NYPD Shootings: Murder Rates of Single- and Multi-victim Shootings

Ron Sielinski

2024-06-05

Introduction

New York City has kept track of all the shootings that have taken place within its boroughs since 2006. This analysis compares the murder rates of single- and multi-victim shootings.

Motivating Hypothesis

The more people who get shot in an incident, the more likely that someone will die. That makes sense mathematically: If only one person gets shot, only one person has a chance of dying. But if twelve people get shot, twelve people have a chance of dying.

But that doesn't necessarily mean that multi-victim incidents are more deadly on a *per-victim* basis. If a shooting involves only one victim, there's a good chance that the perpetrator will have meant to harm the victim, increasingly the likelihood of fatal injuries. In a multi-victim shooting, however, there's a reasonable chance that some of the victims will have been innocent bystanders and their injuries incidental. On a per-victim basis, we might actually find that multi-victim shootings are *less* deadly.

This analysis will explore over 20,000 shootings that taken place in New York City since 2006 to find out.

Raw Data

The City of New York publishes its shooting data at https://data.cityofnewyork.us.

```
###################
## Load Packages
###################

library(tidyverse)

###################
## Load the Data
###################

# load data from the City of New York
data_file <- 'https://data.cityofnewyork.us/api/views/833y-fsy8/rows.csv?accessType=DOWNLOAD'
ny_dat <- read_csv(data_file)

# clean up the environment
rm(data_file)
```

Preparing the Data

The city also publishes a guide to the data at https://data.cityofnewyork.us/Public-Safety/NYPD-Shooting-Incident-Data-Historic-/833y-fsy8/about_data. For this analysis, we only need a small subset of the data:

- `INCIDENT_KEY`: A unique identifier for each incident
- `STATISTICAL_MURDER_FLAG`: A Boolean that indicates whether the shooting resulted in the victim's death
- `PERP_AGE_GROUP`, `PERP_SEX`, `PERP_RACE`: A set of factors that describe the perpetrator

```
###################
## Tidy the Data
###################

# for convenience, convert column names to lower case (to make typing easier)
colnames(ny_dat) <- colnames(ny_dat) |>
  str_to_lower()

# perpetrator factors
ny_dat$perp_age_group <- as.factor(ny_dat$perp_age_group)
ny_dat$perp_sex <- as.factor(ny_dat$perp_sex)
ny_dat$perp_race <- as.factor(ny_dat$perp_race)

# keep only fields needed to analyze multi-victim incidents
ny_dat <- ny_dat |>
  select(incident_key, statistical_murder_flag, starts_with('perp'))
```

Exploring the Data

Victims

The first question that we want to answer is how many incidents have more than one victim?

```
#########
## EDA
#########

# create a data frame that summarizes the number of victims and murders for
# each incident
incident_cnts <- ny_dat |>
  summarize(victims = n(),
            murders = sum(statistical_murder_flag),
            .by = incident_key) |>
  arrange(desc(victims), desc(murders))

head(incident_cnts)
```

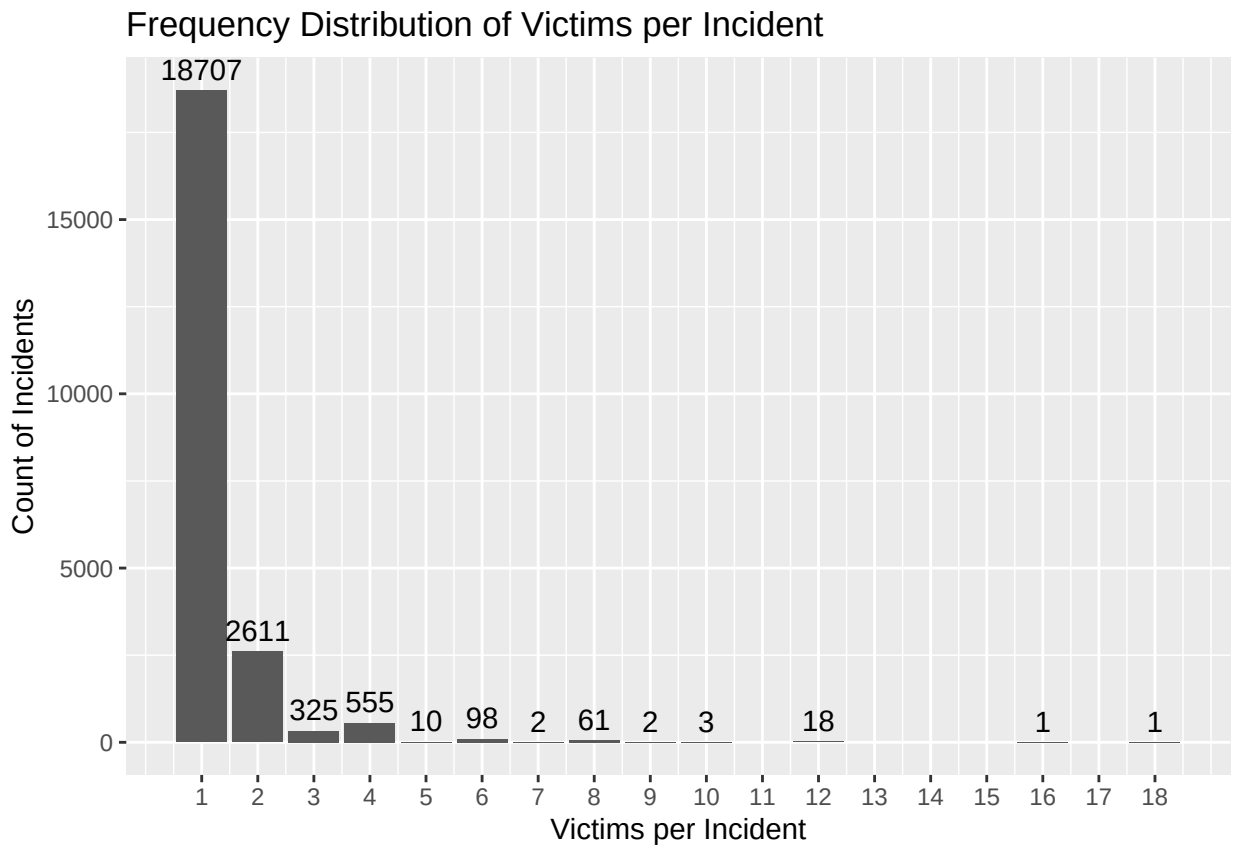
```
## # A tibble: 6 x 3
##   incident_key victims murders
##          <dbl>   <int>   <int>
## 1    173354054      18       9
## 2    263503175      16       8
```

```
## 3     85875439       12        6
## 4    212640102       12        6
## 5    215776333       12        6
## 6    215034244       12        6
```

The results show that one incident had eighteen victims!

