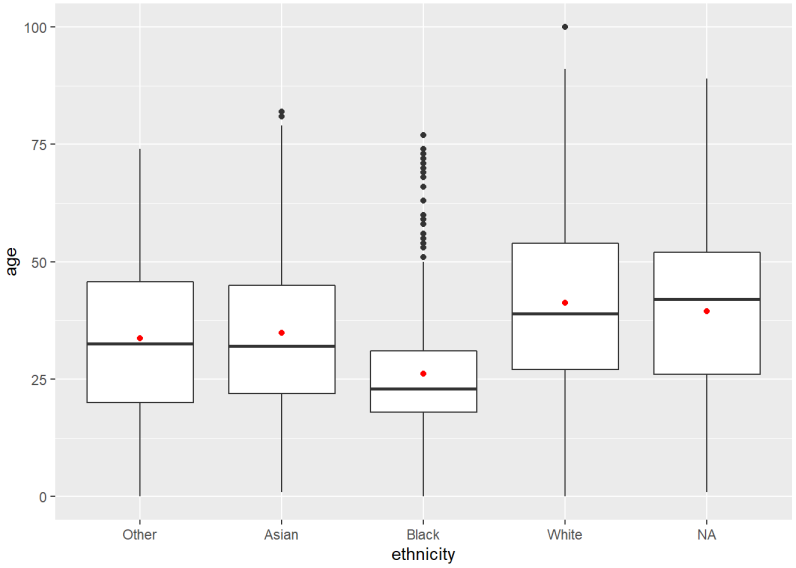
```
# ridgeline density chart for homicides by victim ethnicity
ggplot(mmap8, aes(x = year, y = ethnicity, fill = ethnicity)) +
  geom_density_ridges(scale = 3, alpha =0.8) +
   scale_fill_brewer(palette = "RdYlBu") +
  scale_y_discrete(expand = c(0, 3)) +     # will generally have to set the `expand` option
  scale_x_continuous(expand = c(0, 0)) +   # for both axes to remove unneeded padding
  coord_cartesian(clip = "off") + # to avoid clipping of the very top of the top ridgeline
  labs(x = "Year", y = "Ethnicity", fill = "Legend", title = "Number of homicides in London by victim ethnicity, 2008-2019")
+
  theme_ridges()
```

```
## Picking joint bandwidth of 1.08
```




```
# boxplot of age of victim by ethnicity of victim
qplot(x = ethnicity, y = age, data = mmap5, geom = "boxplot")+
  stat_summary(fun.y = "mean", color ="red", geom="point")
```

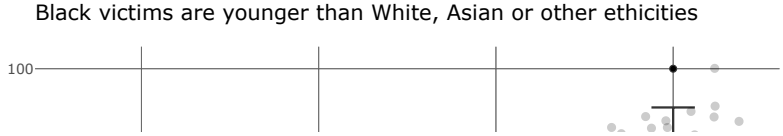
```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```

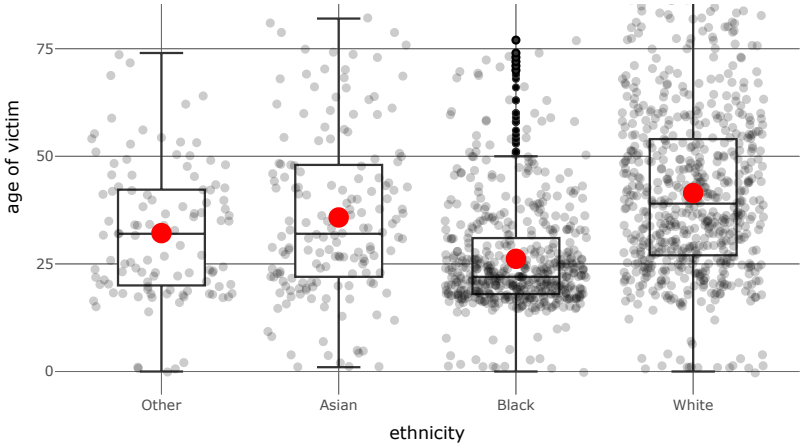



```
# jitter box plots for victims by ethincity
p14 <- ggplot(data = mmap8, aes(x = ethnicity, y = age), na.rm = TRUE)+ geom_boxplot(outlier.shape = NA) + geom_jitter(alpha
= 1/5) +
  ylab("age of victim") +
  ggtitle("Black victims are younger than White, Asian or other ethicities") +
  stat_summary(fun.y="mean", colour="red", geom="point",shape = 16, size = 4) +
  theme_minimal()
```

```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```

```
ggplotly(p14)
```

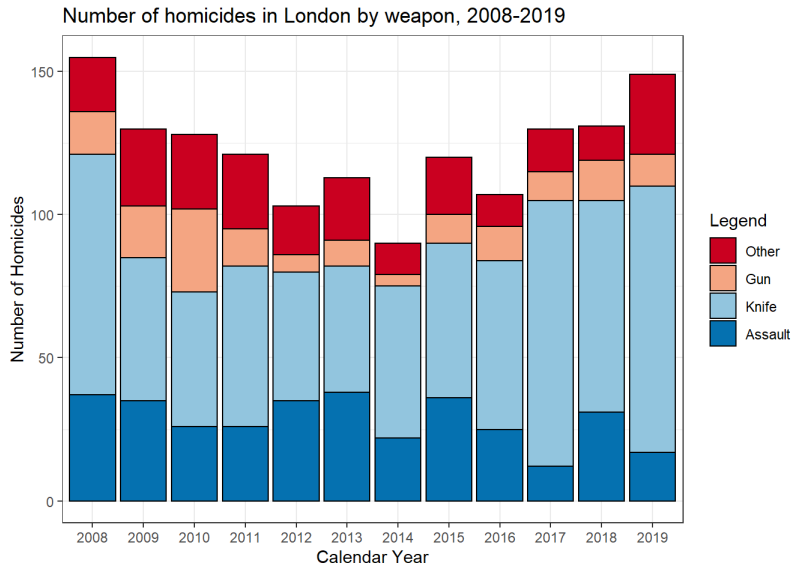




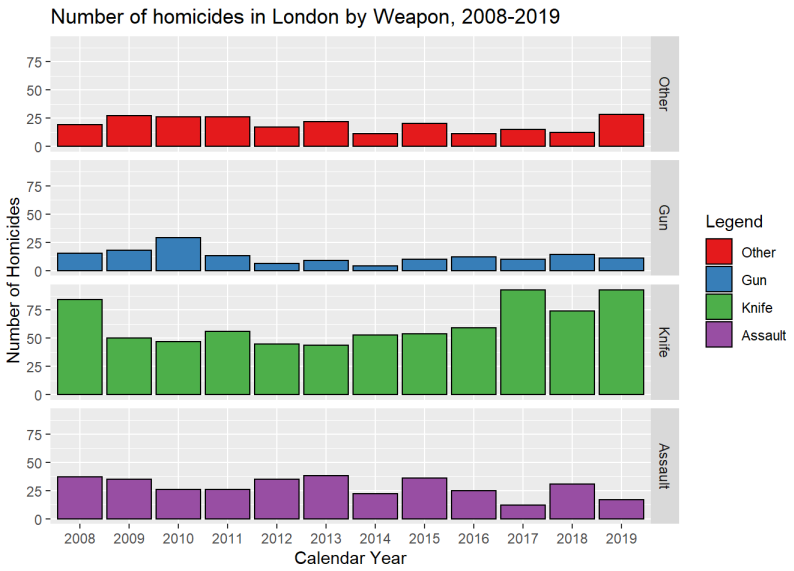

### Weapons

Since 2008 approximately 50% of all homicides were knife enabled, followed by assault with almost 25%. Firearms offences accounted for just over 10%. Firearms homicides have the lowest clearance rate, for cases where the Status is classified as solved or unsolved (exc. cases awaiting outcome) - just 55% were solved. The clearance rate for all other weapons/methods used cumulatively is over 90%.

