Assortativity, PLSC 508

Joseph Risi

2022-10-06

- The data for this tutorial comes from here, and it is used in the following paper: *Ba, Bocar A., Dean Knox, Jonathan Mummolo, and Roman Rivera. 2021. "The Role of Officer Race and Gender in Police-Civilian Interactions in Chicago." Science.*
- I conducted a lot of data cleaning myself. My Github repository can be found here where one can find all my data cleaning steps.
- The researchers who created this data submitted a series of FOIA requests to the Chicago Police Department and obtained their administrative records from 2012 - 2015. In this particular exercise, we are going to be looking at their stop and frisk data. In particular, we will be looking at officers who are recorded as having made a stop and frisk together.

```
library(here)
library(dplyr)
library(readr)
library(igraph)
library(ggplot2)
library(assortnet)
library(lubridate)
```

Step 1 is to read in officer attributes or in this case, node attributes.

```
# Age and experience are calculated using the last year of data available
officers <-
    read_csv(here("create-networks", "output", "officers.csv")) %>%
    mutate(age = 2015 - birth_year,
           exp = (ymd("2015-12-01") - appointed_month) / dyears(1))
```

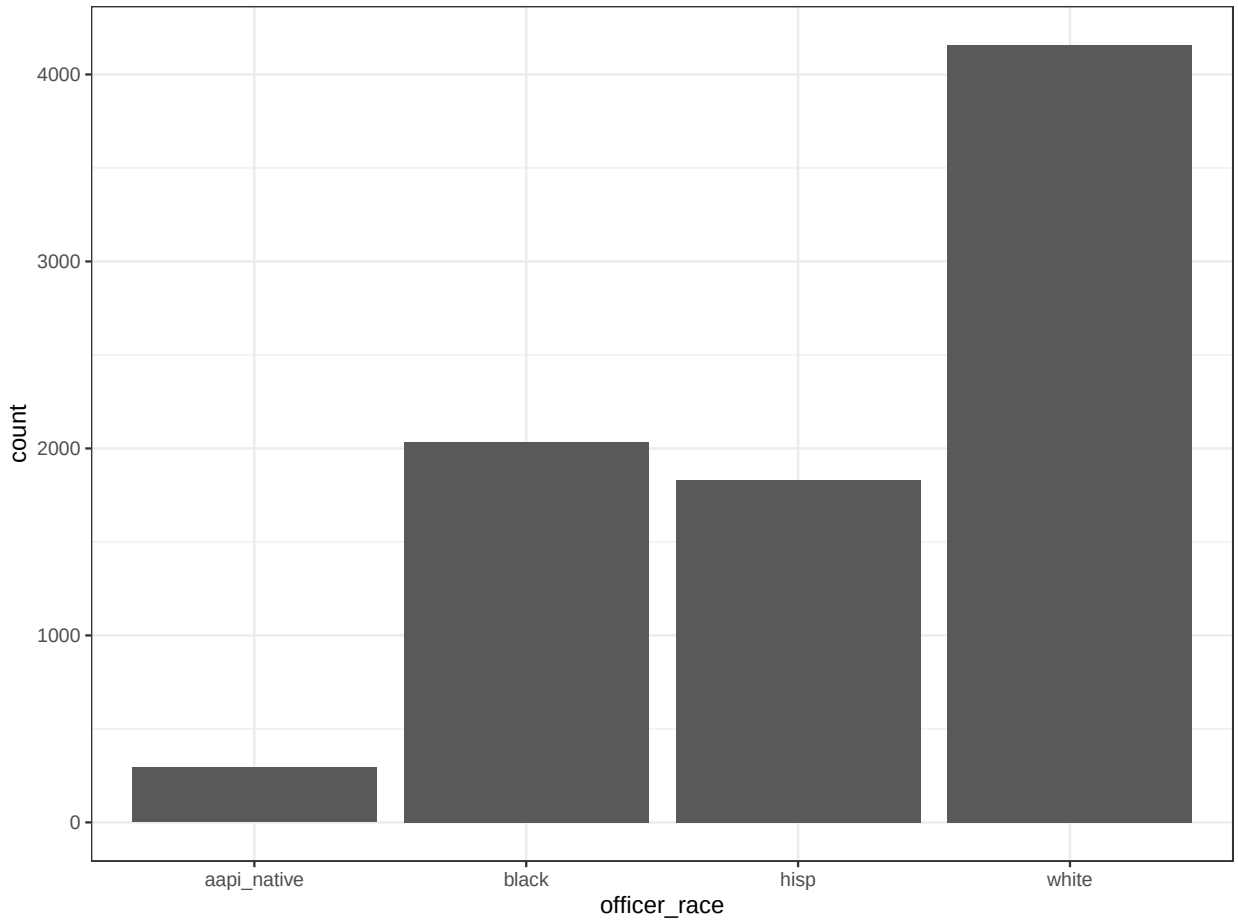
```
## Rows: 8319 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): officer_race, officer_sex
## dbl  (2): officer_id, birth_year
## lgl  (1): spanish
## date (1): appointed_month
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Graph each of the node attributes (officer race, officer sex, officer age, officer years of experience, and Spanish speaking ability).

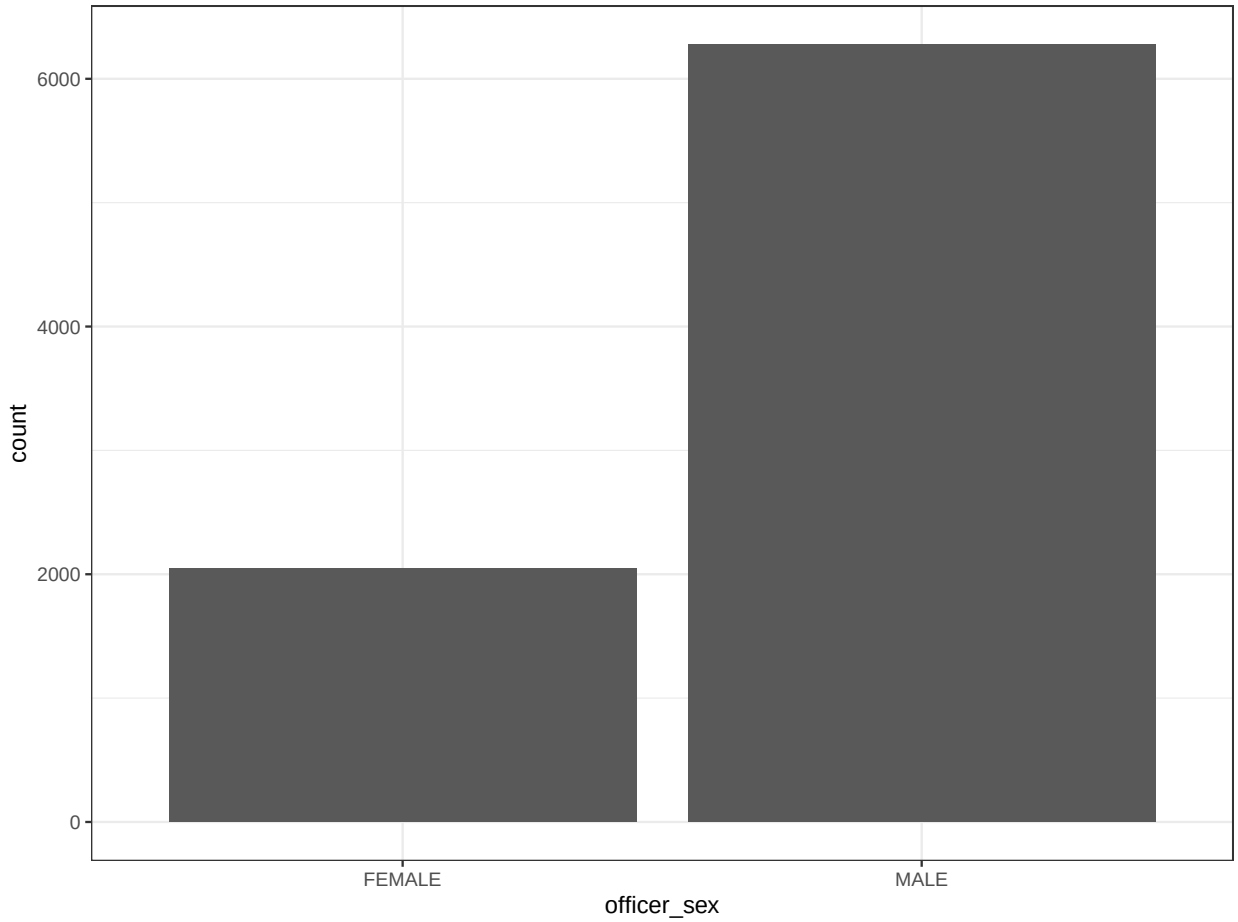
- 50% of officers are White.
- 75% are male.
- Median and mean age is around 44 and is approximately normally distributed with a standard deviation around 9.
- Median and mean years of experience is around 15 years and is approximately normally distributed

with a standard deviation around 8 years.

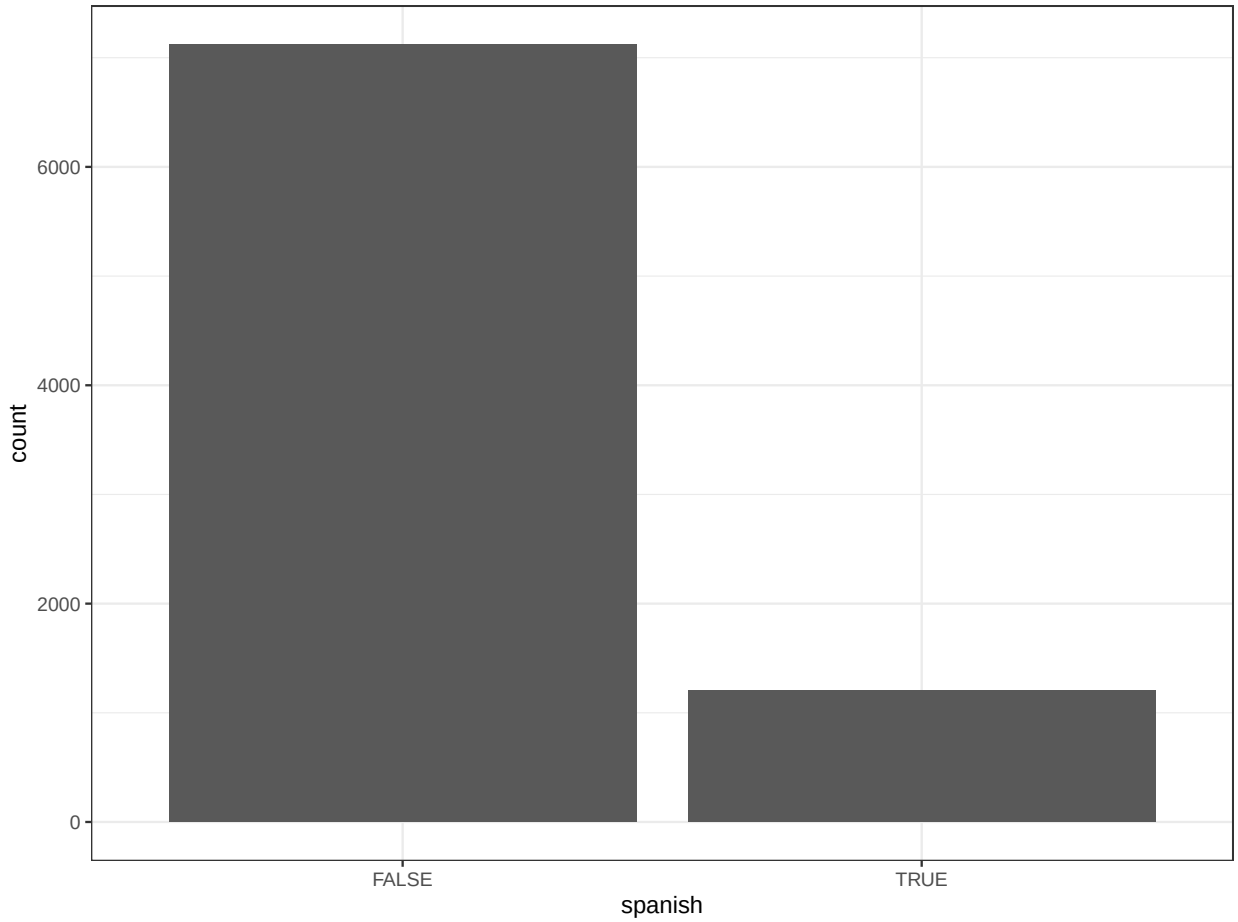
```
ggplot(officers, aes(x = officer_race)) + geom_bar() + theme_bw()
```



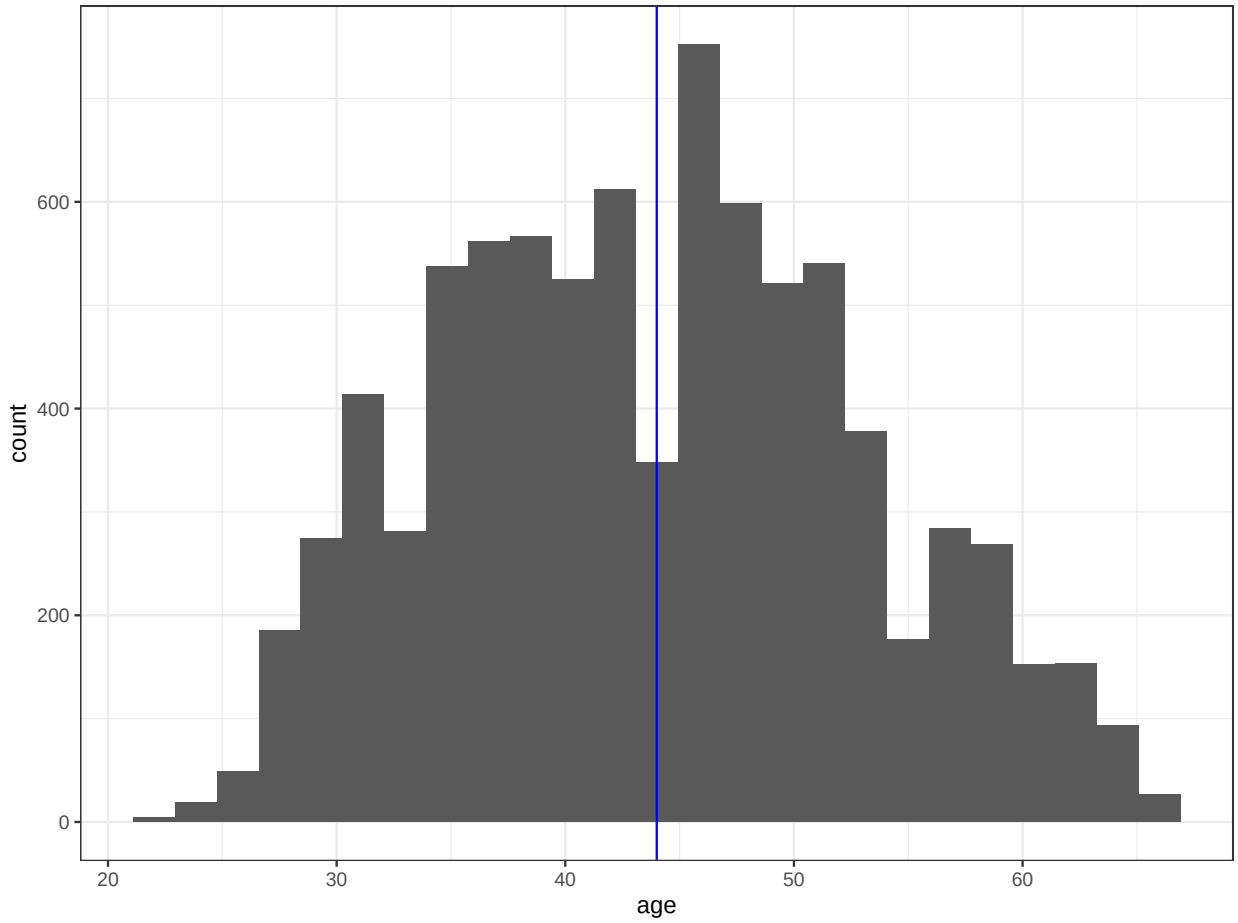