However, there is only one eighteen-victim incident out of 22394 incidents.

```
# look at the frequency distribution of victims per incident
incident_cnts |>
  count(victims) |>
  ggplot(aes(x = victims, y = n)) +
  geom_bar(stat = 'identity') +
  scale_x_continuous(breaks = pretty(incident_cnts$victims,
                                     n = length(unique(incident_cnts$victims)))) +
  geom_text(aes(x = victims, y = n, label = n), vjust = -0.5) +
  labs(title = 'Frequency Distribution of Victims per Incident',
       y = 'Count of Incidents',
       x = 'Victims per Incident')
```

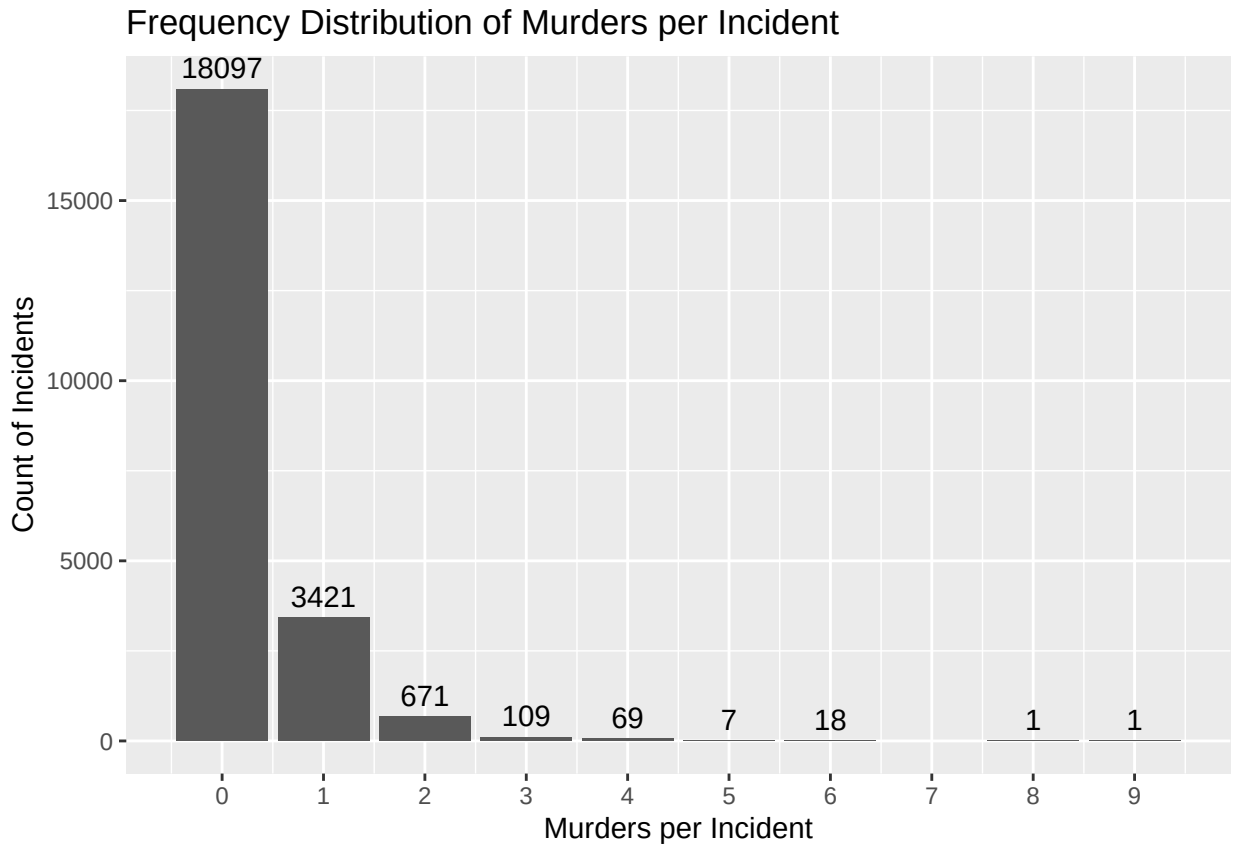


Plotting the frequency distribution of the victims per incident reveals that most incidents had only one victim. (By definition, an incident involves at least one victim, so no incidents can have zero victims.)

Given the nature of the data (i.e., a count of victims) and the shape of the curve, the count of victims per incident appears to follow a Poisson distribution.

Murders

```
# look at the frequency distribution of murders per incident
# we'll reuse this plot, so store it into the environment
murders_plot <- incident_cnts |>
  count(murders) |>
  ggplot(aes(x = murders, y = n)) +
  geom_bar(stat = 'identity') +
  scale_x_continuous(breaks = pretty(incident_cnts$murders,
                                     n = length(unique(incident_cnts$murders)))) +
  geom_text(aes(x = murders, y = n, label = n), vjust = -0.5) +
  labs(title = 'Frequency Distribution of Murders per Incident',
       y = 'Count of Incidents',
       x = 'Murders per Incident')

murders_plot
```



A frequency distribution plot of the murders per incident looks again like a Poisson distribution: The count of incidents decreases rapidly as the murders per incident increases.

In this case, however, it's possible for incidents to have no murders, which is most often the case.

Perpetrators

A final question to explore is the number of perpetrators (or shooters) per incident. The data doesn't specifically count or uniquely identify perpetrators, but for each incident, we can use any changes in the

perpetrator(s) age, sex, or race to determine whether multiple shooters were involved.