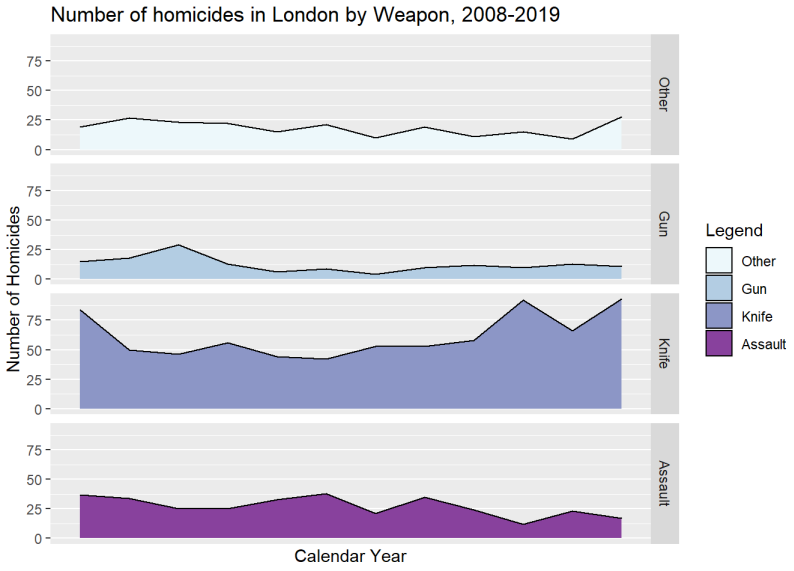
```
# stacked bar chart Number of homicides in London by weapon, 2008-2019
  ggplot(mmap7, aes(x = year, fill = weapon)) +
  geom_bar(stat='count', position = 'stack', colour = "black") +
  scale_x_discrete(breaks = c(2008:2019)) +
  scale_fill_brewer(palette = "RdBu") +
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "Legend", title = "Number of homicides in London by weapon, 20
08-2019")+
    theme_bw()
```




```
# bar chart Number of homicides in London by Weapon, 2008-2019 - facet horizontal
ggplot(mmap7, aes(x = year, fill = weapon)) +
  geom_bar(stat='count', colour = "black") +
  facet_grid(weapon~.)+
  scale_x_discrete(breaks = c(2008:2019)) +
  scale_fill_brewer(palette = "Set1") +
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "Legend", title = "Number of homicides in London by Weapon, 20
08-2019")
```



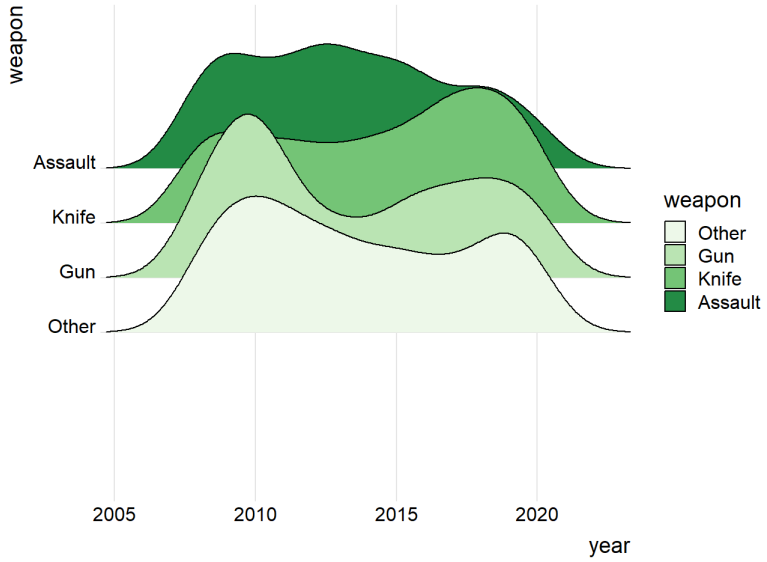

```
# Density chart Number of homicides in London by Weapon, 2008-2019
  ggplot(mmap8, aes(x = year, fill = weapon)) +
  geom_density(stat='count', colour = "black") +
  facet_grid(weapon~.)+
  scale_x_discrete(breaks = c(2008:2020)) +
  scale_fill_brewer(palette = "BuPu") +
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "Legend", title = "Number of homicides in London by Weapon, 20
08-2019")
```



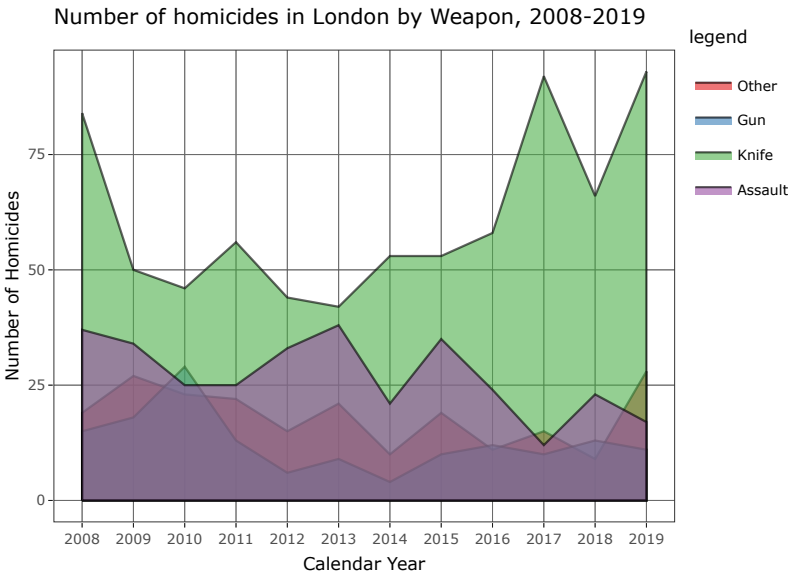

```
# ridgeline density chart for weapon, 2008-2020
ggplot(mmap5, aes(x = year, y = weapon, fill = weapon)) +
  geom_density_ridges(scale = 3) +
   scale_fill_brewer(palette = "Set5") +
  scale_y_discrete(expand = c(0, 3)) +     # will generally have to set the `expand` option
  scale_x_continuous(expand = c(0, 0)) +   # for both axes to remove unneeded padding
  coord_cartesian(clip = "off") + # to avoid clipping of the very top of the top ridgeline
  theme_ridges()
```

```
## Warning in pal_name(palette, type): Unknown palette Set5
```

```
## Picking joint bandwidth of 1.1
```



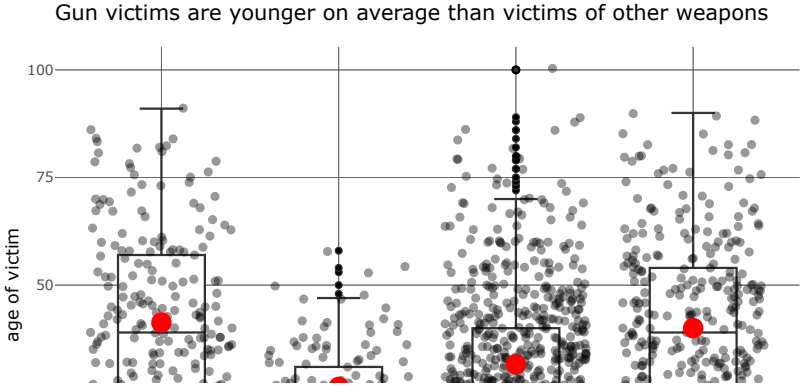

```
# density chart showing Number of homicides in London by ethnicity, 2008-2019
 p17 <- ggplot(mmap8, aes(x = year, fill = weapon)) +
  geom_density(stat='count', alpha = 0.6, colour = "black") +
  scale_x_continuous(breaks = c(2008:2020)) +
  scale_fill_brewer(palette = "Set1") +
  labs(x = "Calendar Year", y = "Number of Homicides", fill = "legend", title = "Number of homicides in London by Weapon, 20
08-2019")+
   theme_bw()
ggplotly(p17)
```




```
# jitter box plots for victims by type of weapon
p19 <- ggplot (data = mmap8, aes(x = weapon, y = age))+ geom_boxplot() + geom_jitter(alpha = 2/5) +
  ylab("age of victim") +
  ggtitle("Gun victims are younger on average than victims of other weapons") +
  stat_summary(fun.y="mean", colour="red", geom="point",shape = 16, size = 4) +
  theme_minimal()
```

```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```