```
ggplot(officers, aes(x = officer_sex)) + geom_bar() + theme_bw()
```



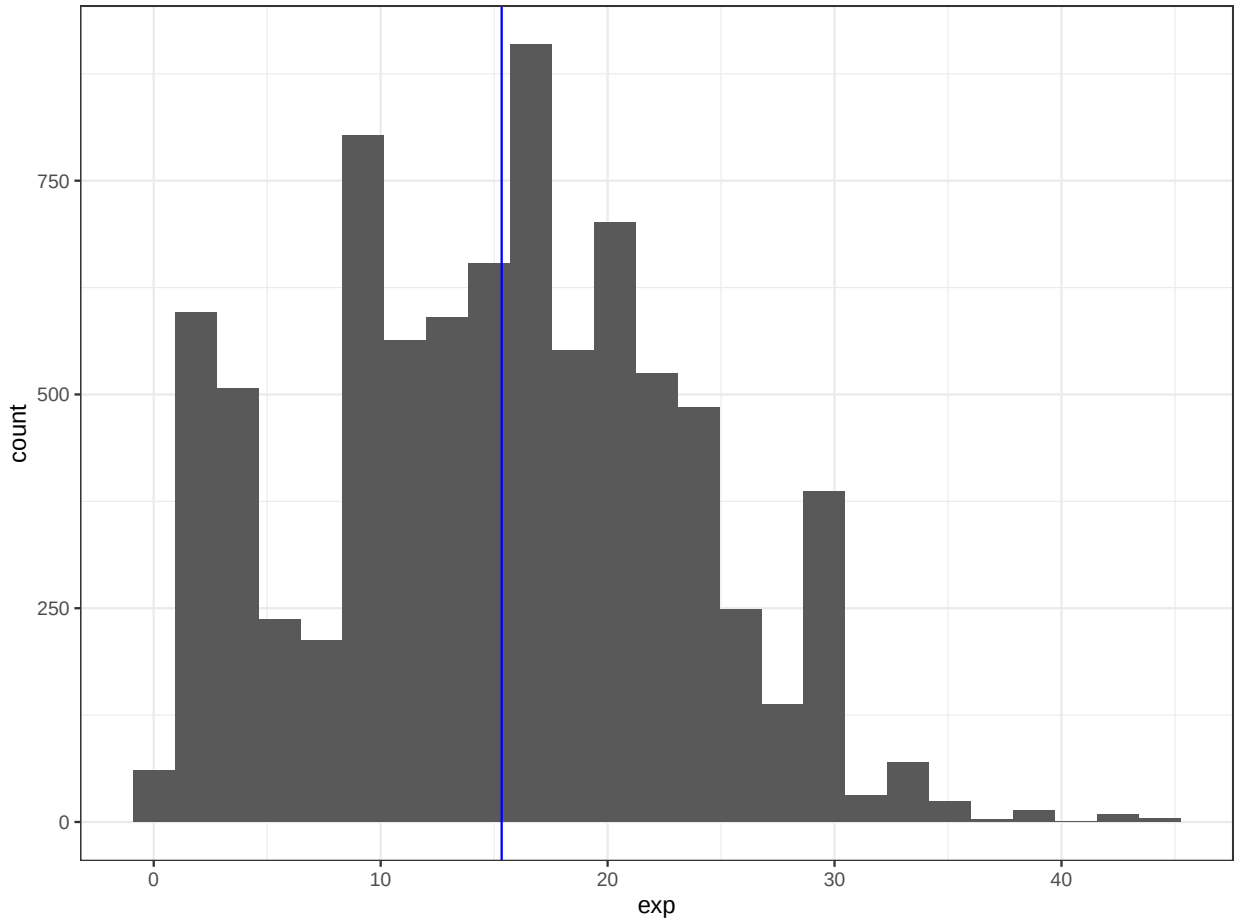