Here's a single incident that had shooters from five separate demographic groups.

```
## # A tibble: 5 x 4
##   incident_key perp_age_group perp_sex perp_race
##          <dbl> <fct>          <fct>    <fct>
## 1     90128858 <18            F        BLACK
## 2     90128858 18-24          F        BLACK
## 3     90128858 25-44          M        BLACK
## 4     90128858 <18            F        WHITE HISPANIC
## 5     90128858 <18            M        BLACK
```

The obvious limitation with this approach is that it's unable to distinguish between perpetrators who are the same age, sex, and race. As such, the number of differences that we detect across the set of factors will be the *minimum* number of shooters.

```
# for each incident, count the number of victims with the same set of perp factors
# each row in ny_dat is a victim, so the count of rows is the count of victims
perp_factor_cnts <- ny_dat |>
  summarize(victims_by_perp_factors = n(),
            murders_by_perp_factors = sum(statistical_murder_flag),
            .by = c(incident_key, perp_age_group, perp_sex, perp_race))

# for each incident, count the number of perp factor sets and the total number
# of victims and murders
factor_cnts  <- perp_factor_cnts |>
  summarize(factor_sets = n(),
            victims_cnt = sum(victims_by_perp_factors),
            murders_cnt = sum(murders_by_perp_factors),
            .by = incident_key)

# clean up the environment
rm(perp_factor_cnts)

# look at incidents that have involved the most number of shooters
factor_cnts |>
  rename(shooter_cnt_min = factor_sets) |>
  arrange(desc(shooter_cnt_min)) |>
  head()
```

```
## # A tibble: 6 x 4
##   incident_key shooter_cnt_min victims_cnt murders_cnt
##          <dbl>           <int>       <int>       <int>
## 1     90128858               5           5           5
## 2    231246225               5           5           5
## 3    140088782               5           5           5
## 4    249207672               5           5           0
## 5     51559646               5           5           5
## 6     56662014               5           5           0
```

The result shows us that multiple incidents have involved five (or more) shooters, including incidents where of the shooters may have all murdered each other.

This adds a new dimension to the original hypothesis. Without explicitly stating it, we had presumed that all incidents were perpetrated by single shooters. However, given that there were incidents that involved multiple shooters, there's a good chance that multiple guns were fired—potentially at close range—increasing the chance of death for everyone involved.

Statistical Models

Per-victim Model

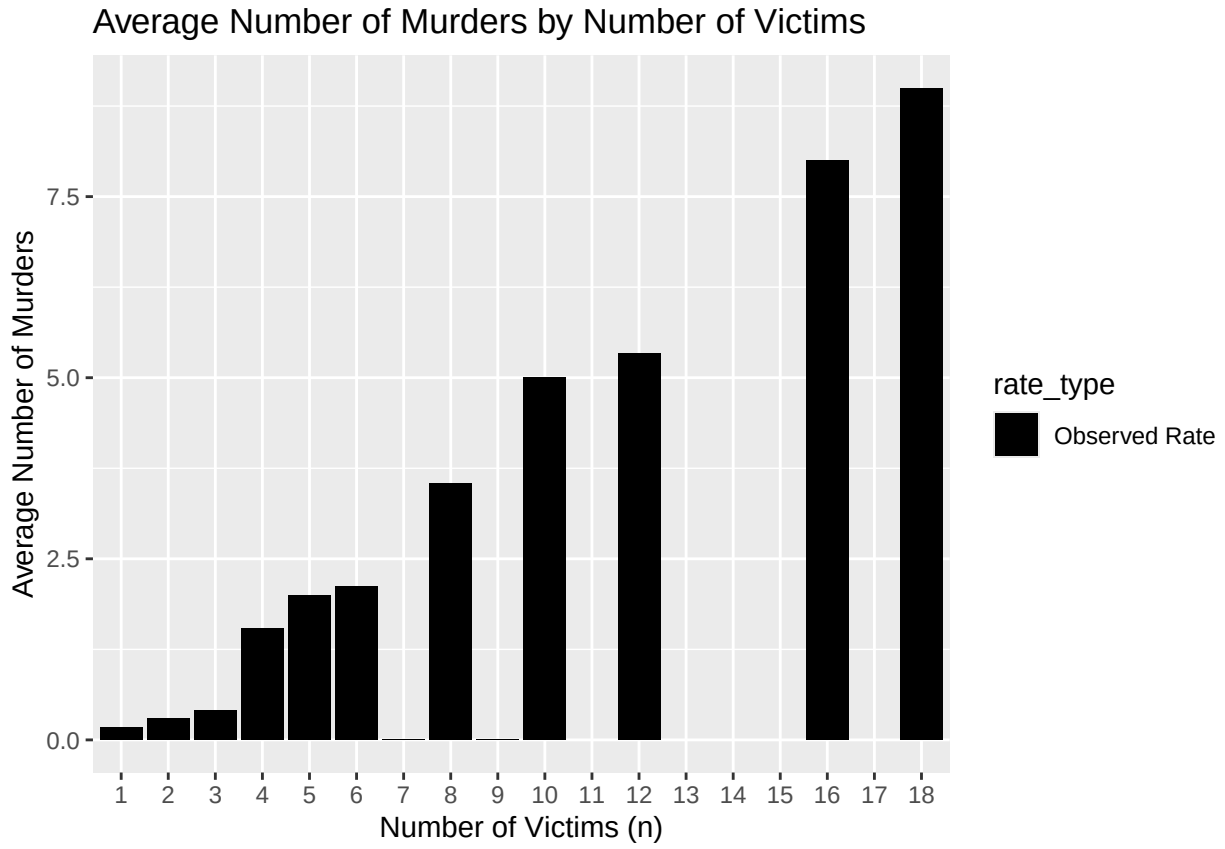
Because the frequency distribution plots (above) focus on just victims and just murders, they don't offer much insight into the relationship between victims and murders.

Plotting the average number of murders by the number of victims shows us that, as the number of victims (n) increases, so does the average number of murders.

It's worth noting, that for some values of n, there were no incidents. Looking back on the frequency distribution plot of victims per incident reminds us that there were no incidents with 11, 13, 14, 15, or 17 victims. For large values of n, we might not have enough data to draw statistically significant inferences.

```
#########################
## Bar Plot of Actuals
#########################

# range for the number of victims (n)
n_victims <- min(incident_cnts$victims):max(incident_cnts$victims)
# range for the number of murders
n_murders <- min(incident_cnts$murders):max(incident_cnts$murders)

# calculate the average number of murders for each n
murder_rates <- incident_cnts |>
  summarize(incidents = n(),
            murders = sum(murders),
            # average number of murders
            per_incident_rate = sum(murders) / n(),
            .by = victims) |>
  # observed murders per victim
  mutate(per_victim_rate = per_incident_rate / victims) |>
  arrange(victims)

# add rows to the data frame for any missing actuals
murder_rates <- data.frame(victims = n_victims) |>
  left_join(murder_rates, by = 'victims')

# convert victims to a factor to enhance plotting
#murder_rates$victims <- factor(murder_rates$victims, levels = n_victims)

# plot the result
murder_rates |>
  select(victims, starts_with('per_incident')) |>
  pivot_longer(cols = starts_with('per_incident'),
               names_to = 'rate_type',
               values_to = 'rate') |>
  ggplot(aes(x = factor(victims, levels = n_victims), y = rate, fill = rate_type)) +
  geom_bar(stat = 'identity', position = position_dodge(width = 0.9)) +
```

```
  #geom_text(aes(x = factor(victims), y = rate, label = sprintf("%.2f", rate)),
  #          position = position_dodge(width = 0.9),
  #          hjust = 1.25,
  #          angle = 90,
  #          color = 'white') +
  scale_x_discrete(drop = F) +
  scale_fill_manual(values = c('black'), labels = c('Observed Rate')) +
  labs(title = 'Average Number of Murders by Number of Victims',
       y = 'Average Number of Murders',
       x = 'Number of Victims (n)')
```