```
ggplotly(p19)
```




# Rates and trends

- Those aged 17-19 have the highest risk of homicide at more than 5 per 100,000 population
- Those aged 20-24 have the next highest risk of homicide at more than 4 per 100,000 population
- Those aged 12 and under and 35 and over experience rates of homicide lower than the London average of 1.5 (during time frame observed)
- Southwark has the highest total volume, 76, and highest average rate over the previous 10 years, 2.42 - this includes victims of the London Bridge attack in June 2017 during which 8 people were killed.
- Seven other boroughs have averaged rates in excess of 2 per 100,000, those being Haringey, Islington, Lambeth, Hackney, Greenwich, Westminster and Newham
- The two highest quarters for homicide since 2008 were in 2017 (48 homicides in Q2, Apr-Jun) and 2018 (47 homicides in Q1, Jan-Mar). The next five highest quarters were all prior to Q1 of 2011.

## Homicide Rate by age group over the 11 year period

The 17-19 age group has an average annual rate of homicide of 5.03 per 100,000 followed by 4.19 for those aged 20-24. This compares to an average of 1.5 for all residents. Those aged 12 and under and 35 and over experience rates lower than average for London.

```
# create new data frame to show ageGroup data by year
ageyear <- mmap7 %>%
  select(ageGroup, year) %>%
  group_by(ageGroup, year) %>%
  tally() %>%
  spread(ageGroup, n)


# calculate total homicides per ageGroup across 2008 to 2019
homicidesA <- sum(ageyear$`A. Child 0-6`, na.rm= TRUE)
homicidesB <- sum(ageyear$`B. Child 7-12`, na.rm= TRUE)
homicidesC <- sum(ageyear$`C. Teen 13-16`, na.rm= TRUE)
homicidesD <- sum(ageyear$`D. Teen 17-19`, na.rm= TRUE)
homicidesE <- sum(ageyear$`E. Adult 20-24`, na.rm= TRUE)
homicidesF <- sum(ageyear$`F. Adult 25-34`, na.rm= TRUE)
homicidesG <- sum(ageyear$`G. Adult 35-44`, na.rm= TRUE)
homicidesH <- sum(ageyear$`H. Adult 45-54`, na.rm= TRUE)
homicidesI <- sum(ageyear$`I. Adult 55-64`, na.rm= TRUE)
homicidesJ <- sum(ageyear$`J. Adult 65 over`, na.rm= TRUE)

# consolidate total homicides per ageGroup across 2008 to 2019 into a new vector homicides_ageGroup
homicides_ageGroup <- c(homicidesA, homicidesB, homicidesC, homicidesD, homicidesE, homicidesF, homicidesG, homicidesH, homi
cidesI, homicidesJ)

# create vector agegp with unique values for the variable "ageGroup" in mmap5
agegp <- unique(mmap5$ageGroup)

# refactor to order the values for the variable "ageGroup" from youngest to oldest
agegp <- factor(c("A. Child 0-6", "B. Child 7-12", "C. Teen 13-16", "D. Teen 17-19", "E. Adult 20-24", "F. Adult 25-34","G.
 Adult 35-44","H. Adult 45-54","I. Adult 55-64","J. Adult 65 over"))

# create vector to calculate average homicides per year over 11 years from 2008 to 2019
homicides_ageGroup_avg <- homicides_ageGroup/11

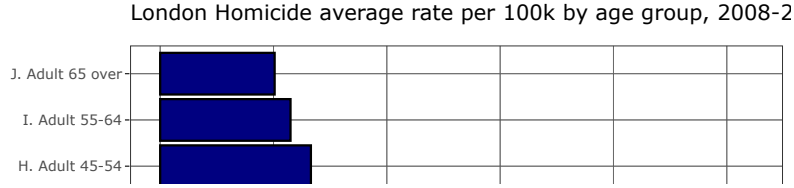
# create population vector by age group - Source: ONS small area population estimates mid-2016 (latest ONS survey for UK)
popn <- c(872040, 659384, 374989, 278134, 557975, 1654605, 1408946, 1151408, 828359, 1039161)

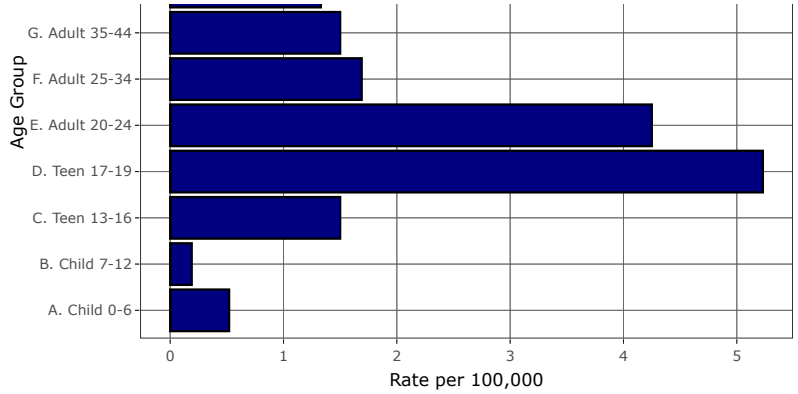
# create homicide-rate2 vector
homicide_rate2  <- homicides_ageGroup_avg / popn * 100000


# Create a data frame from vectors
mmap11 <- data.frame(agegp, round(homicide_rate2,2))

colnames(mmap11) <- c("Age_Group", "Homicide_Rate")


p20 <- ggplot(mmap11, aes(x = Age_Group, y = Homicide_Rate)) +
  geom_bar(stat = 'identity', fill = "navyblue", colour = "black") +
  ggtitle("London Homicide average rate per 100k by age group, 2008-2019") +
  xlab("Age Group")+
  ylab("Rate per 100,000")+
  coord_flip()+
  theme_bw()
ggplotly(p20)
```



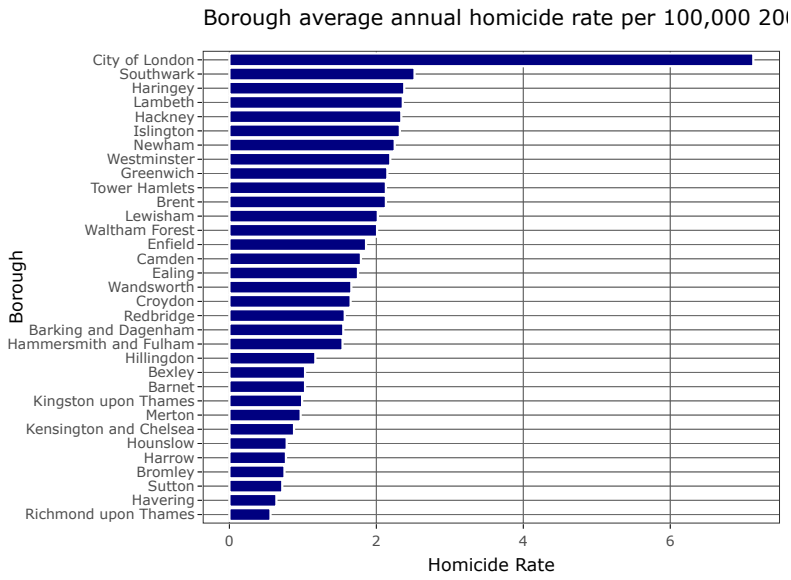



## Rate by borough

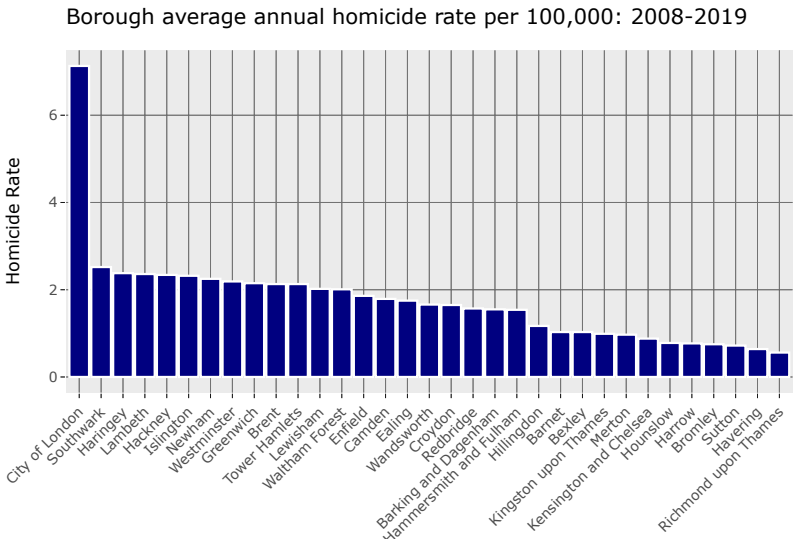
The London Borough of Southwark recorded the highest total volume (76) and highest average rate (2.42) of homicide - this includes the London Bridge attack in June 2017, during which 8 people were killed.

Boroughs making up inner London (by ONS Statistical Definition) occupy 9 of the highest 10 for rate per 100,000 (Southwark, Haringey, Islington, Lambeth, Hackney, Westminster, Newham, Lewisham and Tower Hamlets), the other being Greenwich. By contrast, just two inner London boroughs have a rate lower than the average (Hammersmith & Fulham, Kensington & Chelsea).