```
ggplot(officers, aes(x = spanish)) + geom_bar() + theme_bw()
```




```
ggplot(officers, aes(x = age)) +
    geom_histogram(bins = 25) +
    theme_bw() +
    geom_vline(aes(xintercept = median(age)),
               color = "blue")
```




```
ggplot(officers, aes(x = exp)) +
    geom_histogram(bins = 25) +
    theme_bw()+
    geom_vline(aes(xintercept = median(exp)),
           color = "blue")
```



Read in the officer stops edge list. Most officers have only 1 recorded stop. An important data cleaning step I took is that I am not considering any stops which happened while an officer was off-duty. As a result, some stops may appear as if only one officer made them, but it may be the case that it was really at least two officers who made the stop.

```
# The count function allows us to count the number of times the same officer
# pair appears (for each officer pair).
stops_edgelist <-
    read_csv(here("create-networks",
                  "output",
                  "stops-directed-edgelist.csv")) %>%
    count(officer_id.x, officer_id.y)
```

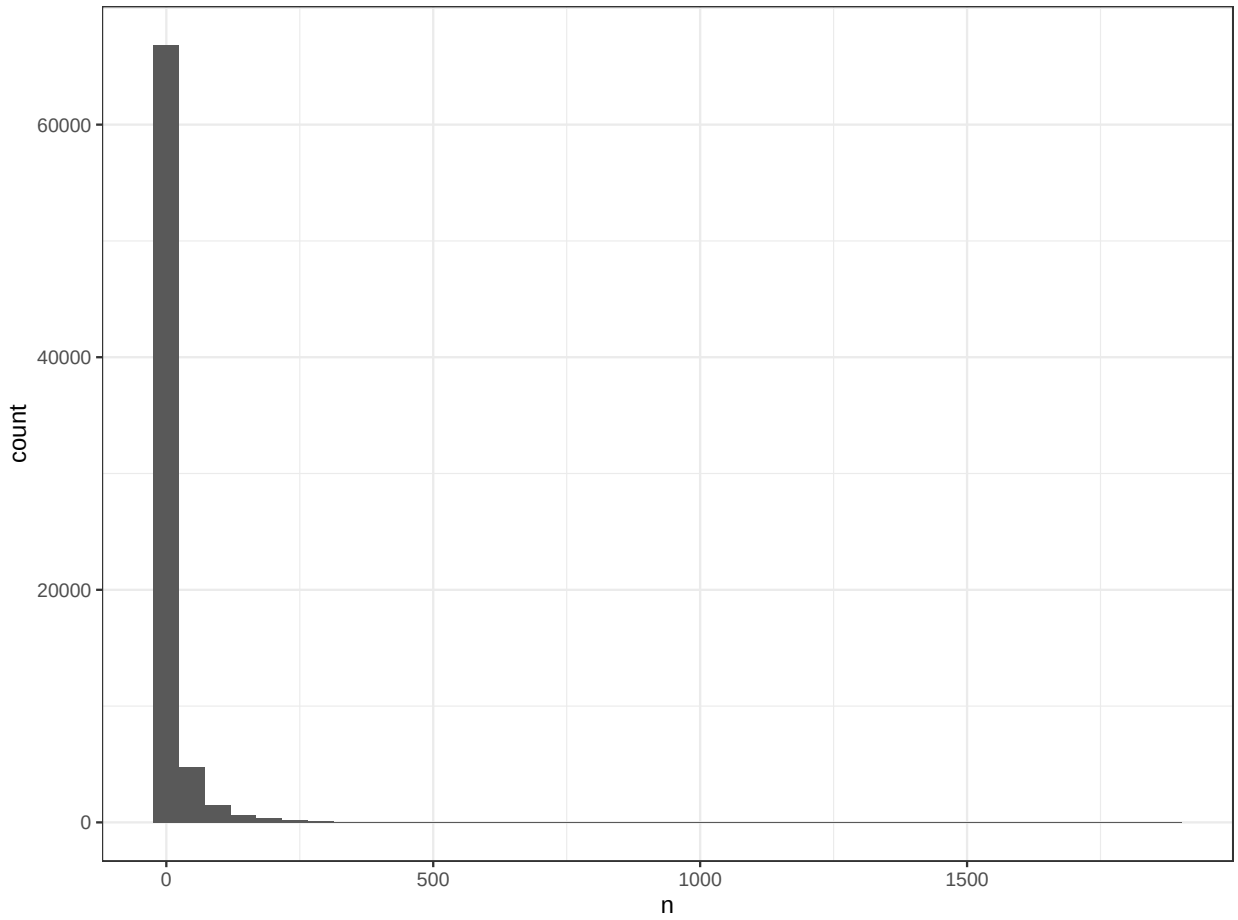
```
## Rows: 930241 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (3): officer_id.x, officer_id.y, stop_id
## lgl (1): po_first
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
summary(stops_edgelist$n)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##    1.00    1.00    2.00   12.47    6.00 1879.00
```