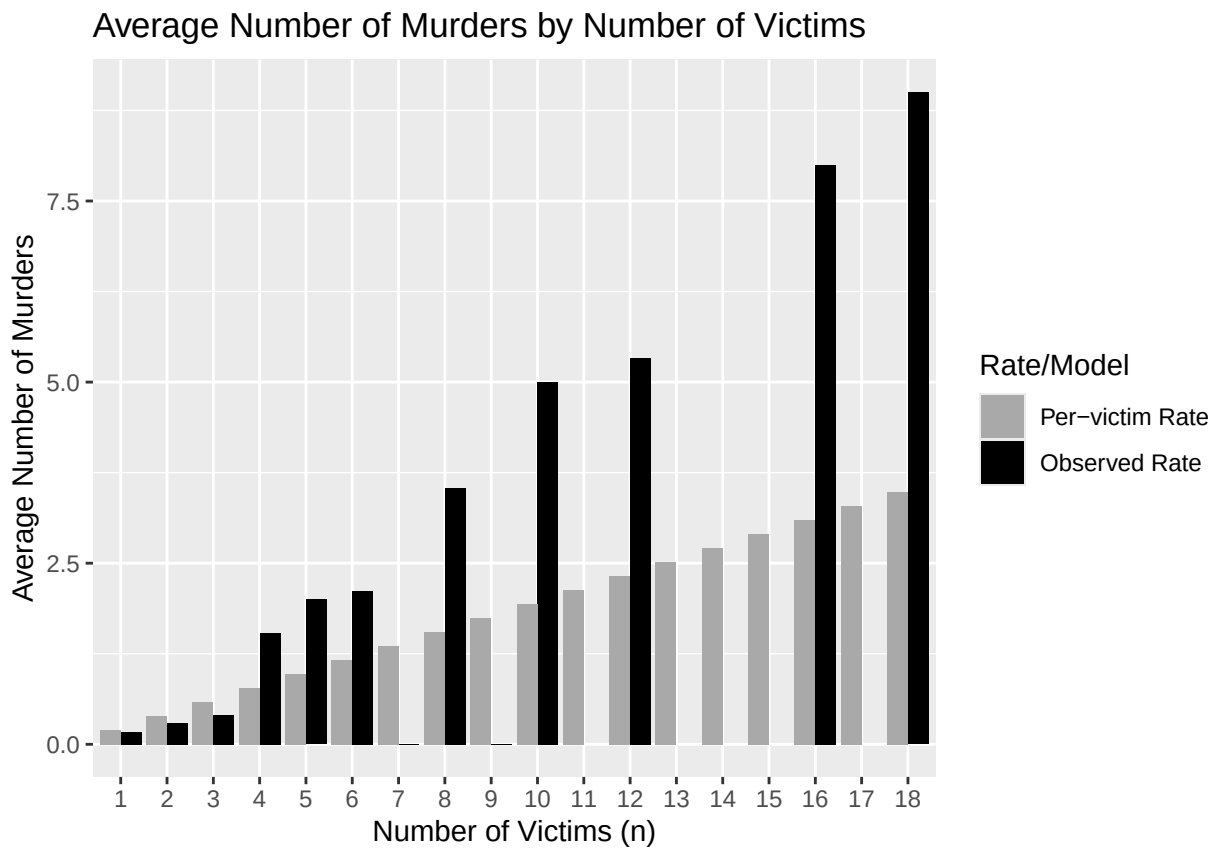
To address the missing data, we can create a model that predicts the number of murders we could have expected.

The simplest model is based on a global murders-to-victims rate. We simply multiply n by the per-victim rate to predict the the average number of murders we'd expect for each n.

However, a plot of the observed and expected values suggests that the resulting model performs poorly, overestimating the average number of murders for small values of n and underestimating the average for large n.

```
####################################
## Global Murders-to-victims Rate
####################################

# calculate a global murders-to-victims rate
global_murders_per_victim <- sum(incident_cnts$murders) / sum(incident_cnts$victims)
```

```r
# use global_rate to predict the number of murders for each n
murder_rates <- murder_rates |>
  mutate(
    per_incident_global = victims * global_murders_per_victim,
    per_victim_global = global_murders_per_victim,
    murders_global = incidents * per_incident_global
  )

# plot the results
murder_rates |>
  select(victims, starts_with('per_incident')) |>
  pivot_longer(cols = starts_with('per_incident'),
               names_to = 'rate_type',
               values_to = 'rate') |>
  ggplot(aes(x = factor(victims, levels = n_victims), y = rate, fill = rate_type)) +
  geom_bar(stat = 'identity', position = position_dodge(width = 0.9)) +
  #geom_text(aes(x = factor(victims), y = rate, label = sprintf("%.2f", rate)),
  #          position = position_dodge(width = 0.9),
  #          hjust = 1.25,
  #          angle = 90,
  #          color = 'white') +
  scale_x_discrete(drop = F) +
  scale_fill_manual(name = 'Rate/Model',
                    values = c('darkgrey', 'black'),
                    labels = c('Per-victim Rate ', 'Observed Rate')) +
  labs(title = 'Average Number of Murders by Number of Victims',
       y = 'Average Number of Murders',
       x = 'Number of Victims (n)')
```



To quantify the model's performance, we can calculate its Root Mean Squared Error (RMSE).

```
# Calculate the root mean squared error
RMSE <- function(actual_values, estimated_values) {
  ((actual_values - estimated_values) ^ 2) |>
    mean() |>
    sqrt()
}

RMSE(incident_cnts$murders, incident_cnts$victims * global_murders_per_victim)
```

```
## [1] 0.5179426
```