```
# homicides by 100,000 population in boroughs (averaged over the 11 years of data) - horizontal
p30 <- ggplot(mmap10, aes(x= reorder(borough, homicide_rate), y=homicide_rate)) +
geom_bar(stat="identity", fill="navyblue", color = "white")+
  coord_flip()+
  labs(x = "Borough", y = "Homicide Rate", title = "Borough average annual homicide rate per 100,000 2008-2019")+
  theme_bw()
ggplotly(p30)
```




```
# homicides by 100,000 population in boroughs (averaged over the 11 years of data) - vertical
p31 <- ggplot(mmap10, aes(x= reorder(borough, -homicide_rate), y=homicide_rate)) +
geom_bar(stat="identity", fill="navyblue", color = "white")+
  labs(x = "Borough", y = "Homicide Rate", title = "Borough average annual homicide rate per 100,000: 2008-2019")+
  theme(axis.text.x = element_text(angle = 45, hjust=1))
ggplotly(p31)
```



Borough

## Status This section explores patterns around which homicides are getting solved.

Over the period 2008 to 2020 there were clear trends about what types of homicides are getting solved. These include:

- Only 44% of homicides committed by guns are solved, compared to 77% of assaults
- Only 66% of homicides where the victim is black are solved compared to 80% of white vicitms and 77% of Asian victims
- 89% of homicides in Havering are solved compared to 59% in Bromley

Research on borough socio economic profiles:

About Havering. https://www.haveringdata.net/wp-content/uploads/2018/04/Published-version-201718_Havering-Demographic-Profile-v3-4-1.pdf (https://www.haveringdata.net/wp-content/uploads/2018/04/Published-version-201718_Havering-Demographic-Profile-v3-4-1.pdf)

Havering is a relatively affluent local authority but there are pockets of deprivation to the north (Gooshays and Heaton wards) and south (South Hornchurch) of the borough. It has the oldest population in London with a median age of approximately 40 years old.Havering is one of the most ethnically homogenous places in London, with 83% of its residents recorded as White British, higher than both London and England. About 90% of the borough population were born in the United Kingdom. It is projected that the Black African population will increase from 4.1% in 2017 to 5.3% of the Havering population in 2032.

About Bromley: It's not regarded as an especially dangerous area and has relatively low levels of diversity. It is not very affluent. It has a couple of counties including Crystal Place and Penge & Cator that are regarded as quite dangerous.

```
# Trends in Status

## create tables showing probability of being solved given weapon, ethnicity and borough

# by weapon type
status <- table(mmap5$weapon, mmap5$status, dnn = c("weapon", "status"))
status
```

```
##          status
## weapon    Awaiting Trial Solved Unsolved
##   Other               28    191       31
##   Gun                 28     72       65
##   Knife              140    595       76
##   Assault             52    271       35
```

```
statusProp <- prop.table(status, 1)
statusProp
```

```
##          status
## weapon    Awaiting Trial     Solved   Unsolved
##   Other       0.11200000 0.76400000 0.12400000
##   Gun         0.16969697 0.43636364 0.39393939
##   Knife       0.17262639 0.73366215 0.09371147
##   Assault     0.14525140 0.75698324 0.09776536
```

```
# save as data frame
weaponStatus <- data.frame(statusProp)

# filter to just include probability of being solved
weaponStatus1 <- weaponStatus %>% filter(status == "Solved")

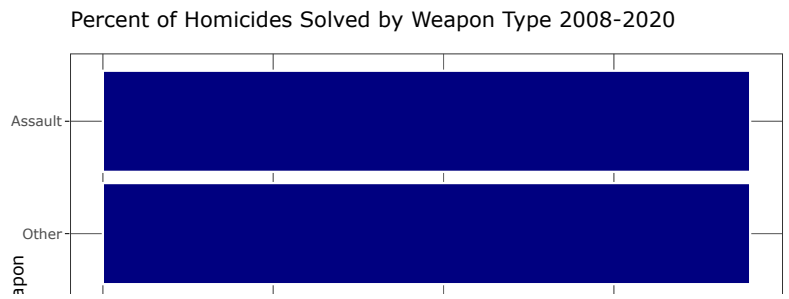
# change to percent
weaponStatus1$percent <- round(weaponStatus1$Freq,2)*100

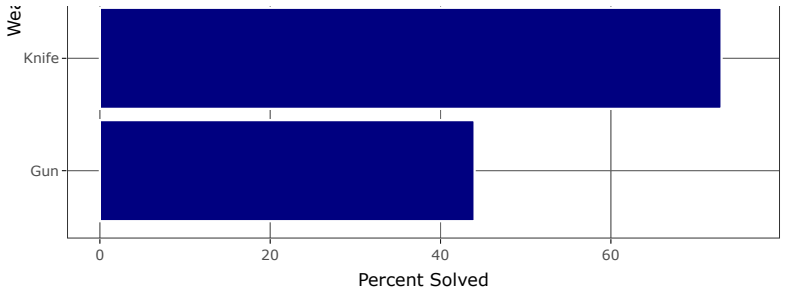
mmap13 <- weaponStatus1 %>% arrange(weaponStatus1$percent)

mmap13$percent <- as.integer(mmap13$percent)


# bar chart Percent of Homicides Solved by Weapon Type 2008-2020
p22 <- ggplot(mmap13, aes(x= reorder(weapon, percent), y=percent))+
geom_bar(stat="identity", fill="navyblue", color = "white")+
geom_label(size = 4, aes(label = percent, hjust = 0.2))+
  coord_flip()+
  labs(x = "Weapon", y = "Percent Solved", title = "Percent of Homicides Solved by Weapon Type 2008-2020")+
  theme_bw()
ggplotly(p22)
```

```
## Warning in geom2trace.default(dots[[1L]][[1L]], dots[[2L]][[1L]], dots[[3L]][[1L]]): geom_GeomLabel() has yet to be imple
mented in plotly.
##   If you'd like to see this geom implemented,
##   Please open an issue with your example code at
##   https://github.com/ropensci/plotly/issues
```






```
# by ethnicity

levels(mmap8$ethnicity)
```

```
## [1] "Other" "Asian" "Black" "White"
```

```
status2 <- table(mmap8$ethnicity, mmap8$status, dnn = c("ethnicity", "status"))
status2
```

```
##          status
## ethnicity Awaiting Trial Solved Unsolved
##     Other              9     81       19
##     Asian             16    129       17
##     Black             78    366       88
##     White             54    514       59
```