```
ggplot(stops_edgelist, aes(x = n)) + geom_histogram(bins = 40) + theme_bw()
```

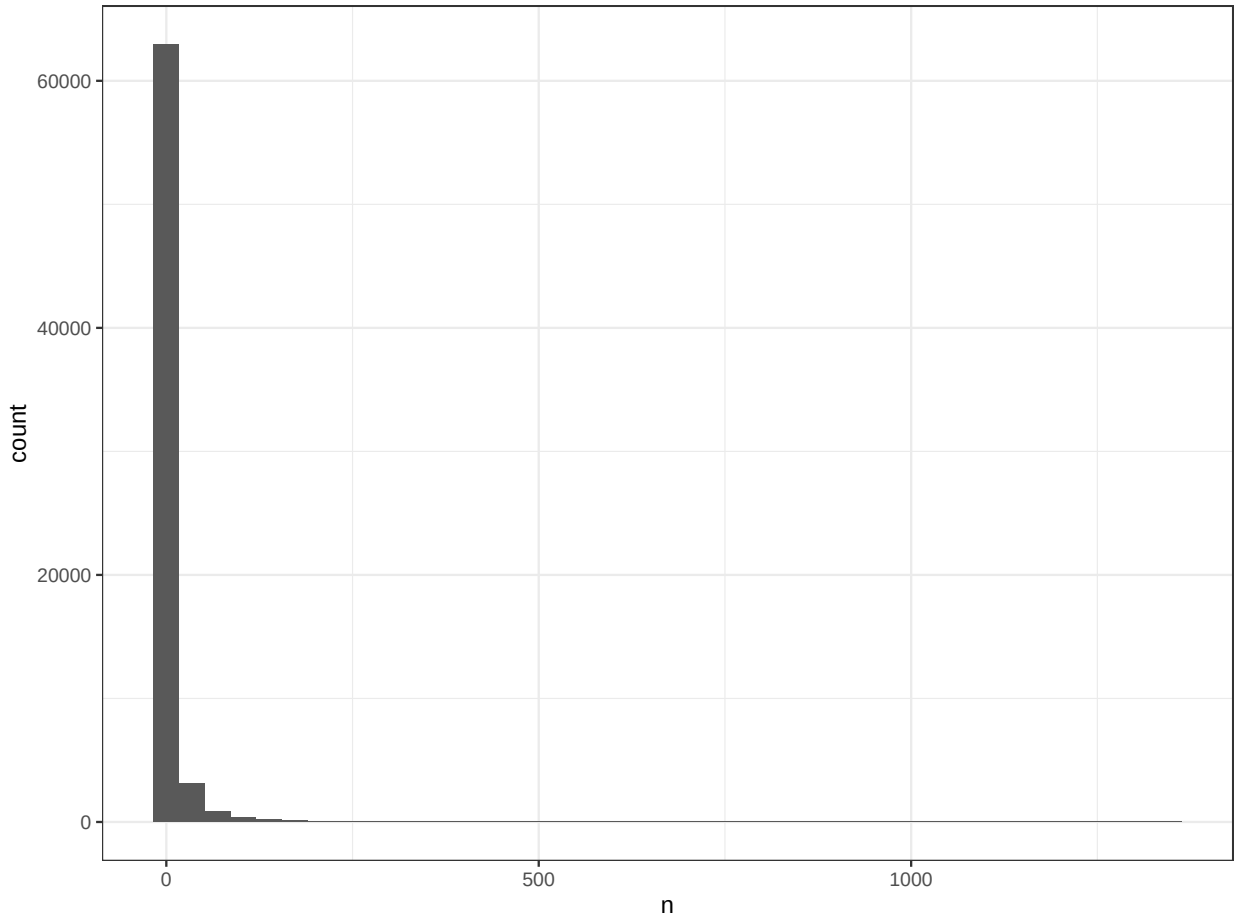


Getting rid of "loops" or instances where only 1 officer recorded a stop does not change the distribution all that much.

```
# Get rid of any loops or any officer pairs in which the same officer is listed
# twice.
stops_edgelist_noloops <-
    stops_edgelist %>%
    filter(officer_id.x != officer_id.y)