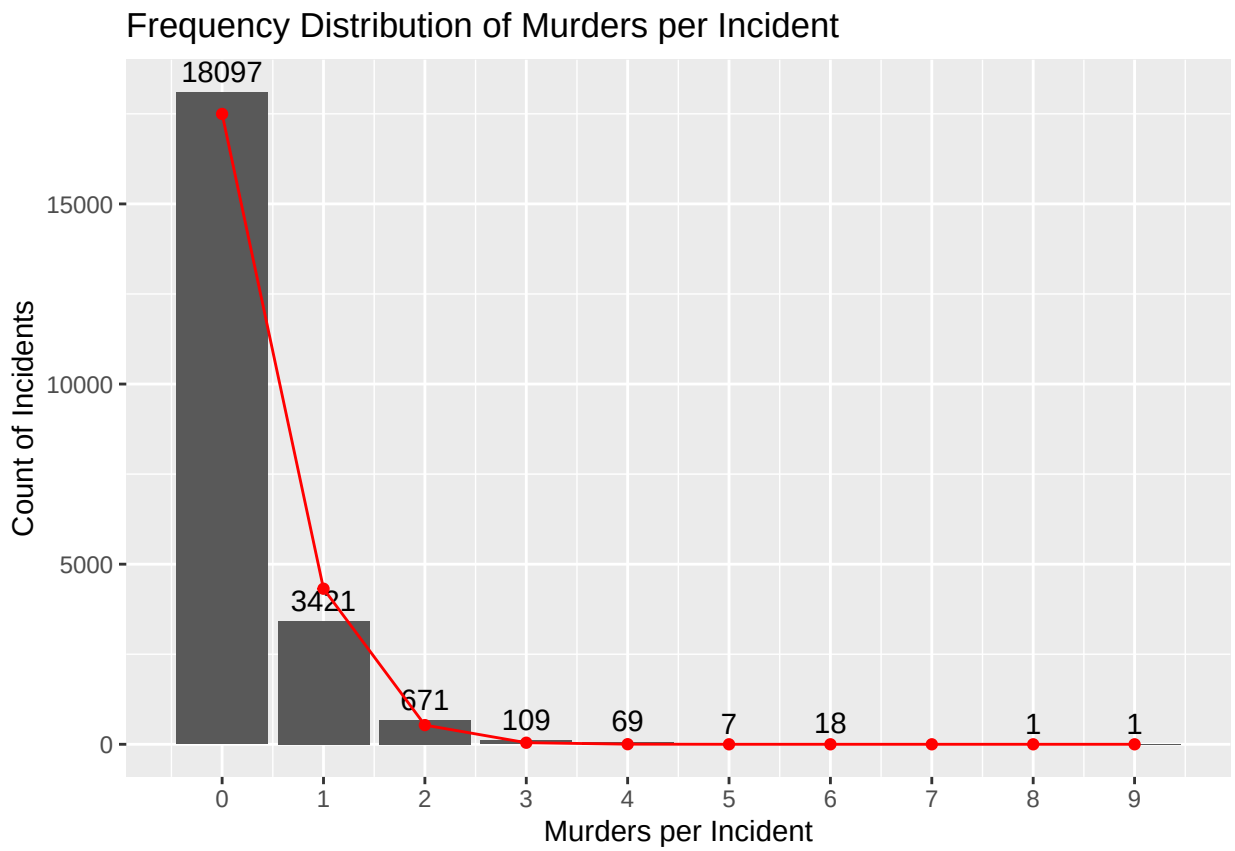
The RMSE for estimates based on the per-victim model is 0.52.

Per-incident Model

Alternatively, we can model the number of murders using a Poisson distribution. In doing so, we shift our perspective slightly from the average number of murders per *victim*, to the average number of murders per *incident*.

We can then use the properties of a Poisson distribution to model the number of murders that we can expect for each number of murders per incident.

```r
################################################
## model murders using a Poisson distribution
################################################

cnt_all_incidents <- nrow(incident_cnts)

# average number of murders per incident
murders_lambda <- mean(incident_cnts$murders)

# we'll want to compare the model results with actuals, so count the incidents
# and sum the murders for each number of murders
pois_model <- incident_cnts |>
  summarize(incidents_cnt = n(),
            murders_cnt = sum(murders),
            .by = murders) |>
  # calculate the percentage of incidents that had n murders
  mutate(incidents_pct = incidents_cnt / cnt_all_incidents)

# add rows to the data frame for values of n that had no murders
missing_n <- data.frame(murders = n_murders[!(n_murders %in% pois_model$murders)],
                        incidents_cnt  = 0,
                        murders_cnt = 0,
                        incidents_pct  = 0
)
pois_model <- pois_model |>
  bind_rows(missing_n) |>
  arrange(murders)

# use the Poisson model to predict the percentage of incidents with n murders
# and the resulting number of incidents
pois_model <- pois_model |>
  mutate(model_pct = dpois(murders, lambda = murders_lambda),
         incidents_pred = round(cnt_all_incidents * model_pct, 0),
         murders_pred = incidents_pred * murders)

# confirm that the model accounts for all incidents (numbers should be close)
#sum(pois_model$incidents_cnt)
#sum(pois_model$incidents_pred)

# confirm that the model accounts for all murders (numbers should be close)
#sum(pois_model$murders_cnt)
#sum(pois_model$murders_pred)

# clean up the environment
rm(missing_n)

# plot the data and the model
murders_plot +
  geom_point(data = pois_model, aes(x = murders, y = incidents_pred), color = 'red') +
  geom_line(data = pois_model, aes(x = murders, y = incidents_pred), color = 'red')
```



Validate the Poisson Model

We can test the model's goodness of fit with a chi-squared test, which compares the observed and expected percentage of murders for each n.

```
####################
## Validate model
####################

# perform the chi-squared test
# the expected counts/frequencies can't be too small (ideally, >= 5)
# one workaround is to combine bins
max_expected_bin <- max(which(pois_model$incidents_pred >= 5))

# for observed incidents, get the percentages for each bin up to the shared_max
# and combine whatever remains into another bin
incidents_pcts <- c(pois_model$incidents_pct[1:max_expected_bin],
                    sum(pois_model$incidents_pct[(max_expected_bin + 1):nrow(pois_model)])
)

# do the same for the modeled bins
model_pcts <- c(pois_model$model_pct[1:max_expected_bin],
                sum(pois_model$model_pct[(max_expected_bin + 1):nrow(pois_model)])
)

# A high p-value suggests that the observed data is consistent with the Poisson
# distribution, within the bounds of normal statistical variation
```