```
status2Prop <- round(prop.table(status2, 1),2)
status2Prop
```

```
##          status
## ethnicity Awaiting Trial Solved Unsolved
##     Other           0.08   0.74     0.17
##     Asian           0.10   0.80     0.10
##     Black           0.15   0.69     0.17
##     White           0.09   0.82     0.09
```

```
# save as data frame
ethnicityStatus <- data.frame(status2Prop)

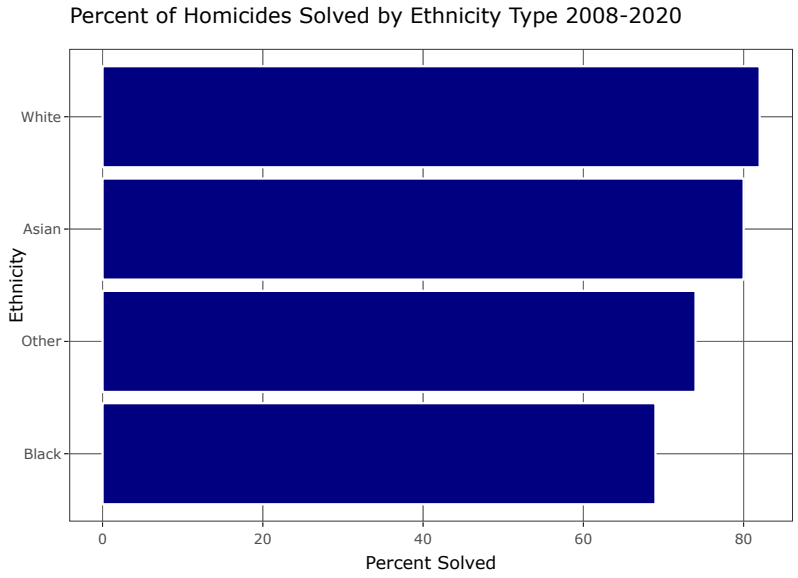

# filter to just include probability of being solved

ethnicityStatus1 <- ethnicityStatus %>% filter(status == "Solved")

# change to percent
ethnicityStatus1$percent <- round(ethnicityStatus1$Freq,2)*100

mmap14 <- ethnicityStatus1 %>% arrange(ethnicityStatus1$percent)

# bar chart Percent of Homicides Solved by Weapon Type 2008-2020
p23 <- ggplot(mmap14, aes(x= reorder(ethnicity, percent), y=percent))+
geom_bar(stat="identity", fill="navyblue", color = "white")+
  coord_flip()+
  labs(x = "Ethnicity", y = "Percent Solved", title = "Percent of Homicides Solved by Ethnicity Type 2008-2020")+
  theme_bw()
ggplotly(p23)
```



```
# by ageGroup

levels(mmap8$ageGroup)
```

```
##  [1] "A. Child 0-6"     "B. Child 7-12"    "C. Teen 13-16"    "D. Teen 17-19"
##  [5] "E. Adult 20-24"   "F. Adult 25-34"   "G. Adult 35-44"   "H. Adult 45-54"
##  [9] "I. Adult 55-64"   "J. Adult 65 over"
```

```
status4 <- table(mmap8$ageGroup, mmap8$status, dnn = c("ageGroup", "status"))
status4
```

```
##                   status
## ageGroup           Awaiting Trial Solved Unsolved
##   A. Child 0-6                 3     39        3
##   B. Child 7-12                1     11        0
##   C. Teen 13-16                5     50        7
##   D. Teen 17-19               32    108       17
##   E. Adult 20-24              29    174       57
##   F. Adult 25-34              30    221       48
##   G. Adult 35-44              18    185       21
##   H. Adult 45-54              17    133       11
##   I. Adult 55-64              11     83        8
##   J. Adult 65 over            11     86       11
```

```
status4Prop <- round(prop.table(status4, 1),2)
status4Prop
```

```
##                   status
## ageGroup           Awaiting Trial Solved Unsolved
##   A. Child 0-6             0.07   0.87     0.07
##   B. Child 7-12            0.08   0.92     0.00
##   C. Teen 13-16            0.08   0.81     0.11
##   D. Teen 17-19            0.20   0.69     0.11
##   E. Adult 20-24           0.11   0.67     0.22
##   F. Adult 25-34           0.10   0.74     0.16
##   G. Adult 35-44           0.08   0.83     0.09
##   H. Adult 45-54           0.11   0.83     0.07
##   I. Adult 55-64           0.11   0.81     0.08
##   J. Adult 65 over         0.10   0.80     0.10
```

```
# save as data frame
AgeGroupStatus <- data.frame(status4Prop)


# filter to just include probability of being solved

AgeGroupStatus1 <- AgeGroupStatus %>% filter(status == "Solved")

# change to percent
AgeGroupStatus$percent1 <- round(AgeGroupStatus1$Freq,2)*100


# bar chart Percent of Homicides Solved by Weapon Type 2008-2020
p24 <- ggplot(AgeGroupStatus1, aes(x= reorder(ageGroup, percent), y=percent))+
geom_bar(stat="identity", fill="navyblue", color = "white")+
  coord_flip()+
  labs(x = "AgeGroup", y = "Percent Solved", title = "Percent of Homicides Solved by Age Group 2008-2020")+
  theme_bw()
ggplotly(p24)
```

```
## Error in tapply(X = X, INDEX = x, FUN = FUN, ...): object 'percent' not found
```

```
# by location
status3 <- table(mmap5$borough, mmap5$status, dnn = c("borough", "status"))
status3
```

```
##                         status
## borough                  Awaiting Trial Solved Unsolved
##   Barking and Dagenham               10     21        5
##   Barnet                              7     33        4
##   Bexley                              2     24        2
##   Brent                              11     54       12
##   Bromley                             7     16        4
##   Camden                              9     33        8
##   City of London                      1      3        2
##   Croydon                             9     49       12
##   Ealing                             11     47        8
##   Enfield                            14     46        8
##   Greenwich                          13     47        7
##   Hackney                            11     50       10
##   Hammersmith and Fulham              4     21        6
##   Haringey                           11     53        7
##   Harrow                              2     18        1
##   Havering                            2     15        1
##   Hillingdon                          5     29        5
##   Hounslow                            2     15        6
##   Islington                          10     42        8
##   Kensington and Chelsea              3     11        1
##   Kingston upon Thames                3     13        3
##   Lambeth                            14     62        8
##   Lewisham                            7     48       12
##   Merton                              5     17        0
##   Newham                             11     63       12
##   Redbridge                          11     35        6
##   Richmond upon Thames                3      9        0
##   Southwark                          15     61       11
##   Sutton                              2     13        1
##   Tower Hamlets                       6     58        8
##   Waltham Forest                     15     38        8
##   Wandsworth                          7     39       13
##   Westminster                         5     46        8
```

```
status3Prop <- prop.table(status3, 1)
status3Prop
```

```
##                         status
## borough                  Awaiting Trial     Solved   Unsolved
##   Barking and Dagenham       0.27777778 0.58333333 0.13888889
##   Barnet                     0.15909091 0.75000000 0.09090909
##   Bexley                     0.07142857 0.85714286 0.07142857
##   Brent                      0.14285714 0.70129870 0.15584416
##   Bromley                    0.25925926 0.59259259 0.14814815
##   Camden                     0.18000000 0.66000000 0.16000000
##   City of London             0.16666667 0.50000000 0.33333333
##   Croydon                    0.12857143 0.70000000 0.17142857
##   Ealing                     0.16666667 0.71212121 0.12121212
##   Enfield                    0.20588235 0.67647059 0.11764706
##   Greenwich                  0.19402985 0.70149254 0.10447761
##   Hackney                    0.15492958 0.70422535 0.14084507
##   Hammersmith and Fulham     0.12903226 0.67741935 0.19354839
##   Haringey                   0.15492958 0.74647887 0.09859155
##   Harrow                     0.09523810 0.85714286 0.04761905
##   Havering                   0.11111111 0.83333333 0.05555556
##   Hillingdon                 0.12820513 0.74358974 0.12820513
##   Hounslow                   0.08695652 0.65217391 0.26086957
##   Islington                  0.16666667 0.70000000 0.13333333
##   Kensington and Chelsea     0.20000000 0.73333333 0.06666667
##   Kingston upon Thames       0.15789474 0.68421053 0.15789474
##   Lambeth                    0.16666667 0.73809524 0.09523810
##   Lewisham                   0.10447761 0.71641791 0.17910448
##   Merton                     0.22727273 0.77272727 0.00000000
##   Newham                     0.12790698 0.73255814 0.13953488
##   Redbridge                  0.21153846 0.67307692 0.11538462
##   Richmond upon Thames       0.25000000 0.75000000 0.00000000
##   Southwark                  0.17241379 0.70114943 0.12643678
##   Sutton                     0.12500000 0.81250000 0.06250000
##   Tower Hamlets              0.08333333 0.80555556 0.11111111
##   Waltham Forest             0.24590164 0.62295082 0.13114754
##   Wandsworth                 0.11864407 0.66101695 0.22033898
##   Westminster                0.08474576 0.77966102 0.13559322
```