summary(stops_edgelist_noloops$n)
```

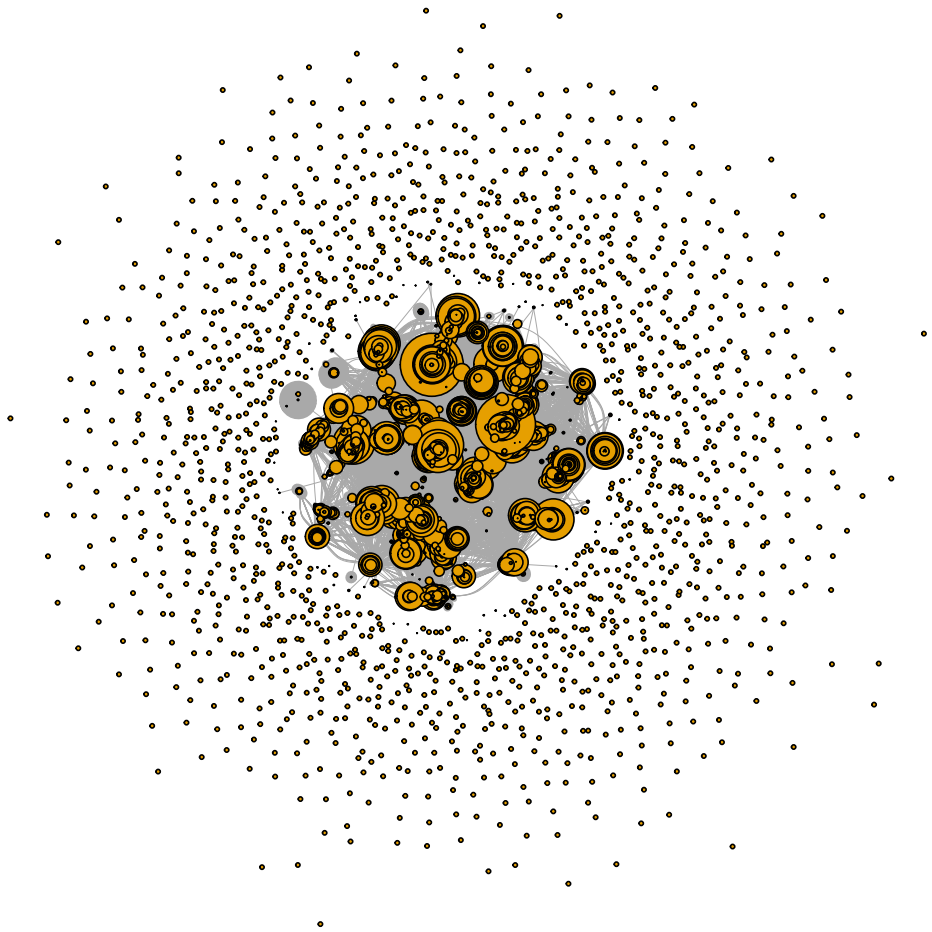
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    1.000    1.000    2.000    7.456    4.000 1347.000
```

```
ggplot(stops_edgelist_noloops, aes(x = n)) +
    geom_histogram(bins = 40) +
    theme_bw()
```



Now, let's look at the network. It is a big network! It has a very interesting structure.

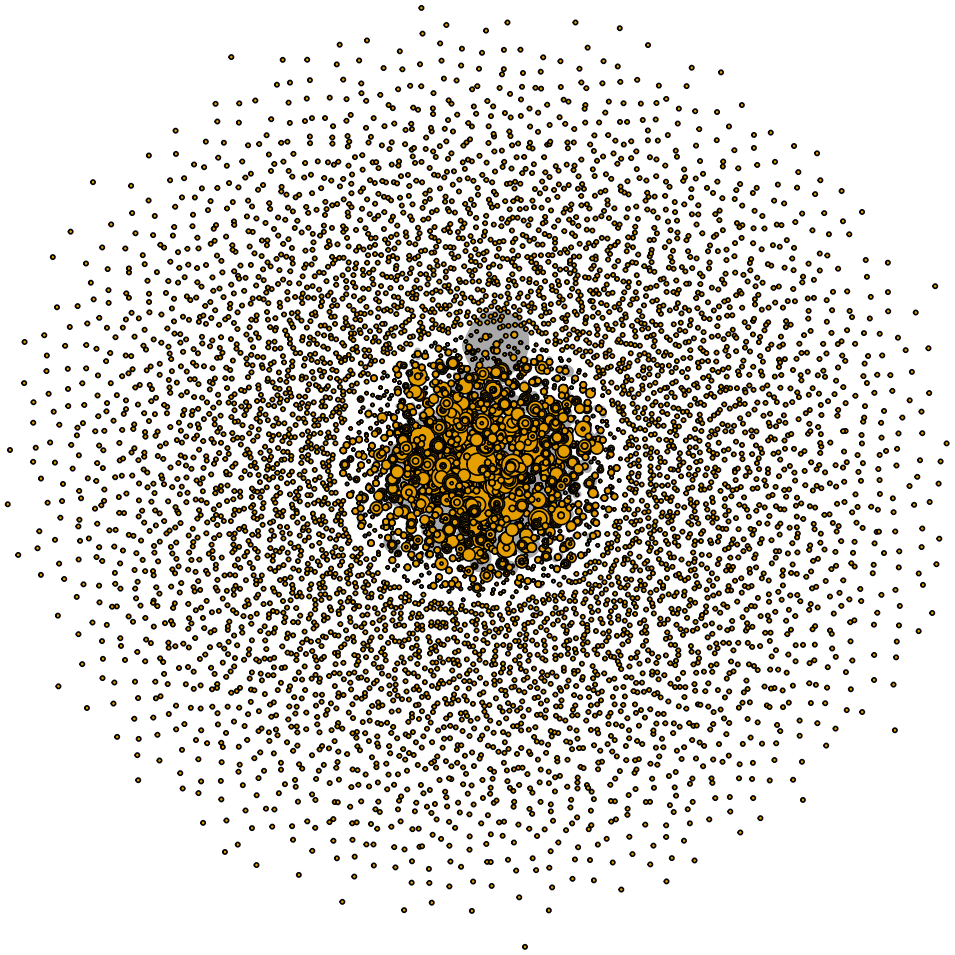
```
# Create the igraph object
stops_net <-
    graph_from_data_frame(d = stops_edgelist_noloops,
                          vertices = officers,
                          directed = F)

# Set the weight of edges
E(stops_net)$weight <- stops_edgelist_noloops$n

# Set the size of the nodes in the plot to be equal to their degree / 10
V(stops_net)$size <- degree(stops_net) / 10

# Set the width of edges in the plot to be equal to their weight / 50
E(stops_net)$width <- E(stops_net)$n / 50

# Get rid of arrow heads
E(stops_net)$arrow.size <- 0

plot(stops_net, vertex.label = NA)
```