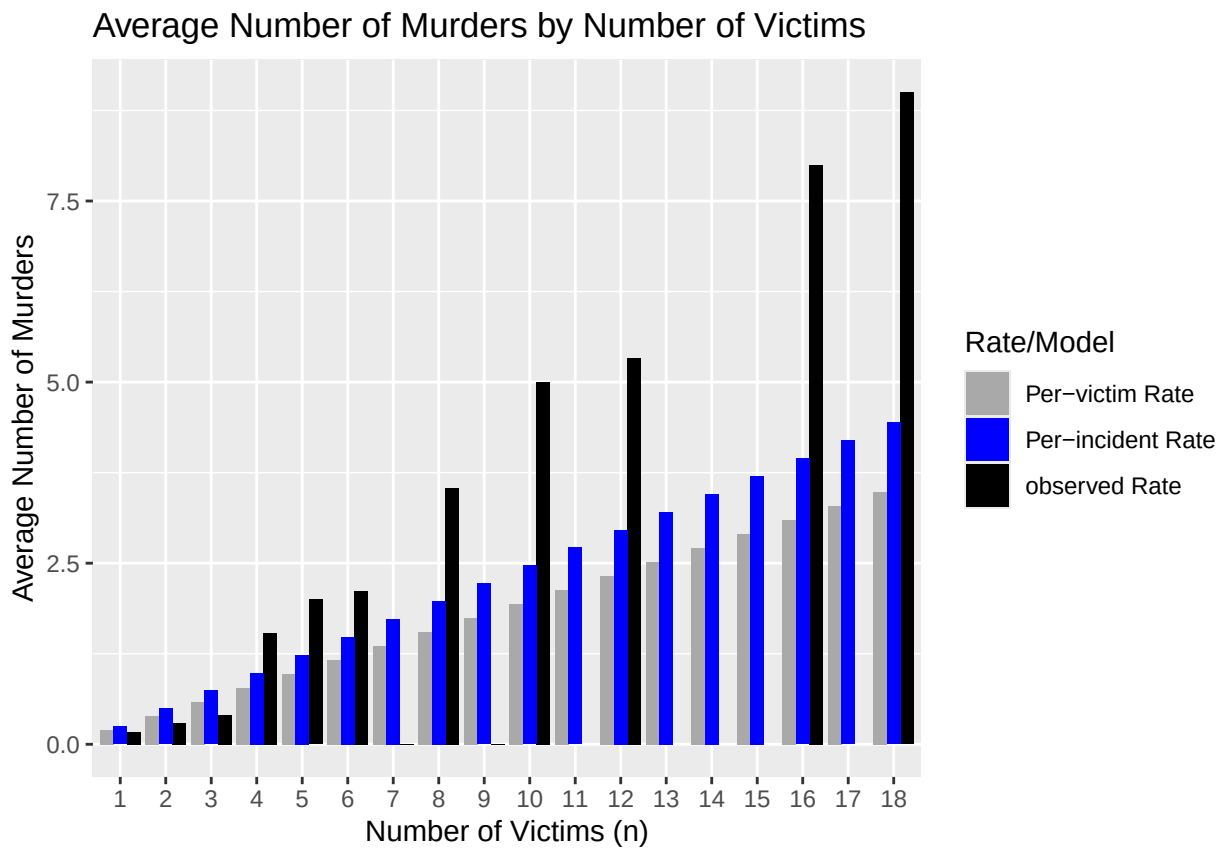
```r
chi_squared_test <- chisq.test(x = incidents_pcts, p = model_pcts)

# remember, we're looking at the first few bins (and everything else is binned together)
print(chi_squared_test)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  incidents_pcts
## X-squared = 0.15151, df = 4, p-value = 0.9973
```

A high p-value suggests that the observed data is consistent with the expected values generated by our Poisson model (within the bounds of normal statistical variation). In this case, the p-value is 0.997, which is reassuringly high.

Note, however, that we could only test the model's first 4 bins, because the number of incidents with >3 murders was too small to include in the test. (All incidents with >3 murders were combined into a single bin.) Nonetheless, incidents with <4 murders comprise 99.6% of incidents, so incidents with >3 murders are low-probability outliers.

```r
# add results from the Poisson model to the results data frame
murder_rates <- murder_rates |>
  mutate(
    # use the Poisson model to predict the number of murders
    per_incident_pois = victims * murders_lambda,
    per_victim_pois = per_incident_pois / victims,
    murders_pois = incidents * per_incident_pois
  )

# plot the results
murder_rates |>
  select(victims, starts_with('per_incident')) |>
  pivot_longer(cols = starts_with('per_incident'),
               names_to = 'rate_type',
               values_to = 'rate') |>
  ggplot(aes(x = factor(victims, levels = n_victims), y = rate, fill = rate_type)) +
  geom_bar(stat = 'identity', position = position_dodge(width = 0.9)) +
  #geom_text(aes(x = factor(victims), y = rate, label = sprintf("%.2f", rate)),
  #          position = position_dodge(width = 0.9),
  #          hjust = 1.25,
  #          angle = 90,
  #          color = 'white') +
  scale_x_discrete(drop = F) +
  scale_fill_manual(name = 'Rate/Model',
                    values = c('darkgrey', 'blue', 'black'),
                    labels = c('Per-victim Rate', 'Per-incident Rate', 'observed Rate')) +
  labs(title = 'Average Number of Murders by Number of Victims',
       y = 'Average Number of Murders',
       x = 'Number of Victims (n)')
```



For $n > 3$, the per-incident model (i.e., Poisson model) comes closer to predicting the observed murder rates than the per-victim model. For small n, the per-victim model is better.

Here, it's important not to confuse n (the number of victims) with the bins in the chi-squared test (murders per incident). Again, incidents with <4 murders comprise 99.6% of incidents and could have been spread across all values of n.

A more robust way to compare the overall performance of the per-incident and per-victim models is through their RSME values.

Although the Poisson models uses the same murders-per-incident rate for every incident, 0.25 murders per incident, we can still calculate the model's RMSE.

```
RMSE(incident_cnts$murders, rep(murders_lambda, nrow(incident_cnts)))
```

```
## [1] 0.5899636
```

Despite the fact that the per-incident model does a better job of predicting the average number of murders across more values of n, its RMSE (0.59) is higher (i.e., worse) than the RMSE of the per-victim model (0.52). One reason is that the latter model performs better for incidents with three or fewer victims, which comprise 96.65% of all incidents, so their errors have a substantial cumulative effect on the overall RMSE of a model.