```
# save as data frame
boroughStatus <- data.frame(status3Prop)

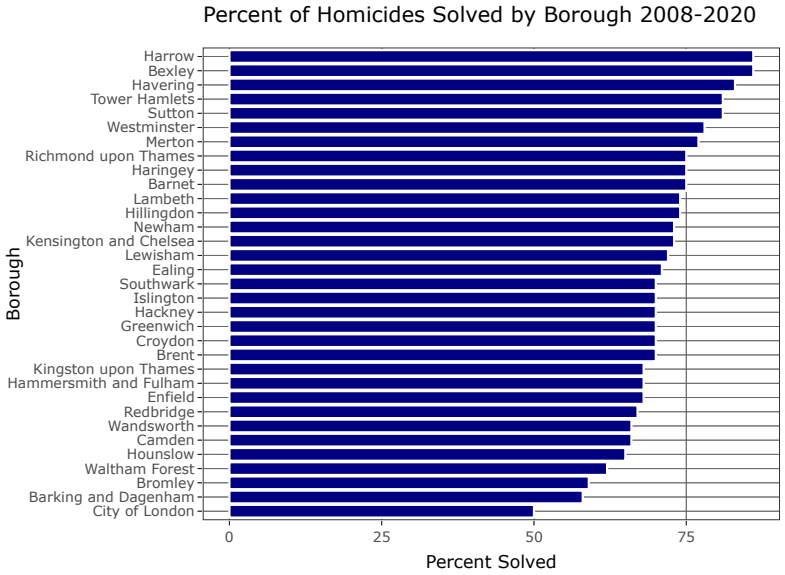
# filter to just include probability of being solved

boroughStatus1 <- boroughStatus %>% filter(status == "Solved")

# change to percent
boroughStatus1$percent <- round(boroughStatus1$Freq,2)*100

View(boroughStatus1)

# bar chart Percent of Homicides Solved by Weapon Type 2008-2020
p25 <- ggplot(boroughStatus1, aes(x= reorder(borough, percent), y=percent))+
geom_bar(stat="identity", fill="navyblue", color = "white")+
  coord_flip()+
  labs(x = "Borough", y = "Percent Solved", title = "Percent of Homicides Solved by Borough 2008-2020")+
  theme_bw()
ggplotly(p25)
```




# Geography

## choropleth map

```
# Install ggplot2 mapping packages

library(rgeos)
```

```
## Warning: package 'rgeos' was built under R version 4.0.3
```

```
## Loading required package: sp
```

```
## Warning: package 'sp' was built under R version 4.0.3
```

```
## rgeos version: 0.5-5, (SVN revision 640)
##  GEOS runtime version: 3.8.0-CAPI-1.13.1
##  Linking to sp version: 1.4-4
##  Polygon checking: TRUE
```

```
library(ggmap)
```

```
## Warning: package 'ggmap' was built under R version 4.0.3
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
##
## Attaching package: 'ggmap'
```

```
## The following object is masked from 'package:cowplot':
##
##     theme_nothing
```

```
## The following object is masked from 'package:plotly':
##
##     wind
```

```
library(maptools)
```

```
## Warning: package 'maptools' was built under R version 4.0.3
```

```
## Checking rgeos availability: TRUE
```

```
library(broom)
```

```
## Warning: package 'broom' was built under R version 4.0.3
```

```
library(mapproj)
```

```
## Warning: package 'mapproj' was built under R version 4.0.3
```

```
## Loading required package: maps
```

```
## Warning: package 'maps' was built under R version 4.0.3
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
library(rgdal)
```

```
## Warning: package 'rgdal' was built under R version 4.0.3
```

```
## rgdal: version: 1.5-18, (SVN revision 1082)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.0.4, released 2020/01/28
## Path to GDAL shared files: C:/Users/david/OneDrive/Personal and Family/R/win-library/4.0/rgdal/gdal
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 6.3.1, February 10th, 2020, [PJ_VERSION: 631]
## Path to PROJ shared files: C:/Users/david/OneDrive/Personal and Family/R/win-library/4.0/rgdal/proj
## Linking to sp version:1.4-4
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
## use options("rgdal_show_exportToProj4_warnings"="none") before loading rgdal.
```

```
# create Leaflet interactive cloropeth chart for Homicide Rate

# read shp file for just London boroughs
London_Ward2 <- readShapeSpatial("London_Borough_Excluding_MHW.shp")
```

```
## Warning: readShapeSpatial is deprecated; use rgdal::readOGR or sf::st_read
```

```
## Warning: readShapePoly is deprecated; use rgdal::readOGR or sf::st_read
```

```
# convert coordinates into lat long degrees
proj4string(London_Ward2) <- CRS("+init=epsg:27700")
```

```
## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj =
## prefer_proj): Discarded datum OSGB_1936 in CRS definition
```

```
London_Ward2.wgs84 <- spTransform(London_Ward2, CRS("+init=epsg:4326"))


library(leaflet)

# For Leaflet choropleth maps, we need to use a SpatialPolygonDataFrame

class(London_Ward2.wgs84)
```

```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
names(London_Ward2.wgs84)
```

```
## [1] "NAME"       "GSS_CODE"   "HECTARES"   "NONLD_AREA" "ONS_INNER"
## [6] "SUB_2009"   "SUB_2006"
```

```
# Rename the borough variable in mmap10 to "NAME" to match the "SpatialPolygonsDataFrame" file London_Ward2.wgs84

names(mmap10)[names(mmap10) == "borough"] <- "NAME"

str(mmap10)
```

```
## 'data.frame':    33 obs. of  2 variables:
##  $ NAME         : Factor w/ 33 levels "Barking and Dagenham",..: 7 28 14 22 12 19 25 33 11 4 ...
##  $ homicide_rate: num  7.13 2.52 2.38 2.36 2.34 2.32 2.25 2.19 2.15 2.13 ...
```

```
mmap10 <- mmap10 %>% mutate(NAME = fct_recode(NAME,

                              "City of London" = "City and County of the City of London",

                            "Westminster" = "City of Westminster"))
```

```
## Warning: Problem with `mutate()` input `NAME`.
## i Unknown levels in `f`: City and County of the City of London, City of Westminster
## i Input `NAME` is `fct_recode(...)`.
```

```
## Warning: Unknown levels in `f`: City and County of the City of London, City of
## Westminster
```

```
levels(mmap10$NAME)
```

```
##  [1] "Barking and Dagenham"   "Barnet"                 "Bexley"
##  [4] "Brent"                  "Bromley"                "Camden"
##  [7] "City of London"         "Croydon"                "Ealing"
## [10] "Enfield"                "Greenwich"              "Hackney"
## [13] "Hammersmith and Fulham" "Haringey"               "Harrow"
## [16] "Havering"               "Hillingdon"             "Hounslow"
## [19] "Islington"              "Kensington and Chelsea" "Kingston upon Thames"
## [22] "Lambeth"                "Lewisham"               "Merton"
## [25] "Newham"                 "Redbridge"              "Richmond upon Thames"
## [28] "Southwark"              "Sutton"                 "Tower Hamlets"
## [31] "Waltham Forest"         "Wandsworth"             "Westminster"
```