Does the structure replicate when looking only at officer pairs in the top 95th percentile for most stops together?

```
# Keep only officer pairs in the top 5%
stops_top <-
```

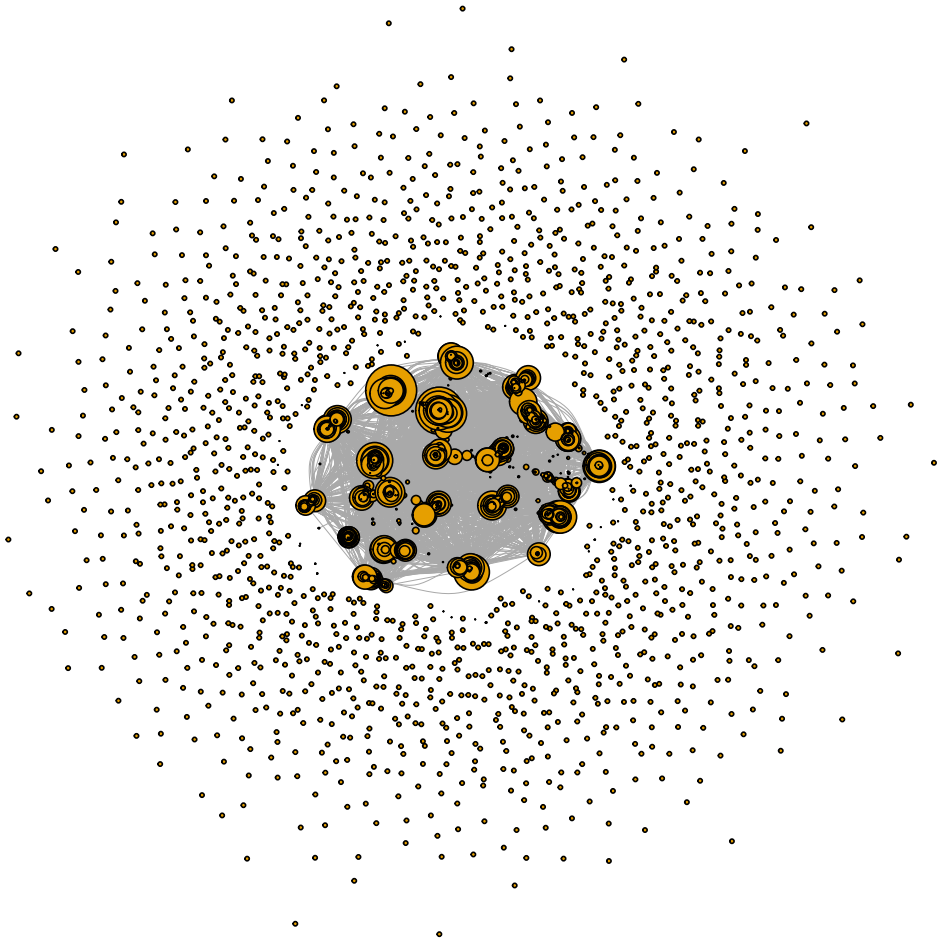
```
    stops_edgelist_noloops %>%
    filter(n > 27)

# Create the igraph object
stops_top_net <-
    graph_from_data_frame(d = stops_top, vertices = officers, directed = F)

# Set the weight of edges
E(stops_top_net)$weight <- stops_top$n

# Set the size of the nodes in the plot to be equal to their degree / 1.5
V(stops_top_net)$size <- degree(stops_top_net) / 1.5

# Set the width of edges in the plot to be equal to their weight / 50
E(stops_top_net)$width <- E(stops_top_net)$n / 30

# Get rid of arrow heads
E(stops_top_net)$arrow.size <- 0

plot(stops_top_net, vertex.label = NA)
```



Does the structure replicate when we looking only at officer pairs in the bottom 75th percentile for least amount of stops together?

```
# Keep only officer pairs in the bottom 75%
stops_bottom <-
```

```
    stops_edgelist_noloops %>%
    filter(n <= 4)

# Create the igraph object
stops_bottom_net <-
    graph_from_data_frame(d = stops_bottom, vertices = officers, directed = F)

# Set the weight of edges
E(stops_bottom_net)$weight <- stops_bottom$n

# Set the size of the nodes in the plot to be equal to their degree / 10
V(stops_bottom_net)$size <- degree(stops_bottom_net) / 10

# Set the width of edges in the plot to be equal to their weight / 5
E(stops_bottom_net)$width <- E(stops_bottom_net)$n / 5

# Get rid of arrow heads
E(stops_bottom_net)$arrow.size <- 0

plot(stops_bottom_net, vertex.label = NA)
```



Now we can go on to calculate assortativity or how likely it is for vertices of similar attributes to form an edge with each other. First is race. There is a moderate association where officers of the same race/ethnicity are more likely to make a stop together. It is much more pronounced for the 95th percentile group.

```
# an annoying feature of igraph when calculating the assortativity of
# categorical varialbes is that you have to conver them to numbers.
assortativity.nominal(stops_net,
                      as.integer(as.factor(V(stops_net)$officer_race)),
                      directed = F)
```

```
## [1] 0.2827152
```

```
assortativity.nominal(stops_top_net,
                      as.integer(as.factor(V(stops_top_net)$officer_race)),
                      directed = F)
```

```
## [1] 0.4107843
```

```
assortativity.nominal(stops_bottom_net,
                      as.integer(as.factor(V(stops_bottom_net)$officer_race)),
                      directed = F)
```

```
## [1] 0.2600014
```

Next we can calculate assortativity for officer sex. There is still a slight association where officers of the same sex are more likely to make a stop together, but the association is weaker than it was for race. However, the association is again stronger in the 95th percentile group.

```
assortativity.nominal(stops_net,
                      as.integer(as.factor(V(stops_net)$officer_sex)),
                      directed = F)
```

```
## [1] 0.1402298
```

```
assortativity.nominal(stops_top_net,
                      as.integer(as.factor(V(stops_top_net)$officer_sex)),
                      directed = F)
```

```
## [1] 0.3069452
```

```
assortativity.nominal(stops_bottom_net,
                      as.integer(as.factor(V(stops_bottom_net)$officer_sex)),
                      directed = F)
```

```
## [1] 0.1130603
```