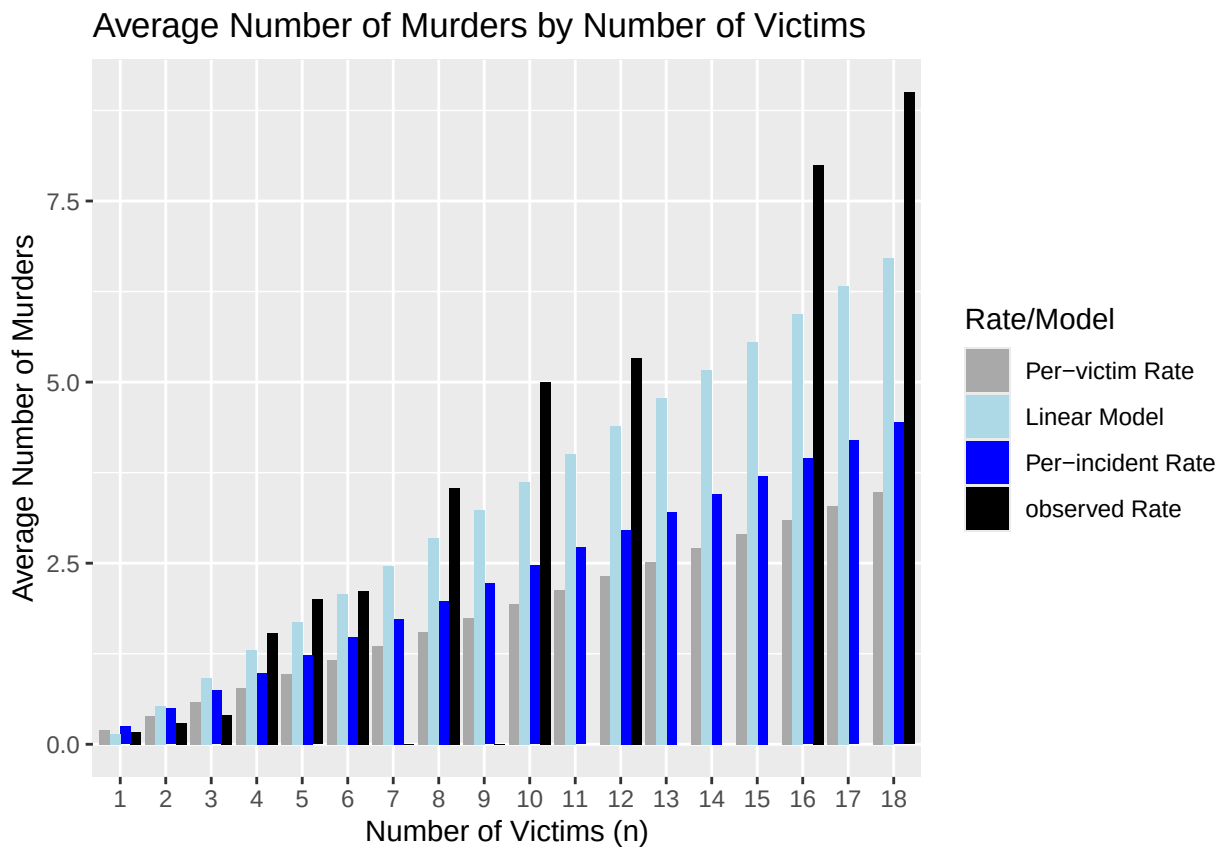
Linear Model

A linear model takes yet another approach, our final. The least squares method tries to capture the linear relationship between the number of victims and the number of murders across all data points, assuming each additional victim is associated with a *constant* change in the number of murders.

```r
# Build a simple linear model, based solely on the number of victims
lm_fit <- lm(murders ~ victims, data = incident_cnts)
summary(lm_fit)
```

```
##
## Call:
## lm(formula = murders ~ victims, data = incident_cnts)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3909 -0.1403 -0.1403 -0.1403  3.9276
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.246089   0.005956  -41.32   <2e-16 ***
## victims      0.386420   0.003895   99.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4917 on 22392 degrees of freedom
## Multiple R-squared:  0.3054, Adjusted R-squared:  0.3054
## F-statistic:  9845 on 1 and 22392 DF,  p-value: < 2.2e-16
```

```r
# use the model to predict the number of murders
murder_rates$per_incident_linear <- predict(lm_fit, newdata = murder_rates)

# calculate the per-victim rate
murder_rates <- murder_rates |>
  mutate(per_victim_linear = per_incident_linear / victims)

# plot the results
murder_rates |>
  select(victims, starts_with('per_incident')) |>
  pivot_longer(cols = starts_with('per_incident'),
               names_to = 'rate_type',
               values_to = 'rate') |>
  ggplot(aes(x = factor(victims, levels = n_victims), y = rate, fill = rate_type)) +
  geom_bar(stat = 'identity', position = position_dodge(width = 0.9)) +
  #geom_text(aes(x = factor(victims), y = rate, label = sprintf("%.2f", rate)),
  #          position = position_dodge(width = 0.9),
  #          hjust = 1.25,
  #          angle = 90,
  #          color = 'white') +
  # add the linear model
  #geom_line(aes(y = linear_model), color = 'red', linetype = 'dashed') +
  #geom_point(aes(y = linear_model), color = 'red') +
  scale_x_discrete(drop = F) +
  scale_fill_manual(name = 'Rate/Model',
                    values = c('darkgrey', 'lightblue', 'blue', 'black'),
                    labels = c('Per-victim Rate', 'Linear Model', 'Per-incident Rate', 'observed Rate')) +
  labs(title = 'Average Number of Murders by Number of Victims',
       y = 'Average Number of Murders',
       x = 'Number of Victims (n)')
```




```
RMSE(incident_cnts$murders, predict(lm_fit, newdata = incident_cnts))
```

```
## [1] 0.4916948
```