```
# merge mmap data into London_Ward.wgs84 SpatialPolygonsDataFrame using "DISTRICT"
merge.mmap10 <- sp::merge(London_Ward2.wgs84, mmap10, by = "NAME", duplicateGeoms = TRUE)

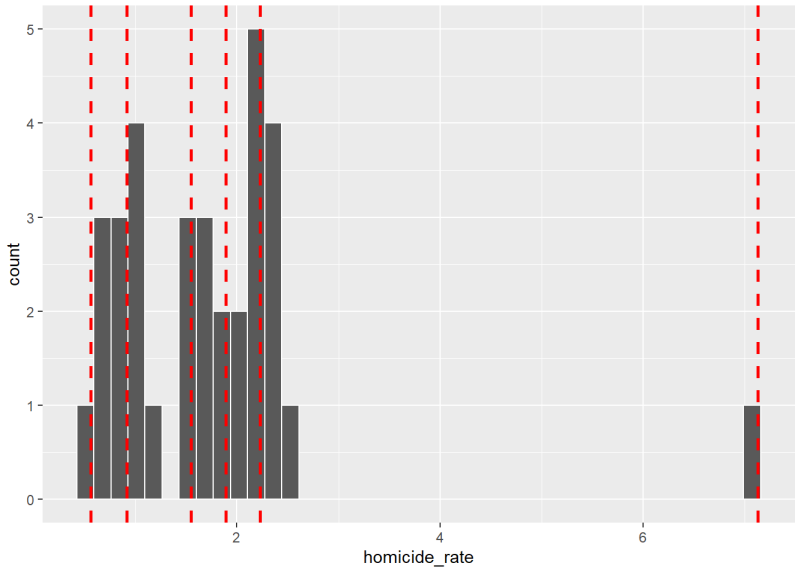

# create a discrete colour scale based on quantiles for five levels

bins <- quantile(
  mmap10$homicide_rate,
  probs = seq(0,1,.2), names = FALSE, na.rm = TRUE)
bins
```

```
## [1] 0.560 0.916 1.548 1.890 2.226 7.130
```

```
# bins now contains five sequential colour levels so that 20 per cent of the data falls within each bin. The following histo
gram visualises the breaks used to create the scale.

ggplot(data = mmap10,
       aes(x = homicide_rate)) +
  geom_histogram(colour = "white", bins = 40) +
  geom_vline(
    xintercept = quantile(
      mmap10$homicide_rate,
      probs = seq(0,1,0.2), na.rm = TRUE),
    colour = "red", lwd = 1, lty = 2)
```



```
# Bins can be used to create a colour scale, named pal, using the colorBin() function, which maps the bins to a palette. Sel
ect the YlOrRd palette from the ColourBrewer package

pal <- colorBin(
  "YlOrRd",
  domain = mmap10$homicide_rate,
  bins = bins
  )

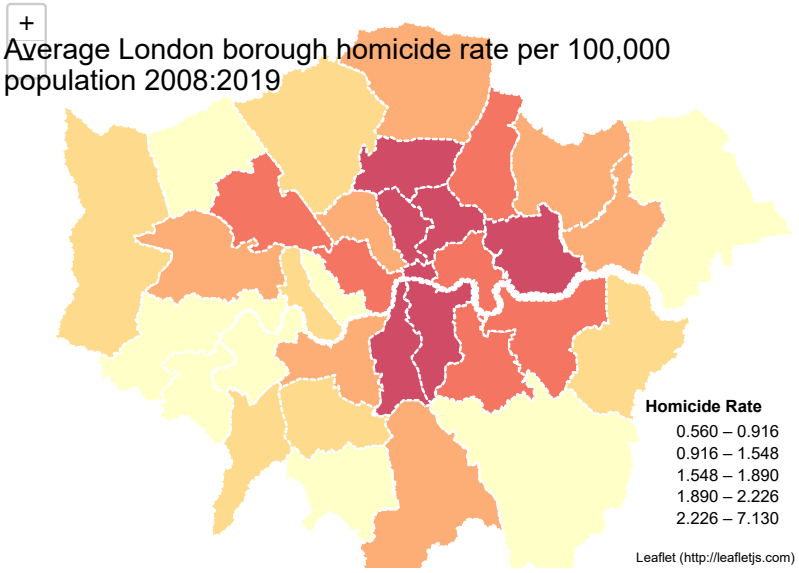
# add the colour scale to the choropleth map.
labels <- sprintf(
  "%s
%g homicide_rate",
  merge.mmap10$NAME,
  merge.mmap10$homicide_rate
) %>% lapply(htmltools::HTML)

library(htmlwidgets)
```

```
## Warning: package 'htmlwidgets' was built under R version 4.0.3
```

```
library(htmltools)

title <- tags$div(
   HTML('<h3>Average London borough homicide rate per 100,000 population 2008:2019</h3>')
 )
p3 <- leaflet(merge.mmap10) %>%
  setView(lng = -0.118092 , lat = 51.509865, zoom = 10)
p3 %>% addPolygons(
  fillColor = ~pal(homicide_rate),
  weight = 2,
  opacity = 1,
  color = "white",
  dashArray = "3",
  fillOpacity = 0.7,
  highlight = highlightOptions(
    weight = 5,
    color = "#666",
    dashArray = "",
    fillOpacity = 0.7,
    bringToFront = TRUE),
  label = labels,
  labelOptions = labelOptions(
    style = list("font-weight" = "normal", padding = "3px 8px"),
    textsize = "15px",
    direction = "auto")) %>%
  addLegend(pal = pal,
            values = ~homicide_rate,
            opacity = 0.7, title = "Homicide Rate",
  position = "bottomright") %>%
  addControl(title, position = "topright")
```




# Geography

## Leaflet Maps

This section is a sample of maps using the Leaflet package.

## Layered point map by weapon

The layered point map simple shows the distribution of all homicide offenses in the data for 2008-2020 and provides coloured circles based on the type of weapon used in the homicide. This visualisation is an extension of the original visualisation created by Iain Agar in 2018 but includes new data for 2019 and 2020

```
pal <- colorFactor(palette = c("red", "blue", "dark green", "black"),
                   levels = c("Knife", "Gun", "Assault", "Other")) # Creating a 4 colour palette

knife <- mmap5 %>%
  filter(weapon == "Knife") # Grouping layers - knife

gun <- mmap5 %>%
  filter(weapon == "Gun") # Grouping layers - gun

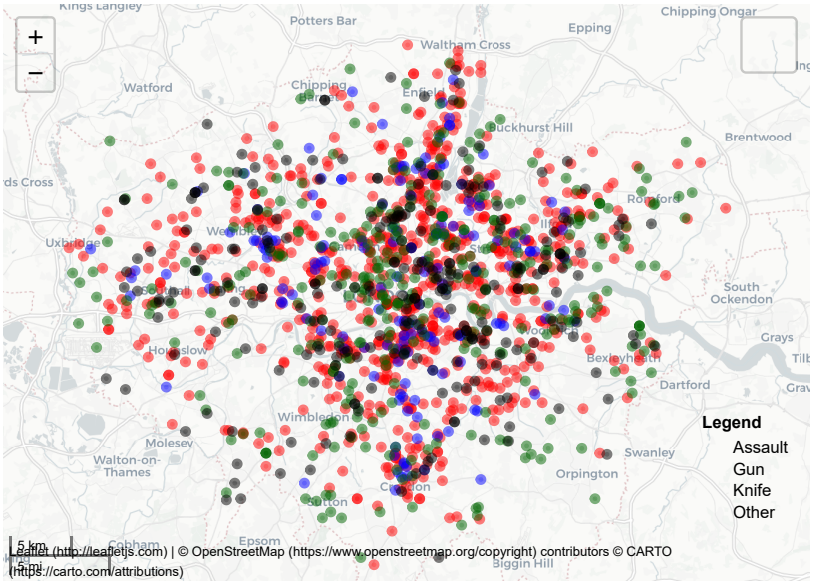
assault <- mmap5 %>%
  filter(weapon == "Assault") # Grouping layers - none