There is a moderately strong association for officer age. It is likely that officers of the same age are going to be making stops together.

```
assortativity(stops_net, V(stops_net)$age, directed = F)
```

```
## [1] 0.4198769
```

```
assortativity(stops_top_net, V(stops_top_net)$age, directed = F)
```

```
## [1] 0.5216466
```

```
assortativity(stops_bottom_net, V(stops_bottom_net)$age, directed = F)
```

```
## [1] 0.3971142
```

There is a very strong association for officer experience.

```
assortativity(stops_net, V(stops_net)$exp, directed = F)
```

```
## [1] 0.5703216
```

```
assortativity(stops_top_net, V(stops_top_net)$exp, directed = F)
```

```
## [1] 0.6922482
```

```
assortativity(stops_bottom_net, V(stops_bottom_net)$exp, directed = F)
```

```
## [1] 0.5482494
```

We can also look at degree assortativity which is capturing the idea if officers who make lots of stops make stops together. We do see a slight association suggesting officers who make more stops are slightly more likely to be making those stops together.

```
assortativity_degree(stops_net, directed = F)
```

```
## [1] 0.2044267
```

```
assortativity_degree(stops_top_net, directed = F)
```

```
## [1] 0.3673275
```

```
assortativity_degree(stops_bottom_net, directed = F)
```

```
## [1] 0.1743104
```

All of the above measures of assortativity have not been taking into account the edge weights. What happens if take account the edge weights when calculating the degree? We find the relationship gets stronger. In particular in the 95th percentile group, we see officers who make the most stops are making those stops together which may be explained by the fact that those pairs of officers are the ones who are making the most stops to begin with.

```
assortativity(stops_net, graph.strength(stops_net), directed = F)
```

```
## [1] 0.2635851
```

```
assortativity(stops_top_net, graph.strength(stops_top_net), directed = F)
```

```
## [1] 0.6863283
```

```
assortativity(stops_bottom_net, graph.strength(stops_bottom_net), directed = F)
```

```
## [1] 0.1609175
```

For the curious, I create my own mixing matrix below and calculate my own assortativity coefficient to make it more clear what is happening. My measure matches what is provided in the igraph package up to 3 decimal places. Beyond that, I am not exactly sure why we do not get the same exact value.

```
# Find the officer race for each pair of officers
race_pairs <-
    stops_edgelist_noloops %>%
    inner_join(select(officers, officer_id, officer_race),
               by = c("officer_id.x" = "officer_id")) %>%
    inner_join(select(officers, officer_id, officer_race),
               by = c("officer_id.y" = "officer_id"))

# Create the mixing matrix
mm <- prop.table(table(race_pairs$officer_race.x, race_pairs$officer_race.y))

# Equation for calculating the assortativity coefficient.
mm
```

```
##
```

```
##               aapi_native       black        hisp       white
##   aapi_native 0.003689601 0.006464181 0.012957880 0.022137607
##   black       0.004944066 0.132087724 0.026712713 0.042489448
##   hisp        0.011452522 0.031036926 0.093273119 0.110614245
##   white       0.021665338 0.055034092 0.119041294 0.306399244
```

```
(sum(diag(mm)) - sum(colSums(mm) * rowSums(mm))) / (1 - sum(colSums(mm) * rowSums(mm)))
```

```
## [1] 0.2829372
```

Finally, I use the assortnet package to calculate the assortativity coefficient using the weights in the graph. igraph does not allow you to use weights in conjunction with other variables (as far as I am aware).

Notice when I use the weights how the assortativity increases. By not including weights, I was underestimating the assortativity.

```
# Have to turn the igraph network object into an adjacency matrix and use the
# attr argument to tell igraph to make the adjacency matrix weighted.
assortment.discrete(as_adjacency_matrix(stops_net, attr = "weight"),
                    as.integer(as.factor(V(stops_net)$officer_race)))
```

```
## $r
## [1] 0.368164
##
## $mixing_matrix
##             4          2          3           1        ai
## 4  0.37759871 0.02588365 0.10811990 0.020869829 0.5324721
## 2  0.02588365 0.11501515 0.02212180 0.005595770 0.1686164
## 3  0.10811990 0.02212180 0.11155516 0.013572762 0.2553696
## 1  0.02086983 0.00559577 0.01357276 0.003503542 0.0435419
## bi 0.53247210 0.16861638 0.25536962 0.043541903 1.0000000
```

```
# puzzling that these two commands do not give the same value
# I believe in theory they should, but I might be missing something.
assortment.discrete(as_adjacency_matrix(stops_net, attr = "weight"),
                    as.integer(as.factor(V(stops_net)$officer_race)),
                    weighted = F)
```

```
## $r
## [1] 0.2703784
##
## $mixing_matrix
##             4           2          3           1         ai
## 4  0.29864236 0.052324911 0.11508211 0.021556863 0.48760625
## 2  0.05232491 0.133610246 0.03025845 0.005942079 0.22213569
## 3  0.11508211 0.030258452 0.09026576 0.011893773 0.24750010
## 1  0.02155686 0.005942079 0.01189377 0.003365255 0.04275797
## bi 0.48760625 0.222135687 0.24750010 0.042757971 1.00000000
```

```
# matches igraph
assortment.discrete(as_adjacency_matrix(stops_net),
                    as.integer(as.factor(V(stops_net)$officer_race)))
```

```
## $r
## [1] 0.2827152
##
## $mixing_matrix
##             4           2          3           1        ai
```

```
## 4  0.30639924 0.048761770 0.11482777 0.021901473 0.4918903
## 2  0.04876177 0.132087724 0.02887482 0.005704123 0.2154284
## 3  0.11482777 0.028874819 0.09327312 0.012205201 0.2491809
## 1  0.02190147 0.005704123 0.01220520 0.003689601 0.0435004
## bi 0.49189026 0.215428436 0.24918091 0.043500398 1.0000000
```