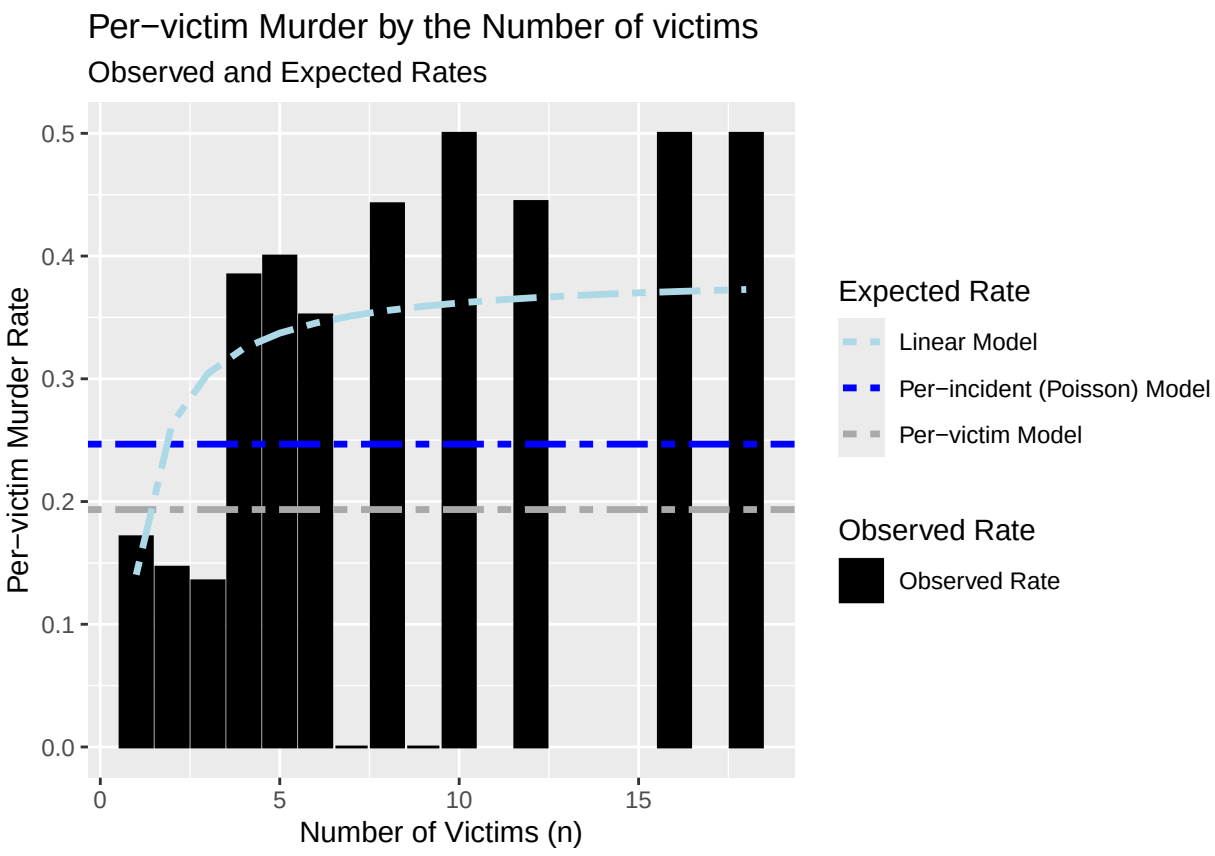
The RMSE for the linear model is 0.49, better than both the per-incident model (0.59) and the per-victim model (0.52).

Conclusion

Previous charts showed the expected average for the number of murders for each n. The plot below normalizes the number of murders by the number of victims, effectively showing the expected per-victim murder rate for each n.

```
##############################
## per-victim murder rates
##############################

# plot the per-victim murder rates
murder_rates |>
  ggplot(aes(x = victims, y = per_victim_rate)) +
  geom_col(aes(fill = 'Observed Rate'), color = 'black') +
  # add the global per-victim rate
  geom_hline(aes(yintercept = global_murders_per_victim, color = 'Per-victim Model'),
             linetype = 'twodash', linewidth = 1.2) +
```

```r
  # add the Poisson model's rate
  geom_hline(aes(yintercept = murders_lambda, color = 'Per-incident (Poisson) Model'),
             linetype = 'twodash', linewidth = 1.2) +
  # add the linear model's rate
  geom_line(aes(y = per_victim_linear, color = 'Linear Model'),
            linetype = 'twodash', linewidth = 1.2) +
  scale_color_manual(
    name = 'Expected Rate',
    values = c('Per-victim Model' = 'darkgrey',
               'Linear Model' = 'lightblue',
               'Per-incident (Poisson) Model' = 'blue',
               'Observed Rate' = 'black'),
    labels = c('Linear Model', 'Per-incident (Poisson) Model',
               'Per-victim Model', 'Observed Rate')) +
  scale_fill_manual(
    name = 'Observed Rate',
    values = c('Observed Rate' = 'black'),
    labels = c('Observed Rate')) +
  labs(title = 'Per-victim Murder by the Number of victims',
       subtitle = 'Observed and Expected Rates',
       y = 'Per-victim Murder Rate',
       x = 'Number of Victims (n)')
```



The per-victim and per-incident models predict that the murder rate stays the same for all values of n, which would suggest that all shootings are equally deadly, regardless of n. The linear model, on the other hand, predicts that the murder rate increases as n increases, which would suggest that multi-victim shootings are

actually *more* deadly than sinlge-victim shootings. (The model predicts an increased murder rate for *every* increment of n.)

Based on RMSE alone, we should favor the linear model: However, the RMSE for the linear model (0.49) is only slightly better than the per-victim model (0.52).

Moreover, the observed rate in the above plot suggests that the per-victim and linear models both have regions where one works better than the other. At a high-level, the observed rate is bifurcated—less than 0.2 murders per victim for $n < 4$, and ~0.4 murders per victim for $n > 3$—a difference of approximately 2x.

The per-victim model appears to work best for $n < 4$, and the linear model appears to work best for $n > 3$.

It's possible that the best way to model the murder rate would be to use a *combination* of models. There are multiple ways to combine models and achieve improved overall performance: mixture models, ensemble models, and more.

It's worth noting, however, that the upward trend in the murder rate for larger values of n could be an artifact of small sample sizes and that the linear model overfit to the few available incidents.

To mitigate the risk of small-sample bias elsewhere in this analysis, we used averages (i.e., global averages for the per-victim and per-incident models) and binning (i.e., combining bins with very few incidents to create a larger, more stable group for analysis). But nothing has been done to mitigate the risk of small-sample bias for the linear model. (The inclusion of confidence intervals might have made the effective of any small-sample bias more apparent.)

The data itself is another potential source of bias:

- The online guide to the data specifically refers to `INCIDENT_KEY` as a "randomly generated persistent ID for each arrest". That suggests that an incident might be counted only if a perpetrator is arrested. That might make the analysis blind to shootings—and murders—where an arrest did *not* occur.
- Likewise, the definition of "murder" is not clear. For example, does murder depend on a verdict or plea of guilty? Would the historic count of murders change if a perpetrator were acquitted? Similarly, is manslaughter (an *unintentional* killing) considered the same as—or different from—murder (an *intentional* killing)? A clear definition of "murder" is important for both a correct framing of the analysis and interpretation of results.

More research and analysis are required to address these issues.

Based on the analysis presented in this paper, we can reasonably conclude that the inflection point in murder rates isn't between single- and multi-victim shootings: It's between three and four victims. On average, the murder rate goes up when shootings involve four or more victims.