other <- mmap5 %>%
  filter(weapon == "Other") # Grouping layers - other


library(leaflet)

mmap5 %>%
 leaflet() %>%
  setView(lng=-0.1, lat=51.51, zoom=10) %>%
  addProviderTiles("CartoDB") %>%
  addScaleBar(position = "bottomleft") %>%
  addCircleMarkers(data = knife,
                   lng = ~longitude,
                   lat = ~latitude,
                   radius = 2,
                   color = ~pal(weapon),
                   group = "Knife",
                   popup = ~paste("<b>", ID, "</b>", "<br/>", date, "<br/>", age, "<br/>", sex, "<br/>", weapon, "<br/>", st
atus)) %>%
  addCircleMarkers(data = gun,
                   lng = ~longitude,
                   lat = ~latitude,
                   radius = 2,
                   color = ~pal(weapon),
                   group = "Gun",
                   popup = ~paste("<b>", ID, "</b>", "<br/>", date, "<br/>", age, "<br/>", sex, "<br/>", weapon, "<br/>", st
atus)) %>%
  addCircleMarkers(data = assault,
                   lng = ~longitude,
                   lat = ~latitude,
                   radius = 2,
                   color = ~pal(weapon),
                   group = "Assault",
                   popup = ~paste("<b>", ID, "</b>", "<br/>", date, "<br/>", age, "<br/>", sex, "<br/>", weapon, "<br/>", st
atus)) %>%
  addCircleMarkers(data = other,
                   lng = ~longitude,
                   lat = ~latitude,
                   radius = 2,
                   color = ~pal(weapon),
                   group = "Other",
                   popup = ~paste("<b>", ID, "</b>", "<br/>", date, "<br/>", age, "<br/>", sex, "<br/>", weapon, "<br/>", st
atus)) %>%

  addLegend(position = "bottomright",
            pal = pal,
            values = c("Knife", "Gun", "Assault", "Other"),
            title = "Legend") %>%
  addLayersControl(
    overlayGroups = c("Knife",
                      "Gun",
                      "Assault",
                      "Other"))
```




## Layered point map by ethnicity

*This layered point map shows the distribution of all homicide offences in the data for 2008-2020 and provides coloured circles based on the ethnicity of the victim.

```
pal <- colorFactor(palette = c("red", "blue", "dark green", "black", "purple"),
                   levels = c("Black", "White", "Asian", "Mixed", "Unknown")) # Creating a 5 colour palette

black <- mmap5 %>%
  filter(ethnicity == "Black") # Grouping layers - black

white <- mmap5 %>%
  filter(ethnicity == "White") # Grouping layers - white

asian <- mmap5 %>%
  filter(ethnicity == "Asian") # Grouping layers - asian

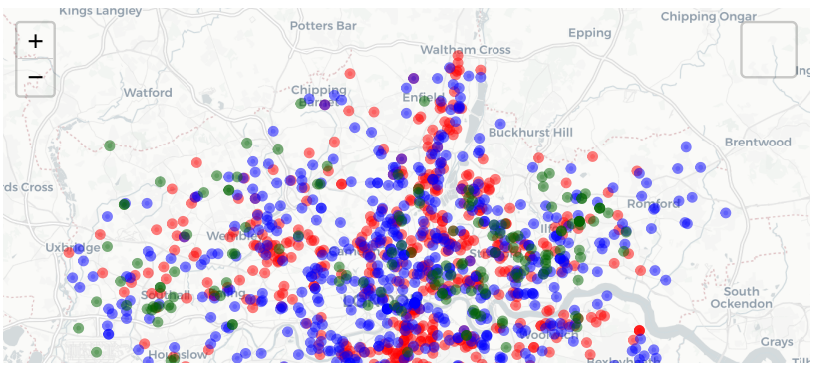
mixed <- mmap5 %>%
  filter(ethnicity == "Mixed") # Grouping layers - mixed

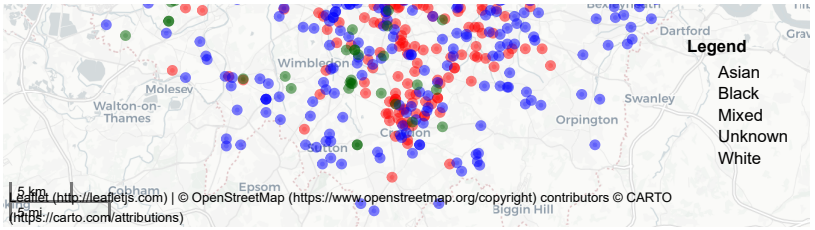
unknown <- mmap5 %>%
  filter(ethnicity == "Unknown") # Grouping layers - unknown


library(leaflet)

mmap5 %>%
 leaflet() %>%
  setView(lng=-0.1, lat=51.51, zoom=10) %>%
  addProviderTiles("CartoDB") %>%
  addScaleBar(position = "bottomleft") %>%
  addCircleMarkers(data = black,
                   lng = ~longitude,
                   lat = ~latitude,
                   radius = 2,
                   color = ~pal(ethnicity),
                   group = "Black",
                   popup = ~paste("<b>", ID, "</b>", "<br/>", date, "<br/>", age, "<br/>", sex, "<br/>", weapon, "<br/>", st
atus)) %>%
  addCircleMarkers(data = white,
                   lng = ~longitude,
                   lat = ~latitude,
                   radius = 2,
                   color = ~pal(ethnicity),
                   group = "White",
                   popup = ~paste("<b>", ID, "</b>", "<br/>", date, "<br/>", age, "<br/>", sex, "<br/>", weapon, "<br/>", st
atus)) %>%
  addCircleMarkers(data = asian,
                   lng = ~longitude,
                   lat = ~latitude,
                   radius = 2,
                   color = ~pal(ethnicity),
                   group = "Asian",
                   popup = ~paste("<b>", ID, "</b>", "<br/>", date, "<br/>", age, "<br/>", sex, "<br/>", weapon, "<br/>", st
atus)) %>%
  addCircleMarkers(data = mixed,
                   lng = ~longitude,
                   lat = ~latitude,
                   radius = 2,
                   color = ~pal(ethnicity),
                   group = "Mixed",
                   popup = ~paste("<b>", ID, "</b>", "<br/>", date, "<br/>", age, "<br/>", sex, "<br/>", weapon, "<br/>", st
atus)) %>%
    addCircleMarkers(data = unknown,
                   lng = ~longitude,
                   lat = ~latitude,
                   radius = 2,
                   color = ~pal(ethnicity),
                   group = "Unknown",
                   popup = ~paste("<b>", ID, "</b>", "<br/>", date, "<br/>", age, "<br/>", sex, "<br/>", weapon, "<br/>", st
atus)) %>%

  addLegend(position = "bottomright",
            pal = pal,
            values = c("Black", "White", "Asian", "Mixed", "Unknown"),
            title = "Legend") %>%
  addLayersControl(
    overlayGroups = c("Black",
                      "White",
                      "Asian",
                      "Mixed", "Unknown"))
```






## Cluster map, female victims

The cluster map is filtered to show the location of all female homicides in the period 2008 to 2020. Key concentrations at the county level include:

- Lower Holloway
- Hoxton/Shorditch
- Stepney
- Southwark

```
femaleVictims <- mmap5 %>%
  filter(sex == 'F')

femaleVictims %>% # Map showing all point data
 leaflet() %>%
  setView(lng =-0.1, lat=51.51, zoom=10) %>%
  addProviderTiles(providers$OpenStreetMap) %>%
  addScaleBar %>%
  addCircleMarkers(lng = ~longitude, lat = ~latitude, popup = ~paste0("<b>", ID, "</b>", "<br/>", date, "<br/>", age, "<br/
>", sex, "<br/>", weapon, "<br/>", borough), radius = 4, clusterOptions = markerClusterOptions)
```